Neural Processing Letters (2020) 51:1789–1833
https://doi.org/10.1007/s11063-019-10167-w

A Novel Model Predictive Runge–Kutta Neural Network Controller for Nonlinear MIMO Systems

Kemal Uçak[1]

Published online: 7 January 2020

Abstract
In this paper, a novel model predictive Runge–Kutta neural network (RK-NN) controller based on Runge–Kutta model is proposed for nonlinear MIMO systems. The proposed adaptive controller structure incorporates system model which provides to approximate the K-step ahead future behaviour of the controlled system, nonlinear controller where Runge–Kutta neural network (RK-NN) controller is directly deployed and adjustment mechanism based on Levenberg–Marquardt optimization method so as to optimize the weights of the Runge–Kutta neural network (RK-NN) controller. RBF neural network is employed as constituent network in order to identify the changing rates of the controller dynamics. So, the learning ability of RBF neural network and Runge Kutta integration method are combined in the MIMO nonlinear controller block. The control performance of the proposed MIMO RK-NN controller has been examined via simulations performed on a nonlinear three tank system and Van de Vusse benchmark system for different cases, and the obtained results indicate that the RK-NN controller and Runge–Kutta model achieve good control and modeling performances for nonlinear MIMO dynamical systems.

Keywords Adaptive nonlinear MIMO controller · Nonlinear model predictive control · Runge–Kutta based system identification · Runge–Kutta EKF · Runge–Kutta neural network controller

1 Introduction

Change is an inevitable existential truth of universe. This reality, depending on time, affects all physical entities by altering their states. Dynamics is the study of this evolution of states of physical systems as a function of time [1].

Nonlinearity is the most crucial characteristic which ensnarls the identification and control of dynamics. Since system dynamics may exhibit unpredictable nonlinear behaviour and interact especially in multiple input multiple output (MIMO) systems, machine learning

✉ Kemal Uçak
ucak@mu.edu.tr

[1] Department of Electrical and Electronics Engineering, Faculty of Engineering, Muğla Sıtkı Koçman University, Kötekli, 48000 Muğla, Turkey

techniques have recently been exploited not only to identify the dynamical behaviour of nonlinear multiple input multiple output (MIMO) systems but also to efform the nonlinear dynamics as desired via nonlinear adaptive MIMO controller architectures.

Artificial Neural Networks (ANNs), imitating the learning mechanisms of biological neural networks as far as optimization theory facilitates, can be utilized to learn complex functional relations via a limited amount of input-output training data [2]. Due to ANNs nonlinear learning and high generalization ability, they can be deployed to design nonlinear adaptive MIMO controller structures in order to cope with nonlinearity and interaction between system dynamics in nonlinear MIMO systems.

Conventional controller structures such as MIMO PID controller have linear relations between controller input and control signal. The performance of the conventional controller structures can be enhanced by introducing adaptation to the controller parameters. However, the linear relationship between input and output can not be changed at every adjustment step. PID controller has almost no memory and has low generalization ability when compared to ANN controller since PID remembers only the previous two tracking error data and approximates the control signal depending on these two previous tracking error instances. However, in ANN controller, in addition to previous tracking errors, previous system behaviours and control signals can also be deployed as input features so as to enhance the approximation of the optimal control signal. Therefore, the controller structures with nonlinear behaviour such as ANN controller can be deployed to control highly nonlinear MIMO systems. Adaptive control structures based on ANN can be structurally examined under three main headings:

- Adaptive conventional controller structures based on ANN system model.
- Composite controllers where ANN structure is deployed as feedforward controller.
- Nonlinear ANN controllers where ANN structure is directly deployed as controller.

In adaptive conventional controller structures based on ANN system model, the parameters of conventional controller such as PID, state feedback controller etc. are adjusted via gradient based optimization rules. The adaptation rules necessitates system Jacobian information which is approximated via ANN model of controlled system. Tan and Keyser [3] introduced an adaptive PID controller where the NN is implemented to approximate the system Jocabian of a time delay system. Recursive Least Squares (RLS) learning algorithm is utilized to attain parameters of the network in online manner. Zhang, Li and Liu proposed to deploy radial basis function(RBF) NN to identify the system Jacobian in order to utilize in gradient descent algorithm in training of PID [4]. Iplikci proposed to adjust PID parameters using offline trained ANN approximating K-step ahead future behaviour of system to construct the Jacobian matrix via K-step system Jacobian [5]. Akyar and Omatu deployed NN structure as self-tuning regulator(STR) to identify the dynamics of PID parameters and aimed to fit a regression function to the behaviour of controller parameters [6]. The technical literature is very rich in terms of NN based PID structures [7–14] owing to the simple structure and robustness of PID.

In composite controller structures, the overall control architecture is composed of a feedback controller in which usually a conventional controller is preferred and a feedforward controller where ANN is utilized.The feedback controller is employed to stabilize the controlled system and then ANN controller gradually takes over the control task and mimics the inverse dynamics of the controlled system. Nordgren and Meckl [15] used a hybrid NN controller with PD feedback compensator to control two coupled pendulums. Yamada and Yabuta utilized a proportional controller in feedback controller block and NN in feedforward structure to control a one degree of freedom force control system [16]. NN structure was deployed as a feedforward controller where PD is implemented as feedback controller to

control a nonlinear robot arm in [17]. Ji and Familoni [18] employed a hybrid controller in which the diagonal recurrent neural network (DRNN) structure with PID feedback controller is executed so as to enhance control performance for simultaneous perturbation stochastic approximation (SPSA) control system.

In nonlinear ANN controller structures where ANN is directly utilized as controller, unlike the ANN system model, an additional ANN structure can be employed to identify the dynamics of the control signal applied to the controlled system. For this purpose, predictive system model can be utilized to approximate the K-step ahead future behaviour and also Jacobian of the controlled system. Wu, Hogg and Irwin employed an adaptive NN controller which is trained with a new separate NN model of the controlled system to control a turbogenerator system [19]. Khalid, Omatu and Yusof used an adaptive NN controller adjusted via offline trained NN model of the controlled system [20]. Owing to their powerful nonlinear approximation ability, direct ANN controllers or ANN model based ANN controllers are frequently deployed to solve tracking problems of linear and nonlinear systems [21–33].

In technical literature, in addition to ANN model based adaptive controller architectures, various effective adaptive control architectures based on soft computing methods have been offered for nonlinear control systems [34–38]. However, the mentioned methods suffer from the computational load of the system identification procedure. Since the accuracy and also computational load of the system identification blocks are vital in the real time execution of the proposed control algorithm, the utilization of adaptive control algorithms with low computational load and high identification accuracy is of great importance in adaptive control theory. Whereas the convergence and accurate adjustment of the controller parameters are directly affected by system model, the implementation of the algorithm for various kinds of systems is restricted because of the computational load of the identification step. Therefore, so as to enhance the applicability of the adaptive control algorithms, a novel system identification technique based on Runge–Kutta model has been introduced by Iplikci [39] for nonlinear MIMO systems to be executed in a nonlinear model predictive control (NMPC) structure. The proposed identification method requires the differential equations of the controlled system to be derivable [39,40]. Since this is possible for many kinds of dynamical systems at the present time, the adaptive controller structures based on RK-model can be successfully deployed for wide ranges of nonlinear MIMO systems [39,40].

In technical literature, various controller structures based on RK-system identification technique have been proposed. The precessor form of the RK based identification technique has been proposed by Iplikci in [39] to be implemented in the nonlinear model predictive control (NMPC) framework. In NMPC structures, a finite-horizon open-loop optimal control problem is solved during each sampling period. Therefore, NMPC necessitates the approximation of the future behaviour and also system Jacobian of the controlled system in response to the candidate control signals. The adjustment rules to acquire the control signal vector in NMPC are derived via the Taylor expansion of the objective function. Consequently, the control problem is degraded to attain the sensitivity of the controlled system outputs with respect to control signals (system Jacobian). Therefore, RK system model is deployed to identify the dynamics of the controlled system. The Runge–Kutta identification block incorporates raw RK system model, RK based model parameter estimator block and RK based EKF block. RK-model of the system is used for control, state estimation and model parameter adjustment [39,41]. RK based EKF block is utilized to approximate the current states of the controlled system via input-output data pair of the controlled system so as to estimate the future behaviours of the states since only system input-output data pairs are available. RK model parameter estimator block provides to obtain deviated system model parameters or the system parameters which can not be acquired accurately. Çetin et al. proposed an adaptive

MIMO PID controller based on RK model in which the controller parameters are optimized via Levenberg–Marquardt learning algorithm and RK model is employed to identify K- step ahead system Jacobian information [42]. The proposed auto-tuning PID mechanism incorporates the robustness of PID structure, fast convergence from the MPC framework and gradient based adaptation ability [42]. Beyhan [41] introduced a Runge–Kutta model based nonlinear observer in which the proposed RK based identification method in [39] is utilized to forecast the K-step ahead sensitivity of the system outputs with respect to system states so as to attain the system states optimized via Levenberg–Marquardt algorithm.

In this paper, a novel predictive Runge–Kutta neural network controller has been introduced for nonlinear multiple input multiple output (MIMO) systems. The adaptation mechanism comprises a Runge–Kutta RBF neural network controller to identify the dynamics of the optimal control signal so as to compel the system outputs to the desired references and a Runge–Kutta system model to approximate the K-step ahead system Jacobian informations required in adaptation law. Neural networks such as multi layer perceptrons (MLP) which are constituted to acquire the relationship between input-output system states cannot apprehend the long-term behaviour of the identified systems well and long-term approximation precision is usually not adequate enough since the network learns the system states, instead of changing rates of system states [43], which motivates us to deploy Runge–Kutta neural network to approximate the changing rates of the nonlinear control law. Therefore, Runge–Kutta neural network structure, which comprehends the strong aspects of the Runge–Kutta integration method and artificial neural networks, is opted for the nonlinear controller block. RBF type neural network is utilized as constituent subnetwork in RK-NN structure because of its superior approximation competency [44,45], its fast learning ability under favour of locally tuned neurons [46–48] in comparison with other neural networks such as MLP etc. and its more compact form than other neural networks [49]. In order to estimate the dynamic behavior of the controlled nonlinear system, RK based identification method proposed by Iplikci [39] is employed due to its low computational load and high identification precision. The main contributions and differences of this paper from the existing studies in the literature can be pointed out as

- Runge–Kutta integration method is deployed in both controller block to obtain optimal control signal and system model block to identify the dynamics of the controlled system.
- The fast learning and convergence speed of RBF neural network and accurate integration ability of Runge–Kutta method are fused in Runge–Kutta RBF neural network controller structure to acquire the optimal control signal for nonlinear MIMO systems.

The performance evaluation of the proposed adaptive nonlinear controller has been carried out on nonlinear three tank system and Van de Vusse benchmark system for various circumstances. The obtained results demonstrate the closed-loop control and system identification accomplishments of the the disclosed Runge–Kutta neural network controller and Runge–Kutta system model. The paper is arranged as follows: Sect. 2 overviews the proposed Runge–Kutta Neural Network controller. The basic principles of Runge–Kutta model utilized in system identification block proposed by Iplikci [39] is described in Sect. 3. Construction of the optimization problem and derivation of adjustment rules to deploy Runge–Kutta neural network directly as an adaptive controller and the proposed adjustment mechanism are explained in detail in Sect. 4. In Sect. 5, the control performance of the proposed method has been evaluated on a nonlinear three tank system and Van de Vusse benchmark system. The paper is concluded with a brief conclusion part in Sect. 6.

2 The Proposed Runge–Kutta Neural Network Controller

Adaptation and nonlinearity in control parameters provide flexibility to nonlinear control structures in order to attune the alterations occurring in nonlinear system dynamics. Therefore, approximation of nonlinear system dynamics and design of optimal controller parameters so as to compel the system dynamics to the desired operating conditions are vital steps in adaptive controller structures. For this purpose, in this section, firstly, a brief information about the mechanism and requirements of adaptive control structures are given in Sect. 2.1. Then, in order to facilitate understanding of the proposed adaptive control mechanism, its detailed outline is provided in Sect. 2.2. The content of the mechanism is detailed in the following sections.

2.1 An Overview of Adaptive Control

An adaptive control mechanism contains system model, nonlinear controller and adaptation law blocks as illustrated in Fig. 1 where $\mathbf{u}$ is the control signal applied to the system, $\mathbf{f_c}$ denotes a nonlinear control law, $\boldsymbol{\Theta}$ stands for adjustable controller parameters, $\mathbf{C}_{in}$ represents the input of the controller, $\mathbf{y}$ denotes system output and $\mathbf{y_m}$ is system model output. Accurate adjustment of controller parameters depends on the precise approximation of the system output in response to alternations on controller parameters. Therefore, system model is a substantial part of the mechanism to observe the possible future behaviour of the system. In adaptation law block, the current adjustment rules for controller parameters are derived by considering the history of the system dynamics and the future behaviour of the system via the obtained system model. Then, by using the optimized controller parameters in controller block, the optimal control signal which is anticipated to force the system dynamics to the reference signal can be accurately achieved. As can be seen from Fig. 1, depending on utilized system model, controller structure and adaptation law, numerous adaptive controller structures can be introduced for nonlinear systems [50]. It is possible to employ any controller with adjustable parameters in the controller block given in Fig. 1 [2]. In this work, Runge–Kutta neural network controller is used as a nonlinear controller. As for the system model part, various machine learning based modeling techniques such as ANN [51–54], ANFIS [55,56],

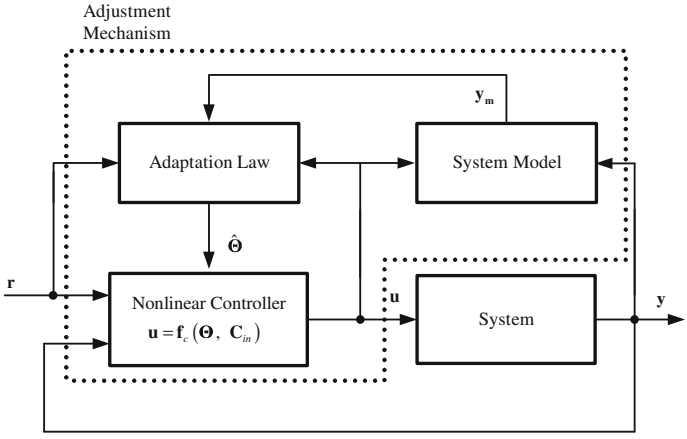

Fig. 1 Basic adaptive control mechanism

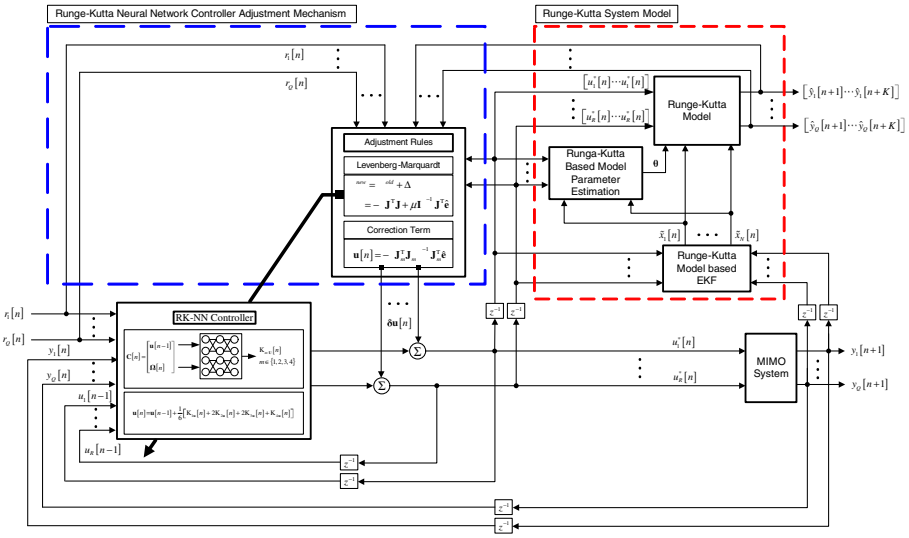

Fig. 2 Model predictive Runge–Kutta neural network controller structure based on Runge–Kutta model

SVR [5,57,58] etc have previously been deployed to identify the system dynamics. In the proposed controller structure, the dynamics of the controlled system are approximated using Runge–Kutta system model proposed by Iplikci [39] in order to enhance model accuracy and diminish computational load of the control mechanism.

2.2 Runge–Kutta Neural Network Controller Structure Based on Runge–Kutta Model

The adaptation mechanism of the Runge–Kutta neural network controller architecture based on Runge–Kutta model is depictured in Fig. 2 where R is the dimension of the system input signal and Q represents the dimension of the controlled system output. The proposed mechanism has two main structures to be meticulously examined: Runge–Kutta neural network controller to identify the dynamics of the optimal control signal and Runge–Kutta system model to predict the future behaviour of the controlled system. In order to enhance intelligibility and simplicity, Runge–Kutta neural network controller is abbreviated as RK-NN$_{\text{controller}}$ and Runge–Kutta system model is RK$_{\text{model}}$ throughout the entire article. The learning, prediction and control phases of RK-NN$_{\text{controller}}$ and RK$_{\text{model}}$ in adjustment mechanism are consecutively carried out in an online manner. In the adjustment mechanism, firstly, the control signals ($\mathbf{u}[n]$) are computed via the current weights ($\boldsymbol{\Theta}^{old} = \left[\alpha_1^{old} \cdots \alpha_M^{old}\right]^T$) of the RK-NN$_{\text{controller}}$ as in (1):

$$\mathbf{u}\left[\mathbf{n}\right]^{old} = \mathbf{f_{NN}}(\boldsymbol{\Theta}^{old}, \mathbf{C}_{in}) \tag{1}$$

where $\hat{\boldsymbol{\Theta}}$ stands for the weights of RK-NN$_{\text{controller}}$ structure and $\mathbf{C}_{in}$ is the input feature vector of RK-NN$_{\text{controller}}$. Then, the attained control signals ($\mathbf{u}\left[\mathbf{n}\right]^{old}$) are recurrently applied to the RK$_{\text{model}}$ K-times so as to observe the K-step ahead future behaviour of the controlled system in response to $\boldsymbol{\Theta}^{old}$ controller parameters. RK$_{\text{model}}$ is composed of Runge–Kutta model based EKF(RK$_{\text{EKF}}$), Runge–Kutta based model parameter estimation (RK$_{\text{estimator}}$) and raw Runge–Kutta system model subblocks, as can be seen from RK$_{\text{model}}$. In order to approximate K-step ahead system behaviour, firstly, the current states of the controlled system

($[\tilde{x}_1[n]\cdots\tilde{x}_N[n]]$) are attained via $\mathrm{RK}_{\mathrm{EKF}}$. Then, if there is a substantial deviation in system parameters (θ), the corresponding $\mathrm{RK}_{\mathrm{model}}$ parameters ($\hat{\theta}$) are optimized depending on the obtained current states of the system ($[\tilde{x}_1[n]\cdots\tilde{x}_N[n]]$) and control signals via $\mathrm{RK}_{\mathrm{estimator}}$ subblock. Then, by considering the possibility that the system parameters may alter and substantial deviations between $\mathrm{RK}_{\mathrm{model}}$ parameters ($\hat{\theta}$) and their nominal values θ may ensue, the corresponding $\mathrm{RK}_{\mathrm{model}}$ parameters($\hat{\theta}$) are optimized depending on the obtained current states of the system ($[\tilde{x}_1[n]\cdots\tilde{x}_N[n]]$) and control signals via $\mathrm{RK}_{\mathrm{estimator}}$ subblock. Finally, using the current optimized values of model parameters (θ), system states ($[\tilde{x}_1[n]\cdots\tilde{x}_N[n]]$) and control signals ($[u_1^{\star}[n]\cdots u_R^{\star}[n]]$) in $\mathrm{RK}_{\mathrm{model}}$, K-step ahead future behaviour of the controlled system can be acquired. The feasible weights of RK-$\mathrm{NN}_{\mathrm{controller}}$ which compel the system output to track the reference signal can be attained using the obtained K-step ahead system behaviour and adaptation law. For this purpose, the following objective function must be minimized:

$$\begin{aligned} F(\mathbf{u}[\mathbf{n}],\mathbf{e}_q) &= \sum_{q=1}^{Q}\sum_{k=1}^{K}\Big[r_q[n+k]-\hat{y}_q[n+k]\Big]^2+\sum_{r=1}^{R}\lambda_r\Big[u_r[n]-u_r[n-1]\Big]^2 \\ &= \sum_{q=1}^{Q}\sum_{k=1}^{K}\Big[\hat{e}_q[n+k]\Big]^2+\sum_{r=1}^{R}\lambda_r\Big[u_r[n]-u_r[n-1]\Big]^2 \end{aligned} \tag{2}$$

where K stands for the prediction horizon, Q denotes the number of the controlled outputs, R is the number of the control signals and λ's represent penalty terms utilized to restrict the deviation of the control signals [40]. The network weights of the RK-$\mathrm{NN}_{\mathrm{controller}}$ can be optimized via Levenberg–Marquardt optimization rule as follows:

$$\boldsymbol{\Theta}^{new}=\boldsymbol{\Theta}^{old}+\Delta\boldsymbol{\Theta},\quad \Delta\boldsymbol{\Theta}=-\left[\mathbf{J}^T\mathbf{J}+\mu\mathbf{I}\right]^{-1}\mathbf{J}^T\hat{\mathbf{e}} \tag{3}$$

where $\mathbf{J}$ emblematises a $(QK+R)xZ$ dimension system Jacobian matrix given as

$$\mathbf{J}=\begin{bmatrix} \frac{\partial\hat{e}_1[n+1]}{\partial\alpha_1^{old}} & \cdots & \frac{\partial\hat{e}_1[n+1]}{\partial\alpha_Z^{old}} \\ \vdots & \ddots & \vdots \\ \frac{\partial\hat{e}_Q[n+K]}{\partial\alpha_1^{old}} & \cdots & \frac{\partial\hat{e}_Q[n+K]}{\partial\alpha_Z^{old}} \\ \sqrt{\lambda_1}\frac{\partial\Delta u_1[n]}{\partial\alpha_1^{old}} & \cdots & \sqrt{\lambda_1}\frac{\partial\Delta u_1[n]}{\partial\alpha_Z^{old}} \\ \vdots & \ddots & \vdots \\ \sqrt{\lambda_R}\frac{\partial\Delta u_R[n]}{\partial\alpha_1^{old}} & \cdots & \sqrt{\lambda_R}\frac{\partial\Delta u_R[n]}{\partial\alpha_Z^{old}} \end{bmatrix} = -\begin{bmatrix} \frac{\partial\hat{y}_1[n+1]}{\partial\alpha_1^{old}} & \cdots & \frac{\partial\hat{y}_1[n+1]}{\partial\alpha_Z^{old}} \\ \vdots & \ddots & \vdots \\ \frac{\partial\hat{y}_Q[n+K]}{\partial\alpha_1^{old}} & \cdots & \frac{\partial\hat{y}_Q[n+K]}{\partial\alpha_Z^{old}} \\ -\sqrt{\lambda_1}\frac{\partial\Delta u_1[n]}{\partial\alpha_1^{old}} & \cdots & -\sqrt{\lambda_1}\frac{\partial\Delta u_1[n]}{\partial\alpha_Z^{old}} \\ \vdots & \ddots & \vdots \\ -\sqrt{\lambda_R}\frac{\partial\Delta u_R[n]}{\partial\alpha_1^{old}} & \cdots & -\sqrt{\lambda_R}\frac{\partial\Delta u_R[n]}{\partial\alpha_Z^{old}} \end{bmatrix}_{[(QK+R)xZ]} \tag{4}$$

and $\hat{\mathbf{e}}$ is the vector of the prediction errors

$$\hat{\mathbf{e}} = \begin{bmatrix} \hat{e}_1[n+1] \\ \vdots \\ \hat{e}_1[n+K] \\ \vdots \\ \vdots \\ \hat{e}_Q[n+1] \\ \vdots \\ \hat{e}_Q[n+K] \\ \sqrt{\lambda_1}\Delta u_1[n] \\ \vdots \\ \sqrt{\lambda_R}\Delta u_R[n] \end{bmatrix} = \begin{bmatrix} \hat{e}[n+1] \\ \vdots \\ \hat{e}[n+K] \\ \vdots \\ \vdots \\ \hat{e}[n+(Q-1)K+1] \\ \vdots \\ \hat{e}[n+QK] \\ \sqrt{\lambda_1}\Delta u_1[n] \\ \vdots \\ \sqrt{\lambda_R}\Delta u_R[n] \end{bmatrix}$$

$$= \begin{bmatrix} r_1[n+1] - \hat{y}_1[n+1] \\ \vdots \\ r_1[n+K] - \hat{y}_1[n+K] \\ \vdots \\ \vdots \\ r_Q[n+1] - \hat{y}_Q[n+1] \\ \vdots \\ r_Q[n+K] - \hat{y}_Q[n+K] \\ \sqrt{\lambda_1}\Big[\Delta u_1[n] - \Delta u_1[n-1]\Big] \\ \vdots \\ \sqrt{\lambda_R}\Big[\Delta u_R[n] - \Delta u_R[n-1]\Big] \end{bmatrix}_{[(QK+R)x1]} \tag{5}$$

As can be seen from(4), in order to constitute the system Jacobian matrix, it is required to approximate the sensitivity of the system outputs with respect to RK-NN$_{\text{controller}}$ parameters ($\frac{\partial \hat{y}_Q[n+K]}{\partial \alpha_Z^{old}}$). The term ($\frac{\partial \hat{y}_Q[n+K]}{\partial \alpha_Z^{old}}$) in Jacobian Matrix(4) can be expanded via chain rule as follows:

$$\frac{\partial \hat{y}_Q[n+K]}{\partial \alpha_Z^{old}} = \left[\sum_{r=1}^{R} \frac{\partial y_Q[n+K]}{\partial u_r[n+1]} \frac{\partial u_r[n+1]}{\partial \alpha_Z^{old}}\right] \tag{6}$$

where $\frac{\partial y_Q[n+K]}{\partial u_r[n+1]}$ is the sensitivity of the Qth system outputs with respect to rth control inputs and $\frac{\partial u_r[n+1]}{\partial \alpha_Z^{old}}$ indicates the sensitivity of the rth control signal with respect to Zth RK-NN$_{\text{controller}}$ parameter. The term $\frac{\partial u_r[n+1]}{\partial \alpha_Z^{old}}$ can be easily acquired via the relationship between rth control signal and Zth RK-NN$_{\text{controller}}$ parameter. In the ideal case, it is anticipated that $\hat{y}_q[n+1],\ q \in \{1,\ldots,Q\}$ converges to $y_q[n+1],\ q \in \{1,\ldots,Q\}$ during the course of online working [2]. Thus, K-step ahead unknown system Jacobian information term ($\frac{\partial y_Q[n+K]}{\partial u_r[n+1]}$) can be successfully computed via RK$_{\text{model}}$ so as to construct the Jacobian

matrix for Levenberg–Marquardt algorithm. Thus, as a consequence of adaptation law in (3), RK-NN$_{\text{controller}}$ parameters are anticipated to iteratively converge their optimal values in long run [5]. Occasionally, mostly in the transient-state and to some extent in the steady-state, it is possible that the RK-NN$_{\text{controller}}$ parameters may not be optimally adjusted owing to modeling inaccuracies and external disturbances, this induces a control action $\mathbf{u}[n]$ that may not be adequate to force the system output toward the desired trajectory as a result of the non-optimal controller parameters [5]. To solve this problem, a correction term $\delta\mathbf{u}[n]$ to be added to the control action ($\mathbf{u}[n]$) is proposed to restore the deteriorations resulting from non-optimal control action [5]. $\delta\mathbf{u}[n]$ correction term aims to minimize the objective function F and is computed using the second-order Taylor approximation of the objective function F as follows [5]:

$$F(\mathbf{u}[\mathbf{n}]+\delta\mathbf{u}[\mathbf{n}]) \cong F(\mathbf{u}[\mathbf{n}]) + \frac{\partial F(\mathbf{u}[\mathbf{n}])}{\partial \mathbf{u}[\mathbf{n}]}\delta\mathbf{u}[\mathbf{n}] + \frac{1}{2}\frac{\partial^2 F(\mathbf{u}[\mathbf{n}])}{\partial^2 \mathbf{u}[\mathbf{n}]}(\delta\mathbf{u}[\mathbf{n}])^2 \tag{7}$$

According to the the first order optimality conditions, the derivative of the approximate F with respect to $\delta\mathbf{u}[n]$ can be attained as

$$\frac{\partial F(\mathbf{u}[\mathbf{n}]+\delta\mathbf{u}[\mathbf{n}])}{\partial \delta\mathbf{u}[\mathbf{n}]} \cong \frac{\partial F(\mathbf{u}[\mathbf{n}])}{\partial \mathbf{u}[\mathbf{n}]} + \frac{\partial^2 F(\mathbf{u}[\mathbf{n}])}{\partial^2 \mathbf{u}[\mathbf{n}]}\delta\mathbf{u}[\mathbf{n}] = 0 \tag{8}$$

Thus, using the equality in (8), $\delta\mathbf{u}[n]$ can be acquired as

$$\delta\mathbf{u}[\mathbf{n}] = -\frac{\frac{\partial F(\mathbf{u}[\mathbf{n}])}{\partial \mathbf{u}[\mathbf{n}]}}{\frac{\partial^2 F(\mathbf{u}[\mathbf{n}])}{\partial^2 \mathbf{u}[\mathbf{n}]}} \tag{9}$$

As seen in (9), $\delta\mathbf{u}[n]$ term is composed of gradient ($\frac{\partial F(\mathbf{u}[\mathbf{n}])}{\partial \mathbf{u}[\mathbf{n}]}$) and Hessian ($\frac{\partial^2 F(\mathbf{u}[\mathbf{n}])}{\partial^2 \mathbf{u}[\mathbf{n}]}$) terms. The gradient vector can be easily constituted via (2) as

$$\frac{\partial F(\mathbf{u}[\mathbf{n}])}{\partial \mathbf{u}[\mathbf{n}]} = 2\mathbf{J}_m^T\hat{\mathbf{e}} \tag{10}$$

where

$$\mathbf{J}_m = \begin{bmatrix} -\frac{\partial \hat{y}_1[n+1]}{\partial u_1[n]} & \cdots & -\frac{\partial \hat{y}_1[n+1]}{\partial u_R[n]} \\ \vdots & \ddots & \vdots \\ -\frac{\partial \hat{y}_Q[n+K]}{\partial u_1[n]} & \cdots & -\frac{\partial \hat{y}_Q[n+K]}{\partial u_R[n]} \\ \sqrt{\lambda_1} & \cdots & \sqrt{\lambda_1} \\ \vdots & \ddots & \vdots \\ \sqrt{\lambda_R} & \cdots & \sqrt{\lambda_R} \end{bmatrix}_{[(QK+R)x1]} \tag{11}$$

The Hessian ($\frac{\partial^2 F(\mathbf{u}[\mathbf{n}])}{\partial^2 \mathbf{u}[\mathbf{n}]}$) term can be estimated as in (12) in order to diminish the computational load and complexity of the Hessian term resulting from 2nd order derivatives

$$\frac{\partial^2 F(\mathbf{u}[\mathbf{n}])}{\partial^2 \mathbf{u}[\mathbf{n}]} = 2\mathbf{J}_m^T\mathbf{J}_m \tag{12}$$

Thus, by substituting (10) and (12) in (9), the correction term($\delta\mathbf{u}[\mathbf{n}]$) can be computed as

$$\delta\mathbf{u}[\mathbf{n}] = -\left[\mathbf{J}_m^T\mathbf{J}_m\right]^{-1}\mathbf{J}_m^T\hat{\mathbf{e}} \tag{13}$$

Then, using the trained RK-NN$_{\text{controller}}$ parameters ($\boldsymbol{\Theta}^{new}$), new control ($\mathbf{u}^{new}$) signal can be calculated via (1). By adding the suboptimal correction term, the optimal control signal which is applied to the real system in order to compel the system output to track reference signal can be obtained as $\mathbf{u}^{\star}[n] = \mathbf{u}[n] + \delta\mathbf{u}[n]$ [40]. By now, the proposed adjustment mechanism is outlined to provide laconic information. The elaborations related to working principle of each block in RK$_{\text{model}}$ and the adjustment rules for RK-NN$_{\text{controller}}$ are given in Sects. 3 and 4, respectively. The detailed pseudo code of the proposed adaptive control architecture is presented in Sect. 4.3 so as to implement the control algorithm adroitly.

3 Nonlinear System Identification via Runge–Kutta System Model

In this section, nonlinear system identification block based on RK proposed by Iplikci [39] is examined. The main idea behind the RK based identification method is to discretize the continuous-time MIMO system dynamics via 4th order Runge–Kutta integration method in order to acquire an adaptive and data sampled identification technique [40]. Therefore, firstly, brief information about the basics of Runge–Kutta discretization method deployed to discretize the continuous time MIMO system is presented in Sect. 3.1. One step ahead future behaviour of the system can be approximated via Runge–Kutta discretization if the current value of system states and system parameters utilized in state functions are available. That is, the current states of the system and model parameters are two vital components of the method in order to utilize Runge–Kutta discretization method effectively for nonlinear MIMO systems. Since the identification method based on Runge–Kutta is data sampled and accuracy of the current states influences the correct approximation of future system behaviour, RK-model based EKF method is deployed to estimate current states of the system using available input-output data pairs of controlled system. In Sect. 3.2, the details related to RK-model based EKF are given. Occasionally, it may be difficult to acquire the model parameters using conventional modelling methods or system parameters may digress from their actual value because of internal or external factors. Estimation of the system parameters is vital so as to be able to execute control task successfully. For this purpose, in order to adjust the RK-model parameters when model parameters start to deviate from their actual values, the Runge–Kutta Model based online model parameter estimation block is deployed as given in detail in Sect. 3.3. Thus, after all fundamental components of the RK based nonlinear system identification block proposed by Iplikci [39] are diffusively viewed, a predictive model of system can be acquired. In Sect. 3.4, RK-system model based predictive model deployed to approximate K-step ahead future system behaviour is investigated.

3.1 An Overview of MIMO Systems and Runge–Kutta System Model

Consider an N-dimensional continuous-time MIMO system as illustrated in Fig. 3a where the state equations of the system are denoted as

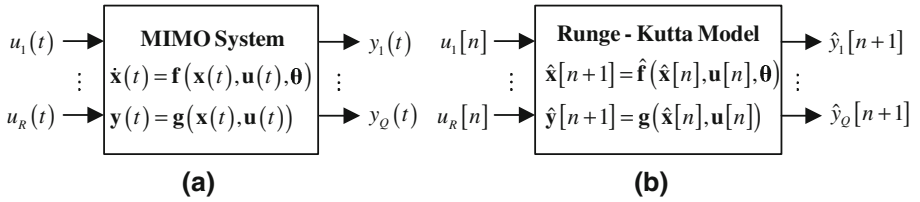

Fig. 3 A continuous-time multiple-input multiple output (MIMO) system (**a**) and its Runge–Kutta model (**b**)

$$\begin{aligned}
\dot{x}_1(t) &= f_1(x_1(t),\ldots,x_N(t),u_1(t),\ldots,u_R(t),\theta)\\
&\vdots\\
\dot{x}_N(t) &= f_N(x_1(t),\ldots,x_N(t),u_1(t),\ldots,u_R(t),\theta)
\end{aligned} \tag{14}$$

subject to state and input constraints of the form

$$\begin{aligned}
x_1(t)\in X_1,\ldots,x_N(t)\in X_N, \quad \forall t\geq 0\\
u_1(t)\in U_1,\ldots,u_R(t)\in U_R, \quad \forall t\geq 0
\end{aligned} \tag{15}$$

where X_i's and U_i's symbolise the box constraints for the states and inputs as given below, respectively

$$\begin{aligned}
X_i &\in \left\{ x_i\in\Re \mid x_{i_{min}}\leq x_i\leq x_{i_{max}} \right\} \quad for\ i=1,\ldots,N\\
U_i &\in \left\{ u_i\in\Re \mid u_{i_{min}}\leq u_i\leq u_{i_{max}} \right\} \quad for\ i=1,\ldots,R
\end{aligned} \tag{16}$$

and the output equations are

$$\begin{aligned}
y_1(t) &= g_1(x_1(t),\ldots,x_N(t),u_1(t),\ldots,u_R(t))\\
&\vdots\\
y_Q(t) &= g_Q(x_1(t),\ldots,x_N(t),u_1(t),\ldots,u_R(t))
\end{aligned} \tag{17}$$

where R is the number of inputs, N emblematises the number of states, Q stands for the number of outputs and θ indicates the parameters of the system [39]. The above system Eqs. (14–17) for nonlinear MIMO systems can be expressed in more compact form as [39]

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x},\mathbf{u},\theta)\\
\mathbf{y} &= \mathbf{g}(\mathbf{x},\mathbf{u})\\
\mathbf{x}\in\mathbf{X}\ ,\ \mathbf{u}\in\mathbf{U}
\end{aligned} \tag{18}$$

where f_i and g_i terms are assumed to be known and continuously differentiable with respect to their input variables, the state variables and θ, and also presumed that the state and input constraint sets $\mathbf{X}$ and $\mathbf{U}$ are compact [39]. By using Runge–Kutta integration algorithm, the current states and control inputs of the system can be discretized as $x_1[n]\cdots x_N[n]$ and $u_1[n]\cdots u_R[n]$ where n denotes the sampling instant as $t=nT_s$. One-step ahead system states and outputs, i.e $x_i[n+1]$ and $y_i[n+1]$, can be estimated via the fourth-order Runge–Kutta integration algorithm as follows

$$\begin{aligned}
\hat{x}_1[n+1] &= \hat{x}_1[n] + \frac{1}{6}K_{1X_1}[n] + \frac{2}{6}K_{2X_1}[n] + \frac{2}{6}K_{3X_1}[n] + \frac{1}{6}K_{4X_1}[n] \\
&\vdots \\
\hat{x}_N[n+1] &= \hat{x}_N[n] + \frac{1}{6}K_{1X_N}[n] + \frac{2}{6}K_{2X_N}[n] + \frac{2}{6}K_{3X_N}[n] + \frac{1}{6}K_{4X_N}[n]
\end{aligned} \tag{19}$$

and

$$\begin{aligned}
y_1[n+1] &= g_1(\hat{x}_1[n+1], \ldots, \hat{x}_N[n+1], u_1[n], \ldots, u_R[n]) \\
&\vdots \\
y_Q[n+1] &= g_Q(\hat{x}_1[n+1], \ldots, \hat{x}_N[n+1], u_1[n], \ldots, u_R[n])
\end{aligned} \tag{20}$$

where

$$\begin{aligned}
K_{1X_1}[n] &= T_s f_1(\hat{x}_1[n], \ldots, \hat{x}_N[n], u_1[n], \ldots, u_R[n], \theta) \\
&\vdots \\
K_{1X_N}[n] &= T_s f_N(\hat{x}_1[n], \ldots, \hat{x}_N[n], u_1[n], \ldots, u_R[n], \theta)
\end{aligned} \tag{21}$$

$$\begin{aligned}
K_{2X_1}[n] &= T_s f_1(\hat{x}_1[n] + \frac{1}{2}K_{1X_1}[n], \ldots, \hat{x}_N[n] + \frac{1}{2}K_{1X_N}[n], u_1[n], \ldots, u_R[n], \theta) \\
&\vdots \\
K_{2X_N}[n] &= T_s f_N(\hat{x}_1[n] + \frac{1}{2}K_{1X_1}[n], \ldots, \hat{x}_N[n] + \frac{1}{2}K_{1X_N}[n], u_1[n], \ldots, u_R[n], \theta)
\end{aligned} \tag{22}$$

$$\begin{aligned}
K_{3X_1}[n] &= T_s f_1(\hat{x}_1[n] + \frac{1}{2}K_{2X_1}[n], \ldots, \hat{x}_N[n] + \frac{1}{2}K_{2X_N}[n], u_1[n], \ldots, u_R[n], \theta) \\
&\vdots \\
K_{3X_N}[n] &= T_s f_N(\hat{x}_1[n] + \frac{1}{2}K_{2X_1}[n], \ldots, \hat{x}_N[n] + \frac{1}{2}K_{2X_N}[n], u_1[n], \ldots, u_R[n], \theta)
\end{aligned} \tag{23}$$

$$\begin{aligned}
K_{4X_1}[n] &= T_s f_1(\hat{x}_1[n] + K_{3X_1}[n], \ldots, \hat{x}_N[n] + K_{3X_N}[n], u_1[n], \ldots, u_R[n], \theta) \\
&\vdots \\
K_{4X_N}[n] &= T_s f_N(\hat{x}_1[n] + K_{3X_1}[n], \ldots, \hat{x}_N[n] + K_{3X_N}[n], u_1[n], \ldots, u_R[n], \theta)
\end{aligned} \tag{24}$$

The Runge–Kutta integration method or discrete time representation of the MIMO system in (19) and (20) can be shown in a more compact form as

$$\begin{aligned}
\hat{\mathbf{x}}[n+1] &= \hat{\mathbf{f}}(\hat{\mathbf{x}}[n], \mathbf{u}[n], \theta) \\
\hat{\mathbf{y}}[n+1] &= \mathbf{g}(\hat{\mathbf{x}}[n], \mathbf{u}[n])
\end{aligned} \tag{25}$$

Thus, in the case that current state variables $x_1[n] \cdots x_N[n]$ and input signals $u_1[n] \cdots u_R[n]$ at the sampling instants $t = nT_s$ are available, one-step ahead system states and outputs can be approximated via (25). If the newly obtained states are iteratively applied to the discretizated model in (25), it is possible to attain a nonlinear predictive model which assists to approximate K-step ahead future behaviour of the system and also system Jacobian (sensitivity of the system outputs with respect to control signal) which is a very vital part of model based control structures. As can be seen from the compact form in (25), determination of the current states of the system ($\hat{\mathbf{x}}[n]$) and system model parameters (θ) are vital to acquire K step ahead future system output predictions since the states of the controlled MIMO system are unavailable.

Therefore, in the following Sect. 3.2, firstly, in order to attain the currents states of the system using the available system inputs and outputs, Runge–Kutta model based EKF is introduced. Then, in order to approximate the unknown or undesignated system parameters (θ), Runge–Kutta model based online system model parameter estimator is viewed in Sect. 3.3.

3.2 Runge–Kutta Model Based EKF(RK_{EKF})

The main objective of RK_{EKF} is to approximate the current system states ($\hat{x}_1[n] \cdots \hat{x}_N[n]$) required to attain future behaviour of the system states given in (25) at any time during the control period. For this reason, it is significant to remember EKF. The simplicity and computational efficiency of EKF provides it to be the most popular tool for state estimation problems [59]. A nonlinear discrete MIMO system can be expressed as follows:

$$\begin{aligned} \mathbf{x}[n+1] &= \mathbf{h}\big(\mathbf{x}[n], \mathbf{u}[n]\big) + \mathbf{w}[n] \\ \mathbf{y}[n+1] &= \mathbf{g}\big(\mathbf{x}[n], \mathbf{u}[n]\big) + \mathbf{v}[n] \end{aligned} \tag{26}$$

where $\mathbf{x}$ stands for the N-dimensional state vector to be estimated, $\mathbf{u} \in \Re^R$ represents the input vector and $\mathbf{y} \in \Re^Q$ indicates the output vector, $\mathbf{w}$ denotes the vector of system noise with covariance matrix $\mathbf{Q}$ and $\mathbf{v}$ emblematises the vector of measurement noise with covariance matrix $\mathbf{R}$. In EKF, estimation of the system states are performed in two main steps: prediction and correction. In prediction step, the states and covariance matrix of the states are computed as follows:

$$\begin{aligned} \tilde{\mathbf{x}}^-[n] &= \mathbf{h}\big(\tilde{\mathbf{x}}[n-1], \mathbf{u}[n-1]\big) \\ \mathbf{P}^-[n] &= \mathbf{A}[n]\mathbf{P}[n-1]\mathbf{A}^T[n] + \mathbf{Q} \end{aligned} \tag{27}$$

where $\tilde{\mathbf{x}}^-[n]$ and $\mathbf{P}^-[n]$ indicate the predicted state and covariance matrix at time n, $\tilde{\mathbf{x}}[n-1]$ and $\mathbf{P}[n-1]$ denote the corrected state and covariance matrix at time $n-1$ and $\mathbf{A}[n]$ is the state transition matrix of linearized system [39,40,59]. In correction step, by means of the measurements from the system, the predicted states $\tilde{\mathbf{x}}^-[n]$ and covariance matrix of the states $\mathbf{P}^-[n]$ are corrected as follows:

$$\begin{aligned} \mathbf{K}[n] &= \mathbf{P}^-[n]\mathbf{H}^T[n]\Big(\mathbf{H}[n]\mathbf{P}^-[n]\mathbf{H}^T[n] + \mathbf{R}\Big)^{-1} \\ \tilde{\mathbf{x}}[n] &= \tilde{\mathbf{x}}^-[n] + \mathbf{K}[n]\Big(\mathbf{y}[n] - \mathbf{g}\big(\tilde{\mathbf{x}}^-[n], \mathbf{u}[n-1]\big)\Big) \\ \mathbf{P}[n] &= \Big(\mathbf{I} - \mathbf{K}[n]\mathbf{H}[n]\Big)\mathbf{P}^-[n] \end{aligned} \tag{28}$$

where $\mathbf{K}[n]$ is the Kalman gain of filter, $\tilde{\mathbf{x}}[n]$ and $\mathbf{P}[n]$ are corrected and estimated system state vector and corresponding covariance matrix [40]. Jacobian $\mathbf{A}[n]$ and $\mathbf{H}[n]$ for EKF can be obtained as follows [39,40]:

$$\mathbf{A}[n] = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=\tilde{\mathbf{x}}[n-1] \\ \mathbf{u}=\mathbf{u}[n-1]}}, \quad \mathbf{H}[n] = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=\tilde{\mathbf{x}}[n-1] \\ \mathbf{u}=\mathbf{u}[n-1]}} \tag{29}$$

Since EKF is convenient for systems in discrete form, in this study, the nonlinear systems under investigation which are in continuous form can be represented with discrete models via Runge–Kutta discretization method in (25). Thus, the Jacobian matrices $\mathbf{A}[n]$ and $\mathbf{H}[n]$ can be acquired as follows:

$$\mathbf{A}[n] = \frac{\partial \hat{\mathbf{f}}}{\partial \mathbf{x}}\bigg|_{\begin{bmatrix}\mathbf{x}=\tilde{\mathbf{x}}[n-1]\\ \mathbf{u}=\mathbf{u}[n-1]\end{bmatrix}}, \quad \mathbf{H}[n] = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\bigg|_{\begin{bmatrix}\mathbf{x}=\tilde{\mathbf{x}}[n-1]\\ \mathbf{u}=\mathbf{u}[n-1]\end{bmatrix}} \tag{30}$$

where

$$\frac{\partial \hat{\mathbf{f}}}{\partial \mathbf{x}}\bigg|_{\begin{bmatrix}\mathbf{x}=\tilde{\mathbf{x}}[n-1]\\ \mathbf{u}=\mathbf{u}[n-1]\end{bmatrix}} = \begin{bmatrix}\dfrac{\partial f_i(\tilde{\mathbf{x}}[n-1],\mathbf{u}[n-1])}{\partial \tilde{x}_j[n-1]}\end{bmatrix} = \begin{bmatrix}\dfrac{\partial \tilde{x}_i[n]}{\partial \tilde{x}_j[n-1]}\end{bmatrix} \tag{31}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, N$, and

$$\begin{aligned}\frac{\partial \tilde{x}_i[n]}{\partial \tilde{x}_j[n-1]} &= \delta_{i,j} + \frac{1}{6}\frac{\partial K_{1X_i}[n-1]}{\partial \tilde{x}_j[n-1]} + \frac{2}{6}\frac{\partial K_{2X_i}[n-1]}{\partial \tilde{x}_j[n-1]} \\ &\quad + \frac{2}{6}\frac{\partial K_{3X_i}[n-1]}{\partial \tilde{x}_j[n-1]} + \frac{1}{6}\frac{\partial K_{4X_i}[n-1]}{\partial \tilde{x}_j[n-1]}\end{aligned} \tag{32}$$

$$\begin{aligned}\frac{\partial K_{1X_i}[n-1]}{\partial \tilde{x}_j[n-1]} &= T_s \frac{\partial f_i(\tilde{x}_1[n-1],\ldots,\tilde{x}_N[n-1],u_1[n-1],\ldots,u_R[n-1])}{\partial \tilde{x}_j[n-1]} \\ &= Ts\frac{\partial f_i}{\partial x_j}\bigg|_{\begin{bmatrix}\mathbf{x}=\tilde{\mathbf{x}}[n-1]\\ \mathbf{u}=\mathbf{u}[n-1]\end{bmatrix}}\end{aligned} \tag{33}$$

$$\begin{aligned}&\frac{\partial K_{2X_i}[n-1]}{\partial \tilde{x}_j[n-1]} \\ &= T_s\left(\sum_{p=1}^{N}\frac{\partial f_i}{\partial x_p}\left(\delta_{p,j} + \frac{1}{2}\frac{\partial K_{1X_p}[n-1]}{\partial \tilde{x}_j[n-1]}\right)\right)\bigg|_{\begin{bmatrix} x_1=\tilde{x}_1[n-1]+\frac{1}{2}K_{1X_1}[n-1] \\ \vdots \\ x_N=\tilde{x}_N[n-1]+\frac{1}{2}K_{1X_N}[n-1] \\ \mathbf{u}=\mathbf{u}[n-1]\end{bmatrix}}\end{aligned} \tag{34}$$

$$\begin{aligned}&\frac{\partial K_{3X_i}[n-1]}{\partial \tilde{x}_j[n-1]} \\ &= T_s\left(\sum_{p=1}^{N}\frac{\partial f_i}{\partial x_p}\left(\delta_{p,j} + \frac{1}{2}\frac{\partial K_{2X_p}[n-1]}{\partial \tilde{x}_j[n-1]}\right)\right)\bigg|_{\begin{bmatrix} x_1=\tilde{x}_1[n-1]+\frac{1}{2}K_{2X_1}[n-1] \\ \vdots \\ x_N=\tilde{x}_N[n-1]+\frac{1}{2}K_{2X_N}[n-1] \\ \mathbf{u}=\mathbf{u}[n-1]\end{bmatrix}}\end{aligned} \tag{35}$$

$$\begin{aligned}&\frac{\partial K_{4X_i}[n-1]}{\partial \tilde{x}_j[n-1]} \\ &= T_s\left(\sum_{p=1}^{N}\frac{\partial f_i}{\partial x_p}\left(\delta_{p,j} + \frac{\partial K_{3X_p}[n-1]}{\partial \tilde{x}_j[n-1]}\right)\right)\bigg|_{\begin{bmatrix} x_1=\tilde{x}_1[n-1]+K_{3X_1}[n-1] \\ \vdots \\ x_N=\tilde{x}_N[n-1]+K_{3X_N}[n-1] \\ \mathbf{u}=\mathbf{u}[n-1]\end{bmatrix}}\end{aligned} \tag{36}$$

where

$$\delta_{i,j} = \begin{cases} 1, & \mathrm{i} = \mathrm{j} \\ 0, & \mathrm{i} \neq \mathrm{j} \end{cases}$$

In a nutshell, the current states of the system can be successfully approximated by its Runge–Kutta model deployed in the EKF algorithm via (27–36) [39]. Therefore, the EKF method employed in this section is called as Runge–Kutta model based EKF ($\mathrm{RK_{EKF}}$).

3.3 The Runge–Kutta Model Based Online Oarameter Estimation ($\mathrm{RK_{estimator}}$)

Obtaining the model parameters via conventional modeling techniques may be difficult due to the nonlinearity of the system or inadequacy of the modeling methods. When system model parameters digress from their actual values, the identification performance of $\mathrm{RK_{model}}$ aggravates [40]. Therefore, online estimation and readjustment(rehabilitation) of deteriorated or unmeasured system parameters are crucial to enhance the identification performance of $\mathrm{RK_{model}}$ [40]. If the Runge–Kutta model of the system is deployed, the current states of the system can be easily associated to its previous states $(x_1[n], \ldots, x_N[n])$, inputs $(u_1[n], \ldots, u_R[n])$ and parameters (θ) via (19,21–24) [39,40]. The parameter vector of the system $(\theta[n])$ can be adjusted as

$$\theta[n+1] = \theta[n] - \frac{\mathbf{J}_\theta^T \mathbf{e}}{\mathbf{J}_\theta^T \mathbf{J}_\theta} \tag{37}$$

where

$$\mathbf{J}_\theta = \begin{bmatrix} \frac{\partial e_1[n+1]}{\partial \theta[n]} \cdots \frac{\partial e_N[n+1]}{\partial \theta[n]} \end{bmatrix}^T = -\begin{bmatrix} \frac{\partial \hat{x}_1[n+1]}{\partial \theta[n]} \cdots \frac{\partial \hat{x}_N[n+1]}{\partial \theta[n]} \end{bmatrix}^T \tag{38}$$

and

$$\mathbf{e} = \begin{bmatrix} e_1[n+1] \\ \vdots \\ e_N[n+1] \end{bmatrix} = \begin{bmatrix} x_1[n+1] - \hat{x}_1[n+1] \\ \vdots \\ x_N[n+1] - \hat{x}_N[n+1] \end{bmatrix} \tag{39}$$

by assuming that previous state $\mathbf{x}[n]$ and current state $\mathbf{x}[n+1]$ of a non-linear system (14) are given directly (or estimated by EKF) at time $[n+1]T_s$ and that the previous control input $\mathbf{u}[n]$ is known [39]. The sensitivity of the system states with respect to model parameters $(\frac{\partial \hat{x}_i[n+1]}{\partial \theta[n]})$ necessary for the construction of Jacobian in (38) can be attained as

$$\begin{aligned} \frac{\partial \hat{x}_i[n+1]}{\partial \theta[n]} = \frac{\partial \hat{x}_i[n]}{\partial \theta[n]} + \frac{1}{6}\frac{\partial K_{1X_i}[n]}{\partial \theta[n]} + \frac{2}{6}\frac{\partial K_{2X_i}[n]}{\partial \theta[n]} \\ + \frac{2}{6}\frac{\partial K_{3X_i}[n]}{\partial \theta[n]} + \frac{1}{6}\frac{\partial K_{4X_i}[n]}{\partial \theta[n]} \end{aligned} \tag{40}$$

where

$$\begin{aligned} \frac{\partial K_{1X_i}[n]}{\partial \theta[n]} &= T_s \frac{\partial f_i(\tilde{x}_1[n], \ldots, \tilde{x}_N[n], u_1[n], \ldots, u_R[n]), \theta[n]}{\partial \theta[n]} \\ &= Ts\left[\frac{\partial f_i}{\partial \theta}\right]\Bigg|_{\begin{bmatrix} \mathbf{x} = \tilde{\mathbf{x}}[n] \\ \mathbf{u} = \mathbf{u}[n] \\ \theta = \theta[n] \end{bmatrix}} \end{aligned} \tag{41}$$

$$\frac{\partial K_{2X_i}[n]}{\partial \theta[n]} = T_s\left[\frac{\partial f_i}{\partial \theta} + \frac{1}{2}\sum_{j=1}^{N}\frac{\partial f_i}{\partial x_j}\frac{\partial K_{1X_j}[n]}{\partial \theta}\right]\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n] + \frac{1}{2}K_{1X_1}[n] \\ \vdots \\ x_N = \hat{x}_N[n] + \frac{1}{2}K_{1X_N}[n] \\ \mathbf{u} = \mathbf{u}[n] \\ \theta = \theta[n] \end{bmatrix}} \tag{42}$$

$$\frac{\partial K_{3X_i}[n]}{\partial \theta[n]} = T_s\left[\frac{\partial f_i}{\partial \theta} + \frac{1}{2}\sum_{j=1}^{N}\frac{\partial f_i}{\partial x_j}\frac{\partial K_{2X_j}[n]}{\partial \theta}\right]\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n] + \frac{1}{2}K_{2X_1}[n] \\ \vdots \\ x_N = \hat{x}_N[n] + \frac{1}{2}K_{2X_N}[n] \\ \mathbf{u} = \mathbf{u}[n] \\ \theta = \theta[n] \end{bmatrix}} \tag{43}$$

$$\frac{\partial K_{4X_i}[n]}{\partial \theta[n]} = T_s\left[\frac{\partial f_i}{\partial \theta} + \sum_{j=1}^{N}\frac{\partial f_i}{\partial x_j}\frac{\partial K_{3X_j}[n]}{\partial \theta}\right]\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n] + K_{3X_1}[n] \\ \vdots \\ x_N = \hat{x}_N[n] + K_{3X_N}[n] \\ \mathbf{u} = \mathbf{u}[n] \\ \theta = \theta[n] \end{bmatrix}} \tag{44}$$

Thus, all derivations for $\mathrm{RK}_{\mathrm{estimator}}$ can be achieved.

3.4 K-Step Ahead Future System Behaviour Predictions and Jacobian Computations

The K-step ahead future behaviour of the system can be acquired by feeding back and repetitively applying the obtained values of states to RK-model given in (25), and by assuming that the control signal vector $\mathbf{u}[n]$ remains unchanged during the prediction phase between time instants $[t + T_s \;\; t + KT_s]$ [39,40]:

$$\begin{aligned} \hat{\mathbf{x}}[n+k] &= \hat{\mathbf{f}}(\hat{\mathbf{x}}[n+k-1], \mathbf{u}[n], \theta) \\ \hat{\mathbf{y}}[n+k] &= \mathbf{g}(\hat{\mathbf{x}}[n+k-1], \mathbf{u}[n]) \quad \text{for } k = 1, \ldots, K \end{aligned} \tag{45}$$

Thus, a series of future predictions can be approximated for each output as [39]

$$\left[\hat{y}_q[n+1] \cdots \hat{y}_q[n+K]\right] \quad \text{for } q = 1, \ldots, Q \tag{46}$$

In order to derive the required derivatives for system Jacobian which is the most crucial part of the model based adaptive mechanism, firstly, (19 - 24) can be reexpressed in an iterative way as follows [39,40]:

$$\begin{aligned} \hat{x}_i[n+k] = \hat{x}_i[n+k-1] &+ \frac{1}{6}K_{1X_i}[n+k-1] + \frac{2}{6}K_{2X_i}[n+k-1] \\ &+ \frac{2}{6}K_{3X_i}[n+k-1] + \frac{1}{6}K_{4X_i}[n+k-1] \end{aligned} \tag{47}$$

for $i = 1, \ldots, N$ and

$$\hat{y}_q[n+k] = g_q(\hat{x}_1[n+k-1], \ldots, \hat{x}_N[n+k-1], u_1[n], \ldots, u_R[n]) \tag{48}$$

for $q = 1, \ldots, Q$ where

$$\begin{aligned}
K_{1X_i}[n+k-1] &= T_s f_i\big(\hat{x}_1[n+k-1], \ldots, \hat{x}_N[n+k-1], u_1[n], \ldots, u_R[n], \theta\big) \\
K_{2X_i}[n+k-1] &= T_s f_i\big(\hat{x}_1[n+k-1] + \frac{1}{2}K_{1X_1}[n+k-1], \ldots, \hat{x}_N[n+k-1] \\
&\quad + \frac{1}{2}K_{1X_N}[n+k-1], u_1[n], \ldots, u_R[n], \theta\big) \\
K_{3X_i}[n+k-1] &= T_s f_i\big(\hat{x}_1[n+k-1] + \frac{1}{2}K_{2X_1}[n+k-1], \ldots, \hat{x}_N[n+k-1] \\
&\quad + \frac{1}{2}K_{2X_N}[n+k-1], u_1[n], \ldots, u_R[n], \theta\big) \\
K_{4X_i}[n+k-1] &= T_s f_i\big(\hat{x}_1[n+k-1] + K_{3X_1}[n+k-1], \ldots, \hat{x}_N[n+k-1] \\
&\quad + K_{3X_N}[n+k-1], u_1[n], \ldots, u_R[n], \theta\big)
\end{aligned} \tag{49}$$

Thus, sensitivity of the system outputs with respect to the control signals($\frac{\partial \hat{y}_q[n+k]}{\partial u_r[n]}$) can be derived as follows:

$$\begin{aligned}
&\frac{\partial \hat{y}_q[n+k]}{\partial u_r[n]} \\
&= \left(\frac{\partial g_q}{\partial u_r} + \sum_{i=1}^{N} \frac{\partial g_q}{\partial x_i}\frac{\partial \hat{x}_i[n+k]}{\partial u_r[n]}\right)\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n+k] \\ \vdots \\ x_N = \hat{x}_N[n+k] \\ g_q = g_q(\hat{x}_1[n+k], \ldots, \hat{x}_N[n+k], u_1[n], \ldots, u_R[n]) \\ u_r = u_r[n] \end{bmatrix}}
\end{aligned} \tag{50}$$

$\frac{\partial \hat{x}_i[n+k]}{\partial u_r[n]}$ term can be acquired as follows:

$$\begin{aligned}
\frac{\hat{x}_i[n+k]}{\partial u_r[n]} &= \frac{\hat{x}_i[n+k-1]}{\partial u_r[n]} + \frac{1}{6}\frac{K_{1X_i}[n+k-1]}{\partial u_r[n]} + \frac{2}{6}\frac{K_{2X_i}[n+k-1]}{\partial u_r[n]} \\
&\quad + \frac{2}{6}\frac{K_{3X_i}[n+k-1]}{\partial u_r[n]} + \frac{1}{6}\frac{K_{4X_i}[n+k-1]}{\partial u_r[n]}
\end{aligned} \tag{51}$$

where

$$\begin{aligned}
\frac{\partial K_{1X_i}[n]}{\partial u_r[n]} &= T_s \frac{\partial f_i(\tilde{x}_1[n], \ldots, \tilde{x}_N[n], u_1[n], \ldots, u_R[n], \theta[n])}{\partial u_r[n]} \\
&= T_s\left(\frac{\partial f_i}{\partial u_r} + \sum_{j=1}^{N} \frac{\partial f_i}{\partial x_j}\frac{\partial \hat{x}_j[n+k-1]}{\partial u_r[n]}\right)\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n+k-1] \\ \vdots \\ x_N = \hat{x}_N[n+k-1] \end{bmatrix}}
\end{aligned} \tag{52}$$

and

$$\frac{\partial K_{2X_i}[n]}{\partial u_r[n]} = T_s\left(\frac{\partial f_i}{\partial u_r} + \sum_{j=1}^{N}\frac{\partial f_i}{\partial x_j}\left(\frac{\partial \hat{x}_j[n+k-1]}{\partial u_r[n]} + \frac{1}{2}\frac{\partial K_{1X_j}[n+k-1]}{\partial u_r[n]}\right)\right)\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n] + \frac{1}{2}K_{1X_1}[n] \\ \vdots \\ x_N = \hat{x}_N[n] + \frac{1}{2}K_{1X_N}[n] \end{bmatrix}} \tag{53}$$

$$\frac{\partial K_{3X_i}[n]}{\partial u_r[n]} = T_s\left(\frac{\partial f_i}{\partial u_r} + \sum_{j=1}^{N}\frac{\partial f_i}{\partial x_j}\left(\frac{\partial \hat{x}_j[n+k-1]}{\partial u_r[n]} + \frac{1}{2}\frac{\partial K_{2X_j}[n+k-1]}{\partial u_r[n]}\right)\right)\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n] + \frac{1}{2}K_{2X_1}[n] \\ \vdots \\ x_N = \hat{x}_N[n] + \frac{1}{2}K_{2X_N}[n] \end{bmatrix}} \tag{54}$$

$$\frac{\partial K_{4X_i}[n]}{\partial u_r[n]} = T_s\left(\frac{\partial f_i}{\partial u_r} + \sum_{j=1}^{N}\frac{\partial f_i}{\partial x_j}\left(\frac{\partial \hat{x}_j[n+k-1]}{\partial u_r[n]} + \frac{\partial K_{3X_j}[n+k-1]}{\partial u_r[n]}\right)\right)\Bigg|_{\begin{bmatrix} x_1 = \hat{x}_1[n] + K_{3X_1}[n] \\ \vdots \\ x_N = \hat{x}_N[n] + K_{3X_N}[n] \end{bmatrix}} \tag{55}$$

Consequently, all derivations required to constitute system Jacobian information can be attained [40].

4 Runge–Kutta RBF Neural Network Controller

4.1 An Overview of Runge–Kutta Neural Network

Let us consider a MIMO nonlinear system characterized by the following ODE

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) \tag{56}$$

with the initial condition $\mathbf{x}(0) = \mathbf{x}_0$ [40]. Assuming that f is known, one-step ahead behaviour of the system dynamics can be estimated via 4th order Runge–Kutta integration formulas as follows:

$$\mathbf{x}[n+1] = \mathbf{x}[n] + \frac{1}{6}T_i\left[K_{1\mathbf{x}}[n] + 2K_{2\mathbf{x}}[n] + 2K_{3\mathbf{x}}[n] + K_{4\mathbf{x}}[n]\right] \tag{57}$$

where T_i indicates the Runge–Kutta integration stepsize [53], $K_{1\mathbf{x}}[n]$, $K_{2\mathbf{x}}[n]$, $K_{3\mathbf{x}}[n]$ and $K_{4\mathbf{x}}[n]$ are the slopes used to compute the changing rates of the system states [53]. These slopes can be acquired as [39,43,53]

$$
\begin{aligned}
K_{1\mathbf{x}}[n] &= f(\mathbf{x}_c[n])\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]} \\
K_{2\mathbf{x}}[n] &= f(\mathbf{x}_c[n])\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]+\frac{1}{2}T_i K_{1\mathbf{x}}[n]} \\
K_{3\mathbf{x}}[n] &= f(\mathbf{x}_c[n])\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]+\frac{1}{2}T_i K_{2\mathbf{x}}[n]} \\
K_{4\mathbf{x}}[n] &= f(\mathbf{x}_c[n])\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]+T_i K_{3\mathbf{x}}[n]}
\end{aligned}
\tag{58}
$$

Using the powerful function approximation ability of neural network (NN) structures, the unknown f function can be precisely identified so as to predict these four slopes such that NN can successfully carry out long-term prediction of the state trajectory $\mathbf{x}(t)$ of the system described in (56) [40]. The structure emerged by combining the powerful integration feature of Runge–Kutta method and powerful approximation and generalization abilities of NN structure is called as *Runge–Kutta neural network* (RK-NN). The input and output relationship of the RK-NN can be expressed as

$$
\mathbf{x}[n+1] = \mathbf{x}[n] + \frac{1}{6}T_i\Big[K_{1\mathbf{x}}[n] + 2K_{2\mathbf{x}}[n] + 2K_{3\mathbf{x}}[n] + K_{4\mathbf{x}}[n]\Big] \tag{59}
$$

where

$$
\begin{aligned}
K_{1\mathbf{x}}[n] &= N_f(\mathbf{x}_c[n], \boldsymbol{\Theta})\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]} \\
K_{2\mathbf{x}}[n] &= N_f(\mathbf{x}_c[n], \boldsymbol{\Theta})\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]+\frac{1}{2}T_i K_{1\mathbf{x}}[n]} \\
K_{3\mathbf{x}}[n] &= N_f(\mathbf{x}_c[n], \boldsymbol{\Theta})\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]+\frac{1}{2}T_i K_{2\mathbf{x}}[n]} \\
K_{4\mathbf{x}}[n] &= N_f(\mathbf{x}_c[n], \boldsymbol{\Theta})\Big|_{\mathbf{x}_c[n]=\mathbf{x}[n]+T_i K_{3\mathbf{x}}[n]}
\end{aligned}
\tag{60}
$$

and $N_f(\mathbf{x}[n], \boldsymbol{\Theta})$ is a NN structure with input $\mathbf{x}[n]$ and network weights $\boldsymbol{\Theta}$. The network structure of the RK-NN is depictured in Fig. 4. It must be emphasized that the four $N_f(\mathbf{x}[n], \boldsymbol{\Theta})$ subnetworks in Fig. 4 are identical, which means they have the same network structure and utilize the same corresponding weights [43]. The slopes $K_{1\mathbf{x}}[n]$, $K_{2\mathbf{x}}[n]$, $K_{3\mathbf{x}}[n]$ and $K_{4\mathbf{x}}[n]$ can be approximated by consecutively applying the obtained output of the constituent subnetwork to itself as given in Fig. 4 and (60). The fact that n subnetworks of an n order RK-NN are identical facilitates the realization of the RK-NN in both software or hardware implementations [43]. That is, the real network size of an n-order RK-NN is the same as that of its constituent subnetwork [43].

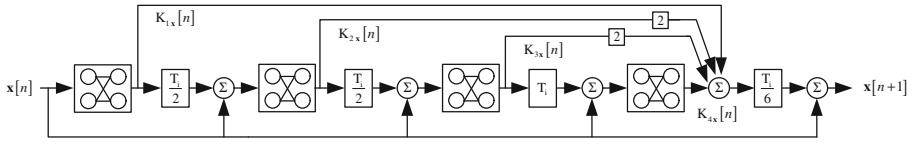

Fig. 4 Runge–Kutta neural network structure

4.2 Adjusment Rules for Runge–Kutta RBF Neural Network Controller

In this section, the adjustment rules for the weights of the RK-NN$_{\text{controller}}$ deployed to obtain feasible control signal vector are derived. Consider that the control law produced by RK-NN$_{\text{controller}}$ is expressed as

$$\begin{aligned}
\mathbf{u}[n] &= \mathbf{f}_{NN}(\boldsymbol{\Theta}[n], \mathbf{C}[n]) = \begin{bmatrix} u_1[n] \\ \vdots \\ u_R[n] \end{bmatrix}_{Rx1} \\
\boldsymbol{\Theta}[n] &= \begin{bmatrix} \alpha_1[n] \\ \vdots \\ \alpha_Z[n] \end{bmatrix}_{Zx1} \\
\mathbf{C}[n] &= \begin{bmatrix} c_1[n] \\ \vdots \\ c_R[n] \\ \vdots \\ c_N[n] \end{bmatrix} = \begin{bmatrix} u_1[n-1] \\ \vdots \\ u_R[n-1] \\ \boldsymbol{\Omega}[n] \end{bmatrix} = \begin{bmatrix} \mathbf{u}[n-1] \\ \boldsymbol{\Omega}[n] \end{bmatrix}
\end{aligned} \tag{61}$$

where R indicates the number of control inputs, Z denotes the number of adjustable parameters of RK-NN$_{\text{controller}}$ and $\mathbf{C}[n]$ represents the input feature vector of RK-NN$_{\text{controller}}$. Input vector $\mathbf{C}[n]$ is composed of entries which are previously approximated by network ($\mathbf{u}[n-1]$) and not ($\boldsymbol{\Omega}[n]$). Thus, the control signal produced by RK-NN$_{\text{controller}}$ can be reexpressed as

$$\begin{aligned}
\mathbf{u}[n] &= \begin{bmatrix} u_1[n] \\ \vdots \\ u_R[n] \end{bmatrix} = \mathbf{u}[n-1] + \Delta\mathbf{u}[n] \\
&= \mathbf{u}[n-1] + \frac{1}{6}T_c\Big[K_{1\mathbf{u}}[n] + 2K_{2\mathbf{u}}[n] + 2K_{3\mathbf{u}}[n] + K_{4\mathbf{u}}[n]\Big]
\end{aligned} \tag{62}$$

where

$$\begin{aligned}
K_{1\mathbf{u}}[n] &= N_{\mathbf{u}}(\mathbf{C}[n], \boldsymbol{\Theta}[n])\Big|_{\mathbf{C}[n]=\begin{bmatrix} \mathbf{u}[n-1] \\ \boldsymbol{\Omega}[n] \end{bmatrix}} \\
K_{2\mathbf{u}}[n] &= N_{\mathbf{u}}(\mathbf{C}[n], \boldsymbol{\Theta}[n])\Big|_{\mathbf{C}[n]=\begin{bmatrix} \mathbf{u}[n-1] + \frac{1}{2}T_cK_{1\mathbf{u}}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix}} \\
K_{3\mathbf{u}}[n] &= N_{\mathbf{u}}(\mathbf{C}[n], \boldsymbol{\Theta}[n])\Big|_{\mathbf{C}[n]=\begin{bmatrix} \mathbf{u}[n-1] + \frac{1}{2}T_cK_{2\mathbf{u}}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix}} \\
K_{4\mathbf{u}}[n] &= N_{\mathbf{u}}(\mathbf{C}[n], \boldsymbol{\Theta}[n])\Big|_{\mathbf{C}[n]=\begin{bmatrix} \mathbf{u}[n-1] + T_cK_{3\mathbf{u}}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix}}
\end{aligned} \tag{63}$$

and "T_c" indicates the Runge–Kutta integration stepsize [53] of controller, $N_{\mathbf{u}}$ denotes the regression function of NN structure, $\mathbf{u}[n]$ is the estimated states and $\boldsymbol{\Omega}[n]$ symbolises the remaining features such as previous system outputs ($\mathbf{y}$) or reference signal $\mathbf{r}$. Thus, the con-

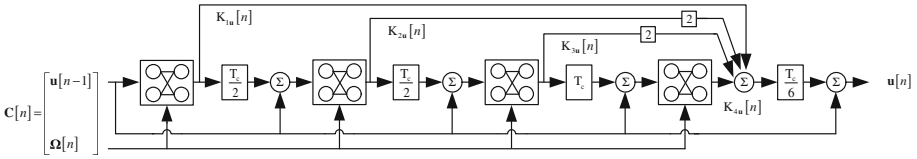

Fig. 5 Structure of RK-NN$_{\text{controller}}$

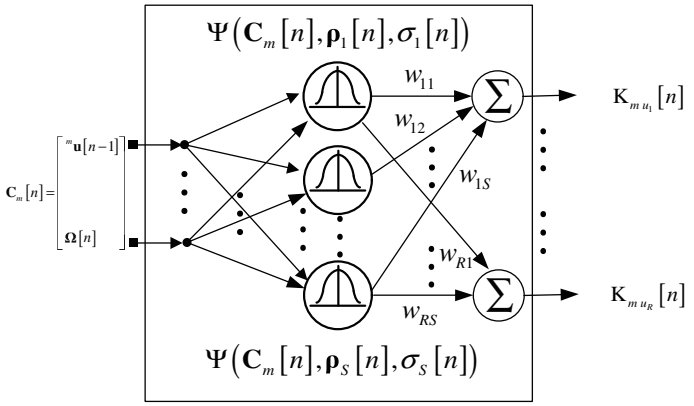

Fig. 6 Constituent RBF neural network structure for RK-NN$_{\text{controller}}$

troller structure can be illustrated as in Fig. 5. RBF neural network is deployed as constituent subnetwork in RK-NN$_{\text{controller}}$ owing to the following salient properties [60]:

- Universal approximator [44].
- Superior approximation competency [45].
- Fast learning ability under favour of locally tuned neurons [46–48] in comparison with other neural networks such as MLP.
- Possesses more compact form than other neural networks [49].

The constituent RBF neural network is depictured in Fig. 6 where $m \in \{1, 2, 3, 4\}$ and R is the number of the control inputs. The RBF neural network is composed of input, hidden and output layers. The input feature vector or data constitutes the input layer. In hidden layer, the data in input space is mapped to a hidden space via nonlinear functions [61]. The output layer, whose parameters are linear, is the layer from which the data obtained in the hidden layer is weighted and combined. The intuitive nature of RBF network structure, in which each neuron can be considered as approximating a small region of the surface around its centre, provides adjustment of the centres and bandwidth of neurons in an intelligent manner [62]. In Fig. 6, input feature vector $\mathbf{C}_m[n]$ and the entry $^m\mathbf{u}[n-1]$ are utilized since the input feature vector of the networks changes depending on the approximated slope ($K_{m\mathbf{u}}[n]$). The mth slope of the Rth output of the constituent RBF network can be computed as

$$\begin{aligned} K_{mu_R}[n] &= N_{u_R}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n]) = \sum_{i=1}^{S} w_{R,i}[n]\boldsymbol{\Psi}(\mathbf{C}_m[n], \rho_i[n], \sigma_i[n]) \\ &= \sum_{i=1}^{S} w_{R,i}[n] exp\left(\frac{-||\mathbf{C}_m[n] - \rho_i[n]||^2}{\sigma_i^2[n]}\right), \quad m \in \{1, 2, 3, 4\} \end{aligned} \tag{64}$$

where S is the number of the neurons deployed in RBF, $\rho_i[n]$ and $\sigma_i[n]$ denote center vector and the bandwidth of neurons respectively. The output vector of the constituent RBF network can be computed in matrix notation as follows:

$$\mathbf{K_{MU}} = \begin{bmatrix} \mathbf{K}_{mu_1} \\ \vdots \\ \mathbf{K}_{mu_R} \end{bmatrix} = \mathbf{W}\boldsymbol{\Psi} \tag{65}$$

where

$$\begin{gathered} \mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1S} \\ \vdots & \ddots & \vdots \\ w_{R1} & \cdots & w_{RS} \end{bmatrix}, \ \boldsymbol{\Psi} = \begin{bmatrix} \Psi_1 \\ \vdots \\ \Psi_S \end{bmatrix} \\ \Psi_S = \Psi\big(\mathbf{C}_m[n], \rho_S[n], \sigma_S[n]\big) \end{gathered} \tag{66}$$

and

$$\begin{gathered} \mathbf{C}_m = \begin{bmatrix} {}^m\mathbf{u}[n-1] \\ \vdots \\ \boldsymbol{\Omega}[n-1] \end{bmatrix}, \ \rho = \begin{bmatrix} \rho_1 \cdots \rho_S \end{bmatrix} = \begin{bmatrix} \rho_{11} & \cdots & \rho_{S1} \\ \vdots & \ddots & \vdots \\ \rho_{1N} & \cdots & \rho_{SN} \end{bmatrix} \\ \rho_{\mathbf{S}} = \begin{bmatrix} \rho_{S1} \\ \vdots \\ \rho_{SN} \end{bmatrix} = , \ \sigma = \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_S \end{bmatrix} \end{gathered} \tag{67}$$

where N is the number of the features in input vector and S is the number of the neurons utilized in RBF. RK-NN$_{\text{controller}}$ parameters to be adjusted are given as follows:

$$\boldsymbol{\Theta} = \begin{bmatrix} w_{11} \cdots w_{RS} \ \rho_{11} \cdots \rho_{SN} \ \sigma_1 \cdots \sigma_S \end{bmatrix}^T \tag{68}$$

Thus, using Levenberg–Marquardt rule in (3), the weights of the constituent subnetwork can be optimized as

$$\boldsymbol{\Theta}^{new} = \boldsymbol{\Theta}^{old} + \Delta\boldsymbol{\Theta}\,, \ \Delta\boldsymbol{\Theta} = -\big[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\big]^{-1}\mathbf{J}^T\hat{\mathbf{e}} \tag{69}$$

where $\mathbf{J}$ is a $\big(QK + R\big)x\big(S\big(R + N + 1\big)\big)$ dimension Jacobian matrix given as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \hat{e}_1[n+1]}{\partial w_{11}} & \cdots & \frac{\partial \hat{e}_1[n+1]}{\partial w_{RS}} & \frac{\partial \hat{e}_1[n+1]}{\partial \rho_{11}} & \cdots & \frac{\partial \hat{e}_1[n+1]}{\partial \rho_{SN}} & \frac{\partial \hat{e}_1[n+1]}{\partial \sigma_1} & \cdots & \frac{\partial \hat{e}_1[n+1]}{\partial \sigma_S} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{e}_Q[n+K]}{\partial w_{11}} & \cdots & \frac{\partial \hat{e}_Q[n+K]}{\partial w_{RS}} & \frac{\partial \hat{e}_Q[n+K]}{\partial \rho_{11}} & \cdots & \frac{\partial \hat{e}_Q[n+K]}{\partial \rho_{SN}} & \frac{\partial \hat{e}_Q[n+K]}{\partial \sigma_1} & \cdots & \frac{\partial \hat{e}_Q[n+K]}{\partial \sigma_S} \\ \sqrt{\lambda_1}\frac{\partial \Delta u_1[n]}{\partial w_{11}} & \cdots & \sqrt{\lambda_1}\frac{\partial \Delta u_1[n]}{\partial w_{RS}} & \sqrt{\lambda_1}\frac{\partial \Delta u_1[n]}{\partial \rho_{11}} & \cdots & \sqrt{\lambda_1}\frac{\partial \Delta u_1[n]}{\partial \rho_{SN}} & \sqrt{\lambda_1}\frac{\partial \Delta u_1[n]}{\partial \sigma_1} & \cdots & \sqrt{\lambda_1}\frac{\partial \Delta u_1[n]}{\partial \sigma_S} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{\lambda_R}\frac{\partial \Delta u_R[n]}{\partial w_{11}} & \cdots & \sqrt{\lambda_R}\frac{\partial \Delta u_R[n]}{\partial w_{RS}} & \sqrt{\lambda_R}\frac{\partial \Delta u_R[n]}{\partial \rho_{11}} & \cdots & \sqrt{\lambda_R}\frac{\partial \Delta u_R[n]}{\partial \rho_{SN}} & \sqrt{\lambda_R}\frac{\partial \Delta u_R[n]}{\partial \sigma_1} & \cdots & \sqrt{\lambda_R}\frac{\partial \Delta u_R[n]}{\partial \sigma_S} \end{bmatrix}_{[(QK+R)x(Sx(R+N+1))]} \tag{70}$$

and $\hat{\mathbf{e}}$ is the vector of the prediction errors

$$\hat{\mathbf{e}} = \begin{bmatrix} \hat{e}_1[n+1] \\ \vdots \\ \hat{e}_1[n+K] \\ \vdots \\ \vdots \\ \hat{e}_Q[n+1] \\ \vdots \\ \hat{e}_Q[n+K] \\ \sqrt{\lambda_1}\Delta u_1[n] \\ \vdots \\ \sqrt{\lambda_R}\Delta u_R[n] \end{bmatrix} = \begin{bmatrix} \hat{e}[n+1] \\ \vdots \\ \hat{e}[n+K] \\ \vdots \\ \vdots \\ \hat{e}[n+(Q-1)K+1] \\ \vdots \\ \hat{e}[n+QK] \\ \sqrt{\lambda_1}\Delta u_1[n] \\ \vdots \\ \sqrt{\lambda_R}\Delta u_R[n] \end{bmatrix}$$

$$= \begin{bmatrix} r_1[n+1] - \hat{y}_1[n+1] \\ \vdots \\ r_1[n+K] - \hat{y}_1[n+K] \\ \vdots \\ \vdots \\ r_Q[n+1] - \hat{y}_Q[n+1] \\ \vdots \\ r_Q[n+K] - \hat{y}_Q[n+K] \\ \sqrt{\lambda_1}\Big[\Delta u_1[n] - \Delta u_1[n-1]\Big] \\ \vdots \\ \sqrt{\lambda_R}\Big[\Delta u_R[n] - \Delta u_R[n-1]\Big] \end{bmatrix}_{[(QK+R)x1]} \tag{71}$$

As can be seen from the Jacobian given in(70), it is required to compute the $\frac{\partial \hat{e}_Q[n+K]}{\partial w_{RS}}$, $\frac{\partial \hat{e}_Q[n+K]}{\partial \rho_{SN}}$, $\frac{\partial \hat{e}_Q[n+K]}{\partial \sigma_S}$, $\frac{\partial \Delta u_R[n]}{\partial w_{RS}}$, $\frac{\partial \Delta u_R[n]}{\partial \rho_{SN}}$ and $\frac{\partial \Delta u_R[n]}{\partial \sigma_S}$ terms. By using chain rule, the mentioned terms can be derived as follows:

$$\frac{\partial \hat{e}_Q[n+K]}{\partial w_{ri}} = \frac{\partial \hat{e}_Q[n+K]}{\partial y_Q[n+K]} \frac{\partial y_Q[n+K]}{\partial u_r[n+1]} \left[\sum_{m=1}^{4} \frac{\partial u_r[n]}{\partial K_{mu_r}[n]} \frac{\partial K_{mu_r}[n]}{\partial w_{ri}} \right],$$
$$r \in \{1, \ldots R\},\ i \in \{1, \ldots S\}$$
$$\frac{\partial \hat{e}_Q[n+K]}{\partial \rho_{ij}} = \frac{\partial \hat{e}_Q[n+K]}{\partial y_Q[n+K]} \left[\sum_{r=1}^{R} \frac{\partial y_Q[n+K]}{\partial u_r[n]} \left[\sum_{m=1}^{4} \frac{\partial u_r[n]}{\partial K_{mu_r}[n]} \frac{\partial K_{mu_r}[n]}{\partial \rho_{ij}} \right] \right],$$
$$i \in \{1, \ldots S\},\ j \in \{1, \ldots N\}$$

$$\frac{\partial \hat{e}_Q[n+K]}{\partial \sigma_i} = \frac{\partial \hat{e}_Q[n+K]}{\partial y_Q[n+K]} \left[\sum_{r=1}^{R} \frac{\partial y_Q[n+K]}{\partial u_r[n]} \left[\sum_{m=1}^{4} \frac{\partial u_r[n]}{\partial K_{mu_r}[n]} \frac{\partial K_{mu_r}[n]}{\partial \sigma_i} \right] \right] ,$$

$$i \in \{1, \ldots S\}$$

$$\frac{\partial \Delta u_r[n]}{\partial w_{ri}} = \left[\sum_{m=1}^{4} \frac{\partial u_r[n]}{\partial K_{mu_r}[n]} \frac{\partial K_{mu_r}[n]}{\partial w_{ri}} \right] , \; r \in \{1, \ldots R\}, \; i \in \{1, \ldots S\}$$

$$\frac{\partial \Delta u_r[n]}{\partial \rho_{ij}} = \left[\sum_{m=1}^{4} \frac{\partial u_r[n]}{\partial K_{mu_r}[n]} \frac{\partial K_{mu_r}[n]}{\partial \rho_{ij}} \right] , \; i \in \{1, \ldots S\}, \; j \in \{1, \ldots N\}$$

$$\frac{\partial \Delta u_r[n]}{\partial \sigma_i} = \left[\sum_{m=1}^{4} \frac{\partial u_r[n]}{\partial K_{mu_r}[n]} \frac{\partial K_{mu_r}[n]}{\partial \sigma_i} \right] , \; i \in \{1, \ldots S\}$$

$$\frac{\partial u_r[n]}{\partial K_{1u_r}[n]} = \frac{T_c}{6}, \; \frac{\partial u_r[n]}{\partial K_{1u_r}[n]} = \frac{T_c}{3}, \; \frac{\partial u_r[n]}{\partial K_{3u_r}[n]} = \frac{T_c}{3}, \; \frac{\partial u_r[n]}{\partial K_{4u_r}[n]} = \frac{T_c}{6} \tag{72}$$

where

$$\frac{\partial K_{mu_r}[n]}{\partial w_{ri}} = \begin{cases} \frac{\partial K_{1u_r}[n]}{\partial w_{ri}} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial w_{ri}} \right] , \; \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\ \frac{\partial K_{2u_r}[n]}{\partial w_{ri}} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial w_{ri}} + \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{1u_r}[n]} \frac{\partial K_{1u_r}[n]}{\partial w_{ri}} \right] , \\ \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + \frac{1}{2} T_c K_{1u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\ \frac{\partial K_{3u_r}[n]}{\partial w_{ri}} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial w_{ri}} + \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{2u_r}[n]} \frac{\partial K_{2u_r}[n]}{\partial w_{ri}} \right] , \\ \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + \frac{1}{2} T_c K_{2u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\ \frac{\partial K_{4u_r}[n]}{\partial w_{ri}} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial w_{ri}} + \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{3u_r}[n]} \frac{\partial K_{3u_r}[n]}{\partial w_{ri}} \right] , \\ \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + T_c K_{3u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \end{cases} \tag{73}$$

$$
\frac{\partial K_{mu_r}[n]}{\partial \rho_{ij}} = \begin{cases}
\frac{\partial K_{1u_r}[n]}{\partial \rho_{ij}} = \left[\frac{\partial N_{u_r}\left(\mathbf{C}_m[n],\boldsymbol{\Theta}[n]\right)}{\partial \rho_{ij}}\right], \\
\mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\
\frac{\partial K_{2u_r}[n]}{\partial \rho_{ij}} = \left[\frac{\partial N_{u_r}\left(\mathbf{C}_m[n],\boldsymbol{\Theta}[n]\right)}{\partial \rho_{ij}} + \frac{\partial N_{u_r}\left(\mathbf{C}_m[n],\boldsymbol{\Theta}[n]\right)}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{1u_r}[n]} \frac{\partial K_{1u_r}[n]}{\partial \rho_{ij}}\right], \\
\mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + \frac{1}{2}T_c K_{1u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\
\frac{\partial K_{3u_r}[n]}{\partial \rho_{ij}} = \left[\frac{\partial N_{u_r}\left(\mathbf{C}_m[n],\boldsymbol{\Theta}[n]\right)}{\partial \rho_{ij}} + \frac{\partial N_{u_r}\left(\mathbf{C}_m[n],\boldsymbol{\Theta}[n]\right)}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{2u_r}[n]} \frac{\partial K_{2u_r}[n]}{\partial \rho_{ij}}\right], \\
\mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + \frac{1}{2}T_c K_{2u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\
\frac{\partial K_{4u_r}[n]}{\partial \rho_{ij}} = \left[\frac{\partial N_{u_r}\left(\mathbf{C}_m[n],\boldsymbol{\Theta}[n]\right)}{\partial \rho_{ij}} + \frac{\partial N_{u_r}\left(\mathbf{C}_m[n],\boldsymbol{\Theta}[n]\right)}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{3u_r}[n]} \frac{\partial K_{3u_r}[n]}{\partial \rho_{ij}}\right], \\
\mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + T_c K_{3u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix}
\end{cases} \tag{74}
$$

$$\frac{\partial K_{mu_r}[n]}{\partial \sigma_i} = \begin{cases} \frac{\partial K_{1u_r}[n]}{\partial \sigma_i} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial \sigma_i}\right], \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\ \frac{\partial K_{2u_r}[n]}{\partial \sigma_i} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial \sigma_i} + \frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{1u_r}[n]} \frac{\partial K_{1u_r}[n]}{\partial \sigma_i}\right], \\ \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + \frac{1}{2}T_c K_{1u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\ \frac{\partial K_{3u_r}[n]}{\partial \sigma_i} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial \sigma_i} + \frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{2u_r}[n]} \frac{\partial K_{2u_r}[n]}{\partial \sigma_i}\right], \\ \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + \frac{1}{2}T_c K_{2u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \\ \frac{\partial K_{4u_r}[n]}{\partial \sigma_i} = \left[\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial \sigma_i} + \frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial c_r[n]} \frac{\partial c_r[n]}{\partial K_{3u_r}[n]} \frac{\partial K_{3u_r}[n]}{\partial \sigma_i}\right], \\ \mathbf{C}_m[n] = \begin{bmatrix} c_1[n] = u_1[n-1] \\ \vdots \\ c_r[n] = u_r[n-1] + T_c K_{3u_r}[n] \\ \boldsymbol{\Omega}[n] \end{bmatrix} \end{cases} \quad (75)$$

where $r \in \{1, \ldots, R\}$, $i \in \{1, \ldots, S\}$ and $j \in \{1, \ldots, N\}$. In order to acquire the above derivations, $\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial w_{ri}}$, $\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial \rho_{ij}}$, $\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial \sigma_i}$ and $\frac{\partial N_{u_r}(\mathbf{C}_m[n],\boldsymbol{\Theta}[n])}{\partial c_j[n]}$ terms can be derived via the regression function of the constituent RBF subnetwork. Using the regression function of RBF network in (63 - 64), the above terms can be computed as

$$\begin{aligned} \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial w_{ri}} &= \boldsymbol{\Psi}(\mathbf{C}_m[n], \rho_i[n], \sigma_i[n]) = exp\left(\frac{-\|\mathbf{C}_m[n] - \rho_i[n]\|^2}{\sigma_i^2[n]}\right) \\ \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial \rho_{ij}} &= 2\, w_{ri}\, \boldsymbol{\Psi}(\mathbf{C}_m[n], \rho_i[n], \sigma_i[n])\left(\frac{c_j[n] - \rho_{ij}[n]}{\sigma_i^2[n]}\right) \\ \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial \sigma_i} &= 2\, w_{ri}\, \boldsymbol{\Psi}(\mathbf{C}_m[n], \rho_i[n], \sigma_i[n])\left(\frac{\|\mathbf{C}_m[n] - \rho_i[n]\|^2}{\sigma_i^3[n]}\right) \\ \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial c_j[n]} &= -\sum_{i=1}^{S} \frac{\partial N_{u_r}(\mathbf{C}_m[n], \boldsymbol{\Theta}[n])}{\partial \rho_{ij}} \\ &= -2\sum_{i=1}^{S} w_{ri}\, \boldsymbol{\Psi}(\mathbf{C}_m[n], \rho_i[n], \sigma_i[n])\left(\frac{c_j[n] - \rho_{ij}[n]}{\sigma_i^2[n]}\right) \end{aligned} \quad (76)$$

where $r \in \{1, \ldots, R\}$, $i \in \{1, \ldots, S\}$ and $j \in \{1, \ldots, N\}$.

4.3 Pseudocode for Proposed Adjustment Mechanism

The details related to the working principle of the proposed adjustment mechanism are presented step by step via the following pseudo code. In the control procedure given below, $\mathbf{u}^{-}[n]$ denotes the control signal produced by RK-NN$_{\text{controller}}$ obtained at the previous step and $\mathbf{u}^{+}[n]$ indicates the control signal estimated with the trained RK-NN$_{\text{controller}}$ at the current step.

Step 1 Initialization

– Initialize RK-NN$_{\text{controller}}$, RK$_{\text{EKF}}$ and RK$_{\text{model}}$ parameters.

Step 2 Computation of control signal ($\mathbf{u}^{-}[n]$) via RK-NN$_{\text{controller}}$

– Set time step n.
– Compute the control signal $\mathbf{u}^{-}[n]$ via RK-NN$_{\text{controller}}(\boldsymbol{\Theta}^{-})$ trained at previous step $(n-1)$ via (62, 63).

Step 3 Runge–Kutta model based-EKF (Prediction Phase)

– Apply candidate control signal ($\mathbf{u}^{-}[n]$) once to Runge–Kutta Model based- EKF to acquire current states via (27–36)
$\tilde{\mathbf{x}}[n] = [\tilde{x}_1[n] \cdots \tilde{x}_N[n]]$

Step 4 Runge–Kutta model based parameter estimation (Prediction phase)

– Utilize model parameters obtained at current step (n) θ_n.

Step 5 Runge–Kutta model of the system (Prediction phase)

– Apply candidate control signal ($\mathbf{u}^{-}[n]$) K-times to Runge–Kutta model using estimated states ($\tilde{\mathbf{x}}[n]$) and estimated model parameters (θ_n) so as to;

5.1 Obtain K-step ahead future behaviour of the system dynamics and prediction errors vector via (45–49, 71)
5.2 Compute objective function given by (2)
If $F(\mathbf{u}[n], \mathbf{e}_q) > \varepsilon_{closed-loop}$
Jump to next substep **5.3**.
else
Continue with RK-NN$_{\text{controller}}$ parameter adjusted at previous step and jump to **step 8**
end
5.3 Attain K-step ahead Jacobian of the system via (50–55) which is required to form Jacobian matrix for RK-NN$_{\text{controller}}$
5.4 Construct $\mathbf{J}_m$ via (11) and obtain the correction term $\delta\mathbf{u}[n]$ for control action via (13)

Step 6 Construction of Jacobian matrix for RK-NN$_{\text{controller}}$

– Construct the Jacobian matrix in (69) for RK-NN$_{\text{controller}}$ via (70–74) and K-step ahead Jacobian of the system via (50–55) which is attained at **step 5**.

Step 7: Training phase for RK-NN$_{\text{controller}}$ ($\boldsymbol{\Theta}$) (Levenberg–Marquardt Algorithm)

– Optimize the parameters of RK-NN$_{\text{controller}}$ via (69)

Step 8: Computation of control signal ($\mathbf{u}^{+}[n]$) using trained RK-NN$_{\text{controller}}$

– Compute the control signal $\mathbf{u}[n]$ produced by trained RK-NN$_{\text{controller}}$ via (62, 63)

Step 9: Control of real system

– Apply the control action ($\mathbf{u}^{+}[n] + \delta\mathbf{u}[n]$) to the real system so as to force the system dynamics to the desired position. $\delta\mathbf{u}[n]$ has already been computed in **step 5.4**. Thus, real system states and system outputs can be acquired.

Step 10: Runge–Kutta model based parameter estimation (Training phase)

– Attain $\mathbf{J}_\theta$ via (38, 40–44) and $\mathbf{e}$ via (39)
– Adjust the deteriorated system model parameters using Runge–Kutta model based parameter estimation block as $\theta[n+1] = \theta[n] - \frac{\mathbf{J}_\theta^T \mathbf{e}}{\mathbf{J}_\theta^T \mathbf{J}_\theta}$ given in (37).

Step 12: Incrementation of time step

– Increment $n = n + 1$ and back to **step 2** for new cycle.

5 Simulation Results

The performance evaluation of the proposed model predictive Runge–Kutta neural network controller is assessed on a nonlinear three tank system (TTS) and Van de Vusse system. Nathless, it is possible to deploy the proposed adjustment mechanism to control a diverse range of nonlinear MIMO systems and to solve the fundamental control problems that frequently appear in practice such as nonlinearity, instability, etc.

5.1 Three-Tank System

The three tank system(TTS) is ofttimes utilized to evaluate the control performance of MIMO type controller structures proposed for nonlinear MIMO systems in technical literature [39, 42,57]. The schematic diagram of the TTS is depictured in Fig. 7. The whole system is

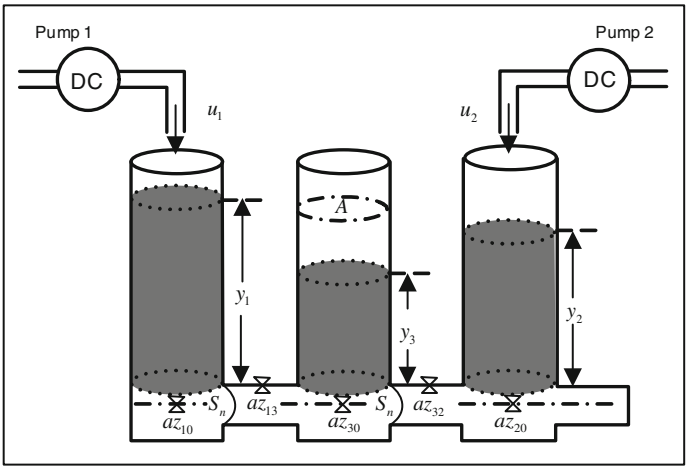

Fig. 7 Three tank system

composed of three ideal cylindrical tanks serially interconnected to each other, two pumps in order to move the water at the bottom of the reservoir, valves providing the flow between tanks and water reservoir in the bottom. The system dynamics describing the dynamic behavior of the system can be denoted via the following differential equations:

$$\begin{aligned}\dot{y}_1 &= \frac{1}{A}\Big[u_1(t) - Q_{13}(t) - Q_{10}(t)\Big]\\ \dot{y}_2 &= \frac{1}{A}\Big[u_2(t) + Q_{32}(t) - Q_{20}(t)\Big]\\ \dot{y}_3 &= \frac{1}{A}\Big[Q_{13}(t) - Q_{32}(t) - Q_{30}(t)\Big]\end{aligned} \tag{77}$$

where

$$\begin{aligned}Q_{13}(t) &= az_{13}\, S_n\, sgn\big(y_1(t) - y_3(t)\big)\sqrt{2g|y_1(t) - y_3(t)|}\\ Q_{32}(t) &= az_{32}\, S_n\, sgn\big(y_3(t) - y_2(t)\big)\sqrt{2g|y_3(t) - y_2(t)|}\\ Q_{10}(t) &= az_{10}\, S_n\, \sqrt{2g|y_1(t)|}\\ Q_{20}(t) &= az_{20}\, S_n\, \sqrt{2g|y_2(t)|}\\ Q_{30}(t) &= az_{30}\, S_n\, \sqrt{2g|y_3(t)|}\end{aligned} \tag{78}$$

and $y_i(t)$ denotes the liquid level of the ith tank as the ith output, $u_i(t)$ stands for the supply flow rate of the ith pump as the ith input and $Q_{ji}(t)$ emblematises the flow rate between tank j and i [39,40,42,63]. The diorism and numerical values of all symbols utilized in Fig. 7 and (77, 78) are given in Table 1. The closed-loop TTS control objective is to independently force the liquid levels of tank 1 and tank 2 to the desired reference levels by adjusting the flow rates of pump 1 ($u_1(t)$) and pump2 ($u_2(t)$) within allowed intervals [39,40,42,63,64]. For this purpose, the controlled outputs of the system are chosen as $y_1(t)$ and $y_2(t)$ while $u_1(t)$ and $u_2(t)$ are control inputs.The third output of the system ($y_3(t)$), liquid level of the middle tank, is an uncontrollable dynamic [64]. In the simulations, sampling time is utilized as $Ts = 1$ s and the magnitudes of the control signals are astricted as $u_{1_{min}} = u_{2_{min}} = 0\,\mathrm{m}^3/\mathrm{s}$ and $u_{1_{max}} = u_{2_{max}} = 10^{-4}\,\mathrm{m}^3/\mathrm{s}$ [39]. The obtained control signals are applied to the system during $\tau_{1_{min}} = \tau_{2_{min}} = \tau_{1_{max}} = \tau_{2_{max}} = T_s = 1.0$ s continuation period. Runge–Kutta based EKF is deployed to estimate the dynamic behavior of the states since only the input-output signals are available and states of the system can not be measured [39]. The performance of the system has been assessed for three separate cases:

(1) Nominal case with no measurement noise and parametric uncertainty.
(2) Measurement noise is added to the controlled outputs of the system.
(3) Parametric uncertainty in a system parameter.

Table 1 System parameters for three tank system [39,40,63]

Parameter description	Value
az_{13}: outflow coefficient between tank 1 and tank 3	0.52
az_{32}: outflow coefficient between tank 3 and tank 2	0.55
az_{10}: outflow coefficient from tank 1 to reservoir	0.26
az_{20}: outflow coefficient from tank 2 to reservoir	0.28
az_{30}: outflow coefficient from tank 3 to reservoir	0.45
A: cross section of the cylinders	0.0154 [m^2]
S_n: section of connection pipe n	5×10^{-5} [m^2]
g: gravitation coefficient	9.81[m/s^2]

For all cases, the number of the neurons in RBF networks (S) is assigned as "2" and input feature vectors of RK-NN$_{\text{controller}}$ are designated as $\mathbf{C}[n]=\begin{bmatrix} u_1[n-1] \\ u_2[n-1] \\ y_1[n] \\ y_2[n] \\ e_1[n]-e_1[n-1] \\ e_1[n] \\ e_1[n]-2e_1[n-1]+e_1[n-2] \\ e_2[n]-e_2[n-1] \\ e_2[n] \\ e_2[n]-2e_2[n-1]+e_2[n-2] \end{bmatrix} = \begin{bmatrix} \mathbf{u}[n-1] \\ \mathbf{\Omega}[n] \end{bmatrix}$. The prediction horizon of model predictive(MP) structure is settled as $K = 5$.

5.1.1 Nominal Case with No Measurement Noise and Parametric Uncertainty

The tracking performance of the RK-NN$_{\text{controller}}$ for the nominal case is depicted in Fig. 8a, d for staircase reference signal and the produced control signals by RK-NN$_{\text{controller}}$ and control signal correction terms are also illustrated in Fig. 8b, c, e, f. As can be palpably seen from Fig. 8, the RK-NN$_{\text{controller}}$ can successfully perform the control task by only small transient and steady-state errors. There is a crucial point required to be examined and interpreted in Fig. 8. While the second reference ($r_2(t)$) signal is settled as 0.15 m, the first reference signal ($r_1(t)$) is gradually altered between 0−800 s in the style of staircase. The fluctuation in tank 1 interacts with tank 2 through tank 3 [40]. Therefore, in order to keep the $r_2(t)$ signal depending on incremental value of $r_1(t)$ and inherently $u_1(t)$, it is required to reduce $u_2(t)$, which can be clearly observed in Fig. 8b, e. In a nutshell, it can be concluded that RK-NN$_{\text{controller}}$ effectively copes with and supresses the coupling occurring between tank 1 and 2. The convergence and adjustments of the RK-NN$_{\text{controller}}$ parameters to their optimal values depending on alternations on reference signals are given in Fig. 9. Thus, it can be patently seen from Fig. 9 that RK-NN$_{\text{controller}}$ parameters are evolved in order to learn and attune the consistently excited dynamics of the controlled system. As previously mentioned over Fig. 2, the control signal ($\mathbf{u}[n] = \mathbf{u}_{\text{RK-NN}}[n] + \delta\mathbf{u}[n]$) has two significant parts where $\mathbf{u}_{\text{RK-NN}}[n]$ is the control signal produced by RK-NN$_{\text{controller}}$ and $\delta\mathbf{u}[n]$ is the correction term. The duty

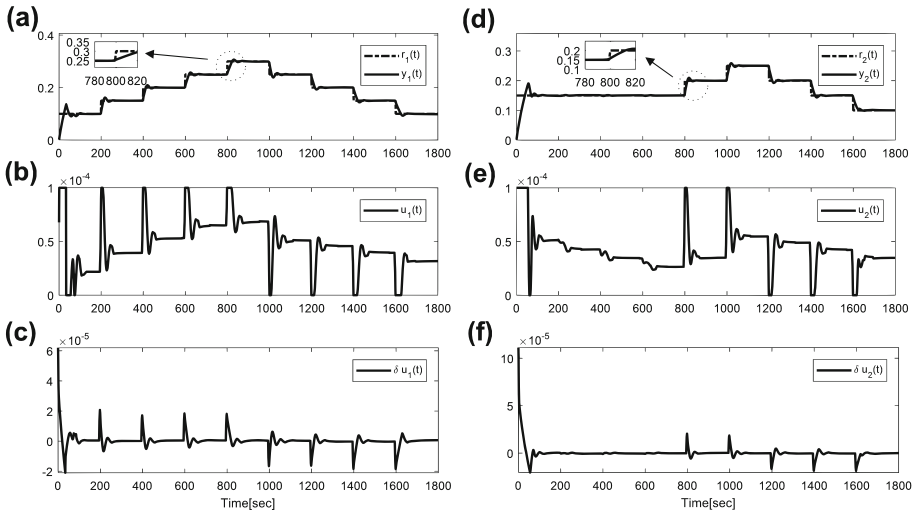

Fig. 8 System outputs (**a**, **d**), control signals (**b**, **e**) and correction terms (**c**, **f**) of three tank system for the nominal case with no measurement noise and parametric uncertainty(staircase reference inputs)

share of the RK-NN$_{\text{controller}}$($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term($\delta\mathbf{u}[n]$) are illustrated in Fig. 10. RK-NN$_{\text{controller}}$ carries out a large part of the control task. The tracking performance of the RK-NN$_{\text{controller}}$ for sinusoidal reference signals is examined as illustrated in Fig. 11 and it is observed that RK-NN$_{\text{controller}}$ ably tracks the sinusoidal reference inputs. The share of the control task between RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term($\delta\mathbf{u}[n]$) are depicted in Fig. 12. As can be explicitly seen from Fig. 12, at the beginning of the control process, the contribution of the correction term to control signal is 100%. As the controller parameters are adjusted and controller parameters converge to their optimal values, the control task is taken over by RK-NN$_{\text{controller}}$.

5.1.2 Meaurement Noise Case

Due to the measurement devices, it is an inevitable fact that measurement noise has an effect on the measured system dynamics. Therefore, control performance evaluation of adaptive controller structures under measurement noise is crucial since the designed controller structures are desired to be robust against the measurement noise conditions. Accordingly, in order to evaluate the performance and robustness of the controller under the influence of measurement noise, additive zero mean Gaussian noises with standard deviations of $\sigma_{y_1(t)} = \sigma_{y_2(t)} = 0.05$ are added to the measured controlled outputs of the system ($y_1(t)$, $y_2(t)$). The performance of the controller for staircase reference input and control signals produced by RK-NN$_{\text{controller}}$ are illustrated in Fig. 13. As can be seen from Fig. 13, in spite of measurement noise, the controlled dynamics of the system can be successfully moved to the desired reference signals by only small transient and steady-state errors. The control task is mostly undertaken by RK-NN$_{\text{controller}}$ as given in Fig. 14. Hence, it can be concluded that the RK-NN$_{\text{controller}}$ is more dominant than the correction term.

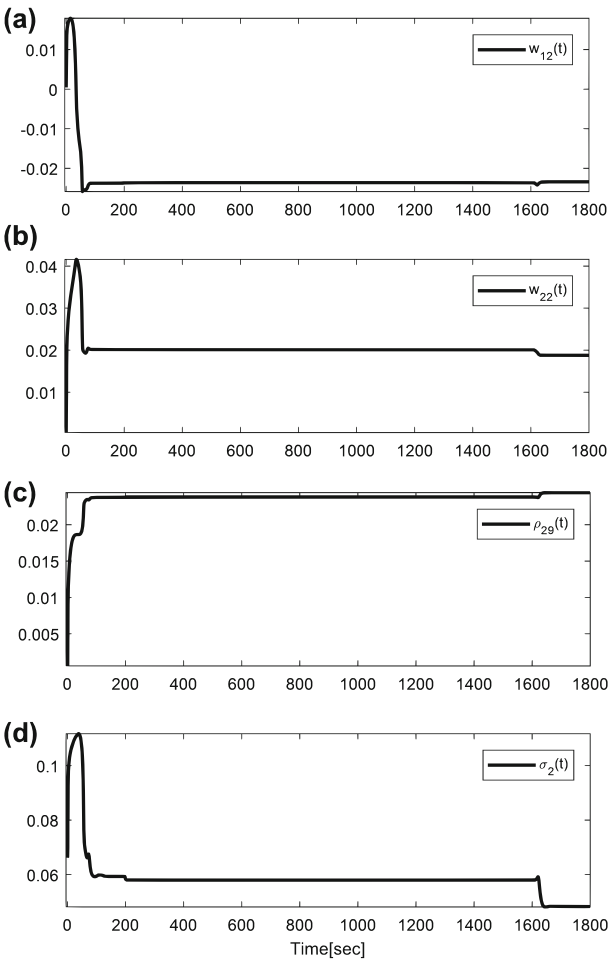

Fig. 9 Adaptation of RK-$NN_{controller}$ parameters for three tank system (staircase reference inputs)

5.1.3 Parametric Uncertainty in System Parameters

Since the parameters of RK-$NN_{controller}$ are optimized by taking into account K-step ahead future behaviour of the controlled system whose dynamics are identified by means of RK_{model}, control performance is excessively influenced by the identification accuracy of the RK_{model}. Therefore, in order to attain unknown RK model parameters, $RK_{estimator}$ subblock in RK_{model} is deployed to enhance the identification performance of RK_{model} as can be seen from Fig. 2. The tracking performance of the proposed adjustment mechanism when there is an uncertainty in a system parameter and also model parameter estimation performance of $RK_{estimator}$ subblock utilized in RK_{model} are appraised in this subsection. For this purpose, the desired reference signals are set to 0.25 and 0.2 m for tank 1 and tank 2, respectively. The numerical value of the valve parameter between tank 1 and tank 3 (outflow parameter $az_{13}(t)$) is destinated as the time-varying uncertain system parameter, which varies as $az_{13}(t) = 0.52 + 0.28 \sin\left(\frac{2}{150}\pi t\right)$. The tracking performance of the RK-$NN_{controller}$ for the

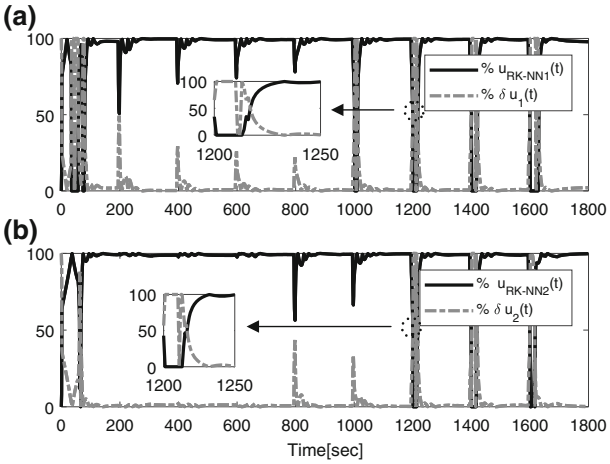

Fig. 10 Duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term ($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the nominal case with no measurement noise and parametric uncertainty (staircase reference inputs)) (Three tank system)

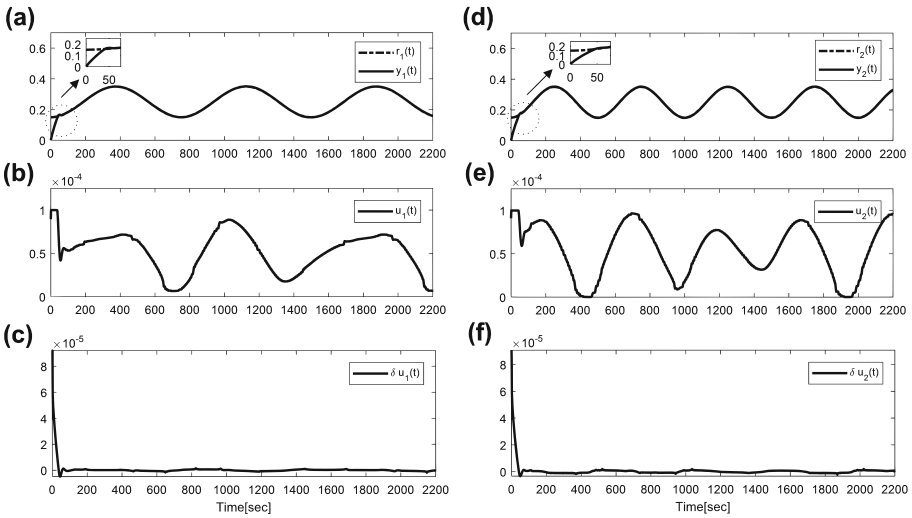

Fig. 11 System outputs (**a**, **d**), control signals (**b**, **e**) and correction terms(**c**, **f**) of three tank system for the nominal case with no measurement noise and parametric uncertainty (sinusoidal reference inputs)

uncertainty in system parameter case is shown in Fig. 15. Also, as can be seen from Fig. 15 (**e**), RK$_{\text{estimator}}$ subblock precisely approximates the correct values of the uncertain system parameter in a timely manner and then maintains it in the long run [39]. For uncertainty in system parameter case, the control task is again mostly undertaken by RK-NN$_{\text{controller}}$ as illustrated in Fig. 16 as in nominal and measurement noise cases.

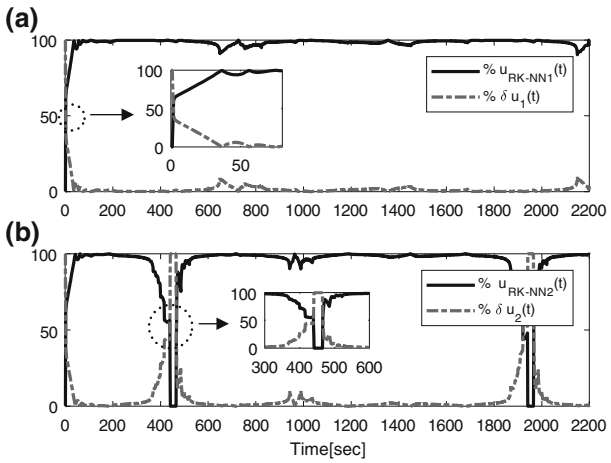

Fig. 12 Duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term ($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the nominal case with no measurement noise and parametric uncertainty (sinusoidal reference inputs)) (Three tank system)

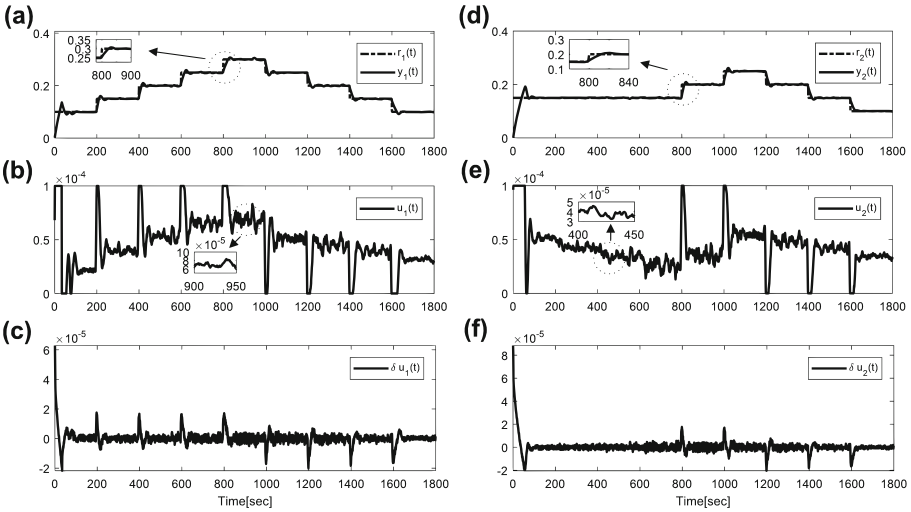

Fig. 13 System outputs (**a**, **d**), control signals (**b**, **e**) and correction terms (**c**,**f**) of three tank system for the case with measurement noise (staircase reference inputs)

5.2 Van de Vusse Chemical Reaction

The second nonlinear MIMO system deployed to execute the performance evaluation of the proposed RK-NN$_{\text{controller}}$ is Van de Vusse chemical reaction. Since the system is a non-isothermal system influenced by thermal effect, and the resulting system exhibits strictly non-minimum-phase behavior [39], Van de Vusse chemical reaction has been frequently employed to evaluate the control performance of the developed adaptive control methodologies in adaptive nonlinear control theory. The system is required to be controlled actively so as to cope with the occurring divergent behaviours since the system comprises severe nonlinearity with strong coupling between its dynamics. The chemical reaction mechanism attributed to Van de Vuse is expressed via the following reaction scheme.

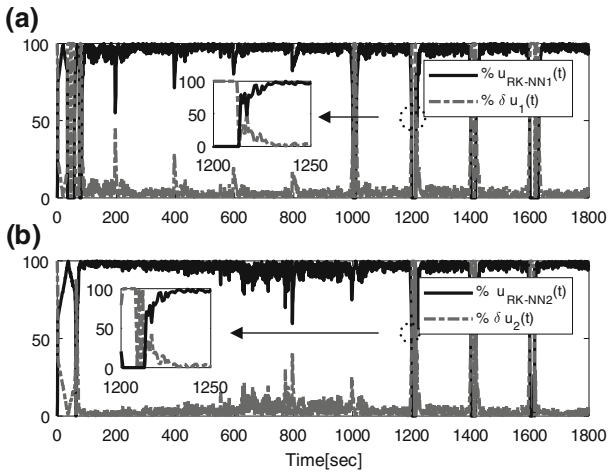

Fig. 14 Duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term ($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the case with measurement noise (staircase reference inputs)) (Three tank system)

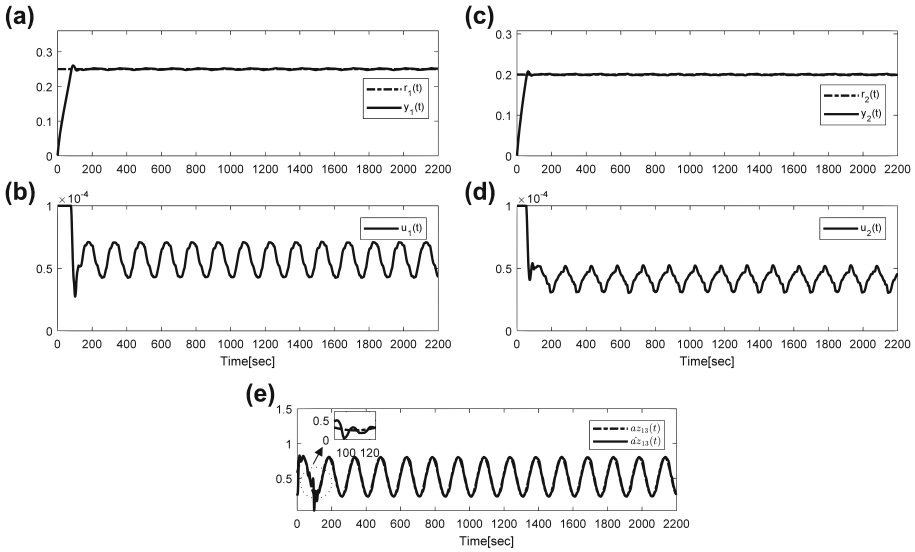

Fig. 15 System outputs (**a**,**c**), control signals (**b**,**d**) and uncertain outflow parameter ($az_{13}(t)$) (**e**) and its estimation for the case with parametric uncertainty (Three tank system)

$$\begin{aligned} &\mathbf{A} \xrightarrow{k_1} \mathbf{B} \xrightarrow{k_2} \mathbf{C} \\ &2\mathbf{A} \xrightarrow{k_3} \mathbf{D} \end{aligned} \tag{79}$$

where **A** denotes the inlet reactant, **B** stands for the desired product, **C** and **D** are unwanted by products and k_i's denote the reaction rates [65–69]. It is aimed to produce the cyclopentenol (**B**) from cyclopentadiene (**A**) by acid-catalysed electrophilic addition of water in dilute solution in the considered chemical reaction given in (79) [40,66]. When cyclopentadiene (**A**) and cyclopentenol (**B**) go under reaction, dicyclopentadiene (**D**) is produced as a side product, and cyclopentanediol (**C**) is a consecutive product by addition of another water molecule [66]. The detailed chemical reaction scheme is given as

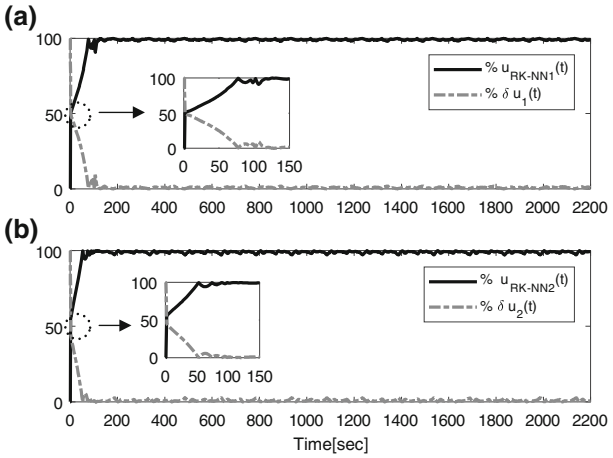

Fig. 16 Duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the case with parametric uncertainty) (Three tank system)

$$\underset{Cyclopentadiene(\mathbf{A})}{\mathbf{C_5H_6}} \xrightarrow{+H_2O(k_1)} \underset{Cyclopentenol(\mathbf{B})}{\mathbf{C_5H_7OH}} \xrightarrow{+H_2O(k_2)} \underset{Cyclopentanediol(\mathbf{C})}{\mathbf{C_5H_8(OH)_2}}$$
$$\underset{Cyclopentadiene(\mathbf{A})}{\mathbf{2C_5H_6}} \xrightarrow{k_3} \underset{Dicyclopentadiene(\mathbf{D})}{\mathbf{C_{10}H_{12}}} \tag{80}$$

The system dynamics describing the mole balances for species **A** and **B**, and the energy balance for the reactor given to realize the chemical reaction scheme in (79–80) can be described via the following differential equations:

$$\begin{aligned}
\dot{C}_A(t) &= \frac{F}{V}\Big(C_{A0} - C_A(t)\Big) - k_{10}e^{\frac{-E_1}{T}}C_A(t) - k_{30}e^{\frac{-E_3}{T}}C_A^2(t) \\
\dot{C}_B(t) &= -\frac{F}{V}C_B(t) + k_{10}e^{\frac{-E_1}{T}}C_A(t) - k_{20}e^{\frac{-E_2}{T}}C_B(t) \\
\dot{T}(t) &= \frac{1}{\rho C_p}\left[k_{10}e^{\frac{-E_1}{T}}C_A(t)(-\Delta H_1) + k_{20}e^{\frac{-E_2}{T}}C_B(t)(-\Delta H_2) + k_{30}e^{\frac{-E_3}{T}}C_A^2(t)(-\Delta H_3)\right] \\
&\quad + \frac{F}{V}(T_0 - T(t)) + \frac{Q}{\rho C_p}
\end{aligned} \tag{81}$$

where C_A and C_B stand for the molar concentrations of **A** and **B**, T denotes the reactor temperature, F/V represents the dilution rate and Q is the rate of the heat added or removed per unit volume, C_p and ρ symbolise the heat capacity and density of the reacting mixture, respectively, ΔH_i emblematise the heats of the reaction and E are activation energies [39,67,70,71]. The descriptions and values of the physical and chemical parameters are tabulated in Table 2 [39,65,70,71]. In the closed-loop MIMO control system, the aim is to control the molar concentration of **B** ($y_1 = C_B$) and the temperature of the reactor ($y_2 = T$) by regulating the dilution rate ($u_1 = F/V$) and the rate of heat added or removed per unit volume ($u_2 = Q$) within allowed intervals [39,70]. In the simulations, sampling time is chosen as $T_s = 0.01$ h and the lower and upper limits of the control signals are given as $u_1 = [0, 500]\ \text{h}^{-1}$ and $u_2 = [-1000, 0]$ kJ/l h [39]. The system is exposed to the computed control signal during $\tau_{1_{min}} = \tau_{2_{min}} = \tau_{1_{max}} = \tau_{2_{max}} = T_s = 0.01$ h continuation period.

Table 2 Physico-chemical parameters for Van de Vusse [39,65,71]

Description of parameter	Symbol	Value of parameter
Collision factor for reaction k_1	k_{10}	$1.287x10^{12}$ $[h^{-1}]$
Collision factor for reaction k_2	k_{20}	$1.287x10^{12}$ $[h^{-1}]$
Collision factor for reaction k_3	k_{30}	$9.043x10^{9}$ $[h^{-1}l/mol]$
Activation energy for reaction k_1	E_1	9758.3 [K]
Activation energy for reaction k_2	E_2	9758.3 [K]
Activation energy for reaction k_3	E_3	8560.0 [K]
Enthalpies of reaction k_1	ΔH_1	4.2 [kJ/mol]
Enthalpies of reaction k_2	ΔH_2	-11 [kJ/mol]
Enthalpies of reaction k_3	ΔH_3	-41.85 [kJ/mol]
The concentration of A in the feed stream	C_{A0}	5.0 [mol/l]
Feed temperature	T_0	403.15 [K]
Density	ρ	0.9342 [kg/l]
Heat capacity	C_p	3.01 [kJ/kgK]
Reactor volume	V	10.0 [l]

The performance evaluation of the closed-loop system has been carried out for three separate cases:

(1) Nominal case with no measurement noise and parametric uncertainty.
(2) Measurement noise case (the measured output of the system is subjected to measurement noise).
(3) Parametric uncertainty case in a system parameter.

For all cases, the number of the neurons in RBF networks (S) is settled as "2" and input feature vectors of RK-$NN_{controller}$ are designated as $\mathbf{C}[n] = \begin{bmatrix} u_1[n-1] \\ u_2[n-1] \\ y_1[n] \\ y_2[n] \\ e_1[n]-e_1[n-1] \\ e_1[n] \\ e_1[n]-2e_1[n-1]+e_1[n-2] \\ e_2[n]-e_2[n-1] \\ e_2[n] \\ e_2[n]-2e_2[n-1]+e_2[n-2] \end{bmatrix} = \begin{bmatrix} \mathbf{u}[n-1] \\ \mathbf{\Omega}[n] \end{bmatrix}$. The prediction horizon of model predictive(MP) structure is assigned as $K = 5$.

5.2.1 Nominal Case with No Measurement Noise and Parametric Uncertainty

The control performance of the RK-$NN_{controller}$ for the nominal case is illustrated in Fig. 17a, d for staircase reference signals. The controlled system outputs closely track the reference signals with very small steady-state errors except for transient states arising as a consequence of the abrupt alternations in reference signals. The control signals produced by RK-$NN_{controller}$ and control signal correction terms are also depicted in Fig. 17b, c, e,

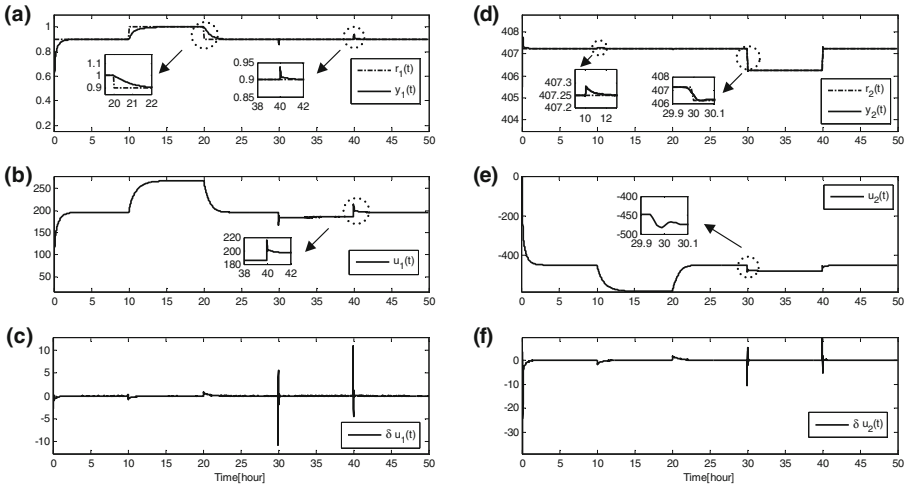

Fig. 17 System outputs (**a**, **d**), control signals (**b**, **e**) and correction terms (**c**, **f**) of Van de Vusse system for the nominal case with no measurement noise and parametric uncertainty(staircase reference inputs)

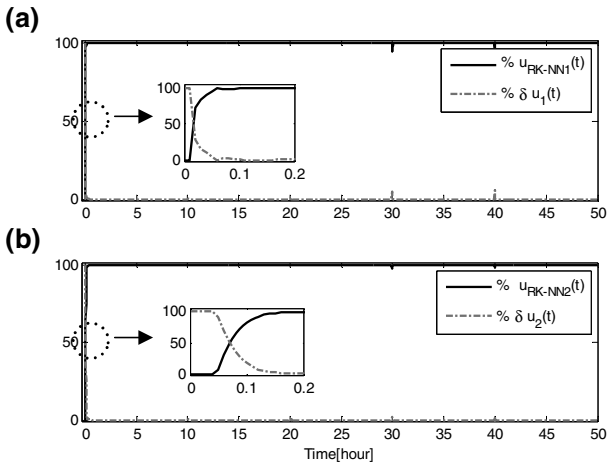

Fig. 18 Duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term ($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the nominal case with no measurement noise and parametric uncertainty (staircase reference inputs)) (Van de Vusse system)

f. If it is focused on the behaviour of the system outputs at [10, 20] h and [30,40] h, it is observed that the RK-NN$_{\text{controller}}$ successfully attunes and also tolerates the occurring strong coupling between C_B and reactor temperature (T). The duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term($\delta\mathbf{u}[n]$) are given in Fig. 18. As can be seen from Fig. 18, RK-NN$_{\text{controller}}$ has undertaken almost all control task except the first moments since it takes some time until the weights of RK-NN$_{\text{controller}}$ converge their optimal values. The tracking performance of the RK-NN$_{\text{controller}}$ for sinusoidal type reference inputs is also evaluated as given in Fig. 19. Whereas the reactor temperature is assigned as constant during control, C_B is compelled to alter sinusoidally as illustrated in Fig. 19. The control signals and corresponding corrections terms are illustrated in Fig. 19b, c, e, f. The control duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term($\delta\mathbf{u}[n]$) parts are illustrated in Fig. 20.

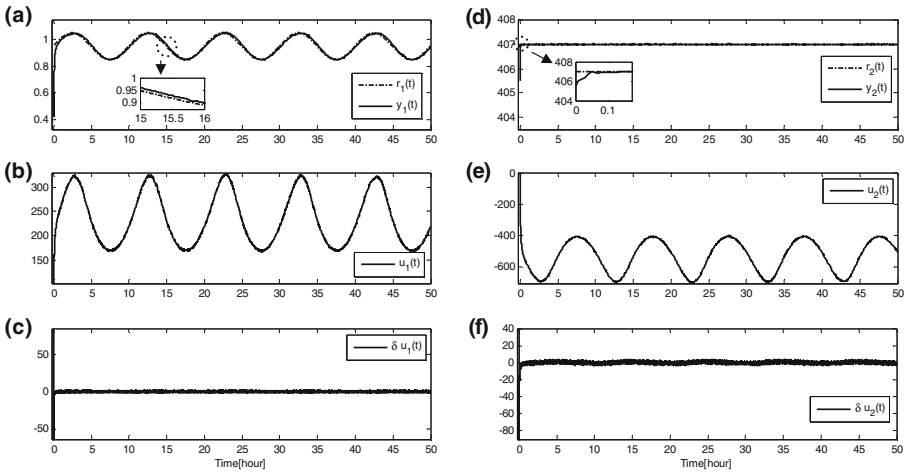

Fig. 19 System outputs (**a**, **d**), control signals (**b**, **e**) and correction terms (**c**, **f**) of Van de Vusse system for the nominal case with no measurement noise and parametric uncertainty (sinusoidal reference input)

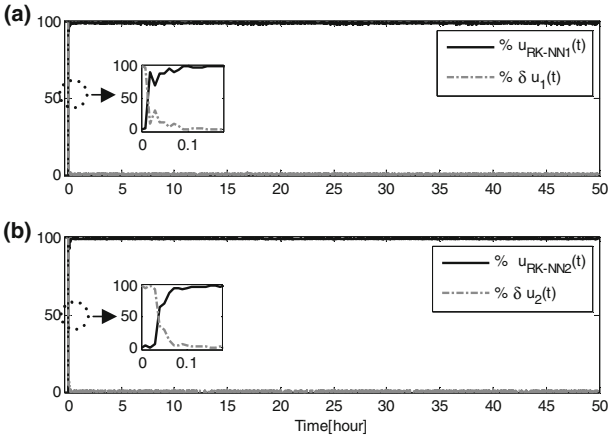

Fig. 20 Duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term ($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the nominal case with no measurement noise and parametric uncertainty (sinusoidal reference input)) (Van de Vusse system)

5.2.2 Meaurement Noise Case

Since nonlinear MIMO systems can generally be exposed to noise resulting from measurement devices, the robustness and control performance of the RK-NN$_{\text{controller}}$ with respect to measurement noise is evaluated. For this purpose, additive zero mean Gaussian noises with standard deviations of $\sigma_{C_B(t)} = 0.01, \sigma_{T(t)} = 0.001$ for C_B and for T are supplemented to the measured outputs of the system ($y_1(t)$, $y_2(t)$). The performance of the controller for staircase reference input and control signals produced by RK-NN$_{\text{controller}}$ for the case when measurement noise is added are illustrated in Fig. 21. As given in Fig. 22, the correction is only effective when the controller parameters are not optimal initially. In the rest of the control process, RK-NN$_{\text{controller}}$ has undertaken almost all control task.

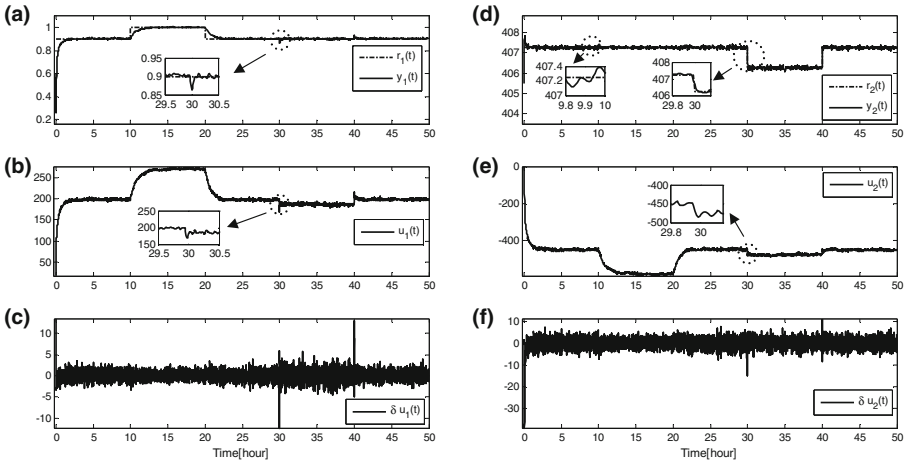

Fig. 21 System outputs (**a**, **d**), control signals (**b**, **e**) and correction terms (**c**, **f**) of Van de Vusse system for the case with measurement noise(staircase reference inputs)

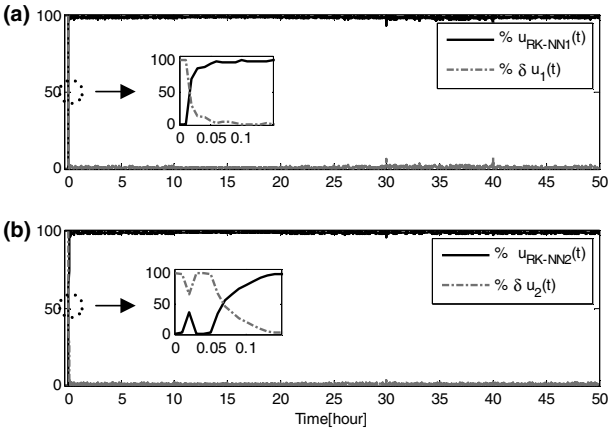

Fig. 22 Duty share of the RK-NN$_{\text{controller}}$ ($\mathbf{u}_{\text{RK-NN}}[n]$) and correction term ($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the case with measurement noise (staircase reference inputs)) (Van de Vusse system)

5.2.3 Parametric Uncertainty in System Parameters

The performance of the RK-NN$_{\text{controller}}$ mostly depends on the identification accuracy of the RK$_{\text{model}}$. For this purpose, in order to approximate the unknown system parameters, RK$_{\text{estimator}}$ subblock is introduced in RK$_{\text{model}}$. In order to peruse robustness of the RK-NN$_{\text{controller}}$ for parametric uncertainty case and evaluate parameter estimation performance of RK$_{\text{estimator}}$ subblock, $C_{A0}(t)$ parameter, which varies as $C_{A0}(t) = 5 + 0.5\sin\left(\frac{2}{10}\pi t\right)$, is chosen as an uncertain time-varying parameter while the desired reference signals are assigned as 0.95 and 407.25 for $C_B(t)$ and T. The tracking performance of the proposed RK-NN$_{\text{controller}}$ adjustment mechanism and also model parameter estimation performance of RK$_{\text{estimator}}$ subblock are illustrated in Fig. 23a, b, c, d and Fig. 23e. It is observed in Fig. 23e that the proposed Runge–Kutta based model parameter estimation block(RK$_{\text{estimator}}$

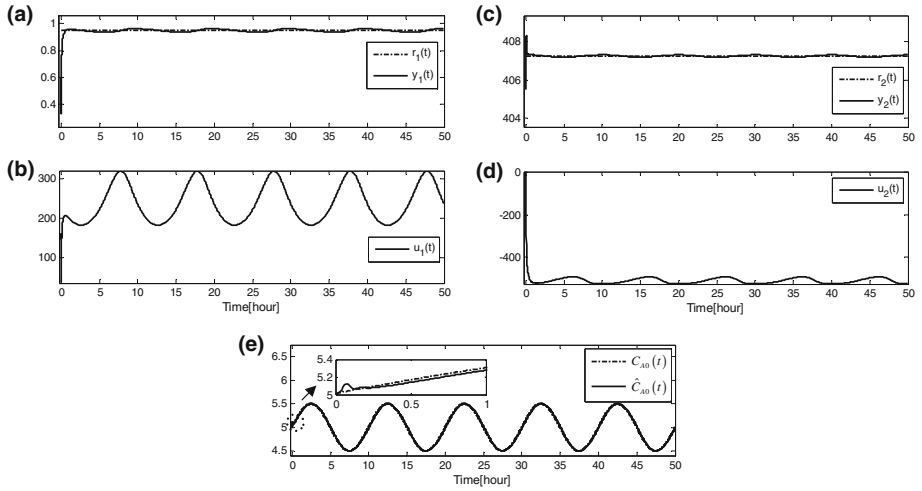

Fig. 23 System outputs (**a**, **c**), control signals (**b**, **d**) and uncertain system parameter ($C_{A0}(t)$) (**e**) and its estimation for the case with parametric uncertainty (Van de Vusse system)

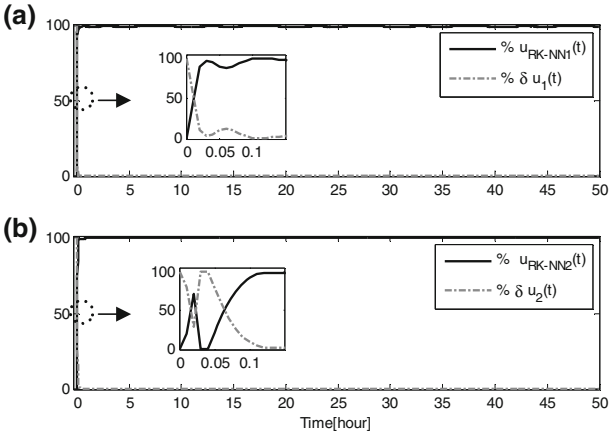

Fig. 24 Duty share of the RK-$NN_{controller}$ ($\mathbf{u}_{RK\text{-}NN}[n]$) and correction term ($\delta\mathbf{u}[n]$) for $u_1(t)$ (**a**) and $u_2(t)$ (**b**) (the case with parametric uncertainty) (Van de Vusse system)

subblock) estimates the correct values of the uncertain parameter precisely and in a timely manner and then maintains it in the long run [39]. As illustrated in Fig. 24, for uncertain system parameter case, the RK-$NN_{controller}$($\mathbf{u}_{RK\text{-}NN}[n]$) takes over the control process.

5.3 Execution Time of RK-$NN_{controller}$ Algorithm

In order to assess the applicability potential of the proposed mechanism in real time, execution time of each operation in the control algorithm have been registered for each case during every sampling period, then the maximum response times of the each operation have been listed in Table 3. When the execution times of the control algorithm for both systems for all cases are analyzed, it is observed that the maximum response times for RK-$NN_{controller}$ are

Table 3 Computation times [ms] for proposed controller (RK-$NN_{controller}$)

Systems	Three tank sytem			Van de Vusse		
Operations	Noiseless	Noisy	Uncertain	Noiseless	Noisy	Uncertain
EKF state estimation	2.2313	0.84626	2.0881	1.1238	1.4159	1.1308
K-step prediction	2.8704	1.6491	3.9	1.5451	1.8898	1.616
RK-$NN_{controller}$ training (LM)	4.8839	5.3537	11.5364	8.5867	10.456	4.5233
Controller law	2.1776	1.7051	0.99181	0.58594	3.1592	0.55235
RK based model parameter estimator training	–	–	1.4984	–	–	0.99741
Miscellaneous tasks	0.4325	0.16515	0.31956	0.44692	0.47771	0.26218
Total loop time	10.5583	8.6814	16.2589	10.9897	16.1442	8.644

less than 17 ms. In comparison to the sampling times of the systems, the execution time is fairly small and it can be predicated that RK-$NN_{controller}$ can be readily utilized in real time applications of nonlinear MIMO systems. Furthermore, it is possible to enhance the execution time by optimizing the control algorithm and implementing it on effective hardwares such as FPGA. In simulations, a PC with 2.2 GHz core i7 CPU and 8 GB RAM has been deployed to implement the control algorithm and codes are not optimized.

6 Conclusion

In this paper, a novel model predictive nonlinear control architecture is proposed in which Runge–Kutta integration method is utilized both in the controller and system identification blocks. The main novelty of this paper is that the dynamics of the control signals can be identified as mathematical expressions for nonlinear systems and RK-NN is directly deployed as controller structure for nonlinear MIMO systems. In addition, Runge–Kutta Integration method is utilized in both controller and system identification blocks. Since RBF Neural Network is deployed in Runge–Kutta Neural Network, the fast learning and convergence speed of RBF neural network and the accurate integration capability of Runge–Kutta method are combined in a Runge–Kutta RBF neural network controller architecture for nonlinear MIMO systems. The adaptation mechanism is composed of two main blocks based on Runge–Kutta method: RK-$NN_{controller}$ to identify and compute the control signal dynamics and RK_{model} to approximate the K-step ahead future dynamical behaviour of the controlled system. RK-$NN_{controller}$ embodies the powerful characteristics of RBF neural network structure and Runge- Kutta integration method. The network parameters of RK-$NN_{controller}$ are optimized via Levenberg–Marquardt learning rule. RK_{model} is composed of three main subblocks: RK raw system model utilized to derive gradient informations necessary for Jacobian calculation; RK based model parameter estimator deployed for online estimation of time-varying and uncertain parameters of the system and RK based EKF used to approximate the unmeasurable states of the controlled system.

The performance evaluation of the proposed nonlinear controller is performed on nonlinear three tank system and Van de Vusse benchmark system. The robustnesses of the controller have also been scrutinised for the nominal, measurement noise and parametric uncertainty cases. The obtained results demonstrate that the proposed RK-$NN_{controller}$ for nonlinear MIMO systems introduces prosperous tracking performance as well as good noise

rejection and high toleration to parametric uncertainties. In future works, it is aimed to deploy Runge–Kutta numerical integration method to enhance new Runge–Kutta type adaptive controller architectures and system identification techniques for nonlinear MIMO systems. It is also possible to extend the introduced adaptive architecture for finite-time control, finite time stabilization of switched systems, synchronization of complex networks and switching systems problems.

Compliance with Ethical Standards

Conflict of interest The author declares that there is no conflict of interests regarding the publication of this paper.

References

1. Lakshmanan M, Rajasekar S (2003) Nonlinear dynamics: integrability, chaos and patterns. Advanced Texts in physics. Springer, Berlin
2. Guo HJ, Lin SF, Liu JH (2006) A radial basis function sliding mode controller for chaotic Lorenz system. Phys Lett A 351(4–5):257–261. https://doi.org/10.1016/j.physleta.2005.10.101
3. Tan YH, Dekeyser R (1994) Adaptive PID control with neural network based predictor. In: International conference on control 94. England
4. Zhang MG, Li WH, Liu MQ (2005) Adaptive PID control strategy based on RBF neural network identification. In: International conference on neural networks and brain (ICNN&B 2005). Beijing
5. Iplikci S (2010) A comparative study on a novel model-based PID tuning and control mechanism for nonlinear systems. Int J Robust Nonlinear Control 20(13):1483–1501. https://doi.org/10.1002/rnc.1524
6. Akhyar S, Omatu S (1993) Self-tuning PID control by neural networks. In: International joint conference on neural network (IJCNN'93). Nagoya
7. Jin W, Gao WZ, Gu SS, Wang FL (1997) PID-Like controller using a modified neural network. Int J Syst Sci 28(8):809–815. https://doi.org/10.1080/00207729708929441
8. Fang MC, Zhuo YZ, Lee ZY (2010) The application of the self-tuning neural network PID controller on the ship roll reduction in random waves. Ocean Eng 37(7):529–538. https://doi.org/10.1016/j.oceaneng.2010.02.013
9. Kang J, Meng WJ, Abraham A, Liu HB (2014) An adaptive PID neural network for complex nonlinear system control. Neurocomputing 135:79–85. https://doi.org/10.1016/j.neucom.2013.03.065
10. Milovanovic MB, Antic DS, Milojkovic MT, Nikolic SS, Peric SL, Spasic MD (2016) Adaptive PID control based on orthogonal endocrine neural networks. Neural Netw 84:80–90. https://doi.org/10.1016/j.neunet.2016.08.012
11. Cong S, Liang Y (2009) PID-Like neural network nonlinear adaptive control for uncertain multivariable motion control systems. IEEE Trans Ind Electron 56(10):3872–3879. https://doi.org/10.1109/TIE.2009.2018433
12. Jing XJ, Cheng L (2013) An optimal PID control algorithm for training feedforward neural networks. IEEE Trans Ind Electron 60(6):2273–2283. https://doi.org/10.1109/TIE.2012.2194973
13. Decanete JF, Ollero A, Diazfondon M (1991) Autonomous controller tuning by using a neural network. In: International workshop on artificial neural networks (IWANN 91). Granada
14. Wang GJ, Fong CT, Chang KJ (2001) Neural-network-based self-tuning PI controller for precise motion control of PMAC motors. IEEE Trans Ind Electron 48(2):408–415
15. Nordgren RE, Meckl PH (1991) A comparison of a neural network and a model reference adaptive controller. In: IEEE 1991 international conference on systems engineering. Fairborn
16. Yamada T, Yabuta T (1990) An extension of neural network direct controller. In: International workshop on intelligent robots and systems, towards a new frontier of applications
17. Lewis FL, Yesildirek A, Liu K (1993) Neural net robot controller: structure and stability proofs. In: IEEE Mediterranean symposium on new directions in control theory and applications. Khania
18. Ji XD, Familoni BO (1999) A diagonal recurrent neural network-based hybrid direct adaptive SPSA control system. IEEE Trans Autom Control 44(7):1469–1473. https://doi.org/10.1109/9.774125
19. Wu QH, Hogg BW, Irwin GW (1992) A neural network regulator for turbogenerators. IEEE Trans Neural Netw 3(1):95–100. https://doi.org/10.1109/72.105421

20. Khalid M, Omatu S, Yusof R (1992) Self learning process control systems by neural networks. In: 31st IEEE conference on decision and control. Tucson
21. Liu YJ, Tang L, Tong SC, Chen CLP (2015) Adaptive NN controller design for a class of nonlinear MIMO discrete-time systems. IEEE Trans Neural Netw Learn Syst 26(5):1007–1018. https://doi.org/10.1109/TNNLS.2014.2330336
22. Tanomaru J, Omatu S (1992) Process control by on-line trained neural controllers. IEEE Trans Ind Electron 39(6):511–521. https://doi.org/10.1109/41.170970
23. Cui XZ, Shin KG (1993) Direct control and coordination using neural networks. IEEE Trans Syst Man Cybern 23(3):686–697. https://doi.org/10.1109/21.256542
24. Saerens M, Soquet A (1989) A neural controller. In: International conference on artificial neural networks
25. Saerens M, Soquet A (1991) Neural controller based on back-propagation algorithm. IEE Proc F Radar Signal Process 138(1):55–62. https://doi.org/10.1049/ip-f-2.1991.0009
26. Tai HM, Wang JL, Ashenayi K (1992) A neural network-based tracking control system. IEEE Trans Ind Electron 39(6):504–510. https://doi.org/10.1109/41.170969
27. Lightbody G, Irwin GW (1995) Direct neural model reference adaptive control. IEE Proc Control Theory Appl 142(1):31–43. https://doi.org/10.1049/ip-cta:19951613
28. Porter WA, Liu W (1995) Neural controllers for systems with unknown dynamics. IEEE Trans Aerosp Electron Syst 31(4):1331–1340. https://doi.org/10.1109/7.464354
29. Zhang Y, Sen P, Hearn GE (1995) An on-line trained adaptive neural controller. IEEE Control Syst Mag 15(5):67–75. https://doi.org/10.1109/37.466260
30. Yuan M, Poo AN, Hong GS (1995) Direct neural control system: nonlinear extension of adaptive control. IEE Proc Control Theory Appl 142(6):661–667. https://doi.org/10.1049/ip-cta:19952122
31. Psaltis D, Sideris A, Yamamura AA (1988) A multilayered neural network controller. IEEE Control Syst Mag 8(2):17–21. https://doi.org/10.1109/37.1868
32. Wang DL A (2008) Model reference based neural network adaptive controller. In: International symposium on knowledge acquisition and modeling. Wuhan
33. Spall JC, Cristion JA (1992) Direct adaptive control of nonlinear systems using neural networks and stochastic approximation. In: 31st IEEE conference on decision and control. Tucson
34. Uçak K, Günel GÖ (2017) Generalized self-tuning regulator based on online support vector regression. Neural Comput Appl 28:S775–S801. https://doi.org/10.1007/s00521-016-2387-4
35. Pham DT, Karaboga D (1999) Self-tuning fuzzy controller design using genetic optimisation and neural network modelling. Artif Intell Eng 13(2):119–130. https://doi.org/10.1016/S0954-1810(98)00017-X
36. Sharkawy AB (2010) Genetic fuzzy self-tuning PID controllers for antilock braking systems. Eng Appl Artif Intell 23(7):1041–1052. https://doi.org/10.1016/j.engappai.2010.06.011
37. Bouallegue S, Haggege J, Ayadi M, Benrejeb M (2012) PID-type fuzzy logic controller tuning based on particle swarm optimization. Eng Appl Artif Intell 25(3):484–493. https://doi.org/10.1016/j.engappai.2011.09.018
38. Bishr M, Yang YG, Lee G (2000) Self-tuning PID control using an adaptive network-based fuzzy inference system. Intell Autom Soft Comput 6(4):271–280. https://doi.org/10.1080/10798587.2000.10642795
39. Iplikci S (2013) Runge–Kutta model-based adaptive predictive control mechanism for non-linear processes. Trans Inst Meas Control 35(2):166–180. https://doi.org/10.1177/0142331212438910
40. Uçak K (2018) A Runge–Kutta neural network-based control method for nonlinear MIMO systems. Soft Comput. https://doi.org/10.1007/s00500-018-3405-5
41. Beyhan S (2013) Runge–Kutta model-based nonlinear observer for synchronization and control of chaotic systems. ISA Trans 52(4):501–509. https://doi.org/10.1016/j.isatra.2013.04.005
42. Cetin M, Iplikci S (2015) A novel auto-tuning PID control mechanism for nonlinear systems. ISA Trans 58:292–308. https://doi.org/10.1016/j.isatra.2015.05.017
43. Wang YJ, Lin CT (1998) Runge–Kutta neural network for identification of dynamical systems in high accuracy. IEEE Trans Neural Netw 9(2):294–307. https://doi.org/10.1109/72.661124
44. Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. Neural Comput 3(2):246–257. https://doi.org/10.1162/neco.1991.3.2.246
45. Girosi F, Poggio T (1990) Networks and the best approximation property. Biol Cybern 63(3):169–176. https://doi.org/10.1007/BF00195855
46. Jayawardena AW, Fernando DAK, Zhou MC (1997) Comparison of multilayer perceptron and radial basis function networks as tools for flood forecasting. In: International conference on destructive water: water-caused natural disasters, their abatement and control. Anaheim
47. Lee CC, Chung PC, Tsai JR, Chang CI (1999) Robust radial basis function neural networks. IEEE Trans Syst Man Cybern B Cybern 29(6):674–685. https://doi.org/10.1109/3477.809023
48. Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1(2):281–294. https://doi.org/10.1162/neco.1989.1.2.281

49. Lee S, Kil RM (1991) A Gaussian potential function network with hierarchically self-organizing learning. Neural Netw 4(2):207–224. https://doi.org/10.1016/0893-6080(91)90005-P
50. Aström KJ, Borisson U, Ljung L, Wittenmark B (1977) Theory and applications of self-tuning regulators. Automatica 13(5):457–476
51. Efe MO, Kaynak O (2000) A comparative study of soft-computing methodologies in identification of robotic manipulators. Robot Auton Syst 30(3):221–230
52. Hagan MT, Demuth HB, De Jesus O (2002) An introduction to the use of neural networks in control systems. Int J Robust Nonlinear Control 12(11):959–985. https://doi.org/10.1002/rnc.727
53. Efe MO, Kaynak O (1999) A comparative study of neural network structures in identification of nonlinear systems. Mechatronics 9(3):287–300. https://doi.org/10.1016/S0957-4158(98)00047-6
54. Efe MO (2011) Neural-Network-Based Control. In: Wilamowski BM, Irwin JD (eds) The industrial electronics handbook: intelligent systems. CRC Press, USA
55. Denai MA, Palis F, Zeghbib A (2004) ANFIS based modelling and control of non-linear systems: a tutorial. In: IEEE international conference on systems, man and cybernetics
56. Jang JR (1993) ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685. https://doi.org/10.1109/21.256541
57. Iplikci S (2010) A support vector machine based control application to the experimental three-tank system. ISA Trans 49(3):376–386. https://doi.org/10.1016/j.isatra.2010.03.013
58. Iplikci S (2006) Online trained support vector machines-based generalized predictive control of non-linear systems. Int J Adapt Control Signal Process 20(10):599–621. https://doi.org/10.1002/acs.919
59. Thrun S, Burgard W, Fox D (2005) Probabilistic robotics. MIT Press, Cambridge, MA
60. Er MJ, Wu SQ, Lu JW, Toh HL (2002) Face recognition with radial basis function (RBF) neural networks. IEEE Trans Neural Netw 13(3):697–710
61. Hadi M, Balasundram SK (2012) Comparison between multi-layer perceptron and radial basis function networks for sediment load estimation in a tropical watershed. J Water Resour Prot 4(10):870–876. https://doi.org/10.4236/jwarp.2012.410102
62. Flynn D, McLoone S, Irwin GW, Brown MD, Swidenbank E, Hogg BW (1997) Neural control of turbo-generator systems. Automatica 33(11):1961–1973. https://doi.org/10.1016/S0005-1098(97)00142-8
63. Amira GmbH (2000) DTS200—laboratory setup three-tank system
64. Theilliol D, Noura H, Ponsart JC (2002) Fault diagnosis and accommodation of a three-tank system based on analytical redundancy. ISA Trans 41(3):365–382. https://doi.org/10.1016/S0019-0578(07)60094-9
65. Chen H, Kremling H, Allgöwer F (1995) Nonlinear predictive control of a benchmark CSTR. In: 3rd European control conference
66. Engell S, Klatt KU (1993) Nonlinear control of a non-minimum-phase CSTR. In: American control conference. San Francisco
67. Vojtesek J, Dostál P (2010) Adaptive control of chemical reactor. In: International conference cybernetics and informatics
68. Jørgensen JB (2007) A critical discussion of the continuous-discrete extended Kalman filter. In: European congress of chemical engineering 6
69. Kulikov GY, Kulikova MV (2014) Accurate state estimation in the Van der Vusse reaction. In: IEEE conference on control applications (CCA). Nice
70. Niemiec MP, Kravaris C (2003) Nonlinear model-state feedback control for nonminimum-phase processes. Automatica 39(7):1295–1302. https://doi.org/10.1016/S0005-1098(03)00103-1
71. Kravaris C, Niemiec M, Berber R, Brosilow CB (1998) Nonlinear model-based control of nonminimum-phase processes. In: Kravaris C, Berber R (eds) Nonlinear model based process control. Springer, Dordrecht, pp 115–142