International Journal of Innovative
Computing, Information and Control
Volume **13**, Number **6**, December **2017**

 ISSN 1349-4198
pp. 1981–1992

NEURAL NONLINEAR MODEL-BASED PREDICTIVE CONTROL WITH FAULT TOLERANCE CHARACTERISTICS APPLIED TO A ROBOT MANIPULATOR

Gabriel Hermann Negri[1], Mariana Santos Matos Cavalca[1]
and Luiz Antonio Celiberto Jr.[2]

[1]Center of Technological Sciences
Santa Catarina State University
Rua Paulo Malschitzki, 200, Bairro Zona Industrial Norte, Joinville, Santa Catarina 89.219-710, Brazil
negri.gabriel@gmail.com; mariana.cavalca@udesc.br

[2]Center of Engineering, Modeling and Applied Social Sciences
Federal University of ABC
Avenida dos Estados, 5001, Bairro Santa Terezinha, Santo André, São Paulo 09.210-580, Brazil
celibertojr@gmail.com

Received October 2016; revised February 2017

Abstract. *Fault-tolerant control systems have been researched with the objective of increasing reliability and safety in process control. Thus, knowledge on the effects of possible faults on the referred plant or process is necessary for designing a control system which maintains its functionality even with the occurrence of faults. In this report, the application of fault handling characteristics with a nonlinear model-based predictive control (NMPC) strategy applied to a robot manipulator model is presented. The utilized NMPC strategy employs a nonlinear prediction model, obtained by training a fully connected cascade artificial neural network (FCC-ANN). The optimization task is solved at each sampling period with the bound optimization by quadratic approximation (BOBYQA) method. Tests by means of numeric simulations were performed, with two different approaches of fault handling added to an NMPC controller. The first is through model adaptation and the second using a robust approach by worst case optimization. The results show that both methods improve the closed-loop response of the control system in comparison to NMPC without a specific fault handling strategy for a 50% actuator torque loss fault.*

Keywords: Faults, NMPC, Robot manipulator, Adaptive control, Robust control

1. **Introduction.** Practically every control system is subject to the occurrence of faults, such as sensor and actuator faults or even mechanical wearing of parts. Considering a controller designed to operate in nominal process conditions, faults might lead to performance loss or even instability. In many cases, the task to be performed and the safety of the operation can be compromised [1].

Fault detection and fault-tolerant control have been extensively researched for robot manipulators. Dixon et al. [2] treated fault detection for robot manipulators (RMs) subject to parametric uncertainty. In such work, the authors mention that RMs are often employed in hazardous or remote places, as in space, underwater and radioactive environments, and thus, fault tolerance is applied to minimizing the risks involved. Another fault-tolerant approach for RMs was presented by She et al. [3], for space applications. Since the RMs sent to space are designed to perform many different tasks and the cost of maintaining a robot in space is very high, fault tolerance is essential to ensure the robot effectiveness. Ma and Yang [4] state that fault tolerance in RMs is necessary for

avoiding the possibilities of damage in the hardware and injuries to people close to the robot. Additionally, a gas welding application with RM and fault detection can be found in [5].

The possible fault modes depend on the process characteristics. In some cases, logical approaches or methods based on discrete event systems theory [6] can be used to detect and identify the occurrence of a determined fault. Once a fault is detected, the controller is able to take decisions as restarting or locking the system or executing the task in an alternative way. At this point, it is important to differentiate events that cause total loss of functionality (failure) or relative performance loss (fault) [7]. Fault-tolerance is a characteristic of controllers which are able to deal with effects caused by system faults, maintaining stability and reliability or even reducing dynamic performance loss [8].

One of the main forms of fault treatment starts with the identification of expected patterns from a determined situation. In the area of multiphase motor control, as an example, attention is typically paid to the detection of an open phase condition, which creates characteristic variations in direct and quadrature axis currents [9]. Regarding control of robot manipulators, joint lock or mechanical wear, among others conditions, are observed. When it is possible to determine the characteristics created by the occurrence of a fault, it is said that such fault has a signature, which can be used to identify it [10].

Faults can be diagnosed through redundancy, using more than one source for the same information. Redundancy can be:

- physical, through the use of multiple sensors for a measured variable, or
- analytical, in which a representative process model is used to generate error signals, namely residuals, between the state variables of the real process and the process simulated with the model.

Physical redundancy seems, at first, more reliable and simple since it is not necessary to define and tune a representative model. However, aspects as sensor calibration, maintenance, cost, noise and increased volume with additional sensors and necessary hardware for interrogating them are drawbacks of physical redundancy [7]. Analytical, or model-based, redundancy, in its turn, is presented as a cheaper solution, which does not demand additional hardware, provided that the utilized microcontroller or processor is sufficient to implement the state estimation algorithms. Figure 1 shows, in a simplified form, the block diagram of an analytical fault identification and treatment system. The control signal applied to the process is simultaneously applied to a reference model. The error between the reference model and the real process output, namely residual, is applied to a residual analysis block, which identifies the type and cause of the fault. Finally, the controller or supervisory system is updated and adapted using the fault information. In the present work, an active fault treatment mechanism was used, which employed an algorithm for estimating the actuator fault magnitude so that the error between the reference

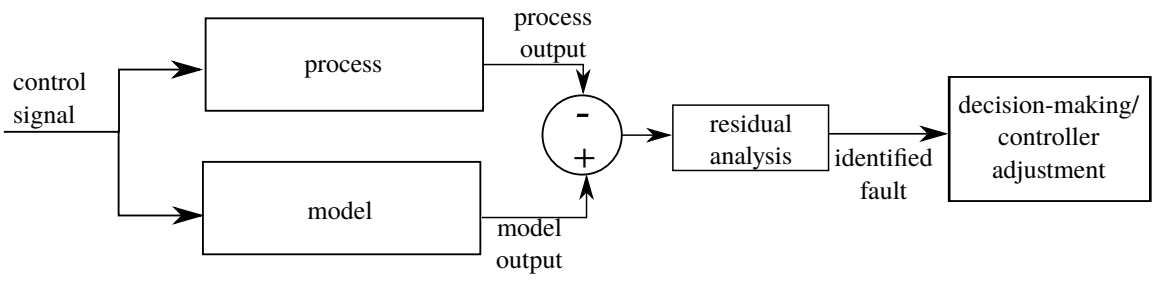

FIGURE 1. Model-based fault identification and treatment system based on [7]

model and the real process is minimized. Also, a robust approach with two reference models was used in order to minimize NMPC cost function in the worst case.

A difficulty found with the use of analytical redundancy in dynamic control systems is the occurrence of faults which do not necessarily cause abrupt changes in the process behavior. However, such type of fault may result in performance and safety problems if not treated, as already mentioned before [1, 11]. The gradual effect of such faults makes the fault identification process more difficult, since the process is also subject to modeling errors, noise and environmental conditions. Thus, for each process, it is necessary to perform an analysis of which method to use and how the residuals are going to be generated and classified.

With the diagnostic and identification system working, the decisions to design and adjust the controller must be made. Within fault-tolerant control context, two main approaches are known:

- robust control: the control algorithm is designed considering a set of possible or expected faults, using previous knowledge on the effect of such faults in the system; therefore, a controller capable of settling all the possible undesirable effects is sought to be designed, with the drawback of generally slower and more conservative dynamics;
- adaptive control: using information on the occurring fault, the controller is automatically adjusted or modified according to the new dynamic behavior of the controlled process.

Among model-based control methods, model-based predictive control (MPC) has the advantages of considering constraints when generating the control actions and, for being model-based, it can be directly adapted from information on new conditions of the process [17]. The main idea in MPC techniques is to minimize a cost function, seeking to optimize the control loop performance [12]. In the present work, two strategies based on nonlinear MPC (NMPC), to attenuate the effect caused by an actuator power loss fault in a two-degrees-of-freedom (2DOF) RM, were tested, being one based on adaptive control and the other on robust control. The utilized prediction model was a fully connected cascade (FCC) artificial neural network (ANN), trained as a one-step-ahead predictor and executed recurrently for N-steps-ahead predictions.

The contributions of this work are to present an NMPC which can be tuned to control an RM with the same performance in different operating regions. Also, the prediction model is an FCC neural network, which is efficient for mapping nonlinear characteristics with fewer synaptic connections than the traditional MLP networks. Regarding to fault tolerance, as NMPC is highly depending on the prediction model, this work presents two forms of dealing with an actuator fault without relying on model linearization, thus exploiting the robot nonlinear dynamics. The main contribution for fault tolerant NMPC is to provide an adaptive and robust control algorithm which is easy to implement and is suitable with the ANN model and the optimization method BOBYQA. Also, it is worth noticing that BOBYQA does not depend on the cost function derivatives, which makes it practical for dealing with nonlinear models such as the FCC.

This paper is organized as follows: in Section 2 the basic concepts on MPC, including an expansion for a nonlinear model (NMPC) are presented; in Section 3 a nonlinear 2DOF RM model and simulation results from the application of an NMPC controller with and without the occurrence of a 50% power loss in one of the two actuators, without using any fault-tolerant approaches are presented. The employed fault-tolerant approaches are presented in Section 4. Simulation results are presented in Section 5. Concluding remarks are discussed in Section 6.

2. **Model-Based Predictive Control (MPC).** MPC is the name given to a family of digital control algorithms which use an explicit mathematical model to represent the process dynamics and predict the process future behavior. Through mathematical analysis of the future behavior, an optimization algorithm is employed to find an optimum sequence of variations in the control actions. For measuring how good a control sequence is, a cost function is evaluated under a possible set of physical and operational constraints of the process. The cost function usually takes account of a compromise between control efforts (the energy employed in the control actions) and reference tracking errors [12, 13], as shown in Equation (1) [15]. Y is a vector with N predicted future output samples, obtained with the application of a vector ΔU containing M control action variations. N and M are named prediction and control horizons, respectively, and must be designed according to the available processing time at each sampling period and on modeling precision. $\bar{Q}$ is a weighthing matrix for future tracking errors between the predicted output vector and the future reference vector Y_{ref} while $\bar{\Gamma}$ is used to weight control efforts.

$$J_c[Y(\Delta U), \Delta U] = (Y - Y_{ref})^T \bar{Q}(Y - Y_{ref}) + \Delta U \bar{\Gamma} \Delta U \tag{1}$$

As shown in Equation (1), it is necessary to calculate Y as a function of ΔU, being ΔU generated iteratively by a search algorithm[1]. In the case of linear models, a state space model or even a sequence of step response sample can be used as prediction models [12]. A more detailed explanation of MPC techniques can be found in [21].

2.1. **Nonlinear FCC-ANN model.** While many successful applications of MPC with linear prediction models can be found in literature, processes with strong nonlinearities require different approaches in order to achieve high dynamic performance. In this work a nonlinear model was considered, with a trained ANN as a nonlinear prediction model [15]. More specifically, the ANN has an FCC topology, trained as a one step ahead predictor, using SuperNN library [16], and executed recurrently during cost function evaluations. The search for the optimal ΔU is performed through BOBYQA (bound optimization by quadratic approximation) optimization algorithm, using NLopt library [18].

The use of ANNs for modeling is interesting for NMPC, because training is performed over experimental acquired data, without depending on an analytical model. Thus, some typically difficult modeling effects, such as joint friction and actuator non-idealities can be identified [10]. Since errors between the model and the real process dynamics may exist, due to possible disturbances or model mismatching, a reference model, using an instance of the trained ANN, is executed in parallel to the real process, applying the same control actions. Then, at each sampling instant, the errors between the state variables of the real process, obtained by feedback information, and the reference model are taken and added to the predicted outputs in the cost function evaluation, enabling null tracking error to constant output references. Online training, state observers and disturbance estimators are alternatives for eliminating steady-state errors [14]. However, the method described earlier is shown to be sufficient for achieving null steady-state error without parameter tuning and with easy implementation.

For RMs, other advantages of using ANN models are that ANNs result in accurate models, without linearization or model simplification, which leads to smaller prediction errors. Also, the ANN model used in the present work does not use acceleration information, which can be problematic in practice [2].

The control algorithm, without considering faults, can be resumed in the following steps, at each sampling period:

[1]In the unconstrained linear model case, an analytical expression can be obtained to calculate the optimal ΔU.

- read process state variables;
- execute a simulation step in the reference model and obtain prediction errors;
- through the optimization algorithm, generate a vector ΔU, evaluate the resulting cost and, iteratively, search for the optimal ΔU;
- increment the control actions with elements of ΔU referred to the first following sampling period.

2.2. **Nonlinear cost function evaluation.** In the nonlinear approach utilized in this work, the cost function is evaluated through the simulated prediction of the process behavior. As a closed expression for prediction, such as used in state-space or step response models, is not practical with an ANN model, prediction is performed through the simulation of the ANN model for a given control action sequence generated by the optimization algorithm, using as initial conditions the process current state. Cost weights were set to the tracking errors, seeking reference tracking. To evaluate the cost function, for a given ΔU, the following steps are taken:

- load current process state variables to FCC-ANN model, and set a variable *cost* $\leftarrow$ $[\Delta\tau_1(k|k)\dots\Delta\tau_1(k+M-1|k)]^T\rho_1[\Delta\tau_1(k|k)\dots\Delta\tau_1(k+M-1|k)] + [\Delta\tau_2(k|k)\dots$ $\Delta\tau_2(k+M-1|k)]^T\rho_2[\Delta\tau_2(k|k)\dots\Delta\tau_2(k+M-1|k)]$;
- execute N simulation steps, varying the control actions according to ΔU in the first M steps;
- at each simulated step, increment *cost* as: $cost \leftarrow cost + \lambda_1[\theta_1(k+i|k) + e_{\theta_1}(k) - R_{\theta_1}(k+i)]^2 + \lambda_2[\theta_2(k+1|k) + e_{\theta_2}(k) - R_{\theta_2}(k+1)]^2 + \lambda_{\Delta 1}[\Delta\theta_1(k)]^2 + \lambda_{\Delta 2}[\Delta\theta_2(k)]^2$, where $R_{\theta_{1,2}}$ are the references for θ_1 and θ_2, $e_{\theta_{1,2}}(k)$ are the prediction errors, as explained before, $\Delta\theta_{1,2}(k+i|k) = \theta_{1,2}(k+i|k) - \theta_{1,2}(k+i-1|k)$, k is the current sampling period, i is the prediction step and the notation $(k+i|k)$ indicates a prediction for instant $k+i$ using process information obtained in k;
- return *cost*.

3. **2DOF Robot Manipulator.** The utilized model was taken from [10] and it is based on a 2DOF robot manipulator, as shown in Figure 2. In such robot, the control actions are the joint torques $\tau_1(t)$ and $\tau_2(t)$. The first link, with angle position $\theta_1(t)$ relative to the gravity force vector, has length l_1 and mass m_1, while the second link, with angle position $\theta_2(t)$ relative to link 1 axis, has length l_2 and mass m_1. In the following, the time dependency (t) is omitted for $\theta_1(t)$, $\theta_2(t)$, $\tau_1(t)$ and $\tau_2(t)$ for reading simplification.

The mathematical model is given through a set of four state variables: the two angles, θ_1 and θ_2 and their respective variation rates, $\dot{\theta_1}$ and $\dot{\theta_2}$, respectively, as shown in Equation

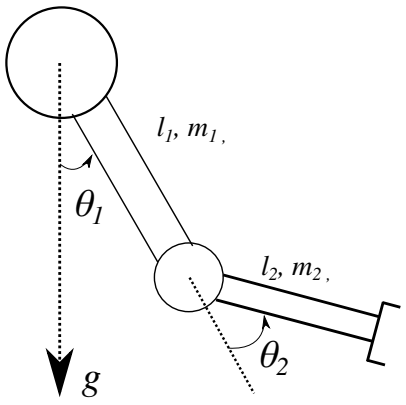

FIGURE 2. 2DOF RM, adapted from [10]

(2) [10].

$$\dot{x} = M^{-1}(T - F - V - G) \tag{2}$$

being:

$$x = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^T \tag{3}$$

$$M = \begin{bmatrix} m_2 l_2^2 + 2m_2 l_1 l_2 c_2 + l_1^2(m_{12}) & m_2 l_2^2 + m_2 l_1 l_2 c_2 \\ m_2 l_2^2 + m_2 l_1 l_2 c_2 & m_2 l_2^2 \end{bmatrix} \tag{4}$$

$$T = \begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}^T \tag{5}$$

$$V = \begin{bmatrix} -m_2 l_1 l_2 s_2 \dot{\theta_2}^2 - 2m_2 l_1 l_2 s_2 \dot{\theta_1}\dot{\theta_2} \\ m_2 l_1 l_2 s_2 \dot{\theta_2}^2 \end{bmatrix} \tag{6}$$

$$G = \begin{bmatrix} m_2 l_2 g s_{12} + (m_{12}) l_1 g s_1 \\ m_2 l_2 g s_{12} \end{bmatrix} \tag{7}$$

with $m_{12} = m_1 + m_2$, $c_2 = \cos(\theta_2)$, $s_2 = \sin(\theta_2)$, $s_1 = \sin(\theta_1)$ and $s_{12} = \sin(\theta_1 + \theta_2)$. The angles θ_1 and θ_2 can be obtained by the integration of $\dot{\theta}_1$ and $\dot{\theta}_2$, which are calculated using (2)-(7).

In addition to the terms presented in [10], vector F was defined as viscous friction and back-electromotive forces from the motors which drive the joints, given by:

$$F = \begin{bmatrix} \beta\dot{\theta}_1 & \beta\dot{\theta}_2 \end{bmatrix}^T \tag{8}$$

The utilized parameters are given in Table 1.

TABLE 1. Model parameters, adapted from [10]

Parameter	**Value**	**Unit**
m_1	5	kg
m_2	5	kg
l_1	0.3	m
l_2	0.3	m
β	5	Nms/rad
g	9.81	m/s^2

The model was simulated in open loop, by applying torque steps in both joints, seeking trajectories for θ_1 and θ_2 varying between $-60°$ to $60°$. With the obtained data, an FCC-ANN with 12 neurons in hidden layers, symmetric sigmoid activation function in the hidden layers and linear activation function in the output layer was trained to validate the methodology. Figure 3 shows a validation test, in which the capability of the FCC-ANN in approaching the robot dynamics can be observed, with very similar responses. Figure 4 shows the errors in θ_1 and θ_2 between the trained FCC-ANN and the simulated model in validation phase.

FCC-NMPC control was applied to the robot model with step reference profiles for θ_1 and θ_2 in ideal conditions and, in sequence, with the occurrence of a 50% power loss fault in the actuator connected to the first joint (τ_1). Such fault is not critical, since it does not prevent the control loop from working, but compromises the control performance if not considered by the controller. The parameters required to set the controller are shown in Table 2. The sampling period T_s was the same as used by [10]. The prediction and control horizons were set as high as possible in a manner that the optimization algorithm was able to converge in a time $T_s/3$. The parameters $\rho_{1,2}$ and $\lambda_{1,2,\Delta1,\Delta2}$ were set empirically

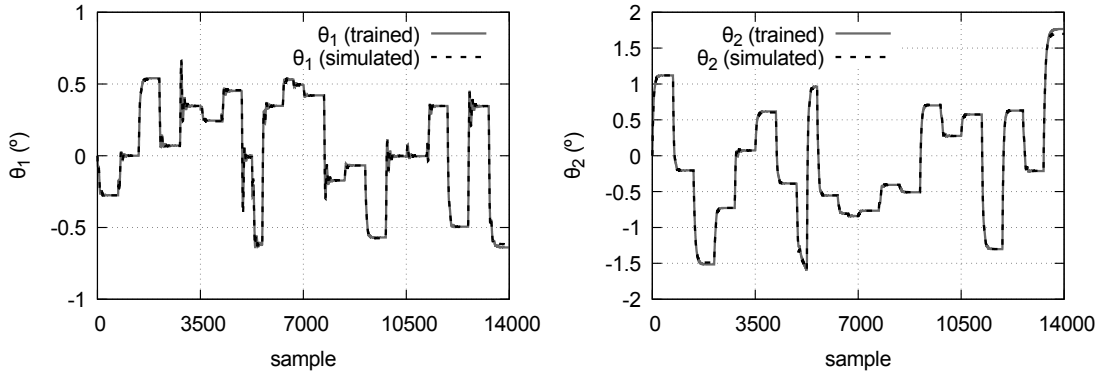

FIGURE 3. Validation of the trained FCC-ANN with the simulated model

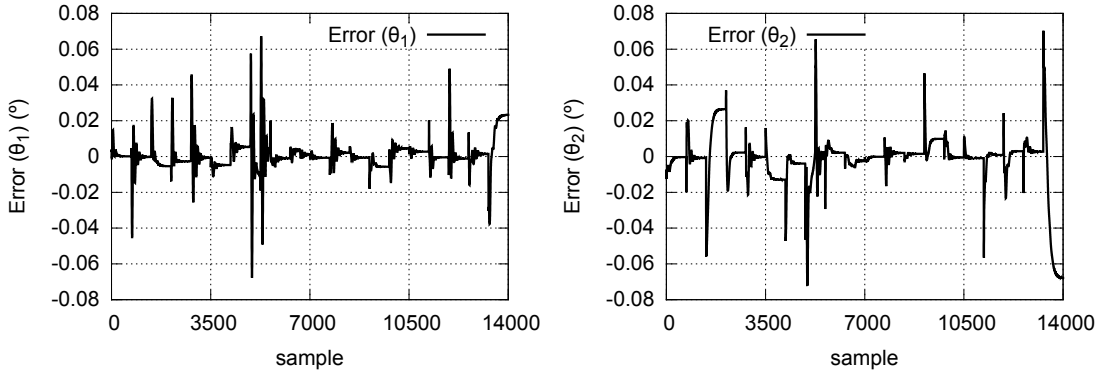

FIGURE 4. θ_1 and θ_2 validation errors between the trained FCC-ANN and the simulated model

TABLE 2. Controller parameters

Description	Parameter	Value
Prediction horizon	N	15
Control horizon	M	6
Sampling period	T_s	0.015 s
Control action τ_1 weight	ρ_1	0.01
Control action τ_2 weight	ρ_2	0.01
Output θ_1 tracking error weight	λ_1	0.07
Output θ_2 tracking error weight	λ_2	1.4
Output variation rate $\Delta\theta_1$ tracking error weight	$\lambda_{\Delta 1}$	900
Output variation rate $\Delta\theta_2$ tracking error weight	$\lambda_{\Delta 2}$	900

after a series of tests, with the objective of obtaining fast responses for θ_1 and θ_2 without overshooting.

Simulation results for θ_1 and θ_2 using the nominal FCC-NMPC controller with and without the fault occurrence, are shown in Figure 5. Such results are presented in order to demonstrate the effect of the fault on the control system performance, and also to show the expected performance without faults. The fault was applied at $t = 8$ s in the simulation. In such figures, it can be observed that both angles presented initial oscillations at the instant of fault occurrence and slower responses after, when compared to the case without the fault. Still, they did not reach the reference during the interval between 10 s and 20 s and presented considerable oscillations before the convergence to the

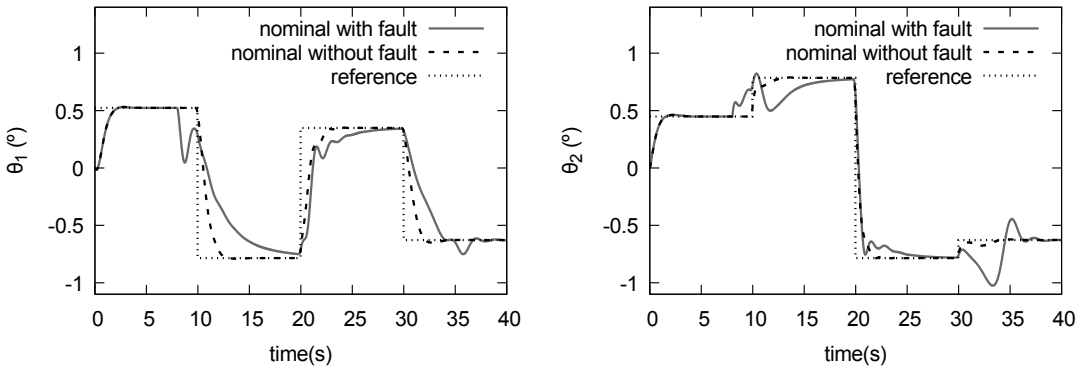

FIGURE 5. θ_1 and θ_2 responses with nominal FCC-NMPC with and without the fault

last reference step. In Section 5, the results obtained with the fault-tolerant techniques are presented along the results using the nominal FCC-NMPC controller for qualitative comparisons.

4. **Strategies for Reducing the Fault Effect.** Two strategies were applied seeking to minimize the fault effect in the control loop response. Firstly, a robustness-based approach using a worst case optimization is presented, followed by an adaptive approach. With the results presented in this section, it is possible to perform a qualitative comparison between an adaptive and a robust approach, while observing that both methods improved the system performance with simple implementation.

4.1. **Robust control approach.** The utilized robust control strategy is based on the min-max approach presented in [19] and presents a single controller, without online parameter adjustment. However, multiple cost functions are used, considering the extreme cases in variables subject to uncertainties. In this case study, a gain uncertainty on τ_1 actuator between 0.5 and 1 was considered. Thus, two models were utilized, each with the mentioned gain boundaries. At each sampling period, the optimization algorithm is executed twice, one for each case. Control increments obtained in the case which present the worst optimized cost are applied to the process. In this approach, online fault identification is not performed, and the controller is expected to handle the control task using offline information through the different cost functions on the effects of the considered fault. The contribution of this work regarding the robust approach is to present a robust control method for a generic nonlinear model by using instances of the NMPC nominal cost function while using the same optimization algorithm of the employed nominal NMPC.

4.2. **Adaptive control approach.** Adaptive control has the characteristic of varying the controller parameters using online identification of changes in the process behavior. One of the main identification methods is the least mean squares (LMS) algorithm [20]. LMS seeks a set of parameters which minimize the sum of the squared errors. Such idea was utilized to implement the adaptive approach.

As the model used within FCC-NMPC is based on a nonlinear ANN structure, parameter adaptation could be performed through online network training. However, online training has practical issues which make its implementation difficult. Thus, a heuristic search for the gain value applied to the model control actions which minimizes the summed squared errors between the model and the last samples from the real process is proposed. The contribution of presenting this approach is to provide a simple and efficient

least squares adaptive algorithm for an ANN model, which can also be used for generic nonlinear models, in the case of an actuator fault. The effectiveness of the method can be observed in Section 5.

Such strategy utilizes three parameters: n_s, n_c and Δg, and their application is explained as follows:

- Three gain variables are initialized with values $g_0 = 1$, $g_1 = 1 - \Delta g$ and $g_2 = 1 + \Delta g$, with Δg a search step size;
- At each sampling period, the last n_s control action applied to the process is applied to three FCC-ANN models, configured with the gains mentioned above;
- If the model with gain g_0 presents the smaller modeling error regarding the last n_s samples by n_c consecutive sampling instants ($n_c > 1$ is used to avoid noise issues), Δg is divided by 2 to refine the search;
- If the model with gain g_1 presents the smaller quadratic error, in the same conditions as mentioned in the earlier case, Δg is reinitialized to its original value, g_1 is attributed to g_0 ($g_0 = g_1$), $g_1 = g_0 - \Delta g$ and $g_2 = g_0 + \Delta g$.
- If the model with gain g_2 presents the smaller quadratic error, in the same conditions as mentioned in the earlier case, Δg is reinitialized to its original value, g_2 is attributed to g_0 ($g_0 = g_2$), $g_1 = g_0 - \Delta g$ and $g_2 = g_0 + \Delta g$.
- Gain g_0 is updated to the prediction model.

The greater n_s is set, the more precise the model evaluation will be. However, the computational burden grows and decision on varying the target variable becomes slower. The parameter n_c also regulates gain variation speed, and is set in a manner to avoid numerical noise issues due to modeling errors and, in real processes, sensor noise. Δg regulates the search variation step. The smaller the step is, the greater the precision of the model is and the slower the convergence is. In the performed simulation tests, the parameters were set as $n_s = 20$, $n_c = 5$ and $\Delta g = 0.02$. Such values were obtained empirically. A statistical analysis for the setting of such parameters is necessary as future work.

5. **Simulation Results.**

In this section, simulation results obtained with both strategies, robust and adaptive, are presented. The following graphs show the responses resulting from the application of the implemented controllers, which were designed to minimize the fault effect, along with the responses from the nominal controller, for qualitative comparison.

5.1. **Simulation with robust approach.**

Figure 6 shows θ_1 and θ_2 responses obtained with the robust control approach presented earlier. It can be observed in θ_1 that at start, when the fault had not occurred yet, nominal control resulted in a response with smaller settling time, when compared to the robust control, and no overshoot. However, with the fault applied at $t = 8$ s, robust control resulted in a faster response in the two following reference steps and reduction in the oscillation at the last step. In θ_2 response, it can also be observed a performance loss at start and, from the occurrence of the fault, a superior performance, with faster responses and reduced oscillations, than with nominal control. This effect is due to the use of a perfectly matched model in the nominal controller when there are no faults, while the robust controller considers the possibility of a fault. When the fault occurs, the robust controller presents a better performance, since the nominal controller model does not consider the change in the plant behavior.

5.2. **Simulation with adaptive control.**

In θ_1 and θ_2 responses, shown in Figure 7, it can be verified that the adaptive strategy resulted in practically the same responses obtained with nominal control before the fault. At the fault instant, at $t = 8$ s, the adaptive

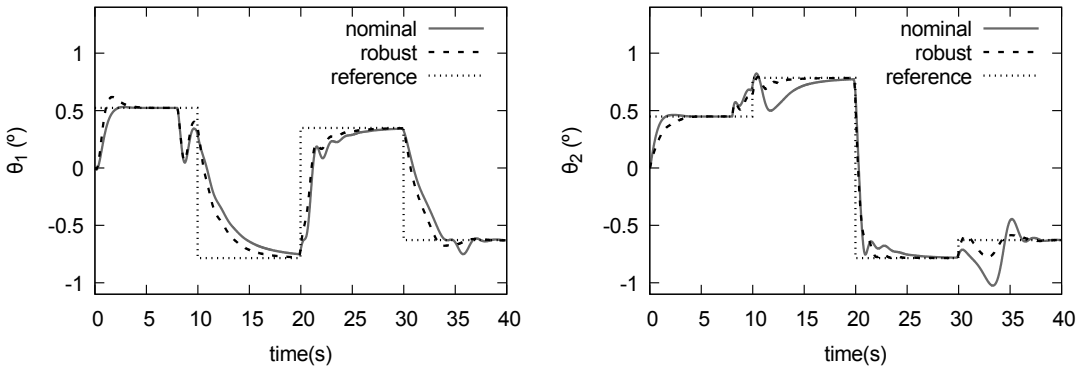

FIGURE 6. θ_1 and θ_2 responses with robust approach

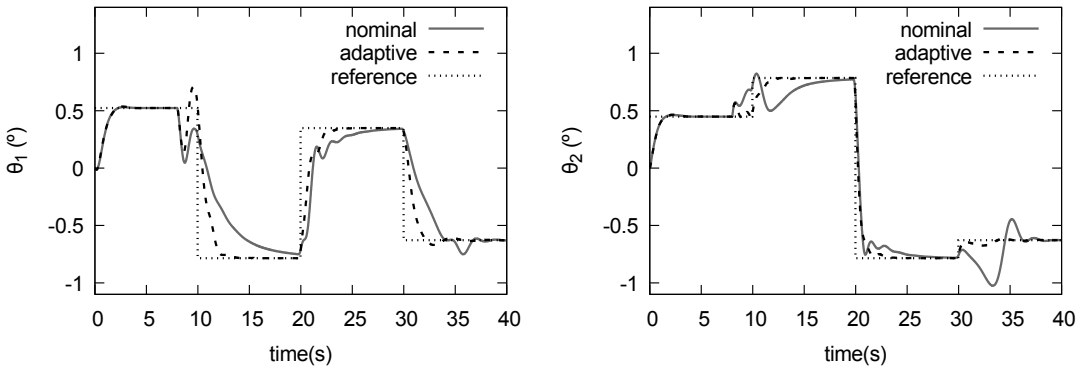

FIGURE 7. θ_1 and θ_2 responses with adaptive approach

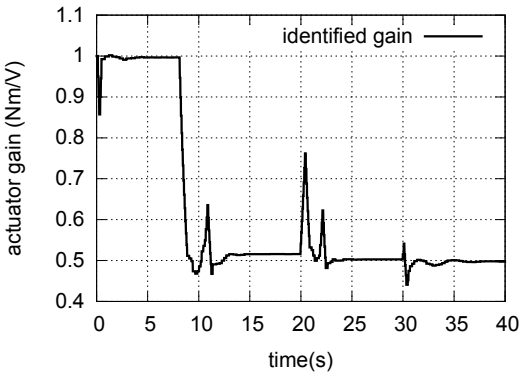

FIGURE 8. Identified gain

controller was able to react, taking θ_2 close to the reference before the application of the second reference step. From the second step, in both angles θ_1 and θ_2, performance was improved through adaptation of gain, enabling faster and less oscillatory responses than with the nominal controller. Such improvement is due to the gain online identification, which attenuates the influence of the fault. It can be observed that the responses from 10 s to 40 s of simulation are similar to the initial transient, which means that the adaptive controller was able to maintain its performance in different operating regions while adjusting the gain parameter.

The gain identified through simulation time is shown in Figure 8. It is possible to observe that the identified gain remains around 1 at start, after an initial oscillation,

and converges to a value of approximately 0.5, except during reference changes, in which the FCC-ANN model mismatch had influence. The identification of a considerable gain variation (from 1 to 0.5) could be used to trigger a fault detection event, in case the variation exceeded a threshold by a determined number of consecutive samples. If a slower convergence is interesting to avoid abrupt variations, the parameters mentioned at Subsection 4.2 can be reconfigured.

5.3. **Performance comparison.** The tracking performance of the tested controllers for θ_1 and θ_2 is presented in Table 3, by means of the sum of the squared error of all samples in four intervals, denoted by $\chi[\theta_{1,2}(\text{a,b,c,d})]$. Each interval corresponds to a change in the reference. Interval (a) is taken from 0 to 10 s – Nt_s, interval (b) from 10 s – $(N-1)t_s$ to 20 s – Nt_s, interval (c) from 20 s – $(N-1)t_s$ to 30 s – Nt_s and interval (d) from 30 s – $(N-1)t_s$ to 40 s, so that the transient responses including the NMPC anticipative effect were considered.

TABLE 3. Performance comparison between nominal, robust and adaptive NMPC for θ_1 and θ_2 reference tracking

Controller	$\chi[\theta_1(\text{a})]$	$\chi[\theta_2(\text{a})]$	$\chi[\theta_1(\text{b})]$	$\chi[\theta_2(\text{b})]$	$\chi[\theta_1(\text{c})]$	$\chi[\theta_2(\text{c})]$	$\chi[\theta_1(\text{d})]$	$\chi[\theta_2(\text{d})]$
Nominal	24.8	7.3	99.3	11.3	65.1	36.8	64.7	21.0
Robust	20.2	9.1	74.0	1.1	45.8	29.0	47.5	2.3
Adaptive	18.8	4.9	52.8	2.3	36.5	27.8	30.3	0.4

In interval (a), the robust controller presented a smaller θ_1 error than the nominal controller and a higher error in θ_2. Thus, it is not possible to state which of the two controllers presented a better performance, since the fault occured during this interval. In the intervals (b), (c) and (d), with the effect of the fault, the robust controller presented smaller tracking errors than the nominal controllers. The more significant error reductions can be observed in $\chi[\theta_1(\text{b})]$ and $\chi[\theta_1(\text{d})]$.

As for the adaptive controller, it can be noticed that it presented smaller errors for θ_1 and θ_2 than the nominal controller in all four intervals. Compared to the robust controller, the adaptive strategy resulted in a greater error only in $\chi[\theta_2(\text{b})]$. However, it presented a smaller $\chi[\theta_1(\text{b})]$. These results indicate that both strategies were effective for reducing the fault influence on trajectory tracking.

6. **Concluding Remarks.** This report is the result of the study of fault handling techniques for the development of fault-tolerant NMPC controllers. With both evaluated strategies, it was verified that there are performance improvements in comparison to the use of a nominal model in both cases, under the influence of a fault.

The robust strategy, although having a simpler implementation, has a higher computational cost than the adaptive strategy, for executing the optimization algorithm twice per sampling period. It was observed that, while the process was at nominal operation, such strategy resulted in a performance loss in relation to the nominal controller, but enabled faster and more stable responses after the occurrence of the fault.

In the tests performed with the adaptive strategy, a greater performance improvement was verified, compared to the robust controller, with the disadvantage of having more parameters to tune, which need to be adjusted according to the process characteristics and requirements of dynamics.

As future works, the implementation of the presented techniques in cases with possibilities of multiple simultaneous faults is suggested. Additionally, practical tests can be performed for evaluating noise influence and modeling difficulties. Another suggestion

for future works is the study of critical faults, which may require abrupt changes in the behavior of the control loop, such as joint lock in redundant robots.

Acknowledgment. The present publication was financed by CAPES – Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil, by means of PROAP (Graduate Support Program) process 4150/2017.

The authors also thank Santa Catarina State University (UDESC) which, by means of the Graduate Monitoring Scholarship Program (PROMOP), grants financial support to G. H. Negri, and Tutorial Learning Program (PET) from Ministry of Education of Brazil.

REFERENCES

[1] M. Staroswiecki, Fault tolerant systems, *Control Systems, Robotics and Automation*, vol.XVI, pp.215-254, 2009.

[2] W. E. Dixon, I. D. Walker, D. M. Dawson and J. P. Hartranft, Fault detection for robot manipulators with parametric uncertainty: A prediction-error-based approach, *IEEE Trans. Robotics and Automation*, vol.16, no.6, pp.689-699, 2000.

[3] Y. She, W. Xu, H. Su, B. Liang and H. Shi, Fault-tolerant analysis and control of SSRMS-type manipulators with single-joint failure, *Acta Astronautica*, vol.120, pp.270-286, 2016.

[4] H.-J. Ma and G.-H. Yang, Simultaneous fault diagnosis for robot manipulators with actuator and sensor faults, *Information Sciences*, vol.366, pp.12-30, 2016.

[5] I. Erski, S. Erkaya, S. Savas and S. Yildirim, Fault detection on robot manipulators using artificial neural networks, *Robotics and Computer-Integrated Manufacturing*, vol.27, no.1, pp.115-123, 2011.

[6] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohidenn and D. C. Teneketzis, Failure diagnosis using discrete-event models, *IEEE Trans. Control Systems Technology*, vol.4, no.2, 1996.

[7] S. Simani, C. Fantuzzi and R. J. Patton, *Model-Based Fault Diagnosis in Dynamic Systems Using Identification Techniques*, Springer-Verlag, 2002.

[8] R. Qi, L. Zhu and B. Jiang, Fault-tolerant reconfigurable control for MIMO systems using online fuzzy identification, *International Journal of Innovative Computing, Information and Control*, vol.9, no.10, pp.3915-3928, 2013.

[9] A. Khlaief, M. Boussak and M. Gossa, Open phase faults detection in PMSM drives based on current signature analysis, *XIX International Conference on Electrical Machines (ICEM)*, Rome, 2010.

[10] R. Tinós, *Deteccção e Diagnóstico de Falhas em Robôs Manipuladores via Redes Neurais Aritificiais*, Master Thesis, USP - Escola de Engenharia de São Carlos, 1999.

[11] H. Alwi, C. Edwards and C. P. Tan, Fault tolerant control and fault detection and isolation, *Advances in Industrial Control: Fault Detection and Fault-Tolerant Control Using Sliding Modes*, Springer, 2011.

[12] S. J. Qin and T. A. Badgwell, A survey of industrial model predictive control technology, *Control Engineering Practice*, vol.11, pp.733-764, 2003.

[13] A. Bemporad, Model predictive control design: New trends and tools, *Proc. of the 45th IEEE Conference on Decision and Control*, San Diego, 2006.

[14] R. Hedjar, Adaptive neural network model predictive control, *International Journal of Innovative Computing, Information and Control*, vol.9, no.3, pp.1245-1257, 2013.

[15] G. H. Negri, *Avaliação de Malhas de Controle Preditivo Baseado em Modelo Não Linear Usando Otimização Livre de Derivadas*, Master Thesis, UDESC – Centro de Ciências Tecnológicas, 2016.

[16] L. H. Negri, *SuperNN*, https://bitbucket.org/lucashnegri/supernn, 2016.

[17] G. Valencia-Palomo, C. M. Astorga-Zaragoza, F. R. Lopez-Estrada, M. Adam-Medina and J. Reyes-Reyes, Discrete-time constrained predictive control for distillation columns, *International Journal of Innovative Computing, Information and Control*, vol.8, no.6, pp.3939-3952, 2012.

[18] S. G. Johnson, *NLopt*, http://ab-initio.mit.edu/wiki/index.php/NLopt, 2015.

[19] M. V. Kothare, V. Balakrishnan and M. Morari, Robust constrained model predictive control using linear matrix inequalities, *Automatica*, vol.32, no.10, 1996.

[20] K. J. Åström, Theory and applications of adaptive control – A survey, *Automatica*, vol.19, pp.471-486, 1983.

[21] E. B. Cavalca, J. de Oliveira, M. S. M. Cavalca and A. Nied, Application of model-based predictive control approaches in a three-phase induction motor, *The 22nd International Congress of Mechanical Engineering (COBEM 2013)*, Ribeirão Preto, SP, Brazil, 2013.