Neural Network Assisted Computationally Simple PI$^\lambda$D$^\mu$ Control of a Quadrotor UAV

Mehmet Önder Efe, *Senior Member, IEEE*

***Abstract*—The applications of Unmanned Aerial Vehicles (UAVs) require robust control schemes that can alleviate disturbances such as model mismatch, wind disturbances, measurement noise, and the effects of changing electrical variables, e.g., the loss in the battery voltage. Proportional Integral and Derivative (PID) type controller with noninteger order derivative and integration is proposed as a remedy. This paper demonstrates that a neural network can be trained to provide the coefficients of a Finite Impulse Response (FIR) type approximator, that approximates to the response of a given analog PI$^\lambda$D$^\mu$ controller having time varying action coefficients and differintegration orders. The results obtained show that the neural network aided FIR type controller is very successful in driving the vehicle to prescribed trajectories accurately. The response of the proposed scheme is highly similar to the response of the target PI$^\lambda$D$^\mu$ controller and the computational burden of the proposed scheme is very low.**

***Index Terms*—Fractional order control, unmanned aerial vehicles, neural networks.**

I. Introduction

A RECENT trend in control of industrial systems is to describe the networking, layers of the control hierarchy, and the communication capabilities of a complicated system by exploiting versatile software tools, intelligent sensors and methods devised specifically for fault tolerance and handling of safety critical issues and components, [1]–[3]. The lowest level of this structure is established typically by Proportional Integral and Derivative (PID) type controllers, a novel form of which is discussed in this paper.

PID type controllers have been standard tools in industry due to their practicality. The availability of well established rules for tuning the parameters of the controller is another reason making them preferable in real-time applications. Fractional order PI$^\lambda$D$^\mu$ controller is a generalization of the integer order PID controller exploiting the richness offered by the noninteger orders of the Laplace variable s. Denoting $C(s)$ as the transfer function associated to it, an analog PI$^\lambda$D$^\mu$ controller is described as given below

$$C(s) := \frac{D(s)}{U(s)} = k_p + \frac{k_i}{s^\lambda} + k_d s^\mu. \tag{1}$$

Manuscript received January 21, 2011; revised January 25, 2011; accepted January 26, 2011. Date of publication April 05, 2011; date of current version May 06, 2011. This work was supported in part by part by Turkish Scientific Council (TÜBİTAK) under Contract 107E137. Paper no. TII-11-01-0031.

The author is with the Department of Electrical and Electronics Engineering, Bahçeşehir University, Beşiktaş, İstanbul, Turkey (e-mail: onderefe@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2011.2123906

The history of fractional calculus dates back to the letter from L'Hôpital to Leibniz in 1695 asking the meaning of a derivative of order $(1/2)$. It was not soon after this but the during the last few decades, the fractional order operators have been made possible to use in automatic control systems. Considering the operator $\mathbf{D} := (\mathrm{d}/\mathrm{d}t)$, one could define the operators $\mathbf{D}^\alpha$ with a noninteger $\alpha \in \Re$. Further, for $\alpha > 0$, one obtains differentiators, while $\alpha < 0$ yields integrators. In the literature, these operators are called differintegration operators and two widely used definitions are by Riemann–Liouville in (2) and by Caputo in (3), where the lower limit of time is zero in both, [4]–[6]

$$\mathbf{D}^\alpha y = y^{(\alpha)} := \frac{1}{\Gamma(r-\alpha)}\left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^r \int_0^t \frac{y(\xi)}{(t-\xi)^{\alpha+1-r}}\mathrm{d}\xi \tag{2}$$

$$\mathbf{D}^\alpha y = y^{(\alpha)} := \frac{1}{\Gamma(r-\alpha)} \int_0^t \frac{y^{(r)}(\xi)}{(t-\xi)^{\alpha+1-r}}\mathrm{d}\xi \tag{3}$$

where $r-1 \le \alpha < r$ and r is an integer. Defining $\mathcal{L}$ as the Laplace transform and s as the Laplace variable, it is possible to write $\mathcal{L}(\mathbf{D}^\beta) = s^\beta$ and such an approach makes it possible to write transfer functions in fractional orders, define the state space systems in fractional order and determine their solutions under various types of operating conditions, [6]–[9]. Clearly, in a significant volume of research outcomes reported so far, the linearity is an underlying assumption in the analysis and design of fractional order control systems. This paper focuses on the approximate realization of analog PI$^\lambda$D$^\mu$ controller via neural networks and an unmanned aerial vehicle platform is utilized to justify the claims.

In [4]–[6] and [8], fundamental issues regarding the fractional calculus, fractional differential equations and a viewpoint from the systems and control engineering are elaborated and several exemplar cases are taken into consideration. One such application area focuses on PID control with derivative and integral actions having fractional orders, i.e., PI$^\lambda$D$^\mu$ control is implemented. In the literature, several applications of PI$^\lambda$D$^\mu$ controllers have been reported. The early notion of the scheme is reported by Podlubny, [10]. In [11] and [12], tuning of the controller parameters is considered when the plant under control is a fractional order one. Ziegler–Nichols type tuning rules are derived in [13] and rules for industrial applications are postulated in [14]. The application of fractional order PID controllers in chemical reaction systems is reported in [15], the issues regarding the frequency domain are considered in [16]. Clearly, the cited volume of works demonstrates that the interest to PID control is growing also in the direction of fractional order versions. Unsurprisingly the reason for this is the widespread use of the variants of PID controller and the confidence of the engineers in industry.

Having this picture in front, the realization of fractional order PI$^{\lambda}$D$^{\mu}$ controllers have become a critical issue. In essence, the problem is to realize the operator $\mathbf{D}^{\beta}$ either in continuous time or in discrete time. Several numerical techniques have been proposed as a remedy to this problem. In [17], a thorough investigation of the rational approximations of fractional order operators is presented, discrete and continuous time cases are addressed, and a comparison in frequency domain is given. In [18] and [19], Crone approximation is focussed on. The method schedules a set of poles and zeros to approximate a desired operator in frequency domain. For a better match, one needs to consider large number as the approximation order yet this causes a significant increase in the computational complexity. Further, increasing the approximation order may practically result in a transfer function that is difficult to realize due to the numerical problems. In [20], analogue realizations based on continued fraction expansions are elaborated and relations to nested multiple loop control systems are established. In this paper, we propose artificial neural networks for emulating the analog PI$^{\lambda}$D$^{\mu}$ controller as a whole for a given set of action coefficients (k_p, k_i, and k_d) and differintegration orders (λ and μ). In [21], a special case of what is presented here is elaborated. A particular neural network structure is proposed to imitate the classical PID controller. The motivation here is to obtain a universal PI$^{\lambda}$D$^{\mu}$ controller module approximating to the response of a target module at a particular sampling period, which is $T = 1$ ms in this paper. As the test bed, we choose a quadrotor type unmanned vehicle platform and a Finite Impulse Response (FIR) type filter to approximate the response of a given $C(s)$ under 1 ms of sampling. To calculate the necessary FIR filter coefficients, we adopt a neural network-based solution that is described next.

This paper is organized as follows. Section II describes used neural network structure and the necessary details in generating the training data. Section III presents the dynamic model of the quadrotor and describes the control problem. The Cartesian controller and the results obtained through a set of simulations are presented in Section IV and concluding remarks as well as a summary of the contributions are given at the end of this paper.

II. Neural Network Assisted Approximation to PI$^{\lambda}$D$^{\mu}$ Controller

Consider the feedforward neural network structure shown in Fig. 1. The structure is called feedforward as the flow of information has a one-way nature. Let an input vector and output vector at time t_0 be defined as $I_{t_0} := (u_1(t_0)\, u_2(t_0) \ldots u_m(t_0))$ and $O_{t_0} := (y_1(t_0)\, y_2(t_0) \ldots y_n(t_0))$, respectively. An input output pair, shortly a *pair* or sample, is defined as $S_{t_0} := \{I_{t_0}, O_{t_0}\}$. Consider there are P pairs in a given data set, which we call training set. Based on this, the response of

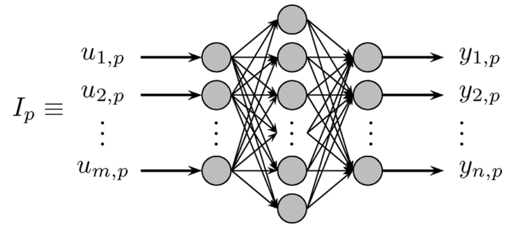

Fig. 1. A feedforward neural network structure with single hidden layer.

a neural network to a set of input vectors can be evaluated and the total cost over the given set can be defined as follows:

$$J(w_t) = \frac{1}{2P}\sum_{p=1}^{P}\sum_{i=1}^{n}(d_{i,p} - y_{i,p}(I_p, w_t))^2 \tag{4}$$

where $d_{i,p}$ is the target output corresponding to the ith output of the neural network that responds to pth pattern. The cost in (4) is also called the mean squared error (MSE) measure. Similarly, $y_{i,p}$ is the response of the neural network to I_p. In (4), the generic symbol w_t stands for the set of all adjustable parameters, i.e., the synaptic strengths (W), or biases (B) at time t. The input output relation of a $\mathcal{H}$-hidden layer neural network with hyperbolic tangent type neuronal activation scheme can be given as follows:

$$O_h = \tanh(W_h O_{h-1} + B_h), h = 1, 2, \ldots, \mathcal{H} \tag{5}$$

where $O_0 = u$ and $O_{\mathcal{H}} = y$. Assume there are a total of $\mathcal{N}$ adjustable parameters denoted by the vector $w = (\omega_1\, \omega_2 \ldots \omega_{\mathcal{N}})$. Each entry of the parameter vector corresponds to a unique parameter in an ordered fashion. The update law is given in (6), where t stands for the discrete time index. Here, $\nabla_w^2 J(w_t) = 2H(w_t)^{\mathrm{T}} H(w_t) + g(H(w_t))$ with $g(H(w_t))$ being a small residual, and $\nabla_w J(w_t) = 2H(w_t)^{\mathrm{T}} E(w_t)$ with E and H being the error vector as given in (7) and Jacobian, respectively. The error vector contains the errors computed via $e_{i,p} := d_{i,p} - y_{i,p}(I_p, w)$ for every training pair, and the Jacobian contains the partial derivatives of each component of E with respect to every parameter in w shown in (6) and (7) at the bottom of the page. Based on these definitions, the update law based on Levenberg–Marquardt optimization technique can be constructed as

$$w_{t+1} = w_t - (\rho I + H(w_t)^{\mathrm{T}} H(w_t))^{-1} H(w_t)^{\mathrm{T}} E(w_t) \tag{8}$$

where $\rho > 0$ is a user-defined scalar design parameter for improving the rank deficiency problem of the matrix $H(w_k)^{\mathrm{T}} H(w_k)$ and I is an identity matrix of dimensions

$$w_{t+1} = w_t - \left(\rho I + \nabla_w^2 J(w_t)\right)^{-1} \nabla_w J(w_t) \tag{6}$$

$$E = (e_{1,1} \quad \cdots \quad e_{n,1} \quad e_{1,2} \quad \cdots \quad e_{n,2} \quad \cdots \quad e_{1,P} \quad \cdots \quad e_{n,P})^{\mathrm{T}} \tag{7}$$

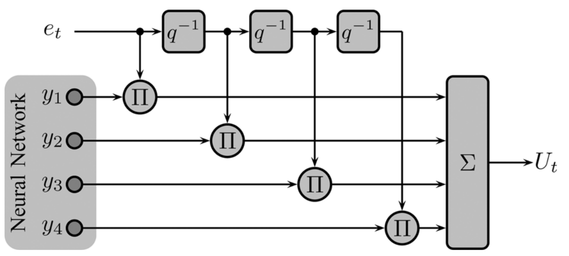

Fig. 2. Neural network assisted FIR approximator to generalize PI$^\lambda$D$^\mu$ controller. Here, q^{-1} stands for the delay operator in time.

$\mathcal{N} \times \mathcal{N}$. It is important to note that for small ρ, (8) approximates to the Gauss–Newton method, and for large ρ, the tuning law becomes the standard error backpropagation algorithm with a step size $(1/\rho)$. Therefore, Levenberg–Marquardt method establishes a good balance between error backpropagation and Gauss–Newton strategies and inherits the prominent features of both algorithms in eliminating the rank deficiency problem with improved convergence. An elegant comparison is presented in [23], where the concept of optimality, different network configurations, learning algorithms, connectivity issues, and size issues are discussed in detail.

Since the input and the output of a PI$^\lambda$D$^\mu$ controller is measurable, the setting described above fits in the problem we would like to solve perfectly. Two different viewpoints can be followed in developing an emulator for a given PI$^\lambda$D$^\mu$ controller. The first one is the traditional scheme, i.e., for a preselected delay depth in input and a preselected delay depth in output, an input vector is formed and the neural network aims at predicting the next output, [22]. For the problems like the one considered here, the prediction performance of the neural network heavily depends upon the delay depths, i.e., the larger the depths the better the result. In addition to this, feeding back the output of such a neural network is highly vulnerable to the accumulating output errors leading eventually instabilities. Therefore, this scheme is not suitable for the PI$^\lambda$D$^\mu$ controller emulation. The second alternative is to use the neural network in between the two spaces, namely the space of action coefficients denoted by $\{k_p, k_d, k_i, \lambda, \mu\} \in \mathcal{A}_i$ and the space of emulator model parameters denoted by $\{a_1, a_2, \ldots, a_l\} \in \mathcal{A}_o$. The second method is useful as it is based on a mapping in between two feature spaces linked to each other via the chosen neural network.

As described by (9), we set a FIR type approximator model having the parameter set $\mathcal{A}_o = \{a_0, a_1, a_2, a_3\}$ and collect the training data from the specified analog PI$^\lambda$D$^\mu$ controller. The input to the neural model is from $\mathcal{A}_i$ and the output is the set of coefficients required to operate the FIR structure in (9). In order to ensure the descriptiveness of the samples in $\mathcal{A}_i$ and $\mathcal{A}_o$, a number of trials have to be done and the mapping results have to be recorded to build the training data set. For each case, the identification toolbox of Matlab is utilized in determining the parameters of the chosen FIR-based model. A block structure of the emulator based on the below model is depicted in Fig. 2, where the role of neural network is seen to provide the required set of coefficients in $\mathcal{A}_o$

$$U_t = a_0 e_t + a_1 e_{t-1} + a_2 e_{t-2} + a_3 e_{t-3}. \qquad (9)$$

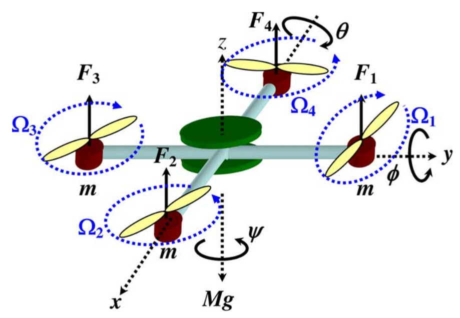

Fig. 3. Schematic view and variable definitions of a quadrotor type UAV.

TABLE I
PHYSICAL PARAMETERS OF THE QUADROTOR UAV

L	Half distance between two motors	0.3 m.
M	Mass of the vehicle	0.8 kg.
g	Gravitational acceleration constant	9.81 m/s^2
I_{xx}	Moment of inertia around $x-$axis	$15.67e-3$
I_{yy}	Moment of inertia around $y-$axis	$15.67e-3$
I_{zz}	Moment of inertia around $z-$axis	$28.346e-3$
b	Thrust coefficient	$192.3208e-7$ Ns^2
d	Drag coefficient	$4.003e-7$ Nms^2
j_r	Propeller inertia coefficient	$6.01e-5$

Clearly, when the structure of the approximator is fixed, such a scheme indicates that a PI$^\lambda$D$^\mu$ controller can be generalized at a certain level of accuracy and we choose an approximator having the structure given by (9), which observes the error and its past three values. We claim that the method proposed here is practically valuable as its computational burden is very low as a delay depth of 3 is highly promising to obtain good results on a test setup described next.

III. Quadrotor Dynamics

The vehicle considered in this study is illustrated in Fig. 3 and the physical parameters are listed in Table I. The dynamical equations describing the quadrotor rotorcraft are given in (10)–(15), where the first three of these equations describe the dynamics in the Cartesian space, whereas the last three express the dynamics in the Euler angles, i.e., the attitude. In (16)–(19), the other variables and inputs seen in the UAV dynamics are given

$$\ddot{x} = (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi)\frac{1}{M}U_1 \qquad (10)$$

$$\ddot{y} = (\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi)\frac{1}{M}U_1 \qquad (11)$$

$$\ddot{z} = -g + \cos\phi \cos\theta \frac{1}{M}U_1 \qquad (12)$$

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\left(\frac{I_{yy} - I_{zz}}{I_{xx}}\right) + \frac{j_r}{I_{xx}}\dot{\theta}\Upsilon + \frac{L}{I_{xx}}U_2 \qquad (13)$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\left(\frac{I_{zz} - I_{xx}}{I_{yy}}\right) - \frac{j_r}{I_{yy}}\dot{\phi}\Upsilon + \frac{L}{I_{yy}}U_3 \qquad (14)$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}\left(\frac{I_{xx} - I_{yy}}{I_{zz}}\right) + \frac{1}{I_{zz}}U_4 \qquad (15)$$

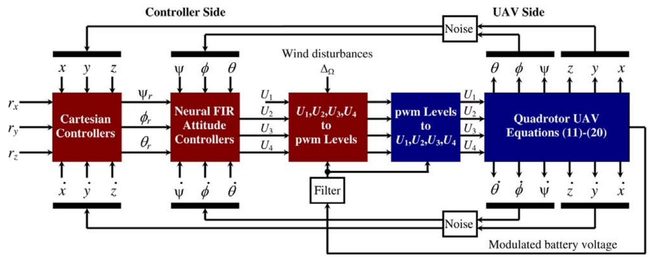
Controller Side
UAV Side
Noise
Wind disturbances
Δ_Ω
Cartesian Controllers
Neural FIR Attitude Controllers
U_1,U_2,U_3,U_4 to pwm Levels
pwm Levels to U_1,U_2,U_3,U_4
Quadrotor UAV Equations (11)-(20)
Filter
Noise
Modulated battery voltage

Fig. 4. The structure of the control system.

where $\Upsilon = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4$ and Ω_i stand for the angular speed of the ith propeller

$$U_1 = b\Omega_1^2 + b\Omega_2^2 + b\Omega_3^2 + b\Omega_4^2 = \sum_{i=1}^{4} F_i \tag{16}$$

$$U_2 = b\Omega_4^2 - b\Omega_2^2 = F_4 - F_2 \tag{17}$$

$$U_3 = b\Omega_3^2 - b\Omega_1^2 = F_3 - F_1 \tag{18}$$

$$U_4 = d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \tag{19}$$

The control problem here is to drive the UAV toward a predefined trajectory in the 3D space by generating an appropriate sequence of Euler angles, which need to be controlled as well. The latter is called attitude control and the command signals to which are produced by the Cartesian controller.

As seen from the definitions of U_is in (17)–(18), and the variables in Fig. 3, the dynamics involving Euler angles have a differential nature. If the angular speed of a propeller is smaller than the one along the same axis, than a rotation occurs in the direction from fast one to slow one. With such a mechanism, it becomes possible to realize any motion by appropriately scheduling the angular speeds of the propellers. On the other hand, keeping the differences of the propellers lying on the same axis constant, increasing or decreasing the angular speeds result in the same motion taking place at different altitudes. According to this discussion, one sees that the motion in the Cartesian space is obtained by suitably driving the Euler angles to their desired values. The control scheme has therefore two stages. The outer control loop computes the desired Euler angles, and the inner control loop (the attitude controller) computes U_is to observe the desired motion.

In spite of the physics-based dynamical representation above, when the problem is considered at the practical level, it is seen that the physical connection in between the UAV and the controllers is maintained at the pwm level, a certain interval of which is only appropriate to drive the electronic speed controllers and we utilize a neural network-based nonlinear module to convert the controller output to pwm signal and install another module to the UAV side to convert a pwm signal to U_is appropriately, [24]. The diagram of the control system with these components is illustrated in Fig. 4. Though we omit the details regarding the handshaking, it is evident that it introduces further nonlinearity to the dynamical representation and contributes to the complexity of the control problem, yet its simulations become convincingly realistic. In the next section, the details regarding the flight as well as the associated difficulties are dwelt on in detail with reference to the diagram seen in Fig. 4.

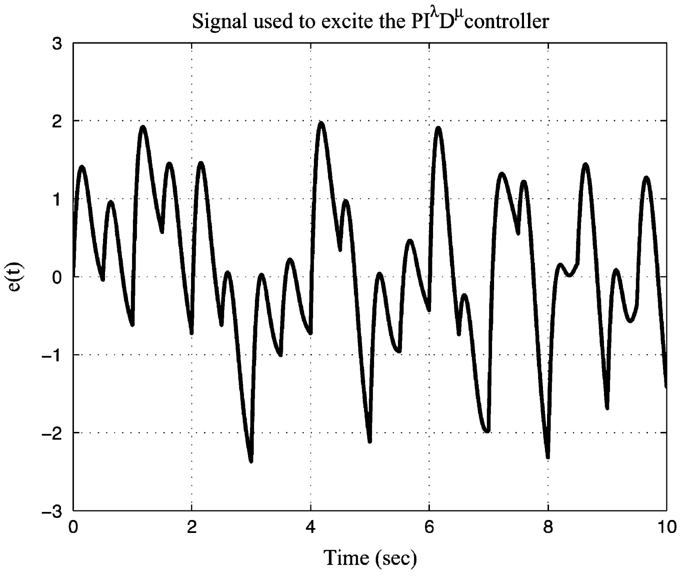

Fig. 5. The excitation signal used to generate necessary data sets.

IV. Simulation Results

As an illustrative example, we consider how a neural network-based approximator could imitate an analog PI$^\lambda$D$^\mu$ controller. For this purpose, we will collect some data to train a simpler model and test both controllers for the same task. We consider the performance and computational complexity issues to validate our claims.

The first stage of the neural network assisted emulation of PI$^\lambda$D$^\mu$ control scheme is to train a neural structure. In this study, we choose a neural network that has a single hidden layer containing 20 neurons in each with hyperbolic tangent type neuronal activation function. The tuning algorithm summarized in the second section is utilized to train the neural network and a total of $P = 3000$ pairs have been used during the training, 200 for checking routine that stops the training and 100 for the test results discussed with the forthcoming figures. During the training, we chose $k_p \in (0, 3), k_d \in (0, (1/2))$, $k_i \in (0, 1)$, $\lambda \in (0, 1)$, and $\mu \in (0, 1)$. The neural network aided FIR type approximator described by (9) is utilized.

After many trials to find out both simple and good performing network structure, during the training, we set $H = 20$ and this has led to the adjustment of a total of 204 parameters by the training scheme. The epoch error has a decreasing nature and we used the excitation signal shown in Fig. 5. The training takes 1983 epochs and it is stopped by the checking routine detecting an increase in J for five consecutive epochs. The final value for the cost in (4) is obtained as $9.2842e - 9$, which is similar to values obtained for slightly more complicated neural configurations and is acceptable for our goals. To be more explicit, the results for the testing data set are shown in Fig. 6, where it is seen that the two curves standing for each target and predicted FIR coefficient are almost indistinguishable. A better indication of the performance is to check the relative error. For the ith output of the neural network structure, we define the relative error as given in (20). The presented results confirm that the selected training data is learned well and it represents the target process accurately as e_{rel} is less than 3% for the entire data set (see

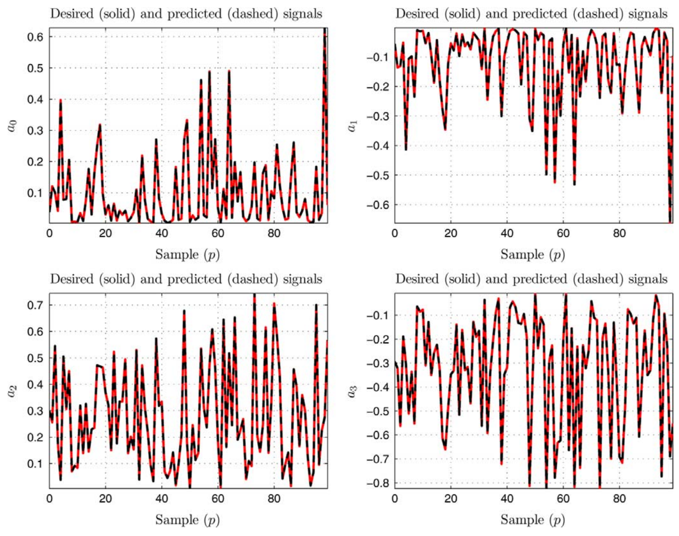

Fig. 6. The prediction performance for all four outputs.

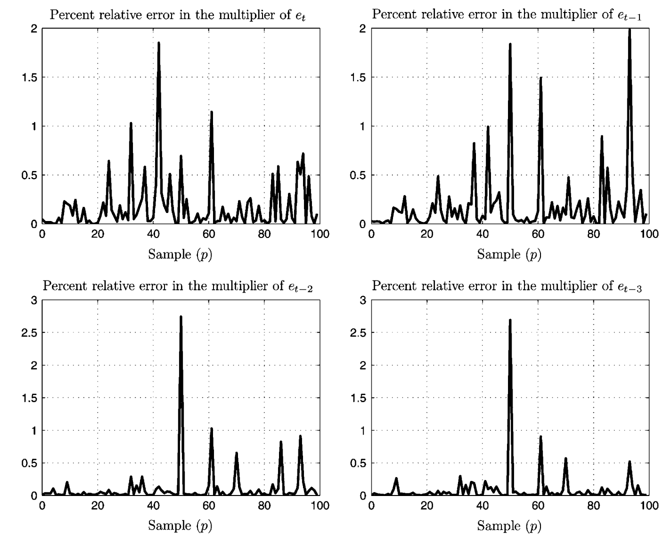

Fig. 7. The prediction performance in terms of the relative error in (20).

Fig. 7). Expectedly, the performance is peculiar to the selected operating conditions such as the exciting signal shown in Fig. 5. Although a healthier method might be to excite the model by white noise, our goal is not to obtain a match for higher frequencies that are not realizable by the propulsion-based actuation considered here. The chosen signal in Fig. 5 fairly represents the likely signals in a flight course of a quadrotor type UAV. We choose the shown excitation sequence to train the neural network and proceed to the validation tests

$$e_{\text{rel},i,p} := 10! \times \frac{|d_{i,p}-y_{i,p}|}{|d_{i,p}|}\%, \; i=1,2,3,4, p=1,2,\ldots,100 \tag{20}$$

In the flight simulations, we consider few different parameter configurations to demonstrate how useful the proposed approach is. The proposed scheme is utilized in the attitude control loop (inner loop) and we use classical controllers in the outer loop. The overall representation of the control system is depicted in Fig. 4, where it is seen that the observed states are corrupted by noise and the control system has a nested structure. Denote the reference Cartesian positions and velocities by r_x, r_y, r_z, and $\dot{r}_x$, $\dot{r}_y$, $\dot{r}_z$. We define $P_z := -4\dot{z} - 4(z - r_z)$ and choose the altitude controller as given in (21)

TABLE II
SIMULATION SETTINGS

Δt	Simulation stepsize	1 ms
H	#of neurons in the hidden layer	20
T	Final time of simulation	130 s.
$(\phi(0), \theta(0), \psi(0))$	Initial attitude of the vehicle	$\left(\frac{\pi}{5}, -\frac{\pi}{6}, -\frac{\pi}{4}\right)$
$(x(0), y(0), z(0))$	Initial position of the vehicle	$(0, 0, 0)$

$$U_1 = M\frac{P_z + g}{\cos\theta\cos\phi}. \tag{21}$$

Substituting (21) into (10) and (11), and adopting the small angle approximation would let us obtain the dynamics below

$$\ddot{x} \approx (P_z + g)\tan\theta \tag{22}$$
$$\ddot{y} \approx -(P_z + g)\tan\phi. \tag{23}$$

In the above dynamics, $\tan\phi$ and $\tan\theta$ can be regarded as the control inputs for observing the desired motion in Cartesian space. To achieve this, the following choices are done:

$$\phi_r = -\arctan\left(\frac{P_y}{P_z + g}\right) \tag{24}$$
$$\theta_r = \arctan\left(\frac{P_x}{P_z + g}\right) \tag{25}$$

where we set $P_x = -\dot{x} - (x - r_x)$ and $P_y = -\dot{y} - (y - r_y)$ after a number of experiments. Having derived the reference values for the roll (ϕ) and the pitch (θ) angles, we utilize the neural network-based PI$^\lambda$D$^\mu$ controller explained in Section II and illustrated in Fig. 2 to track these Euler angles to obtain the desired motion. The yaw motion is undesired and we set $\psi_r = 0$. The simulations have been carried out with the settings given in Table II, where it is seen that the neural network-based control scheme is expected to alleviate the difficulties caused by large initial errors. Following is an itemized list of issues concerning the presented results.

- The UAV developed in the laboratory is powered by Lithium-Polymer type batteries and it is shown that the outrunner brushless DC motors claim very high currents causing a significant reduction in the battery voltage. This causes an uncertainty to be alleviated by the controller as the same input (pwm) signal causes different lift forces since the battery voltage drops in time [24]. Due to the space limit, the details are not presented here but the change in the battery voltage is simulated as an exponentially converging value starting from 11.1 to 9.9 Volts in 130 s of flight. The difficulty alleviated here is the effect of modulation on the measured battery voltage, which seems to be corrupted heavily and which needs a low pass filtering to guide the propulsion model providing the handshaking at pwm level. As shown in Fig. 4, we use a low-pass filter $\sigma_1(s) = (200)/(s + 200)$ for removing the spurious content from the measured battery voltage. For details regarding the power loss in batteries refer to [24].

- Aside from the initial conditions given in Table II, the remaining set of initial conditions are chosen to be zero, i.e., the vehicle is motionless initially. Since the goal is to demonstrate the performance of the proposed neural PI$^\lambda$D$^\mu$ control scheme, it is adequate to assume nonzero positional initial values for the Euler angles.
- In order to demonstrate the disturbance rejection capability, the angular speeds of the vehicle have been perturbed additively to simulate the effect of weather conditions, such as wind. As described by (26), the perturbations modify the angular speeds according to the altitude. The effect of the wind disturbances is simulated as an additive component that adds up to the angular speeds prescribed by the controller

$$\Delta_{\Omega_i}(t) = \frac{z(t)}{5}\sin\left(\frac{2\pi t}{T_i}\right) \tag{26}$$

where $T_1 = 10$, $T_2 = 12$, $T_3 = 14$, and $T_4 = 16$ s.
- The observations are noisy, the state vector composed of the positions and velocities are corrupted by noise sequences of power $1e-6$, which is considerable to test the performance of the proposed control scheme.
- Finally, the implementation of fractional order differintegration operators within the analog PI$^\lambda$D$^\mu$ controller need to be discussed. During the simulations, the numerical implementation the output of the PI$^\lambda$D$^\mu$ controller is obtained through the use of well known Crone approximation, which prescribes a series of poles and zeros to build a transfer function $k\prod_{i=1}^{N}(1+s/z_i)/(1+s/p_i)$ approximating the desired operator spectrally, [25]. We choose $N = 9$ and a frequency range covering 0.001–1000 rad/s to realize these operators for the comparison studies.

We choose following controller parameters to justify that the PI$^\lambda$D$^\mu$ controller and its neural network-based emulator perform similar in terms of performance. We utilize a second subscript to denote the variables associated with a particular Euler angle. The parameters of the controller have been set after a remarkably rich set of tests requesting a decrease in μ to weaken the differentiating effect and to strengthen the integrative effect by increasing λ as $z(t)$ gets larger. Similarly, the same effects are reflected to the derivative (k_d) and integral coefficients (k_i) as seen below

$$(k_{p,\phi}, k_{p,\theta}, k_{p,\psi}) = \begin{cases} (2, 2, 0.20), & 0 \le z < 10 \\ (2, 2, 0.20), & 10 \le z < 20 \\ (2, 2, 0.20), & z \ge 20 \end{cases} \tag{27}$$

$$(k_{d,\phi}, k_{d,\theta}, k_{d,\psi}) = \begin{cases} (0.20, 0.15, 0.10), & 0 \le z < 10 \\ (0.20, 0.15, 0.10), & 10 \le z < 20 \\ (0.10, 0.10, 0.10), & z \ge 20 \end{cases} \tag{28}$$

$$(k_{i,\phi}, k_{i,\theta}, k_{i,\psi}) = \begin{cases} (0.80, 0.80, 0.02), & 0 \le z < 10 \\ (0.90, 0.90, 0.02), & 10 \le z < 20 \\ (0.95, 0.95, 0.03), & z \ge 20 \end{cases} \tag{29}$$

$$\lambda = \begin{cases} 0.50, & 0 \le z < 10 \\ 0.55, & 10 \le z < 20 \\ 0.60, & z \ge 20 \end{cases}$$

$$\mu = \begin{cases} 0.90, & 0 \le z < 10 \\ 0.85, & 10 \le z < 20\,. \\ 0.80, & z \ge 20 \end{cases} \tag{30}$$

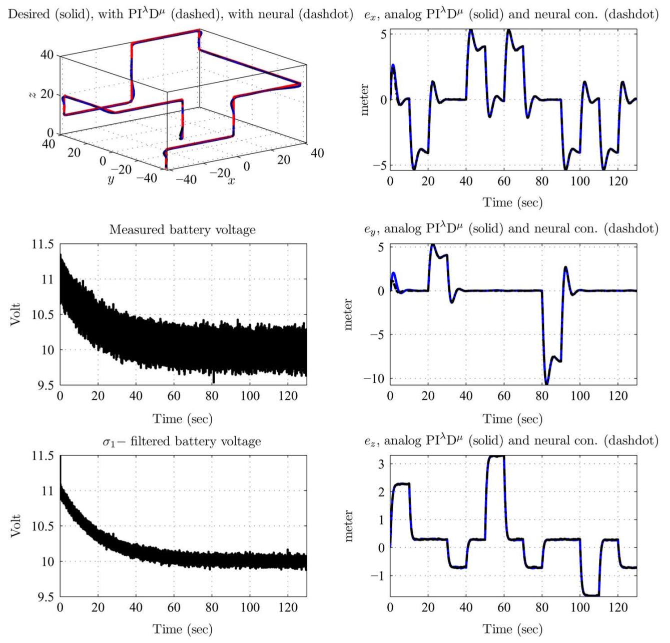

Fig. 8. Change in the battery voltage and results for the Cartesian space variables for analog PI$^\lambda$D$^\mu$ and neural controllers.

As seen clearly from (27)–(29), the parameters associated to the yaw axis (ψ) controller are visibly different from the roll and pitch axes controllers. The control of the yawing motion requires a special care as it is a difficult task to obtain a satisfactorily convergent response, [26]. The altitude has been chosen as the primary variable changing the parameters of the controller to be operated. Naturally, the weather conditions change as the altitude changes and a switching to a new controller structure is needed. We test the analog PI$^\lambda$D$^\mu$ controller and the proposed scheme for the same operating conditions below. Under the aforementioned conditions, the results shown in Fig. 8 are obtained. The top left subplot illustrates three trajectories in the Cartesian space, namely, the desired path, the path followed by the use of the analog PI$^\lambda$D$^\mu$ controller described by (1), and the path followed by the use of the proposed technique. The simulated value of the battery voltage is shown in the middle left subplot and its filtered value is shown on the bottom left subplot. The change in the battery voltage is compensated appropriately and the error trends shown in the right column are obtained. The time evolution of the errors for both the analog PI$^\lambda$D$^\mu$ controller and the proposed controller, and it is seen that the errors caused by the use of both controllers are almost indistinguishable in all every axis of the Cartesian space. This is one important result as the proposed technique simplifies the realization of the analog PI$^\lambda$D$^\mu$ controller via coefficient prediction. The results in the attitude are shown in Fig. 9, where the time evolution of the Euler angles and the tracking errors are depicted. The results on emphasize that the desired angles prescribed by (24)–(25) are followed precisely and the yaw angle is stabilized very quickly. The time evolution of the control signals are compared in Fig. 10, where the essential motion taking place during the early stage of the simulation is captured accurately and the evolution as time progresses is mainly dominated by the quickly changing activity. The two control signals resemble each other

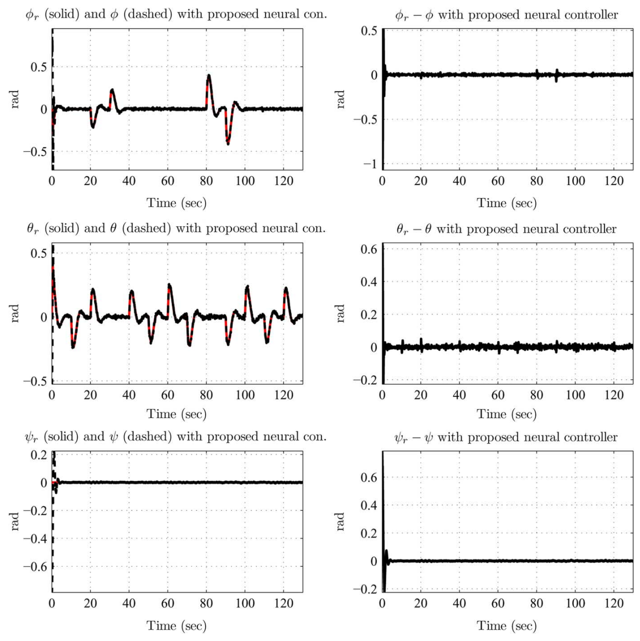

Fig. 9. Results for the Euler angles (attitude) utilizing the proposed neural controller.

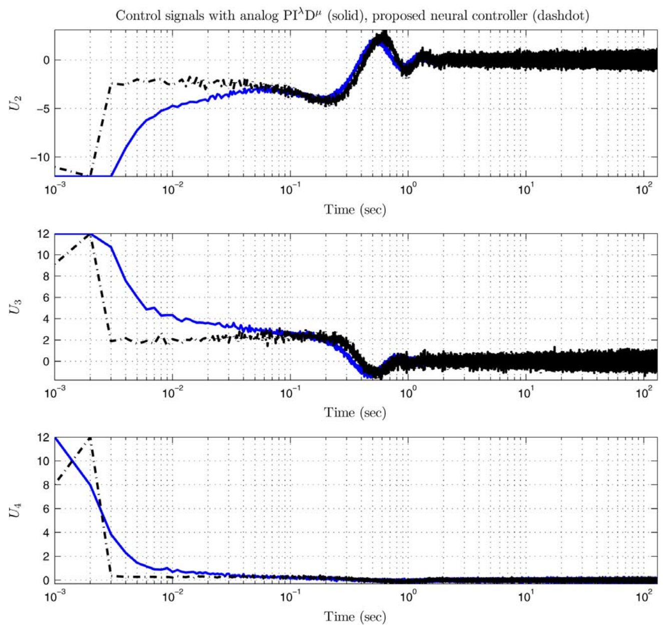

Fig. 10. Time evolution of the applied control signals for both analog PI$^\lambda$D$^\mu$ controller and proposed neural controller.

and this shows that the proposed controller is a good substitute for the analog PI$^\lambda$D$^\mu$ controller.

Having such a promising picture in front of us, one naturally wonders the computational burden of the proposed method. We utilize a neural network having structure 3-20-1 and hyperbolic tangent type activation functions and obtain the necessary parameters to run (9). Together with (9), the time necessary to compute the output is less than 20 μs over one million runs executed on an ordinary 2.26 GHz laptop. The order at microseconds indicate that the proposed technique, when considered with the performance associated to it, is as capable as its analog counterpart, which has a 0.15 s of propagation time over same number of trials. This comparison clearly indicates that the proposed technique is significantly simpler than the analog PI$^\lambda$D$^\mu$ controller.

V. Concluding Remarks

Fractional order systems and control have become an important research field as some of the applications produce much promising results compared to their integer order counterparts. With this motivation, this paper considers an application of PI$^\lambda$D$^\mu$ controller emulated by an artificial neural network aided linear model. Due to the powerful mapping capability of neural structures and the availability of fast and useful training schemes, the problem of implementing a PI$^\lambda$D$^\mu$ controller could be considered as a neural network-based coefficient scheduling problem. The paper explains the procedure for generating the training data with a brief on Levenberg–Marquardt optimization technique and applies the proposed controller to a quadrotor type UAV, which displays a nonlinear and inextricably intertwined dynamics involved with winds, powering uncertainties and measurement noise. The proposed model effectively emulates the analog PI$^\lambda$D$^\mu$ controller and provides extensive flexibility for microprocessor or FPGA-based real-time applications, [27]. The paper:

- Describes an easily reconfigurable PI$^\lambda$D$^\mu$ controller module making it possible to prototype adaptive PI$^\lambda$D$^\mu$ control systems.
- Advances the subject area to the development of PI$^\lambda$D$^\mu$ controllers that are emulated by particular models having a simple structure.
- Enriches the subject area with the models that predict the coefficients of a prespecified structure (e.g., FIR based) instead of emulating a PI$^\lambda$D$^\mu$ controller directly. The latter necessitates more complicated models entailing the observation of a significant amount of past inputs and outputs (see [28] for such a case) and refer to (2) and (3) to see the influence of the course of time on the current output.
- Makes it possible to develop the PI$^\lambda$D$^\mu$ control applications with computational simplicity and precision enabling real time embedded realization on commercial-off-the-shelf microcontrollers.
- Advances the subject area to the applications having industrial nature and entailing PI$^\lambda$D$^\mu$ type controller for best performance.
- Describes a technique that can be extended to a large class of intelligent systems. Though not elaborated here, the applications that prefer other forms of intelligence, such as fuzzy logic, support vector machines, radial basis function networks or their hybrid combinations can be used for the function provided by the neural network component of the presented approach. Although we utilize neural networks here, one could follow the presented scheme for any of the above architectures.

In summary, the paper demonstrates that the coefficients of a simple structure can be scheduled accurately to achieve a well performing closed-loop control system involving nonlinearities, noisy observations, uncertainties, and stringent performance expectations.

References

[1] M. H. Kim, S. Lee, and K. C. Lee, "Kalman predictive redundancy system for fault tolerance of safety-critical systems," *IEEE Trans. Ind. Informat.*, vol. 57, no. 1, pp. 46–53, Jan. 2010.

[2] H. Schweppe, A. Zimmermann, and D. Grill, "Flexible on-board stream processing for automotive sensor data," *IEEE Trans. Ind. Informat.*, vol. 57, no. 1, pp. 81–92, Jan. 2010.

[3] C.-L. Lai and P.-L. Hsu, "Design the remote control system with the time-delay estimator and the adaptive smith predictor," *IEEE Trans. Ind. Informat.*, vol. 57, no. 1, pp. 73–80, Jan. 2010.

[4] K. B. Oldham and J. Spanier, *The Fractional Calculus*. New York: Academic, 1974.

[5] I. Podlubny, *Fractional Differential Equations*, 1st ed. New York: Elsevier, 1998.

[6] S. Das, *Functional Fractional Calculus for System Identification and Controls*, 1st ed. New York: Springer, 2008.

[7] D. Matignon and B. d'Andera-Novel, "Observer based controllers for fractional differential systems," in *Proc . Int. Conf. Decision Control*, San Diego, CA, 1997, pp. 4967–4972.

[8] M. D. Ortigueira, "Introduction to fractional linear systems. Part 1: Continuous time case," *IEE Proc. Vis. Image Signal Process*, vol. 147, no. 1, pp. 62–70, 2000.

[9] B. M. Vinagre, C. A. Monje, and A. J. Calderon, "Fractional order systems and fractional order control actions," in *Proc. 41st IEEE Int. Conf. Decision Control*, Las Vegas, NV, Dec. 10–13, 2002, pp. 15–38.

[10] I. Podlubny, "Fractional-order systems and (PID mu)-D-lambda-controllers," *IEEE Trans. Autom. Control*, vol. 44, no. 1, pp. 208–214, Jan. 1999.

[11] C. Zhao, C. Xue, and Y.-Q. Chen, "A fractional order PID tuning algorithm for a class of fractional order plants," in *Proc. IEEE Int. Conf. Mech. Autom.*, Niagara Falls, Canada, 2005, pp. 216–221.

[12] R. Caponetto, L. Fortuna, and D. Porto, "Parameter tuning of a non integer order PID controller," in *Proc Prof. 15th Int. Symp. Math. Theory of Networks and Syst.*, Notre Dame, IN, 2002, paper no. 7434.

[13] D. Valerio and L. Sa Da Costa, "Tuning of fractional PID controllers with Ziegler-Nichols-type rules," *Signal Process.*, vol. 86, pp. 2771–2784, 2006.

[14] C. A. Monje, B. M. Vinagre, V. Feliu, and Y.-Q. Chen, "Tuning and auto-tuning of fractional order controllers for industry applications," *Control Eng. Practice*, vol. 16, pp. 798–812, 2006.

[15] J. F. Leu, S. Y. Tsay, and C. Hwang, "Design of optimal fractional-order PID controllers," *J. Chinese Inst. Chem. Eng.*, vol. 33, no. 2, pp. 193–202, 2002.

[16] B. M. Vinagre, I. Podlubny, I. Dorcak, and V. Feliu, "On fractional PID controllers: A frequency domain approach," in *Proc. IFAC Workshop Digital Control. Past, Present, Future of PID Control*, Terrasa, Spain, 2000, pp. 53–58.

[17] B. M. Vinagre, I. Podlubny, A. Hernandez, and V. Feliu, "Some approximations of fractional order operators used in control theory and applications," in *Proc. 41st IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 10–13, 2002, pp. 82–99.

[18] A. Oustaloup, P. Melchior, P. Lanusse, O. Cois, and F. Dancla, "The CRONE toolbox for matlab," in *Proc. 41st IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 10–13, 2002, pp. 240–245.

[19] O. Cois, P. Lanusse, P. Melchior, F. Dancla, and A. Oustaloup, "Fractional systems toolbox for MATLAB: Applications in system identification and crone CSD," in *Proc. 41st IEEE Conf. Decision Control*, Las Vegas, NV, Dec. 10–13, 2002, pp. 246–251.

[20] I. Podlubny, I. Petras, B. M. Vinagre, P. O'Leary, and L. Dorcak, "Analogue realizations of fractional-order controllers," *Nonlinear Dynamics*, vol. 29, pp. 281–296, 2002.

[21] S. Cong and Y. Liang, "PID-like neural network nonlinear adaptive control for uncertain multivariable motion control systems," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3872–3879, Oct. 2009.

[22] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, 1990.

[23] B. M. Wilamowski, "Neural network architectures and learning algorithms," *IEEE Ind. Electron. Mag.*, vol. 3, no. 4, pp. 56–63, Dec. 2009.

[24] M. T. Köroglu, M. Önkol, and M. Ö. Efe, "Experimental modelling of propulsion transients of a brushless DC motor and propeller pair under limited power conditions: A neural network based approach," in *Proc. 2nd IFAC Int. Conf. Intell. Control Syst. Signal Process. (ICONS09)*, İstanbul, Turkey, Sept. 21–23, 2009.

[25] D. Valério, Ninteger v.2.3 Fractional Control Toolbox for MatLab. 2005.

[26] G. Cai, B. M. Chen, K. Peng, M. Dong, and T. H. Lee, "Modeling and control of the yaw channel of a UAV helicopter," *IEEE Trans. Ind. Electron.*, vol. 55, no. 9, pp. 3426–3434, Sep. 2008.

[27] A. Dinu, M. N. Cirstea, and S. E. Cirstea, "Direct neural networks hardware implementation algorithm," *IEEE Trans. Ind. Electron.*, vol. 57, no. 5, pp. 1845–1848, May 2010.

[28] S. Abbisso, R. Caponetto, O. Diamante, L. Fortunat, and D. Porto, "Non integer order integration by using neural networks," in *Proc. 2001 IEEE Int. Symp. Circuits Syst., (ISCAS'2001)*, May 6–9, 2001, vol. 2, pp. 688–691.

Mehmet Önder Efe (S'95–M'04–SM'08) received the B.Sc. degree from Department of Electronics and Communications Engineering, İstanbul Technical University, İstanbul, Turkey, in 1993, M.S. degree from the Department of Systems and Control Engineering, Boğaziçi University, Boğaziçi, Turkey, in 1996, and the Ph.D. degree from the Department of Electrical and Electronics Engineering, Boğaziçi University, June 2000.

From August 1996 to December 2000, he was with the Mechatronics Research and Application Center, Boğaziçi University, as a Research Assistant. During 2001, he was a Postdoctoral Research Fellow at the Department of Electrical and Computer Engineering, Carnegie Mellon University and was a member of the Advanced Mechatronis Laboratory team. Between January 2002 and July 2003, he was with the Department of Electrical Engineering, Ohio State University, as a Postdoctoral Research Associate. He worked at the Collaborative Center of Control Science. As of September 2003, he started working at the Department of Mechatronics Engineering, ılım University, as an Assistant Professor. He was entitled Associate Professor in 2004 and Full Professor in 2009. In 2004, he joined the Department of Electrical and Electronics Engineering , TOBB University of Economics and Technology. He was the Head of the department from August 2004 to July 2007 and from June 2008 to August 2010. He has initiated the M.S. and Ph.D. programs in electrical and electronics engineering at TOBB ETU. He has taken several administrative positions at TOBB ETU. Since December 2010, he has been with the Department of Electrical and Electronic Engineering, Bahçeşehir University, as a Full Professor. He is the author/coauthor of two books and more than 140 technical publications focusing on the applications of computational intelligence, unmanned aerial vehicles and systems and control theory.

Dr. Efe was the Head of IEEE CSS Turkey Chapter from January 2007 to December 2008. He serves as an Associate Editor for the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the *Transactions of the Institute of Measurement and Control*, the *International Journal of Industrial Electronics and Control* and *Advances in Fuzzy Systems*.