ISA Transactions 72 (2018) 161–177

Contents lists available at ScienceDirect

ISA Transactions

journal homepage: www.elsevier.com/locate/isatrans

Research article

Model predictive control for systems with fast dynamics using inverse neural models

Marios Stogiannos [a,b], Alex Alexandridis [a,*], Haralambos Sarimveis [b]

[a] *Department of Electronic Engineering, Technological Educational Institute of Athens, Agiou Spiridonos, Aigaleo 12243, Greece*
[b] *School of Chemical Engineering, National Technical University of Athens, Iroon Polytechneiou 9, Zografos 15780, Greece*

A R T I C L E I N F O

Article history:
Received 6 April 2017
Received in revised form
19 September 2017
Accepted 22 September 2017
Available online 18 October 2017

Keywords:
Applicability domain
Inverse models
Inverted pendulum
Model predictive control
Neural networks
Radial basis function

A B S T R A C T

In this work, a novel model predictive control (MPC) scheme is introduced, by integrating direct and indirect neural control methodologies. The proposed approach makes use of a robust inverse radial basis function (RBF) model taking into account the applicability domain criterion, in order to provide a suitable initial starting point for the optimizer, thus helping to solve the optimization problem faster. The performance of the proposed controller is evaluated on the control of a highly nonlinear system with fast dynamics and compared with different control schemes. Results show that the proposed approach outperforms the rivaling schemes in terms of response; moreover, it solves the optimization problem in less than one sampling period, thus effectively rendering MPC-based controllers capable of handling systems with fast dynamics.

1. Introduction

Artificial neural networks (ANNs or simply NNs) [1] are considered an ideal solution for the modeling of highly nonlinear systems or processes, and during the last twenty years they have been extensively used for the realization of such models [2–5], in order to design novel control schemes. The success of NN-based controllers originates from their inherent ability to model unknown systems or processes by applying specialized training algorithms exclusively on experimental data, providing an outstanding alternative in cases where conventional methods fail to form appropriate control laws.

Among the different NN architectures used to design control schemes, radial basis function neural networks (RBFNNs) [6] present many advantages, including increased accuracy, better interpolation capability, simpler structures and faster training algorithms [7,8]. These particular characteristics have made RBFNNs a preferred choice in formulating state-of-the-art monitoring [9,10] and control [3,5,11] schemes. On the other hand, the main disadvantage of all black-box modeling techniques including NNs, is extrapolation [12], a phenomenon appearing in cases where the training dataset is not sufficiently covering the input space. Extrapolation results in unreliable predictions, which can ultimately lead the system to instability.

There are two main design approaches when it comes to implementing NN-based control strategies, namely direct design and indirect design. In indirect design control techniques the NN acts as a dynamical model of the system, predicting the system state vector and/or outputs [13], whereas in the case of direct design, the NN approximates the inverse dynamics of the system [14] and acts directly as a controller. Indirect design is usually integrated in an appropriate model predictive control (MPC) methodology [15]; in this case an NN-based nonlinear dynamic system model is constructed using historical input-output data, so that receding horizon predictions can be successfully obtained [16–19]. An optimization problem must then be formulated and solved, in order to obtain the optimal series of actions, so as to drive the system to the desired state. MPC techniques can manage MIMO systems, while also taking into account input-output-state restrictions and model-system mismatches; due to these advantages, MPC has found many successful applications in diverse fields [20–22].

Despite all their merits, MPC-based indirect design control methodologies share a significant drawback, namely the restriction that the nonlinear optimization problem must be solved in real-time [15]. The time required to solve the on-line optimization problem must be less than the sampling time period, so that the control scheme has enough time to obtain the control action and apply it to the system. If the solution of the optimization problem requires more time, there is a risk of control failure that may lead to instability, as the optimal value of the manipulated variable will not be computed and applied on time. However, the available time window between two consecutive control steps may not even be

* Corresponding author.
E-mail address: alexx@teiath.gr (A. Alexandridis).

http://dx.doi.org/10.1016/j.isatra.2017.09.016
0019-0578/© 2017 ISA. Published by Elsevier Ltd. All rights reserved.

adequate for solving a sub-optimal MPC [23]. For these reasons, standard MPC methodologies are not applicable to systems with fast dynamics. The explicit MPC [24,25] technique was invented in order to overcome this problem, by partitioning the input space and assigning a different optimal control law in each region. Current explicit MPC theory guarantees global optimality for linear systems [26], but application on nonlinear systems requires approximating techniques [27], which may be even more time-consuming than solving the real-time optimization problem itself. The reduction of the computational cost in MPC methodologies is the prime aspect of current relevant literature [28,29], but, as of this time, fast optimization algorithms [30,31] are limited to linear systems. An interesting alternative for achieving results that can be similar to those produced by fully nonlinear MPC controllers in terms of performance, while at the same time significantly reducing the computational burden, is to linearize online the model around the current operating point, thus resulting to a quadratic optimization problem. Due to its effectiveness and simplicity, the online linearization approach has found many successful industrial applications [32–36]. A different method which eliminates the computational burden associated with solving the optimization problem, involves the NN approximation of the suboptimal control signal of MPC formulations [37,38].

On the other hand, direct design control techniques avoid altogether the optimization problem by using the NN as an explicit control law, directly predicting the manipulated variable values that are used as system inputs at each control step [4], while previous input-output-state values comprise the inverse neural model's input vector. The performance of direct design control methodologies stems from the fact that their implementation is as simple as computing a nonlinear function at each time step, without the hassle of solving an optimization problem. One of the most common direct control schemes are the inverse neural controllers (INCs), which have been extensively used in modern applications, as they are very fast in calculating control actions [4]. Other implementations suggest that model parameter adaptation with offset-free control is also possible in real-time [39]. A more recent study [40] has shown that INCs can be made robust in multiple ways, in order to avoid extrapolation, suppress any steady-state error, reject external disturbances, adapt to unknown system parameters and account for system-model mismatches.

Notwithstanding the aforementioned advancements offered by direct approaches regarding disturbance handling and steady-state error elimination, it should be noted that these methods usually deliver just one feasible trajectory towards the setpoint, totally ignoring the aspect of optimality. Indirect methods are usually better equipped to handle the latter, albeit at the cost of abolishing the obvious advantages in terms of control action calculation speed offered by direct methods. The scope of this work is the formulation of a novel indirect control scheme, in a way that retains all the advantages of nonlinear MPC, while also tackling its main disadvantage, which is the increased time required to solve the optimization problem. In order to do that, an inverse dynamic RBFNN-based system model is built and robustified with the applicability domain (AD) technique, which enhances the prediction reliability of the model [40]. The inverse model is incorporated to the MPC scheme, in order to provide a suitable initial vector to the optimization problem, aiming to decrease the solution time. The proposed controller performance is compared to a direct inverse neural controller employing the applicability domain and an error-correcting technique (INCADEC) [40], a nonlinear MPC controller and a discrete PID (DPID). All control formulations are evaluated on the control of the nonlinear system of the inverted pendulum on cart [41], which is a well-known benchmark for automatic control methodologies sharing a common operating principle with many commercial, industrial and military applications. To perform the evaluation, appropriate control scenarios are used to test set-point tracking and disturbance rejection, the controllers' stand-up and balancing capabilities, as well as their ability to handle noise and system/model mismatches.

The rest of this paper is organized as follows. The next section describes the radial basis function architecture, as well as the selected training algorithm. The third section presents the theory behind the INNEM initialization routine, as well as its incorporation into the MPC framework. Section 4 presents the test cases and thoroughly discusses the results. Finally, the last section provides the concluding remarks produced after the test cases are examined and interpreted.

2. Radial basis function neural network architecture and training techniques

RBF neural networks belong to the feedforward neural network architectures. The main difference of RBF networks compared to the well-known multi-layer perceptron (MLP) architecture is that they employ only one hidden layer, every node of which makes use of a radially symmetrical activation function in order to compute the hidden node response. The advantages of RBFs over other feedforward architectures include simple network structures, fast training algorithms and increased prediction accuracy, but on the side effect their extrapolation ability is smaller. This drawback is alleviated to a great extent by the proposed methodology.

The RBF architecture consists of three layers, namely the input, the hidden and the output layer, as shown in Fig. 1. The input layer distributes data from the N input variables to the L hidden layer nodes. All hidden nodes correspond to a center vector representing the center of the RBF in the input space. In this context, one can see that the hidden layer performs a nonlinear transformation of the input space to a new space of usually higher dimensionality. The input $\mu_l(\mathbf{u}_k)$ to the l-th hidden node (called activity) is a distance metric between the k-th input vector $\mathbf{u}_k$ and the hidden node center vector $\mathbf{c}_l$. The distance metric employed in this work is the Euclidean distance.

$$\mu_l(\mathbf{u}_k) = \|\mathbf{u}_k - \mathbf{c}_l\|_2 = \sqrt{\sum_{i=1}^{N}\left(u_{i,k} - c_{i,l}\right)^2},\ l = 1, 2, \dots, L,\ \ k = 1, 2, \dots, K \tag{1}$$

where K is the number of available training samples, $\mathbf{u}_k = \left[u_{1,k}, u_{2,k}, \dots, u_{N,k}\right]^T$ is the input data vector and $\mathbf{c}_l = \left[c_{1,l}, c_{2,l}, \dots, c_{N,l}\right]^T$ is the node center vector, respectively. The activation function

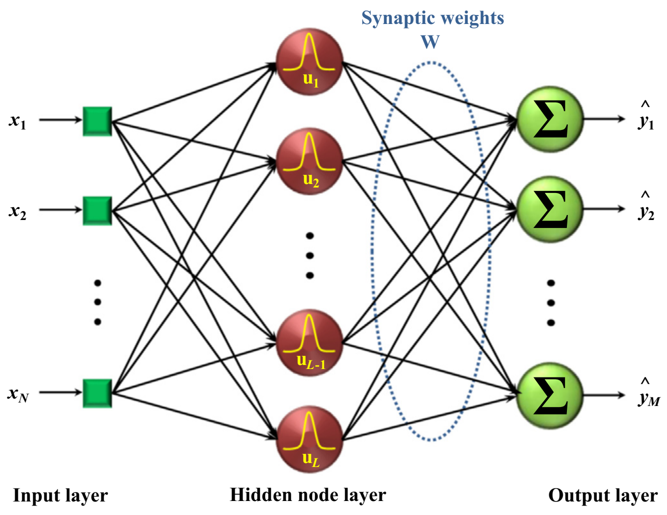


Fig. 1. Typical structure of an RBFNN with Gaussian basis functions.

employed in this work is the Gaussian function, which produces the output of each hidden node:

$$g(\mu_l) = e^{\left(-\frac{\mu_l^2}{\sigma_l^2}\right)} \tag{2}$$

where σ_l is the width of the Gaussian function assigned to the l-th hidden node. In this work we select the Gaussian function among other common choices for RBF activation functions, like the cubic spline or the thin plate spline, as this choice has been shown to lead to models presenting very accurate predictions, small network structures and easily implemented training algorithms [42–45]. The width parameter can be calculated by fairly simple heuristic rules, like the p-nearest neighbor technique [6]. The network's responses $\hat{\mathbf{y}}_k$ for the k-th input data produced by the output layer, are weighted linear combinations of the hidden node outputs:

$$\hat{\mathbf{y}}_k = \mathbf{g}_k \cdot \mathbf{W} \tag{3}$$

where $\mathbf{W}$ is the LxM weight matrix of the network, whereas $\mathbf{g}_k$ is a line vector containing the outputs of the L hidden nodes regarding the k-th input data.

The error minimization methods used for training RBFNNs usually consist of two distinct phases. During the first phase, a training algorithm is used for locating the hidden layer node centers within the given input state-space and for computing the hidden node basis function parameters. Among the various algorithms that have been proposed for the first stage, the fuzzy means (FM) algorithm [46,47] was used in this work, because it is very fast (it needs only one iteration to calculate the number and locations of the hidden node centers), it produces small size models of high accuracy, and it is deterministic, i.e. it avoids random initialization of the network parameters, which is very common among rival algorithms. The main idea behind the FM algorithm is partitioning the input space in fuzzy subspaces; the centers of these subspaces are candidates for populating the hidden layer as RBF centers. The algorithm selects among them a subset which covers sufficiently the input vectors, using a customized distance function. In the second training phase, standard linear regression is applied to obtain the weights w_l connecting the hidden with the output layer.

Model selection is performed through the number of fuzzy sets used for partitioning the input space. To be more specific, the training procedure is applied on the training data, using different numbers of fuzzy sets; the resulting models are then evaluated on a different subset of data, namely the validation set. The best network is finally selected as the model achieving the best result on the validation set. The latter can be determined using various error-related cost functions; in this work, the root mean squared error (RMSE) is applied, due to its popularity in evaluating function approximation problems.

The choice of RBFNNs, in conjunction with the aforementioned advantages provided by the FM training algorithm, contribute to the success of the proposed methodology. Small network sizes result to faster calculation of the predicted future outputs, which increases the speed of the proposed MPC scheme.

3. The proposed methodology

Let us assume a single input – single output (SISO) system with N state variables $x_n(k)$, where $n = 1, 2, \dots, N$, whose dynamics are described by the following nonlinear equation in the discrete state space:

$$y(k+1) = f(\mathbf{x}(k), v(k)) \tag{4}$$

where $\mathbf{x}(k) = \left[\, x_1(k) \;\; x_2(k) \;\; \dots \;\; x_N(k) \,\right]$ is the state vector and $v(k)$ is the manipulated variable.

3.1. MPC methodology based on an RBF prediction model

The following discrete dynamic forward RBF model of the system denoted by $\mathrm{RBF}_{\mathrm{for},y}$ can be used to estimate the next value of the output (controlled) variable $y(k+1)$ as a function of the current state vector $\mathbf{x}(k)$ and the manipulated variable $v(k)$:

$$\hat{y}(k+1) = \mathrm{RBF}_{\mathrm{for},y}(\mathbf{x}(k), v(k)) \tag{5}$$

where $\hat{y}$ are the predictions for the assigned system output and $\mathrm{RBF}_{\mathrm{for},y}$ is an RBF model generated by applying the FM algorithm on historical input – output data, under the assumptions that all state variables are measurable and a sufficient number of training examples is available. Due to the iterative nature of MPC-based algorithms, future states must be predicted as well, so an appropriate model must be obtained for all required states.

$$\hat{\mathbf{x}}(k+1) = \mathrm{RBF}_{\mathrm{for},x}(\mathbf{x}(k), v(k)) \tag{6}$$

where $\mathrm{RBF}_{\mathrm{for},x}$ denotes the forward RBF model producing the $\hat{\mathbf{x}}$ state predictions for all N states, where $\hat{\mathbf{x}} = \left[\, \hat{x}_1 \;\; \hat{x}_2 \;\; \dots \;\; \hat{x}_N \,\right]$ is the state vector prediction.

MPC is based on the formulation of an optimization problem at each discrete time instance k, where a cost function is minimized and the optimum sequence of control moves that drives the controlled variable to the setpoint value is produced. The cost function typically consists of two parts, the first part being the minimization of the difference between the predicted output values $\hat{y}(k+i)$ and the setpoint $\omega(k)$ over the prediction horizon, and the second being the minimization of the control moves over the control horizon:

$$\min_{v(k),\dots,v(k+h_c-1)} \sum_{i=1}^{h_p} \left(\Theta_i \cdot \left(\hat{y}(k+i) - \omega(k) \right) \right)^2 + \sum_{i=0}^{h_c-1} \left(\Omega_i \cdot \Delta v(k+i) \right)^2 \tag{7}$$

where Θ and Ω are the error and move suppression coefficients, h_c and h_p are the control and prediction horizons and Δv is the difference between two subsequent control actions. The minimization problem is solved subject to a set of constraints, the first being the following:

$$\hat{y}(k+i) = \mathrm{RBF}_{\mathrm{for},y}(\mathbf{x}(k+i-1), v(k+i-1)) + E(k),\ 1 \le i \le h_p \tag{8}$$

where $E(k)$ is the modeling error, i.e. the difference of the current measured output from the previous model prediction for the current output, calculated as follows

$$E(k) = y(k) - \mathrm{RBF}_{\mathrm{for},y}(\mathbf{x}(k-1), v(k-1)) \tag{9}$$

The modeling error is assumed to be the same for the entire prediction horizon. The manipulated variable values are bounded by user-defined lower and upper limits $v_{\min}$ and $v_{\max}$ for the entire control horizon.

$$v_{\min} \le v(k+i) \le v_{\max},\ 1 \le i \le h_c \tag{10}$$

There can also be a restriction regarding the difference in the value of two successive actions expressed by:

$$\left| v(k+i) - v(k+i-1) \right| \le v_{\mathrm{deltabound}} \tag{11}$$

Finally, the last manipulated variable value of the control horizon must remain unchanged until the end of the prediction horizon.

$$\Delta v(k+i) = 0,\ h_c + 1 \le i \le h_p \tag{12}$$

The applied constraints may include any kind of operating limitations. The typical MPC formulation for a simple SISO system is depicted in Fig. 2, where it is also shown that the discrete signal generated by the controller can be converted to continuous through a simple zero-order hold element, before entering the system under control:

$$v(t) = v(kT),\ kT \le t < (k+1)T \tag{13}$$

where T is the sampling time.

Closed-loop stability of MPC control schemes which employ RBF models has been rigorously researched and proven. The work of Mason and Kambhampati [48] must be mentioned regarding stability with non-adaptive models, while key research concerning closed-loop stability of MPC controllers with adaptive RBF models was published by Han et al. [49] and Peng et al. [50].

As already mentioned, the aim of the optimization algorithm at each time step k is to produce an optimal series of control actions, denoted by the vector $\mathbf{v}_{\mathrm{opt}}(k)$, by reaching a solution to the problem defined by Eqs. (7)-(12).

$$\mathbf{v}_{\mathrm{opt}}(k) = \left[\, v_{\mathrm{opt},1}(k) \quad v_{\mathrm{opt},2}(k) \quad \dots \quad v_{\mathrm{opt},h_c}(k) \,\right] \tag{14}$$

Iterative optimization techniques make use of a vector $\mathbf{v}_{\mathrm{init}}(k)$, containing initial values for the control moves through the control horizon:

$$\mathbf{v}_{\mathrm{init}}(k) = \left[\, v_{\mathrm{init},1}(k) \quad v_{\mathrm{init},2}(k) \quad \dots \quad v_{\mathrm{init},h_c}(k) \,\right] \tag{15}$$

The choice of $\mathbf{v}_{\mathrm{init}}(k)$ affects the number of iterations needed for the optimization algorithm to converge, and thus, the required computational time. The standard MPC approach used at each time instance k, is to shift the previous optimal vector $\mathbf{v}_{\mathrm{opt}}(k-1)$ by one step and fill the gap of the last value by repeating the penultimate value as follows

$$\mathbf{v}_{\mathrm{init}}(k) = \left[\, v_{\mathrm{opt},2}(k-1) \quad v_{\mathrm{opt},3}(k-1) \quad \dots \quad v_{\mathrm{opt},h_c}(k-1) \quad v_{\mathrm{opt},h_c}(k-1) \,\right],\ k > 1 \tag{16}$$

Fig. 3 shows the conventional initialization method of the optimization problem. In practice, only the first action of each manipulated variable vector is applied to the system and the optimization problem is formulated and solved again during the next discrete time step. Ideally, each subsequent optimal vector would differ from the previous one only with regard to its last position, i.e. the position which was filled-in using the penultimate value. In a real implementation though, two subsequent optimal vectors may be much more different, possibly due to: a) external disturbances that foil the optimality of the previously calculated values of the manipulated variable vector, b) system-model mismatches and c) setpoint changes induced by the user. Depending on the scale of the change, the required time to solve the optimization problem may again increase over the sampling time and result in failure to control the system.

3.2. Inverse model employing the applicability domain

A discrete inverse model [51] is meant to predict a proper manipulated variable value that will drive the system from the current state to a desired setpoint, within one sampling time. In this work, an inverse discrete dynamic RBF model is used to correlate the next system output $y(k+1)$ and the current state vector $\mathbf{x}(k)$ to the current manipulated variable $v(k)$.

$$v(k) = \mathrm{RBF}_{\mathrm{inv}}(\mathbf{x}(k), y(k+1)) \tag{17}$$

where $\mathrm{RBF}_{\mathrm{inv}}$ is the function representing the neural network, which approximates the inverse system dynamics. The value of the next system output $y(k+1)$ is then replaced with the desired setpoint $\omega(k)$ and the previous equation becomes as follows:

$$v(k) = \mathrm{RBF}_{\mathrm{inv}}(\mathbf{x}(k), \omega(k)) \tag{18}$$

Assuming that the state variables are measurable and an adequate volume of training samples is available, the FM algorithm can be employed offline to generate the inverse RBF model.

The trained RBFNN may then receive the current state vector and the desired setpoint to produce a prediction for the current manipulated variable value at each discrete time step. A critical issue to be addressed is the one of extrapolation, i.e. to avoid situations where the inverse model will need to provide predictions in parts of the input space, where the model has not been

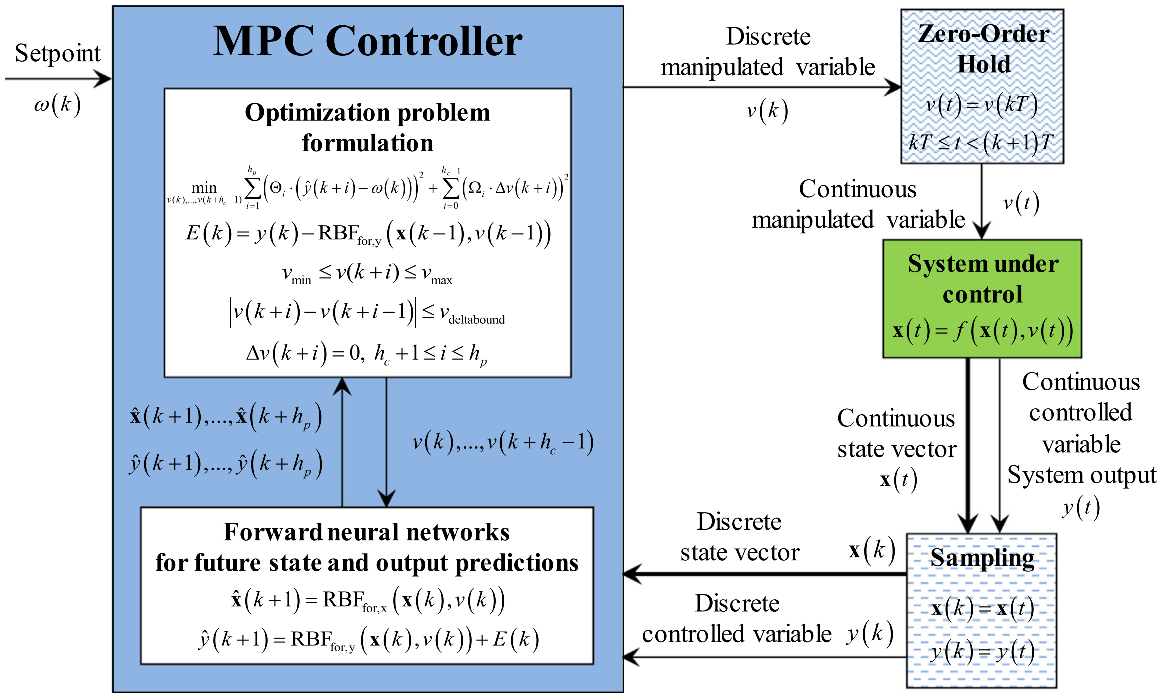

Setpoint
$\omega(k)$
MPC Controller
Optimization problem formulation
$\min_{v(k),\dots,v(k+h_c-1)} \sum_{i=1}^{h_p}\left(\Theta_i\cdot(\hat{y}(k+i)-\omega(k))\right)^2+\sum_{i=0}^{h_c-1}\left(\Omega_i\cdot\Delta v(k+i)\right)^2$
$E(k)=y(k)-\mathrm{RBF}_{\mathrm{for,y}}(\mathbf{x}(k-1),v(k-1))$
$v_{\min}\le v(k+i)\le v_{\max}$
$|v(k+i)-v(k+i-1)|\le v_{\mathrm{deltabound}}$
$\Delta v(k+i)=0,\ h_c+1\le i\le h_p$
$\hat{\mathbf{x}}(k+1),\dots,\hat{\mathbf{x}}(k+h_p)$
$\hat{y}(k+1),\dots,\hat{y}(k+h_p)$
$v(k),\dots,v(k+h_c-1)$
Forward neural networks for future state and output predictions
$\hat{\mathbf{x}}(k+1)=\mathrm{RBF}_{\mathrm{for,x}}(\mathbf{x}(k),v(k))$
$\hat{y}(k+1)=\mathrm{RBF}_{\mathrm{for,y}}(\mathbf{x}(k),v(k))+E(k)$
Discrete manipulated variable
$v(k)$
Zero-Order Hold
$v(t)=v(kT)$
$kT\le t<(k+1)T$
Continuous manipulated variable
$v(t)$
System under control
$\mathbf{x}(t)=f(\mathbf{x}(t),v(t))$
Continuous state vector
$\mathbf{x}(t)$
Continuous controlled variable System output
$y(t)$
Discrete state vector
$\mathbf{x}(k)$
Discrete controlled variable
$y(k)$
Sampling
$\mathbf{x}(k)=\mathbf{x}(t)$
$y(k)=y(t)$

Fig. 2. MPC control scheme employing forward neural networks.

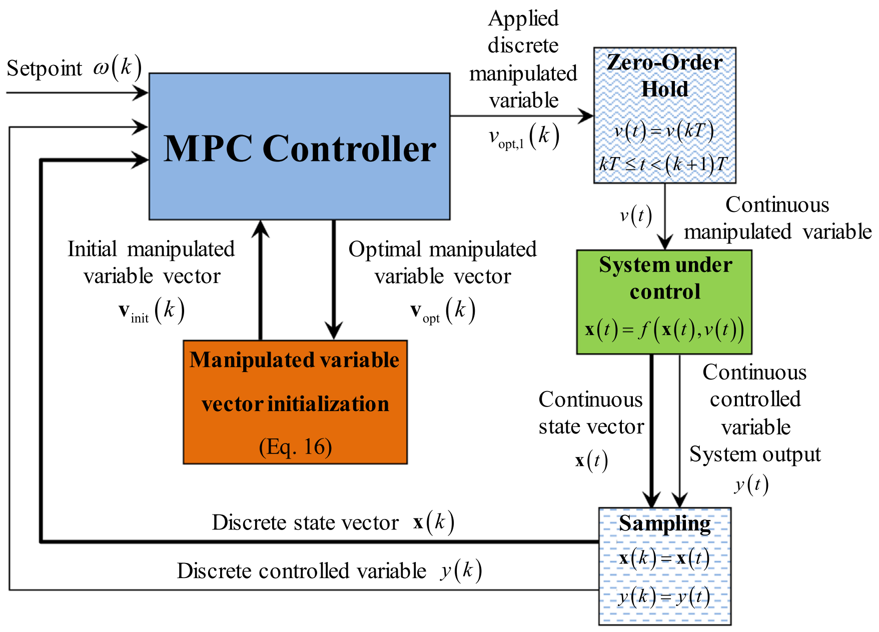

Setpoint $\omega(k)$
MPC Controller
Applied discrete manipulated variable
$v_{\text{opt},1}(k)$
Zero-Order Hold
$v(t)=v(kT)$
$kT \le t < (k+1)T$
$v(t)$
Continuous manipulated variable
Initial manipulated variable vector
$\mathbf{v}_{\text{init}}(k)$
Optimal manipulated variable vector
$\mathbf{v}_{\text{opt}}(k)$
Manipulated variable vector initialization
(Eq. 16)
System under control
$\mathbf{x}(t)=f(\mathbf{x}(t),v(t))$
Continuous state vector
$\mathbf{x}(t)$
Continuous controlled variable System output
$y(t)$
Discrete state vector $\mathbf{x}(k)$
Discrete controlled variable $y(k)$
Sampling
$\mathbf{x}(k)=\mathbf{x}(t)$
$y(k)=y(t)$

Fig. 3. MPC control scheme using conventional initialization.

adequately trained, due to lack of a sufficient number of training data, or because the action produced by the model is not feasible within one sampling time interval [40].

To counter the adverse effects of the extrapolation phenomenon on the model's predictions, the applicability domain (AD) technique [52] is integrated with the inverse model, as described in details in [40]; what follows is a brief description of the method.

An input vector $\mathbf{u}(k) = [\,\mathbf{x}(k) \quad \omega(k)\,]$ is considered to fulfill the AD criterion when the following expression is true:

$$\mathbf{u}(k)\cdot\left(\mathbf{U}^T\cdot\mathbf{U}\right)^{-1}\cdot\mathbf{u}^T(k) \le 3\frac{N+1}{K} \tag{19}$$

where N is the number of input variables, K is the number of training samples and $\mathbf{U}$ is a KxN matrix containing all training samples. With respect to the setpoint, the previous equation is of 2nd order, thus producing two solutions $\omega_{\max}(k)$ and $\omega_{\min}(k)$, which mark the upper and lower bounds of the setpoint value, for which the model is guaranteed not to extrapolate. A graphical example is shown in Fig. 4, where the AD boundary is produced by treating the inequality of Eq. (19) as an equality and solving it for $\omega(k)$, for all the training data.

To take into account inaccuracies present in the training data, a narrowing parameter r is introduced, with the purpose of further restricting the AD limits. The resulting narrowed bounds $\omega'_{\max}(k)$ and $\omega'_{\min}(k)$ are compared with the original setpoint and in case the latter falls out of those two limits, it is substituted by the closest one of the two, thus effectively dealing with the extrapolation problem.

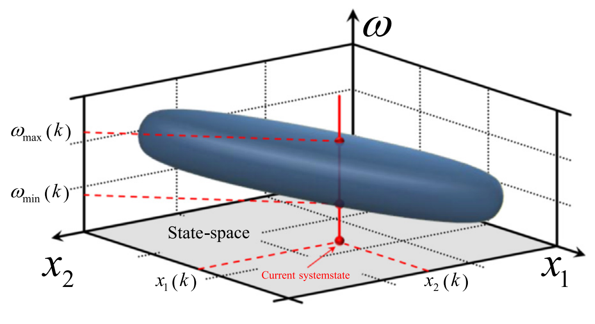


Fig. 4. A 3-D Applicability Domain of a system with 2 inputs.

The resulting modified setpoint value ω_{RE} replaces the original setpoint ω in the RBFNN inverse model of Eq. (18), resulting to the INNEM (inverse neural non-extrapolating model) prediction engine.

This prediction scheme comprised of the inverse model and the AD may be directly used as a controller, with an addition of an error-correcting term, so as to further account for modeling mismatches caused by noise during data collection, insufficient data etc., which can gravely affect the performance of an NN-based controller. The full control scheme is called INCADEC (inverse neural controller using the applicability domain and error correction) and is described in details in [40].

3.3. Integration of INNEM initialization to an MPC framework

This section describes how the performance of the MPC control scheme is accelerated, by integrating it with the aforementioned INNEM initialization routine; the latter helps to provide a starting point $\mathbf{v}_{\text{init}}(k)$ for the MPC minimization problem at each control step k. To be more specific, for each step of the control horizon INNEM provides an estimation of the respective control move, whereas the forward model RBF$_{\text{for}}$ is used to predict the state vector during the next step of the control horizon. The procedure is visualized in Fig. 5, while a detailed description is given in Algorithm 1.

Algorithm 1. – INNEM Initialization Algorithm.

```
Input:   $\mathbf{x}(k) = [\,x_1(k) \quad x_2(k) \quad .. \quad x_N(k)\,]$: current system state
         $\omega(k)$: requested setpoint
         $h_c$: control horizon size
         $r$: AD narrowing parameter
Output:  $\mathbf{v}_{\text{init}}(k) = [\,v_{\text{init},1}(k) \quad v_{\text{init},2}(k) \quad \dots \quad v_{\text{init},h_c}(k)\,]$: initial values vector
1: Set $\hat{\mathbf{x}}(k) = \mathbf{x}(k)$.
2: For $i=1:h_c$ Do:
3:     Calculate a feasible setpoint value $\omega_{\text{RE}}(k+i-1)$, by
       feeding the AD with the predicted system state
       $\hat{\mathbf{x}}(k+i-1)$, the user requested setpoint $\omega(k)$ and the
       narrowing parameter $r$.
4:     Predict a manipulated variable value $\hat{v}(k+i-1)$ that
```

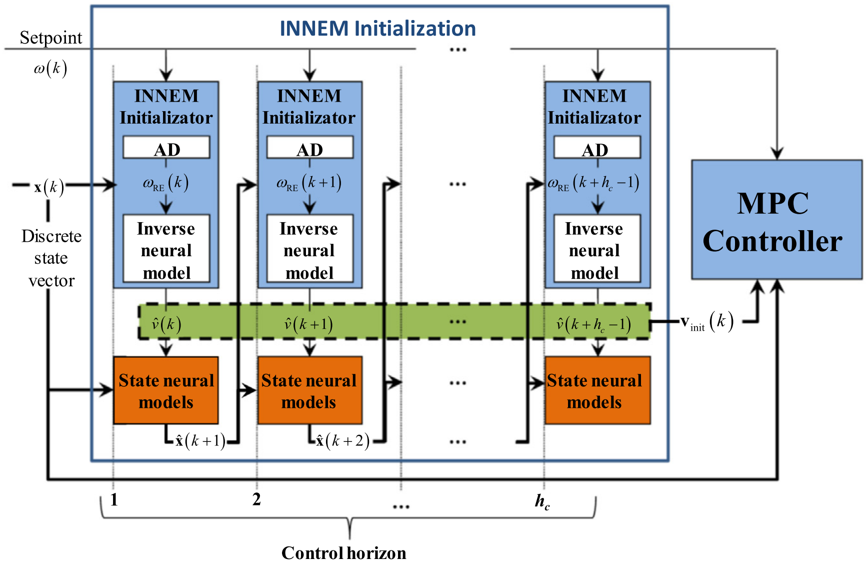


Fig. 5. The INNEM initialization routine.

```
        will drive the system to $\omega_{RE}(k+i-1)$ within one
        sample, by feeding the inverse model $RBF_{inv}$ with
        $\hat{\mathbf{x}}(k+i-1)$ and $\omega_{RE}(k+i-1)$.
5:      Set $v_{init,i}(k) = \hat{v}(k+i-1)$.
6:      Predict future system states $\hat{\mathbf{x}}(k+i)$, by feeding
        $\hat{\mathbf{x}}(k+i-1)$ and $\hat{v}(k+i-1)$ to the forward model
        $RBF_{for,x}$.
7: End For
```

A graphical representation of the MPC controller with the INNEM initialization (MPC-INNEM) is given in Fig. 6.

Remark 1. Integration of the INNEM initialization procedure adds a minimal computational burden, as it involves only a few function calculations. On the other hand, it can induce a significant reduction to the time needed for solving the on-line MPC problem, as it provides the optimizer with an initial sequence of moves that could control the system, taking into account possible disturbances and/or setpoint changes through the inverse dynamic model.

Remark 2. The closed-loop stability of the proposed scheme is not affected by the INNEM initialization routine, since the proposed initial actions vector just improves and accelerates the performance of the MPC controller.

Remark 3. The method is not constrained to using RBFNNs, but can employ any type of forward and inverse models, as long as they perform with sufficient prediction accuracy.

4. Case study: control of a nonlinear inverted pendulum on cart

4.1. The inverted pendulum on cart

The proposed controller's performance was evaluated on the well-known control problem of the inverted pendulum on cart.

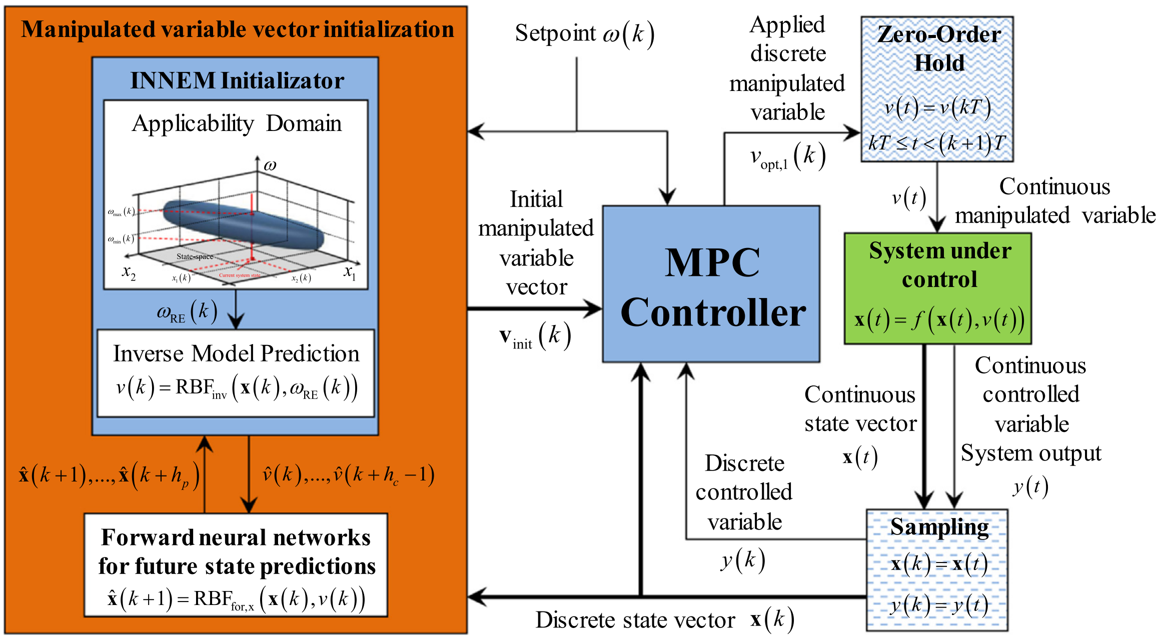

Manipulated variable vector initialization
INNEM Initializator
Applicability Domain
ω
x_2
x_1
State-space
$\omega_{RE}(k)$
Inverse Model Prediction
$v(k) = RBF_{inv}(\mathbf{x}(k), \omega_{RE}(k))$
$\hat{\mathbf{x}}(k+1),...,\hat{\mathbf{x}}(k+h_p)$
$\hat{v}(k),...,\hat{v}(k+h_c-1)$
Forward neural networks for future state predictions
$\hat{\mathbf{x}}(k+1) = RBF_{for,x}(\mathbf{x}(k), v(k))$
Setpoint $\omega(k)$
Initial manipulated variable vector
$\mathbf{v}_{init}(k)$
MPC Controller
Applied discrete manipulated variable
$v_{opt,1}(k)$
Zero-Order Hold
$v(t) = v(kT)$
$kT \le t < (k+1)T$
$v(t)$
Continuous manipulated variable
System under control
$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), v(t))$
Continuous state vector
$\mathbf{x}(t)$
Continuous controlled variable
System output
$y(t)$
Discrete controlled variable
$y(k)$
Sampling
$\mathbf{x}(k) = \mathbf{x}(t)$
$y(k) = y(t)$
Discrete state vector $\mathbf{x}(k)$

Fig. 6. MPC control scheme incorporating the proposed INNEM initialization routine.

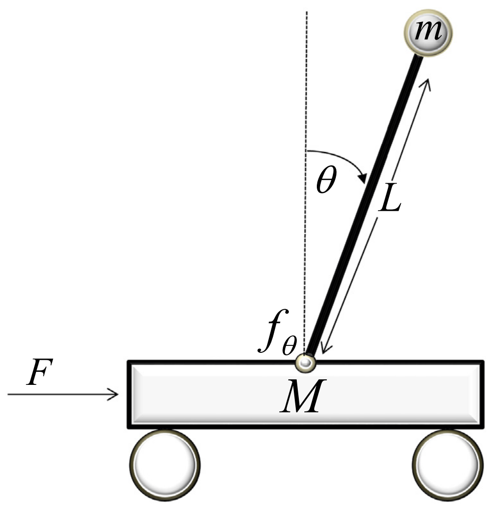


Fig. 7. A typical inverted pendulum on cart.

For comparison purposes, a discrete PID (DPID), an INCADEC and a nonlinear MPC controller were also applied on the same test cases. A typical inverted pendulum is depicted on Fig. 7 and consists of a pole with one end attached to a cart, while the other end is free and has a spherical mass attached on it. The pole is able to perform a 360° rotation. A force F is applied on the cart to move it along only one axis, which also serves as manipulated variable. The pole angular position is the controlled variable. The following state equations, derived by basic physics laws, fully describe the system:

$$\frac{dp}{dt} = v$$
$$\frac{dv}{dt} = \frac{-m\cdot g\cdot \sin(\theta)\cdot \cos(\theta) + m\cdot L\cdot\dot{\theta}^2\cdot \sin(\theta) + f_\theta\cdot m\cdot\dot{\theta} + F}{M + \left(1-\cos^2(\theta)\right)\cdot m}$$
$$\frac{d\theta}{dt} = \dot{\theta}$$
$$\frac{d\dot{\theta}}{dt} = \frac{(M+m)\cdot\left(g\cdot \sin(\theta) - f_\theta\cdot\dot{\theta}\right) - \left(L\cdot m\cdot\dot{\theta}^2 \sin(\theta) + F\right)\cdot \cos(\theta)}{L\cdot\left(M + \left(1-\cos^2(\theta)\right)\cdot m\right)} \tag{20}$$

where p is the cart position, v is the cart velocity, θ is the pole angular position, $\dot{\theta}$ is the pole angular velocity and $\mathbf{x} = \left[x_1 \; x_2 \; x_3 \; x_4\right] = \left[p \; v \; \theta \; \dot{\theta}\right]$ is the state vector. All parameters along with their values are summarized in Table 1. Based on the state equations, the inverted pendulum was simulated, together with the respective controllers, on an Intel Core i7-5820K processor with 16GBs of memory.

Table 1
Inverted pendulum on cart system parameters.

Parameter	Symbol	Description / Value
Cart position	p	State variable (m)
Cart velocity	v	State variable (m/s)
Pole position	θ	State variable / Controlled variable / System output (rad)
Pole velocity	$\dot{\theta}$	State variable (rad/s)
Applied force	F	Manipulated variable /System input (N)
Cart mass	M	1 kg
Pendulum mass	m	0.5 kg
Gravitational acceleration	g	9.81 m/s^2
Pole length	L	0.3 m
Friction coefficient	f_θ	1.5 N/m/s

4.2. RBF model training

A dataset of 1000 datapoints was created by exciting the simulated system with forces between $\pm$ 100 N starting from preset initial states, which covered the entire operating region, and recording the resulting state after one sampling time set at 0.1 s. Each datapoint $\left[\mathbf{x}(k) \;\; F(k)\right]$ is a vector containing the state variables and applied force at time step k. The dataset was split into a training and a validation subset with a 70–30% ratio, so as to train the forward and inverse models, which were integrated in the controllers. The FM algorithm was used for training the models, whereas an exhaustive search procedure was applied for model selection, testing all possible partitions when the number of fuzzy sets ranged within [3 150]. The best model was obtained by evaluating the RMSE performance metric on the validation subset of all FM trained networks and choosing the one with the lowest value.

The forward model formulas shared by all MPC-based neural controllers are of the form:

$$\delta\mathbf{x}(k+1) = \mathrm{RBF_{for}}\left(\mathbf{x}(k), F(k)\right) \tag{21}$$

where $\delta\mathbf{x}(k+1) = \mathbf{x}(k+1) - \mathbf{x}(k)$, i.e. the forward model predicts the difference of the state vector values between two consecutive samples. During the training stage, the inverse model which will be used by the INCADEC controller and the proposed control scheme is given by:

$$F(k) = \mathrm{RBF_{inv}}\left(\mathbf{x}(k), \delta\theta(k+1)\right) \tag{22}$$

where $\delta\theta(k+1) = \theta(k+1) - \theta(k)$, i.e. the difference between two consecutive values of the pole position. However, when the inverse model is used for predictions during the controllers' operation, $\delta\theta(k+1)$ is substituted by $\delta\omega(k) = \omega_{\mathrm{RE}}(k) - \theta(k)$, which represents the difference between the modified setpoint calculated by INNEM and the current value of the pole position:

$$F(k) = \mathrm{RBF_{inv}}\left(\mathbf{x}(k), \delta\omega(k)\right) \tag{23}$$

Some insight within the selected models can be obtained by looking at Figs. 8 and 9, where a random part of the dataset is shown along with the respective model predictions. More specifically, Fig. 8 depicts part of the validation subset of the forward model $\mathrm{RBF_{for}}$, along with the respective predictions, for the pole position state variable $\delta\theta(k)$, while Fig. 9 presents a similar comparison for the inverse model $\mathrm{RBF_{inv}}$. Table 2 sums up the characteristics and performance metrics for the best obtained forward and inverse RBF models, for both training and validation subsets. To justify this choice, Figs. 10 and 11 show the influence of the tested fuzzy sets on the structure (number of hidden nodes) and the quality (RMSE quality metric) of the forward and inverse models, respectively.

4.3. Controller tuning

The DPID controller was tuned by linearizing the system around the unstable equilibrium point, using its respective state equations, and then applying the IMC technique. More specifically, the mean absolute error (MAE) criterion for the actual closed loop was minimized by using a 1st order filter and optimizing the λ parameter [53]. MAE was calculated as follows:

$$\mathrm{MAE} = \frac{\sum_{k=1}^{K_t}\left|\delta\omega(k) - \delta\theta(k+1)\right|}{K_t} \tag{24}$$

where K_t is the number of simulation time steps. The tuning procedure, though, could not get the DPID controller to successfully drive the system in the attempted control tests. Only after

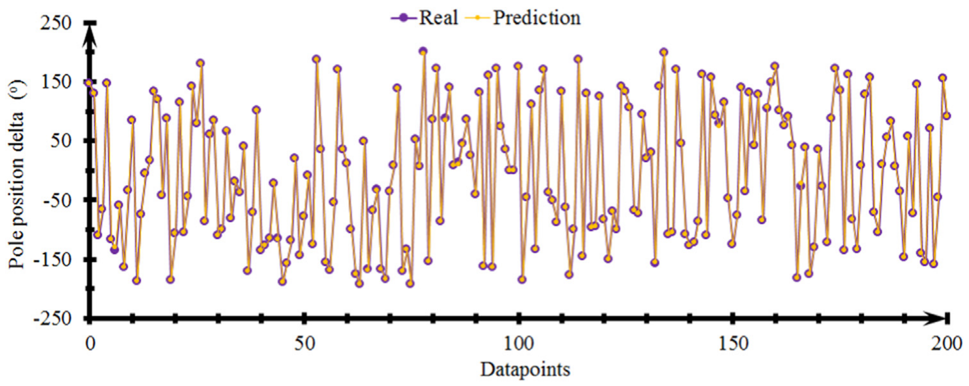


Fig. 8. Real and predicted values for a random part of the validation dataset (forward model).

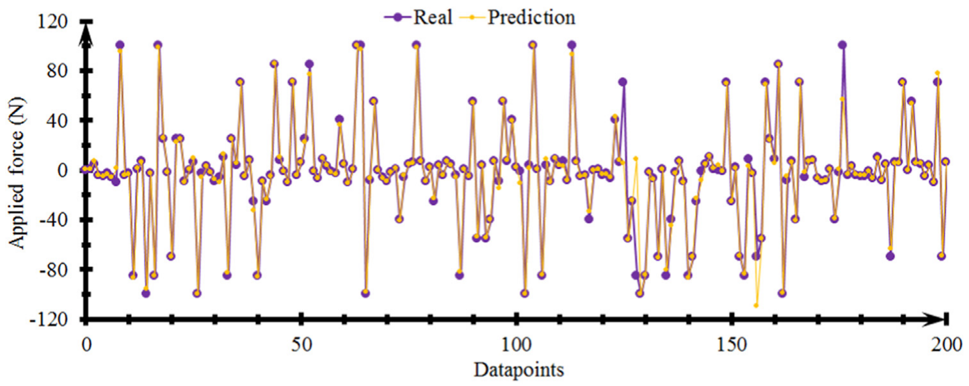


Fig. 9. Real and predicted values for a random part of the validation dataset (inverse model).

Table 2
Model characteristics and performance metrics.

	Forward model Cart velocity	**Forward model Pole position**	**Forward model Pole velocity**	**Inverse model Applied force**
Fuzzy partitioning	[58 58 58 58 58]	[45 45 45 45 45]	[58 58 58 58 58]	[147 147 147 147 147]
# RBF nodes	259	187	259	215
RMSE$_{\text{training}}$	0.01	1.73	0.06	8.45
RMSE$_{\text{validation}}$	0.05	2.63	0.19	9.29

decreasing the sampling time to one tenth, i.e. 0.01 sec, was it possible to get comparable results, without leading the system to instability. This fact must be noted, as it gives the DPID the great advantage of responding to the system dynamics ten times faster than the other controllers.

To this day many techniques have been devised to tune MPC-based controllers properly [54]. These techniques range from simple rules of thumb [55,56], to more complex ones [57]. In our case, tuning of both the MPC and MPC-INNEM controllers, was performed by using a trial and error procedure, where the objective was to achieve the fastest response, while minimizing overshoot. The control horizon was set to the lowest value that produced a satisfactory response, as longer horizons needlessly burden the optimization problem. The prediction horizon was set so as to facilitate the system reaching steady-state after the end of the control horizon. All weight values were determined with the purpose of avoiding oscillations, without hindering the response. Both MPC-based controllers solve the nonlinear optimization problem by using the active-set method [32,44]. Finally, the narrowing parameter r of both the MPC-INNEM and the INCADEC controller was selected by trial and error, with the

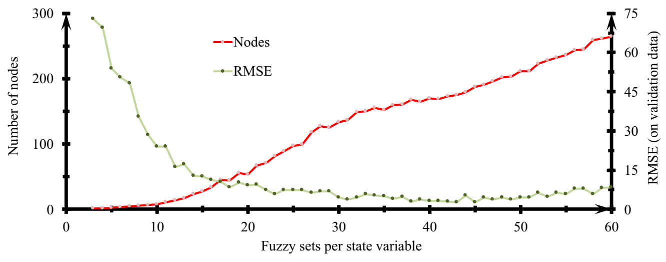


Fig. 10. Number of nodes and RMSE vs. number of fuzzy sets (forward model).

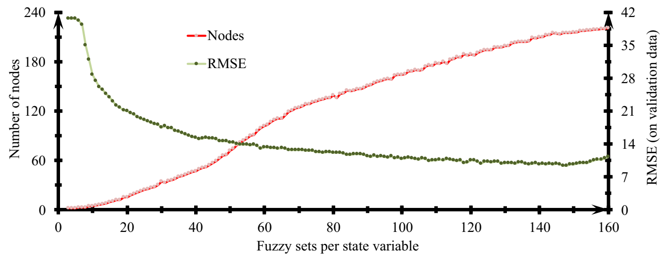


Fig. 11. Number of nodes and RMSE vs. number of fuzzy sets (inverse model).

Table 3
Control schemes tuning parameters.

Parameter	Symbol	Value
All controllers (DPID,INCADEC, MPC, MPC-INNEM)		
Saturation (upper limit)	$F_{\max}$	100 N
Saturation (lower limit)	$F_{\min}$	− 100 N
DPID Controller		
Sampling	T	0.01 s
INCADEC Controller		
Sampling	T	0.1 s
AD Narrowing parameter	r_{INCADEC}	1.6
MPC and MPC-INNEM Controllers		
Sampling	T	0.1 s
Control horizon	h_c	3
Prediction horizon	h_p	5
Error coefficients	Θ	[5.0 4.0 3.0 1.0 0.1]
Move suppression coefficients	Ω	[0.01 0.1 0.1]
MPC-INNEM Controller		
Sampling	T	0.1 s
AD Narrowing parameter	$r_{\text{MPC-INNEM}}$	1.4

objective of minimizing MAE. The tuned parameters used for all controllers are summarized in Table 3.

4.4. Control performance metrics

The control schemes were compared using certain performance metrics, i.e. MAE and the mean absolute steady state error (MASSE), which evaluates the controllers' ability to attain error-free steady state. Furthermore, the controllers' action choices are evaluated using two different metrics conforming to a least-energy type criterion, namely the root sum of the selected actions (RSSA) and the root sum of selected differences between consecutive actions (RSSdA); the former indicates the total energy used by the controller to drive the system throughout each test, whereas the latter indicates the level of wear that the selected actions impose on the system.

$$\text{RSSA} = \sqrt{\sum_{k=1}^{K_t} \left(F(k)\right)^2} \tag{25}$$

$$\text{RSSdA} = \sqrt{\sum_{k=1}^{K_t} \left(F(k) - F(k-1)\right)^2},\ F(0) = 0 \tag{26}$$

where $F(k)$ is the applied action (Force) at the k-th simulation step and K_t is the total number of simulation steps.

Finally, the controllers are also evaluated based on the maximum $T_{\max}$ and the mean T_{mean} duration of the control action calculation.

4.5. Test case 1: Stand-up and balancing

The first case examines two different problems, namely the ability of the controllers to raise the pole over the lower hemisphere (stand-up) and then to drive and stabilize the system to the unstable equilibrium point (balancing). The common method to perform stand-up is to apply random system excitation until the pole passes over the horizontal position to the upper hemisphere and then, allow the controller to take over in order to drive and stabilize the system to the vertical position. The controller responses, actions and action calculation times, regarding the 234° initial pole position, are shown in Fig. 12a–c respectively, while the corresponding performance metrics are summarized in Table 4. It should be noted that all figures regarding action calculation times are presented on a semi-logarithmic scale, for better readability.

As it can be seen in Fig. 12a, the DPID and INCADEC controllers totally fail to reach the stand-up position, as they are trapped right below 90 degrees. The MPC controllers, on the other hand, take advantage of their prediction horizons and form a strategy which manages to deal with the stand-up problem. As far as the action calculation duration is concerned, the nonlinear MPC controller exhibits high computational times, significantly exceeding the sampling time, a fact which would lead to control failure in a real-world problem. The proposed controller on the other hand, is able to solve the optimization problem within the bounds of one sampling time, as it requires much less iterations than its nonlinear counterpart, taking benefit from the INNEM-provided initial point.

Right after that, the controllers are once again put to the test by initiating the system at the lower vertical position of the 180°, which represents the stable equilibrium point. The responses, actions and action calculation times are depicted in Fig. 13a–c respectively, whereas Table 5 shows the resulting performance metrics. The DPID controller once more fails to get past the lower hemisphere. The INCADEC on the other hand, takes advantage of the momentum build-up, which brings the pole over the horizontal axis at a position which allows for a proper thrust reversal and successfully drives the system to the setpoint, although it puts a lot of stress on the actuating subsystems, as indicated by the RSSA and RSSdA metrics. The MPC and MPC-INNEM follow the same strategy, albeit from different directions, due to the different initial solutions presented to the optimization problem at the very first control step. In terms of the MAE and MASSE metrics, the two MPC schemes outperform the other controllers, but are almost equal to one another, since it can be clearly seen that the actions between the two controllers are of same absolute value, but opposite signs. Once again, only the MPC-INNEM control scheme would be able to drive the system successfully in a real-world application, since the time needed to solve the optimization problem is always less than the sampling time, as opposed to the nonlinear MPC.

The initial position of 75° poses an easy problem for all controllers. The DPID manages to reach the setpoint this time, but, even though it implements ten times more actions than the other controllers, it still performs very poorly. The INCADEC controller presents an oscillatory response and reaches the setpoint in more than twice the time needed by the MPC-based controllers, which are faster, require the least energy, and pose the least wear on the system. Although the difference in the response of the two MPC schemes is indiscernible, it should be noted that once more standard MPC would fail in a real-world implementation, as opposed to MPC-INNEM which computes the optimal actions in less than one sampling time. Fig. 14–c show the responses, actions and action calculation times respectively, whereas Table 6 summarizes the resulting performance metrics.

4.6. Test case 2: Setpoint tracking

This case presents the controllers' performance in a setpoint tracking problem, where the goal is to successively stabilize the pole to different setpoint values. The setpoints are chosen so as to cover the whole operating range of the pole, but also to present a high level of difficulty (a) by containing major changes that increase the cart and pole velocities, thus making the system even harder to control, and (b) by including setpoints that are naturally difficult to be attained due to system inertia. The system is initialized at the position of 0° and the controllers are given a time duration of 1 sec to stabilize the pole to each setpoint. The responses, control actions and action calculation times of all controllers are shown in Fig. 15a–c respectively. All performance metrics of this test case are summarized in Table 7.

As seen in Fig. 15a the DPID only manages to drive the system to the first setpoint, but fails all others. Besides that, Fig. 15b shows that the DPID chooses highly oscillatory actions in the beginning of each

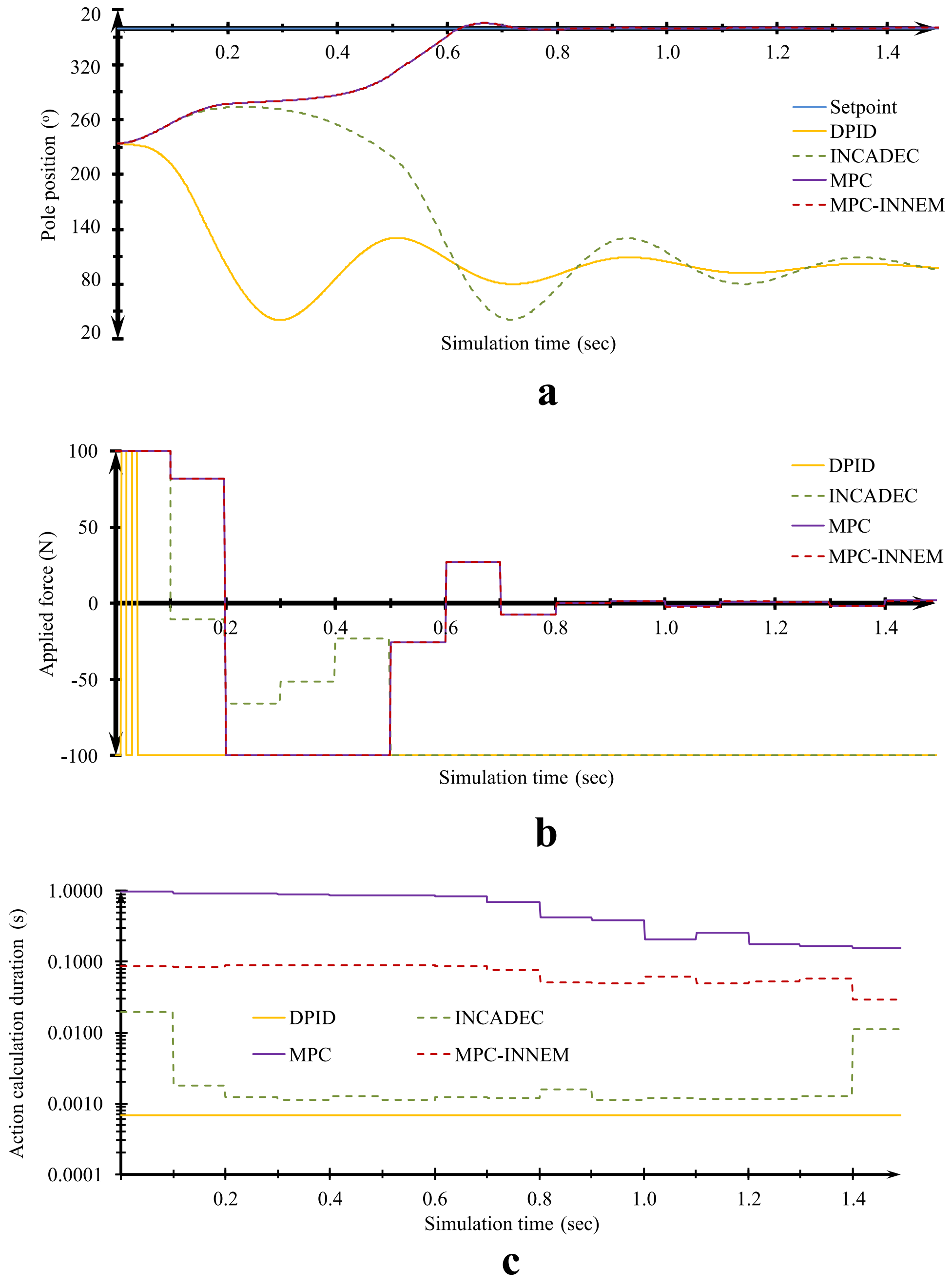


Fig. 12. Test case 1, Stand-up and balancing problem (Initial position of 234°): (a). Controller responses. (b). Control actions. (c). Action calculation durations (Semi-logarithmic scale).

Table 4
Test case 1: Performance metrics for the stand-up and balancing problem (Initial position of 234°).

	MAE (deg)	MASSE (deg)	RSSA (N)	RSSdA (N)	T_{max} (s)	T_{mean} (s)
DPID	100.442	98.371	1224.745	412.311	0.001	0.001
INCADEC	98.905	98.371	342.979	179.372	0.019	0.003
MPC	26.917	0.074	220.877	230.129	0.914	0.588
MPC-INNEM	26.936	0.075	220.874	230.135	0.091	0.072

setpoint change, requiring a great amount of energy and imposing a great wear on the mechanical components of the system, a fact also reflected in the corresponding RSSA and RSSdA metrics.

The INCADEC scheme follows each setpoint change, but most of the time the preset time window is not enough to attain true steady state. This behavior is attributed to the insufficiency of the inverse model training data in conjunction with the choice to keep the r parameter value constant between tests, so that direct comparison of the controller's performance throughout the test cases is possible.

The MPC-based controllers choose an almost identical route of actions for most of the setpoint changes, having the lowest overshoot, the fastest response time, the least energy consumption and posing the least wear on the system compared to their rivals. In the last setpoint change, however, the proposed scheme clearly outperforms the nonlinear MPC controller, due to the superior initialization guesses provided by the INNEM mechanism. Of course, the main advantage of the MPC-INNEM is still the very small time needed to converge to the optimal solution, which is about one order of magnitude lower than its nonlinear counterpart. As it can be seen in

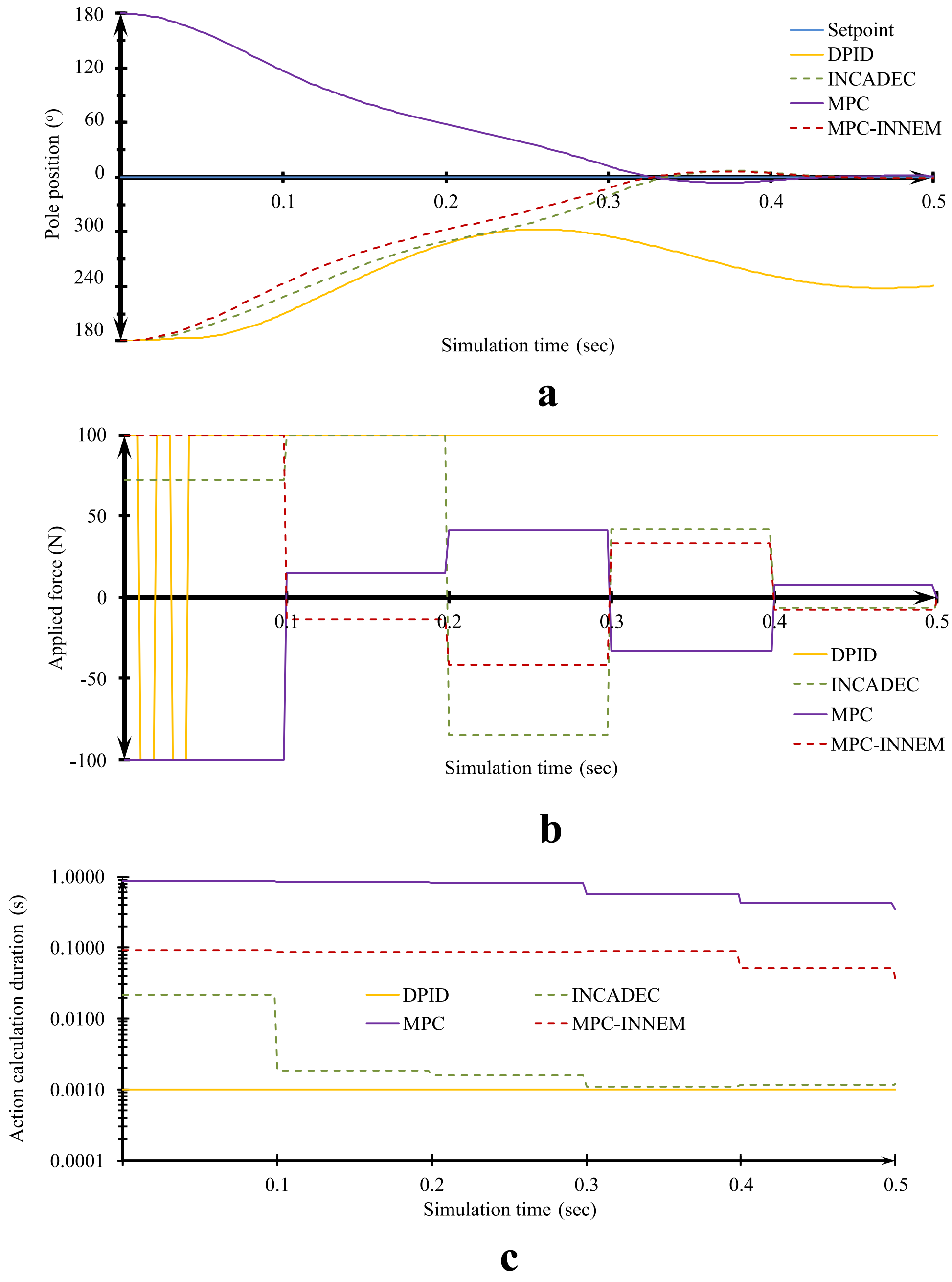


Fig. 13. Test case 1, Stand-up and balancing problem (Initial position of 180°): (a). Controller responses. (b). Control actions. (c). Action calculation durations (Semi-logarithmic scale).

Fig. 15c, the calculation time of the nonlinear MPC is always outside the sampling time limits, especially at the first control step of each setpoint, which foils the previous setpoint action strategy, marking the point between control success and control failure.

Table 5
Test case 1: Performance metrics for the stand-up and balancing problem (Initial position of 180°).

	MAE (deg)	MASSE (deg)	RSSA (N)	RSSdA (N)	T_{max} (s)	T_{mean} (s)
DPID	108.758	98.371	707.107	412.311	0.001	0.001
INCADEC	45.820	0.099	155.650	241.822	0.022	0.005
MPC	38.403	0.093	114.364	176.205	0.964	0.792
MPC-INNEM	38.313	0.090	114.458	176.132	0.097	0.085

4.7. Test case 3: Noise addition

In order to evaluate the controller's ability to handle noisy data, a new simulation is performed based on the previous setpoint tracking case, where the difference is the addition of noise to the state variables which are used as inputs to the RBF models; to be more specific, Gaussian noise $\sim N(0, 3)$ and $\sim N(0, 0.3)$ is added to the position and velocity states, respectively. The responses, control actions and action calculation times of all controllers are shown in Fig. 16a–c respectively. The respective performance metrics are summarized in Table 8.

Fig. 16a shows a similar to the nominal simulation response of the DPID controller, i.e. it fails after the initial setpoint change. The INCADEC controller is greatly affected by noise, as the oscillations are more intense compared to the original response (Fig. 15a), to the point where the system loses control after the second setpoint change. The

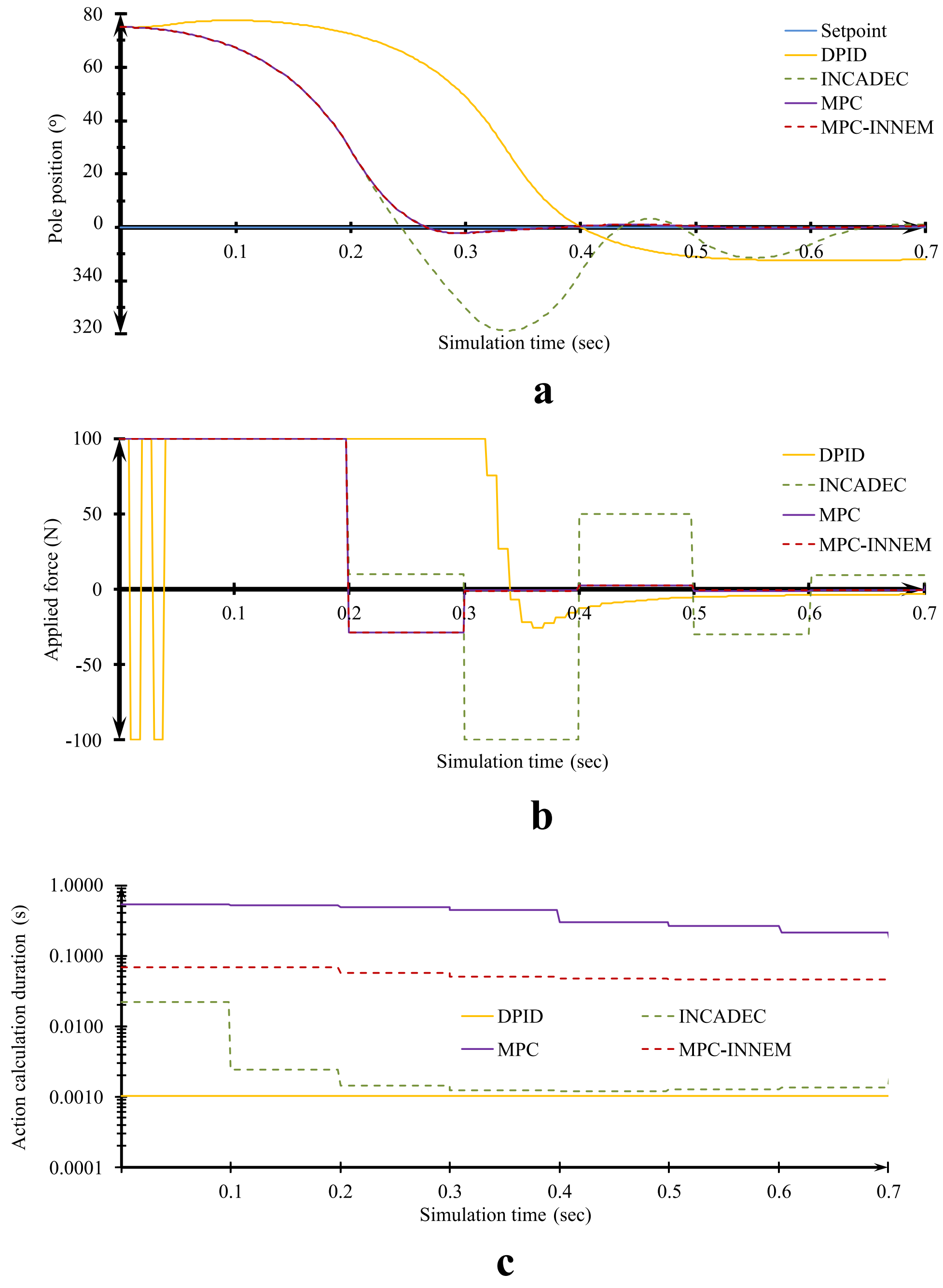


Fig. 14. Test case 1, Stand-up and balancing problem (Initial position of 75°): (a). Controller responses. (b). Control actions. (c). Action calculation durations (Semi-logarithmic scale).

Table 6
Test case 1: Performance metrics for the stand-up and balancing problem (Initial position of 75°).

	MAE (deg)	MASSE (deg)	RSSA (N)	RSSdA (N)	T_{max} (s)	T_{mean} (s)
DPID	37.930	0.000	574.308	417.641	0.001	0.001
INCADEC	22.056	0.084	185.358	242.901	0.022	0.004
MPC	14.259	0.062	144.399	165.703	0.535	0.403
MPC-INNEM	14.288	0.066	144.363	165.482	0.069	0.055

MPC-INNEM scheme is tracking successfully all setpoint changes, but standard nonlinear MPC is not able to track the last change.

The main advantage of the MPC-INNEM, i.e. fast convergence, is validated in Fig. 16c, which clearly shows that the INNEM initialization routine helps keeping the action calculation time lower than the sampling time. On the other hand, the standard nonlinear MPC performance deteriorates and in fact, the corresponding performance metrics in Table 8 show that the mean action calculation time T_{mean} has almost doubled, while T_{max} has reached a value which is 19 times the sampling time, both of which would lead to a total control failure in a real-world application.

4.8. Test case 4: System-model mismatches

This test case evaluates the controllers' ability to handle mismatches between the system and the RBF models. Such mismatches could occur when the system dynamics change after the predictive

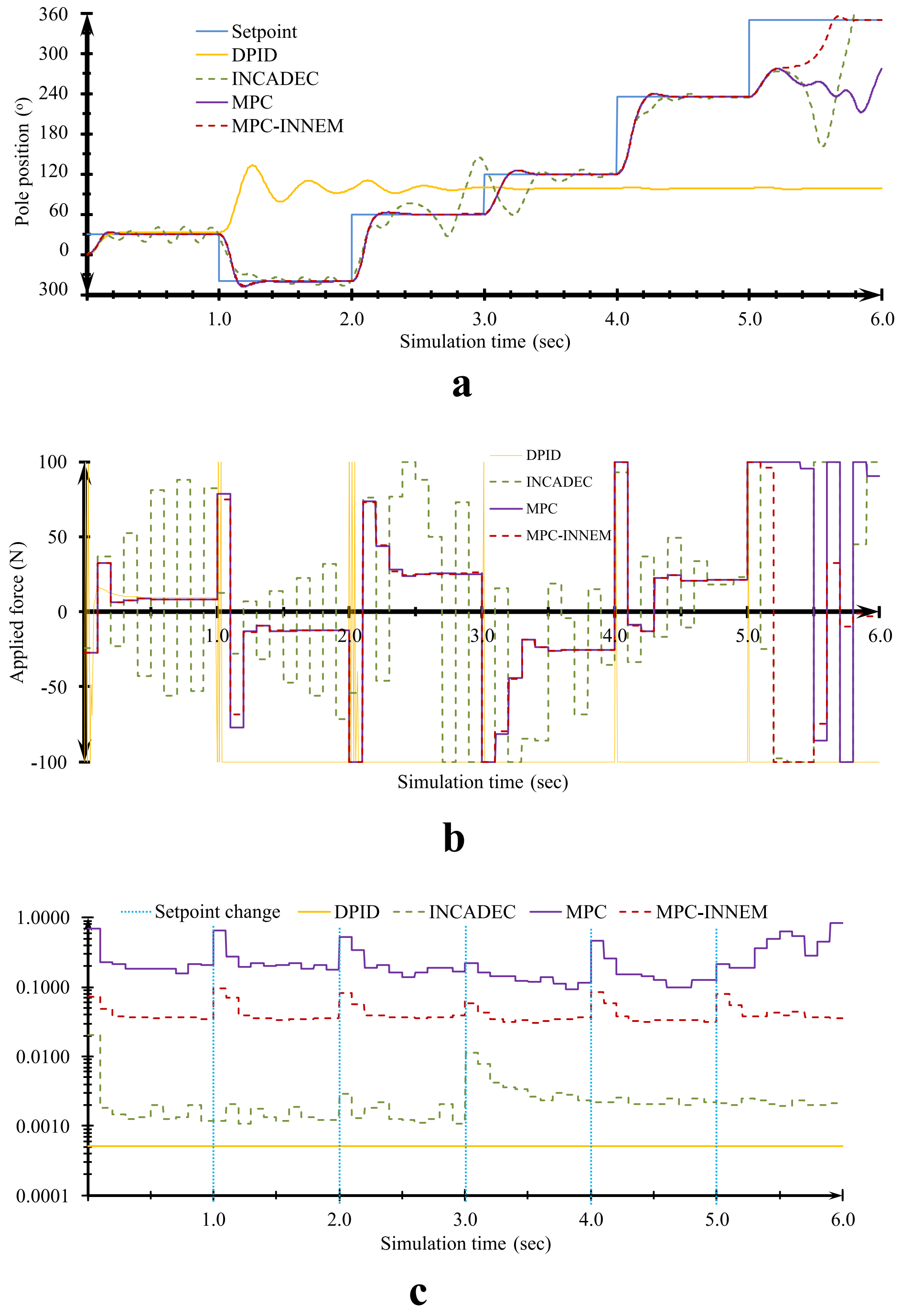


Fig. 15. Test case 2, Setpoint tracking: (a). Controller responses. (b). Control actions. (c). Action calculation durations (Semi-logarithmic scale).

Table 7
Test case 2: Performance metrics for the setpoint tracking problem.

	MAE (deg)	MASSE (deg)	RSSA (N)	RSSdA (N)	T_{max} (s)	T_{mean} (s)
DPID	74.141	74.305	2249.157	914.925	0.001	0.001
INCADEC	27.017	42.855	497.217	721.097	0.021	0.003
MPC	18.177	12.054	401.308	527.305	0.820	0.266
MPC-INNEM	9.049	0.194	357.954	422.515	0.096	0.044

models have been trained. In this simulation, the original mass of the spherical ball attached to one end of the pole is changed by a factor of 20% and, more specifically, from 1.00 kg to 1.2 kg. The controller responses, actions, and action calculation times are shown in Fig. 17a–c, respectively, while the corresponding performance metrics are summarized in Table 9.

The DPID controller once more fails after the first setpoint change, so its performance is not further evaluated. The INCADEC controller successfully drives the system to the last setpoint change, where it can no longer control the system. Its response is becoming more oscillatory though, obviously due to the system/model mismatch. The proposed MPC-INNEM scheme handles successfully the system/model mismatches, and manages to drive the pendulum through all setpoint changes; furthermore, it proves to be superior when compared to standard nonlinear MPC, in terms of improved response and faster action calculation times, as shown in Table 9.

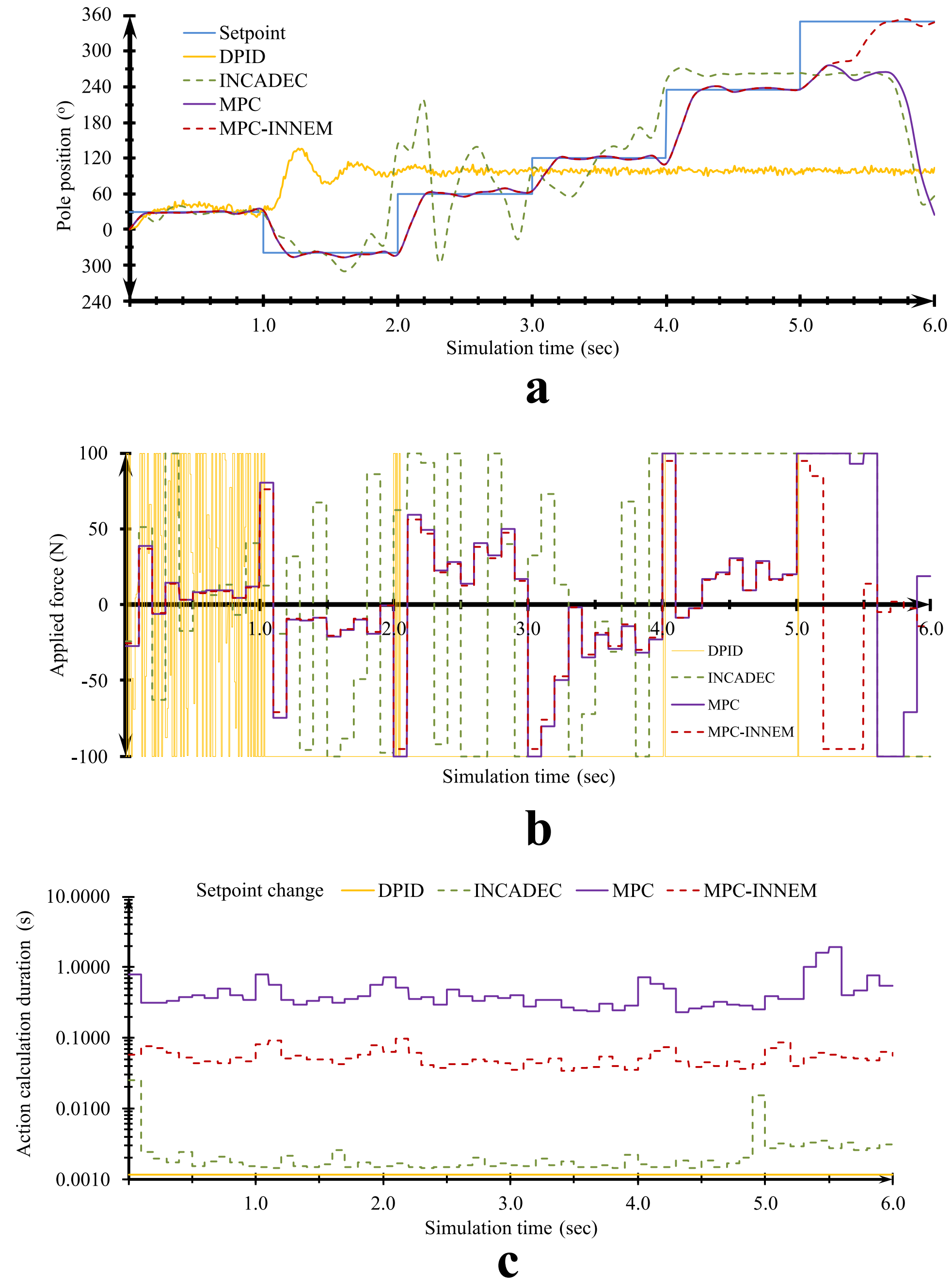


Fig. 16. Test case 3, Noise addition: (a). Controller responses. (b). Control actions. (c). Action calculation durations (Semi-logarithmic scale).

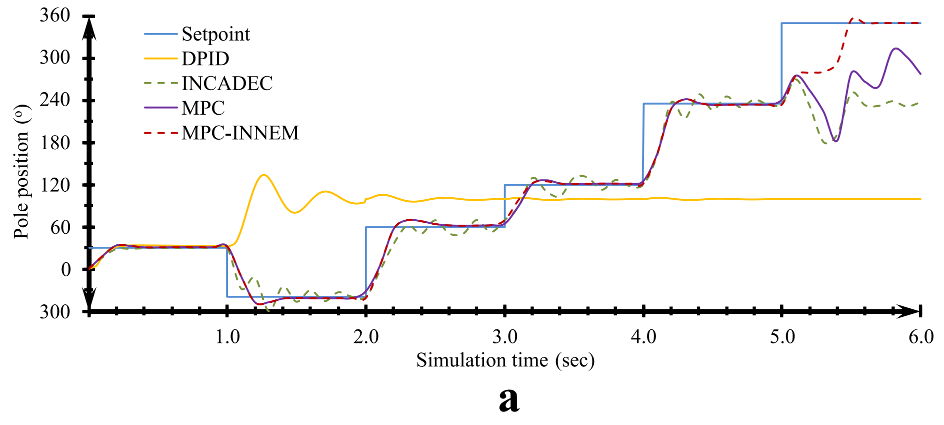
Table 8
Test case 3: Performance metrics for the noise addition problem.

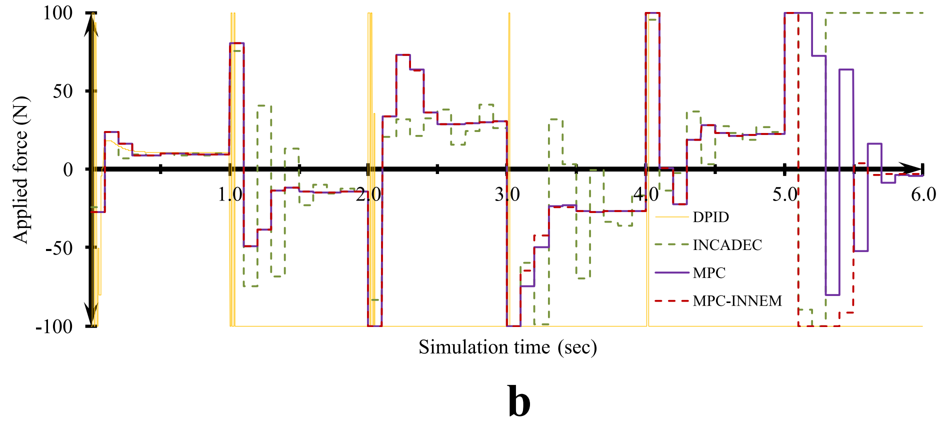
	MAE (deg)	MASSE (deg)	RSSA (N)	RSSdA (N)	T_{max} (s)	T_{mean} (s)
DPID	74.479	75.146	2395.599	1672.543	0.001	0.001
INCADEC	44.916	74.235	634.343	743.698	0.025	0.003
MPC	20.046	8.823	399.750	427.943	1.945	0.452
MPC-INNEM	10.911	3.300	334.979	407.550	0.098	0.053

4.9. Test case 5: External disturbance rejection

This control problem evaluates the ability of the proposed controller in maintaining balance, after an unmeasured external disturbance in the applied force destabilizes the system. The initial pole position is set on the unstable equilibrium point at 0°, a value which serves as the setpoint for the entire simulation. Each disturbance has a spiking nature, in the sense that it is directly added to the computed action of the corresponding control step and it lasts for 0.1 sec. All controllers are given a 1 sec time period to balance the system back at the unstable equilibrium, before the next disturbance comes. The controller responses, control actions and action calculation times are depicted in Figs. 18a–c, respectively. The performance metrics corresponding to this test case are given in Table 10.

The DPID controller successfully counters only the first and smallest disturbance, because of the very small sampling time it uses, which allows for an action before the pole reaches a critical position close to the lower hemisphere. The other two disturbances are too high for the DPID to reject, even with the higher





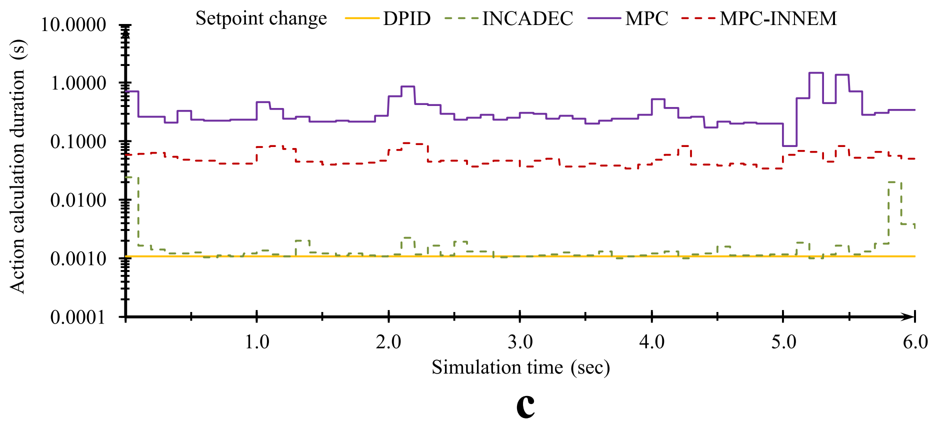


Fig. 17. Test case 4, System-model mismatches: (a). Controller responses. (b). Control actions. (c). Action calculation durations (Semi-logarithmic scale).

Table 9
Test case 4: Performance metrics for the system-model mismatches problem.

	MAE (deg)	MASSE (deg)	RSSA (N)	RSSdA (N)	T_{max} (s)	T_{mean} (s)
DPID	74.178	74.495	2249.866	838.489	0.001	0.001
INCADEC	28.735	20.545	423.405	489.377	0.024	0.002
MPC	19.988	16.428	340.314	413.069	1.511	0.344
MPC-INNEM	9.760	2.374	351.389	395.377	0.093	0.051

sampling rate advantage. The INCADEC controller almost succeeds in rejecting the first disturbance, but fails to do so for the second and third ones.

Both the MPC and MPC-INNEM controllers successfully reject all the disturbances, notwithstanding a different choice in the route of actions. The nonlinear MPC takes advantage of the system momentum and choses very energy-efficient control actions in order to drive the pole to complete a full rotation before finally balancing it on the original position. The MPC-INNEM control scheme on the other hand, chooses to counter the disturbance momentum and bring the system back to the set-point in about the same time as the nonlinear MPC, but with slightly more energy-consuming actions. The fact remains that the MPC-INNEM calculates feasible actions within the duration of one sample, while the nonlinear MPC would fail to apply its more optimal course, due to the calculation time being far over the actual sampling time.

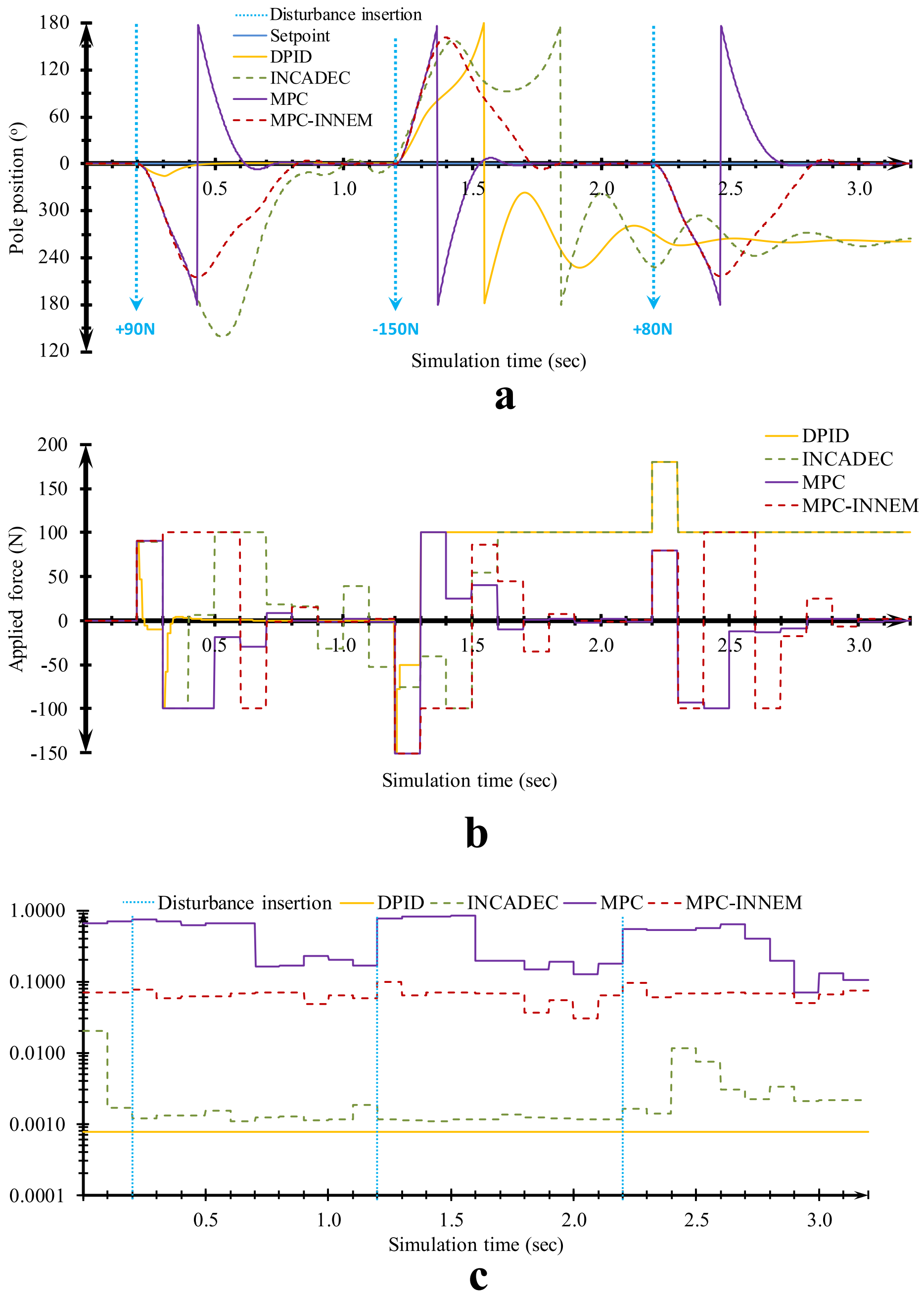


Fig. 18. Test case 5, External disturbance rejection: (a). Controller responses. (b). Control actions. (c). Action calculation durations (Semi-logarithmic scale).

Table 10
Test case 5: Performance metrics for the external unmeasured disturbance rejection problem.

	MAE (deg)	MASSE (deg)	RSSA (N)	RSSdA (N)	T_{max} (s)	T_{mean} (s)
DPID	59.265	62.570	1482.216	295.759	0.001	0.001
INCADEC	82.884	77.289	500.972	359.075	0.020	0.003
MPC	27.625	0.488	260.097	339.891	0.937	0.463
MPC-INNEM	42.434	0.332	369.281	491.505	0.090	0.062

5. Conclusions

In this work, a novel automatic control scheme for nonlinear systems has been developed. The proposed MPC-INNEM controller integrates the novel INNEM initialization methodology, which speeds up the solution of the online optimization problem and, ultimately, makes it possible for MPC-based control schemes to be applied to systems with fast dynamics, or cases where very small sampling times are required. The INNEM initialization routine alternatively employs forward and inverse RBFNN models of the system, in order to build a suboptimal initial guess from where the MPC optimizer starts the search for the optimal solution.

The controller performance has been evaluated on a highly nonlinear system with fast dynamics and an unstable equilibrium point, namely the inverted pendulum on cart. The proposed MPC-INNEM control scheme was shown to decrease calculation time for solving the nonlinear MPC optimization problem to levels lower than the sampling time. A comparison between three different control schemes highlights the superiority of the proposed method in terms of response, as well as alternative performance metrics including control action economy and imposed wear on the mechanical parts of the system.

References

[1] Haykin S. Neural networks: a comprehensive foundation. 2nd ed. Upper Saddle River, NJ: Prentice Hall International; 1999.
[2] Alexandridis A, Sarimveis H. Nonlinear adaptive model predictive control based on self-correcting neural network models. AIChE J 2005;51(09):2495–506.
[3] Aggelogiannaki E, Sarimveis H. A simulated annealing algorithm for prioritized multiobjective optimization - implementation in an adaptive model predictive control configuration. IEEE Trans Syst, Man, Cybern, Part B: Cybern 2007;37:902–15.
[4] Ninos K, Giannakakis C, Kompogiannis I, Stavrakas I, Alexandridis A. , Nonlinear control of a DC-motor based on radial basis function neural networks. In: Proceedings of the international symposium on innovations in intelligent systems and applications (INISTA), Istanbul, Turkey; 2011. p. 611–15.
[5] Qi Z, Peng S, Honghai L, Shengyuan X. Neural-network-based decentralized adaptive output-feedback control for large-scale stochastic nonlinear systems. IEEE Trans Syst, Man, Cybern, Part B: Cybern 2012;42:1608–19.
[6] Moody J, Darken C. Fast learning in networks of locally-tuned processing units. Neural Comput 1989;1:281–94.
[7] Alexandridis A, Chondrodima E, Sarimveis H. Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization. IEEE Trans Neural Netw Learn Syst 2013;24:219–30.
[8] Alexandridis AA, Chondrodima E, Giannopoulos N, Sarimveis H. A fast and efficient method for training categorical radial basis function networks. IEEE Trans Neural Netw Learn Syst 2017, (in press), http://dx.doi.org/10.1109/TNNLS.2016.2598722.
[9] Iliyas SA, Elshafei M, Habib MA, Adeniran AA. RBF neural network inferential sensor for process emission monitoring. Control Eng Pract 2013;21:962–70.
[10] Pani AK, Mohanta HK. Online monitoring of cement clinker quality using multivariate statistics and Takagi-Sugeno fuzzy-inference technique. Control Eng Pract 2016;57:1–17.
[11] Schauer T, Negård NO, Previdi F, Hunt KJ, Fraser MH, Ferchland E, et al. Online identification and nonlinear control of the electrically stimulated quadriceps muscle. Control Eng Pract 2005;13:1207–19.
[12] Haley PJ, Soloway D. Extrapolation limitations of multilayer feedforward neural networks. In: Proceedngs of the international joint conference on neural networks (IJCNN), Baltimore, MD, USA; 1992. p. 25–30.
[13] Henriques J, Gil P, Cardoso A, Carvalho P, Dourado A. Adaptive neural output regulation control of a solar power plant. Control Eng Pract 2010;18:1183–96.
[14] Govindhasamy JJ, McLoone SF, Irwin GW, French JJ, Doyle RP. Neural modelling, control and optimisation of an industrial grinding process. Control Eng Pract 2005;13:1243–58.
[15] Lee JH. Model predictive control: review of the three decades of development. Int J Control Autom Syst 2011;9(01):415–24.
[16] Polycarpou MM, Conway JY. Indirect adaptive nonlinear control of drug delivery systems. IEEE Trans Autom Control 1998;43:849–56.
[17] Ławryńczuk M, Tatjewski P. Nonlinear predictive control based on neural multi-models,. Int J Appl Math Comput Sci 2010;20:7–21.
[18] Camacho EF, Arahal MR. Neural network based adaptive control,. Annu Rev Autom Program 1994;19:13–24.
[19] Declercq F, Keyser R.De. Comparative study of neural predictors in model based predictive control. In: Proceedings of the international workshop on, neural networks for identification, control, robotics, and signal/image processing; 1996. p. 20–28.
[20] Liu S, Shah N, Papageorgiou LG. Multiechelon supply chain planning with sequence-dependent changeovers and price elasticity of demand under uncertainty. AIChE J 2012;58:3390–403.
[21] Sørensen KK, Stoustrup J, Bak T. Adaptive MPC for a reefer container,. Control Eng Pract 2015;44:55–64.
[22] Mahmoud MS, Sabih M, Elshafei M. Using OPC technology to support the study of advanced process control. ISA Trans 2015;55(1):155–67.
[23] Graichen K, Kugi A. Stability and incremental improvement of suboptimal MPC without terminal constraints. IEEE Trans Autom Control 2010;55:2576–80.
[24] Bemporad A, Morari M, Dua V, Pistikopoulos EN. The explicit linear quadratic regulator for constrained systems. Automatica 2002;38:3–20.
[25] Suardi A, Longo S, Kerrigan EC, Constantinides GA. Explicit MPC: hard constraint satisfaction under low precision arithmetic. Control Eng Pract 2016;47:60–9.
[26] Patrinos P, Sarimveis H. A new algorithm for solving convex parametric quadratic programs based on graphical derivatives of solution mappings. Automatica 2010;46:1405–18.
[27] Grancharova A, Johansen TA. Computation, approximation and stability of explicit feedback min–max nonlinear model predictive control. Automatica 2009;45:1134–43.
[28] Aggelogiannaki E, Sarimveis H. Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. Comput Chem Eng 2008;32:1225–37.
[29] Ławryńczuk M. Accuracy and computational efficiency of suboptimal nonlinear predictive control based on neural models. Appl Soft Comput 2011;11:2202–15.
[30] Wang Y, Boyd S. Fast model predictive control using on-line optimization. IEEE Trans Autom Control 2010;18:267–78.
[31] Patrinos P, Sopasakis P, Sarimveis H. A global piecewise smooth Newton method for fast large-scale model predictive control. Automatica 2011;47:2016–22.
[32] Ławryńczuk M. Nonlinear predictive control of a boiler-turbine unit: a state-space approach with successive on-line model linearisation and quadratic optimisation. ISA Trans 2017;67(1):476–95.
[33] Ławryńczuk M. Nonlinear predictive control of dynamic systems represented by Wiener–Hammerstein models. Nonlinear Dyn 2016;86:1193–214.
[34] Peyrl H, Zanarini A, Besselmann T, Liu J, Boéchat M-A. Parallel implementations of the fast gradient method for high-speed MPC. Control Eng Pract 2014;33:22–34.
[35] Ławryńczuk M. Online set-point optimisation cooperating with predictive control of a yeast fermentation process: a neural network approach. Eng Appl Artif Intell 2011;24:968–82.
[36] Saraswati S, Chand S. Online linearization-based neural predictive control of air-fuel ratio in SI engines with PID feedback correction scheme,. Neural Comput Appl 2010;19:919–33.
[37] Parisini T, Zoppoli R. A receding-horizon regulator for nonlinear systems and a neural approximation. Automatica 1995;31:1443–51.
[38] Åkesson BM, Toivonen HT. A neural network model predictive controller. J Process Control 2006;16:937–46.
[39] Waegeman T, Wyffels F, Schrauwen B. Feedback control by online learning an inverse model. IEEE Trans Neural Netw Learn Syst 2012;23:1637–48.
[40] Alexandridis A, Stogiannos M, Kyriou A, Sarimveis H. An offset-free neural controller based on a non-extrapolating scheme for approximating the inverse process dynamics. J Process Control 2013;23:968–79.
[41] Nundrakwang S, Benjanarasuth T, Ngamwiwit J, Komine N. Hybrid controller for swinging up inverted pendulum system. In: Proceedings of the 5th international conference on information communications & signal processing; 2005. p. 488–92.
[42] Han H-G, Qiao J-F, Chen Q-L. Model predictive control of dissolved oxygen concentration based on a self-organizing RBF neural network. Control Eng Pract 2012;20:465–76.
[43] Salahshoor K, Zakeri S, Haghighat Sefat M. Stabilization of gas-lift oil wells by a nonlinear model predictive control scheme based on adaptive neural network models. Eng Appl Artif Intell 2013;26:1902–10.
[44] Bhartiya S, Whiteley JR. Benefits of factorized RBF-based NMPC,. Comput Chem Eng 2002;26(15):1185–99.
[45] Engell S, Fernholz G. Control of a reactive separation process,. Chem Eng Process: Process Intensif 2003;42:201–10.
[46] Sarimveis H, Alexandridis A, Tsekouras G, Bafas G. A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space. Ind Eng Chem Res 2002;41:751–9.
[47] Alexandridis A, Sarimveis H, Bafas G. A new algorithm for online structure and parameter adaptation of RBF networks. Neural Netw 2003;16:1003–17.
[48] Mason J and Kambhampati C. Predictive control of a mixing tank using radial basis function networks. In: Proceedings of the 35th IEEE conference on decision and control; 1996. p. 478–79.
[49] Han H-G, Qian H-H, Qiao J-F. Nonlinear multiobjective model-predictive control scheme for wastewater treatment process. J Process Control 2014;24:47–59.
[50] Peng H, Nakano K, Shioya H. Nonlinear predictive control using neural nets-based local linearization ARX model—stability and industrial application. IEEE Trans Control Syst Technol 2007;15:130–43.
[51] Shin YC, Xu C. Intelligent systems: modeling, optimization, and control. CRC Press, Boca Raton, FL; 2008.
[52] Sahigara F, Mansouri K, Ballabio D, Mauri A, Consonni V, Todeschini R. Comparison of different approaches to define the applicability domain of QSAR models. Molecules 2012;17:4791–810.
[53] Seborg DE, Edgar TF, Mellichamp DA. Process dynamics and control. 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc; 2004.
[54] Garriga JL, Soroush M. Model predictive control tuning methods: a review. Ind Eng Chem Res 2010;49:3505–15.
[55] Rani KY, Unbehauen H. Study of predictive controller tuning methods,. Automatica 1997;33:2243–8.
[56] Wojsznis W, Gudaz J, Blevins T, Mehta A. Practical approach to tuning MPC. ISA Trans 2003;42:149–62.
[57] Shridhar R, Cooper DJ. A tuning strategy for unconstrained multivariable model predictive control. Ind Eng Chem Res 1998;37:4003–16.