Journal of Control, Automation and Electrical Systems (2021) 32:1228–1243
https://doi.org/10.1007/s40313-021-00755-4

Constrained Nonlinear Predictive Control Using Neural Networks and Teaching–Learning-Based Optimization

Mohamed Benrabah[1] · **Kamel Kara**[1] · **Oussama AitSahed**[1] · **Mohamed Laid Hadjili**[2]

Received: 30 April 2020 / Revised: 29 October 2020 / Accepted: 10 June 2021 / Published online: 21 June 2021

Abstract
In this work, an efficient constrained nonlinear predictive control algorithm, based on the meta-heuristic optimization strategy, is proposed. The aim is to develop a predictive control algorithm having a simple design and implementation procedure on the one hand, and that can give satisfactory performance with a large class of nonlinear systems on the other hand. To achieve this goal, a feedforward multilayer neural network is used to predict the future outputs of the system, and the optimization problem of predictive control is resolved using teaching–learning-based optimization strategy. Due to their interesting proprieties, feedforward neural networks are widely used in nonlinear systems identification and control. Indeed, they have a simple structure and can accurately approximate any nonlinear mapping. The teaching–learning-based optimization is used in a large number of applications in different fields of engineering and has gained wide acceptance among the optimization researchers community. Unlike other metaheuristic algorithms, this algorithm requires only common controlling parameters like population size and number of generations for its working; it does not require any algorithm-specific parameters. To assess the effectiveness of the proposed control algorithm, the control of the model of the continuous stirred tank rector and the 2-DOF manipulator robot model is considered. A comparative study, using the conventional PID controller, the fuzzy logic control, the computed torque control and the neural network-based model predictive control using particle swarm optimization, is carried out. To further demonstrate the effectiveness of the proposed controller, the control algorithm is numerically implemented and applied to a system with fast dynamics, namely the induction motor. The obtained results, through the simulation and the experimental study, indicate that the proposed controller presents better control performance than the other controllers. Furthermore, the experimental study shows that the developed control strategy can be effectively used to control, in real time, systems with fast dynamics.

Keywords Predictive control · TLBO · Neural networks · Nonlinear intelligent control · Meta-heuristic

1 Introduction

Model-based predictive control (MPC) is a modern and successful control strategy that is characterized by its ability to control constrained multivariable systems. To obtain the optimal control sequence, it uses an explicit model to predict the system future behaviour over a given horizon and then minimizes a given constrained cost function. Usually, this cost function is a quadratic function of the error between the predicted response and the reference trajectory and includes in most cases the control energy (Camacho and Bordons 2007; Maciejowski 2002; Katz et al. 2018).

Due to its industrial success, considering a wide range of complex systems, a lot of attention was given to this control strategy and several control algorithms, based on linear models, were developed. Richalet et al. (1978) have proposed

✉ Mohamed Benrabah
pr.moh1@yahoo.fr

Kamel Kara
km_kara@yahoo.fr

Oussama AitSahed
aitsahed.oussama@hotmail.com

Mohamed Laid Hadjili
mhadjili@he2b.be

[1] Laboratoire des systèmes électriques et télécommande, Faculté de Technologie, Université Blida 1, B.P 270, Route de Soumaa, Blida, Algeria

[2] Haute Ecole Bruxelles-Ecole Supérieure d’Informatique, 67 Rue Royale, Brussels, Belgium

the model algorithmic control (MAC) technique. Two years later, a step response model-based predictive control, called dynamic matrix control (DMC), has been developed by Cutler and Ramaker (1980). Ydstie (1984) proposed the extended horizon adaptive control (EHAC). Taking into account the disturbances effect, the EHAC algorithm has been improved by De Keyser and Van Cauwenberghe (1985). After that, the famous generalized predictive control (GPC) was introduced (Clarke et al. 1987a, b). Since then, several techniques and improvement on the developed algorithms have been proposed (Kansha and Chiu 2009; Li and Wang 2017; Li et al. 2017). The success of the MPC strategy is related to its capability of handling different types of constraints and its ability to handle multivariables systems, systems with no-minimum phase behaviour and systems with variable or unknown time delays.

Although linear MPC techniques give satisfactory performance in many practical applications (Lucia et al. 2014; Han and Qiao 2014; Subramanian et al. 2018), in the case of highly nonlinear process, severe degradation in control performance can be observed. To ensure higher performance, MPC methods that use a nonlinear prediction model must be investigated. In fact, a lot of attention was given to nonlinear MPC (NMPC), and several control techniques were proposed (Mazinan 2012; Lu et al. 2015; Thangavel et al. 2018; Abdennour et al. 2002). The main difficulties in designing NMPC are obtaining an adequate nonlinear model for the system to be controlled, and online solving the nonlinear and the non-convex optimization problem. Obviously, the efficiency and computational requirement of the controller depend extremely on the accuracy and simplicity of the used model. In fact, there is no clearly suitable modelling approach to represent general nonlinear systems. Among the various nonlinear models developed and used in predictive control, we can find: Volterra series (Maner et al. 1994; Gruber et al. 2013; Diaz-Mendoza and Budman 2009), fuzzy models (Lu et al. 2015; Ait Sahed et al. 2016; Huang et al. 2000), neuronal models (Mazinan 2012; Patan 2018; Han et al. 2013), etc. Due to their flexibility, simplicity and good performances, artificial neural networks (ANN) were extensively used in modelling (Mohammadi et al. 2018; Mahmoudi et al. 2016; Ghorbani et al. 2017) and controlling nonlinear systems (Vargas et al. 2019; Qin et al. 2020; Yuan et al. 2019; Kumpati et al. 1997; Narendra and Parthasarathy 1989). Moreover, it has been shown that neural networks can approximate any given function with arbitrary precision (Kumpati et al. 1997).

Solving the non-convex optimization problem of nonlinear constrained predictive control is difficult and time-consuming task. Since an analytical solution cannot be obtained, several suboptimal methods were used to deal with this optimization problem (Botto et al. 1999; Liu et al. 1998; Soloway and Haley 1996; Sorensen et al. 1999). To obtain a convex quadratic optimization problem, some of the suboptimal methods use prediction models of particular structure (Botto et al. 1999; Liu and Kadirkamanathan 1998). Some others are based on numerical optimization algorithms, which are known to have a slow convergence rate and could easily be trapped in local minima (Soloway and Haley 1996; Sorensen et al. 1999). An optimization strategy that could mitigate the non-convex and nonlinear optimization problem of predictive control is based on the meta-heuristic algorithms. These algorithms, such as genetic algorithms (GA) (Sarimveis and Bafas 2003; Li et al. 2012), particle swarm optimization (PSO) (Mazinan 2012; Coelho et al. 2005; Zhixiang et al. 2008), artificial bee colony (ABC) (Ait Sahed et al. 2015,2016) and evolutionary algorithm (EA) (Zimmer et al. 2015), are easy to implement, have good performance and could locate adequate solutions in a reasonable time. In fact, a lot of researches have been conducted on solving the NMPC optimization problem using meta-heuristic algorithms (Ait Sahed et al. 2016; Jiang et al. 2012; Wang et al. 2019).

A relatively recent efficient optimization method, called, teaching–learning-based optimization (TLBO), was designed for solving complex, constrained and nonlinear optimization problems (Rao et al. 2011,2012). It is based on classroom teaching–learning process in the form of two phases, a teacher phase, where students learn from their teacher, and a learners phase, where the students learn by interacting between themselves. Furthermore, this algorithm was found to be more efficient than several other meta-heuristic algorithms such as ABC, multimembered evolutionary strategy (M-ES), particle evolutionary swarm optimization (PESO), cultural differential evolution (CDE), co-evolutionary differential evolution (CoDE) and genetic algorithm (Rao et al. 2011; Rao and Waghmare 2014; Patel and Savsani 2016). Another interesting advantage of using the TLBO algorithm is the fact that, except the common control parameters (population size and number of generations), and unlike to other meta-heuristic algorithms, this algorithm does not require any algorithm-specific parameters (Rao 2016). A bad choice of any value of the specific parameters of the meta-heuristic algorithms leads to a slow convergence rate and could easily be trapped in local minima. Therefore, before implementing a meta-heuristic algorithm which depends on some specific parameters, an optimization step of these parameters is necessary. However, this is not required in the case of the TLBO algorithm.

In this work, a TLBO-based neural network model predictive controller (NNMPC-TLBO) is proposed. The aim is to develop a simple and an efficient controller that can be used to control any nonlinear system. The proposed control algorithm uses a simple static neural network, namely the multilayer perceptron (MLP), as a prediction model of the system to be controlled, and the TLBO strategy to solve the obtained constrained optimization problem. The efficiency and performance of the developed control algorithm are assessed by considering the control of two nonlinear systems: the continuous stirred tank reactor (CSTR) model and the two-degree-of-freedom manipulator robot model. To further demonstrate the effectiveness of the proposed controller, the control algorithm is numerically implemented and applied to the speed control of an induction motor. Furthermore, a comparative study with the conventional PID, the fuzzy logic control (FLC), the computed torque control (CTC) and the PSO-based neural network model predictive controller (NNMPC-PSO) controllers is carried out. The paper is organized as follows. The TLBO algorithm is described in Sect. 2. The proposed NNMPC-TLBO algorithm is presented in Sect. 3. In Sect. 4, the simulation results are given and discussed, while the experimental results are presented in Sect. 5. Finally, some conclusions are given in Sect. 6.

2 Teaching–Learning-Based Optimization Algorithm

TLBO is a teaching–learning process inspired algorithm that was proposed by Rao et al. (2011; 2012). Like other meta-heuristic methods, TLBO is a population-based method that uses a population of solutions to search for the global solution. The population is considered as a group of learners, and the search process is divided into two phases: the teacher and the learners phases. In the teacher phase, the learners learn from their teacher, while in the learners phase, the learners learn by interacting between themselves. The taught subjects are considered to be the cost function variables, and the fitness value of the optimization problem is considered as the learners' results. The teacher is considered to be the best solution in the entire population.

2.1 Teacher Phase

The algorithm starts with the teacher phase where the learners learn from their teacher. Assuming that m subjects (variables) exist with n learners (population size), at any iteration i, this phase can be decomposed into five sub-steps:

1. A teacher is selected from the population by choosing the best learner $X^i_{j\ k_{\text{best}}} (j = 1, 2, ..., m, k = 1, 2, ..., n)$ in each subject.
2. The learner mean result M_{ji}, for each subject j, is calculated using the following equation:

$$M_{ji} = \frac{\sum_{l=1}^{n} X^i_{jl}}{n}. \tag{1}$$

3. The distance between the teacher and the learners mean result, named the difference mean, is evaluated, for each subject, as follows:

$$d^i_{j\ k} = r_i (X^i_{j\ k_{\text{best}}} - T_F M_{ji}) \tag{2}$$

where r_i is a random number between 0 and 1, T_F is a random integer number, called the teaching factor, and its purpose is to determine the changed mean value. According to Rao (2016), good performances are obtained when T_F is chosen to be 1 or 2, where $T_F = \text{round}(1 + \text{rand}(0, 1))$.
4. The existing solutions are updated by adding the difference mean to each one of them:

$$Xnew^i_{j\ k} = X^i_{j\ k} + d^i_{j\ k}. \tag{3}$$

5. Finally, greedy selection is applied between the old and new updated solution $(Xnew^i_{j\ k}\ and\ X^i_{j\ k})$ in order to only keep the best solution.

2.2 Learners Phase

It is the phase where the learners try to improve themselves by interacting with each other. The learner X_A chooses randomly to interact with the learner X_B and learn from him. This phase can be decomposed into three sub-steps:

1. Choose randomly q pairs of solutions such that $F^i_A \neq F^i_B$, where F^i_A and F^i_B are the objective function values of X_A and X_B at the $i^t h$ iteration, respectively.
2. For each pair, update the solutions using the following equations (in case of a minimization problem):

$$Xnew^i_{j\ A} = X^i_{j\ A} + r_i (X^i_{j\ A} - X^i_{j\ B}), \quad \text{if } F^i_A < F^i_B \tag{4}$$

$$Xnew^i_{j\ A} = X^i_{j\ A} + r_i (X^i_{j\ B} - X^i_{j\ A}), \quad \text{if } F^i_B < F^i_A. \tag{5}$$

3. For each updated solution, evaluate the objective function. Accept the updated solution only if it enhances the solution.

2.3 Initialization Step

Before starting the TLBO algorithm, an initialization step is required to initialize the following parameters:

- The dimension m of the optimization problem (m = number of optimized variables).
- The population size n. The choice of the population size has a large impact on the meta-heuristic algorithms performance. So, choosing the population size depending on some factors, two of them are the converging speed and the solution accuracy (Chen et al. 2014; Bolufé-Rohler and Chen 2013).

- The maximum number of iterations $k_{\max}$. The choice of this parameter value depends on the chosen sampling time.
- The admissible maximum and minimum values for each variable $X_{j\min}$ and $X_{j\max}$.
- The initial solutions $X^1_{j\,k}$ are randomly chosen using the following equation:

$$X^1_{j\,k} = rand(0,1)\cdot(X_{j\max} - X_{j\min}) + X_{j\min}. \tag{6}$$

- The minimum value of the termination criterion ε.

To illustrate the TLBO algorithm, a flowchart is given in Fig. 1.

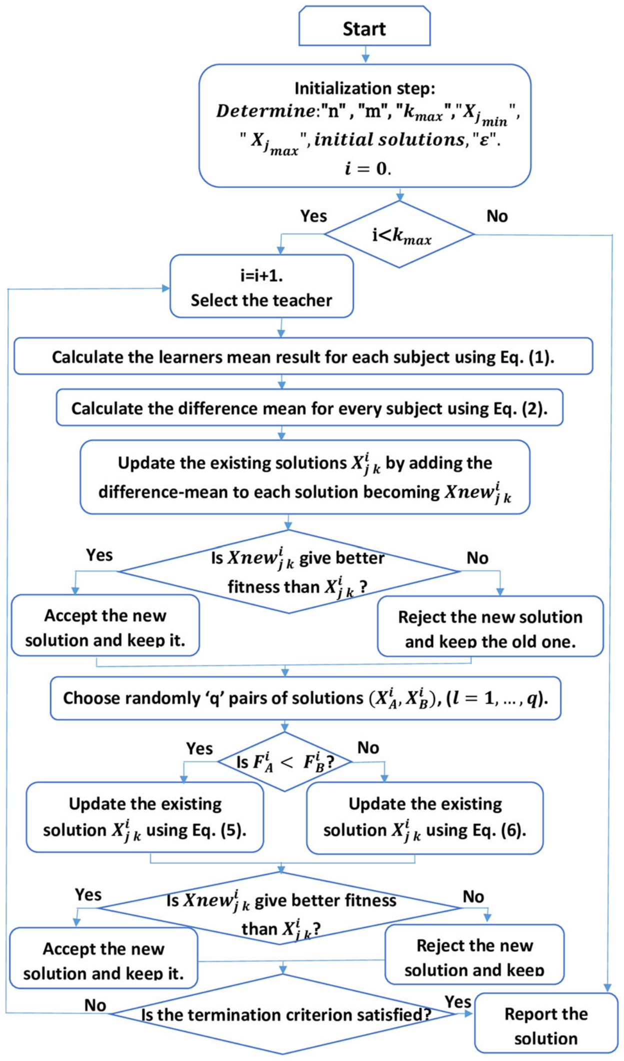
Start
Initialization step: Determine:"n" , "m", "k_{max}","$X_{j_{min}}$", "$X_{j_{max}}$", initial solutions, "ε". i = 0.
Yes
i<k_{max}
No
i=i+1. Select the teacher
Calculate the learners mean result for each subject using Eq. (1).
Calculate the difference mean for every subject using Eq. (2).
Update the existing solutions $X^i_{j\,k}$ by adding the difference-mean to each solution becoming $Xnew^i_{j\,k}$
Yes
Is $Xnew^i_{j\,k}$ give better fitness than $X^i_{j\,k}$?
No
Accept the new solution and keep it.
Reject the new solution and keep the old one.
Choose randomly 'q' pairs of solutions (X^i_A, X^i_B), $(l = 1,\dots,q)$.
Yes
Is $F^i_A < F^i_B$?
No
Update the existing solution $X^i_{j\,k}$ using Eq. (5).
Update the existing solution $X^i_{j\,k}$ using Eq. (6).
Yes
Is $Xnew^i_{j\,k}$ give better fitness than $X^i_{j\,k}$?
No
Accept the new solution and keep it.
Reject the new solution and keep
No
Is the termination criterion satisfied?
Yes
Report the solution

Fig. 1 TLBO algorithm flow chart

3 Neural Network Model Predictive Control based on TLBO Algorithm

3.1 Principle

Two main steps are common to almost all MPC algorithms: the explicit model used to predict the future process behaviour and the optimization problem from which a control sequence is derived. The different steps to design a MPC law are given as follows:

- At each sampling time, a reference trajectory is specified over a given prediction horizon.
- Using the system model, the future values of the system outputs are computed over a given prediction horizon.
- A future control sequence is calculated, over a given control horizon, by minimizing a given cost function.
- Only the first element of the calculated control sequence is to be applied on the system.

The NMPC optimization problem is generally defined as follows:

$$\begin{aligned}\min_{u(k)} J(\Delta u(k), \hat{y}(k), c(k)) &= \sum_{i=N_1}^{N_2}\left[\left(\hat{y}(k+i/k) - c(k+i)\right)^T Q \right. \\ &\left.\left(\hat{y}(k+i/k) - c(k+i)\right)\right] \\ &+ \sum_{i=1}^{N_u}\left[\Delta u^T(k+i-1) R \Delta u(k+i-1)\right]\end{aligned} \tag{7}$$

subject to

$$\begin{aligned}&\Delta u(k+i-1) = 0 \quad \text{for} \quad i > N_u \\ &y_{\min} < \hat{y}(k) < y_{\max} \\ &u_{\min} < u(k) < u_{\max} \\ &\Delta u_{\min} < \Delta u(k) < \Delta u_{\max} \quad \text{for} \quad i \le N_u\end{aligned}$$

where
$u(k)$ is the control input.
$\Delta u(k)$ is the control increment, and it is defined by:

$$\begin{aligned}&\Delta u(k) = u(k) - u(k-1) \quad \text{for} \quad i \le N_u \\ &\Delta u(k) = 0 \quad \text{for} \quad i > N_u\end{aligned}$$

N_1 and N_2 are the minimum and the maximum prediction horizon.
N_u is the control horizon.
Q and R are positive definite matrix and positive semi-definite matrix, respectively.

$\hat{y}(k)$ is the estimated value of the system output at time k.
$c(k)$ is the reference trajectory value at time k.

3.2 Constraints Handling

Predictive control is known by its ability to efficiently and directly handle the constraints by incorporating theme in the formulation of the optimization problem. Inputs and outputs constraints are due to physical limitations, safety reasons, economical and environmental objectives. Several approaches can be envisaged to handle these constraints (Coello 2002; Michalewicz and Schoenauer 1996; Koziel et al. 1999). In the proposed NNMPC-TLBO algorithm, the input constraints are directly handled by bounding the search space, while for handling the output ones, the penalty function approach is used. This penalty approach is based on adding new variables, called slack variables, to the cost function, that will heavily penalize any constraints violation. Using this approach, the optimization problem given in Eq. 7 will be reformulated to have the following expression:

$$\begin{aligned}\min_{u(k)} & J(\Delta u(k), \hat{y}(k), c(k)) \\ &= \sum_{i=N_1}^{N_2}\left[\left(\hat{y}(k+i/k)-c(k+i)\right)^T \Gamma_y \right. \\ &\left.\left(\hat{y}(k+i/k)-c(k+i)\right)\right] \\ &+ \sum_{i=1}^{N_u}\left[\Delta u^T(k+i-1) R \Delta u(k+i-1)\right]\end{aligned} \tag{8}$$

subject to

$$\begin{aligned}&\Delta u(k+i-1)=0 \quad \text{for} \quad i>N_u \\ &u_{\min} < u(k) < u_{\max} \\ &\Delta u_{\min} < \Delta u(k) < \Delta u_{\max} \quad \text{for} \quad i \le N_u\end{aligned}$$

where the output-dependent weight function $\Gamma_y(y)$ was chosen to replace the imposed output constraint, and it has the following expression:

$$\begin{bmatrix}\Gamma_{y_1}(y_1) & 0 & \dots & 0 \\ 0 & \Gamma_{y_2}(y_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Gamma_{y_n}(y_n)\end{bmatrix}$$

where

$$\Gamma_{y_i}(y_i) = \begin{cases}\Gamma_{y_i}(0) & \text{if} \quad y_{min_i} \le y_i \le y_{max_i} \\ \Gamma_{y_i}(0)[1+C_{i_y}] & \text{if} \quad y_i < y_{min_i} \text{ or } y_i > y_{max_i}\end{cases}.$$

$i = 1, 2, \dots, n$ (n is the outputs number).

$y(k) = [y_1(k), y_2(k), \dots, y_n(k)]^T$.
C_{i_y} is used to define the penalization degree: $C_{i_y} = 0$ indicates no constraint, while $C_{i_y} = \infty$ indicates hard constraint. In population-based algorithms, the input constraints can be systematically handled by bounding the search space to the inputs admissible values. The inputs constraints to be handled are:

– constraints on the input increment:

$$\Delta u_{\min} < \Delta u(k) < \Delta u_{\max} \tag{9}$$

– constraints on the input magnitude:

$$\begin{aligned}&u_{\min} < u(k) < u_{\max} \\ &u_{\min} - u(k-1) < u(k) - u(k-1) < u_{\max} - u(k-1) \\ &\quad \rightarrow u_{\min} - u(k-1) < \Delta u(k) < u_{\max} - u(k-1).\end{aligned} \tag{10}$$

Hence, from Eqs. 9 and 10, the upper and the lower bounds of the input increments will be given by:

$$L_{\min}(k) < \Delta u(k) < L_{\max}(k)$$

where

$$L_{\min}(k) = \begin{cases}\Delta u_{\min} & \text{if} \quad \Delta u_{\min} > u_{\min} - u(k-1) \\ u_{\min} - u(k-1) & \text{otherwise}\end{cases}$$
$$L_{\max}(k) = \begin{cases}\Delta u_{\max} & \text{if} \quad \Delta u_{\max} < u_{\max} - u(k-1) \\ u_{\max} - u(k-1) & \text{otherwise.}\end{cases}$$

3.3 Control Algorithm

Assuming that the MPC parameters (N_u, N_1, N_2 and C_{i_y}), the TLBO parameters (m, n and $k_{\max}$) and the neural network model have been established, the proposed NNMPC-TLBO algorithm can be described by the following steps:

- Step 1: Initialization
  - Let us take the control inputs, at the sampling time k, for the ith iteration, as $X^i_{j\ k}$,($j = 1, \dots, m$), where m denotes the number of control inputs.
  - For $j = 1 : m$
    - For $k = 1 : n$
      - . Choose the initial solution $X^1_{j\ k}$ using Eq. 6.
    - End
  - End
  - $i = 1$
- Step 2: Reference trajectory
  - Specify the reference trajectory between $k + N_1$ and $k + N_2$.

```
– Step 3: Teacher phase
    – Step 3_1: Determination of the teacher
        • For $k = 1 : n$
            . 1.Calculate the predicted values of the system outputs using the prediction model.
            . 2.Evaluate the objective function $F_k^i$ using Eq. 8.
        • End
        • $F_{k_{best}} = F_1^i$
        • For $k = 2 : n$
            . If $F_k^i < F_{k_{best}}$
            . $F_{k_{best}} = F_k^i$
            . $X_{j\ k_{best}} = X_{j\ k}^i$
            . End if
        • End
    – Step 3_2: Mean result calculation
        – For $j = 1 : m$
            – Calculate the mean result $M_{ji}$ using Eq. 1.
        – End
    – Step 3_3: difference mean calculation
        • For $j = 1 : m$
            . For $k = 1 : n$
              Calculate $d_{jk}^i$ using Eq. 2.
            . End
        • End
    – Step 3_4: Solution updating
        • For $j = 1 : m$
            . For $k = 1 : n$
              Calculate the new solutions $Xnew_{j\ k}^i$ using Eq. 3.
            . End
        • End
    – Step 3_5: Greedy selection
        • For $k = 1 : n$
            . 1.Using $Xnew_{j\ k}^i$, calculate the predicted values of the system outputs using the prediction model.
            . 2.Evaluate the objective function $new\_F_k^i$ using Eq. 8.
            . If $new\_F_k^i < F_k^i$
              $F_k^i = new\_F_k^i$
              $X_{j\ k}^i = Xnew_{j\ k}^i$
            . End if
            . If$F_k^i < F_{k_{best}}$
              $F_{k_{best}} = F_k^i$
              $X_{j\ k_{best}} = X_{j\ k}^i$
            . End if
        • End
– Step 4: Learner phase
    – Step 4_1: choosing the pairs to interact
        • Choose randomly q pairs of solutions such that $F_A^i \neq F_B^i$, where $F_A^i$ and $F_B^i$ are the objective function values of $X_A$ and $X_B$, respectively.
    – Step 4_2: Solutions updating
        • For $h = 1 : q$
          Update the solution $Xnew_{j\ k}^i$ using Eqs. 4 and 5.
        • End
    – Step 4_3: evaluating the new solutions
        • For $k = 1 : n$
            . 1.Using $Xnew_{j\ k}^i$, calculate the predicted values of the system outputs using the prediction model.
            . 2.Evaluate the objective function $new\_F_k^i$ using Eq. 8.
        • End
        • For $k = 1 : n$
            . If $new\_F_k^i < F_k^i$
              $F_k^i = new\_F_k^i$
              $X_{j\ k}^i = Xnew_{j\ k}^i$
            . End if
            . If $F_k^i < F_{k_{best}}$
              $F_{k_{best}} = F_k^i$
              $X_{j\ k_{best}} = X_{j\ k}^i$
            . End if
        • End
– Step 5: iterative process
    – if $F_{k_{best}} < \varepsilon$ or $i > k_{\max}$
      $i = 1$
      Go to step 6.
    – else
      $i = i + 1$
      Go back to step 3.
    – End if
– Step 6:
    – 1. Apply the obtained control value (the first element of $X_{j\ k_{best}}$ on the system.
    – 2. Wait for the next sampling time, and then go back to step 2.
```

4 Simulation Study

The efficiency of the proposed NNMPC-TLBO control algorithm is investigated in this section, by considering the control of the models of two highly nonlinear systems. The obtained results are then compared to those obtained using the NNMPC-PSO, the conventional PID controller, the computed torque control and the fuzzy logic controller. The first system is the continuous stirred tank reactor; it is a highly nonlinear chemical process. The second one is the two-degree-of-freedom manipulator robot; it is a coupled multi-input multi-output mechanical system.

4.1 Continuous Stirred Tank Reactor

The CSTR is a chemical process where a product A will be converted to another product B via an exothermic chemical reaction. The reactor volume v is constant, and the mixture is considered perfect with a uniform temperature T. The reactor operates continuously, its output (the mixture concentration C_a) changes nonlinearly according to the mixture temperature T, and the temperature changes according to the system input coolant flow q_c. This process is depicted in Fig. 2.

The CSTR is described by the following equations:

$$\begin{aligned} \dot{C}_a(t) =& \frac{q}{v}\left(C_{a0} - C_a(t)\right) - k_0 C_a(t) e^{\frac{-E}{RT(t)}} \\ \dot{T}(t) =& \frac{q}{v}\left(T_0 - T(t)\right) + k_1 C_a(t) e^{\frac{-E}{RT(t)}} \\ &+ k_2 q_c(t)\left(1 - e^{\frac{-k_3}{q_c(t)}}\left(T_{c0} - T(t)\right)\right) \end{aligned} \tag{11}$$

where

$$k_1 = -\frac{\Delta H k_0}{\rho C_p}, \quad k_2 = \frac{\rho_c C_{pc}}{\rho C_p v}, \quad k_3 = \frac{h_a}{\rho_c C_{pc}}$$

where v is the reactor volume expressed in litre, C_{a0} is the initial mixture concentration expressed in mol/l, T_0 is the initial temperature expressed in Kelvin, T_{c0} is the initial jacket temperature expressed in Kelvin, T_c is the jacket temperature expressed in Kelvin, $C_a(t)$ is the mixture concentration expressed in mol/l, $T(t)$ is the mixture temperature expressed in Kelvin and $q_c(t)$ is the coolant flow expressed in l/min.

The constants values are given in Table 1.

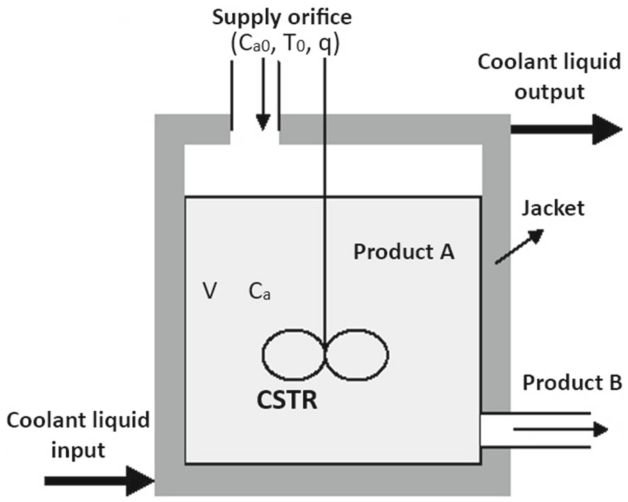

Fig. 2 Continuous Stirred Tank Reactor

Table 1 CSTR constants values

Constant	Value	Constant	Value
q	100	ρ	1000
v	100	ρ_c	1000
k_0	7.2e10	C_p	1
E	10,000	C_{pc}	1
T_0	350	h_a	700,000
T_{c0}	350	C_{a0}	1

4.1.1 Process Identification

The first step of any MPC design is obtaining a model for the system to be controlled. This model is used to predict the future values of the system outputs. In the proposed NNMPC-TLBO controller, a static MLP with the following configuration is used.

- The input layer contains four inputs : $\left(q_c(k),\ q_c(k-1),\ C_a(k-1),\ C_a(k-2)\right)$.
- One hidden layer containing ten neurons with a sigmoid activation function.
- The output layer, which contains a single neuron with a linear activation function, gives the estimated output $\hat{C}_a(k)$.

Two different datasets, which cover all possible system working regimes, are generated using the system model expressed in equation (11). The first one is used to train the neural network, and the second one is used to test the trained model. Figure 3 gives the system and the model outputs which are superposed and the modelling error which is quite small. The values of the root mean square of the modelling error (RMSE) and the correlation coefficient (R^2), which are 6.909e-5 and 0.99999943, respectively, show the good accuracy of the obtained model.

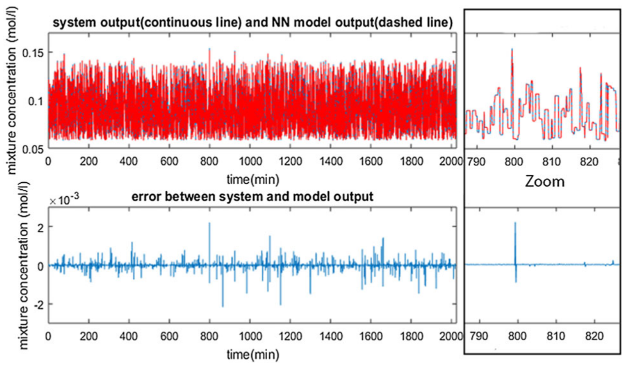

Fig. 3 Test results of the obtained model

4.1.2 Controller Implementation

Four controllers are designed to control the mixture concentration $C_a(t)$, namely the proposed NNMPC-TLBO, the NNMPC-PSO, the fuzzy logic controller (FLC) and the conventional PID controller. In case of the first two controllers, the control block diagram is given in Fig. 4. The parameters of the PSO algorithm used in the second controller are gathered in Table 2. The design procedure and the parameters values of the FLC and the PID controllers can be found in Benrabah et al. (2019). Both NNMPC controllers use the same control parameters values given in Table 3. In this case, no constraint on the output was imposed. The obtained control results are given in Fig. 5, when the reference trajectory moves to 0.08, 0.11, 0.14, 0.11 and 0.08 mol/l. For each controller and for the same initial conditions, the average values, over a thousand runs, of the mean squared error (MSE), the mean absolute error (MAE) and the RMSE are given in Table 4. The execution time is very important in order to evaluate the computing efficiency and the real-time applicability of the proposed controller. Starting the NNMPC-TLBO and the NNMPC-PSO from the same initial conditions, the required time to obtain a value of the control signal is evaluated for both controllers and given in Table 5.

From Fig. 5 and Table 4, it can be seen that a good tracking accuracy of the reference trajectory is obtained for both

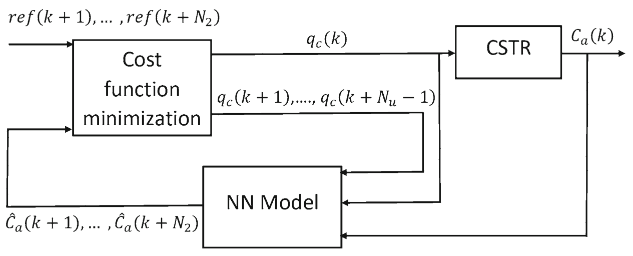
$ref(k+1),\ldots,ref(k+N_2)$
Cost function minimization
$q_c(k)$
CSTR
$C_a(k)$
$q_c(k+1),\ldots,q_c(k+N_u-1)$
NN Model
$\hat{C}_a(k+1),\ldots,\hat{C}_a(k+N_2)$

Fig. 4 CSTR control block diagram

Table 2 PSO parameters values

Parameter	Value	Parameter	Value
c_1	2	ω	1
c_2	2	ω_d	0.99

Table 3 MPC parameters values

Parameter	Value	Parameter	Value
N_u	2	$k_{\max}$	10
N_1	1	n	8
N_2	3	m	2
R	9.1809e-04	Sampling time	6 s
$u_{\min}$	88	$u_{\max}$	112

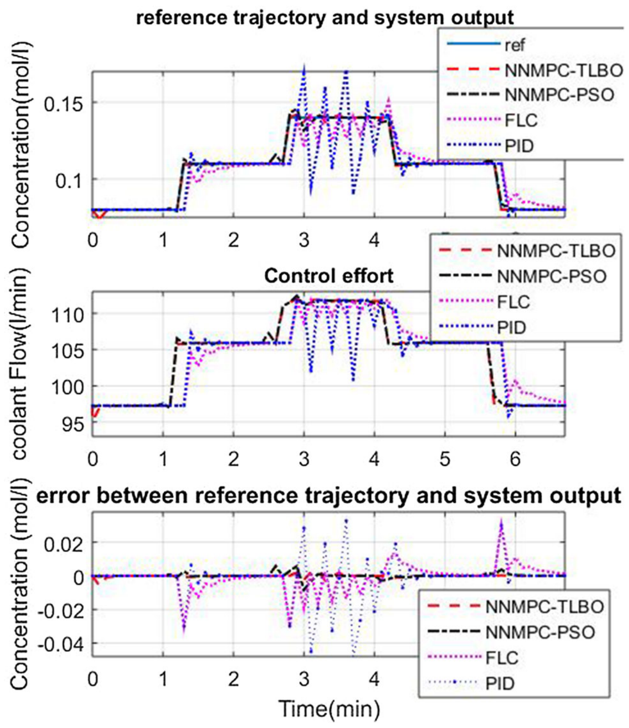

Fig. 5 Control results of the CSTR model using NNMPC-TLBO, NNMPC-PSO, FLC and PID controllers

NNMPC-TLBO and NNMPC-PSO controllers. However, in case of the FLC and the PID controllers, a degradation in the control performance for large amplitude of the reference trajectory is observed. Comparing the MSE, the MAE and the RMSE values given in Table 4, we note that the proposed controller is more efficient in this experiment than the NNMPC-PSO, the FLC and the PID controllers. From Table 5, we notice that the computing time of the NNMPC-TLBO is lower than that of the NNMPC-PSO.

In a second case, an output constraint limiting the overshoot to no more than 2% is imposed. Since the FLC and the PID controllers cannot handle constraints, the comparison is only made between the NNMPC-TLBO and NNMPC-PSO. The NNMPC-TLBO and NNMPC-PSO are implemented using the parameters values given in Tables 2 and 3. The constraints handling parameter C_{1_y} is taken equal to 100. The obtained results are summarized in Fig. 6. Table 6 gives the maximal values of the process output and the maximal values of the control input, for both controllers, when the reference trajectory moves to 0.08, 0.11 and 0.14 mol/l. Using the same initial conditions, the average values, over a thousand runs, of the computing time, the MSE, the MAE and the RMSE are given in Table 7.

It can be observed from Fig. 6 and Table 6 that the imposed constraints are respected for both controllers, and the obtained overshoots for the NNMPC-TLBO controller when the reference trajectory is set to 0.11 mol/l or 0.14 mol/l

Table 4 MSE, MAE and RMSE values for each controller

Controller	NNMPC-TLBO	NNMPC-PSO	FLC	PID
MSE	3.523e-06	4.670e-06	6.47e-05	1.67e-04
MAE	7.225e-04	9.635e-04	0.0042	0.0058
RMSE	0.0018	0.0021	0.0080	0.0130

Table 5 Computing time for NNMPC-TLBO and NNMPC-PSO algorithms

Controller	NNMPC-TLBO	NNMPC-PSO
Computing time (ms)	6.3588	6.4003

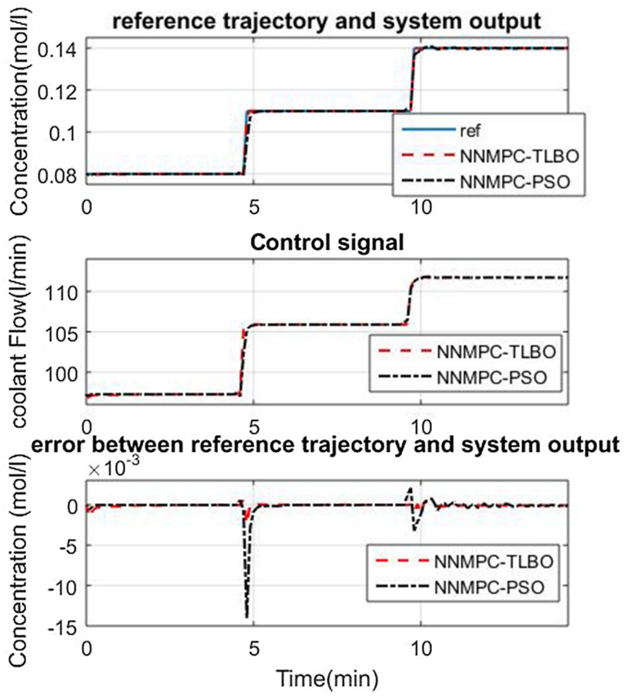

Fig. 6 Control results using NNMPC-TLBO and NNMPC-PSO controllers for the case of imposed output constraints

are smaller than those obtained in the case of the NNMPC-PSO controller. Comparing the computing time, MSE, MAE and RMSE values given in Table 7, we note that the proposed algorithm is faster and more efficient than the NNMPC-PSO controller.

Table 7 Computing time, MSE, MAE and RMSE values in case of an imposed output constraint

Controller	NNMPC-TLBO	NNMPC-PSO
MSE	1.1534e-06	3.2040e-06
MAE	2.5586e-04	3.3341e-04
RMSE	9.7822e-04	0.0015
Computing time (ms)	3.9463	5.0843

4.2 Two-Degree-of-Freedom Manipulator Robot

The dynamic model of the considered manipulator (Fig. 7), is expressed as follows (Lin. 2007):

$$\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} P_{11} \\ P_{21} \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} + \begin{bmatrix} g_{1q} \\ g_{2q} \end{bmatrix} = \begin{bmatrix} \tau_{fin1} \\ \tau_{fin2} \end{bmatrix} \tag{12}$$

where

τ_{fin1}, τ_{fin1} are the control torques for joints 1 and 2, respectively,

$$\begin{aligned} Q_{11} =& I_1 + I_2 + m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2}\cos\theta_2 \\ &+ m_{33}(l_1^2 + l_2^2 + 2l_1 l_2 \cos\theta_2). \\ Q_{12} =& I_2 + m_2 l_{c2}^2 + m_2 l_1 l_{c2}\cos\theta_2 + m_{33}(l_2^2 + l_1 l_2 \cos\theta_2). \\ Q_{21} =& Q_{12}. \\ Q_{22} =& I_2 + m_2 l_{c2}^2 + m_{33} l_2^2. \\ P_{11} =& -l_1(2\dot{\theta}_1 + \dot{\theta}_2)\dot{\theta}_2 \sin\theta_2 (l_{c2} m_2 + l_2 m_{33}). \\ P_{21} =& l_1 l_{c2}\dot{\theta}_1^2 \sin\theta_2 (m_2 + m_{33}). \end{aligned}$$

Table 6 Input and output CSTR limits values using both NNMPC_TLBO and NNMPC_PSO controllers

		NNMPC_TLBO	NNMPC_PSO
Reference = 0.08 (mol/l)	Maximal value of C_a (mol/l)	0.0807	0.0806
	Overshoot (%)	0.875%	0.75%
Reference = 0.11 (mol/l)	Maximal value of C_a (mol/l)	0.1101	0.1121
	Overshoot (%)	0.0909%	1.9091%
Reference = 0.14 (mol/l)	Maximal value of C_a (mol/l)	0.1404	0.1408
	Overshoot (%)	0.2857%	0.5714%

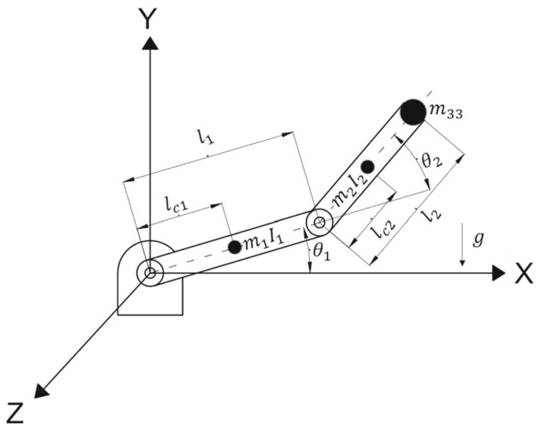

Fig. 7 Two-degree-of-freedom planar robotic manipulator

Table 8 Parameters values of the considered manipulator

Parameter	Value	Parameter	Value
m_1	0.392924 (kg)	l_{c1}	0.104648 (m)
m_2	0.094403 (kg)	l_{c2}	0.081788 (m)
m_{33}	0.2 (kg)	I_1	0.0011411 (kg m^2)
g	9.81 (m/s^2)	I_2	0.0020247 (kg m^2)
l_1	0.2032 (m)	b_{1q}	0.141231 (N)
l_2	0.1524 (m)	b_{2q}	0.353078 (N)

$$\begin{aligned}
f_1 &= b_{1q}\dot{\theta}_1.\\
f_2 &= b_{2q}\dot{\theta}_2.\\
g_{1q} &= m_1 l_{c1} g \cos\theta_1 + m_2 g(l_{c2}\cos(\theta_1+\theta_2) + l_1\cos\theta_1)\\
&\quad + m_{33} g(l_2\cos(\theta_1+\theta_2) + l_1\cos\theta_1).\\
g_{2q} &= (m_2+m_{33}) g l_{c2}\cos(\theta_1+\theta_2).
\end{aligned}$$

The numerical values for the different constants are given in Table 8.

4.2.1 System Identification

Two static MLPs are used to emulate the dynamics of the considered manipulator. Their structure is defined as follows:

– Both MLPs have the same input vector, which is given by:

$$[\tau_{fin1}(k), \tau_{fin1}(k-1), \tau_{fin2}(k), \tau_{fin2}(k-1), \theta_1(k), \theta_1(k-1), \theta_2(k), \theta_2(k-1)].$$

– Each MLP has one hidden layer containing 20 neurons with a sigmoid activation function.
– Each MLP has one neuron in the output layer with a linear activation function.

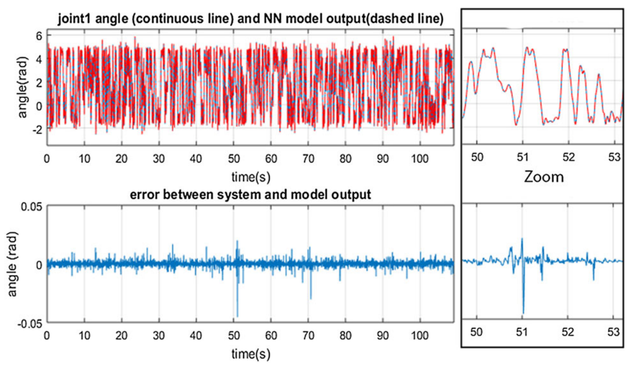

Fig. 8 Joint1 NN model validation result

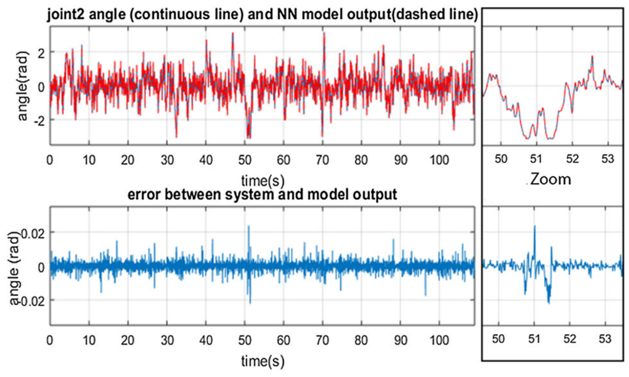

Fig. 9 Joint2 NN model validation result

– The first MLP gives the estimated value $\hat{\theta}_1(k)$ of joint1 angle, while the second one gives the estimated value $\hat{\theta}_2(k)$ of joint2 angle.

The neuronal models are trained using the dataset generated from the state model given in Eq. 12. Figures 8 and 9 show the test results of the obtained models, where it can be seen that the modelling error is quite small. The values of the $RMSE$ and R^2 for the model of θ_1 and the model of θ_2 are : 0.00200654, 0.99999950718, 0.00211524 and 0.99999277711, respectively. The RMSE values are close to zero, and R^2 values are close to one, which indicates that the obtained models have good accuracy.

4.2.2 Controller Implementation

Three controllers, namely computed torque control (CTC), NNMPC-TLBO and NNMPC-PSO, were designed to control the angular positions of joints 1 and 2 of the considered manipulator.

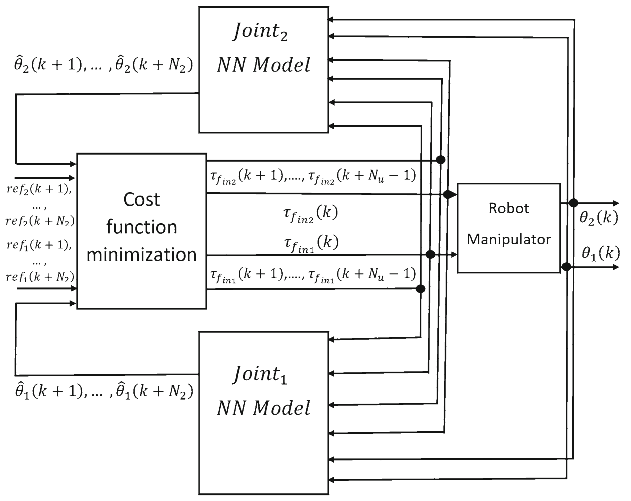
$Joint_2$ NN Model
$\hat{\theta}_2(k+1),\dots,\hat{\theta}_2(k+N_2)$
Cost function minimization
$ref_2(k+1),\dots,ref_2(k+N_2)$
$ref_1(k+1),\dots,ref_1(k+N_2)$
$\tau_{fin_2}(k+1),\dots,\tau_{fin_2}(k+N_u-1)$
$\tau_{fin_2}(k)$
$\tau_{fin_1}(k)$
$\tau_{fin_1}(k+1),\dots,\tau_{fin_1}(k+N_u-1)$
Robot Manipulator
$\theta_2(k)$
$\theta_1(k)$
$Joint_1$ NN Model
$\hat{\theta}_1(k+1),\dots,\hat{\theta}_1(k+N_2)$

Fig. 10 Robot manipulator control block diagram

Table 9 MPC parameters values

Parameter	Value	Parameter	Value
N_u	1	$k_{\max}$	10
N_1	1	n	8
N_2	3	m	2
R	0	Sampling time	0.01 s
$\tau_{fin1\,\min}$	−30	$\tau_{fin2\,\min}$	−20
$\tau_{fin1\,\max}$	30	$\tau_{fin2\,\max}$	20

The CTC control law is given by:

$$\begin{bmatrix}\tau_{fin1}\\ \tau_{fin2}\end{bmatrix}=\begin{bmatrix}Q_{11} & Q_{12}\\ Q_{21} & Q_{22}\end{bmatrix}\left(\begin{bmatrix}\ddot{\theta}_{d_1}\\ \ddot{\theta}_{d_2}\end{bmatrix}+k_v\begin{bmatrix}\dot{\theta}_{d_1}-\dot{\theta}_1\\ \dot{\theta}_{d_2}-\dot{\theta}_2\end{bmatrix}+k_p\begin{bmatrix}\theta_{d_1}-\theta_1\\ \theta_{d_2}-\theta_2\end{bmatrix}\right)+\begin{bmatrix}P_{11}\\ P_{21}\end{bmatrix}+\begin{bmatrix}f_1\\ f_2\end{bmatrix}+\begin{bmatrix}g_{1q}\\ g_{2q}\end{bmatrix}$$

where θ_{d_1} and θ_{d_2} are the desired references trajectories. The k_v and k_p values, optimized using the PSO algorithm, are 26.3077 and 2.6853e+02, respectively. The control block diagram of the NNMPC-TLBO and NNMPC-PSO is given in Fig. 10, and the design parameters for both controllers are given in Table 9. The parameters of the PSO algorithm are given in Table 2.
The efficiency of the proposed controller is evaluated by considering the two following cases.

In the first case, no outputs constraints are imposed and two different reference trajectories are used. The obtained results are shown in Fig. 11 for the multistep trajectory and Fig. 12 for the sinusoidal trajectory.

A good tracking accuracy of the reference trajectories is obtained for all controllers. However, the tracking error of the proposed controller is slightly smaller than that of the other controllers.

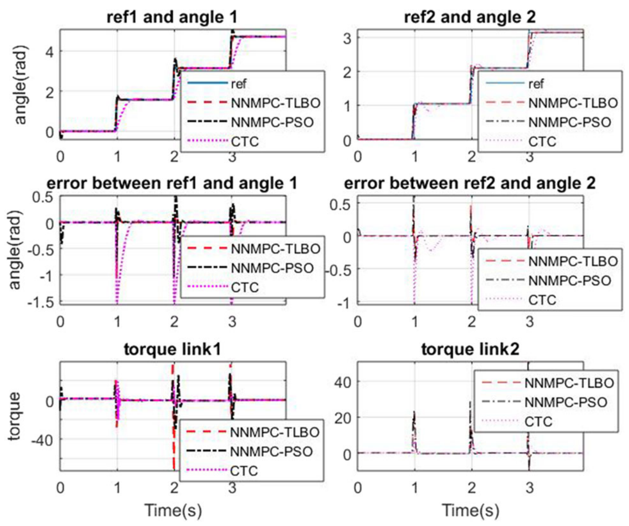

Fig. 11 Control performance using NNMPC-TLBO, NNMPC-PSO and CTC controllers with a multistep trajectory

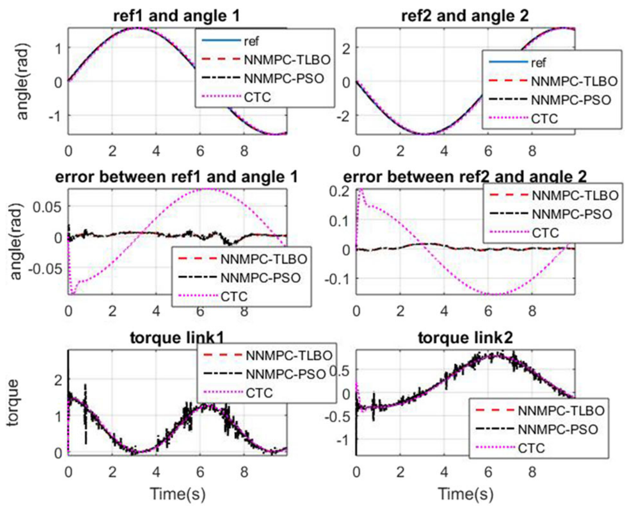

Fig. 12 Control performance using NNMPC-TLBO, NNMPC-PSO and CTC controllers with a sinusoidal trajectory

To see the difference in the control performance between the three controllers, the same experiment is repeated thousand times, and at each run, the population of the PSO and TLBO algorithm is randomly initialized. The average values of the MSE, the MAE and the RMSE are given in Table 10, where it can be seen that the NNMPC-TLBO controller has better tracking accuracy than the NNMPC-PSO and the CTC controllers. The computing time for both NNMPC-TLBO and NNMPC-PSO algorithms is calculated and given in Table 11, and it can be seen that the computing time of the proposed controller is smaller than that of the NNMPC-PSO algorithm.

In the second case, output constraints, limiting the overshoot to no more than 1%, are imposed. The design parameters given in Table 9 are used for both controllers. The output-dependent weight function $\Gamma_y(y)$ has the following

Table 10 MSE, MAE and RMSE values using NNMPC-TLBO, NNMPC-PSO and CTC controllers

		NNMPC-TLBO	NNMPC-PSO	CTC
Multistep trajectory	MSE	0.0125	0.0153	0.1525
	MAE	0.0284	0.0361	0.1838
	RMSE	0.1116	0.1195	0.3905
Sinusoidal trajectory	MSE	6.69e-05	1.0087e-04	0.0145
	MAE	0.0082	0.0085	0.1424
	RMSE	0.0082	0.0097	0.1206

Table 11 Computing time for NNMPC-TLBO and NNMPC-PSO algorithms in case of reference and multistep trajectories

		NNMPC-TLBO	NNMPC-PSO
Computing time (ms)	Multistep trajectory	4.7905	6.0743
	Sinusoidal trajectory	6.389	6.8098

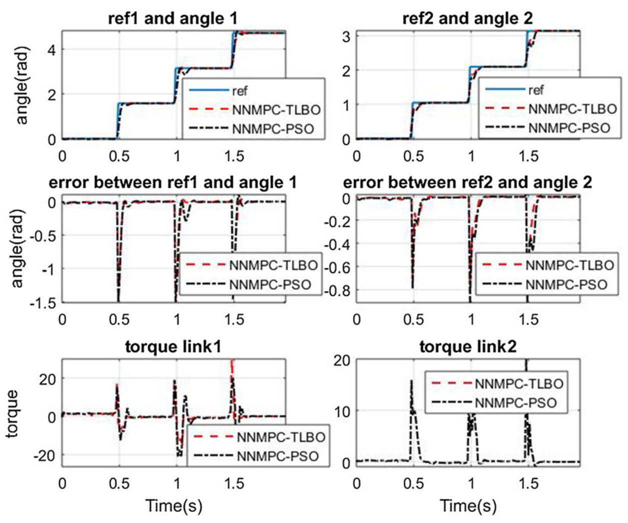

Fig. 13 Control performance using NNMPC-TLBO and NNMPC-PSO controllers with output constraints

Table 12 Computing time, MSE, MAE and RMSE values using NNMPC-TLBO and NNMPC-PSO controllers with output constraints

	NNMPC-TLBO	NNMPC-PSO
MSE	0.0874	0.0939
MAE	0.1108	0.1264
RMSE	0.2956	0.3064
Computing time (ms)	6.6749	7.7482

parameters: $C_{1_y} = 100$, $C_{2_y} = 100$. Figure 13 gives the obtained control performance, and Table 12 gives the average values, over thousand runs, of the computing time, the MSE, the MAE and the RMSE.
From Fig. 13 and Table 12, it can be concluded that the TLBO-based controller has better control performance than the PSO-based controller.

Remark The following set-up has been used to calculate all computing times cited above:

1. MATLAB environment.
2. Desktop computer with the following hardware:
   - Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz.
   - 4.00 Go of RAM
   - Windows 10 Pro x64.

5 Experimental Study

To further demonstrate the effectiveness of the proposed controller, the experimental set-up shown in Fig. 14 is used to control the speed of an induction motor. In addition to the three-phase squirrel-cage induction motor, this experimental set-up contains the following elements: a three-phase voltage source inverter, a TMS320F28335 DSP board and a single computer board RASPBERRY PI 3B+. The control algorithm is implemented in the single computer board RASPBERRY, and the DSP board is used to generate the six required pulse width modulated (PWM) signals and measure the motor speed. The control block diagram is given in Fig. 15.

5.1 Identification of the System

A neural network model, for the considered induction motor, is derived using the experimental collected data. This model is a simple static MLP with an input layer containing four inputs $[v_1(k), v_2(k), v_3(k), \omega(k-1)]$, a hidden layer of four neurons with sigmoid activation function and an output layer that contains one linear neuron representing the estimated angular speed $\hat{\omega}(k)$. $v_1(k)$, $v_2(k)$ and $v_3(k)$ are the applied voltages on the motor at the sampling time k and $\omega(k-1)$ is the measured angular speed of the motor. The response of

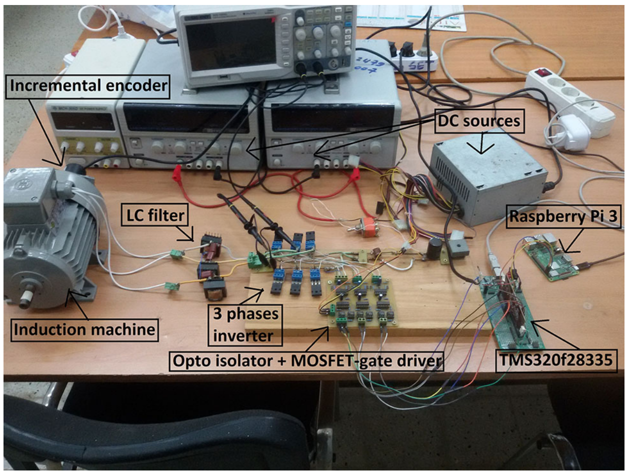

Fig. 14 Experimental set-up

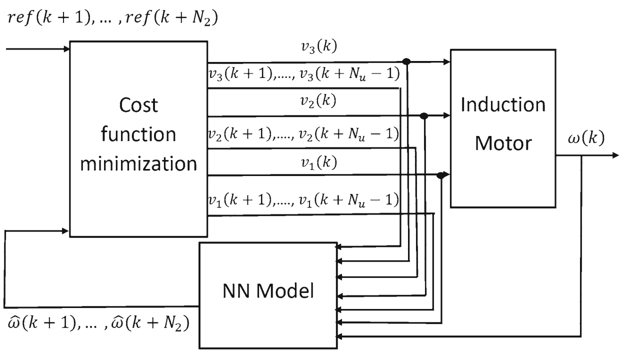
$ref(k+1),\dots,ref(k+N_2)$
Cost function minimization
$v_3(k)$
$v_3(k+1),\dots,v_3(k+N_u-1)$
$v_2(k)$
$v_2(k+1),\dots,v_2(k+N_u-1)$
$v_1(k)$
$v_1(k+1),\dots,v_1(k+N_u-1)$
Induction Motor
$\omega(k)$
NN Model
$\hat{\omega}(k+1),\dots,\hat{\omega}(k+N_2)$

Fig. 15 The induction motor control block diagram

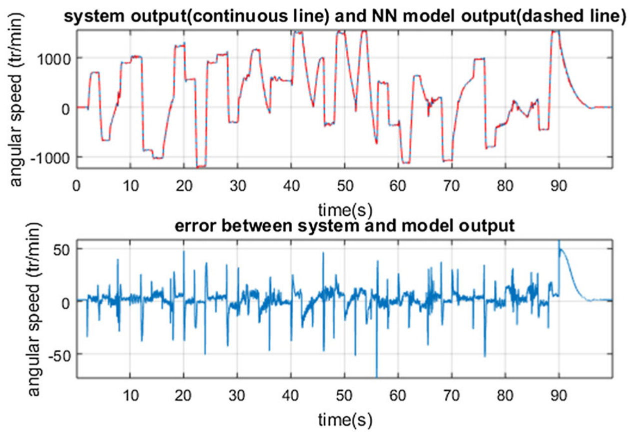

Fig. 16 Model testing results for the induction motor

the obtained model to the input test is given in Fig. 16, where it can be seen the modelling error is quite small. The RMSE and the R^2 values of the obtained model are 11.19669 and 0.99976, respectively. Since the amplitude of the system output ranges between − 1500 and 1500 tr/min, the RMSE value is acceptable. The R^2 value is close to one, which indicates that the model has good accuracy.

Table 13 Values of the design parameters

Parameter	Value	Parameter	Value
N_u	1	$k_{\max}$	10
N_1	1	n	10
N_2	3	m	3
R	10	Sampling time	0.01 s

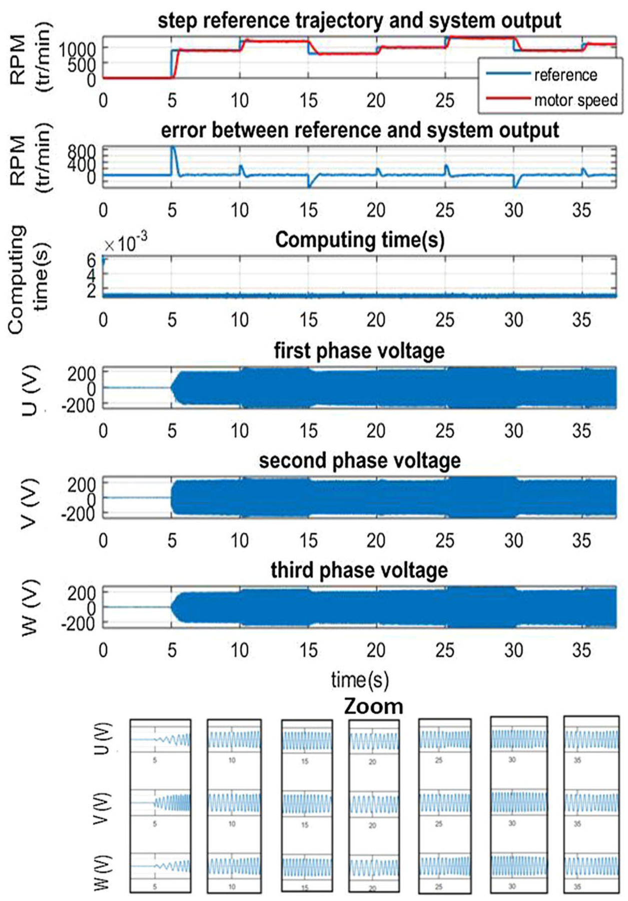

Fig. 17 NNMPC-TLBO controller performance for the case of a multistep trajectory

5.2 Controller Implementation

The controller is implemented using the design parameters given in Table 13 and the obtained neural network model. The control objective is to force the motor speed to track two different reference trajectories: the sinusoidal and the multistep trajectories. The results are given in Figs. 17 and 18.

Figures 17 and 18 show that good tracking accuracy is obtained for both trajectories, and the maximal value for the overshoot does not exceed 4%. For both (step and sinusoidal) reference trajectories, the maximum values of the computing time are 5.225 ms and 6.402 ms, respectively, which indicates

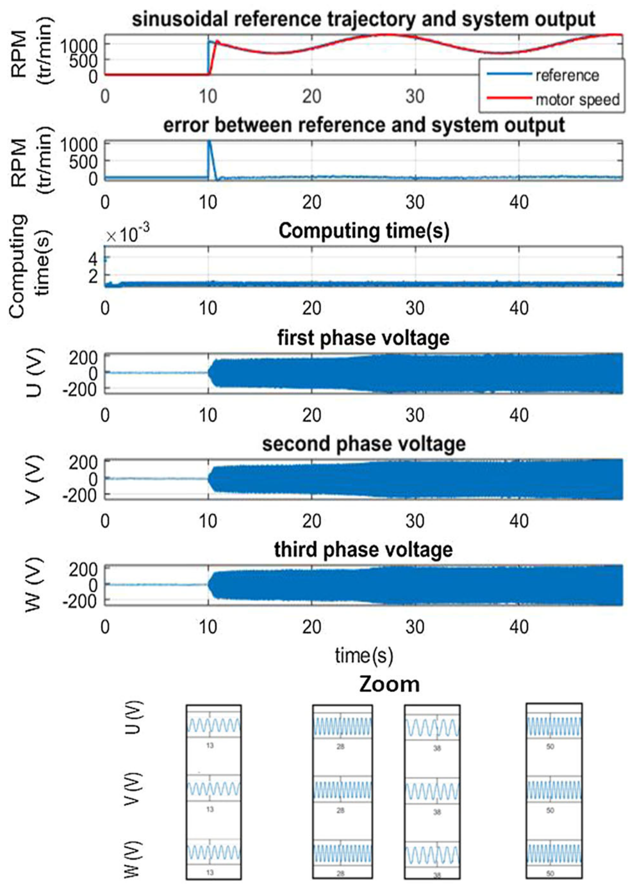

Fig. 18 NNMPC-TLBO controller performance for the case of a sinusoidal trajectory

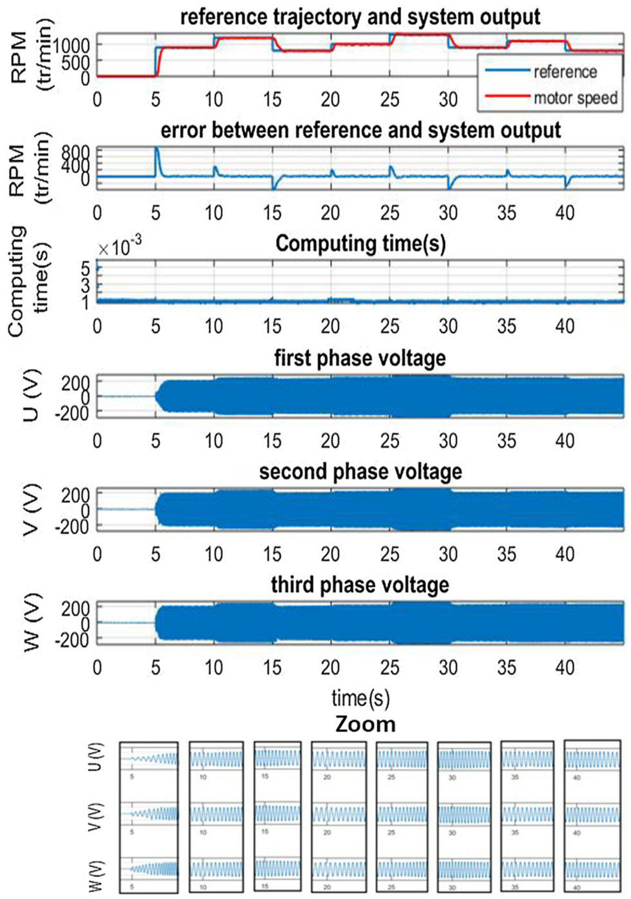

Fig. 19 NNMPC-TLBO controller performance in the presence of output constraints

that the proposed controller can be implemented in real time to control systems with fast dynamics.

In a second experiment, to further reduce the observed overshoot, a constraint on the output limiting the overshoot to no more than 2% is included. A multistep reference trajectory and the parameter $C_y = 100$ of the output-dependent weight function $\Gamma_y(y)$ are used in this experiment. The obtained control results are given in Fig. 19, where it can be seen that the overshoot value does not exceed 2% and the maximum value of the computing time is 5.839ms.

Table 14 gives the values of the MSE, the MAE and the RMSE for the released experiments. From this table, we can conclude that the proposed controller gives good control performances and the imposed constraints are respected.

6 Conclusion

An efficient constrained nonlinear predictive control algorithm, called neural network model predictive control using teaching–learning-based optimization, has been proposed. This controller uses a neural network model to predict the future values of the system outputs and the TLBO method to solve the resulting optimization problem. Neural networks are known for their simplicity and precision in approximating any nonlinear mapping, and the TLBO method does not require any algorithm-specific parameters. Although the proposed controller incorporate in its design the imposed constraints, its structure is simple and can be easily implemented to control, in real time, nonlinear systems with fast dynamics. It has been shown, using both simulation and experimental studies, that the proposed controller gives good control performance in terms of the tracking accuracy, computing time, constraints handling and the overshoot amplitude. The NNMPC-TLBO algorithm can be successfully used to control different classes of nonlinear systems. Indeed, the control of a highly nonlinear chemical process, a coupled multivariable mechanical system and an electrical machine were considered.

Table 14 MSE, MAE and RMSE using NNMPC-TLBO controller

	MSE	MAE	RMSE
Multistep trajectory (without constraints)	101.3805	0.2908	10.0688
Sinusoidal trajectory (without constraints)	67.3236	0.1897	8.2051
Multistep trajectory (with constraints)	86.6623	0.2888	9.3093

Acknowledgements This work was supported by the DGRSDT (Direction Générale de la Recherche Scientifique et du Développement Technologique).

References

Abdennour, R., Ksouri, M., & M'Sahli, F. (2002). Nonlinear model-based predictive control using a generalised hammerstein model and its application to a semi-batch reactor. *The International Journal of Advanced Manufacturing Technology*, *20*(11), 844–852. https://doi.org/10.1007/s001700200225.

Ait-Sahed, O., Kara, K., & Benyoucef, A. (2015). Artificial bee colony-based predictive control for non-linear systems. *Transactions of the Institute of Measurement and Control*, *37*(6), 780–792. https://doi.org/10.1177/0142331214546796.

Ait-Sahed, O., Kara, K., Benyoucef, A., & Hadjili, M. L. (2016). An efficient artificial bee colony algorithm with application to nonlinear predictive control. *International Journal of General Systems*, *45*(4), 393–417. https://doi.org/10.1080/03081079.2015.1086344.

Bolufe-Rohler, A., & Chen, S. (2013). Minimum population search: Lessons from building a heuristic technique with two population members. In *2013 IEEE congress on evolutionary computation* (pp. 2061–2068). IEEE https://doi.org/10.1109/cec.2013.6557812.

Botto, M. A., Boom, T. J. J. V. D., Krijgsman, A., & Costa, J. S. D. (1999). Predictive control based on neural network models with i/o feedback linearization. *International Journal of Control*, *72*(17), 1538–1554. https://doi.org/10.1080/002071799220038.

Camacho, E. F., & Bordons, C. (2007). *Model predictive control*. London: Springer. https://doi.org/10.1007/978-0-85729-398-5.

Chen, S., Montgomery, J., & Bolufé-Röhler, A. (2014). Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence*, *42*(3), 514–526. https://doi.org/10.1007/s10489-014-0613-2.

Clarke, D., Mohtadi, C., & Tuffs, P. (1987a). Generalized predictive control—Part I. The basic algorithm. *Automatica, 23*(2), 137–148. https://doi.org/10.1016/0005-1098/2887/2990087-2.

Clarke, D., Mohtadi, C., & Tuffs, P. (1987b). Generalized predictive control—Part II. Extensions and interpretations. *Automatica, 23*(2), 149–160. https://doi.org/10.1016/0005-1098/2887/2990088-4.

Coelho, J., de Moura, Oliveira P., & Cunha, J. B. (2005). Greenhouse air temperature predictive control using the particle swarm optimisation algorithm. *Computers and Electronics in Agriculture*, *49*(3), 330–344. https://doi.org/10.1016/j.compag.2005.08.003.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, *191*(11–12), 1245–1287. https://doi.org/10.1016/s0045-7825/2801/2900323-1.

Cutler, C. R., & Ramaker, B. L. (1980). Dynamic matrix control? A computer control algorithm. In *Joint automatic control conference*, 17, (p. 72).

Díaz-Mendoza, R., & Budman, H. (2009). Robust nonlinear model predictive control using volterra models and the structured singular value (μ). *IFAC Proceedings Volumes*, *42*(11), 375–380. https://doi.org/10.3182/20090712-4-tr-2008.00059.

Edgar, T., Gong, J., Lou, H., & Huang, Y. (2000). Fuzzy model predictive control. *IEEE Transactions on Fuzzy Systems*, *8*(6), 665–678. https://doi.org/10.1109/91.890326.

Ghorbani, S., Barari, M., & Hosseini, M. (2017). A modern method to improve of detecting and categorizing mechanism for micro seismic events data using boost learning system. https://doi.org/10.20944/preprints201708.0072.v1.

Gruber, J., Ramirez, D., Limon, D., & Alamo, T. (2013). Computationally efficient nonlinear min-max model predictive control based on volterra series models: Application to a pilot plant. *Journal of Process Control*, *23*(4), 543–560. https://doi.org/10.1016/j.jprocont.2013.01.007.

Han, H., & Qiao, J. (2014). Nonlinear model-predictive control for industrial processes: An application to wastewater treatment process. *IEEE Transactions on Industrial Electronics*, *61*(4), 1970–1982. https://doi.org/10.1109/tie.2013.2266086.

Han, H. G., Wu, X. L., & Qiao, J. F. (2013). Real-time model predictive control using a self-organizing neural network. *IEEE Transactions on Neural Networks and Learning Systems*, *24*(9), 1425–1436. https://doi.org/10.1109/tnnls.2013.2261574.

Jiang, H., Kwong, C., Chen, Z., & Ysim, Y. (2012). Chaos particle swarm optimization and t-s fuzzy modeling approaches to constrained predictive control. *Expert Systems with Applications*, *39*(1), 194–201. https://doi.org/10.1016/j.eswa.2011.07.007.

Kansha, Y., & Chiu, M. S. (2009). Adaptive generalized predictive control based on JITL technique. *Journal of Process Control*, *19*(7), 1067–1072. https://doi.org/10.1016/j.jprocont.2009.04.002.

Katz, J., Burnak, B., & Pistikopoulos, E. N. (2018). The impact of model approximation in multiparametric model predictive control. *Chemical Engineering Research and Design*, *139*, 211–223. https://doi.org/10.1016/j.cherd.2018.09.034.

Keyser, R. D., & Cauwenberghe, A. V. (1985). Extended prediction self-adaptive control. *IFAC Proceedings Volumes*, *18*(5), 1255–1260. https://doi.org/10.1016/s1474-6670/2817/2960736-5.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, *7*(1), 19–44. https://doi.org/10.1162/evco.1999.7.1.19.

Levin, A., & Narendra, K. (1997). Identification of nonlinear dynamical systems using neural networks. In *Neural systems for control* (pp. 129–160). Elsevier. https://doi.org/10.1016/b978-012526430-3/50007-6.

Li, C., Tan, J., Huo, G., Zhao, T., & Dian, S. (2017). Constrained dynamic matrix control based on improved differential evolution algorithm. In *2017 Chinese automation congress (CAC)* (pp. 2971–2975). IEEE. https://doi.org/10.1109/cac.2017.8243284.

Li, Y., Shen, J., Lee, K. Y., & Liu, X. (2012). Offset-free fuzzy model predictive control of a boiler-turbine system based on genetic algorithm. *Simulation Modelling Practice and Theory*, *26*, 77–95. https://doi.org/10.1016/j.simpat.2012.04.002.

Li, Z., & Wang, G. (2017). Generalized predictive control of linear time-varying systems. *Journal of the Franklin Institute*, *354*(4), 1819–1832. https://doi.org/10.1016/j.jfranklin.2016.10.021.

Lin, F. (2007). *Robust control design: An optimal control approach* (Vol. 18). New York: Wiley.

Liu, G. P., & Kadirkamanathan, V. (1998). Predictive control for non-linear systems using neural networks. *International Journal of Control*, *71*(6), 1119–1132. https://doi.org/10.1080/002071798221515.

Lu, Q., Shi, P., Lam, H. K., & Zhao, Y. (2015). Interval type-2 fuzzy model predictive control of nonlinear networked control systems. *IEEE Transactions on Fuzzy Systems*, *23*(6), 2317–2328. https://doi.org/10.1109/tfuzz.2015.2417975.

Lucia, S., Tatulea-Codrean, A., Schoppmeyer, C., & Engell, S. (2014). An environment for the efficient testing and implementation of robust NMPC. In *2014 IEEE conference on control applications (CCA)* (pp. 1843–1848). IEEE. https://doi.org/10.1109/cca.2014.6981581.

Maciejowski, J. M. (2002). *Predictive control: With constraints*. London: Pearson education.

Mahmoudi, J., Arjomand, M. A., Rezaei, M., & Mohammadi, M. H. (2016). Predicting the earthquake magnitude using the multilayer perceptron neural network with two hidden layers. *Civil Engineering Journal*, *2*(1), 1–12. https://doi.org/10.28991/cej-2016-00000008.

Maner, B., Doyle, F., Ogunnaike, B., & Pearson, R. (1994). A nonlinear model predictive control scheme using second order volterra models. In *Proceedings of 1994 American control conference-ACC'94* (pp. 3253–3257). IEEE. https://doi.org/10.1109/acc.1994.735176.

Mazinan, A. H. (2012). A new algorithm to AI-based predictive control scheme for a distillation column system. *The International Journal of Advanced Manufacturing Technology*, *66*(9–12), 1379–1388. https://doi.org/10.1007/s00170-012-4415-6.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, *4*(1), 1–32. https://doi.org/10.1162/evco.1996.4.1.1.

Mohamed, B., Kara, K., Oussama, A., & Hadjili, L. (2019). Adaptive neural network PID controller. In *2019 IEEE international conference on environment and electrical engineering and 2019 IEEE industrial and commercial power systems Europe (EEEIC / I&CPS Europe)* (pp. 1–6). IEEE. https://doi.org/10.1109/eeeic.2019.8783803.

Mohammadi, J., Ataei, M., Kakaei, R. K., Mikaeil, R., & Haghshenas, S. S. (2018). Prediction of the production rate of chain saw machine using the multilayer perceptron (MLP) neural network. *Civil Engineering Journal*, *4*(7), 1575. https://doi.org/10.28991/cej-0309196.

Narendra, K., & Parthasarathy, K. (1989). Adaptive identification and control of dynamical systems using neural networks. In *Proceedings of the 28th IEEE conference on decision and control* (pp. 1737–1738). IEEE. https://doi.org/10.1109/cdc.1989.70448.

Patan, K. (2018). Two stage neural network modelling for robust model predictive control. *ISA Transactions*, *72*, 56–65. https://doi.org/10.1016/j.isatra.2017.10.011.

Patel, V. K., & Savsani, V. J. (2016). A multi-objective improved teaching-learning based optimization algorithm (MO-ITLBO). *Information Sciences*, *357*, 182–200. https://doi.org/10.1016/j.ins.2014.05.049.

Qin, X., Wang, C., Li, L., Peng, H., & Ye, L. (2020). Finite-time lag synchronization of memristive neural networks with multi-links via adaptive control. *IEEE Access*, *8*, 55398–55410. https://doi.org/10.1109/ACCESS.2020.2977262.

Rao, R., Savsani, V., & Vakharia, D. (2011). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, *43*(3), 303–315. https://doi.org/10.1016/j.cad.2010.12.015.

Rao, R., Savsani, V., & Vakharia, D. (2012). Teaching-learning-based optimization: An optimization method for continuous non-linear large scale problems. *Information Sciences*, *183*(1), 1–15. https://doi.org/10.1016/j.ins.2011.08.006.

Rao, R. V. (2016). *Teaching learning based optimization algorithm*. Berlin: Springer. https://doi.org/10.1007/978-3-319-22732-0.

Rao, R. V., & Waghmare, G. (2014). A comparative study of a teaching-learning-based optimization algorithm on multi-objective unconstrained and constrained functions. *Journal of King Saud University: Computer and Information Sciences*, *26*(3), 332–346. https://doi.org/10.1016/j.jksuci.2013.12.004.

Richalet, J., Rault, A., Testud, J., & Papon, J. (1978). Model predictive heuristic control. *Automatica*, *14*(5), 413–428. https://doi.org/10.1016/0005-1098/2878/2990001-8.

Sarimveis, H., & Bafas, G. (2003). Fuzzy model predictive control of non-linear processes using genetic algorithms. *Fuzzy Sets and Systems*, *139*(1), 59–80. https://doi.org/10.1016/s0165-0114/2802/2900506-7.

Soloway, D., & Haley, P. (1996). Neural generalized predictive control. In *Proceedings of the 1996 IEEE international symposium on intelligent control* (pp. 277–282). IEEE. https://doi.org/10.1109/isic.1996.556214.

Sørensen, P., Nørgaard, M., Ravn, O., & Poulsen, N. (1999). Implementation of neural network based non-linear predictive control. *Neurocomputing*, *28*(1–3), 37–51. https://doi.org/10.1016/s0925-2312/2898/2900114-3.

Subramanian, S., Nazari, S., Alvi, M. A., & Engell, S. (2018). Robust NMPC schemes for the control of mobile robots in the presence of dynamic obstacles. In *2018 23rd international conference on methods & models in automation & robotics (MMAR)* (pp. 768–773). IEEE. https://doi.org/10.1109/mmar.2018.8485841.

Thangavel, S., Lucia, S., Paulen, R., & Engell, S. (2018). Dual robust nonlinear model predictive control: A multi-stage approach. *Journal of Process Control*, *72*, 39–51. https://doi.org/10.1016/j.jprocont.2018.10.003.

Vargas, J. A., Pedrycz, W., & Hemerly, E. M. (2019). Improved learning algorithm for two-layer neural networks for identification of nonlinear systems. *Neurocomputing*, *329*, 86–96. https://doi.org/10.1016/j.neucom.2018.10.008.

Wang, Y., Luo, X., Zhang, F., & Wang, S. (2019). GPU-based model predictive control for continuous casting spray cooling control system using particle swarm optimization. *Control Engineering Practice*, *84*, 349–364. https://doi.org/10.1016/j.conengprac.2018.12.006.

Ydstie, B. (1984). Extended horizon adaptive control. *IFAC Proceedings Volumes*, *17*(2), 911–915. https://doi.org/10.1016/s1474-6670/2817/2961089-9.

Yuan, M., Luo, X., Wang, W., Li, L., & Peng, H. (2019). Pinning synchronization of coupled memristive recurrent neural networks with mixed time-varying delays and perturbations. *Neural Processing Letters*, *49*, 239–262. https://doi.org/10.1007/s11063-018-9811-y.

Zhixiang, H., Hui, C., & Heqing, L. (2008). Neural networks predictive control using AEPSO. In *2008 27th Chinese control conference* (pp. 180–183). IEEE. https://doi.org/10.1109/chicc.2008.4605861.

Zimmer, A., Schmidt, A., Ostfeld, A., & Minsker, B. (2015). Evolutionary algorithm enhancement for model predictive control and real-time decision support. *Environmental Modelling & Software*, *69*, 330–341. https://doi.org/10.1016/j.envsoft.2015.03.005.