Research Article

Fault compensation by online updating of genetic algorithm-selected neural network model for model predictive control

Seong Hyeon Hong[1] · Jackson Cornelius[2] · Yi Wang[1] · Kapil Pant[2]

Received: 21 August 2019 / Accepted: 17 October 2019 / Published online: 25 October 2019

Abstract

This paper presents a methodology that combines a dual-net model and the model predictive control (MPC) to compensate degraded system performance caused by slow-paced faults/anomalies. The dual-net model is comprised of an offline and an online artificial neural networks (ANNs) along with a switch that selects one of them for MPC. Through selective online updating of weight parameters, the online ANN is able to accurately capture the fault-induced variations in system dynamics, and can be used for MPC reconfiguration and fault compensation. Specifically, the system dynamics is identified by training a multilayer perceptron (MLP). To improve the model accuracy, a meta-optimization approach based on the genetic algorithm is applied to optimize the MLP hyperparameters and the training algorithm. A dual-thread decision maker is proposed to manage the robust model updating scheme and the dual-net model switch. A case study of numerical simulation using an unmanned quadrotor is undertaken to verify the feasibility of the proposed method to mitigate performance degradation. Salient performance in the response prediction and control, subject to gradually growing anomaly is successfully demonstrated. Quantitatively, the proposed updating model outperforms the offline ANN model and yields 2× and 4× lower errors, respectively, for prediction and control of the system response.

Keywords Neural network · Meta-optimization · Model predictive control · Anomaly mitigation

1 Introduction

Machine learning techniques, in particular, artificial neural networks (ANNs), have emerged as effective and popular approaches to identify complex behavior in nonlinear systems in the past three decades and enable accurate and robust control [1–3]. This is because it is often difficult to construct accurate, physics-based, control-oriented models due to the complexity and unknown dynamics of the systems. Thus, the use of the ANN can overcome these issues and allows to capture the nonlinear behavior and develop high-quality control strategies in the form of a set of predefined mathematical structures [4].

Among different classes of ANN control systems, ANN-based model predictive control (MPC) has garnered significant interest due to its salient applicability to nonlinear model predictive control (NMPC) applications. ANNs serve as system models to forecast future dynamic behaviors, and these predictions then can be utilized by the controller to determine the optimal control inputs that minimize the predefined cost function. One of the key requirements of NMPC is an accurate system model of a simple mathematical structure to represent the nonlinear system behavior. Therefore, ANNs are highly desirable methods for identifying NMPC-compatible models. Moreover, MPC is able to easily incorporate input, output, and state constraints, which makes it a popular approach for many practical systems. ANN-based MPC has found broad uses in real-world applications, including a water level regulation of a tank unit, a piezoelectric actuator, and a stirred tank reactor [5–7]. They all utilize the most commonly used ANNs as the main architecture

✉ Yi Wang, yiwang@cec.sc.edu | [1]University of South Carolina, Columbia, SC 29208, USA. [2]CFD Research Corporation, Huntsville, AL 35806, USA.

SN Applied Sciences (2019) 1:1488 | https://doi.org/10.1007/s42452-019-1526-9

for system modeling, such as the multilayer perceptron (MLP) and radial basis function network (RBFN). An MLP consists of at least three layers of neurons (or nodes), where each neuron uses an activation function except for the input nodes. A RBFN is similar to MLP but neurons utilize radial basis functions as activation functions. Furthermore, the recurrent neural network (RNN) is also a widely used ANN structure for MPC in both system identification and control [8, 9]. For example, Han et al. [10] proposed a self-organizing RNN to control the dissolved oxygen concentration in a wastewater treatment process. An RNN has a unique structure where the outputs from the previous step are fed into the current step, which makes it a good approximator for time series data. In addition, a fully connected cascade (FCC) network, which has direct connections from all input neurons to all output and hidden neurons, was applied by Negri et al. [11] in MPC for pressure control of a water tank.

Albeit, various ANNs are capable of improving computational efficiency and control performance, ANN-based MPCs suffer from the difficulty to compensate disturbances [12]. Mild disturbances caused by mismodeling and environment may be moderated by the receding horizon technique. Nevertheless, actual mechanical systems all undergo slow paced degradation, arising from wearing, tearing, corrosion, minor damage, and failure. This will cause the behavior drift of the actual system from the ANN model identified from the normal operation, leading to the deviation of the model prediction from the actual responses read from the sensors. Consequently, the performance of the MPC synthesized for the nominal (original) system will be compromised, causing a nonzero steady state tracking error also known as an offset. Offset-free tracking for MPC has been accomplished by disturbance modeling and observer design by various groups [13–15]. Another approach in dealing with such disturbances is to employ an adaptive ANN model [16–18]. The adaptive ANN model updates its weights (or even a structure) in real-time with the online collected data. When the system dynamics is shifted, changes in the system are projected into the data, enabling the ANN to learn and capture the new dynamics. The foremost merit of this method is that no a priori information of the disturbance is required. Nonetheless, it also has several distinct disadvantages. First, often the adaptive ANN updates its weights at every time step (or in every few steps), even when it is not necessary, which is computationally inefficient and makes it susceptible to noise when no anomaly is present. Second, usually to enable the online training, the selected ANN structure is too small to represent the actual system for a wide range of inputs and outputs. Lastly, there is limited measure to effectively circumvent overfitting or other training issues when the model is updated recursively, which may be a serious issue for safety–critical systems, such as vehicles and power plants.

In order to address aforementioned limitations, in this paper we present a robust and feasible ANN-based MPC methodology to maintain generalized accurate model representation and efficiently compensate for degraded performance due to dynamic behavior shifts that are mostly caused by slow-paced anomalies of actual mechanical systems, such as wearing, tearing, fatigue, corrosion, and etc. It should be noted that rapid, abrupt system faults or failures are not the focus of consideration in the present study. The framework includes three key components: ANN meta-optimization, dual-net model, and MPC. In our methodology, the system dynamics is described by an ANN-based plant model in the NARMAX formulation, for which the MPC will be developed. Therefore, the first step is to determine the ANN architecture for salient model performance, which can be achieved though the meta-optimization methods. ANN meta-optimization using evolutionary algorithms can be found in various literatures [19–22]. In this effort the ANN topology of the plant model is optimized using the genetic algorithm (GA) to select the most appropriate values of the time window size of the input and output delays in the NARMAX formulation and the size of the hidden layer that minimizes the training and validation error. The concept of the dual-net model has been originally introduced by Puttige and Anavatti [23]. Specifically, both offline and online ANN models in the form of multilayer perceptrons (MLPs) for the desired system are trained. Then the two ANNs are connected in parallel with a switch to select the one that predicts actual system response more accurately during the previous epochs. It is shown that the dual-net model is able to outperform the individual model in quantitative prediction. In this study, the dual-net model is built on the optimal MLP structure determined by the GA above. First the offline ANN model is trained beforehand in conjunction with the meta-optimization using a set of data of great diversity collected during previous nominal operations. The offline ANN remains unchanged throughout the entire period of the current operation and serves two purposes: (i) to identify the extent of the deviation in system dynamics as a result of the slow-paced anomalies above to inform the users; and (ii) to be utilized by the MPC whenever the online ANN is not ready for use, such as poor prediction due to overfitting or other training issues. Once the offline ANN becomes available, the online ANN model is initialized as a duplicate copy of the offline ANN. During the operation, the online ANN is updated whenever the system model prediction deviates notably from the actual system response due to anomalies. Then the MPC is used to design optimal control trajectories given the prescribed cost function and constraints in the presence of operational anomalies. Because of system degradation, its dynamics and responses read from sensors will deviate from those of the nominal system, and the MPC synthesized for the offline nominal system will exhibit steady state error even if it satisfies the control criteria. To tackle the issue, a dual-thread decision maker is proposed to manage the model updating and switch within the dual-net model and coordinate its prediction with the MPC to compensate the degraded system performance.

It should be pointed out that online ANN updating to mitigate the disturbance in the ANN-based MPC is established in literatures. Therefore, combining ANN and MPC is not the focus of the present study. Our objective is to combine the dual-net model and MPC and coordinate them in an organized manner during online operations to establish a deployable framework that safely maintains the performance level when the system is undergoing degradation. There are several novelties in the present effort that distinguish it from the existing work, including the use of GA-optimized ANN model in MPC, which to the best of our knowledge, has not been adequately investigated. The dual-thread decision maker to manage the online ANN updating and coordinate the dual-net model in MPC is also proposed, which serves as the cornerstone to organize the entire process for enhanced robustness and efficiency. Lastly, through the case study of numerical simulation, the feasibility of compensating degraded system performance in the presence of gradual anomalies by integrating the above key components is verified. The salient improvement in ANN prediction and control performance obtained through the proposed methodology relative to the non-updating benchmark is also demonstrated and quantitatively characterized.

This paper is organized as follows. In Sect. 2, the proposed framework/methodology of online ANN-based MPC, including GA-based meta-optimization, dual-net model, MPC, and dual-thread decision maker is described. In Sect. 3, a case study of controlling an unmanned quadrotor and the procedure of how to implement the anomaly is explained. The results of system identification and modeling using the GA, and prediction and control performance of the proposed methodology are discussed in Sect. 4. Finally, Sect. 5 concludes the paper with a summary of achievements and future work.

2 Online updated artificial neural network and model predictive control

2.1 Methodology

The present methodology for online monitoring, prognostics, and control of mechanical systems is illustrated in Fig. 1a. The physical plant is controlled by the control inputs determined by the MPC through optimization. There are two key components in the MPC module: the dual-net model and the optimizer. The dual-net model, comprised of two ANNs, is a digital representation of the physical plant, and uncovers the system dynamics and the relationship between control inputs and responses. It predicts the plant response at its current time given the historical values of the response and the inputs of the actual plant. The difference between the predicted output and the actual output at the same time instant will be calculated, which can be used for: (i) driving an online machine learning algorithm to re-train the ANN in the dual-net model and update the weights throughout the operation to capture the latest plant dynamics. The details of online ANN training are given in the section below; and (ii) detecting anomaly of the system, e.g., consistent deviation of actual response from the predicted, within a specified time window may indicate the anomaly or the increasing severity of the anomaly, i.e., monitoring and prognostics. Note that (i) will only operate in the presence of anomaly

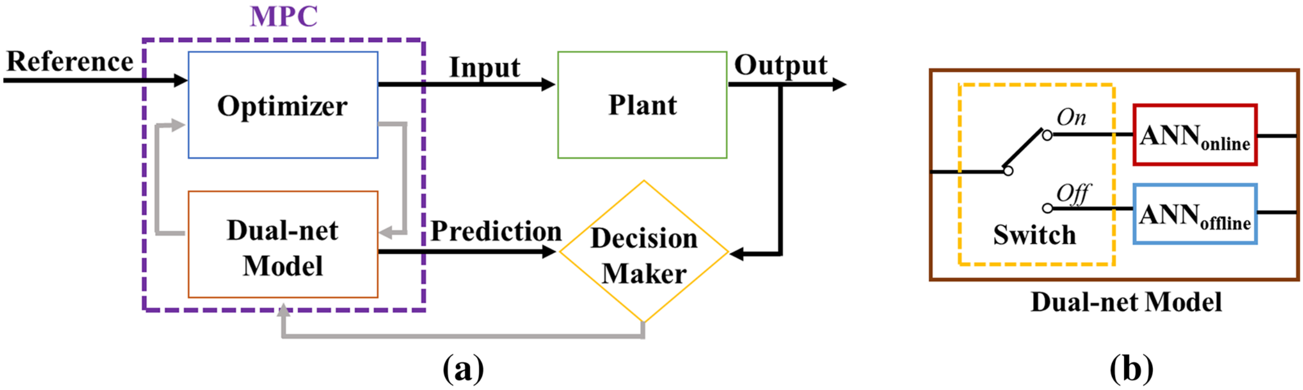


Fig. 1 Schematic of the simulation overview

as the sensor readings collected from the degraded system provide valuable information to update the ANN model. One caveat to this approach is that sensor faults are assumed to be absent since all data-driven models require accurate data for training.

The structure of the dual-net model is shown in Fig. 1b, which includes two ANNs: online and offline, connected in parallel to a switch in the work flow. The offline ANN trained beforehand remains unchanged throughout the operation. When the system is in a normal status, decision maker selects the *off* branch in the switch to utilize the offline ANN for MPC configuration. On the other hand, when the system anomaly occurs and causes the deviation of the model-predicted response from the actual system response, the online ANN is re-trained during operation with the accumulated data to accurately capture the latest system dynamics. One potential issue of the proposed methodology is that the online re-training of the network utilizing the biased training data may be overfitted, leading to poor control performance. Therefore, the decision maker decides in situ which model is better and should be used along with MPC by assessing the prediction accuracy of both models in the presence of anomalies. If the online updated ANN outperforms the offline ANN in prediction, the decision maker selects the *on* branch in the switch to use the former for MPC.

The model predictive control (MPC) module uses the dual-net model to generate a sequence of control signals for actuators at the desirable interval that drive the system to follow the reference signal and to mitigate adverse effects arising from the anomaly. Our MPC is based on the receding horizon technique, and its cost function considers the model-predicted response relative to the reference and the temporal variations of the control signals over a specified time horizon. A numerical optimization program is then harnessed to determine the control inputs that minimize a performance criterion over the horizon. The detailed description of each component in Fig. 1 will be presented in the following sections.

2.2 Genetic algorithm-guided neural network modeling

Developing a mathematical model of a physical plant is often challenging due to its complexity and lack of knowledge in the behavior of the system. One approach to model the unknown system is to establish the input-response relationship using the time series data generated from the system, i.e., data-driven modeling. The classical model, Nonlinear Auto-Regressive Moving Average with eXogeneous inputs (NARMAX) is a general and effective representation of the nonlinear discrete-time system as shown in Eq. (1) [24]

$$y(k) = F(y(k-1), \dots, y(k-n_y), u(k), \dots, u(k-n_u)) \tag{1}$$

where $u(k)$ and $y(k)$ are respectively, the input and response at the current time step, n_u and n_y are input and output delays, respectively, and $F(\cdot)$ is a nonlinear function that quantitatively describes the NARMAX relationship and can be determined using available input-response data. Equation (1) clearly shows that the response $y(k)$ at the current time step depends on its historical values and the current and previous inputs. In this paper, the artificial neural network (ANN) is used, which is one of the most widely used data-driven modeling approach to approximate the nonlinear function F.

Often a recurrent neural network (RNN) is used in the dynamic system modeling due to its salient capability to predict longer horizon. However, a multi-layer perceptron (MLP) is selected in the present work for two reasons: (i) because our ultimate goal is to update the ANN model during the online operation, and the MLP features a simpler structure leading to a better choice for online updating; and (ii) the model cast in the NARMAX formulation with GA-optimized input and output delays is able to capture the nonlinear dynamic behavior and allows a recursive use of this one-step ahead predictor. Furthermore, MLP is comprehensive enough to approximate any nonlinear continuous function by a three-layer (input, hidden, output) network structure [19]. Therefore, the number of hidden layers is constrained to be 1 in this work. The schematic of the ANN is shown in Fig. 2 and the associated equation is as follows

$$\begin{aligned} y_n(k) &= W^{(2)} \tanh\left(W^{(1)} X(k)\right) \\ X(k) &= \begin{bmatrix} 1 & y(k-1) & \cdots & y(k-n_y) & u(k) & u(k-1) & \cdots & u(k-n_u) \end{bmatrix}^{\mathrm{T}} \end{aligned} \tag{2}$$

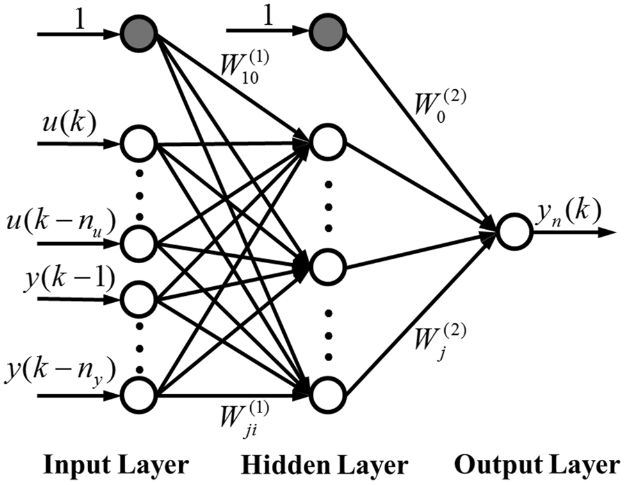


Fig. 2 A multi-layer perceptron (MLP) structure of ANN

Table 1 Hyperparameters of interest and their types and range for the ANN model

Hyperparameter	Type	Parameter range
Input delay	Integer	[1–30]
Output delay	Integer	[1–30]
Hidden layer neurons	Integer	[1–50]
Training algorithm	Integer (list index)	[1–12]

where $W^{(1)}$ and $W^{(2)}$ are the input-to-hidden and hidden-to-output weight matrices, respectively. The hyperbolic tangent function is used as an activation function of the hidden layer herein. Given a dataset from a physical plant, constructing a high-quality ANN model is not straightforward. Due to the large number of model hyperparameters and data configurations to proceed ANN training, determining the optimal model inputs and the MLP structure for a given task using the *trial and error* approach can be time-consuming and tedious. Therefore, automated search of hyperparameters within a broad range is carried out to achieve the optimal MLP model for enhanced model accuracy that otherwise is not available through manual selection. Table 1 shows the hyperparameters of the MLP model that are identified for automated tuning, and the corresponding search range. For both the input and output delays, the lower and upper limits are chosen to be 1 and 30, respectively, and the number of neurons in the hidden layer is selected within the range of 1–50. Larger delays and hidden neurons usually improve the training accuracy but at the cost of increased size and complexity of an ANN. As a result it will require more training time and resource usage. Note that if optimal values within the search space are selected close to the upper limits, then the range must be extended to allow more freedom to the search. In addition to the size of the MLP, twelve different training algorithms are explored, including gradient descent (GD), Levenberg–Marquardt (LM), Bayesian regularization (BR), BFGS quasi-Newton, and others. Indeed, the accuracy of the training algorithm during ANN training heavily depends on the specific data set, e.g., the noise level.

To efficiently assess the influence of these hyperparameters on MLP performance within a broad space and automatically select the highest-performance configurations for a specific problem, the meta-optimization method is utilized. Meta-optimization is essentially to use one optimization method to tune the parameters of another optimization method, i.e., the ANN training process. Evolutionary methods are widely exploited for this purpose; therefore, a genetic algorithm (GA)-based hyperparameter optimization module is also developed. An overview of our GA-based meta-optimization workflow is shown in Fig. 3. The GA is one of the most popular meta-optimization techniques inspired by biological evolutionary concepts [25]. It is motivated to evolve an initial population of random gene sequences, toward a final population of "fit" gene sequences that demonstrate optimal performance on a fitness function. The fitness function is used to assess the performance of a given gene sequence, i.e., the hyperparameter configuration. These genes can be represented as bit-strings, double vectors, integer vectors, or a mixture of these. For our problem, double vector gene representations are used to encode the network hyperparameters-of-interest. After a population of genes is evaluated on the fitness function, they are ranked based on fitness. Operations, such as selection, crossover, and mutation are used to evolve the population of genes to maximize fitness. Particularly in this work, Gaussian mutation, scattered crossover and stochastic uniform selection algorithms are applied. The penalty (or fitness) function is computed by decoding the gene sequences into physically useable hyperparameters, training an MLP model using the hyperparameters, and computing the final MLP cost on the validation set, which is found by computing the mean squared error (MSE) on the predictions. Once the stopping criteria of the GA is met, e.g., a maximum number of the generation is reached or a minimum penalty is achieved, the genes of highest performance in the population are decoded into the selected hyperparameters.

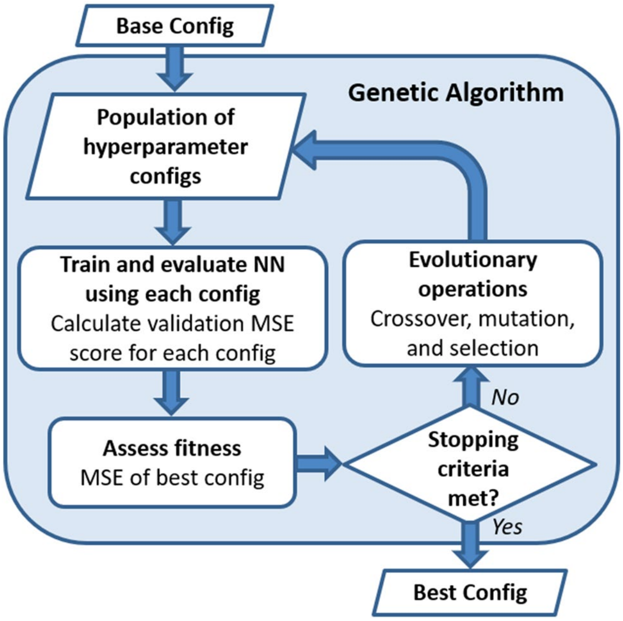

Base Config
Genetic Algorithm
Population of hyperparameter configs
Train and evaluate NN using each config
Calculate validation MSE score for each config
Evolutionary operations
Crossover, mutation, and selection
No
Assess fitness
MSE of best config
Stopping criteria met?
Yes
Best Config

Fig. 3 Flowchart describing the genetic algorithm for ANN hyperparameter optimization

2.3 Model predictive control

Model predictive control (MPC) consists of three parts: the cost function, the optimizer, and the system model. In this paper, the system model is represented by the dual-net model as described above. That is, the MPC takes the predicted response (y_n) over a specified time horizon from the dual-net model and the reference response (y_r) as inputs, and generates the control signals over another time horizon determined by a numerical optimization problem that minimizes the following performance criterion, viz., cost function over the specified horizon:

$$J = \sum_{j=N_1}^{N_2} (y_r(k+j) - y_n(k+j))^2 + \rho \sum_{j=1}^{N_u} (\Delta u(k+j))^2 \Delta u(k+j) = u(k+j) - u(k+j-1) \tag{3}$$

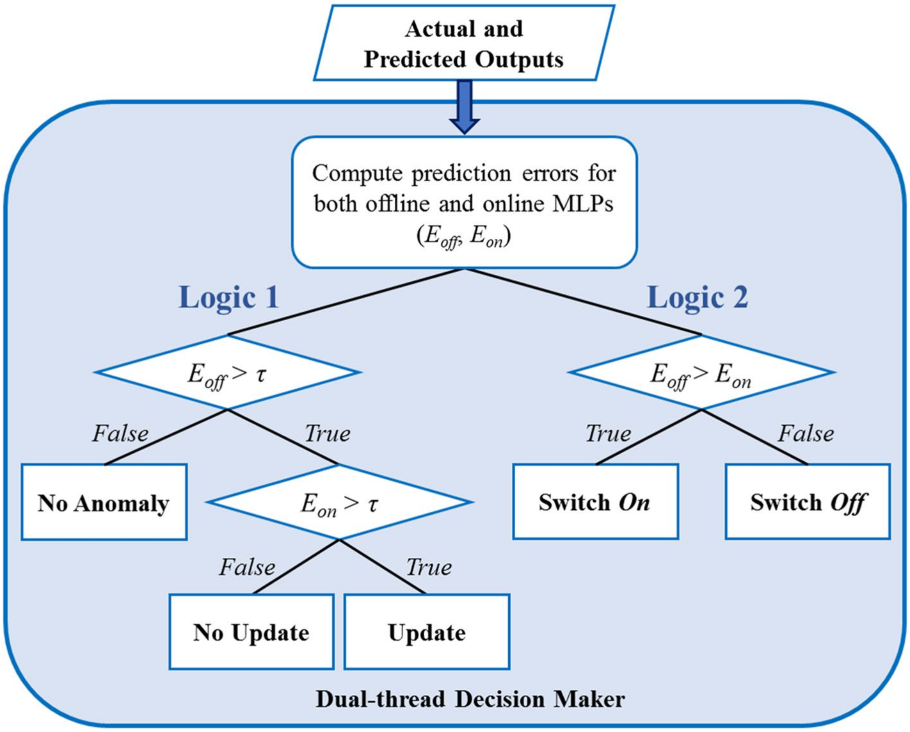
where N_1 is the minimum costing horizon; N_2 is the maximum costing horizon; N_u is the control horizon, and y_r is the reference input, y_n is the predicted ANN output, and ρ is a control weighing factor. According to Eq. (3), the cost function includes not only the mean squared error (MSE) between the reference response and the ANN predictions, but also the changes in the control signal u as a penalty term, where $\rho > 0$ is the penalty parameter. Therefore, ρ decides how much the change in control input is allowed. Larger N_2 and N_u will improve the control performance, but it will increase the computational load during both the offline and the online stage. The goal of MPC is to compute $[u(k+1), \ldots, u(k+N_u)]$ by minimizing Eq. (3) for every control epoch. For our simulation study, N_1, N_2, N_u and ρ are selected empirically, which yield consistent and reliable performance in this work. Selecting these control parameters is not critical since they do not have impact on the steady state error caused by the disturbance due to the dynamic shifts of the system as studied herein. The stability of ANN-based MPC is proved by the Lyapunov synthesis method in literatures [26, 27]. Most widely used algorithms to solve this type of optimization problem are Newton, quasi-Newton and Levenberg–Marquardt related methods. In this paper, a bounded BFGS quasi-Newton method is adopted because of its computational efficiency and reliability.

2.4 Dual-thread decision maker

There are two Boolean logic threads that govern the entire online anomaly detection, ANN model updating, and compensation process, which is termed the dual-thread decision maker hereafter. As shown in Fig. 4, E_{off} and E_{on} refer to the mean squared prediction errors of the offline and online ANNs (or MLPs) relative to the actual system response, respectively, for a specified time window.

The first logic thread, on the left, is for the anomaly detection and model updating. There are a variety of anomaly detection approaches, including clustering, nearest neighbors, statistical, subspace, classifier and others [28, 29], and their applications have been reported

Fig. 4 Flowchart describing the dual-thread decision maker


Actual and Predicted Outputs
Compute prediction errors for both offline and online MLPs (E_{off}, E_{on})
Logic 1
Logic 2
$E_{off} > \tau$
$E_{off} > E_{on}$
False
True
True
False
No Anomaly
$E_{on} > \tau$
Switch On
Switch Off
False
True
No Update
Update
Dual-thread Decision Maker

in numerous systems [30–33]. Nevertheless, finding the optimal one is out of scope of this research as our primary focus is to investigate the feasibility of the methodology to maintain system performance by updating the ANN model and MPC design during the operation using online data, especially when the plant experiences a slow-paced degradation or drift in system dynamics. Therefore, we employ the out-of-limits (OOL) approach, which is the most widely used method [34]. OOL simply uses predefined threshold values, denoted as τ in Fig. 4, and alerts whenever the difference between the sensor and the predicted data exceeds the threshold value.

The first logic is divided into two stages, respectively, comparing E_{off} and E_{on} against the threshold value τ in the given order. Recall that the offline ANN models the dynamics of the original, nominal system. Therefore, when the criterion $E_{off} > \tau$ is true, it indicates the presence of the anomaly. The second criterion of Logic 1, i.e., $E_{on} > \tau$ is used to determine necessity of updating the online ANN model. If the online ANN has already been updated and allows accurate prediction, making $E_{on} > \tau$ false, then the online ANN updating becomes unnecessary (no update will be performed). On the other hand, if there are continuously growing anomalies or training failures that cause $E_{on} > \tau$ to be true, then the online ANN will be updated again until the prediction error reaches below the threshold. Several points should be noted: the online ANN is initialized with a copy of the offline ANN, that is, initially, the weights of the online ANN are the same as those of offline ANN. This will actually reduce the training time since the training process will not begin with randomly assigned weights, and the weights of offline ANN are essentially a good starting point for the online ANN updating. Moreover, when the anomaly is detected for the first time, the online ANN will also be updated immediately because of $E_{on} = E_{off} > \tau$.

Although updated, the online ANN does not guarantee to be utilized by the MPC. This is because online ANN training is highly susceptible to the overfitting and other related issues that may provide inaccurate prediction. If used by the MPC, it may deteriorate system performance or even lead to system failure. The second logic thread (on the right from the figure) and the switch in the dual-net model are introduced to prevent this issue. That is, the second logic thread compares the accuracies of the offline and the updated ANNs when predicting the actual plant responses for a specified time window, and then decides the action of switch. If the updated ANN outperforms the offline ANN, i.e., $E_{on} < E_{off}$, then the switch is turned *on* and the updated ANN is used in MPC to compute the control actions and vice versa (as shown in Fig. 1b). Throughout the entire process both logic threads operate independently at every time step to determine when to update the online ANN and which model to use for MPC reconfiguration.

2.5 Dealing with overfitting for online training

Updating the ANN using the online data is a formidable task mainly due to the overfitting issue. This is because the range and the diversity of the online operational data is usually extremely limited. For example, if the objective of an unmanned aerial vehicle (UAV) during the operation is to maintain its flight at a certain altitude, then the data collected online from the UAV will have a small range in altitude variation. This is critical in ANN training since the foundation of ANN is to train a generalized model using a wide range of data. Moreover, once the controller is active in the closed loop, the operators no longer have direct manipulation on the control inputs applied to the physical plant. In other words, we can only provide reference values that the controller will strive to meet. Therefore, when the online data is accumulated with the controller in the loop, the data will depend on the control scheme. For instance, the control weighing factor (ρ) in MPC introduced in Sect. 2.3 restricts the changes in the input, and eventually restrains the diversity of data.

Accordingly, actions are required to prevent the ANN from overfitting. The effects of the data volume used for online ANN update is first investigated. We start with data sets of small sizes, and eventually find through trials that increasing the data volume reduces the overfitting effect, and the larger data volume is favorable to creating more generalized ANN models. However, an inordinately large data volume could significantly increase the data accumulation time and slow down the response rate of mitigating the disturbance. For this particular work, we decide to use 2 h of data accumulated online to update the ANN. The data volume will vary depending on the systems, disturbances and objectives.

Other than increasing the data volume, the technique of early stopping for ANN training is also used with the stronger condition. Early stopping is a way to terminate ANN training when the performance error of the validation set begins to grow while the performance error of the training set continues to decrease. This means that the ANN is being overfitted to the training set, and the network is losing its generality. Whenever this event occurs, the training algorithm counts the number of occurrences. Once the number exceeds the predefined value, the training stops. Stronger condition refers to reducing this predefined value and increasing the ratio of the validation set with respect to the training set.

Lastly, another means we take to mitigate the overfitting issue during online training is to apply Bayesian regularization (BR) as the training algorithm. Although

Levenberg–Marquardt (LM) is found to be the excellent algorithm for the offline ANN modeling, due to its fast convergence rate, there is a large chance of being overfitted for the online use (see Sect. 4.1 below). In other words, LM allows more rapid changes in the weights than the BR within one iteration. Applying early stopping in conjunction with BR is found to be a good option for the present work.

3 Case study and numerical experiment

To verify the concept of the online ANN updating and degraded performance compensation, an unmanned quadrotor system is chosen to represent the actual plant, which has a well-known, physics-based mathematical model that is easily accessible. MPC for unmanned quadrotors has been demonstrated recently by several groups [35–38]. Zhang et al. [27] recently proposed ANN-based MPC for formation flight of multiple unmanned quadrotors, which uses RNN to update the weight parameters at every time step that is different from our approach based on the dual-net model.

3.1 Plant model

A simple schematic of the quadrotor is displayed in Fig. 5, where (Ω_1, Ω_2, Ω_3, Ω_4) are angular velocities of each rotor. The full equations of motion are given by [39],

$$\begin{aligned}
\ddot{\phi} &= \frac{I_{yy} - I_{zz}}{I_{xx}} \dot{\theta}\dot{\psi} - \frac{J_r}{I_{xx}} \Omega_r \dot{\theta} + \frac{l}{I_{xx}} u_2 \\
\ddot{\theta} &= \frac{I_{zz} - I_{xx}}{I_{yy}} \dot{\phi}\dot{\psi} - \frac{J_r}{I_{yy}} \Omega_r \dot{\phi} + \frac{l}{I_{yy}} u_3 \\
\ddot{\psi} &= \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\theta}\dot{\phi} + \frac{l}{I_{zz}} u_4 \\
\ddot{x} &= -\frac{u_1}{m} (\sin\phi \sin\psi + \cos\phi \cos\psi \sin\theta) \\
\ddot{y} &= -\frac{u_1}{m} (\cos\phi \sin\psi \sin\theta - \cos\psi \sin\phi) \\
\ddot{z} &= g - \frac{u_1}{m} (\cos\phi \cos\theta)
\end{aligned} \tag{4}$$

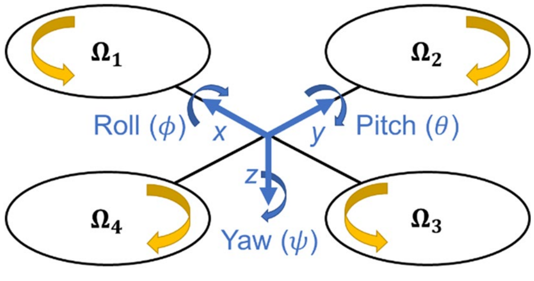


Fig. 5 Schematic of the quadrotor

where (x, y, z) and (θ, ϕ, ψ) represent translational and rotational motions in the body fixed coordinate system, respectively; (I_{xx}, I_{yy}, I_{zz}) are the area moment of inertias about each body frame axis; (u_1, u_2, u_3, u_4) are the inputs that create motions in the directions of (z, θ, ϕ, ψ), respectively; Ω_r is the relative speed of rotors; J_r is the rotor's inertia; l and m are the arm length and the total mass of the quadrotor, respectively; and g is the gravity. Moreover, inputs (u_1, u_2, u_3, u_4) are computed by multiplying the transformation matrix as shown in Eq. (5) [39].

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ 0 & -K_f & 0 & K_f \\ K_f & 0 & -K_f & 0 \\ K_m & -K_m & K_m & -K_m \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{5}$$

here K_f and K_m are the aerodynamic force and moment constants, respectively. The actual system has 4 inputs with 3 translational and 3 rotational motions. In order to simplify the problem to verify the feasibility of our methodology, only the yaw angle (ψ) and the altitude (z) of the quadrotor are considered in this work.

3.2 Offline model training and reference signal

For system identification, two separate offline MLPs are trained with the model structure described in Sect. 2.2, each representing a multi-input single-output (MISO) system. In other words, two MLPs are trained to predict the yaw angle and altitude, separately. This is a more suitable approach than training a single ANN that represents a multi-input multi-output (MIMO) system, because the model accuracy can be compromised if two totally different motions are modeled from the same set of weights. Also if both ANNs are not separated, they need to be updated simultaneously for both the yaw angle and the altitude. On the other hand, if the ANN is only responsible for estimating a single state, then the ANN updates can be performed independently. Again, the online ANN model is initialized as a copy of the offline ANN model at the beginning of the simulation.

In the numerical experiments, random step reference signals are implemented for both yaw angle and altitude. For the yaw angle, the magnitude of each step is chosen randomly between -10 and 10 degrees with respect to the previous angle. Here, the period of each step is also chosen randomly between 10 and 20 s. Similarly, the altitude reference signal is produced by a series of random step functions with a period between 15 and 30 s, and its amplitude varies between −5 and 5 from the previous step. A short interval of the prescribed references and actual outputs are shown in Fig. 6. Gaussian sensor noise is added with the magnitude

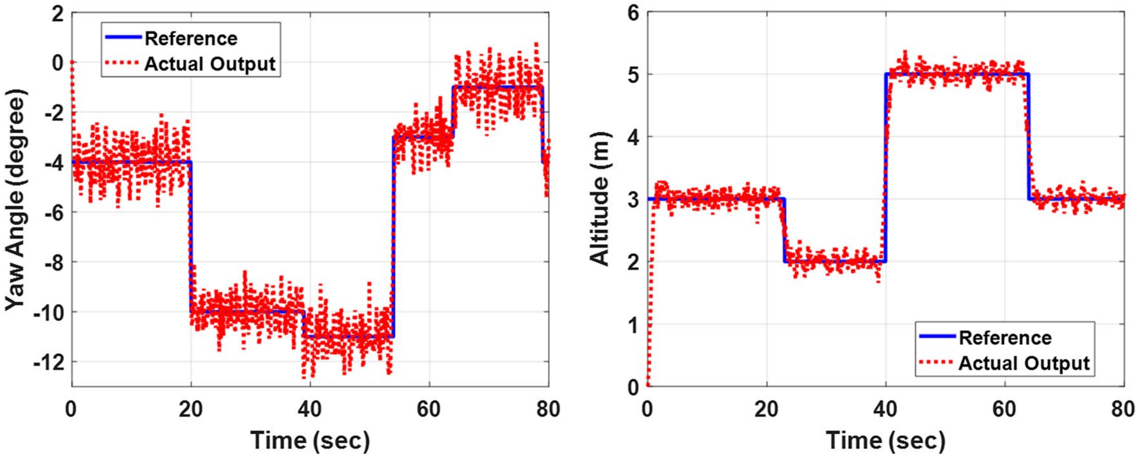


Fig. 6 Reference signals and actual outputs

of 0.01 rad and 0.1 m to the yaw angle and the altitude response, respectively.

3.3 Anomaly

The slow shift of the dynamics is obtained by prescribed degradation of blades of the quadrotor and associated aerodynamic parameters. In the quadrotor system, Eqs. (6) and (7) are used to represent the propulsion force/moment of the vehicle

$$F_i = \frac{1}{2}\rho A C_T r^2 \Omega_i^2 \text{ and } F_i = K_f \Omega_i^2 \tag{6}$$

$$M_i = \frac{1}{2}\rho A C_D r^2 \Omega_i^2 \text{ and } M_i = K_m \Omega_i^2 \tag{7}$$

where F_i is the aerodynamic force produced by rotor i, M_i is the aerodynamic moment produced by rotor i, ρ is the air density, A is the blade area, C_T and C_D are aerodynamic coefficients, r is the radius of blades. In this work, K_f and K_m are altered continuously as a prescribed function during the first few hours of the simulation to mimic the slow-paced blade degradation arising from deformation, wearing, and yielding. Figure 7 shows the magnitude of the modified aerodynamic constants of all four rotors with respect to time. The aerodynamic force constants are assumed to be equal for all four rotors to avoid any pitch and roll motions. The aerodynamic moment constants are assumed to be equal for rotors in pairs: (1, 3) and (2, 4) for the same reason. These changes will introduce disturbances in system models, leading to steady state errors.

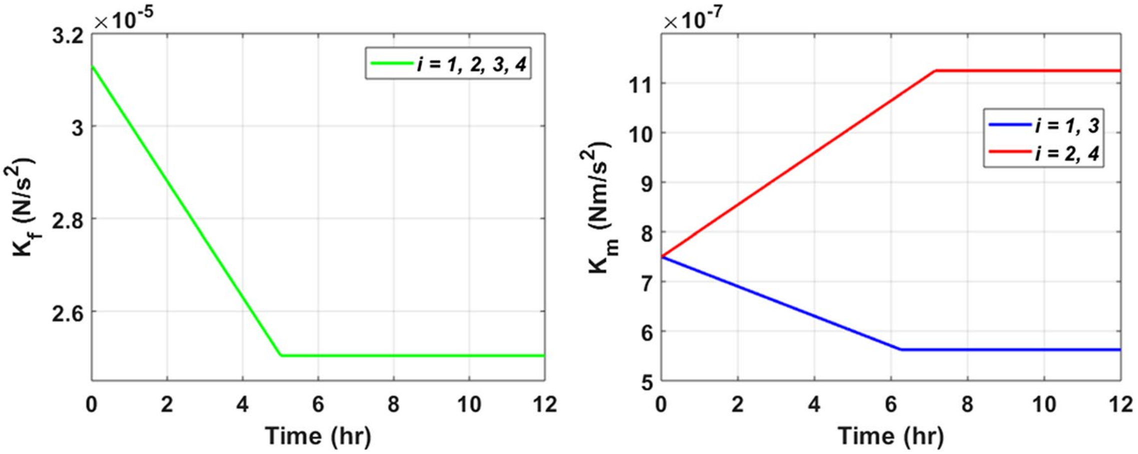


Fig. 7 Aerodynamic force (left) and moment (right) constants

4 Results and discussion

4.1 Artificial neural network hyperparameter selection

We first describe the results of using the genetic algorithm (GA) and the data produced by the physics-based quadrotor model to select hyperparameter for ANN training. Figure 8 illustrates the training data used to identify the dynamics along the axes of the yaw angle (top) and the altitude (bottom). The left column represents the input (angular velocity) and the right column represents the output (yaw angle and altitude). As described above, constraints on the angular velocity of the rotor, i.e., $\Omega_1 = \Omega_3$ and $\Omega_2 = \Omega_4$ are imposed to allow the system to vary only in the yaw angle and the altitude while keeping the pitch and the roll axis fixed. The input data is created by a series of random step functions with periods chosen randomly between 0.1 and 2 s. The yaw angle is limited to 5 revolutions and the maximum operational range in altitude is set to be ± 100 meters. The Gaussian noise is added to the output data, and the magnitudes of the noise are 0.01 rad and 0.1 m for yaw angle and altitude, respectively.

The process of the MLP hyperparameter selection using the GA as shown in Fig. 3 is then conducted, and the results are listed in Table 2. For the validation purpose, the entire GA-based selection is repeated two times. For each, 30 populations are created for 20 generations, which leads to a total of 600 individual designs. As the generation increases, we are able to observe a trend of the solution. Towards the end of each run, most of the populations have similar selections of hyperpameters with respect to those listed in the table. The input delay saturates to around 10 and the output delay converges to around 20. This implies that the window size of the delayed output has more impact toward the prediction accuracy compared to that of the inputs. The optimal number of the hidden neurons show more variance than the input/output delays, although the difference in model accuracy within this confined range of hidden neurons is negligible. The range of hidden neurons is found to be approximately between 25 and 40. Moreover, for most of the populations, the training algorithm converges to Levenberg–Marquardt (LM) method. LM is a reasonable choice for the single-layer MLP in our case study because it is known as an accurate method for nonlinear function fitting with fast computational speed. If the size and the layer of the ANN are larger, this method is likely to be eliminated since it does not cope well with complex networks. As a reminder, Bayesian regularization (BR) method is applied for online training instead of LM method (see Sect. 2.5 above). The final choice for the MLP structure is a window size of 10 for the input delay (for each input), 20 for the output delay, 36 neurons in

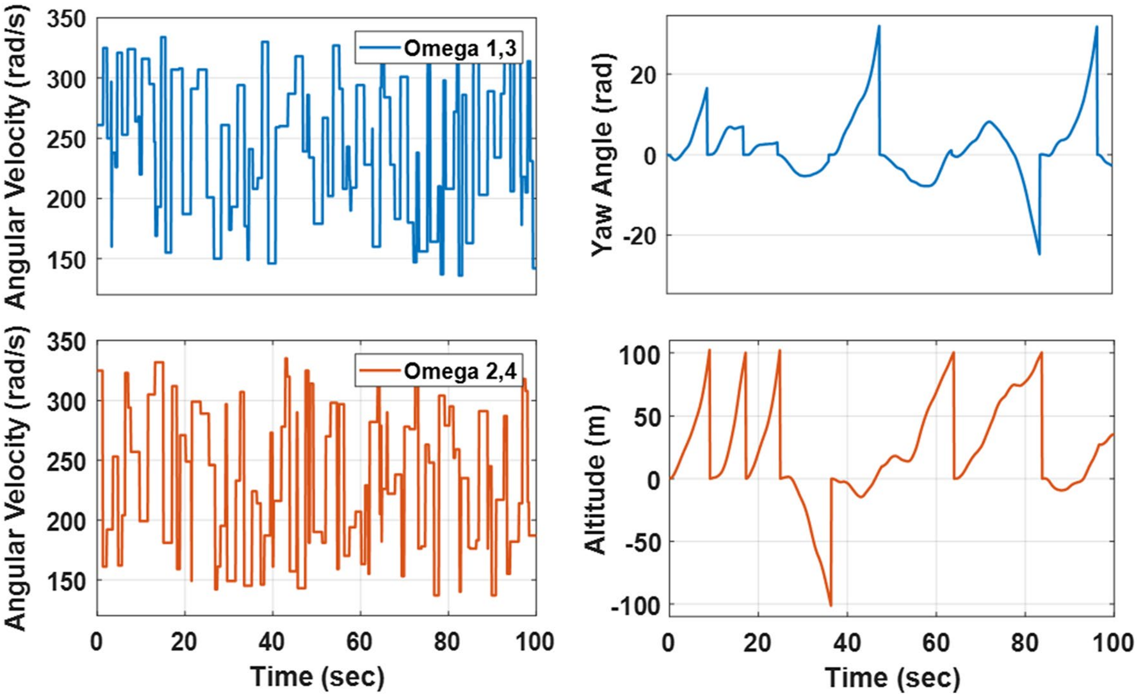


Fig. 8 Input (left) and output (right) of the ANN training data

the hidden layer, and the LM method for the training algorithm. In summary, the MLP model for the yaw and the altitude dynamics will each have 40 input nodes, 36 hidden nodes and 1 output node (Table 2). Note that the GA-based meta-optimization is only used for offline MLP training due to its large computational cost.

4.2 Anomaly compensation by ANN-based MPC

The result of anomaly prediction and compensation by the dual-net model is presented in this section. The status of dual-thread decision maker is displayed in Fig. 9, where the value of "0" and "1" represents false and true, respectively, in the dual-thread logic. Figure 9a, b show the status of Logic 1 and Logic 2, respectively, where blue represents the yaw angle and brown the altitude. We can see from Fig. 9a that updating occurs when both online and offline prediction errors are beyond the threshold value, which are 0.02 rad for the yaw angle and 0.2 m for the altitude in this case study. Once Logic 1 becomes true, the online data starts to be collected for two hours, and such a period of data accumulation is determined through trial-and-error, and exhibits great potential to mitigate the overfitting issue for online training. The data accumulation is followed by the online ANN training and updating, and the moment of this update is indicated by the peak values (greater than one) from the same curve. As the anomaly causes the deviation in ANN prediction of both the yaw angle and the altitude, Logic 1 becomes true for both outputs. Then the system continuously updates both online ANNs until the requirements of the prediction accuracy are satisfied. For the online ANN of the yaw angle, after four updates, its prediction reaches the desired accuracy and below the required tolerance value, and Logic 1 is reset

Table 2 Selected hyperparameters for ANN model by the genetic algorithm

	Input delay	Output delay	Hidden neuron	Train algorithm
Run 1	12	21	36	LM
Run 2	10	18	26	LM
Final choice	10	20	36	LM

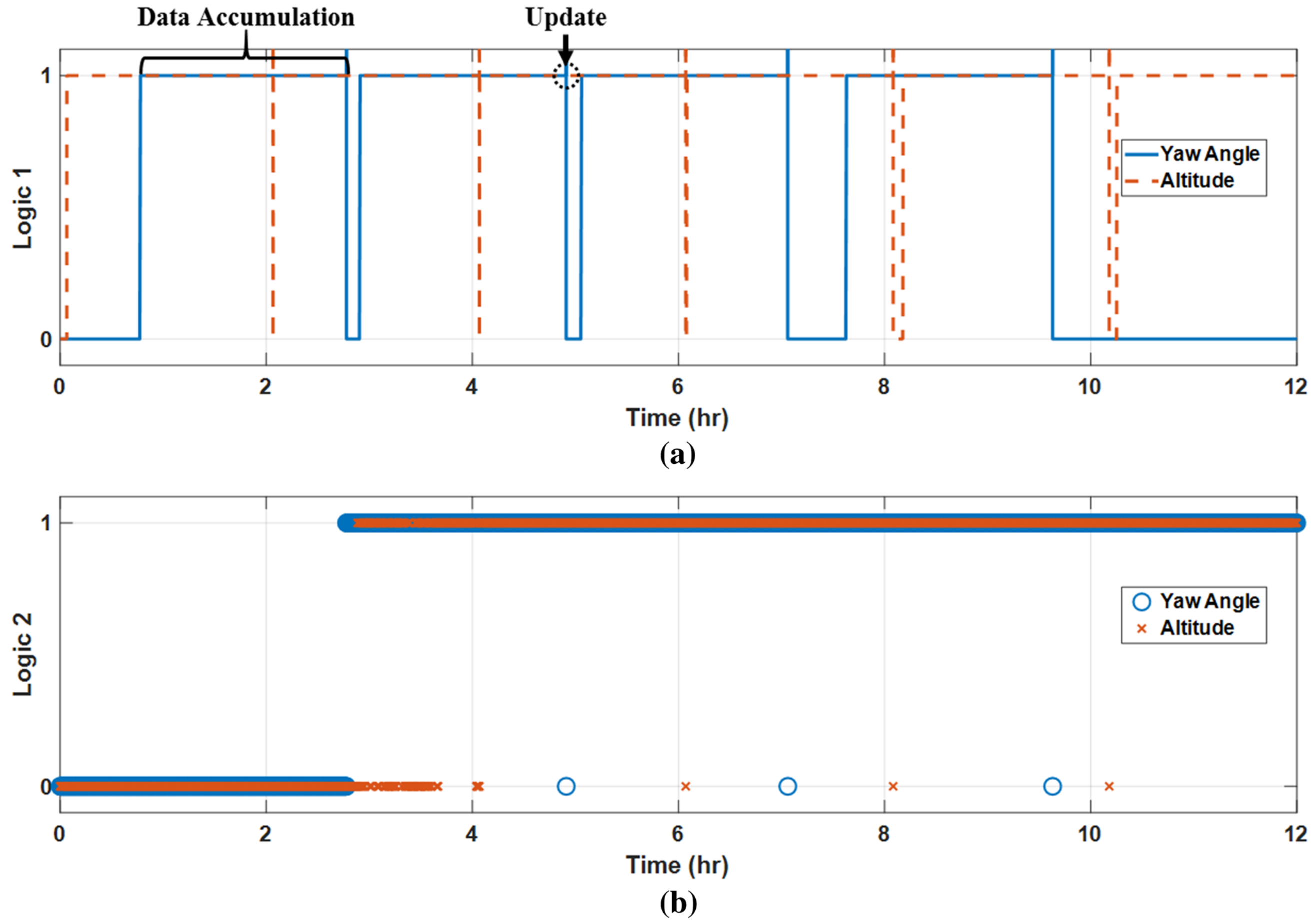


Fig. 9 Dual-thread decision maker status: **a** Logic 1 and **b** Logic 2

to zero. For the altitude, Logic 1 immediately turns back to one right after one update, indicating that the error of prediction cannot reach below the prescribed tolerance in a consistent manner. The status of Logic 2 that manages the switch and utilization of the dual-net model for MPC is shown in Fig. 9b. When Logic 2 is true, the predicted values of the online ANN is more accurate than the offline ANN, allowing MPC to utilize the online ANN. Otherwise, Logic 2 becomes false and the offline ANN is used in the MPC. The figure verifies that the online ANN mostly outperforms the offline ANN in response prediction, especially after one update for the yaw angle and two updates for the altitude, as the former is dominantly used in the MPC after 4 h.

The curves of the prediction error for the two responses, the yaw angle and the altitude, viz., the discrepancy between the model predictions and the actual plant response are depicted in Fig. 10a, b, respectively. Two sets of results obtained from the offline ANN (red) and the dual-net (green) models are presented. In the former only the offline trained ANN is used throughout the entire simulation, while the latter uses the dual-net model governed by the dual-thread decision maker for online updating and MPC model selection. Note that MPC performance depends on the prediction accuracy of the ANN models, and a large prediction error will lead to MPC degradation. It indicates that by updating the ANN following the dual-thread decision maker above, the prediction errors can be reduced to 2°–3° for the yaw angle and 0.2–0.3 m for the altitude. Throughout the entire simulation, the online updated ANNs allow more accurate evaluation of the responses, except for few spikes within the first two or three updates. The spikes signify that the updated ANNs may suffer from slightly poor model generalization. In other words, the networks are overfitted due to the restricted training data. Fortunately, as more online data is collected for training, the network generality is improved significantly, producing more consistent prediction manifested by the well bounded and smooth error.

Figure 11 shows the actual responses of the plant at the 3rd, 6th, 9th, and 12th hour of the simulation involving the MPC. Similar to Fig. 10, two sets of results, the offline ANN (red) and dual-net (green) are presented. The corresponding reference signal is shown in blue in the same figure. The results for the yaw angle and the altitude are shown in the left and the right column, respectively. It is observed that as the anomaly becomes more

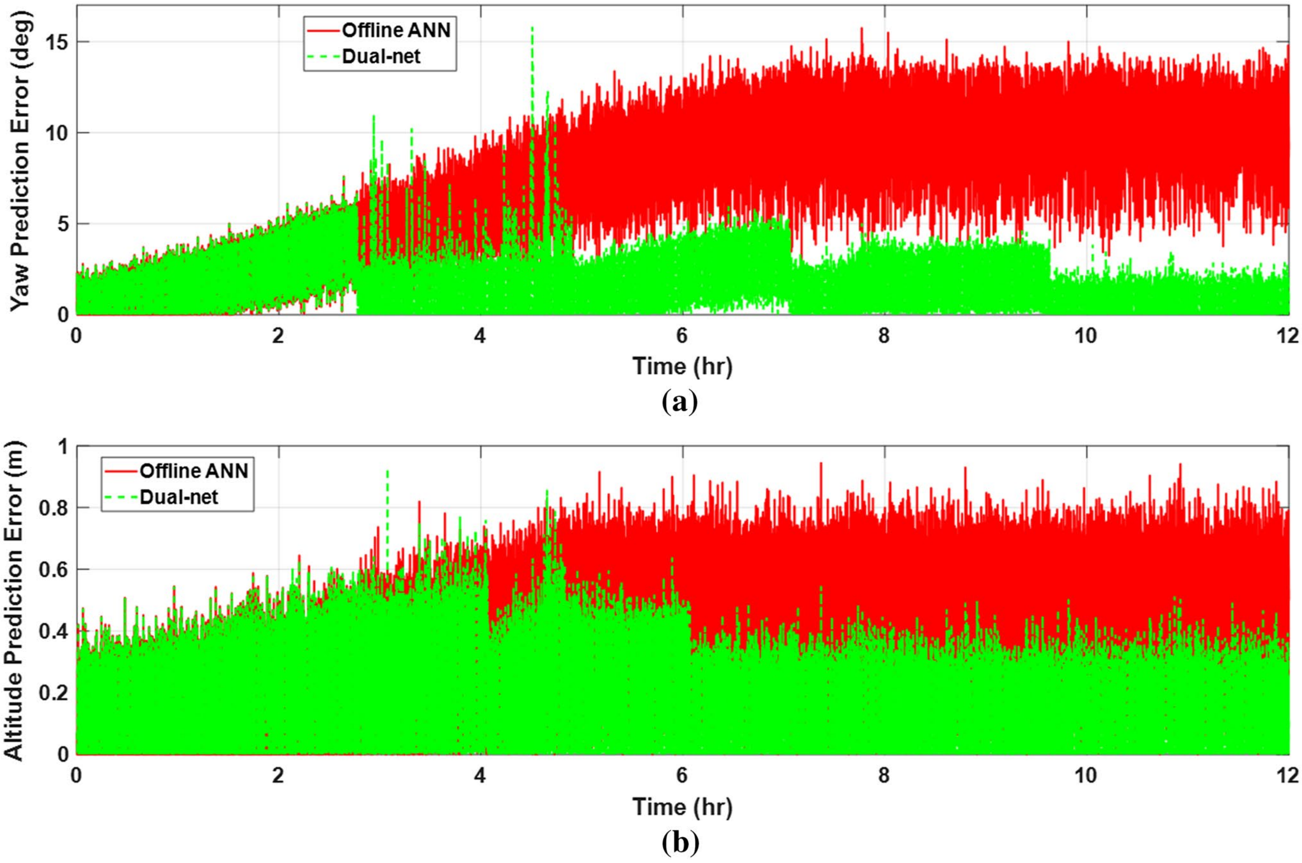


Fig. 10 Prediction errors of the offline ANN and the dual-net models for the **a** yaw angle; and **b** altitude

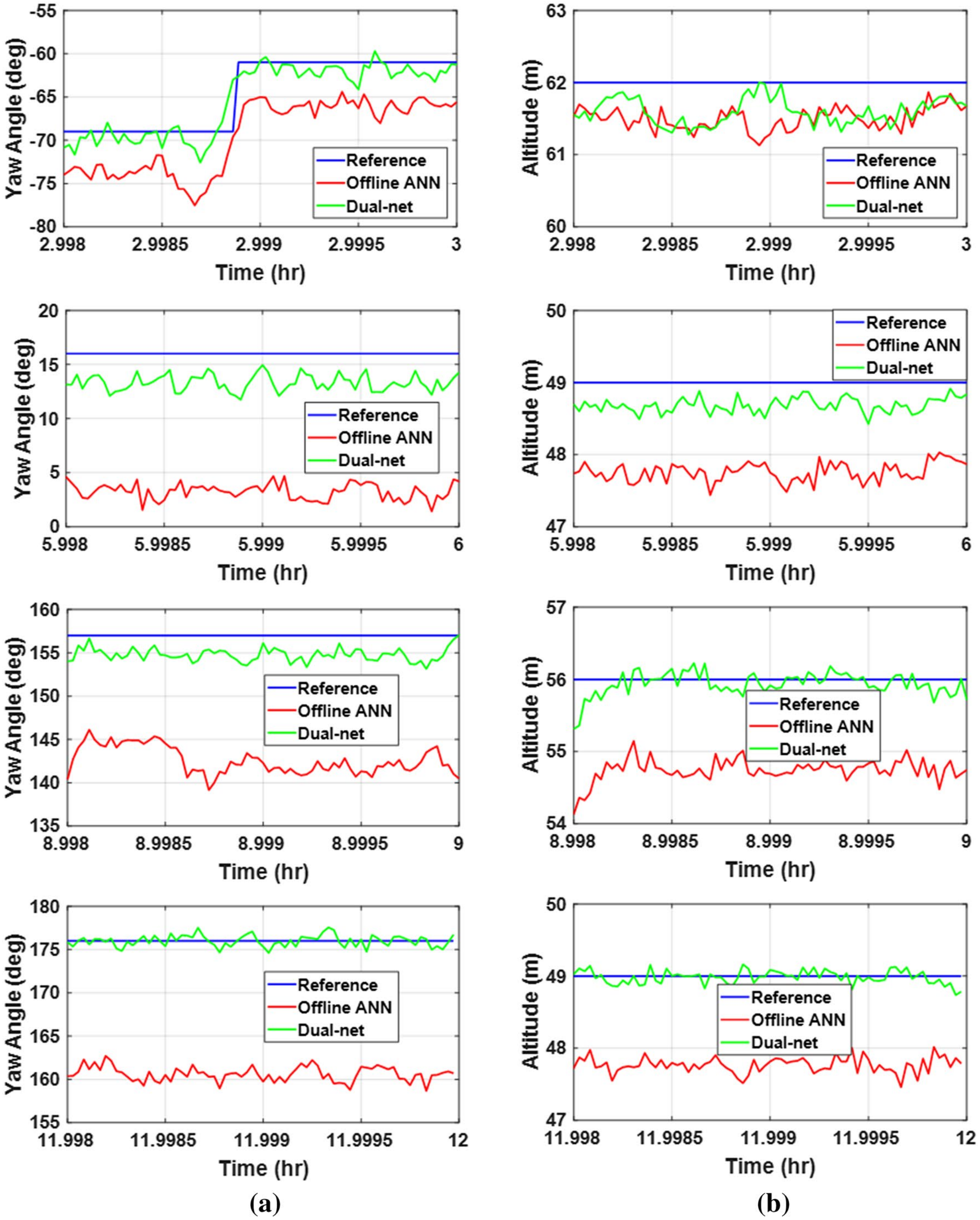


Fig. 11 Plant responses: **a** yaw angle and **b** altitude, produced by MPC using the offline ANN and the dual-net models

severe, the responses produced by the MPC based on the offline ANN model deviate more from the desired references. Eventually, the steady state errors of the responses are approximately 15° and 1 m, when the anomaly reaches its maximum. On the other hand, when the dual-net model is employed following the protocol above, the responses remain closer to the reference signal. In fact, because of its ability to capture variations in plant dynamics, the dual-net model outperforms the offline ANN model in response prediction throughout the simulation and steers the plant to reach the time-dependent reference signals even subject to continuously increasing anomaly.

As shown in Figs. 9b and 10a, for the yaw angle, the dual-net model starts to participate in MPC for anomaly compensation before the 3rd hour, and continues till the end of the simulation. The use of the online ANN to mitigate degraded performance of the altitude control occurs before the 3rd hour, and dominates over the offline ANN from the 4th hour till the end. As a result, improved performance by the dual-net model is evident. It is clearly shown that at the end of the simulation the actual plant responses of both the yaw angle and the altitude produced by the online updated ANN-based MPC match the reference signals very well. To quantitatively characterize the performance of the dual-net model relative to the offline ANN, the numerical errors are computed and presented in Table 3, where the prediction error denotes the deviation between the ANN prediction and the actual plant response. The control error refers to the discrepancy between the prescribed reference signal and the actual plant response. The values listed in the table are computed by taking the absolute mean of the entire error vectors collected throughout the simulation. It is found that the prediction errors and the control errors of using the offline ANN in the presence of anomaly are approximately 4× greater for the yaw angle and 2× greater for the altitude than that of the dual-net model. Both graphical and numerical results verify the feasibility of recovering the performance by updating the online ANN during the operation subject to gradually increasing anomaly.

5 Conclusion

A methodology is proposed to integrate the dual-net model, which consists of the offline and online ANNs, and the model predictive control (MPC) to compensate for the degraded performance caused by slow-paced, continuously growing anomalies in mechanical systems. The foremost novelty lies in the combination of the dual-net model with MPC and the dual-thread decision maker to independently determine and organize the online ANN model updating and the model switch for MPC. The new elements proposed will improve the online learning/updating efficiency and ANN model robustness, and hence, opening up new possibilities to realize operational autonomy for mechanical systems with anomalies on the computing resource-limited platform.

Table 3 Comparison of control and prediction errors between the offline ANN and the dual-net model

		Offline ANN	Dual-net
Yaw angle	Control error (°)	11.89	2.27
	Prediction error (°)	8.01	1.81
Altitude	Control error (m)	1.115	0.426
	Prediction error (m)	0.375	0.159

The ANN system identification/modeling based on the MLP is used to construct the offline baseline model, and further improved in prediction accuracy by the GA to select the optimal network structure and hyperparameters, including the time window size for input and output delays and the hidden layer size, and also the training algorithm. Such an optimized MLP is used to initialize another copy of the online ANN model, which along with the offline ANN model forms the aforementioned dual-net model and will be updated online as necessary. The dual-net model is then combined with the MPC for online synthesis of control actions to be applied to the physical plant. Under the dual-thread decision maker framework, new ANN updating and switch schemes for MPC are proposed. That is, when the ANN prediction accuracy is worse than the prescribed threshold value, the system is triggered to accumulate the operational data for a specified period of time followed by online ANN training using the accumulated data, in which the structure of the online ANN remains unchanged and only the weights are updated. Both the offline and the online ANNs run in parallel throughout the simulation and are compared with the actual plant response, and the one exhibiting better prediction accuracy is selected for MPC prediction in the next horizon. Finally, the case study of the unmanned quadrotor model is undertaken to verify the proposed methodology through numerical simulation. The dual-thread decision maker and the dual-net model demonstrate salient performance in both the accuracy of predicting the actual plant response and the quality of system control subject to growing anomaly. In summary, the updated ANN-based MPC outperforms that solely based on the offline ANN in the presence of anomaly as manifested quantitatively by 4× and 2× reduction in the control and the prediction error. The results verify the feasibility of compensating the degraded performance caused by the shifts in system dynamics.

Despite salient results, there are several limitations in the current method for ANN updating. The authors implemented several techniques to address the overfitting issue. However, there are still few spikes of errors remaining in the predicted results. This will be more critical when the online data is even less diverse. Therefore, switching model for MPC synthesis is used as an additional means of security to ensure desirable MPC performance and stable operation. The root of the overfitting issue is attributed to the large number of fitting parameters (about 1500) during online training. Nonetheless, most of the anomaly scenarios (e.g., loose-fitting, wearing, fatigue, and other) occur in a gradual manner and cause incremental variations in system dynamics. Therefore, updating the entire network weights each time may be unnecessary in terms of both resource usage and model quality. The future research will focus on further investigating and mitigating these issues.

Acknowledgements This research was sponsored by the DoD/Army Research Laboratory (ARL) under the Contract Number W911QX-18-P-0180. The authors would like to acknowledge Mr. Eric Mark at the ARL for his support and feedback on the present work.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27
2. Hagan MT, Demuth HB, Jesús OD (2002) An introduction to the use of neural networks in control systems. Int J Robust and Nonlinear Control IFAC-Affil J 12(11):959–985
3. Mohammadzaheri M, Chen L, Grainger S (2012) A critical review of the most popular types of neuro control. Asian J Control 14(1):1–11
4. Draeger A, Engell S, Ranke H (1995) Model predictive control using neural networks. IEEE Control Syst 15(5):61–66
5. Patan K, Korbicz J (2012) Nonlinear model predictive control of a boiler unit: a fault tolerant control study. Int J Appl Math Comput Sci 22(1):225–237
6. Cheng L, Liu W, Hou ZG, Yu J, Tan M (2015) Neural-network-based nonlinear model predictive control for piezoelectric actuators. IEEE Trans Ind Electron 62(12):7717–7727
7. Wang T, Gao H, Qiu J (2016) A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. IEEE Trans Neural Netw Learn Syst 27(2):416–425
8. Akpan VA, Hassapis GD (2011) Nonlinear model identification and adaptive model predictive control using neural networks. ISA Trans 50(2):177–194
9. Pan Y, Wang J (2012) Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. IEEE Trans Ind Electron 59(8):3089–3101
10. Han HG, Zhang L, Hou Y, Qiao JF (2016) Nonlinear model predictive control based on a self-organizing recurrent neural network. IEEE Trans Neural Netw Learn Syst 27(2):402–415
11. Negri GH, Cavalca MS, de Oliveira J, Araújo CJ, Celiberto LA (2017) Evaluation of nonlinear model-based predictive control approaches using derivative-free optimization and FCC neural networks. J Control Autom Electr Syst 28(5):623–634
12. Yan Z, Wang J (2014) Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks. IEEE Trans Neural Netw Learn Syst 25(3):457–469
13. Morari M, Maeder U (2012) Nonlinear offset-free model predictive control. Automatica 48(9):2059–2067
14. Tatjewski P (2014) Disturbance modeling and state estimation for offset-free predictive control with state-space process models. Int J Appl Math Comput Sci 24(2):313–323
15. Sena HJ, Ramos VS, Silva FV, Fileti AMF (2017) Adaptive offset remover based on Kalman filter integrated to a model predictive controller. Chem Eng 57:1093–1098
16. Alexandridis A, Sarimveis H (2005) Nonlinear adaptive model predictive control based on self-correcting neural network models. AIChE J 51(9):2495–2506
17. Kusiak A, Xu G (2012) Modeling and optimization of HVAC systems using a dynamic neural network. Energy 42(1):241–250
18. Vatankhah B, Farrokhi M (2018) Nonlinear adaptive model predictive control of constrained systems with offset-free tracking behavior. Asian J Control 21(5):2232–2244
19. Lam HK, Ling SH, Leung FH, Tam PKS (2001) Tuning of the structure and parameters of neural network using an improved genetic algorithm. In: The 27th annual conference of the IEEE Industrial Electronics Society, 2001. IECON'01, vol 1, pp 25–30. IEEE
20. Curteanu S, Cartwright H (2011) Neural networks applied in chemistry. I. Determination of the optimal topology of multilayer perceptron neural networks. J Chemom 25(10):527–549
21. Zhang R, Tao J, Gao F (2016) A new approach of Takagi-Sugeno fuzzy modeling using an improved genetic algorithm optimization for oxygen content in a coke furnace. Ind Eng Chem Res 55(22):6465–6474
22. Zhang R, Tao J (2016) Data-driven modeling using improved multi-objective optimization based neural network for coke furnace system. IEEE Trans Ind Electron 64(4):3147–3155
23. Puttige VR, Anavatti SG (2008) Real-time system identification of unmanned aerial vehicles: a multi-network approach. JCP 3(7):31–38
24. Tian R, Yang Y, van der Helm FC, Dewald J (2018) A novel approach for modeling neural responses to joint perturbations using the NARMAX method and a hierarchical neural network. Front Comput Neurosci 12:96
25. Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85
26. Patan K (2015) Neural network-based model predictive control: fault tolerance and stability. IEEE Trans Control Syst Technol 23(3):1147–1155
27. Zhang B, Sun X, Liu S, Deng X (2019) Recurrent neural network-based model predictive control for multiple unmanned quadrotor formation flight. Int J Aerosp Eng. https://doi.org/10.1155/2019/7272387
28. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv (CSUR) 41(3):15
29. Goldstein M, Uchida S (2016) A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PLoS ONE 11(4):e0152173
30. Iverson D (2008) Data mining applications for space mission operations system health monitoring. In: SpaceOps 2008 Conference, p 3212
31. Martínez-Heras JA, Donati A (2014) Enhanced telemetry monitoring with novelty detection. AI Mag 35(4):37–46
32. Lavin A, Ahmad S (2015) Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In: 2015 IEEE 14th international conference on machine learning and applications (ICMLA), pp 38–44. IEEE
33. Jeong H, Park B, Park S, Min H, Lee S (2018) Fault detection and identification method using observer-based residuals. Reliab Eng Syst Saf 184:27–40
34. Hundman K, Constantinou V, Laporte C, Colwell I, Soderstrom T (2018) Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. arXiv:1802.04431
35. Ma D, Xia Y, Li T, Chang K (2016) Active disturbance rejection and predictive control strategy for a quadrotor helicopter. IET Control Theory Appl 10(17):2213–2222

36. Jiajin L, Rui L, Yingjing S, Jianxiao Z (2017) Design of attitude controller using explicit model predictive control for an unmanned quadrotor helicopter. In: 2017 Chinese Automation Congress (CAC), pp 2853–2857. IEEE
37. Kuyumcu A, Bayezit I (2017) Augmented model predictive control of unmanned quadrotor vehicle. In: 2017 11th Asian control conference (ASCC), pp 1626–1631. IEEE
38. Cheng H, Yang Y (2017) Model predictive control and PID for path following of an unmanned quadrotor helicopter. In: 2017 12th IEEE conference on industrial electronics and applications (ICIEA), pp 768–773. IEEE
39. ElKholy HM (2014) Dynamic modeling and control of a quadrotor using linear and nonlinear approaches. American University in Cairo

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.