ARL-CR-0842 ● DEC 2019

DEVCOM ARMY RESEARCH LABORATORY

Optimized Artificial Neural Network Model and Compensator in Model Predictive Control for Anomaly Mitigation

prepared by Seong Hyeon Hong and Yi Wang
University of South Carolina
Columbia, South Carolina

and

Jackson Cornelius and Kapil Pant
CFD Research Corporation
Huntsville, Alabama

under contract W911QX-18-P-180

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-CR-0842 ● DEC 2019

Optimized Artificial Neural Network Model and Compensator in Model Predictive Control for Anomaly Mitigation

prepared by Seong Hyeon Hong and Yi Wang
University of South Carolina
Columbia, South Carolina

and

Jackson Cornelius and Kapil Pant
CFD Research Corporation
Huntsville, Alabama

under contract W911QX-18-P-180

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
December 2019	Contractor Report	3 November 2018–3 November 2019

4. TITLE AND SUBTITLE
Optimized Artificial Neural Network Model and Compensator in Model Predictive Control for Anomaly Mitigation

5a. CONTRACT NUMBER
W911QX-18-P-180

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Seong Hyeon Hong, Jackson Cornelius, Yi Wang, and Kapil Pant

5d. PROJECT NUMBER
SBIR A181-034-0726

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of South Carolina
Columbia, South Carolina

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
CCDC Army Research Laboratory
ATTN: FCDD-RLC-EM
Adelphi, MD 20783-1138

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
ARL-CR-0842

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report presents a new artificial neural network (ANN)-based system model that concatenates an optimized artificial neural network (OANN) and a neural network compensator (NNC) in series to capture temporally varying system dynamics caused by slow-paced degradation/anomaly. The OANN comprises a complex, fully connected multilayer perceptron, trained offline using nominal, anomaly-free data, and remains unchanged during online operation. Its hyperparameters are selected using genetic algorithm-based meta-optimization. The compact NNC is updated continuously online using collected sensor data to capture the variations in system dynamics, rectify the OANN prediction, and eventually minimize the discrepancy between the OANN-predicted and actual response. The combined OANN-NNC model then reconfigures the model predictive control (MPC) online to alleviate disturbances. Through numerical simulation using an unmanned quadrotor as an example, the proposed model demonstrates salient capabilities to mitigate anomalies introduced to the system while maintaining control performance. We compare the OANN-NNC with other online modeling techniques (adaptive ANN and multi-network model), showing it outperforms them in reference tracking of altitude control by at least 0.5 m and yaw control by 1°. Moreover, its robustness is confirmed by the MPC consistency regardless of anomaly presence, eliminating the need for additional model management during online operation.

15. SUBJECT TERMS
artificial neural network, model predictive control, anomaly mitigation, genetic algorithm, compensator

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	40	Eric Mark
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER (Include area code) (301) 394-1266

Contents

List of Figures

List of Tables

Acknowledgments

This research was sponsored by the Department of Defense and US Army Combat Capabilities Development Command Army Research Laboratory (ARL) under contract number W911QX-18-P-0180. The authors would like to acknowledge Mr Eric Mark at ARL for his support and feedback on this work.

1. Introduction

Safety-critical systems, such as aircraft, spacecraft, and power plants, are equipped with monitoring subsystems that provide sensor data to evaluate their operational states in real time. This allows prompt detection and response to potential anomalies before they become catastrophic, and extends the in-service life through effective health and usage management. In this context, the application of a data-driven model to analyze the real-time, operational data collected for accurate prediction of the system behavior and performance is of vital importance in engineering domains. Among various data-driven models, artificial neural networks (ANNs) have grown to be one of the most popular approaches to describe complex nonlinear systems (Billings 2013; Han et al. 2018; Lin et al. 2018; Raissi et al. 2018).

ANN models have been combined with model predictive control (MPC) techniques to synthesize robust controllers that enhance the operational autonomy of nonlinear systems. MPC is an optimal control method that configures the control parameters and laws to minimize the difference between the reference signal and the predicted response output of the system using model prediction. MPC also features salient capabilities to incorporate system constraints, making it attractive to engineers from various fields. As a result, ANN-based MPC has found widespread use in numerous applications, including water regulation in a tank unit, operation of a piezoelectric actuator, a stirred tank reactor, a wastewater treatment process, and a parking guidance framework (Patan and Korbicz 2012; Cheng et al. 2015; Han et al. 2016; Wang et al. 2016; Negri et al. 2017; Shin et al. 2018), where a variety of ANN architectures have been examined, including multilayer perceptron (MLP), recurrent neural network (RNN), radial basis function network, and fully connected cascade network.

Despite the popularity and salient performance, ANN-based MPC exhibits deficiencies in disturbance rejection (Yan and Wang 2014). The receding horizon technique may alleviate minor disturbances caused by environmental factors and sensor noises. Nonetheless, in contrast to other feedback-based control techniques, MPC relies on the predictions of the future horizon for prompt responses; therefore, it is subject to more difficulties when handling larger disturbances. In practice, most mechanical and aerospace systems experience a temporally varying, slow-paced degradation, arising from wear, minor failures and damage, corrosion, and others, which cause changes in the system dynamics and deviations of the sensor readings from the model-predicted values. In other words, when the actual system is deployed for operation, the disturbance caused by its degradation and associated mismodeling is almost ubiquitous. As a result, MPC performance will be compromised, giving rise to a nonzero steady-state error (also known as an offset).

Offset-tracking in MPC has been demonstrated through disturbance modeling (Morari and Maeder 2012; Tatjewski 2014; Sena et al. 2017). Most disturbance modeling requires a priori knowledge about the disturbance. In cases where information about the disturbance is not available, a data-driven modeling technique can be applied. In ANN-based MPC, the ANN model may train its weights (or even its architecture) during the operation to enhance prediction accuracy through model updates whenever new sensor data become available (Alexandridis and Sarimveis 2005; Kusiak and Xu 2012; Vatankhah and Farrokhi 2018). The sensor data contain information for quantitatively characterizing the discrepancy between the actual and model-predicted system response, which allows one to adjust the weight parameters properly to accommodate the change. This type of ANN is referred to as an adaptive ANN (AANN), which adjusts to the changes in real time solely based on data from the monitoring subsystem.

Although AANN models have been broadly utilized in many MPC applications, there are several limitations to this approach. First, the structure of the AANN model needs to be compact in size, since it is difficult to update a large number of weight parameters at once during operation. Second, due to its small size, the model can only represent the actual system accurately in a limited range. In other words, the AANN model is not generalized for the entire range of references, inputs, and response outputs that can be encountered in reality. Lastly, online training is vulnerable to overfitting or other training issues, especially when the collected data are limited in diversity and generality.

To mitigate these issues, we present a new ANN-based MPC framework that compensates for degraded performance arising from slow-paced anomalies and dynamic shifts of the mechanical system, such as wear, fatigue, corrosion, and others. This effort includes several novelties. First, the new ANN model is based on a unique architecture that combines an optimized ANN (OANN) and a neural network compensator (NNC) in series to capture temporally varying system dynamics. Second, the OANN features a complex, fully connected MLP that is trained offline using prior nominal data and remains unchanged during online operation. Because of its offline training nature, the key hyperparameters of the OANN are selected using computationally demanding, meta-optimization techniques to achieve excellent predictive performance. The meta-optimization, as presented in various literatures (Lam et al. 2001; Curteanu and Cartwright 2011; Zhang et al. 2016; Zhang and Tao 2017), can be realized using evolutionary algorithms. In this work, we employ the genetic algorithm (GA) to optimize the OANN and seek the optimal ANN hyperparameters in order to minimize the prediction error of the testing data. The rationale to adopting the GA-optimized ANN of a complex structure as the backbone is to improve the accuracy and

generality of the model by training on prior data with sufficient volume and diversity. Thus, the OANN can serve as a robust baseline, trend model that even when subject to significant noises will exceed the ANNs of small sizes when no disturbance is present. Third, the larger structure of the OANN renders it formidable to train/update the entire set of the weight parameters during operation where the anomaly could occur and the system dynamics will shift. To address this deficiency, the NNC of a simple structure is attached at the end of the OANN to adjust its response prediction to match the actual response. Different from the OANN, the NNC is updated continuously online using collected sensor data to capture the variations in the system dynamics. Essentially, the OANN-NNC combination can be deemed as a large network, in which only the last few layers are allowed to update while the preceding layers are frozen. Last, the combined OANN-NNC model is utilized for future horizon prediction to reconfigure MPC online and alleviate the disturbances. Many of the existing ANN-based MPC approaches update the ANN model at every control epoch (Hedjar 2013). Therefore, the ANN model is restricted in size to reduce the number of weight parameters to be trained online, which can give rise to poor prediction accuracy and control performance, and even the generality issue given limited sensor data for training. On the contrary, the offline-trained OANN in this approach does not have such a restriction and encompasses all the necessary features in the model, and therefore, ensures robust, and at least trend, prediction of the system response. The accuracy requirement for prediction and control is satisfied by the online updating of the compact NNC. It should be stressed that the NNC only predicts the discrepancy between the actual and OANN-predicted response, which, in general, varies more mildly in contrast to the response itself, and hence, is easier to model even with limited sensor data.

The remainder of this report is organized as follows. In Section 2, the methodology of the proposed OANN-NNC–based MPC for disturbance rejection is introduced. The ANN modeling and optimization using GA, NNC structure, and MPC development are described thoroughly. In Section 3, a case study based on numerical simulation is performed to verify the proposed framework. The system model of interest, anomaly implementation, various compensators, and benchmark ANNs for comparison are all elucidated in detail. The results of the GA-selected OANN and MPC performances obtained by different ANNs are presented in Section 4. The report is concluded with a summary and future work in Section 5.

2. Methodology

Figure 1 illustrates the proposed methodology and the OANN-NNC–based MPC framework for anomaly mitigation, where u is the system input (or control signal); y is the system response/output; y_n is the OANN-predicted output; y_p is the NNC output, which can be treated as an adjustment of y_n to address the disturbance caused by the system anomaly/degradation; and y_r is the reference signal. As discussed previously, the entire model prediction in our framework is divided into two submodel processes concatenated in series (i.e., the OANN of the complex structure including ~1,000 trainable weight parameters and a very simple NNC model with ~2 to 4 parameters). The former will be trained offline using adequate, nominal, anomaly-free data and will remain constant during online operation. Hence, the computationally demanding meta-optimization can be adopted to search the optimal hyperparameters and structure of the OANN. Only the NNC will be updated at each epoch to capture the induced disturbance and bridge the gap between the actual y (dashed line in Fig. 1) and the ANN model-predicted system response y_n during online operation where slow-paced anomalies could occur. The adjusted system response y_p for the future horizon will be used to reconfigure the MPC for enhanced performance. Throughout the numerical simulation, it is assumed that that the sensor measurements are fault-free because an accurate data set is necessary for training the data-driven model, although they are contaminated by the noises at the appropriate level to verify that the compensator is robust enough to handle both the noises and the anomaly-induced disturbance.

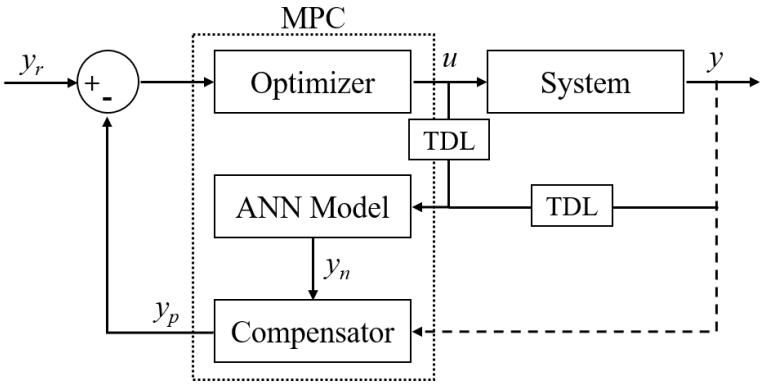

Fig. 1 Flowchart of ANN-based MPC with a compensator

2.1 System Identification and OANN

Similar to several reported efforts (Cheng et al. 2015; Han et al. 2016), our system model is formulated as a nonlinear autoregressive moving-average exogenous (NARMAX), which simply uses the delayed control signals u and the feedback

outputs y of the system as the overall network inputs, and establishes a functional mapping relationship between them to capture the system behavior (Chen and Billings 1989). In this report, the mapping is approximated by the ANN model, which can be realized through various neural network architectures. RNNs have gained popularity in MPC applications for their ability to accurately predict multiple time steps for the entire horizon (Akpan and Hassapis 2011; Pan and Wang 2012). On the other hand, an MLP with one hidden layer is considered a baseline structure of the system model. MLPs are naturally one-step-ahead predictors and may not be very suitable for predicting a long horizon. However, in the present effort, MLP is adopted because the GA-based meta-optimization is adequate to select the optimum MLP structure to minimize the validation error, resulting in a highly accurate system model by selecting appropriately large delayed values, which replaces the recursive computations in an RNN. Since our OANN model is trained offline with sufficient data, it is anticipated to have the appropriate complexities selected by GA, leading to enhancements in both accuracy and generality. Specifically, the number of hidden neurons, the delay window size of both the inputs and outputs, and the training algorithm are determined by GA (Whitley 1994).

GA is a randomized optimization technique that can find the optimum solution without needing to compute gradients. It initially creates random gene sequences with information about the hyperparameters. These gene sequences are evaluated accordingly by training the MLP with a combination of hyperparameters within each gene sequence and compute the mean squared error (MSE) on the predictions of the validation set. Then the gene sequences go through crossover, mutation, and selection to generate the next set of gene sequences. This process repeats until the optimized solution (a gene sequence with the highest score) is found or a maximum number of generations is reached. In this effort, gene sequences are represented as double vectors and Gaussian mutation; scattered crossover and stochastic uniform selection methods are applied.

Within GA optimization is the ANN model to describe the aforementioned mapping relationship in NARMAX. Equation 1 shows the mathematical representation of the MLP model:

$$\begin{aligned} y_n(k) &= W^{(2)} \tanh(W^{(1)} X(k) + b^{(1)}) + b^{(2)} \\ X(k) &= [u(k) \quad u(k-1) \quad \cdots \quad u(k-d_u) \quad y(k-1) \quad \cdots \quad y(k-d_y)]^{\mathrm{T}} \end{aligned}, \quad (1)$$

where, d_u and d_y are the input and output delays, and ($W^{(1)}$, $b^{(1)}$) and ($W^{(2)}$, $b^{(2)}$) are the input-to-hidden and hidden-to-output weight matrices, respectively. As seen from the equation, the hyperbolic tangent is the activation function of the hidden

layer and no activation function is applied at the output layer, indicating that this is a regression problem. The list of all the hyperparameters of the offline OANN model to be selected by GA are shown in Table 1, along with their types and search ranges. The number of system input and output parameters, respectively, denoted by n_u and n_y, determine the total number of inputs to the corresponding OANN model, which is $(n_u \times (d_u + 1)) + (n_y \times d_y)$. The size of the hidden layer is also determined by GA, as shown in the table. Both delays and hidden layer neurons are bounded in range. If the optimal values selected are close to the upper limits, then the range must be extended to mitigate the issue of poor hyperparameter selection due to empirical specification of the bounds. Usually, larger delays and hidden neurons tend to improve model accuracy only until a certain limit is reached. Moreover, the larger network will require more training time and resource usage. In addition to the size of the network, the optimal training algorithm is also selected. Twelve different training algorithms, such as gradient descent, Levenberg–Marquardt (LM), Bayesian regularization, Broyden–Fletcher–Goldfarb–Shanno quasi-Newton, and others are compared by GA, and the algorithm yielding balanced performance is chosen.

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

Hyperparameter	Type	Range
Input delay (d_u)	integer	[1, 30]
Output delay (d_y)	integer	[1, 30]
Hidden layer neurons	integer	[1, 50]
Training algorithm	integer (list index)	[1, 12]

2.2 Neural Network Compensator

The intent of the NNC is to capture the disturbance caused by slow-paced system degradation and the drift in dynamics with a simple model structure to avoid training failures and reduce the computational load during online operation. The NNC-predicted correction superimposed on the previous OANN prediction will accurately represent the latest dynamics of the system.

Figure 2 displays the two proposed NNC structures, which are, respectively, a single-layer and double-layer model. The corresponding equations are

$$y_p - y_n = C_0 + C_1 y_n \tag{2}$$

$$y_p - y_n = C_2 + C_3 \tanh(C_0 + C_1 y_n), \tag{3}$$

where (C_0, C_1, C_2, C_3) are the weight parameters of the NNC that need to be computed online using sensor data collected during operation. The single-layer NNC in Eq. 2 and the double-layer NNC in Eq. 3, respectively, have two and four weight parameters to train. There are several points of note in the previous NNC formulation. First, both NNC models are heuristic and only take the offline ANN-predicted response y_n as the input. Such a model architecture attaching the NNC to the offline ANN can be essentially considered as a large network that only allows the last one or a few layers to be updated while freezing the other weight parameters in the preceding feature extraction layers. Although heuristic, the NNCs were found to perform very well to capture the dynamic drifts and disturbance in our case studies of the quadrotor. For different dynamics systems, NNCs in Eqs. 2 and 3 may need to be adapted. Second, both NNCs output the predicted disturbance, $d_p = y_p - y_n$. Recall that y_p is the combined prediction from the offline OANN and the NNC, and will be used for MPC. Therefore, the NNCs are intended to reconcile the difference between the actual and OANN predicted outputs (i.e., $y - y_n$, leading to enhanced agreement between y and y_p). Third, the NNCs estimate d_p instead of directly predicting the actual response y. This is because, in general, the variation of $y - y_n$ is milder than that of y and can be captured by a more compact model structure. The advantage of updating a compact model is also apparent. During the operation, the collected sensor data have limited diversity as the reference signals may only vary within a small range for a specific operation/mission. If a model of great complexity is trained with limited, biased data, there will be a high possibility of model overfitting. The compact model structure like the NNC is less prone to overfitting. Moreover, by introducing the NNC to capture the dynamic disturbance, the OANN model can actually adopt a more complex structure to incorporate all key nonlinear factors/features, and hence, make the entire modeling and updating scheme more efficient and robust. Last, the stochastic gradient descent method is employed to compute the weights of the NNC at every epoch, which can be readily accomplished using backpropagation. The updated NNCs along with the OANN model are then utilized by MPC to compute the receding horizon. In short, the NNC is a more efficient and robust approach than updating the entire ANN model during online operation.

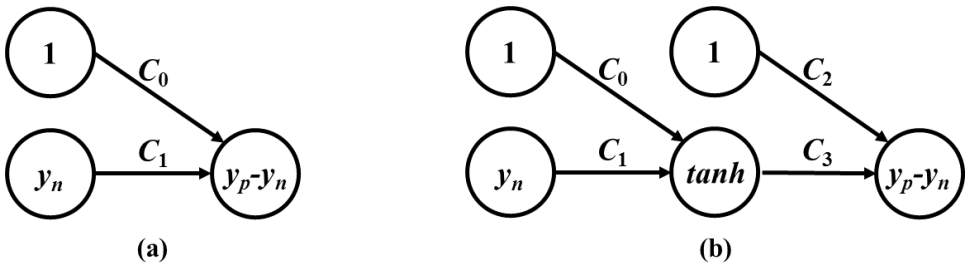

Fig. 2 Proposed structure of the NNC: a) single layer and b) double layer

2.3 Model Predictive Control

MPC typically comprises the cost function, the optimizer, and the system model. In this report, the system model to predict the response (y_p) is represented by the combination of the OANN and the NNC, respectively, described in Sections 2.1 and 2.2. MPC takes the predicted response (y_p) over a specified time horizon and the reference response (y_r) as inputs, and generates the optimal control signal by minimizing the cost function J:

$$J = \sum_{j=N_1}^{N_2} (y_r(k+j) - y_p(k+j))^2 + \rho \sum_{j=1}^{N_u} (u(k+j) - u(k+j-1))^2 , \quad (4)$$

where N_1 is the minimum costing horizon, N_2 is the maximum costing horizon, N_u is the control horizon, and ρ is the control weighing factor. The first and the second terms in the cost function are referred to as the error and stabilizer terms, respectively. The error term is simply the MSE between the reference signal and the response predictions adjusted by the NNC. The stabilizer term is the MSE between the consecutive control signals. In other words, ρ decides the change rate of the control signal u. If ρ is small, rapid changes in the consecutive control signal are allowed. The goal of MPC is to compute $[u(k+1),\ldots,u(k+N_u)]$ such that it minimizes the cost function for every control epoch. Given accurate prediction y_p by the model, in general, a large N_2 and N_u will boost control performance, while increasing the computational load and compromising the speed of control synthesis. However, determining these optimal control parameters is not the focus of this effort since they are independent from the disturbance caused by system degradation. For our simulation study, N_1, N_2, N_u, and ρ are selected empirically to be 1, 4, 3, and 1e-3, respectively. In addition, the stability of ANN-based MPC has been proven by Patan (2015) using the Lyapunov synthesis method. That is, in order to meet the stability criterion, the optimization of Eq. 4 must be performed with the following constraints:

$$\begin{aligned} & y_r(k+N_2+j)-y_p(k+N_2+j)=0, \quad \forall j\in[1,N_c] \\ & \underline{u}\le u(k+j)\le \bar{u}, \quad \forall j\in[0,N_u-1] \\ & u(k+N_u+j)=u(k+N_u+j-1), \quad \forall j\ge 0 \end{aligned} \quad , \qquad (5)$$

where N_c is the horizon constraint; and $\underline{u}$ and $\bar{u}$ are the lower and upper control bounds, respectively. Meeting these requirements, nonlinear MPC is asymptotically stable if $\rho \neq 0$ and $N_c = \max(n_y+1,\ n_u+\tau+1+N_u-N_2)$, where τ is the time delay.

The utilization of the combined OANN and the NNC for MPC and the order of executing each functional block at each control epoch are shown in Fig. 3. The OANN first uses the tapped delay line (TDL) of the inputs and the outputs to predict $y_n(k)$ at the current time step (as shown in Block 1). The discrepancy between the OANN-predicted $y_n(k)$ and actual system response $y(k)$ arising from the slow-paced anomaly will be utilized to re-train and update the weights C_0, C_1, C_2, and C_3 in the NNCs as shown in Block 2. Then, the updated NNC will be deployed to adjust the OANN-prediction y_n as shown in Block 3, yielding enhanced prediction of the system response in the horizon from N_1 to N_2 (i.e., $y_p(k+N_1)$, …, $y_p(k+N_2)$). Finally, the NNC-adjusted prediction y_p is supplied to the MPC to reconfigure the control input in the next epoch $u(k+1)$.

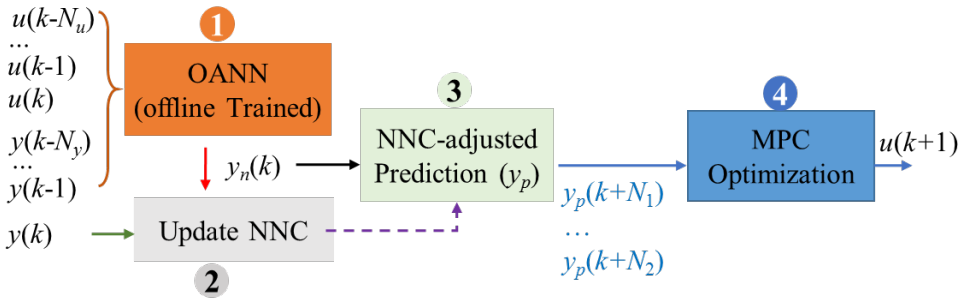

Fig. 3 Procedure to combine the offline OANN and the NNC for MPC

3. Case Study and Numerical Experiment

To verify the proposed ANN-based MPC with the NNC, tracking and regulation of an unmanned quadrotor is numerically simulated, particularly, for yaw angle and altitude. Our specific aim is to ensure that the NNC captures the drift in the system dynamics arising from the disturbance and eliminates the associated steady-state error during operation through MPC reconfiguration. This particular control problem has been selected for investigation since an accurate physics-based quadrotor model is easily attainable in the public domain and can be used as a

surrogate for the physical plant. MPC for an unmanned quadrotor has been demonstrated by various groups (Ma et al. 2016; Jiajin et al. 2017; Kuyumcu and Bayezit 2017; Cheng and Yang 2017; Zhang et al. 2019). Among them, Zhang et al. (2019) applied ANN-based MPC for the formation flight of multiple unmanned quadrotors, which utilized the RNN structure along with a feedback compensator. Since the RNN parameters are definitely more numerous than those in our NNC that need to be updated at each epoch, the RNN structure has to be simple to allow online model adaption. Therefore, the RNN may be only applicable to a specific operating condition. On the other hand, our goal is to obtain the generalized system model that may represent the quadrotor dynamics in a broad range of operation. As discussed previously, our strategy is to divide the model into two parts, the GA-optimized ANN applicable to a wide range for the generalized solution and the NNC to capture the changes in the system dynamics as a result of the slow-paced degradation and disturbance.

3.1 System Model

The equations of dynamic motion used in the present work for a typical quadrotor are given (Bouabdallah 2007):

$$
\begin{aligned}
\ddot{\phi} &= \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} - \frac{J_r}{I_{xx}}\Omega_r\dot{\theta} + \frac{l}{I_{xx}}u_2 \\
\ddot{\theta} &= \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} - \frac{J_r}{I_{yy}}\Omega_r\dot{\phi} + \frac{l}{I_{yy}}u_3 \\
\ddot{\psi} &= \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} + \frac{l}{I_{zz}}u_4 \\
\ddot{x} &= -\frac{u_1}{m}(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta) \\
\ddot{y} &= -\frac{u_1}{m}(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi) \\
\ddot{z} &= g - \frac{u_1}{m}(\cos\phi\cos\theta)
\end{aligned}
\tag{6}
$$

where (θ, ϕ, ψ) and (x, y, z) represent rotational and translational motions, respectively; (I_{xx}, I_{yy}, I_{zz}) are the area moment of inertias about (x, y, z) axis; 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. (u_1, u_2, u_3, u_4) are the inputs to Eq. 6 and can be computed by multiplying the angular velocities of each rotor with the transformation matrix as shown in Eq. 7:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ 0 & -K_f & 0 & K_f \\ K_f & 0 & -K_f & 0 \\ K_m & -K_m & K_m & -K_m \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \tag{7}$$

where K_f and K_m are the aerodynamic force and moment constants of blades, respectively, and (Ω_1, Ω_2, Ω_3, Ω_4) are angular velocities of each rotor. The values of model parameters (i.e., (I_{xx}, I_{yy}, I_{zz}), J_r, l, m, K_f , and K_m) are obtained from ElKholy (2014) as listed in Table 2.

Table 2 Quadrotor parameter values

Parameter	Value	Unit	Parameter	Value	Unit
I_{xx}	7.5×10^{-3}	$kg\,m^2$	l	0.23	m
I_{yy}	7.5×10^{-3}	$kg{\cdot}m^2$	m	0.7	kg
I_{zz}	1.3×10^{-3}	$kg{\cdot}m^2$	K_f	3.13×10^{-5}	$N{\cdot}s^2$
J_r	6×10^{-5}	$kg{\cdot}m^2$	K_m	7.5×10^{-7}	$N{\cdot}m{\cdot}s^2$

3.2 System Degradation

The system degradation and shifts in the quadrotor dynamics are mimicked by continuously, temporally varying the deformation/wearing of the blades as a result of the long-term exposure to harsh environments. This will change the force and moment created by each blade (or rotor), thus affecting the entire dynamics of the system. Specifically, Eqs. 8 and 9 represent the propulsion force and moment of the rotor, respectively:

$$F_i = K_f \Omega_i^2 \ , \quad K_f = \frac{1}{2} \rho A C_T r^2 \tag{8}$$

$$M_i = K_m \Omega_i^2 \ , \quad K_m = \frac{1}{2} \rho A C_D r^2 \ , \tag{9}$$

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 force and moment coefficients, and r is the radius of the blades. According to the equations, the aforementioned anomaly scenario can be imitated by varying the aerodynamic force and moment constants of the blades, K_f and K_m. Increasing K_f refers to creating more force with the same angular velocity and vice versa. Similarly, the moment will decrease as K_m decreases, even when the angular velocity remains constant. The changes in K_f and K_m will affect the altitude and the

yaw angle tracking, respectively. Therefore, throughout the case study, the slow-paced deformation/wearing of the blades is realized by tuning the K_f and K_m parameters.

3.3 Feedback Compensator

In this analysis, the well-established feedback compensator (FBC) for system control serves as a benchmark, to which the proposed NNCs are compared. The equation of the FBC is as follows:

$$\begin{aligned} d(k) &= k_c(y(k) - y_n(k)) + d(k-1) \\ y_p(k+1) &= y_n(k+1) + d(k) \end{aligned}, \tag{10}$$

where k_c is the disturbance gain. It can be seen that the FBC behaves like the traditional proportional integral control, since all the errors are integrated. The error integration scheme eliminates the steady-state bias. The major difference between the NNC and FBC is manifested by 1) the NNCs only taking y_n as the input and 2) the NNC attempting to identify and update the weight parameters to establish the quantitative relationship between the inputs and the disturbance in the system; whereas, FBC estimates the disturbance simply by accumulating the error and the control constant k_c remains constant during online operation. In Section 4, both NNCs are compared with the FBC and to one without any compensator.

3.4 Benchmark Models for Comparison

The proposed approach that combines the GA optimized-ANN with the NNC (OANN-NNC) is first compared with other models reported in the literature. Although control was not considered, Puttige and Anavatti (2008) proposed a variant of the ANN model for unmanned aerial vehicle system identification, which is termed multi-network (multi-net) and can be used for performance benchmarking. They constructed the multi-net model by connecting an online ANN and an offline ANN in parallel with a decision maker, which allows the system to switch to the best-performing ANN model during operation. The online ANN updates itself periodically during operation, often at every epoch. Usually, the online ANN is restricted in size, since updating a large number of weight parameters at every time instant is computationally demanding, which may preclude it from hardware implementation and usage in real time. The offline ANN refers to an ANN that is trained offline and will never be updated during operation. Thus, the offline ANNs do not have the restriction in size, as they can be trained on the more powerful computing platform with sufficient nominal data and their weight parameters remain constant during the operation. The online ANN is usually

biased due to its simple structure, and the offline ANN can be utilized when it is more accurate than the former. However, the offline ANN is vulnerable to internal disturbance caused by variations in the system dynamics, leading to significant errors in MPC, and in this case, the online ANN may be a better alternative. Therefore, the multi-net model selects one of the two ANNs based on the prediction accuracies for MPC during online operation. As originally reported by Puttige and Anavatti (2008), both the offline and online ANNs used herein for comparison adopt 3 input and output delays along with 12 and 4 hidden neurons, respectively, and a batch size of 5 is used to update their online ANN.

Another benchmark model used for comparison in this report is the AANN. It is a standalone ANN structure and updated throughout the operation. The use of the term AANN also distinguishes it from the online ANN in the multi-net model. The AANN is also restricted in size to allow rapid, periodic updating of its weights, and in this work, has one input, two output delays, and six hidden neurons, and its weight parameters are updated at every epoch.

In summary, we compare our combined OANN-NNC against both the AANN and the multi-net models in terms of MPC performance. All three ANNs are MLPs and their composition is shown in Fig. 4, in which the components in red represent the weight parameters that will be updated during operation and those in blue are keep constant.

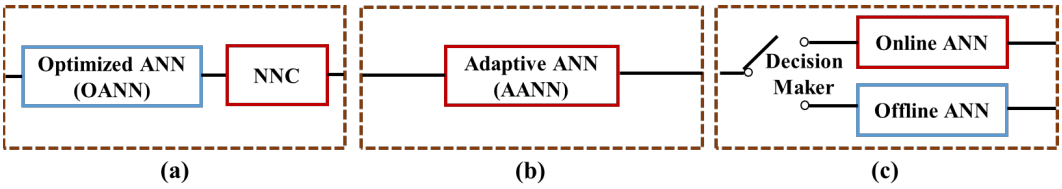

Fig. 4 Different system models utilized in MPC for performance comparison: a) GA-optimized ANN with NNC (OANN-NNC), b) AANN, and c) multi-net

4. Result and Discussion

In this section, we first present the results of the ANN meta-optimization by GA from Section 4.1. The OANN is used as the baseline model and combined with various compensators. The control performance of the proposed NNC is compared with other compensators in Section 4.2. In Section 4.3, the proposed OANN-NNC is compared with the other two ANN model architectures (i.e., AANN and multi-net) applicable to online model updating for the disturbance-free and disturbance-rejection cases.

4.1 ANN Meta-optimization by GA

We first describe the result of using GA to select the hyperparameters of the ANN that is trained offline and used in our combined OANN-NNC model. To generate the training data, inputs of the prescribed random step profile are applied to the actual physics-based model described in Section 3.1. The output data of the yaw angle and altitude are collected accordingly, and their values fall in the range of (-5π 5π) rad and (-100 to 100) m, respectively. Gaussian noise is added to the data with an intensity of 0.1 rad/s for the yaw angle and 1 m/s for the altitude. The data pairs of the prescribed inputs and the outputs are then organized in the form complying with the NARMAX formulation and then used for the ANN model. GA is then implemented as a wrapper around the MLP model to determine the optimal hyperparameters to achieve balanced accuracy and generality through training.

The hyperparameters under consideration in this report include the window size of the input and output, hidden neuron size, and training algorithm. Again, the cost function for the optimization is the MSE on the prediction of the validation set. For each optimization trial, populations of 30 designs are processed for 20 generations. For a single population, a specific MLP decoded from the gene sequences needs to constructed, trained, and validated. Therefore, for each generation, 30 MLPs of different structures are analyzed. Fortunately, as the generation increased, the populations converge to designs within a smaller bound and minor variation. Given the randomness in the ANN training and initial GA population creation, the optimal hyperparameter set may not be unique. For example, even with the same MLP model structure, different weights and performance scores are expected when multiple instance of the model are initialized with various weights for training. Nevertheless, during GA meta-optimization, as the generation increases, the range of the hyperparameters starts to narrow down, and the parameters at the last generation are actually reliable and reproducible to yield excellent prediction results.

The input and output delays converge close to 10 and 20, respectively. This implies that the delay values need to be set much larger than the delays defined for the AANN. Moreover, the output delay has more impact on ANN training than the input delay. The number of the hidden layer neurons shows more variance compared to the delays, ranging from 25 to 40 during the optimization. This also indicates that our ANN performance is less susceptible to the number of the hidden layer neurons within the range. In addition, the most suitable training algorithm is found to be LM. All the populations converged to the LM method after a few generations, implying that it is superior to the others for this particular problem. The results of the GA-based meta-optimization are summarized in Table 3, and the

final choice for the MLP structure is 10 for the input delay, 20 for the output delay, 36 for the hidden layer neurons, and the LM method for the training algorithm. Consequently, the MLP models for the yaw and altitude dynamics will each have 40 input nodes, 36 hidden nodes, and 1 output node. Note that the GA-based meta-optimization results depend on the specific systems under consideration. Moreover, the noise magnitude in the data has a significant effect on hyperparameter selection. We found that the intense noise in the data tends to require a larger number of delay values.

Table 3 GA-selected hyperparameters for the ANN model

	Input delay	Output delay	Hidden neuron	Train algorithm
Confined range	~10	~20	25~40	LM
Final selection	10	20	36	LM

In summary, the structure of different ANNs to represent the system model for comparison is listed in Table 4. The ANN component of the combined OANN-NNC has the largest model structure (the delay and hidden neurons), followed by the multi-net and the AANN, while the NNC has the smallest number of weight parameters (2 or 4) to update online, as discussed in Section 2.2. This indicates that in the offline training, the OANN model may require the largest number of data sets and training time in exchange for the least effort to update the NNC to capture the system degradation and shift in dynamics during operation. On the contrary, the AANN and multi-net models may require less effort in offline training at the cost of updating about 50 parameters online in response to the varied system dynamics. Both OANN-NNC and AANN are updated at every period (i.e., 0.1 s), in order to respond rapidly to the changes in the system. However, for the online ANN part of the multi-net, the updating period is set to 0.5 s and is consistent with the model previously reported by Puttige and Anavatti (2008), which uses a batch size of 5 for updating.

Table 4 Structure of various ANN models for comparison

	OANN-NNC		AANN	Multi-net	
	OANN	NNC		Offline	Online
Input delay	10	0	1	3	3
Output delay	20	0	2	3	3
Hidden neurons	36	0 or 1	6	12	4
Fixed parameters	1513	0	0	157	0
Parameters to update	0	2 or 4	49	0	53
Batch size	NA	1	1	NA	5
Updating period	NA	0.1 s	0.1 s	NA	0.5 s

4.2 NNC Validation

In this section, we exclusively investigate the effects and performance of various compensators for disturbance rejection by the way of numerical simulation, including the two NNCs as discussed in Section 2.2, the FBC in Section 3.3, and their comparison to the scenario without any compensator for disturbance rejection. The OANN obtained through GA meta-optimization described in Section 4.1 is concatenated with these compensators and remains unchanged during operation. The degradation of the system is mimicked by prescribing the temporally varying aerodynamic force and moment constants (K_f and K_m). As discussed previously, changes in K_f and K_m alter the system dynamics of the altitude and yaw angle, respectively, and compromise control performance. Variations of these aerodynamic constants and their effects on MPC are shown in Fig. 5. For the first 60 s, their values are changed gradually, and at the 70th second, there is an abrupt change. In the figure, NNC1 (in green) and NNC2 (in brown), respectively, refer to single- and double-layer NNCs. It clearly shows that if the OANN trained offline using nominal data is not updated and there is no compensator to correct the prediction, the offset error appears in both the altitude and yaw response outputs. However, when the NNCs are combined with the OANN and updated online, the offset errors can be effectively mitigated. Moreover, during the period of the gradually increasing anomaly (in the first 60 s), the disturbance can be rejected as if there is no change in the system dynamics. In other words, when the model can be updated accurately in a rapid manner with sufficient data to accommodate the dynamics variation, no system degradation is visible. However, at the 70th second, when the abrupt anomaly is applied with a larger magnitude, both NNCs need sufficient time and data to learn and adapt to the new system dynamics. This implies that for more serious system degradation, the compensators will take longer to reject the disturbances.

For comparison, the FBC is also grafted onto our OANN model and simulated, which is denoted in cyan in Fig. 5. The results confirm that the NNCs perform almost the same as the FBC and can actually be utilized in lieu of the latter without compromising performance. One notable advantage of the NNC over the FBC is, since the NNC is an extension of the ANN model, we can obtain the new system model. In contrast, the FBC removes the offset error without updating the model, and thus, any information about the changes in the system remains unknown. The tracking errors of different compensators are also quantitatively compared in Table 5, which shows that the root-mean-square error (RMSE) is reduced dramatically by a factor of 2 (altitude) and 5 (yaw angle) when compensators are used. In addition, subject to the random noises applied, the performance of all compensators is also similar.

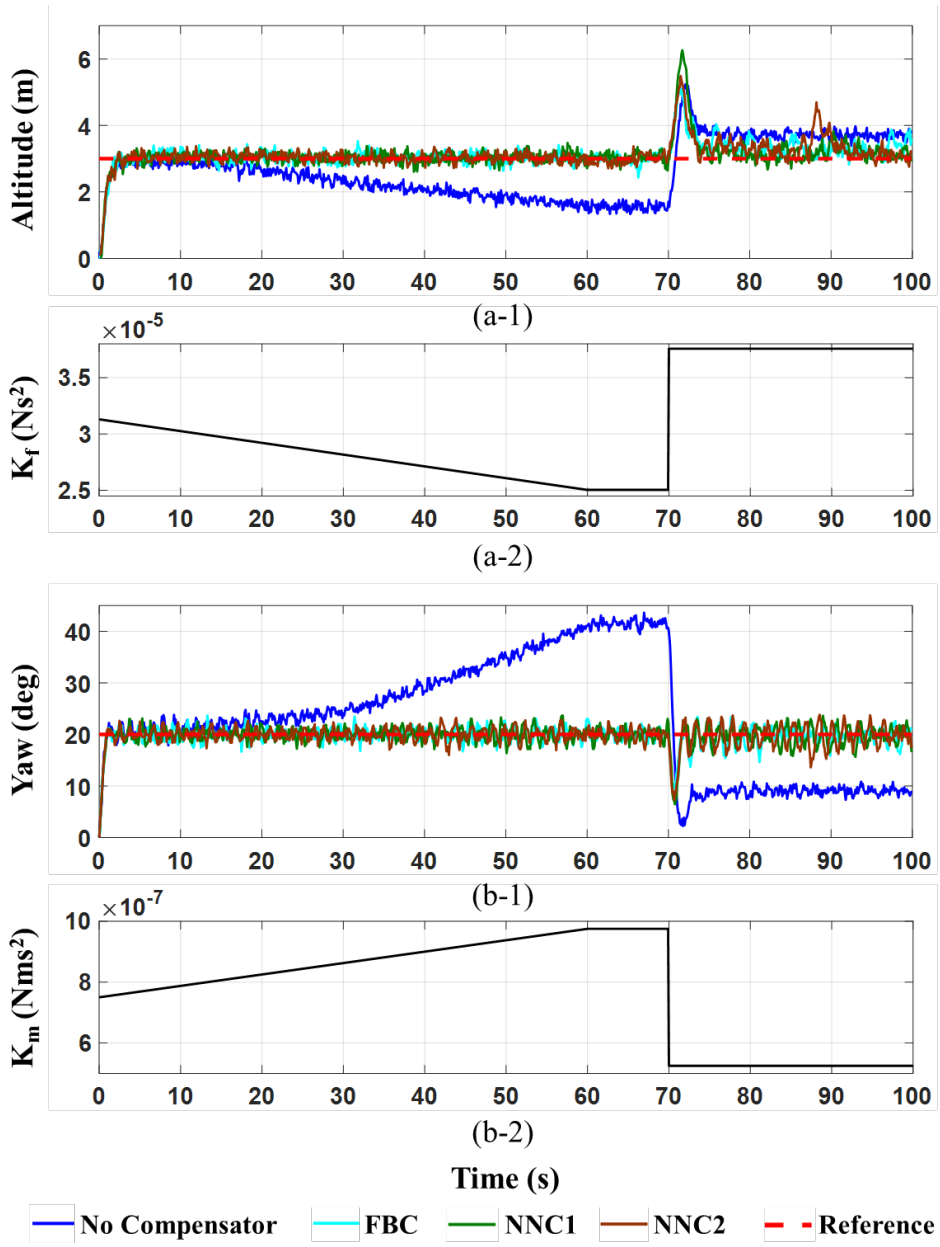

Fig. 5 MPC performances of different compensators with presence of anomaly: a) altitude and b) yaw angle

Table 5 RMSE of MPC performances with various compensators

	No compensator	NNC1	NNC2	FBC
Altitude (m)	0.93	0.46	0.43	0.42
Yaw (°)	11.84	2.10	2.14	1.97

Since it makes no difference in performance, yet has a smaller number of parameters to update, NNC1 is selected to constitute the OANN-NNC model for the analysis that further investigates the performance of various network architectures, including the AANN and multi-net that adjust the system models without using compensators.

4.3 OANN-NNC Validation

In this section, the MPC performance of various model architectures, including the AANN and multi-net are compared with our OANN-NNC. Studies under two kinds of circumstances are performed. First, there is no disturbance and model updating is actually not necessary; and second, the anomaly-induced disturbance is present and the model needs to be updated to reject the disturbance.

4.3.1 Disturbance-Free Scenario

This analysis scrutinizes the generality and robustness of these architectures, as they are proposed primarily to reject the anomaly. The simulations are performed with and without model updating to observe its effects on MPC performance when the disturbance is absent. MPC for all network architectures is carried out with the same control parameters listed in Section 2.3, except for the ρ value in the multi-net case since better performance was observed by reducing it down to 1×10^{-4}.

At first, the reference signals of the step and sinusoidal profiles are used for both altitude and yaw tracking, and the results are displayed in Fig. 6. The left column displays the results when the system models are not updated and the right column shows the updated system models. The control performances based on these models are quantitatively compared in terms of RMSE in Table 6.

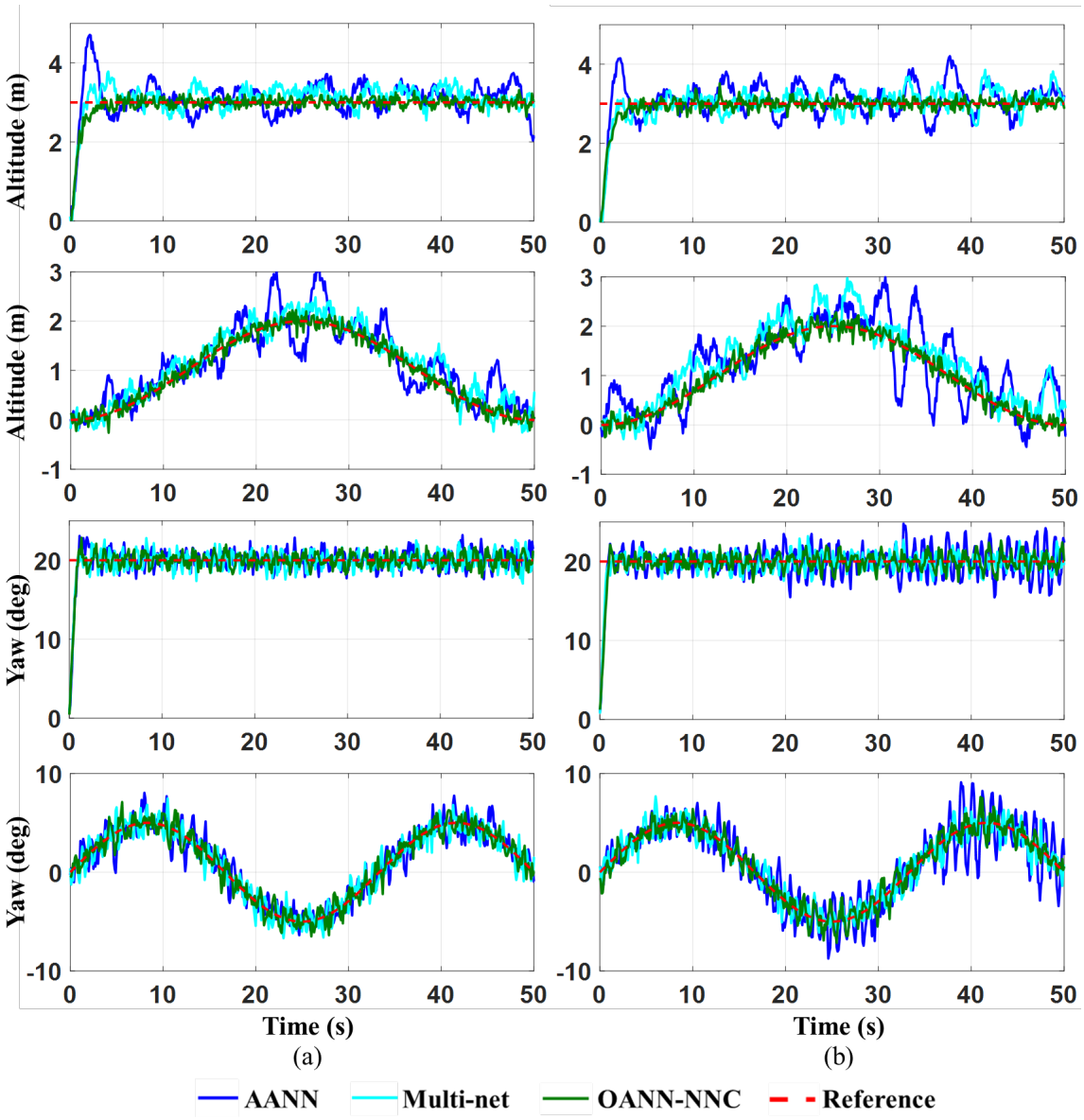

Fig. 6 MPC performance of different system models in the anomaly-free case: a) without and b) with model updating

Table 6 MPC performances of different system models in RMSE in the anomaly/ disturbance-free case

		Without update		With update	
		Altitude (m)	**Yaw (°)**	**Altitude (m)**	**Yaw (°)**
AANN	Step	0.47	1.56	0.51	2.27
	Sinusoidal	0.39	1.16	0.50	2.07
Multi-net	Step	0.35	1.48	0.39	1.43
	Sinusoidal	0.26	0.97	0.37	1.01
OANN-NNC	Step	0.26	1.37	0.26	1.47
	Sinusoidal	0.11	0.84	0.13	0.95

As expected, since there is no anomaly present, steady-state error is not visible throughout the simulation. The OANN-NNC clearly outperforms the other two models, as shown in Fig. 6a. The green curve, which illustrates the output of the OANN-NNC, is closest to the reference signal in red compared to the other two curves in blue and cyan that, respectively, represent the AANN and multi-net. Table 6 also shows that the error values of OANN-NNC for both the step and sinusoidal references exhibit smallest values, which are, respectively, 0.26 m (altitude) and 1.37° (yaw), and 0.11 m (altitude) and 0.84° (yaw). As described previously, the sensor noise for both altitude and yaw angle are implemented in the simulation. Therefore, the control performance essentially depends on how the model reacts with the noise. Without updating, the AANN seems to perform the worst, and the RMSE for the step and sinusoidal reference signals are, respectively, 0.47 m (altitude) and 1.56° (yaw), and 0.39 m (altitude) and 0.16° (yaw). It is followed by the multi-net, and the OANN-NNC surpasses both. As the input and output delays used in the models increase, the reference tracking performance also improves for all the models. This is because with more delays, the model is less susceptible to noise.

When all the models are updated, as shown in Fig. 6b, the performances of both the AANN and multi-net deteriorate, which can be attributed to the fact that both models vary rapidly in response to random noise, leading to unnecessary oscillations. However, updating the NNC does not undermine the model performance for response prediction and control. One of the main reasons is that the offline trained, GA-optimized OANN sets the trend of the model prediction, while the NNC predicts the disturbance and shift in dynamics rather than the entire system response, which makes the response prediction more robust to noise. In addition, the use of the very simple, single-layer NNC model structure also dramatically mitigates the model variance and enhances the generality even with

limited online noisy data. Because of these factors, different model architectures exhibit very distinct control performance, as revealed in the step response of the altitude (top row in Fig. 6). It is clearly observed that the AANN has the largest overshooting, followed by the multi-net and the OANN-NNC, although their prediction and control horizons are the same.

In short, it is not desirable to undertake the unnecessary update for the AANN and multi-net models when the anomaly is not present. Therefore, their model updating is only recommended when the disturbance occurs and is detected (e.g., using various fault identification methods), which nonetheless does not seem to be required for our OANN-NNC, as its control performance is not affected by executing unnecessary system updates. As a result, the OANN-NNC is easier to manage and coordinate with the control reconfiguration during operation. Furthermore, it is also superior to the other ANN models in tracking performance even with the same MPC settings.

Next, altitude tracking is performed for an operation with a large range and an abrupt change in reference signal, as depicted in Fig. 7, to inspect the generality of our modeling methodology. Specifically, in Fig. 7, the reference signal of the altitude varies between 0 and 70 m, while that in Fig. 6 is between 0 and 3 m. In addition, two step changes in reference signals, from 0 to 20 and 20 to 70 m are applied, respectively, at $t = 0$ s and $t = 50$ s. The results without model updating are shown in Fig. 7a, and the plots at the middle and the bottom row are the enlarged view of the reference tracking at time window 1 and 2, respectively. It clearly shows that the AANN performs the worst with the largest overshoot subjected to the abrupt change in the reference signal. The AANN exhibits appreciable steady-state errors for both reference signals (i.e., altitude at 20 and 70 m), even when the anomaly does not exist. The offsets, however, are not evident in Fig. 6a, which implies that the AANN model trained offline cannot precisely represent the system in operation at different altitudes and performance can be compromised. The size of the network needs to be increased to reduce such a bias. On the other hand, when the models are updated throughout the simulation, clearly the performance of the AANN is improved, as revealed by the results in Fig. 7b. While the system remains at the designated altitudes, the model learns rapidly and removes the bias, allowing accurate model prediction around that altitude. However, the AANN manifests apparently larger overshoots when the reference signal jumps from 20 to 70 m compared to the non-updating case. This may be attributed to the fact that the AANN model tends to be overfitted at the first step in the reference, and its generality becomes worse at the second jump in the reference signal, causing excessive fluctuations. The multi-net model generally performs well for both cases along with minor fluctuations in reference tracking. Again, the OANN-NNC

exceeds both in prediction and control performance, and it tracks the reference signal very closely with negligible steady-state errors or fluctuations for both cases. The results and comparison clearly prove the robustness and salience of the OANN-NNC in online training and its applicability for MPC in a large operational range.

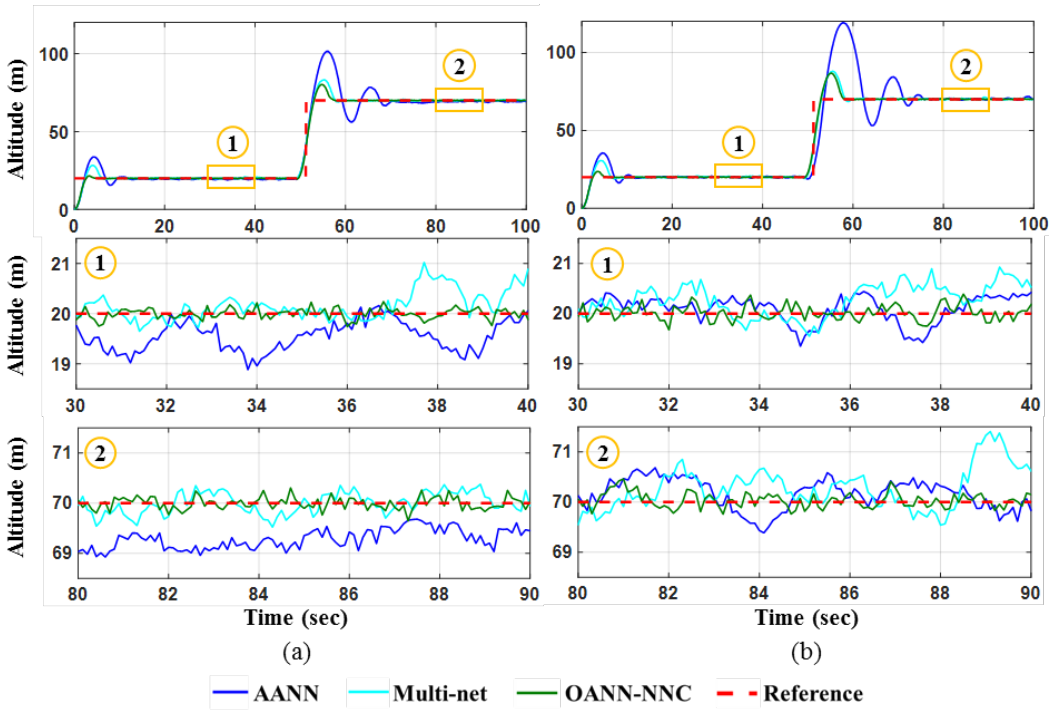

Fig. 7 MPC performances of different system models for a large range altitude tracking in the anomaly-free case: a) without and b) with model updating

4.3.2 Disturbance Rejection

In this section, MPC-based anomaly mitigation and disturbance rejection of the three previous models are compared. The simulated anomalies, same as those in Section 4.2, are again used here. The control parameters and the reference signals are also the same as those used in the anomaly-free case. The system responses are shown in Fig. 8, and the corresponding RMSE of the MPC with various models is listed in Table 7. Note that because of the presence of disturbance, all the models are updated, and there is only one column in Fig. 8 corresponding to the results of the updated model. The multi-net model, which has the largest number of weight parameters (53 in Table 4) for online model updating performs the worst for both altitude and yaw tracking. The offset error is not clearly eliminated along with largest fluctuations, which indicates that updating a large number of parameters during operation is not an easy task, especially when the online data are exposed to noise and limited in the range. Additionally, the switch between the two ANN models in the multi-net causes abrupt spikes in model prediction, which also deteriorates control performance. The AANN model is able to update itself and

adapt to the new system dynamics during the gradual anomaly phase. However, the system tends to fluctuate severely when the abrupt anomaly is applied at the 70th second. Therefore, the AANN can readily reconfigure the controller to compensate for the gradually increasing anomaly, while taking longer to respond to the disturbance of the larger magnitude. On the other hand, the OANN-NNC model exhibits salient performance for both types of anomalies. Throughout the numerical experiments, the OANN-NNC responses track the reference signals very well without noticeable steady-state error. There is actually a spike at the 70th second caused by the abrupt anomaly, which, however, is smeared out by MPC in a few seconds by reconfiguring the weight parameters in the NNC and the control inputs. Quantitatively the overall RMSE of the OANN-NNC is less than that of the AANN by approximately 0.5 m in the altitude and 1° in the yaw.

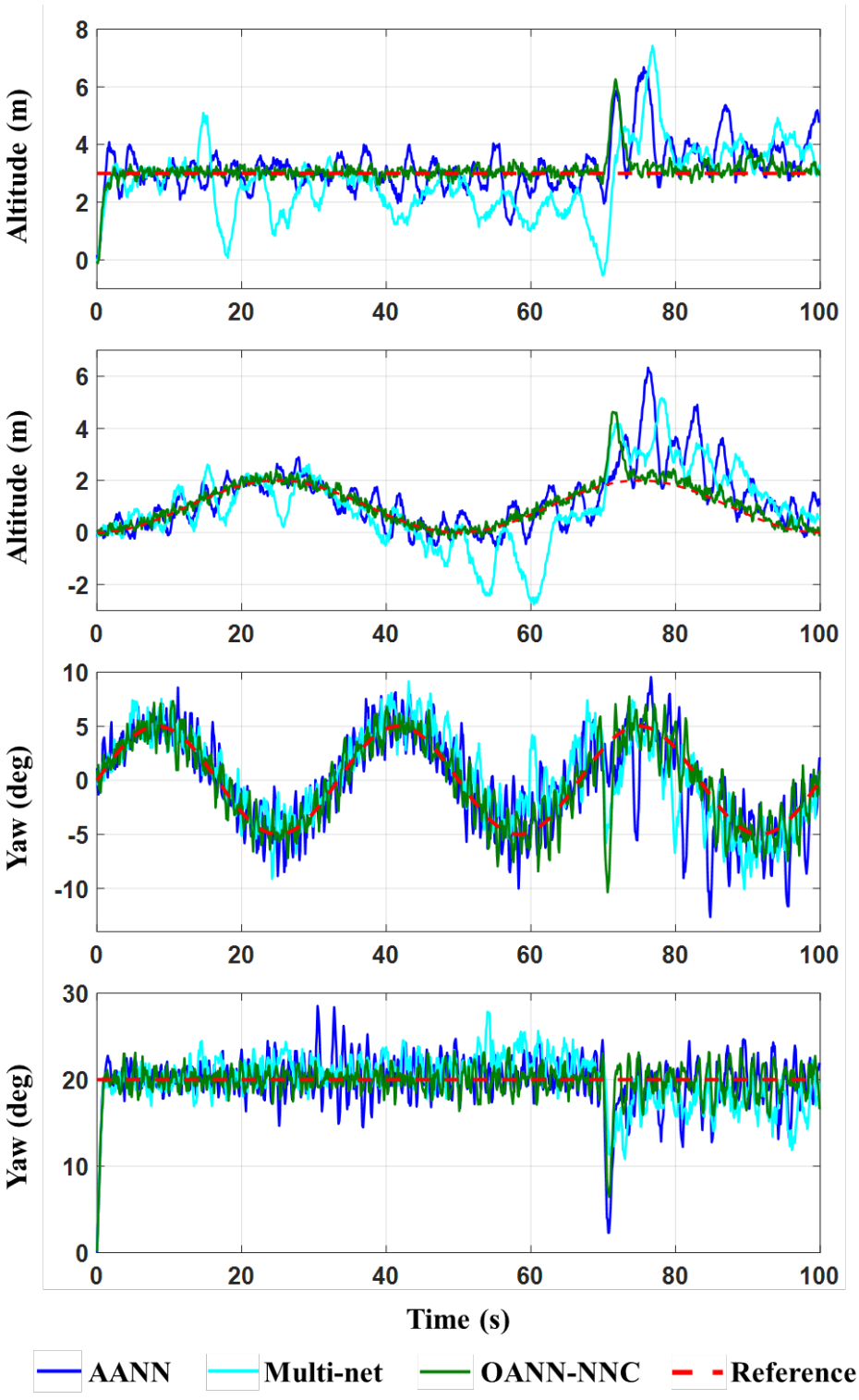

Fig. 8 MPC performances of different system models in the presence of the anomaly

Table 7 MPC performances of different system models in RMSE in the presence of the anomaly

		AANN	Multi-net	OANN-NNC
Altitude (m)	Step	0.89	1.23	0.46
	Sinusoidal	0.98	1.12	0.36
Yaw (°)	Step	2.93	2.88	2.10
	Sinusoidal	2.52	2.35	1.80

Similar to the previous procedure, the models are also investigated for a large range and abrupt changes in the reference signal with the same anomaly K_f as applied in Fig. 5. Two step changes in the altitude reference, respectively, from the 0 to 20 and 20 to 70 m are superimposed onto the anomaly, and the MPC results obtained by the three system models are shown in Fig. 9. Generally, all three models are capable of removing the offset subjected to the changes in references and anomaly by online updating. Nonetheless, the AANN and multi-net models fluctuate more severely than the OANN-CNN model around the reference signal. Moreover, similar to the anomaly-free case in Fig. 7, the AANN overshoots drastically, indicating that the model needs more time to adapt to the new range of operating parameters and new system dynamics. The multi-net model can partially mitigate the large overshooting issue by switching between the two ANN models. However, a few sudden bumps in the response output of the multi-net model are still clearly observed between 70th and 90th seconds, which actually are also found in Fig. 7. It may be caused by either the excessive switch between the two component models in the multi-net model (as the phenomenon is not clearly observed for other two models) or the more severe drift in system behavior at the second altitude reference specified at 70 m. The proposed OANN-CNN model reveals excellent performance in anomaly and noise rejection and reference tracking in a wide range with abrupt changes.

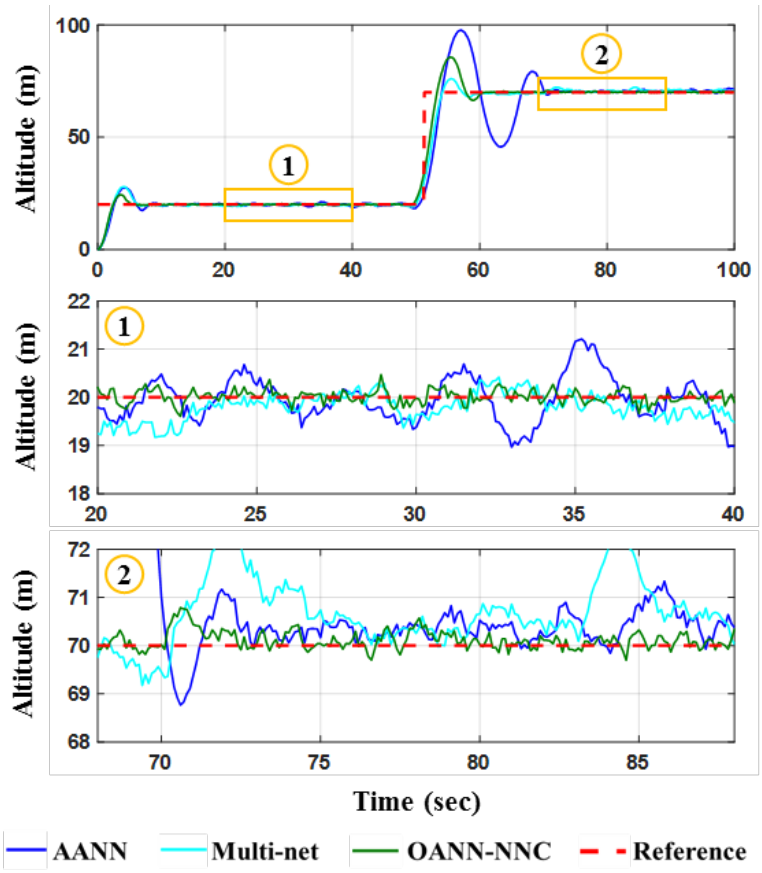

Fig. 9 MPC performances of different system models for a large range altitude tracking in the presence of the anomaly

5. Conclusion

A methodology is proposed to develop a new ANN-based system model that concatenates a GA-optimized ANN (OANN) and a NNC in series to capture temporally varying system dynamics caused by slow-paced degradation/anomaly, such as the wearing, fatigue, and others. The OANN model cast in the NARMAX formulation features a complex, fully connected MLP structure described by a large number (~1,000) of trainable weight parameters, while the NNCs are compact models in the form of neural networks, yielding only two or four weight parameters. The OANN is trained offline using a large amount of anomaly-free data and remains constant during the current operation. On the other hand, the NNC is continuously updated online to capture the disturbances caused by the system degradation/anomaly that could potentially occur during operation, and hence, bridges the gap between the actual system response y and the ANN model-predicted response y_n. Such a model architecture can be essentially considered a large network that only allows the last one or a few layers to be updated while freezing the other weight parameters in the preceding feature extraction layers.

Because of its offline nature, the computationally demanding, GA-based meta-optimization is adopted to search the optimal network structure and hyperparameters of the complex OANN model, including the time window size for input and output delays, the hidden layer size, and the training algorithm. The key advantage of adopting the OANN of complex structure is to incorporate all dominant nonlinear features into the main baseline, trend model and make the entire online updating scheme more efficient and robust. The single- or double-layer NNC is attached to the OANN model and updated during each epoch using the collected sensor data to capture the instantaneous shift in system dynamics and predict the associated disturbance in the future horizon, $d_p = y_p - y_n$. The rationale for estimating d_p rather than the actual output y_p by the NNC is that, in general, the variation in the deviation between y and y_n is milder than that in response y and can be captured by a more compact model structure, especially under the circumstances of limited sensor data and high risk of model overfitting. The NNC-adjusted system response y_p will be used to reconfigure MPC for enhanced performance.

In the case studies, the OANN with a large number of weight parameters exhibits an excellent ability to reject the noises and boost the control performance. The NNC is able to capture the anomaly/degradation-induced disturbance in the system dynamics, rectify the OANN-predicted system response, and remove the offsets. The proposed NNCs are validated and compared with the traditional FBC. Both NNCs are able to perform as well as the FBC, while supplying new information about the shifted system dynamics at the end of the operation, which cannot be provided by the FBC. The proposed OANN-NNC model architecture is compared with the AANN and multi-net models, both of which experience more difficulty in online training, as indicated by the large fluctuations and poor control performance for the quadrotor system under consideration. Quantitatively speaking, the OANN-NNC introduces smaller tracking errors for altitude (~0.5 m) and yaw angle (~1°). Also updating the AANN and multi-net models when no disturbance is present will cause the system to oscillate drastically due to the sensor noises, leading to deteriorated control performance. Nevertheless, updating the NNC is less susceptible to noise owing to the proposed model architecture (i.e., the OANN), and therefore, does not require additional consideration for model updating during operation. The models are also compared in an operation where the altitude reference signal varies abruptly in a large range. Under these conditions, we made the same observation: the OANN-NNC exhibits the best accuracy and generality of online model training and performance in reference tracking.

Future work includes implementing the proposed framework in robotics platforms (both ground and aerial) with monitoring systems for onsite, data-driven self-control reconfiguration in the presence of system degradation.

6. References

Akpan VA, Hassapis GD. Nonlinear model identification and adaptive model predictive control using neural networks. ISA transactions. 2011;50(2):177–194.

Alexandridis A, Sarimveis H. Nonlinear adaptive model predictive control based on self-correcting neural network models. AIChE Journal. 2005;51(9):2495–2506.

Billings SA. Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. Hoboken (NJ): John Wiley & Sons; 2013.

Bouabdallah S. Design and control of quadrotors with application to autonomous flying. Epfl. 2007.

Chen S, Billings SA. Representations of non-linear systems: the NARMAX model. International Journal of Control. 1989;49(3):1013–1032.

Cheng H, Yang Y. 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; 2017 June.

Cheng L, Liu W, Hou ZG, Yu J, Tan M. Neural-network-based nonlinear model predictive control for piezoelectric actuators. IEEE Transactions on Industrial Electronics. 2015;62(12):7717–7727.

Curteanu S, Cartwright H. Neural networks applied in chemistry. I. Determination of the optimal topology of multilayer perceptron neural networks. Journal of Chemometrics. 2011;25(10):527–549.

ElKholy HM. Dynamic modeling and control of a quadrotor using linear and nonlinear approaches [thesis]. [Cairo (Egypt)]: American University in Cairo; 2014.

Han HG, Guo YN, Qiao JF. Nonlinear system modeling using a self-organizing recurrent radial basis function neural network. Applied Soft Computing. 2018;71:1105–1116.

Han HG, Zhang L, Hou Y, Qiao JF. Nonlinear model predictive control based on a self-organizing recurrent neural network. IEEE transactions on neural networks and learning systems. 2016;27(2):402–415.

Hedjar R. Adaptive neural network model predictive control. International Journal of Innovative Computing, Information and Control. 2013;9(3):1245–1257.

Jiajin L, Rui L, Yingjing S, Jianxiao, Z. Design of attitude controller using explicit model predictive control for an unmanned quadrotor helicopter. In 2017 Chinese Automation Congress (CAC) (pp. 2853–2857). IEEE; 2017 October.

Kusiak A, Xu G. Modeling and optimization of HVAC systems using a dynamic neural network. Energy. 2012;42(1):241–250.

Kuyumcu A, Bayezit I. Augmented model predictive control of unmanned quadrotor vehicle. In 2017 11th Asian Control Conference (ASCC) (pp. 1626–1631). IEEE; 2017 Dec.

Lam HK, Ling SH, Leung FH, Tam PKS. Tuning of the structure and parameters of neural network using an improved genetic algorithm. In Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE. 2001;1:25–30).

Lin CM, Le TL, Huynh TT. Self-evolving function-link interval type-2 fuzzy neural network for nonlinear system identification and control. Neurocomputing. 2018;275:2239–2250.

Ma D, Xia Y, Li T, Chang K. Active disturbance rejection and predictive control strategy for a quadrotor helicopter. IET Control Theory & Applications. 2016;10(17):2213–2222.

Morari M, Maeder U. Nonlinear offset-free model predictive control. Automatica. 2012;48(9):2059–2067.

Negri GH, Cavalca MS, de Oliveira J, Araújo CJ, Celiberto LA. Evaluation of nonlinear model-based predictive control approaches using derivative-free optimization and FCC neural networks. Journal of Control, Automation and Electrical Systems. 2017;28(5):623–634.

Pan Y, Wang J. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. IEEE Transactions on Industrial Electronics. 2012;59(8):3089–3101.

Patan K. Neural network-based model predictive control: Fault tolerance and stability. IEEE Transactions on Control Systems Technology. 2015;23(3):1147–1155.

Patan K, Korbicz J. Nonlinear model predictive control of a boiler unit: a fault tolerant control study. International Journal of Applied Mathematics and Computer Science. 2012;22(1):225–237.

Puttige VR, Anavatti SG. Real-time system identification of unmanned aerial vehicles: a multi-network approach. JCP. 2008;3(7);31–38.

Raissi M, Perdikaris P, Karniadakis GE. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. arXiv preprint arXiv:1801.01236, 2018.

Sena HJ, Ramos VS, Silva FV, Fileti AMF. Adaptive Offset Remover Based on Kalman Filter Integrated to a Model Predictive Controller. Chemical Engineering. 2017;57.

Shin JH, Jun HB, Kim JG. Dynamic control of intelligent parking guidance using neural network predictive control. Computers & Industrial Engineering. 2018;120:15–30.

Tatjewski P. Disturbance modeling and state estimation for offset-free predictive control with state-space process models. International Journal of Applied Mathematics and Computer Science. 2014;24(2):313–323.

Vatankhah B, Farrokhi M. Nonlinear adaptive model predictive control of constrained systems with offset-free tracking behavior. Asian Journal of Control. 2018;21(5).

Wang T, Gao H, Qiu J. A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. IEEE Transactions on Neural Networks and Learning Systems. 2016;27(2):416–425.

Whitley D. A genetic algorithm tutorial. Statistics and Computing. 1994;4(2):65–85.

Yan Z, Wang J. Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks. IEEE Trans Neural Netw Learning Syst. 2014;25(3):457–469.

Zhang R, Tao J. Data-driven modeling using improved multi-objective optimization based neural network for coke furnace system. IEEE Transactions on Industrial Electronics. 2017;64(4):3147–3155.

Zhang R, Tao J, Gao F. A new approach of Takagi–Sugeno fuzzy modeling using an improved genetic algorithm optimization for oxygen content in a coke furnace. Industrial & Engineering Chemistry Research. 2016;55(22):6465–6474.

Zhang B, Sun X, Liu S, Deng X. Recurrent neural network-based model predictive control for multiple unmanned quadrotor formation flight. International Journal of Aerospace Engineering. 2019.

List of Symbols, Abbreviations, and Acronyms

AANN	adaptive ANN
ANN	artificial neural network
ARL	US Army Research Laboratory
FBC	feedback compensator
GA	genetic algorithm
LM	Levenberg–Marquardt
MLP	multilayer perceptron
MPC	model predictive control
MSE	mean squared error
NARMAX	nonlinear autoregressive moving-average exogenous
NNC	neural network compensator
OANN	optimized artificial neural network
RMSE	root mean squared error
RNN	recurrent neural network
TDL	tapped delay line

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

1 (PDF) CCDC ARL FCDD RLD CL TECH LIB

2 (PDF) CCDC ARL FCDD RLC EM E MARK
FCDD RLV M A HALL