Automation, Control and Intelligent Systems
2025, Vol. 13, No. 3, pp. 86-115
https://doi.org/10.11648/j.acis.20251303.14

Research Article

Neural Network-Based Online Model Identification and Nonlinear Adaptive Predictive Control of Self-Balancing Two-Wheel LEGO Mindstorms NXTway-GS Robot

Vincent Andrew Akpan[1, *], **Ahmed-Ade Fatai**[2], **Olugbenga Kayode Ogidan**[3]

[1]Department of Biomedical Engineering, The Federal University of Technology, Akure, Nigeria
[2]Department of Physics Electronics, The Federal University, Lokojae, Nigeria
[3]Department of Electrical and Electronics Engineering, Elizade University, Ilara-Mokin, Nigeria

Abstract

The main challenge of the well-celebrated Levenberg-Marquardt algorithm (LMA) is the selection of the searching direction and adaptation parameters. Secondly, the implementation of the LMA for online model identification has faced challenges as it is a batch optimization. As a third challenge, the solution of the Levenberg-Marquardt based on the full-Newton nonlinear optimization (FNNO) for online applications have been limited due to its unguaranteed positive definiteness. This paper presents two versions of the modified Levenberg-Marquardt algorithm (MLMA) for neural network model identification and adaptive predictive control for the online dynamic model identification and adaptive control of a self-balancing two-wheel LEGO Mindstorms NXTway-GS robot. The first version is the online-window-approach of the modified Levenberg-Marquardt algorithm (OWA-MLMA) based on approximate Guass-Newton algorithm (AGNA) for training neural network model predictor. The second version is a neural network-based adaptive predictive control (APC) algorithm based on the full-Newton nonlinear optimization of the modified Levenberg-Marquardt algorithm (FNNO-MLMA) for online adaptive control. A NNARMAX model predictor for the NXT robot is first trained and validated using the OWA-MLMA based on AGNA. The validated NNRAMAX model is then used for the design of the NN-APC based on FNNO-MLMA. Finally, the model identification based on OWA-MLMA and APC based on FNNO-MLMA schemes are simulated in closed-loop for online NNARMAX model identification and adaptive predictive control of the NXT robot. The comparison of the proposed OWA-MLMA based on AGNA shows superior performance over the recursive incremental back-propagation algorithm (INCBPA) while the proposed NN-APC based on FNNO-MLMA shows excellent control and good tracking performances over a discrete-time fixed-parameter proportional-integral-derivative (PID) controller for the NXT robot control. The simulation results show that the developed OWA-MLMA based on AGNA and the APC based on FNNO-MLMA can be adapted for the online dynamic modeling, automation, control and robotics applications.

Keywords

Approximate Gauss Newton Algorithm (AGNA), Full-Newton Nonlinear Optimization (FNNO), LEGO Mindstorms NXT Robot, Modified Levenberg-Marquardt Algorithm (MLMA), Neural Networks, Nonlinear Adaptive Predictive Control (NAPC), Online-Window-Approach (OWA), Self-Balancing Two-Wheel Robot

[*]Corresponding author: vaakpan@futa.edu.ng (Vincent Andrew Akpan)

Received: 21 June 2025; **Accepted:** 5 July 2025; **Published:** 25 September 2025

1. Introduction

The term robot means different things to different people. Science fiction books and movies have strongly influenced what many people expect a robot to be or what it can do. Sadly the practice of robotics is far behind this popular conception [1]. Basically, the design, implementation and control of any class or type of robots present a challenge to the designer [1-3]. In particular, the design of two-wheeled robot control system represents a challenge to the designer [2]. The robot motion in horizontal and vertical planes is described by a nonlinear model whose derivation may be a difficult task [1-3]. The linearization of this model leads to unstable non-minimum phase plant which should be stabilized in the presence of parameter variations, noises, and disturbances [2, 4-6].

During the last few years a growing interest was observed in using laboratory devices in control education. Among different designs robotic and oscillatory devices have become most popular [1-3, 7]. New practical problems such as the control of networks and cyber-physical systems demand both for new theory and for new design and education means. Fortunately, the developments of computer technologies lead to creation of convenient computation and communication environments, supporting design and education.

One of the recent milestones on this way is LEGO Mindstorms NXT [7-10]. The LEGO Mindstorms NXT two-wheeled self-balancing robot (simply referred to in this study as NXT robot) has become a standard robot for developing several classes of humanoid robot [1-3, 7-10].

LEGO Mindstorms NXT Robot and Oscillators have been used to demonstrate the concept of adaptive control in education [7-10]. Autonomous Mobile Robot has been implemented using LEGO EV3 integrated with Raspberry Pi which uses android-based vision control algorithms for human-machine interaction [11]. Different variations and versions of adaptive control has been applied for the study and modeling of the LEGO Mindstorms NXT two-wheeled self-balancing robot based purely on linear models of the NXT robot as well as their implementation for different demonstrations and applications [2, 12, 13]. However, there is a strong differences between adaptive control based on model reference adaptive control (MRAC) and model predictive control (MPC) [14-16]. Rather than the MRAC, the MPC strategy is adopted in this work.

Model predictive control (MPC) is a class of advanced control algorithms that utilize an explicit process model to predict the future response of the process and have been proven to be very successful in many industrial applications [16-22].

The design and implementation of online modeling and control algorithms for any class or type of robots with relatively short sampling time presents several challenges [1-3]. Rather than using linear model of the NXT robot, nonlinear modeling approach based on neural network is proposed in this work [17]. In fact, the algorithms obtained by MPC design techniques which are based on a linear mathematical model of the controlled process are not very efficient because these methods cannot guarantee stable control of the system outside the range of the model validity [17-22].

The present study focuses on the development of a neural network-based nonlinear modeling technique and a neural network-based nonlinear model adaptive predictive control approach for the online model identification and adaptive predictive control of the NXT robot.

The paper proposes an iterative technique for updating the Levenberg-Marquardt parameter which is a difficulty task for adapting and updating the NN parameters which are the weights and biases of the network.

The main challenge of the well-celebrated Levenberg-Marquardt algorithm (LMA) is the selection of the searching direction and adaptation parameters. Secondly, the implementation of the LMA for online model identification has faced challenges as it is a batch optimization. As a third challenge, the solution of the Levenberg-Marquardt based on the full-Newton nonlinear optimization (FNNO) for online applications have been limited due to its unguaranteed positive definiteness. Thus, by creating a continuously updated sliding stack window, the paper proposes an online implementation of the Levenberg-Marquardt algorithm. We refer to the proposed NN training algorithm as the online-window-approach based on modified Levenberg-Marquardt algorithm (OWA-MLMA) for training the NN model predictor.

The use of nonlinear NN model for nonlinear MPC has also been reported where the NN model is identified off-line and employed for online-line nonlinear MPC design based on the full-Newton method [23, 24]. The well-known problem with the full-Newton method is that the second-order Hessian matrix is not guaranteed to be positive definite in an open neighbourhood of a global minimum among other issues which have made this method unsuitable for online nonlinear adaptive model predictive control applications. This issue still remains open under current research and depends also on the accuracy of the online nonlinear model identification of the process [16, 18, 19].

This paper adopts the nonlinear adaptive model predictive control strategy which uses a nonlinear NN model for the controller design [16, 17]. The proposed control strategy is an online nonlinear optimization based on the full-Newton method using the modified Levenberg-Marquardt algorithm (MLMA) [15-17]. The proposed nonlinear adaptive predictive control (NAPC) scheme incorporates an iterative scheme for guaranteed positive definiteness of Hessian matrix and then applies another iterative technique to adjust and update the Levenberg-Marquardt parameter.

The paper is structured as follows. The development of the

mathematical model, formulation of the control problem and the design of the desired reference trajectory for the NXT Robot is presented in Section 2. The neural network-based online-window-approach of the modified Levenberg-Marquardt algorithm (OWA-MLMA) based on approximate Gauss-Newton algorithm (AGNA) for training neural network for nonlinear model identification is formulated in Section 3. The summary of the incremental back-propagation algorithm (INCBPA) is also given in this Section. Then, Section 4 presents the complete formulation of the nonlinear adaptive predictive control (NAPC) algorithm based on MLMA using the full-Newton nonlinear optimization (FNNO-MLMA) and a PID control law for the NXT robot control. In Section 5, the off-line closed-loop implementation of the OWA-MLMA for NNARMAX model identification of the NXT robot is first presented with simulation results. Secondly, the identified NNARMAX model is then used to design the NAPC and PID controller for performance comparison. Lastly, the online closed-loop implementation of the OWA-MLMA and NAPC based on FNNO-MLMA and the simulation results are discussed and presented in this Section. Section 6 concludes the paper and highlights its major contributions and directions for further work.

Figure 1. *Self-balancing two-wheeled LEGO Mindstorms NXTway-GS.*

2. Development of the Mathematical Model, Formulation of the Control Problem, and the Design of the Desired Reference Trajectory for the NXT Robot

2.1. Mathematical Modeling of the NXT Robot as a Two-Wheel Inverted Pendulum

The two-wheeled NXT robot as an inverted pendulum shown in Figure 1 can be considered as a two-wheel inverted pendulum model as shown in Figure 2. The side and plane views of the two-wheel inverted pendulum model is shown in Figure 3. The coordinate system of Figure 3 can be used to formulate and derive the equations of motion for the two-wheel inverted pendulum as discussed in the next sub-section. Note that in Figure 3, ψ is the body pitch angle, $\theta_{l,r}$ is the wheel angle (where *l* and *r* indicates left and right angles), and $\theta_{m_{l,r}}$ is the DC motor angle. The physical parameters of the NXT robot are defined in Table 1.

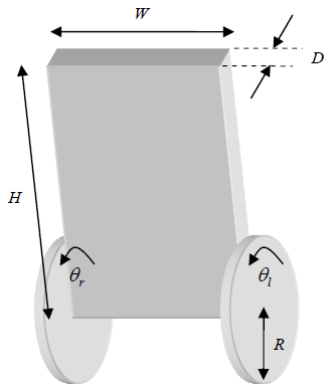
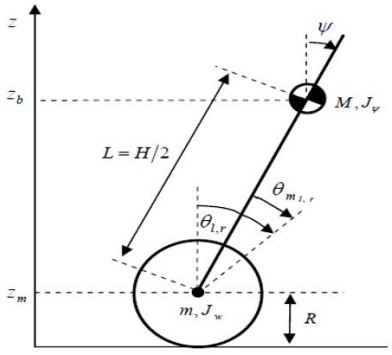
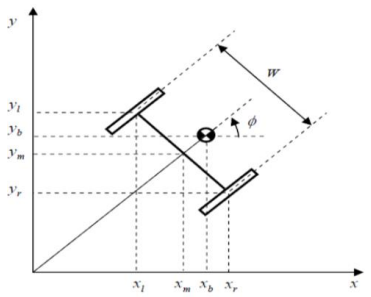


Figure 2. *The schematic of the two-wheeled inverted pendulum robot.*



(a)



(b)

Figure 3. *(a) Side view and (b) the plane view of the two-wheeled inverted pendulum.*

2.1.1. Equations of Motion for the Two-Wheel Inverted Pendulum

We can now derive the equations of motion for the two-wheel inverted pendulum by the Lagrangian method based on the coordinate system in Figure 3. If the direction of the two-wheel inverted pendulum is *x*-axis positive direction at $t = 0$, each coordinates are given as the following [2, 25]:

$$(\theta, \phi) = \left(\frac{1}{2}(\theta_t + \theta_r) \quad \frac{R}{W}(\theta_r - \theta_t) \right) \tag{1}$$

$$\left.\begin{aligned} (x_m, y_m, z_m) &= \left(\int \dot{x}_m dt, \int \dot{y}_m dt, R \right) \\ (\dot{x}_m, \dot{y}_m) &= \left(R\dot{\theta} cos\phi \quad R\dot{\theta} \sin\phi \right) \end{aligned}\right\} \tag{2}$$

$$(x_t, y_t, z_t) = \left(x_m - \frac{W}{2}\sin\phi, \quad y_m + \frac{W}{2}\cos\phi, \quad z_m \right) \tag{3}$$

Table 1. *Physical parameters, numerical values and units for the self-balancing two-wheeled NXT robot.*

S/N	Physical Parameters	Values	Units
1	Gravitational acceleration (g)	9.810	m.sec^{-2}
2	Wheel weight (m)	0.030	kg
3	Wheel radius (R)	0.040	m
4	Wheel inertia moment (J_W)	mR^2/2	kg.m^2
5	Body weight (M)	0.600	kg
6	Body width (W)	0.140	m
7	Body depth (D)	0.040	m
8	Body height (H)	0.144	m
9	Distance of the centre of mass from the wheel axle (L)	H/2	m
10	Body pitch inertia moment (J_ψ)	ML^2/3	kg.m^2
11	Body yaw inertia moment (J_ϕ)	M (W^2 + D^2)/12	kg.m^2
12	DC motor inertia moment (J_m)	1 x 10^{-5}	kg.m^2
13	DC motor resistance (R_m)	6.690	Ω
14	DC motor back EMF constant (K_b)	0.468	V.sec.rad^{-1}
15	DC motor torque constant (K_t)	0.317	Mn.A^{-1}
16	Gear ratio (n)	1.000	-
17	Friction coefficient between body and DC motor (f_m)	0.002	-
18	Friction coefficient between wheel and floor (f_W)	0.000	-

$$(x_r, y_r, z_r) = \left(x_m - \frac{W}{2}\sin\phi, \quad y_m + \frac{W}{2}\cos\phi, \quad z_m \right) \tag{4}$$

$$(x_b, y_b, z_b) = \begin{pmatrix} x_m + L\sin\psi\cos\phi, \\ y_m + L\sin\psi\sin\phi, \\ z_m + L\cos\psi \end{pmatrix} \tag{5}$$

The translational kinetic energy T_1, the rotational kinetic energy T_2 and the potential energy U are given respectively by:

$$\left.\begin{aligned} T_1 = \frac{1}{2}m\left(\dot{x}_l^2 + \dot{y}_l^2 + \dot{z}_l^2\right) + \frac{1}{2}m\left(\dot{x}_r^2 + \dot{y}_r^2 + \dot{z}_r^2\right) \\ + \frac{1}{2}m\left(\dot{x}_b^2 + \dot{y}_b^2 + \dot{z}_b^2\right) \end{aligned}\right\} \tag{6}$$

$$\left.\begin{aligned} T_2 = \frac{1}{2}J_w\dot{\theta}_l^2 + \frac{1}{2}J_w\dot{\theta}_r^2 + \frac{1}{2}J_\psi\dot{\psi}^2 + \frac{1}{2}J_\phi\dot{\phi}^2 \\ + \frac{1}{2}n^2J_m\left(\dot{\theta}_l - \dot{\psi}\right)^2 + \frac{1}{2}n^2J_m\left(\dot{\theta}_r - \dot{\psi}\right)^2 \end{aligned}\right\} \tag{7}$$

$$U = mgz_l + mgz_r + mgz_b \tag{8}$$

The fifth and sixth terms in T_2 are rotational kinetic energy of an armature in left and right DC motor. The Lagrangian (L) has the following expression:

$$L = T_1 + T_2 - U \tag{9}$$

where the generalized coordinates are defined with θ = average angle of left and right wheel, ψ = body pitch angle, and ϕ = body yaw angle.

The Lagrangian equations are the following:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = F_\theta \tag{10}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\psi}}\right) - \frac{\partial L}{\partial \psi} = F_\psi \tag{11}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \frac{\partial L}{\partial \phi} = F_\phi \tag{12}$$

Evaluation of (10)-(12) gives the following set of equations:

$$\left.\begin{aligned} F_\theta = \left[\left(2m + M\right)R^2 + 2J_w + 2n^2J_m\right]\ddot{\theta} \\ + \left(MLR\cos\psi - 2n^2J_m\right)\ddot{\psi} - MLR\dot{\psi}^2\sin\psi \end{aligned}\right\} \tag{13}$$

$$\left.\begin{aligned} F_\psi &= \left(MLR\cos\psi - 2n^2 J_m\right)\ddot{\theta} \\ &+ \left(ML^2 + J_\psi + 2n^2 J_m\right)\ddot{\psi} \\ &- MgL\sin\psi - ML^2\dot{\phi}^2\sin\psi\cos\psi \end{aligned}\right\} \tag{14}$$

$$\left.\begin{aligned} F_\phi = \Bigg[&\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2 J_m\right) \\ &+ ML^2\sin^2\psi \\ &+ 2ML^2\dot{\psi}\dot{\phi}\sin\psi\cos\psi \Bigg]\ddot{\phi} \end{aligned}\right\} \tag{15}$$

In consideration of DC motor torque and viscous friction, the generalized forces are given as the following equations:

$$\left(F_\theta,\, F_\psi,\, F_\phi\right) = \left(F_1 + F_2,\, F_\psi,\, \frac{W}{2R}\left(F_r + F_l\right)\right) \tag{16}$$

where

$$F_l = nK_t i_l + f_m\left(\dot{\psi} - \dot{\theta}_l\right) - f_w\dot{\theta}_l \tag{17}$$

$$F_r = nK_t i_r + f_m\left(\dot{\psi} - \dot{\theta}_r\right) - f_w\dot{\theta}_r \tag{18}$$

$$F_\psi = -nK_t i_l - nK_t i_r - f_m\left(\dot{\psi} - \dot{\theta}_l\right) - f_m\left(\dot{\psi} - \dot{\theta}_r\right) \tag{19}$$

and $i_{l,r}$ is the DC motor current.

It should be noted that we cannot use the DC motor current directly in order to control the motor because it is based on PWM (voltage) control. Therefore, we need to evaluate the relation between current $i_{l,r}$ and voltage $v_{l,r}$ using DC motor equations. If the friction inside the motor is negligible, the DC motor equation is generally as follows:

$$L_m\dot{i}_{i,r} = v_{i,r} + K_b\left(\dot{\psi} - \dot{\theta}_{i,r}\right) - R_m i_{i,r} \tag{20}$$

Here we consider that the motor inductance is negligible and is approximated as zero. Therefore, the current becomes

$$i_{i,r} = \frac{v_{i,r} + K_b\left(\dot{\psi} - \dot{\theta}_{i,r}\right)}{R_m} \tag{21}$$

From (21), the generalized force can be expressed using the motor voltage as follows:

$$F_\theta = \alpha\left(v_l + v_r\right) - 2\left(\beta + f_w\right)\dot{\theta} + 2\beta\dot{\psi} \tag{22}$$

$$F_\psi = -\alpha\left(v_l + v_r\right) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \tag{23}$$

$$F_\phi = \frac{W}{2R}\alpha\left(v_r - v_l\right) - \frac{W^2}{2R^2}\left(\beta + f_w\right)\dot{\phi} \tag{24}$$

where

$$\alpha = \frac{nK_t}{R_m} \text{ and } \beta = \frac{nK_t K_b}{R_m} + f_m \tag{25}$$

2.1.2. State Equations for the Two-Wheel Inverted Pendulum

We can now derive the state equations based on modern control theory by linearizing the equations of motion at a balance point of the robot based on the Taylor's series [2-6]. It means that we consider the limit $\psi \to 0 (\sin\psi \to \psi,\, \cos\psi \to 1)$ and neglect the second-order term like $\dot{\psi}^2$. The motion equations (13-15) are approximated according to the following equations respectively:

$$\left.\begin{aligned} F_\theta = &\left[\left(2m + M\right)R^2 + 2J_w + 2n^2 J_m\right]\ddot{\theta} \\ &+ \left(MLR - 2n^2 J_m\right)\ddot{\psi} \end{aligned}\right\} \tag{26}$$

$$\left.\begin{aligned} F_\psi = &\left(MLR - 2n^2 J_m\right)\ddot{\theta} + \left(ML^2 + J_\psi + 2n^2 J_m\right)\ddot{\psi} \\ &- MgL\psi \end{aligned}\right\} \tag{27}$$

$$F_\phi = \left[\frac{1}{2}mW^2 + J_\phi + \frac{W^2}{2R^2}\left(J_w + n^2 J_m\right)\right]\ddot{\phi} \tag{28}$$

Note that both (26) and (27) have θ and ψ, whereas (28) has ϕ only. These equations can be expressed in the following forms:

$$E\begin{bmatrix}\ddot{\theta}\\ \ddot{\psi}\end{bmatrix} + F\begin{bmatrix}\dot{\theta}\\ \dot{\psi}\end{bmatrix} + G\begin{bmatrix}\theta\\ \psi\end{bmatrix} = H\begin{bmatrix}v_l\\ v_r\end{bmatrix} \tag{29}$$

where

$$E = \begin{bmatrix}\left(2m + M\right)R^2 + 2J_w + 2n^2 J_m & MLR - 2n^2 J_m \\ MLR - 2n^2 J_m & ML^2 + J_\psi + 2n^2 J_m\end{bmatrix}$$

$$F = 2\begin{bmatrix}\beta + f_w & -\beta \\ -\beta & \beta\end{bmatrix},\; G = \begin{bmatrix}0 & 0 \\ 0 & -MgL\end{bmatrix},\; H = \begin{bmatrix}\alpha & \alpha \\ -\alpha & -\alpha\end{bmatrix}$$

and

$$\left.\begin{aligned} &I\ddot{\phi}+J\dot{\phi}+K\left(v_r-v_l\right),\\ &I=\frac{1}{2}mW^2+J_\phi+\frac{W^2}{2R^2}\left(J_w+n^2J_m\right),\\ &J=\frac{W^2}{2R^2}\left(\beta+f_w\right),\quad K=\frac{W}{2R}\alpha \end{aligned}\right\} \tag{30}$$

Here we consider the variables x_1, x_2 as state variables and u as the input defined respectively by the following expressions:

$$x_1=\left[\theta,\psi,\dot{\theta},\dot{\psi}\right]^T,\ x_2=\left[\phi,\dot{\phi}\right]^T,\ u=\left[v_l,v_r\right]^T \tag{31}$$

Consequently, we can derive state equations for the two-wheeled inverted pendulum robotic system from (29) and (30) as follows:

$$\dot{x}_1=A_1x_1+B_1u \tag{32}$$

$$\dot{x}_2=A_2x_2+B_2u \tag{33}$$

where

$$\left.\begin{aligned} A_1&=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&A_1(3,2)&A_1(3,3)&A_1(3,4)\\0&A_1(4,2)&A_1(4,3)&A_1(4,4)\end{bmatrix}\\ B_1&=\begin{bmatrix}0&0\\0&0\\B_1(3)&B_1(3)\\B_1(4)&B_1(4)\end{bmatrix}\end{aligned}\right\} \tag{34}$$

$$A_2=\begin{bmatrix}0&1\\0&-J/I\end{bmatrix},\quad B_2=\begin{bmatrix}0&0\\-K/I&K/I\end{bmatrix} \tag{35}$$

and

$$\left.\begin{aligned} &A_1(3,2)=-gMLE(1,2)/\det(E)\\ &A_1(4,2)=gMLE(1,1)/\det(E)\\ &A_1(3,3)=-2\left[\left(\beta+f_w\right)E(2,2)+\beta E(1,2)\right]/\det(E)\\ &A_1(4,3)=2\left[\left(\beta+f_w\right)E(1,2)+\beta E(1,1)\right]/\det(E)\\ &A_1(3,4)=2\beta\left[E(2,2)+E(1,2)\right]/\det(E)\\ &A_1(4,4)=-2\beta\left[E(1,1)+E(1,2)\right]/\det(E)\\ &B_1(3)=\alpha\left[E(2,2)+E(1,2)\right]/\det(E)\\ &B_1(4)=-\alpha\left[E(1,1)+E(1,2)\right]/\det(E)\end{aligned}\right\} \tag{36}$$

with

$$\det(E)=E(1,1)E(2,2)-E(1,2)^2$$

2.2. Control Problem Formulation for the Self-Balancing Two-Wheel NXT Robot as an Inverted Pendulum

The characteristics of the self-balancing two-wheel NXT robot as a control system are described as follows.

2.2.1. Inputs and Outputs

The input to the actuator is PWM duty of the left and right DC motor even though input u in (31) is voltage. The outputs from sensors are the DC motor angle $\theta_{m_{l,r}}$ and the body pitch angular velocity $\dot{\psi}$. It is easy to evaluate θ and ϕ by using $\theta_{m_{l,r}}$. There are two methods to evaluate ψ by using $\dot{\psi}$, namely: *1)*. Derive ψ by integrating the angular speed numerically and *2)*. Estimate ψ by using an observer based on modern control theory. In this study, we use the second method in the following for the controller design chiefly because of easy stability establishment.

2.2.2. Stability of the NXTway-GS Robot

It is easy to understand that NXT robot balancing position is not stable. We have to move NXT robot in the same direction of body pitch angle to keep balancing. Modern control theory gives many techniques to stabilize an unstable system.

Equation (29) is a similar equation as mass-spring-damper system. Figure 4 shows an equivalent system of two-wheeled inverted pendulum interpreted as mass-spring-damper system. We find out that we can make the two-wheeled inverted pendulum stable by adjusting spring constants and damper friction constants in Figure 4. The Simulink model of the self-balancing two-wheel NXT robot is shown in Figure 5.

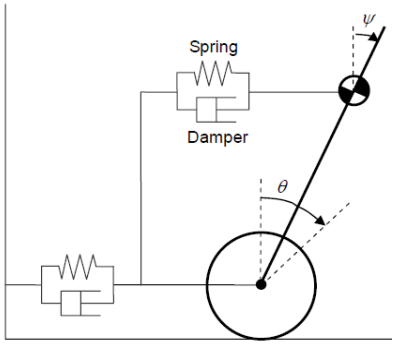


Figure 4. *Equivalent mass-spring-damper system for one side of the two-wheeled inverted pendulum of the NXT robot.*

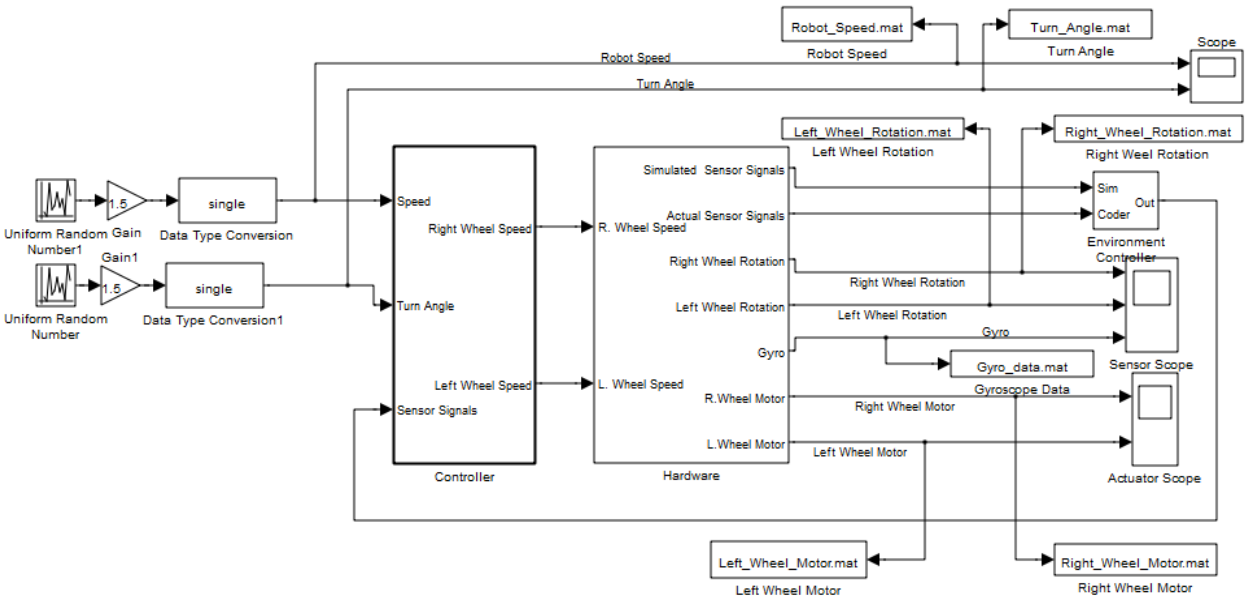


Figure 5. *Simulink model of the self-balancing two-wheel NXT robot.*

2.2.3. Simulation of the MATLAB/Simulink Model of the NXT Robot for Data Generation

The NXT robot uses the model reference feature as the controller and plant parts that have different Simulink model files. This modeling approach allows a parallel development process for controller design and physical plant design. Simulink buses and bus objects are used to keep the interface between the controller model and plant model. The MATLAB/Simulink, from the MathWorks [26], is used to program, model and simulate the mathematical model of the self-balancing two-wheel NXT robot. The data generated from the simulation MATLAB/Simulink model of the self-balancing two-wheel NXT robot are shown in Figure 6(a) and 6(b) for 250,000 data samples. The inputs to the NXT robot are the voltages to the Left and Right wheels of the self-balancing two-wheel NXT robot a shown in Figure 6(a) and 6(b) respectively. The outputs of the NXT robot are the angles of rotation of the Left and Right wheels of the self-balancing two-wheel NXT robot as shown in Figure 7(a) and 7(b) respectively.

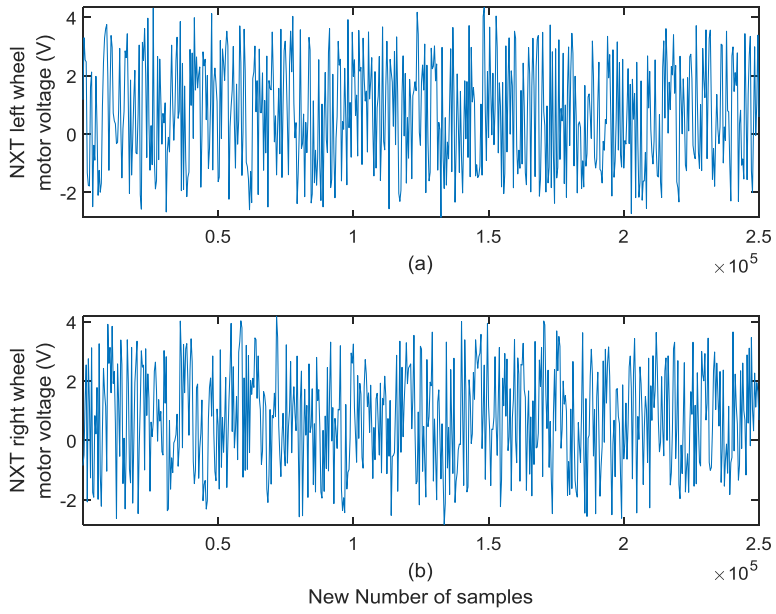


Figure 6. *The NXT robot motor input voltage (V): (a) Left wheel motor and (b) Right wheel motor.*

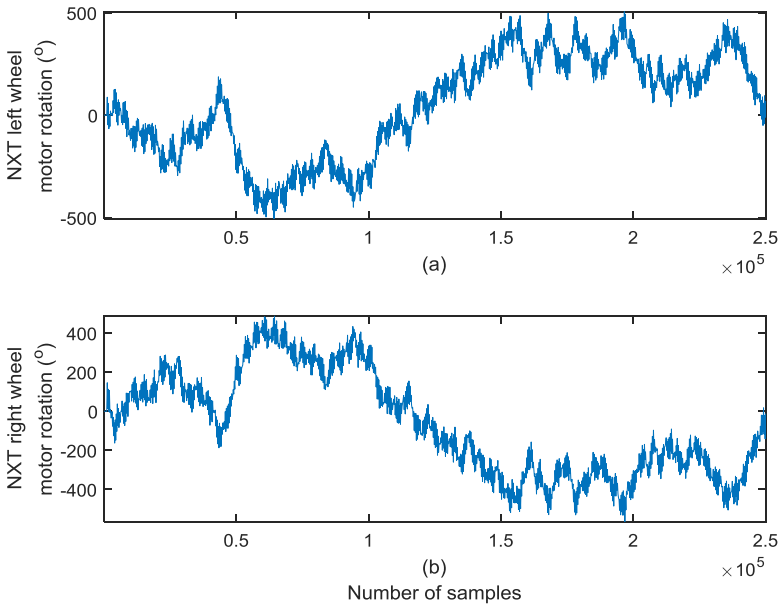


Figure 7. *The NXT robot output wheel rotation in degrees ( ): (a) Left wheel rotation and (b) Right wheel rotation.*

2.3. Desired NXT Robot Trajectory Design

2.3.1. Operation of the Robot Based on the Controlled Variable Based on the Manipulated Variables

The typical operation of the robot based on each of the controlled variables (i.e., the angle of rotation of the right and left wheels as the outputs) of the NXT Robot based on the manipulated variables (i.e., the input voltages to left and right wheels as the inputs) are described as follows:

1) For the robot to achieve forward movement, right and left wheel motor voltages should be kept high (i.e., +5 V). The robot wheel angle of rotation should be maintained at 90 °.
2) To achieve a backward movement of the robot, both the right and left wheel motor voltages should be reversed (i.e., -5 V). The robot wheel angle of rotation should be maintained at 90 °.
3) For the robot to turn to the right direction, the right wheel motor voltage should be stationary at logic low (i.e., stationary at 0 V) while the left wheel motor voltage should be maintained at logic high (i.e., +5 V). This will keep the robot left wheel moving while the robot turns right until the desired angle of rotation is achieved (i.e., from 90 °to 180 °, 270 °, 360 °and possibly back to 90 °as the case may be).
4) For the robot to turn to the left direction, the left wheel motor voltage should be stationary at logic low (i.e., stationary at 0 V) while the right wheel motor voltage should be maintained at logic high (i.e., +5 V). This will keep the robot right wheel moving while the robot turns left until the desired angle of rotation is achieved (i.e., from 90 °to -180 °, -270 °, -360 °and possibly back to 90 ° as the case may be).

2.3.2. Design of the Desired Reference Trajectory for the Robot Based on the Controlled Variables (CV) and the Manipulated Variables (MV)

The NXT-robot wheels have several features for achieving its movement, manipulative and functional characteristics. In the following discussion stationary is logic 0 (i.e. 0 V), logic low is -5 V and logic high is +5 V. Each sampling instant is 20 seconds which corresponds to to the sampling time of the robot as arbitrarily considered in this work.

At the first sampling instant both the left wheel and the right wheel of the NXT robot are stationary for a period of 20 seconds.

At second sampling instant, the left wheel and the right wheel are both at logic high and the NXT robot moves forward.

At the third sampling instant, the left wheel is at logic high while the right wheel is at logic low, then the NXT robot turns to the right wing.

At the fourth sampling instant, the left wheel is stationary and the right wheel is at logic high, the NXT robot is turning to the left wing.

At the fifth sampling instant, the left wheel is at logic low, while the right is also at logic low, the NXT robot moves backward.

At the sixth sampling instant, the left wheel is stationary while the right wheel is at logic high, then the NXT robot turns to the left wing.

At the seventh sampling instant**,** the left wheel is at logic high, while the right wheel is stationary, then theNXT robot then rotates to the right wing.

At the eighth sampling instant, both the wheels are at logic high; this makes the NXT robot to move forward again.

At the ninth sampling instant, the left wheel is stationary while the right wheel is at logic high, then the NXT robot rotates to the left.

At the tenth sampling instant, the left wheel is at logic high while the right wheel is stationary, and this forces the NXT robot to rotate to the right wing.

At the eleventh sampling instant, the left wheel is at logic low, while the right wheel is also at logic low, then the NXT robot movement is reversed and the robot moves backward.

At the twelfth sampling instant, the left wheel is at logic high, while the right wheel is stationary, then the NXT-robot rotates to the right wing.

At the thirteenth sampling instant, the left wheel is stationary, while the right wheel is at logic high, this makes the NXT robot to rotate to the left.

Finally, at the fourteenth sampling instant, the left wheel of the NXT robot is at logic low while the right wheel is at stationary, which forces the NXT robot to come to rest.

On the basis of the prescribed operation of the NXT robot based on the controlled variables (outputs) based on the manipulated variables (inputs) described in sub-section 2.3.1 and the desired reference trajectory design for the NXT robot based on the controlled variables and the manipulated variables discussed in Sub-Section 2.3.2 above, the desired reference trajectory for the NXT robot used in this work is constructed as shown in Figure 8.

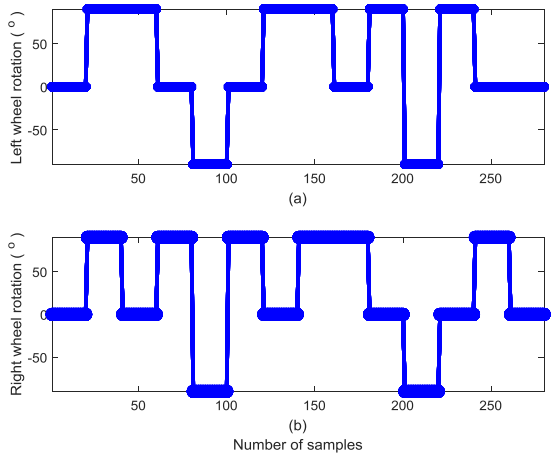


Figure 8. *The desired reference trajectories for the NXT Robot.*

3. Formulation of the OWA-MLMA Based on AGNA for Training NNARMAX Model Predictor

3.1. The OWA-MLMA Based on AGNA

In this section we first introduce the system description. In the next sub-section, we introduce the basic MLP NN and its extension to the series-parallel structure of NNARMAX model identification. Then in the last sub-section, we present the OWA-MLMA for training the NN to adjust and update the NN parameters (i.e. weights and biases) for NNARMAX model identification.

3.1.1. General System Description

The method of representing dynamical systems by vector difference or differential equations is well established in systems [27] and control [2, 16-22] theories. Consider that the *ith* output $y_i(k)$ of a discrete-time nonlinear *m*-input *n*-output multivariable system at time instant *k* responding to the *lth* input $u_l(k)$ can be described by the following Nonlinear AutoRegressive Moving Average with eXogeneous input (NARMAX) model:

$$\left.\begin{aligned} y(k) = J(&u(k-d), u(k-2), \ldots, u(k-d-m), \\ &y(k-1), y(k-2), \ldots, y(k-n), \\ &\varepsilon(k-1), \varepsilon(k-2), \ldots, \varepsilon(k-n)) + \tilde{d}(k) \end{aligned}\right\} \quad (37)$$

where $J(\bullet,\bullet,\bullet)$ is a nonlinear function of its input arguments assumed to be differentiable, $u(k-d), u(k-2), \ldots, u(k-d-m)$, $y(k-1), y(k-2), \ldots, y(k-n)$ and $\varepsilon(k-1), \varepsilon(k-2), \ldots, \varepsilon(k-n))$ are the vectors of the past input, past output and past prediction error values respectively while $\tilde{d}(k)$ are the disturbances and *d* is the system delay. Assuming that the input-output data pair Z^N of the system taken over *NT* period of time is available

$$Z^N = \{[u(1), y(1), \ldots, u(k), y(k)], \quad k = 1, 2, \ldots, N\}, \quad (38)$$

where *N* is the number of data pair and *T* the sampling time. Equation (37) can be expressed in a more compact form as

$$y(k) = J(Z^N, \varphi_0(k), \theta_0(k)) \quad (39)$$

where $\theta_0(k)$ is a known vector of appropriate dimension that defines the parameters of the system and $\varphi_0(k) = [u(k-d), \ldots, u(k-d-m), y(k-1), \ldots, y(k-n), \varepsilon(k-1, \theta(k)), \ldots, \varepsilon(k-n, \theta(k))]^T$ is the state (regression) vector at time k.

To set up a model identification problem, given (38), m, n, with an initial small random value of $\theta_0(k)$; at time $k+1$ we can construct a predictor $\hat{y}(k+1)$ that will produce the *a priori* predictor estimate of (39) $y(k+1)$ based on the state of the system $\varphi_0(k)$ at time k can be define as

$$\hat{y}^0(k+1) = J(Z^N, \varphi(k), \theta(k))$$

where $\theta(k)$ is an unknown adjustable parameter vector of appropriate dimension and the regression (state) vector $\varphi(k) = [u(k-d),\ldots,u(k-d-m),\ y(k-1),\ldots,y(k-n)\ \varepsilon(k-1,\theta(k)),\ldots,\varepsilon(k-n,\theta(k))]^T$. Thus, at time $k+1$ a new value of $\theta(k+1)$ will be known and the *a posteriori* predictor estimate can be computed as

$$\hat{y}(k+1) = J(Z^N, \varphi(k), \theta(k+1)) \tag{40}$$

where the predictor output $\hat{y}(k+1)$ is compared to the system output $y(k+1)$ and the error $\varepsilon(k+1)$ between the two outputs is minimized and used to adjust and update $\theta(k+1)$ such that $\hat{y}(k+1) \approx y(k+1)$. It is evident that the predictor output depends on $\theta(k+1)$. For notational convenience, we define the error here as:

$$\varepsilon(k,\theta(k)) = y(k) - \hat{y}(k \mid \theta(k)) \tag{41}$$

where $y(k) = y(k+1)$, $\hat{y}(k) = \hat{y}(k+1)$ and $\theta(k) = \theta(k+1)$ are the true system output, network predicted output and the adjustable parameters of the network. Thus, the determination and adjustment of $\theta(k)$ becomes an unconstrained nonlinear minimization problem defined here as:

$$\hat{\theta}(k) = \arg\min_{\theta} \vec{J}(Z^N, \varphi(k), \theta(k)) \tag{42}$$

where $\hat{\theta}(k)$ is the value of $\theta(k)$ that minimizes (42) and $\vec{J}(Z^N, \varphi(k), \theta(k))$ is formulated as a total square error (TSE) type cost function given as:

$$\vec{J}(Z^N, \varphi(k), \theta(k)) = \frac{1}{2N}\sum_{k=1}^{N}[\varepsilon(k,\theta(k))]^2 \tag{43}$$

Several techniques for solving (42) exist in literature [27-29]. Following the discussion in the Section I, the NN technique is proposed in this paper for solving (42) based on its nonlinear model approximation capabilities [16, 17, 20, 30-32].

3.1.2. The Neural Network Identification Model

The proposed NN model identification scheme is based on the series-parallel structure shown in Figure 9 where the system is in parallel with the NN identification model (in the dashed box) and *TDL* denotes tapped delay line memory. The NN identification model consists of a training algorithm and the NN model. The inputs to the NN model, via an *m*-TDL and *n*-TDL, are the past *m*-inputs and *n*-outputs contained in $\varphi(k)$ which is obtained from Z^N defined in (38).

The NN identification model of Figure 10 has a recurrent architecture since it contains feedbacks form the output of the system rather than the output of the NN model [17]. This is the so-called teacher forcing method where the network output is forced to follow the system outputs which leads to faster training in real-time [17]. With this method, the NNARMAX model can be trained as a FNN.

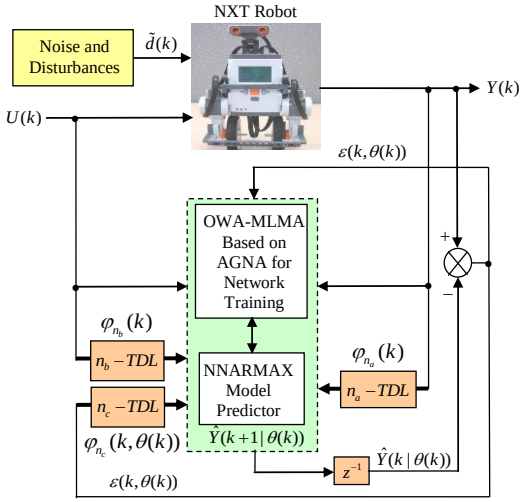


Figure 9. *The NXT robot NNARMAX model identification scheme using OWA-MLMA training algorithm.*

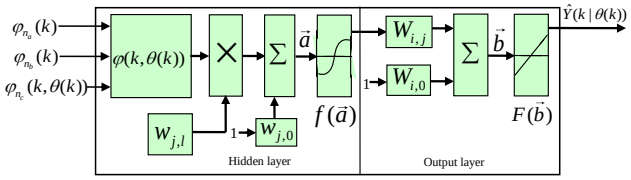


Figure 10. *Architecture of the dynamic feedforward NN (DFNN) model.*

The NNARMAX model architecture proposed here is a MLP NN with two-layers (one hidden and one output layer) shown in Figure 10. To obtain the mathematical description of Figure 10, $\varphi(k)$ in (42) is decomposed into the past input, past output and past prediction error parts $\varphi_{n_b}(k) = [u(k-d),\ldots,u(k-d-m)]^T$, $\varphi_{n_a}(k) = [y(k-1),\ldots y(k-n)]^T$ and $\varphi_{n_a}(k) = [\varepsilon(k-1,\theta(k)),\ldots,\varepsilon(k-n,\theta(k))$ respectively. The outputs of the NN model of Figure 10 can be expressed in terms of the network parameters as:

$$\hat{y}(k \mid \theta(k)) = F_i\left(\sum_{j=1}^{m+n} W_{i,j} f_j(\vec{a}) + W_{i,0}\right) \tag{44}$$

where

$$\vec{a} = \sum_{l_m=1}^{m} w_{j,l_m} \varphi_{l_m}(k) + \sum_{i_n=1}^{n} w_{j,i_n} \varphi_{i_n}(k-1) + w_{j,0}$$

j is the number of hidden neurons; (w_{j,l_m} and w_{j,i_n}) and $W_{i,j}$ are the hidden and output weights respectively; $w_{j,0}$ and $W_{i,0}$ are the hidden and output biases; $F_i(\vec{b})$ is a linear activation function for the output layer and $f_j(\vec{a})$ is an hyperbolic tangent activation function for the hidden layer given here as:

$$f_j(\vec{a}) = 1 - \frac{2}{e^{2\cdot\vec{a}} + 1} \tag{45}$$

Bias is interpreted as a weight acting on the input clamped to 1. Here, the network weights and biases constitute $\theta(k)$.

Regularization can reduce the modeling errors inherent in dynamical systems modeling while improving the robustness and performance of second-order training algorithm for finite data set Z^N [27-29]. One common method of implementing regularization is by weight decay [17, 27, 30-32]. According to this method, the cost function (43) is augmented with a weight decay term defined as $(1/2N)\theta^T(k)D\theta(k)$; where $D = \alpha_d I[\alpha_h \quad \alpha_o]$, I is an identity matrix, α_h and α_o are the weight decay values for the input-to-hidden and hidden-to-output layers respectively. Using (44), the regularized criterion from (43) becomes:

$$\left.\begin{aligned}\vec{J}(Z^N, \varphi(k), \theta(k)) = \\ \frac{1}{2N}\left(\sum_{k=1}^{N}[\varepsilon(k,\theta(k))]^2 + \theta^T(k)D\theta(k)\right)\end{aligned}\right\} \tag{46}$$

and the minimization problem (42) is re-written as

$$\hat{\theta}(k) = \arg\min_{\theta} \vec{J}(Z^N, \varphi(k), \theta(k)) \tag{47}$$

Because NN training is a data-driven method, it should be noted that there are several ways on how Z^N can be presented to the training algorithm at each time step. The most popular way being the batch (off-line) mode where all the data set Z^N is evaluated at each time step and the recursive (online) mode where each data pair of Z^N is evaluated at each time step.

This paper uses a different approach based on a first-in first out sliding stack OWA which store a short history of the input-output data. The stack discards the oldest data as new input-output data pair are progressively added to Z^N at each sampling time step and all the data are evaluated in batch mode at each sampling time step.

3.1.3. Approximate Gauss-Newton Algorithm (AGNA)

The minimization of (47) is based on an iterative procedure which starts with a randomly initial $\theta(k)$ and updates $\hat{\theta}(k)$ iteratively according to the following typical updating rule:

$$\theta_{\tau+1}(k) = \theta_{\tau}(k) + \Delta\theta_{\tau}(k) \tag{48}$$

where $\theta_{\tau}(k)$ denotes $\theta(k)$ at the current iteration τ, $\Delta\theta_{\tau}(k)$ is the searching direction, $\theta_{\tau+1}(k)$ is the global minimum and $\hat{\theta}(k) = \theta_{\tau+1}(k)$ if certain stopping criteria are satisfied.

The most commonly used method for updating $\theta_{\tau+1}(k)$ is the back-propagation algorithm (BPA) [33, 34]. This algorithm uses the gradient method [28, 35-37] and set the $\Delta\theta_{\tau}(k)$ directly proportional to the negative of the gradient of (46) evaluated at $\theta(k) = \theta_{\tau}(k)$. Using (46) and (48), the basic BP algorithm can be stated as:

$$\theta_{\tau+1}(k) = \theta_{\tau}(k) - \mu_{\tau}\nabla\vec{J}(Z^N, \varphi(k), \theta(k))\,|_{\theta(k)=\theta_{\tau}(k)} \tag{49}$$

where μ_{τ} is the step size and ∇ is the gradient NN training using the BPA has been reported to be characterized by poor performance [28, 29, 36]. To improve the performance of the BP algorithm, the Gauss-Newton method has been widely used as a starting point for this purpose.

The AGNA uses the linear approximation error $\tilde{\varepsilon}_{\tau}(k,\theta(k))$ to the error $\varepsilon(k,\theta(k))$ in (45) expressed as:

$$\left.\begin{aligned}\tilde{\varepsilon}_{\tau}(k,\theta(k)) &= \varepsilon[k,\theta_{\tau}(k)] \\ &\quad + [\theta(k) - \theta_{\tau}(k)]^T \nabla[\varepsilon(k,\theta(k))]\big|_{\theta(k)=\theta_{\tau}(k)} \\ &= \varepsilon[k,\theta_{\tau}(k)] \\ &\quad - [\theta(k) - \theta_{\tau}(k)]^T \nabla[\hat{y}(k \mid \theta(k))]\big|^T_{\theta(k)=\theta_{\tau}(k)}\end{aligned}\right\} \tag{50}$$

to obtain (46) as a quadratic criterion $\tilde{J}_{\tau}(\theta(k))$ given as

$$\tilde{J}_{\tau}(\theta(k)) = \frac{1}{2N}\left(\sum_{k=1}^{N}[\tilde{\varepsilon}_{\tau}(k,\theta(k))]^2 + \theta^T(k)D\theta(k)\right) \tag{51}$$

where $\tilde{J}_{\tau}(\theta(k)) = J(Z^N, \varphi(k), \theta(k))$ from now on shall be used for convenience. Note that Equation (17) can also be expressed as [15-17]:

$$\left.\begin{aligned}\tilde{J}_\tau(\theta(k)) &= \tilde{J}_\tau(\theta_\tau(k)) + G[\theta_\tau(k)]^T(\theta(k)-\theta_\tau(k)) \\ &+ \frac{1}{2}(\theta(k)-\theta_\tau(k))^T R[\theta_\tau(k)](\theta(k)-\theta_\tau(k))\end{aligned}\right\} \quad (52)$$

where

$$G[\theta_\tau(k)] = -\frac{1}{N}\left(\sum_{k=1}^{N}\psi(k,\theta_\tau(k))\varepsilon(k,\theta_\tau(k)) + D\theta_\tau(k)\right) \quad (53)$$

and

$$R[\theta_\tau(k)] = \frac{1}{N}\left(\sum_{k=1}^{N}\psi(k,\theta_\tau(k))^T\psi(k,\theta_\tau(k)) + D\right) \quad (54)$$

where $G[\theta_\tau(k)]$ and $R[\theta_\tau(k)]$ are the gradient and exact Gauss-Newton Hessian matrices respectively, and $\psi(k,\theta_\tau(k)) = \nabla\hat{y}(k\,|\,\theta(k))|_{\theta(k)=\theta_\tau(k)}$ denotes the derivative of the network output with respect to $\theta(k)$ evaluated at $\theta(k)=\theta_\tau(k)$. By substituting $\Delta\theta_\tau(k)=\theta(k)-\theta_\tau(k)$ into (52) and setting its derivative to zero as follows:

$$\frac{d\tilde{J}_\tau(\theta(k))}{d\theta(k)} = G[\theta_\tau(k)] + R[\theta_\tau(k)]\Delta\theta_\tau(k) = 0$$

we obtain the AGNA searching direction given as

$$\Delta\theta_\tau(k) = -R[\theta_\tau(k)]^{-1}G[\theta_\tau(k)] \quad (55)$$

Using (49), the AGNA can be stated as:

$$\theta_{\tau+1}(k) = \theta_\tau(k) + \Delta(\theta_\tau(k)) \quad (56)$$

where $\mu_\tau = 1$. Equation (54) is guaranteed to be positive definite since it is based a second-order approximation of (47) which is well-known to be positive definite [35]. The major problems with the Gauss-Newton method are slow convergence due to μ_τ especially when $\theta_{\tau+1}(k)$ is far from $\theta_\tau(k)$ and (53) can sometimes be ill-conditioned. In the next sub-section, we utilize (56) to formulate the proposed OWA-MLMA based on AGNA which addresses the above issues dynamically.

3.1.4. The OWA-MLMA Based on AGNA

The Levenberg-Marquardt's modification to (55) is the introduction of a nonnegative parameter λ_τ to the diagonal of $R[\theta_\tau(k)]$ with a new updating rule given as [35-37]:

$$\theta_{\tau+1}(k) = \theta_\tau(k) - \Delta\theta_\tau(k) \quad (57)$$

where

$$\Delta\theta_\tau = -[R[\hat{\theta}_\tau(k)] + \lambda_\tau I]^{-1}G[\theta_\tau(k)] \quad (58)$$

is the searching direction and I is a diagonal matrix.

The parameter λ_τ characterizes a hybrid of searching directions and has several effects [35-37]: *1).* for large values of λ_τ, Equation (57) reduces to the steepest descent algorithm (with step $1/\lambda_\tau$) which requires a descend search method; and *2).* for small values of λ_τ, Equation (57) reduces to Gauss-Newton algorithm (with step given by (56)).

The convergence of (57) may be slowed if the magnitude of $\hat{\theta}_{\tau+1}(k)$ is large [28, 29, 36]. Here, we modify (58) by introducing a scaling matrix S_τ, so that (58) becomes

$$\Delta\theta_\tau(k) = -[R[\theta_\tau(k)] + (S_\tau)^T\lambda_\tau S_\tau]^{-1}G[\theta_\tau(k)] \quad (59)$$

where the scaling matrix $S_\tau = sI$ is adjusted simultaneously with $\theta_\tau(k)$ in the search for $\theta_{\tau+1}(k)$ by (57), s is the scaling parameter, and I is an identity. The solution to (57) using (59) based on the trust region method [35, 36] can then be stated as a constrained minimization problem defined by:

$$\theta_{\tau+1}(k) = \arg\min_{\theta}\tilde{J}_\tau(\theta(k)) \quad (60)$$

subject to

$$\left|S_\tau(\theta_{\tau+1}(k)-\theta_\tau(k))\right| \le \delta_\tau .$$

where δ_τ is the trust region radius within which $\theta_{\tau+1}(k)$ can be found. A difficulty with the Levenberg-Marquardt method is in selecting and adjusting the parameter λ_τ as well as how $\theta_{\tau+1}(k)$ should be updated. The choice for the proper selection of λ_τ has led to the formulation of several algorithms [28, 29, 35-37].

The approach in this paper builds on the indirect method described in [37] which has been shown in [15-17] to outperform the original Levenberg-Marquardt algorithm [38, 39]. According to this method λ_τ is adjusted based on the ratio α_τ between the actual reduction $(ared)$ and theoretical predicted decrease $(pdec)$ subject to the constraint in (60) stated here as:

$$\alpha_\tau = \frac{ared}{pdec} = \frac{\vec{J}(\theta_\tau(k)) - \vec{J}(\theta_\tau(k)+\Delta\theta_\tau(k))}{\vec{J}(\theta_\tau(k)) - \tilde{J}_\tau(\theta_\tau(k)+\Delta\theta_\tau(k))} \quad (61)$$

where $\vec{J}(\theta_\tau(k)) = \vec{J}(Z^N,\varphi(k),\theta(k))$ in (46) for convenience.

Thus, we summarize the OWA-MLMA based on AGNA

for training NN in a stepwise procedure as follows:

1) Specify $\tau, \tau_{\max}, D,\ \lambda_{\max} \in [1,10^3]$ and m and n for $\varphi(k)$.
2) Initialize
   $\lambda_\tau \in [0.1,10^{-2}]$, $\delta_\tau \in [0.1,10^{-4}]$, $\delta_\tau \in [0.1,10^{-4}]$
   $s \in [0.1,10^{-2}]$, $\theta(k)$ and evaluate $\vec{J}(Z^N, \varphi(k), \theta(k))$ in (46).
3) While $\tau = 1$, compute $G[\theta_\tau(k)]$ and $\Delta\theta_\tau(k)$ using (53) and (58) respectively.
4) Evaluate the ratio α_τ in (61).
5) Update λ_τ according to the following conditions on α_τ:

If $\alpha_\tau > 0.75$, then $\lambda_\tau \leftarrow 0.5 * \lambda_\tau$ and go to *Step6).*

If $\alpha_\tau < 0.25$, then $\lambda_\tau \leftarrow 2 * \lambda_\tau$ and go to *Step 6).*

6) If *ared* > 0, then $\theta_{\tau+1}(k) \leftarrow \theta_\tau(k) + \Delta\theta_\tau(k)$ subject to (53)

Set $\tau \leftarrow \tau + 1$ and $\lambda_{\tau+1} \leftarrow \lambda_\tau$.

7) If $\left|S_\tau(\theta_{\tau+1}(k) - \theta_\tau(k))\right| > \delta_\tau$ or $\tau > \tau_{\max}$ or $\lambda_\tau > \lambda_{\max}$ (number of iterations); go to *Step 8)*, else go to *Step 3).*
8) Accept $\hat{\theta}(k) \leftarrow \theta_{\tau+1}(k)$ and terminate.

3.2. The Incremental or Online Back-Propagation Algorithm (INCBPA)

In order to investigate the performance of the OWA-MLMA based on AGNA, the so-called incremental back-propagation algorithm (INCBPA) is used for this purpose. The INCBPA (or online recursive version of the BPA) was originally proposed by [40] which has been modified in [17] and used in this paper. The incremental back-propagation (INCBP) algorithm is given as:

$$\mu_\tau I = \frac{1}{k}\sum_{\iota=1}^{k} R[\iota, \theta_\tau(k)]^{-1} \tag{62}$$

where μ is the step size and I is an identity matrix of appropriate dimension. Next, the basic back-propagation given as [17, 33, 34]:

$$\hat{\theta}(k) = \theta_\tau(k) - \mu_\tau \frac{dJ[\theta(k)]}{d\theta(k)}\bigg|_{\theta(k)=\theta_\tau(k)} \tag{63}$$

is used to update the algorithm of (62). Finally, all that is required is to specify a suitable step size μ and carry out the recursive computation of the gradient given by (63).

4. Formulation of the NAPC Based on FNNO-MLMA

4.1. NAPC Based on FNNO-MLMA

4.1.1. Formulation of the MPC Control Problem and the MPC Strategy

The block diagram of the NAPC based on FNNO-MLMA control with OWA-MLMA based on AGNA is shown in Figure 11; where $R'(k)$, $E(k)$, $U(k)$, $Y(k)$, $\hat{Y}(k)$, $\hat{Y}(k+\eta\,|\,k)$, $\tilde{d}(k)$ and $z^{-\eta}$ are the desired reference signal, prediction error, control input, system output, η step-delay prediction model output, η step-ahead predicted output, noise/input disturbances and ηstep-delay operator respectively and k is the number of samples based on the new measurement data sample.

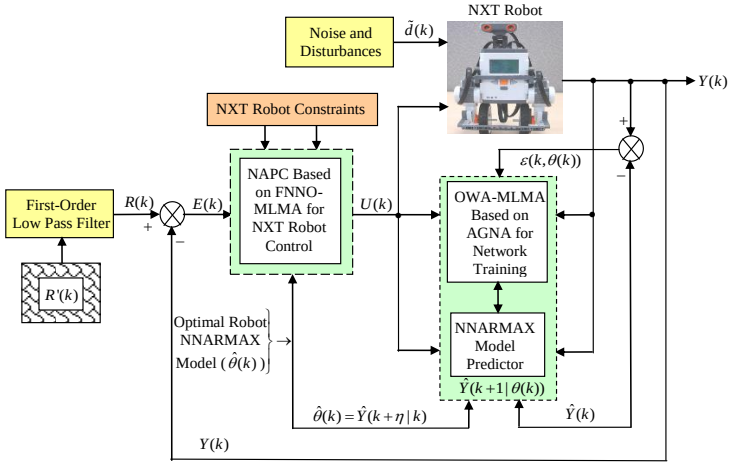


Figure 11. *The NAPC-based FNNO-MLMA control scheme for the NXT robot control with the OWA-MLMA based on AGNA for training the NNARMAX model predictor.*

As in basic MBPC scheme [16, 17, 20], the prediction errors *E (k)* between the process model and the prediction model are compensated by filtering the reference signal using a first-order low-pass digital filter defined here as:

$$R(k) = \frac{B_m}{A_m} R'(k) \tag{64}$$

where $R'(k)$ and $R(k)$ are the desired reference and the filtered reference signals respectively; A_m and B_m are the denominator and numerator polynomials of the filter. In this way, the NAPC based on FNNO-MLMA is deigned, in part, based on the filter tracking error capability; where A_m and B_m serves as tuning parameters used to improve the robustness and internal stability of the control algorithm respectively.

Assuming that at Assuming that the identified NNARMAX model is stable, proper and deterministic; then the NAPC based on FNNO-MLMA uses the linearized model parameters of the identified nonlinear NNARMAX model to accurately predict the current system output $\hat{Y}(k)$ at that same sample time instant *k*.

At time $k + N_u - 1$, the NAPC based on FNNO-MLMA calculates a sequence of control inputs $U(k + N_u - 1 | k)$ consisting of the current $U(k | k)$ and future inputs $U(N_u - 1 | k)$. The current input $U(k) = U(k | k)$ is held constant after N_u control moves; where N_u is the maximum control horizon. The input $U(k)$ is calculated such that a set of $\hat{Y}(k + \eta | k)$ approaches the desired reference signal in an optimal manner over a specified prediction horizon $\eta \in [N_d, N_p]$; where N_d and N_p are the minimum and maximum prediction horizons respectively.

The predicted values are used to calculate the control signals by minimizing an objective function of the form:

$$\left.\begin{aligned} \hat{J}(U(k)) = &\left[R(k) - \hat{Y}(k)\right]^T \kappa \left[R(k) - \hat{Y}(k)\right] \\ &+ \tilde{U}^T(k) \rho \tilde{U}(k) \end{aligned}\right\} \tag{65}$$

subject to the constraints

$$\left.\begin{aligned} &\Delta U(k + \eta) = 0, \qquad N_u \le \eta \le N_p - N_d \\ &\Delta U_{\min} \le \Delta U(k) \le \Delta U_{\max}, \quad Y_{\min} \le Y(k) \le Y_{\max} \end{aligned}\right\} \tag{66}$$

where

$$\left.\begin{aligned} R(k) &\triangleq [R(k + N_d) \ldots R(k + N_p)]^T \\ \hat{Y}(k) &\triangleq [\hat{Y}(k + N_d | k) \ldots \hat{Y}(k + N_p | k)]^T \\ E(k) &= \left[R(k) - \hat{Y}(k)\right] \\ &\triangleq [E(k + N_d | k) \ldots E(k + N_p | k)]^T \\ \tilde{U}(k) &\triangleq [\Delta U(k) \ldots \Delta U(k + N_u - N_d)]^T \end{aligned}\right\} \tag{67}$$

where ΔU is the change in control signal; κ and ρ are two weighting matrices penalizing changes on $\hat{Y}(k)$ and $U(k)$ in Equation (65).

4.1.2. The FNNO-MLMA for Nonlinear Adaptive Predictive Control

This section focuses on the development of the NAPC based on the FNNO-MLMA. Here we assume that $\varphi(k)$ and NN model $\hat{\theta}(k)$ approximates the system (37), $N_d = d = 1$ and that the system information is available up to time $k - 1$, so that the one step-ahead predictor of (37) at time k can be expressed as:

$$\left.\begin{aligned} \hat{Y}(k) = J(&Y(k-1), \ldots, Y(k-n), U(k-d), \\ &\ldots, U(k-d-m)) \end{aligned}\right\} \tag{68}$$

So that the η step-ahead model predictor from (68) becomes:

$$\left.\begin{aligned} \hat{Y}(k + \eta | k) = J(&\hat{Y}(k + \eta - 1), \ldots, \hat{Y}(k + \eta - \\ &\min(\eta, n)), Y(k-1), \ldots, Y(k - \max(n - \eta, 0)), \\ &U(k + \eta - d), \ldots, U(k + \eta - d - m)) \end{aligned}\right\} \tag{69}$$

The η step-ahead predicted output of (69) can be expressed in terms of nonlinear NN model parameters from (44) as:

$$\hat{Y}(k + \eta | k) = \sum_{j=1}^{n_h} W_{i,j} f_j\left(\tilde{a}(\eta, j)\right) + W_{i,0} \tag{70}$$

where $f_j(\bullet)$ is given by (11) but here $\vec{a} = \tilde{a}(\eta, j)$ which is given as:

$$\left.\begin{aligned}\tilde{a}(\eta, j) = &\sum_{\tau=1}^{\min(\eta-d,n)} w_{j,\tau}\hat{Y}(k+\eta-\tau)\\ &+\sum_{\tau=\min(\eta,n)+1}^{n} w_{j,\tau}Y(k+\eta-\tau)\\ &+\sum_{\tau=0}^{m} w_{j,n+1+\tau}U(k+\eta-d-\tau)+w_{j,0}\end{aligned}\right\} \quad (71)$$

4.1.3. Computing the Past and Future Control Signals

The NAPC based on FNNO-MLMA proposed here computes the optimal future control signal based on nonlinear optimization using the nonlinear NNARMAX model developed in the previous section. The minimization of (65) subject to the constraints defined by (66) and (67) can be expressed as:

$$\tilde{U}(k) = \arg\min_{\tilde{U}(k)\in U(k)} J(U(k)) \quad (72)$$

The NN-based NAPC based on FNNO-MLMA proposed for solving (72) is based on the FNNO method [35, 37] and the Levenberg-Marquardt method [41, 42] with the following dynamic adaptive rule:

$$\tilde{U}(k) = U^{(\tau)}(k) + \zeta^{(\tau)}(k) \quad (73)$$

which is applied for updating the sequence of future optimal control signal $\tilde{U}(k)$; where $U^{(\tau)}(k)$ is the current iterate of the control sequence; and $\zeta^{(\tau)}(k)$ the search direction given by the following expression:

$$\zeta^{(\tau)} = -(H[U^{(\tau)}(k)] + \lambda I)^{-1} G[U^{(\tau)}(k)] \quad (74)$$

where λ is the adaptation parameter, I is a diagonal matrix; $H[U^{(\tau)}(k)]$ and $G[U^{(\tau)}(k)]$ are the Hessian and Jacobian matrices given respectively by (75) and (76) as follows:

$$\left.\begin{aligned}G[U^{(\tau)}(k)] = &\left.\frac{\partial J(k,U(k))}{\partial U(k)}\right|_{U(k)=U^{(\tau)}(k)} =\\ &-2\kappa\Phi(k)[U^{(\tau)}(k)]\tilde{E}(k) + 2\rho\frac{\partial\tilde{U}(k)}{\partial U(k)}\tilde{U}(k)\Big|_{U(k)=U^{(\tau)}(k)}\end{aligned}\right\} \quad (75)$$

$$\left.\begin{aligned}H\left[U^{(\tau)}(k)\right] = &\left.\frac{\partial^2 J(k,\tilde{U}(k))}{\partial U^2(k)}\right|_{U(k)=U^{(\tau)}(k)} =\\ &\kappa\frac{\partial}{\partial U(k)}\left(\frac{\partial\hat{Y}^T(k)}{\partial U(k)}E(k)\right) + 2\rho\frac{\partial\tilde{U}^T(k)}{\partial U(k)}\frac{\partial\tilde{U}(k)}{\partial U(k)}\Big|_{U(k)=U^{(\tau)}(k)}\end{aligned}\right\} \quad (76)$$

where $\Phi(k)$ in (75) is the partial derivatives of (71).

To simplify the computation of $\Phi(k)$ in (75), the control signal is decomposed into the past and future control signals as $\hat{a}(\eta, j)$ given in (77) as follows:

$$\left.\begin{aligned}\hat{a}(\eta, j) = &\sum_{\tau=1}^{\min(\eta-d,n)} w_{j,\tau}\hat{Y}(k+\eta-\tau)\\ &+\sum_{\tau=1}^{\min(\eta-d-N_u+2,m+1)} w_{j,n+\tau}U(k+N_u-1)\\ &+\sum_{\tau=\eta-d-N_u+2}^{\min(\eta-d,m)} w_{j,n+1+\tau}U(k-d+\eta-\tau)\\ &+\sum_{\tau=\eta-d+1}^{\min(\eta,n)} w_{j,\tau}\hat{Y}(k+\eta-\tau)\\ &+\sum_{\tau=\eta+1}^{n} w_{j,\tau}Y(k+\eta-\tau)\\ &+\sum_{\tau=\eta-d+1}^{m} w_{j,n+1+\tau}Y(k-d+\eta-\tau)+w_{j,0}\end{aligned}\right\} \quad (77)$$

The first three sums of (77) depend on future control signals while the last three sums depend on past control signals.

Since the past control signals does not contribute to the output predictions because new measurements will be available at the next sample time $k+1$. Thus, $\Phi(k)$ is computed here based only on the future control signals $\forall\ \eta\in[N_d, N_p]$ and $\forall\ l\in[0,\min(\eta-1, N_u-1)]$ as:

$$\Phi(k) = \frac{\partial\hat{Y}(k+\eta)}{\partial U(k+l)} = \sum_{j=1}^{n_h} W_j f'[\hat{a}(\eta, j)]\cdot\hat{b}_l(\eta, j)$$

where $\hat{b}_l(\eta, j)$ given in (78) is computed from the first three sum of $\hat{a}(\eta, j)$ in (77).

$$\left.\begin{aligned}\hat{b}_l(\eta, j) = &\sum_{\tau=1}^{\min(\eta-d,n)} w_{j,\tau}\frac{\partial\hat{Y}(k+\eta-\tau)}{\partial U(k+l)}\\ &+\sum_{\tau=1}^{\min(\eta-d-N_u+2,m+1)} w_{j,n+\tau}\frac{\partial U(k+N_u-\tau)}{\partial U(k+l)}\\ &+\sum_{\tau=\eta-d-N_u+2}^{\min(\eta-d,m)} w_{j,n+1+\tau}\frac{\partial U(k-d+\eta-\tau)}{\partial U(k+l)}\end{aligned}\right\} \quad (78)$$

The second-order derivative for all $\eta\in[N_d, N_p]$, for all $p\in[0,l]$, and for all $l\in[0,\min(\eta-1, N_u-1)]$ is calculated as follows:

$$\frac{\partial^2 \hat{Y}(k+\eta)}{\partial U(k+l)\partial U(k+p)} = \sum_{j=1}^{n_h} W_j \left(b_1 + b_2 \frac{\partial \hat{b}_{lp}(\eta, j)}{\partial U(k+p)} \right)$$

where $\hat{b}_p(\eta, j)$ given in (79) is again computed from the first three sum of $\hat{a}(\eta, j)$ in (77)

$$\hat{b}_p(\eta, j) = \left\{ \begin{aligned} &\sum_{\tau=1}^{\min(\eta-d,n)} w_{j,\tau} \frac{\partial \hat{Y}(k+\eta-\tau)}{\partial U(k+p)} \\ &+ \sum_{\tau=1}^{\min(\eta-d-N_u+2,m+1)} w_{j,n+\tau} \frac{\partial U(k+N_u-\tau)}{\partial U(k+p)} \\ &+ \sum_{\tau=\eta-d-N_u+2}^{\min(\eta-d,m)} w_{j,n+1+\tau} \frac{\partial U(k-d+\eta-\tau)}{\partial U(k+p)} \end{aligned} \right\} \quad (79)$$

where

$$b_1 = f''(\hat{a}(\eta, j)) \cdot \hat{b}_l(\eta, j) \cdot \hat{b}_p(\eta, j),$$

$$b_2 = f'(\hat{a}(\eta, j)),$$

$$\frac{\partial \hat{b}_{lp}(\eta, j)}{\partial u(k+p)} = \sum_{\tau=1}^{\min(\eta-d-l,n)} w_{j,\tau} \frac{\partial^2 \hat{y}(k+\eta-\tau)}{\partial u(k+l)\partial u(k+p)}$$

and f' and f'' are the first and second derivatives of (45).

4.1.4. Computing the Optimal Control Signal

The well-known problem with the Newton method is that the Hessian is not guaranteed to be positive definite in the open neighborhood of a global minimum. Thus, it is necessary to check that the Hessian is positive definite before updating the optimization. This implies checking for the definiteness of the Hessian matrix using:

$$V^{(\tau)}(k) = H[U^{(\tau)}(k)] + \lambda^{(\tau)} I \quad (80)$$

Table 2. *Iterative Algorithm for Selecting λ for Guaranteed Positive Definiteness of the Full Gauss-Newton Hessian Matrix.*

Initialize: $\lambda a = 0.5$, $\lambda b = 1$, $\lambda c = 2$, $\lambda d = 4$, $\lambda e = 6$, $km = [\lambda a, \lambda b, \lambda c, \lambda d, \lambda e]$, $kl = length(km)$, $iter = 0$, $sm = size(V^{(\tau)}(k))$, $L_{a,a}^{(\tau)}(k) = -1$, $p = 1$ to $(N_u)^2$ in step of $(N_u + 1)$. Compute (80)

for $sn = 1$ to sm

while $iter < kl$ or $L_{a,a}^{(\tau)} < 0$, do

for $kn = 1$ to kl, do

for $a = 1$ to p, do

$$L_{a,a}(k) = \left[V_{a,a}^{(\tau)}(k) - \sum_{j=1}^{a-1} L_{a,j}^2(k) \right]^{1/2}$$

if $L_{a,a}^{(\tau)} < 0$, break, end if $L_{a,a}$

for $b = a + 1$ to p, do

$$L_{b,a}(k) = \frac{1}{L_{a,a}} \left[V_{b,a}^{(\tau)}(k) - \sum_{j=1}^{b-1} (L_{b,j}(k) \cdot L_{a,j}(k)) \right]^{1/2}$$

end for *b*, end for a

if $L_{a,a}^{(\tau)} < 0$, $\lambda = \lambda \cdot km(1, kn)$, Re-compute (80)

for $a = 1$ to p, do

$$L_{a,a}(k) = \left[V_{a,a}^{(\tau)}(k) - \sum_{j=1}^{a-1} L_{a,j}^2(k) \right]^{1/2}$$

end for *a*

else

for $a = 1$ to p, do

$$L_{a,a}(k)=\left[V_{a,a}^{(\tau)}(k)-\sum_{j=1}^{a-1}L_{a,j}^{2}(k)\right]^{1/2}$$

for $b=a+1$ to p, do

$$L_{b,a}(k)=\frac{1}{L_{a,a}}\left[V_{b,a}^{(\tau)}(k)-\sum_{j=1}^{b-1}(L_{b,j}(k)\cdot L_{a,j}(k))\right]^{1/2}$$

end for *b*, end for *a*, end for *a*, end if $L_{a,a}$, end for *kn*

$iter=iter+1$

if $iter>kl$ and $L_{a,a}(k)<0$, break, end

end while $iter$, end for sn

to obtain a value of λ that will satisfy this condition. Again, many algorithms have been proposed for this purpose [23, 28, 29, 36].

This paper proposes the adaptive algorithm given in Table 2 which is based on the Cholesky factorization method which ensures that (80) is always positive definite and none ill-conditioned. The algorithm first compute (80) and check for positive definiteness. If this condition is not satisfied, the algorithm iteratively selects new $\lambda^{(\tau)}$ and re-computes (80) and then terminates immediately positive definite of (80) is achieved.

Having satisfied the positive definiteness of (80), it is necessary to determine the optimal control signal as a global minimum. Again, the widely used method is the trust region approach mentioned earlier [37]. Here, we formulate the nonlinear minimization problem around the trust region as follows:

$$\tilde{U}(k)=\underset{\tilde{U}(k)\in U(k)}{\arg\min}\ \hat{J}(U(k)) \tag{81}$$

subject to

$$\left|U^{(\tau+1)}(k)-U^{(\tau)}(k)\right|\le\delta^{(\tau)} \tag{82}$$

where $\delta^{(\tau)}$ is the trust region radius where $\tilde{U}(k)$ can be trusted.

The last issue to be addressed is related to how $\tilde{U}(k)$ in (81) is to be updated and the step size at the next iteration. Although, many algorithms have been proposed for this purpose [35-37], but here we reuse the $\lambda^{(\tau)}$ obtained from Table 2 for this purpose. Note that $\lambda^{(\tau)}$ characterizes a hybrid adaptation parameter and has several effects as discussed in Section 3.1.3. Unlike in Section 3.1.3 where the approximated $R[\theta_\tau(k)]$ is guaranteed to be positive definite, here $H[U^{(\tau)}(k)]$ is not guaranteed to be non-positive definite or ill-conditioned and the algorithm of Table 2 is used.

Thus, we reuse the $\lambda^{(\tau)}$ obtained in Table 2 and apply the indirect method discussed in Section 3.1.4 to $\lambda^{(\tau)}$ is adjusted according to the ratio ϖ_τ between the actual reduction (J_{ared}) and theoretical predicted decrease (J_{pdec}) as defined below:

$$\varpi^{(\tau)}=\frac{J_{ared}}{J_{pdec}}=\frac{\hat{J}(U^{(\tau)}(k))-\hat{J}(U^{(\tau)}(k)+\zeta^{(\tau)})}{\lambda^{(\tau)}(\zeta^{(\tau)})^{T}(\zeta^{(\tau)})-(\zeta^{(\tau)})^{T}G(U^{(\tau)}(k))} \tag{83}$$

Thus, we summarize the NAPC algorithm based on FNNO-MLMA in the following stepwise procedures:

1) Specify initial sequence of future control inputs $\tilde{U}^{(0)}(k)$ and evaluate $\hat{J}(U(k))$ using (65) subject to (66). Initialize $\lambda^{(\tau)}$, $\delta^{(\tau)}$ and set $\tau=0$. while $\tau=1$.
2) Compute $G[U^{(\tau)}(k)]$ in (75) and $H[U^{(\tau)}(k)]$ in (76).
3) Compute the Cholesky factorization of the $V^{(\tau)}(k)$ in (80) using the algorithm of Table 2.
4) Determine the search direction $\zeta^{(\tau)}$ using (77).
5) Evaluate $\hat{J}(k,U^{(\tau)}+\zeta^{(\tau)})$ and compute ϖ_τ using (83).
6) Update λ according to the conditions on ϖ_τ:

If $\varpi^{(\tau)}>0.75$, then $\lambda\leftarrow 0.5*\lambda$ and go to *Step 7)*.

If $\varpi^{(\tau)}<0.25$, then $\lambda\leftarrow 2*\lambda$ and go to *Step 7)*.

7) If $\varpi^{(\tau)}>0$, update $U^{(\tau+1)}(k)\leftarrow U^{(\tau)}(k)+\zeta^{(\tau)}(k)$, and set $\tau=\tau+1$.
8) If $\left\|U^{(\tau+1)}(k)-U^{(\tau)}(k)\right\|<\delta^{(\tau)}$ in (82) *or* $\tau>u_{iter}$ *or* $\lambda>10^{3}$, go to *Step 9)*, else go to *Step 2)*.

9) Accept the sequence $\tilde{U}(k) = U^{(\tau)}(k)$ in (81) subject to the constraints in (66) and terminate the algorithm.

4.2. Discrete-Time Fixed-Parameter PID Controller Design for the NXT Robot Control

4.2.1. The PID Control Strategy

The constant gain or fixed-parameter proportional-integral-derivative (PID) controllers are widely used in many industries because of their simple structure, robust performance and the ease of their implementation [43, 44].

In the simplest case, the NXT robot affected by the environmental disturbances can be controlled by a discrete-time fixed parameter PID controller used in a closed-loop configuration illustrated in Figure 12. This operation is imitated by placing the NXT robot in the PID control loop.

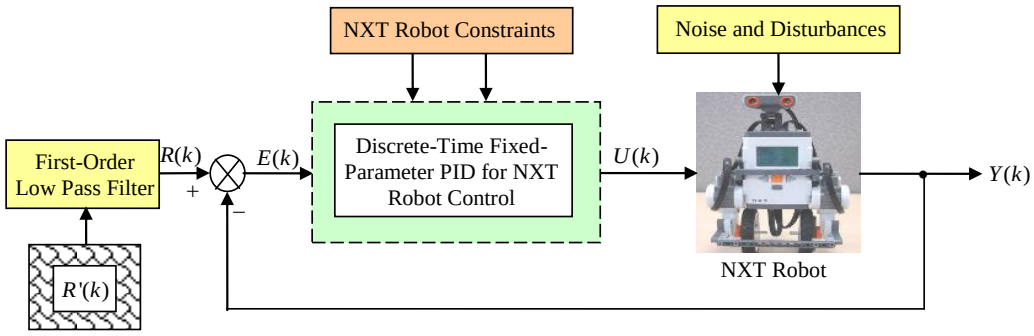


Figure 12. *The discrete-time fixed-parameter PID control scheme.*

The mathematical relationship governing the discrete-time fixed-parameter PID controller that computes the control signal $U(k) = [LWMV(k) \;\; RWMV(k)]$ is defined as follows:

$$\left.\begin{aligned} U(k) = K_P E(k) + K_I \frac{T}{2}\sum_{k=1}^{N}\left[E(k-1)+E(k)\right] \\ + K_D \frac{\left[E(k)-E(k-1)\right]}{T} \end{aligned}\right\} \quad (84)$$

where K_P, K_I and K_D are the proportional, integral and derivative gains respectively, T is the sampling time and $E(k) = R(k) - \hat{Y}(k)$ is the error term defined as the difference between the process $\hat{Y}(k)$, filtered desired reference $R(k)$ and N is the number of simulation samples (see sub-section 4.1.1).

The first, second and third terms in (84) corresponds to the present, past and future control sequence respectively. The minimum and maximum constraints imposed on the PID controller to penalize changes on the NXT robot control inputs U (k) and NXT robot outputsY (k) are defined in (66).

A major problem with PID controller is the "*wind up*" of the integrator resulting in saturation of the integral term for control signal of large magnitude. Rich literatures exist for anti wind-up techniques to overcome this problem [43, 44]. According to this technique, the integrator switches off when the actuator output exceeds a predefined limit (i.e. saturated) subject to the constraints defined in (66) and the update of the integral term is terminated.

4.2.2. The PID Control Law for the Servo Control of the NXT Robot

The starting point for the PID controller design for the NXT robot begins with the servo controller design. The servo controller is used as the preliminary controller for the NXTway-GS robot and where θ is selected as the reference signal for servo control. It is important to note that we cannot use variables other than θ as the reference because the system may become uncontrollable. The block diagram of the preliminary servo controller for NXTway-GS robot is shown in Figure 13.

We can now calculate the feedback (K_f) and integral (K_i) gains by the linear quadratic regulator (LQR) method. To achieve this, we choose the following weight matrix Q and R by experimental trial and error.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 6\times10^{5} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4\times10^{2} \end{bmatrix}, R = \begin{bmatrix} 1\times10^{3} & 0 \\ 0 & 1\times10^{3} \end{bmatrix} \quad (85)$$

where Q (2, 2) is a weight for body pitch angle, and Q (*5*, *5*) is a weight for time integration of difference between measured average angle and referenced one.

Furthermore, the value of the speed gain after LQR calculation needs to be adjusted because it fluctuates excessively

when the NXT robot is in motion. In addition to the gain adjustment, the following control handles are added, namely: *1)*. Rotate NXTway-GS by giving different value to the left and right motor, and *2)*. Use Proportional (*P*) control for wheel synchronization when NXT runs straightforward because the rotation angle of DC motors is not same even though same PWM is applied. Consequently, the NXT robot controller is derived as shown in the block diagram of Figure 14. Here $k_{\dot{\theta}}$, $k_{\dot{\psi}}$ and k_{sync} are the tuning parameters.

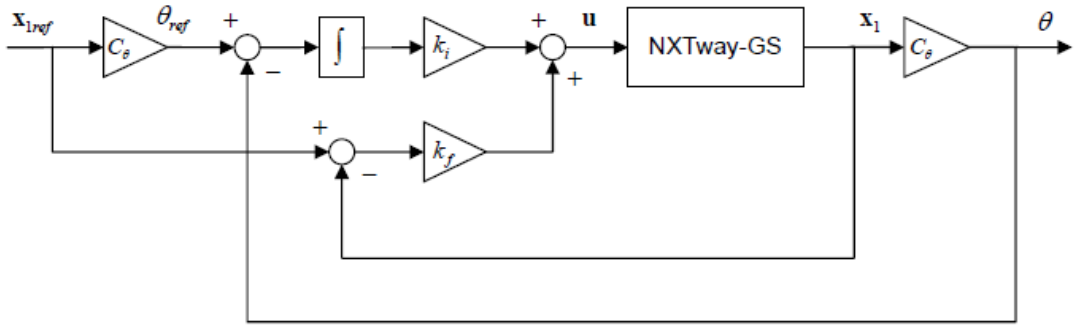


***Figure 13**. The Simulink® model of the NXT robot servo controller.*

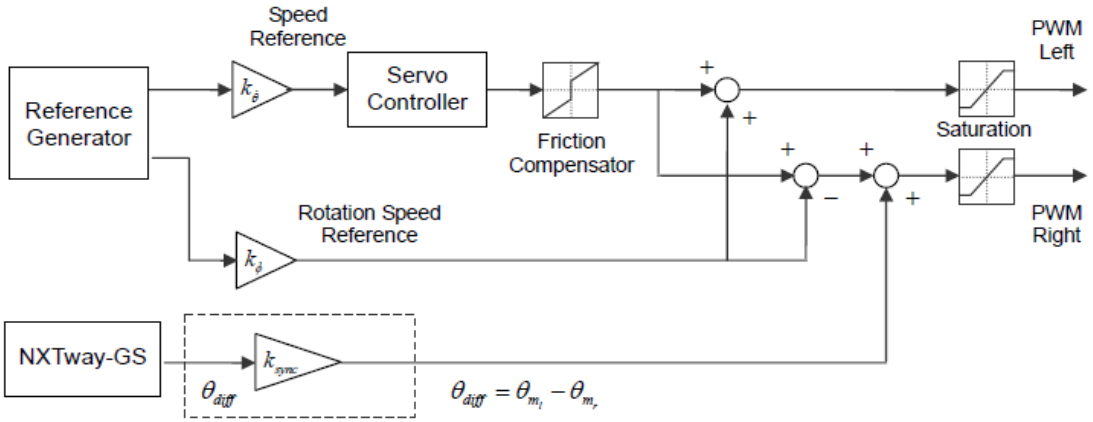


***Figure 14**. The complete Simulink® model of the NXT robot PID controller incorporating the servo controller of Figure 13. Note that the dashed block is a proportional controller for wheel synchronization when NXT robot doe not rotate.*

Servo control is a control technique such that that an output of a system tracks an expected behavior of the desired reference trajectory. PID control and I-PD control in classical control theory are a kind of servo control. It is necessary to add an integrator in the closed loop since it is necessary to track the desired output reference. The block diagram of servo control based on the PID scheme is shown in Figure 13. The gains (controller parameters) of the servo controller can be calculated in the same way as feedback control by considering an expansion system. It has new state that is the difference between the output and referenced value. The servo control gains are derived as the following. The state equation and output function are given as the following:

$$\left.\begin{aligned}\dot{x}(t) &= Ax(t) + B\mathrm{u}(t)\\ y(t) &= Cx(t)\end{aligned}\right\} \tag{86}$$

We use a difference $\mathrm{e}(t) = C(\mathrm{x}(t) - \mathrm{x}_{ref})$, an integral of the difference $\mathrm{z}(t)$, states of the expansion system $\bar{\mathrm{x}}(t) = [\mathrm{x}(t), \mathrm{z}(t)]^T$, and the orders of x and u are *n* and *m* respectively. The state equation of the expansion system is the following:

$$\begin{bmatrix}\dot{\mathrm{x}}(t)\\ \dot{\mathrm{z}}(t)\end{bmatrix} = \begin{bmatrix}A & 0_{n\times m}\\ C & 0_{m\times m}\end{bmatrix}\begin{bmatrix}\mathrm{x}(t)\\ \mathrm{z}(t)\end{bmatrix} + \begin{bmatrix}B\\ 0_{m\times m}\end{bmatrix}\mathrm{u}(t) - \begin{bmatrix}0_{n\times m}\\ I_{m\times m}\end{bmatrix}Cx_{ref} \tag{87}$$

If the expansion system is assumed to be stable, Equation (39) converges to the following expression:

$$\begin{bmatrix} \dot{\mathrm{x}}(\infty) \\ \dot{\mathrm{z}}(\infty) \end{bmatrix} = \begin{bmatrix} A & 0_{n\times m} \\ C & 0_{m\times m} \end{bmatrix} \begin{bmatrix} \mathrm{x}(\infty) \\ \mathrm{z}(\infty) \end{bmatrix} + \begin{bmatrix} B \\ 0_{m\times m} \end{bmatrix} \mathrm{u}(\infty) - \begin{bmatrix} 0_{n\times m} \\ I_{m\times m} \end{bmatrix} C\mathrm{x}_{ref} \tag{88}$$

By subtracting (40) from (39), the following state equation is obtained

$$\left.\begin{aligned} \begin{bmatrix} \dot{\mathrm{x}}_e(t) \\ \dot{\mathrm{z}}_e(t) \end{bmatrix} &= \begin{bmatrix} A & 0_{n\times m} \\ C & 0_{m\times m} \end{bmatrix} \begin{bmatrix} \mathrm{x}_e(t) \\ \mathrm{z}_e(t) \end{bmatrix} + \begin{bmatrix} B \\ 0_{m\times m} \end{bmatrix} \mathrm{u}_e(t) \to \frac{d}{dt}\bar{\mathrm{x}}_e(t) \\ &= \bar{A}\bar{\mathrm{x}}_e(t) + \bar{B}\mathrm{u}_e(t) \end{aligned}\right\} \tag{89}$$

where $\mathrm{x}_e = \mathrm{x}(t) - \mathrm{x}(\infty)$, $\mathrm{z}_e = \mathrm{z}(t) - \mathrm{z}(\infty)$ and $\mathrm{u}_e = \mathrm{u}(t) - \mathrm{u}(\infty)$. We can use feedback control to make the expansion system stable. Thus, the control input is becomes

$$\mathrm{u}_e(t) = -K\bar{\mathrm{x}}_e(t) = -K_f\mathrm{x}_e(t) - K_i\mathrm{z}_e(t) \tag{90}$$

If it is possible that $\mathrm{x}(\infty) \to \mathrm{x}_{ref}$, $\mathrm{z}(\infty) \to 0$ and $\mathrm{u}(\infty) \to 0$ are assumed, the following input $\mathrm{u}(t)$ can be derived as:

$$\mathrm{u}(t) = -K_f\left(\mathrm{x}(t) - \mathrm{x}_{ref}\right) - K_iC\int\left(\mathrm{x}(t) - \mathrm{x}_{ref}\right)dt \tag{91}$$

Several literature on modern control engineering usually describe I-PD type expression (x_{ref} in the first term of (91) equals to zero) as servo control input [4-6]. However, we use discrete-time fixed-parameter PID type expression as (91) to improve the reference tracking performance.

5. NNARMAX Model Identification Using OWA-MLMA Based on AGNA and Adaptive Control Using NAPC Based on FNNO-MLMA of the NXT Robot

5.1. Performance Comparison of the NNARMAX Model Identification of the NXT Robot Model Using OWA-MLMA and INCBPA, Results and Discussions

5.1.1. NNARMAX Model Identification of the NXT Robot Using OWA-MLMA and INCBPA

In this study, the inputs to the NXT robot are the left wheel motor voltage (*LWMV*) and right wheel motor voltage (*RWMV*), that is U = [*LWMVRWMV*]^T. The outputs of the NXT robot are the left wheel rotation angle (*LWRA*) and the right wheel rotation angle (*RWRA*), that is $\hat{y} = [LWRA\ \ RWRA]^T$. The input vector $\varphi(k)$ to the NNARMAX model predictor consists of the regression vectors $\varphi_{n_b}(k) = [LWMV(k)\ RWMV(k)]^T$, $\varphi_{n_a}(k) = [LWRA(k)\ RWRA(k)]^T$ and $\varphi_{n_c}(k,\theta(k)) = [(y_{LWRA}(k,\theta(k)) - \hat{y}_{LWRA}(k,\theta(k)))\ (y_{RWRA}(k,\theta(k)) - \hat{y}_{RWRA}(k,\theta(k)))]^T$ $\hat{y}_{RWRA}(k,\theta(k)))]^T$. The outputs of the NNARMAX model predictor are the predicted values of *LWRA* and *RWRA* given as $\hat{y}(k) = [\hat{y}_{LWRA}(k)\ \ \hat{y}_{RWRA}(k)]^T$.

As discussed in Section 2.2.3, an open loop simulation of the NXT robot MATLAB/Simulink model is performed to obtain 250,000 input-output data pair. The data is split into *N* = 1750,000 training and 75,000 validation data. The training data is scaled to zero mean and unit variance to prevent signals of largest magnitudes from dominating the identified model [wastewater papers [30-32]. Again, the joint weights are rescaled afterwards so that the trained network can work with unscaled test data.

For training and assessing the convergence performance of the network, the following parameters are selected: $p = 2$, $q = 2$, $n_a = 3$, $n_b = 3$, $n_c = 3$, $n_\varphi = 18$ for the NNARMAX model predictors as well as $n_h = 10$, $n_o = 2$, $\alpha_h = 1e-5$ and $\alpha_o = 1e-4$. The details of these parameters are discussed in section 3; where p and q are the number of inputs and outputs of the system, n_a, n_b and n_c are the orders of the regressors, n_φ is the total number of regressors (that is, the total number of inputs to the network), n_h and n_o are the number of hidden and output layers neurons, and α_h and α_o are the hidden and output layers weight decay terms. Also, $\mu = 1e-3$ is selected to initialize the INCBPA while $\lambda_\tau = 0.001$, $s = 0.05$ and $\delta = 0.01$ are arbitrarily selected to initialize the OWA-MLMA. The network is trained for τ = 20 iterations with the just mentioned selected parameters.

5.1.2. Validation of the Trained Network: Results and Discussion

The convergences of the OWA-MLMA based on AGNA

and the INCBPA from NNARMAX model training to capture the NXT robot model are shown in Figure 15. It can be seen in Figure 15, INCBPA shows faster but poor convergences with smaller computation times as shown in the second row of Table 3 when compared to OWA-MLMA. The superior convergences of the OWA-MLMA compared to the INCBPA are shown in Figure 15 with much smaller minimum performance index (MPI) at the expense of higher computation time.

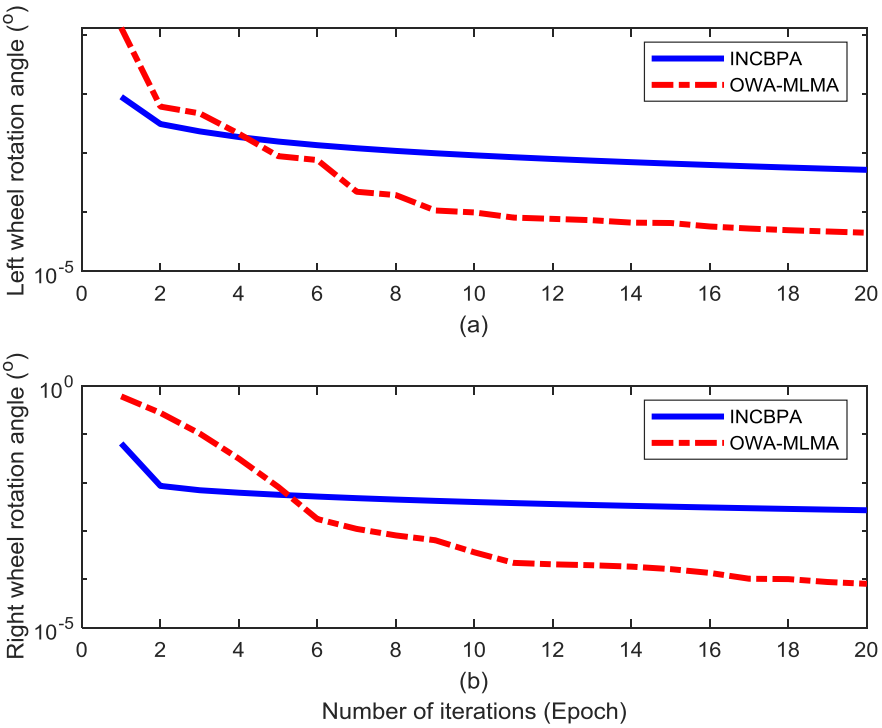


Figure 15. *Convergence and minimum performance index comparison of the OWA-MLMA based on AGNA and INCBPA training algorithms for 20 iterations for the NXT robot modeling.*

Table 3. *Performance comparison of the INCBPA and OWA-MLMA training results.*

S/N	Performance Evaluation Parameters	Left wheel rotation angle		Right wheel rotation angle	
		INCBPA	OWA-MLMA	INCBPA	OWA-MLMA
1	Computation time for model identification (sec)	5.8120e+02	1.5109e+03	5.7036e+02	1.3225e+03
2	Minimum performance index (MPI)	5.2343e-02	4.4952e-05	2.6659e-02	8.0105e-05
3	Total square error (TSE)	4.1499e+01	9.5898e-03	1.1104e+01	2.6953e-03
4	Mean error of one step ahead prediction of training data	1.0442e-01	1.1104e-04	7.0847e-01	2.6953e-04
5	Mean error of one step prediction of test data	3.2832e+01	5.1594e-02	2.8006e+01	1.1955e-03
6	Mean value of 5-step ahead prediction error	4.8886e+01	5.0041e-02	5.1418e+01	5.0736e-02
7	Akaike's final prediction error (AFPE) estimate	3.7989e+01	9.8119e-02	5.1532e+01	1.4494e-02

The total square error (TSE) discussed in Subsection 3.1.1, for the network trained with the INCBPA and the OWA-MLMA algorithms are given in the fourth row of Table 3. The OWA-MLMA has a much smaller TSE when compared to the INCBPA algorithm. These much smaller values of the TSE and the MPIs indicate that OWA-MLMA performs

better than the INCBPA for the same number of iterations. These much smaller errors suggest that the model identified by the OWA-MLMA approximates the NXT robot much better than the INCBPA model.

1) Validation by the One-Step Ahead Predictions Simulation using Scaled Training Data

In the one-step ahead prediction method, the errors obtained from one-step ahead output predictions of the trained network are assessed. In Figure 16, the graphs for the one-step ahead predictions of the scaled training data (blue -) against the trained network output predictions (red --*) using the NNARMAX models trained by INCBPA and proposed OWA-MLMA.

The mean value of the one-step ahead prediction errors are given in the fifth row of Table 3. It can be seen in the figures that the network predictions of the training data almost closely match the original training data. Although, the scaled training data prediction errors by both algorithms are small, the proposed OWA-MLMA appears to have a much smaller error when compared to the INCBPA as shown in the fifth row of Table 3. These small one-step ahead prediction errors are indications that the networks trained using the OWA-MLMA captures and approximate the nonlinear dynamics of the NXT robot to a high degree of accuracy. This is further justified by the small mean values of the TSE obtained for the networks trained using the proposed OWA-MLMA for the NXT robot as shown in the fourth row of Table 3.

2) Validation by the One-Step Ahead Predictions Simulation using Unscaled Test Data

Furthermore, the suitability of the INCBPA and the proposed OWA-MLMA for NNARMAX model identification for use with the NXT robot is investigated by validating the trained network with the 75,000 unscaled dynamic test data. Graphs of the trained network predictions (red --*) of the test data with the actual test data (blue -) using the INCBPA and the proposed OWA-MLMA are shown in Figure 17(a) and (b) for the left and right wheels respectively. The test data predictions prove the effectiveness of these algorithms. The prediction accuracies of the unscaled test data by the networks trained using the INCBPA and the proposed OWA-MLMA evaluated by the computed mean prediction errors shown in the sixth row of Table 3. Again, one can observe that the test data predictions errors obtained with the model trained by the proposed OWA-MLMA appears much smaller when compared to those obtained by the model trained using the INCBPA. The predictions of the unscaled test data given in Figure 17 as well as the mean value of the one step ahead test data prediction errors in the six row of Table 3 verifies the NN ability to model accurately the dynamics of the NXT robot using the proposed OWA-MLMA training algorithm.

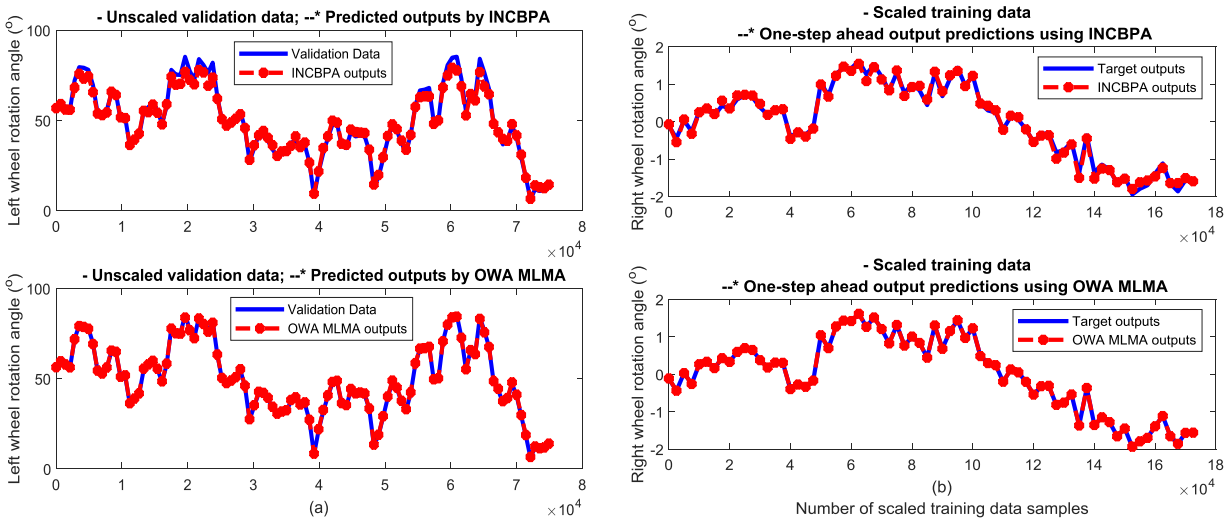


Figure 16. *Comparison of one-step ahead output predictions by the trained network (red --*) with the scaled original training data (blue -) for (a) left and (b) right wheel rotation angle angles by OWA-MLMA based on AGNA and INCBPA training algorithms.*

3) K-Step Ahead Prediction Simulations using Unscaled Training Data

The third method of validation is the K-step ahead predictions where the outputs of the trained network are compared to the unscaled output training data [30-32]. The results of the K-step ahead output predictions (red --*) using the K-step ahead prediction validation method for 5-step ahead output predictions (that is K = 5) compared with the unscaled training data (blue -) are shown in Figure 18(a) and 18(b) for the networks trained using the INCBPA and the proposed OWA-MLMA. The value K = 5 is chosen since it is a typical value used in most model predictive control (MPC) applications [30-32]. The comparison of the 5-step ahead output predictions performance by the network trained using the INCBPA and the proposed OWA-MLMA algorithms indicate the superiority of the proposed OWA-MLMA over the

INCBPA.

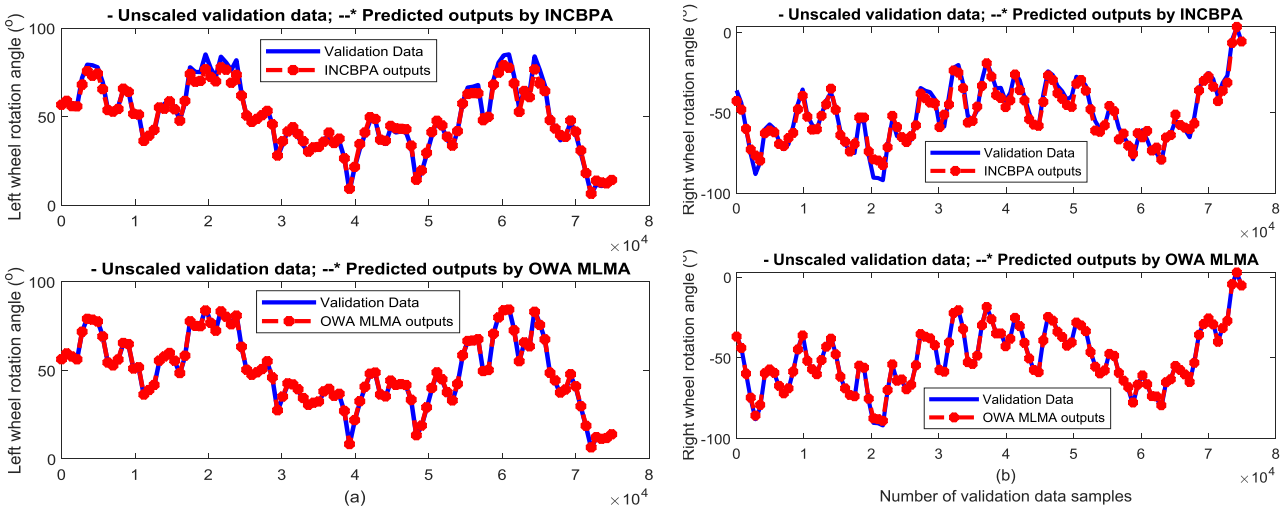


Figure 17. *Comparison of one-step ahead output predictions by the trained network (red --*) with the unscaled test data (blue -) for (a) left and (b) right wheel rotation angles by OWA-MLMA based on AGNA and INCBPA training algorithms.*

The computations of the mean value of the K-step ahead prediction error (MVPE) are given in the seventh row of Table 3 by the network trained using INCBP and the proposed OWA-MLMA respectively. The smaller MVPE of the 5-step ahead predictions are indications that the trained network approximates the dynamics of the NXT robot to a high degree of accuracy with the networks of both algorithms but with the network based on the OWA-MLMA giving much smaller distant prediction errors.

4) Akaike's Final Prediction Error (AFPE) Estimates

The implementation of the AFPE algorithm for the regularized criterion [17, 27, 45] for the network trained using the INCBP and the proposed OWA-MLMA with multiple weight decay gives their respective AFPE estimates which are listed in the eighth row of Table 3. These relatively small values of the AFPE estimate indicate that the trained networks capture the underlying dynamics of the aerobic reactor of the NXT robot and that the network is not over-trained [27, 45]. This in turn implies that optimal network parameters have been selected including the weight decay parameters. Again, the results of the AFPE estimates computed for the networks trained using the proposed OWA-MLMA are much smaller when compared to those obtained using INCBPA.

Table 4. *The NXT robot constraints for adaptive control.*

Constraint Parameters	**Fixed-Parameter PID Controller**		**NAPC based on FNNO-MLMA**	
	LWRA	**RWRA**	**LWRA**	**RWRA**
Minimum control input (u_{min})	-10	-10	-10	-10
Maximum control input (u_{max})	10	10	10	10
Minimum predicted output (y_{min})	-0.5	-0.5	-0.5	-0.5
Maximum predicted output (y_{max})	10	10	10	10
Minimum desired reference (Ref_{min})	-100	-100	-100	-100
Maximum desired reference (Ref_{max})	100	100	100	100

Table 5. *The NXT robot controller tuning parameters.*

Tuning Parameters	Fixed-Parameter PID Controller		NAPC based on FNNO-MLMA	
	LWRA	**RWRA**	**LWRA**	**RWRA**
ICI *(u)*	-80	-80	-10	-10
IPO *(y)*	0	0	0	0
N_d	-	1	1	1
N_u	-	2	3	2
N_p	-	5	7	5
K	-	1	1.5	1
ρ	-	0.8	0.08	0.08
λ	-	-	0.1	0.7
A_m	[1-0.7]	[1-0.7]	[1-0.7]	[1-0.7]
B_m	[00.3]	[00.3]	[00.3]	[00.3]
$Glob_{min}$	-	-	7	5
δ	-	-	1e-6	1e-4
u_{iter}	-	-	10	8
K_P	30	-	-	-
K_I	500	-	-	-
K_D	100	-	-	-

*ICI = Initial control input; IPO = Initial predicted output.

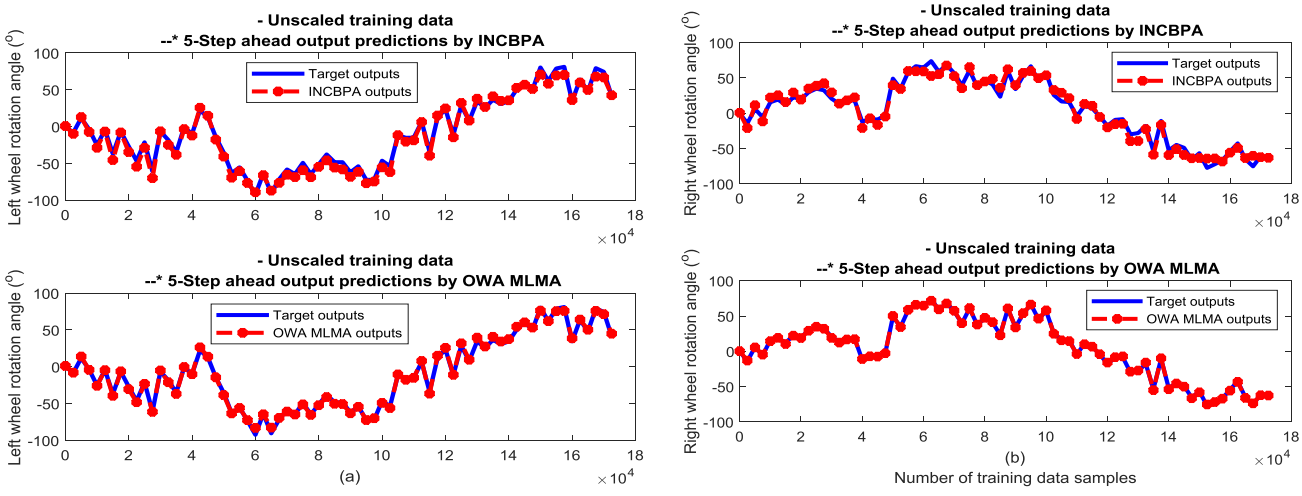


Figure 18. *Comparison of 5-step ahead output predictions by the trained network (red --*) with the unscaled original training data (blue -) for (a) left and (b) right wheel rotation angle by OWA-MLMA based on AGNA and INCBPA training algorithms.*

5.2. Performance Comparison of the Off-Line Closed-Loop Model Identification and Control of the NXT Robot Using NAPC and PID: Results and Discussion

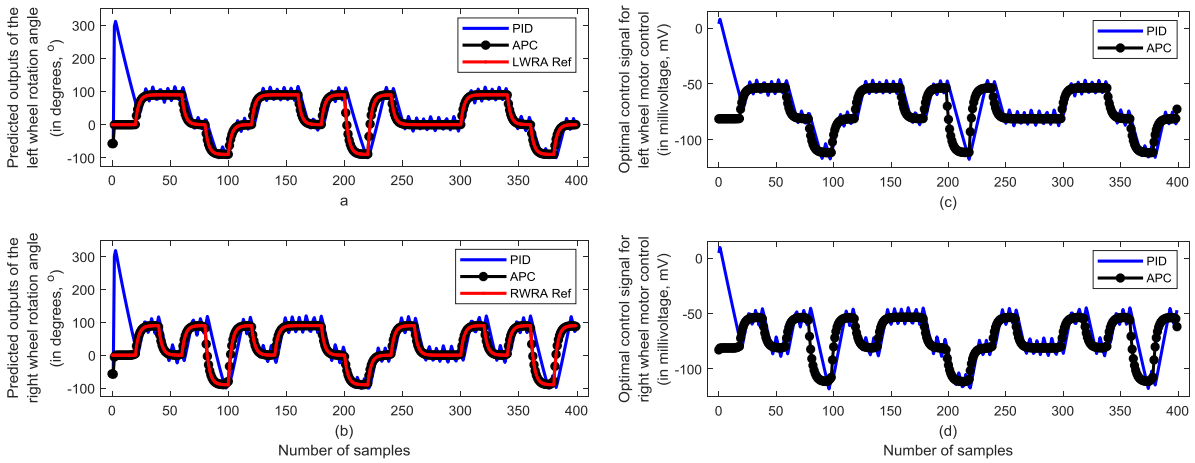
The main control objective here is to ensure that the NXT robot movement closely follows the prescribed designed reference trajectory without overshoot, oscillation or prolonged time lag.

Initially, the NNARMAX model of the NXT robot is identified and validated as in the previous subsection to obtain the optimal network parameters. The fixed-parameter PID and the NAPC based on FNNO-MLMA controllers are then simulated off-line separately for 400 simulation samples subject to the constraints in Table 4 using the NNARMAX model of the NXT robot to obtain the optimal tuning parameters given in Table 5. The data used for the off-line model identification and control is the experimental data obtained from the open-loop MATLAB/Simulink simulation.

The off-line closed-loop PID and the NAPC control performances of the *LWRA* and *RWRA* output predictions are shown in Figure 19(a) and (b) respectively while the control inputs *LWMV* and *RWMV* settings are shown in Figure 19(c) and (d) respectively. The output prediction errors due to the PID and NAPC controllers are shown in Figure 19(e) and (f) respectively for quick comparison. In these simulations, we allow the constraints on the maximum predicted outputs to be +100 °for both *LWRA* and *RWRA* as shown in Table 5 in order to observe any overshoot. As it can be seen in Figure 19(a) and (b), the NAPC based on FNNO-MLMA shows good control performance over the fixed-parameter PID. The fixed-parameter PID exhibits a very large overshoots of 310 ° for both *LWRA* and *RWRA* trajectory tracking with all-round oscillations and hardly track the desired reference signals as evident in Figure 19(a) and (b) as well as significant output prediction errors which is shown in Figure 19(e) and (f). The very large overshoot exhibited by the fixed-parameter PID of Figure 19(a) and (b) results in a serious penalty on the control signals with sudden sharp spike on the voltage source as shown in Figure 19(c) and (d). In Figure 19(a) and (b), one can also observe the extremely difficulty and poor ability of the fixed-parameter PID to track the desired reference trajectory at the 11th and 5th sampling time instant for *LWRA* and *RWRA* respectively. The poor performance of the fixed-parameter PID controller is due to the highly nonlinear behaviour of the NXT robot couple with the abrupt and sudden change of the NXT robot navigations as defined by the desired reference trajectory.

5.3. Online Closed-Loop Identification and Control of the NXT Robot Using NAPC Based on FNNO-MLMA, Results and Discussion

The online closed-loop NAPC based on the FNNO-MLMA control performance for the *LWRA* and *RWRA* output predictions for 280 simulation samples are shown in Figure 20(a) and (b) while the control inputs, *LWMV* and *RWMV* are shown in Figure 20(c) and (d). As it can be seen in Figure 20, the NAPC based on FNNO-MLMA shows good control performance with accurate tracking of the prescribed desired reference trajectory without any overshoots or oscillations. The minimum control efforts of the NAPC based on FNNO-MLMA is evident in Figure 20(c) and (d). It can be agreed that the excellent performance of the online closed-loop implementation of the proposed OWA-MLMA neural network training algorithm combined with the NAPC based on the FNNO-MLMA indicates their suitability for online nonlinear model identification and adoptive control for the NXT robot and can be adapted for other nonlinear applications.



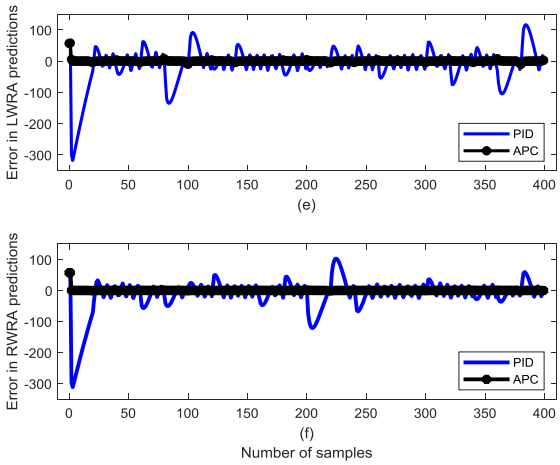


Figure 19*. Off-line NXT robot output predictions by PID (blue -) and nonlinear APC (black.-) for (a) LWRA and (b) RWRA with the manipulated signals (c) LWMV and (d) RWMV to track the desired reference signal (red -). Output prediction errors by PID (blue -) and NAPC (black.-) for (e) LWRA) and (f) RWRA.*

The combined OWA-MLMA based on AGNA for NNARMAX model identification and NAPC based on FNNO-MLMA for adaptive control were implemented on an Intel® Core™2 CPU running at 1.86 GHz under two conditions. The first condition is their direct serial implementation using MATLAB and the obtained computation times are shown in Figure 21(a). The minimum and maximum computation times are 17.8351 and 45.2167 seconds with an average computation time of 31.5259 seconds.

The second condition is their parallel implementation using the MATLAB "*parpool*" command available in the MATLAB Distributed and Parallel Toolbox [46]. The obtained computation times are shown in Figure 21(b). The minimum and maximum computation times are 3.5962 and 5.2273 seconds with an average computation time of 4.4117 seconds.

The "*parpool*" command implements the loop specified by the command in parallel and it uses the four Intel® processors available on the computer system. This MATLAB facility from the Distributed and Parallel Computing Toolbox allows the utilization of the four processors available on the computer for the implementation of the identification and control algorithm at each sampling time step.

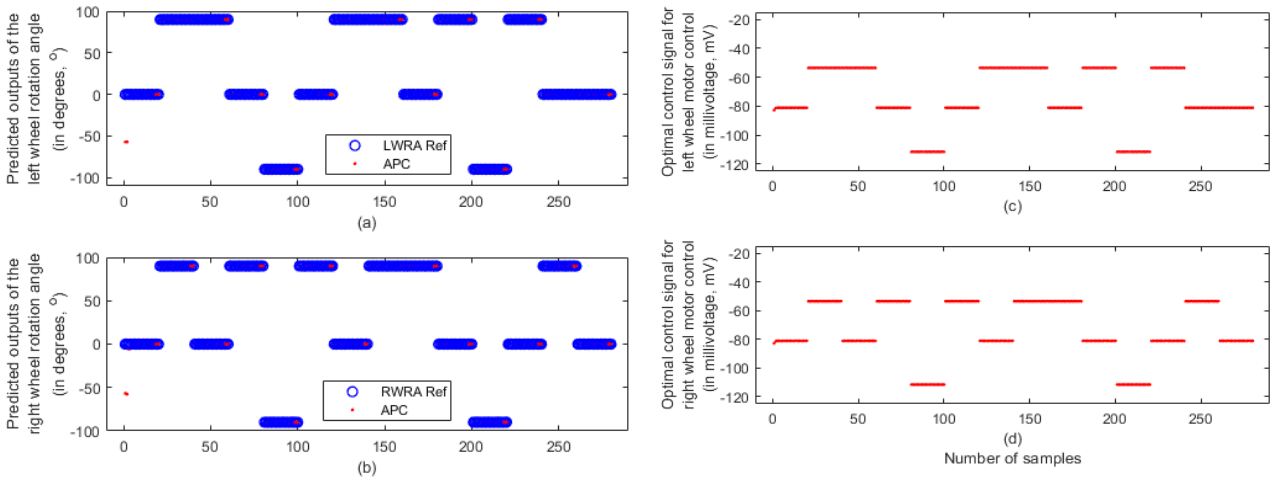


Figure 20*. Output predictions for the Online closed-loop implementation of the OWA-MLMA model identification and nonlinear APC based on FNNO-MLMA controller for the NXT robot: reference trajectory (blue o) and NAPC (red.) for (a) LWRA and (b) RWRA with the manipulated signals (c) LWMV and (d) RWMV.*

As it can be observed in Figure 21, the parallel implementation of the online closed-loop identification and NAPC control reduces the computation time for NXT robot model identification and adaptive control by 7.1460 times when compared to the serial implementation. The reduced computation time is still higher than the sampling time of the NXT robot of 0.5 seconds and this may limit the use of NXT robots for in real-time applications with strict time constraints in an

industrial environment. The online closed-loop identification and NAPC control algorithms could be implemented on a field programmable gate array (FPGA) which would significantly reduce the computation times such that model identification and adaptive control algorithm can be repeated severally within a single sampling time instant before updating the NXT robot for stable adaptive control under different disturbances [47, 48].

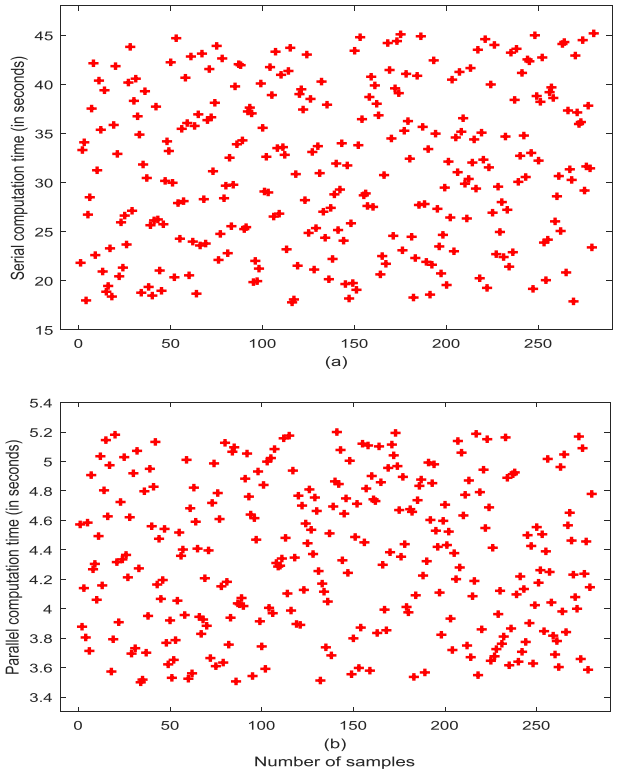


Figure 21*. Computation time for the implementation of the OWA-MLMA training algorithm and the nonlinear APC based on FNNO-MLMA control algorithm: (a) series and (b) parallel implementation.*

6. Conclusion

A new and comprehensive neural network-based online-window-approach modified Levenberg-Marquardt algorithm (OWA-MLMA) based on AGNA for NNARMAX model identification and a neural network-based nonlinear adaptive predictive control (NAPC) based on full-Newton nonlinear optimization modified Levenberg-Marquardt algorithm (FNNO-MLMA) for the adaptive control of a LEGO Mindstorms self-balancing two-wheel NXT robot has been presented, implemented and demonstrated in this paper with proven results.

The OWA-MLMA based on AGNA is a much more improved version of the batch off-line Levenberg-Marquardt algorithm (LMA). The OWA-MLMA based on AGNA is updated recursively as new experimental data are acquired online while old data are discarded which makes it suitable for online neural network training for nonlinear model identification that can be used to design adaptive controllers online. The NAPC based on FNNO-MLMA ensures the online adaptive selection of the search direction for guaranteed stability and positive definitiveness of the second-order derivative of the nonlinear optimization for online nonlinear adaptive controller design.

The developed neural network-based OWA-MLMA based on AGNA for model identification combined with the NAPC based on the FNNO-MLMA have online learning, adaptation, and fault tolerant capabilities as demonstrated in this work. Thus, the combined algorithms can dynamically adapt to changes in time-varying nonlinear systems under different operating conditions.

Furthermore, the sampling time of the NXT robot is 0.5 second and/or with/without noise and disturbances. The average computation time for the combined OWA-MLMA based on AGNA and NAPC based on FNNO-MLMA for serial and parallel implementation are approximately 31.5259 and 4.4117 seconds respectively at each identification and control sampling instant. The combined algorithms are executed on an Intel® Core™ 2 Quad CPU running at 2.66 GHz. Note that none of the serial or parallel implementation meets the sampling time of 0.5 seconds for the NXT robot. Thus, the parallel implementation of the combined identification and control strategies on real-time embedded multiprocessor systems such as field programmable gate array (FPGA) is recommended so that the combined algorithms can be computed much under the sampling time of the NXT robot.

Abbreviations

AFPE	Akaike's Final Prediction Error
AGNA	Approximate Gauss-Newton Algorithm
BPA	Back-propagation Algorithm
FNNO	Full-Newton Nonlinear Optimization
INCBP	Incremental Back-propagation
LMA	Levenberg-Marquardt Algorithm
MLMA	Modified Levenberg-Marquardt Algorithm
MBPC	Model-base Predictive Control
MPI	Minimum Performance Index
MRAC	Model Reference Adaptive Control
NAPC	Nonlinear Adaptive Predictive Control
NN	Neural Network
NNARMAX	Neural Network Nonlinear Autoregressive Moving Average with Exogenous Input
OWA	Open-window-approach
PID	Proportional-integral-derivative
TSE	Total Square Error

Author Contributions

Vincent Andrew Akpan: Data curation, Validation

Ahmed-Ade Fatai: Conceptualization, Project administration, Resources, Software, Supervision, Visualization, Writing – original draft

Olugbenga Kehinde Ogidan: Conceptualization, Data curation, Formal Analysis, Supervision, Writing – original draft

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] P. Corke, "Robotics, Vision and Control Fundamental Algorithms in MATLAB®," 2nd Edition, Springer Springer-Verlag, Heidelberg, 2017.

[2] P. H. Petkov, T. N. Slavov and J. K. Kralev "Design of Embedded Robust Control Systems Using MATLAB®/Simulink®," The Institution of Engineering and Technology, London, United Kingdom, 2018.

[3] C. S. Chin, "Computer-Aided Control System Design: Practical Applications Using MATLAB® and Simulink®," CRC Press - Taylor & Francis Group, Raton, USA, 2013.

[4] K. Ogata, "Modern Control Engineering," Pearson Education, Inc., Prentice Hall, Upper Saddle River, New Jersey, USA, 2010.

[5] W. S. Levine, "The Control Handbook: Control System Advanced Methods," Second Edition, CRC Press - Taylor & Francis Group, New York, USA, 2011.

[6] N. S. Nise, "Control Systems Engineering," Seventh Edition, John Wiley & Sons Inc., California, USA.

[7] A. A. Bobtsov, A. A. Pyrkin, S. A. Kolyubin, S. V. Shavetov, S. A. Chepinskiy, Y. A. Kapitanyuk, A. A. Kapitonov, V. M. Bardov, A. V. Titov and M. O. Surov, "Using of LEGO Mindstorms NXT Technology for Teaching of Basics of Adaptive Control Theory," IFAC Proceedings Volumes, vol. 44, no. 1, pp. 9818-9823, 2011.

[8] S. A. Fillippov, A. L. Fradkov, I. V. Ashikhmina and R. E. Seifullaev, "LEGO Mindstorms NXT Robot and Oscillators in Control Education," IFAC Proceedings Volumes, vol. 43, no. 11, pp. 156-160, 2010.

[9] B. A. Aleexeevich, A. K. Yuri, A. K. Alexander, A. K. Sergey, P. A. Alexandrovich, A. C. Sergey and V. S. Sergey, "LEGO Mindstorms NXT for Teaching the Principles of Adaptive Control to Students," Sci. & Tech. J. of Inf. Tech., Mech. and Opt., vol. 355, no. 1(71), pp. 103-108, 2011.

[10] M. Canale and S. Casale-Brunet, "A multidisciplinary approach for Model Predictive Control Education: A Lego Mindstorms NXT-based framework," Int. J. of Cont., Aut. & Sys, vol. 12, no. 5, pp. 1030 - 1039, 2014. https://doi.org/10.1007/s12555-013-0282-7

[11] H. L. Araujo, J. G. Agudelo, R. C. Vidal, J. A. Uribe, J. F. Remolina, C. Serpa-Imbett, A. M. López and D. P. Guevara, "Autonomous Mobile Robot Implemented in LEGO EV3 Integrated with Raspberry Pi to Use Android-Based Vision Control Algorithms for Human-Machine Interaction," Mach., vol. 10 no. 193, pp. 1-20, 2022. https://doi.org/10.3390/machines10030193

[12] J. Y. Chen, T. F. Wu, P. S. Tsai and K. Y. Lian, "Indirect Adaptive Fuzzy Controller for LEGO Mindstorms NXT Two-Wheeled Robot," App. Mech. & Mat., vols. 278-280, pp. 561-567, 2013, https://doi.org/10.4028/www.scientific.net/amm.278-280.561

[13] Н С. Ащепкова "Development of adaptive control system of model of the robot-loader on the basis of Lego Mindstorms NXT," Tech. Aud. & Prod. Res., vol. vol. 5(6), no. 25, pp. 45-48, 2015. https://doi.org/10.15587/2312-8372.2015.51215

[14] A. Mitov, J. Kralev, T. Slavov and I. Angelov, "Comparison between Model Predictive (MPC) and Model Reference Adaptive Controllers (MRAC) for Electrohydraulic Steering System Implemented as Real-Time Simulink® Program," IOP Conf. Series: Mat. Sci. and Eng., vol. 1002, no. 012034, pp. 1-12, 2020.

[15] V. A. Akpan and G. D. Hassapis, "Training dynamic feedforward neural networks for online nonlinear model identification and control applications," Int. Rev. of Aut. Cont.: Theo. & App., vol. 4, no. 3, pp. 335 - 350, 2011.

[16] V. A. Akpan and G. D. Hassapis, "Nonlinear model identification and adaptive model predictive control using neural networks," ISA Trans.; vol. 5, no. 2, pp. 177-94, 2011.

[17] V. A. Akpan, "Development of new model adaptive predictive control algorithms and their implementation on real-time embedded systems," Aristotle university of Thessaloniki, GR-54124, Thessaloniki, Greece, Ph.D. Dissertation, 517 pages, July, 2011. Available: http://invenio.lib.auth.gr/record/127274/files/GRI-2011-7292.pdf

[18] L. Kalra and C. Georgakis, "Effects of process nonlinearity on the performance of linear model predictive controllers for the environmentally safe operation of a fluid catalytic cracking unit," Ind. Eng. Chem. Res, vol. 33, pp.3063-3069, 1994.

[19] S. J. Qin and T. A. Badgwell, "A survey of model predictive control technology," Cont. Eng. Pract., vol. 11, pp. 733 - 764, 2003.

[20] V. A. Akpan, I. K. Samaras and G. D. Hassapis, "Implementation of Network Control System over a Service-Oriented-Architecture Computer Network Based on Device Profile for Web Services for Industrial Control Applications," Int. J. of Cont. Sci. & Eng., vol. 12, no. 1, pp. 1-25, 2022. Available: http://article.sapub.org/10.5923.j.control.20221201.01.html

[21] N. Khaled and B. Pattel, "Practical Design and Application of Model Predictive Control: MPC for MATLAB® and Simulink® Users," Butterworth-Heinemann, Oxford, United Kingdom, 2018.

[22] L. Grüne and J. Pannek, "Nonlinear Model Predictive Control: Theory and Algorithms," Second Edition, Springer International Publishing, Switzerland, 2017.

[23] G. Colin, Y. Chamaillard, G. Bloch and G. Corde, "Neural control of fast nonlinear systems - Application to turbocharged SI engine with VCT," IEEE Trans. Neu. Net., vol. 18, no. 4, pp. 1101 - 1114, Jul. 2007.

[24] C. Lu and C. Tsai, "Adaptive predictive control with recurrent neural network for industrial process: An application to temperature control of a variable-frequency oil-cooling machine," IEEE Trans. Ind. Elect., vol. 55, no. 3, pp. 1366-1375, Mar. 2008.

[25] Y. Yamamoto, "NXTway-GS Model-Based Design: Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT," Cybernet Systems Co. Limited, Revision 1.4, pp. 1 - 73, 2009.

[26] The MathWorks Inc., MATLAB® & Simulink® 2025, Natick, USA. Available: www.mathworks.com

[27] L. Lung, "System Identification: Theory for the User," 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[28] R. Chiong, "Intelligent Systems for Automated Learning and Application: Emerging Trends and Application," Hershey PA, USA: Information Science Reference, 2010, ch. 4, pp. 204 - 316.

[29] J. Wu, "Multilayer pottsperceptrons with Levenberg-Marquardt learning," IEEE Trans. Neu. Net., vol. 19, no. 12, pp. 2032-2043, Dec. 2008.

[30] V. A. Akpan and R. A. O. Osakwe, "Multivariable NNARMAX Model Identification of an AS-WWTP Using ARLS (Part 1: Dynamic Modeling of the Biological Reactors)," American J. of Int. Sys., vol. 4, no. 2, pp. 43 - 72, 2014. Available: http://article.sapub.org/pdf/10.5923.j.ajis.20140402.03.pdf

[31] V. A. Akpan and R. A. O. Osakwe, "Multivariable NNARMAX Model Identification of an AS-WWTP Using ARLS (Part 2: Dynamic Modeling of the Secondary Settler and Clarifier)," American J. of Int. Sys., vol. 4, no. 3, pp. 77 - 106, 2014. Available: http://article.sapub.org/pdf/10.5923.j.ajis.20140403.02.pdf

[32] V. A. Akpan and R. A. O. Osakwe, "Online Prediction of Influent Characteristics for Wastewater Treatment Plants Management Using Adaptive Recursive NNARMAX Model," American J. of Int. Sys., vol. 4, no. 3, pp. 107 - 130, 2014. Available: http://article.sapub.org/pdf/10.5923.j.ajis.20140403.03.pdf

[33] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," Nat., vol. 323, pp. 533-536, 1986.

[34] P. J. Werbos, "Backpropagation through time: What it does and how to do it," In Proc. IEEE, vol. 78, no. 10, pp. 1550 - 1560, Oct. 1990.

[35] J. E. Dennis and R. B. Schnabel, "Numerical Methods for Unconstrained Optimization and Nonlinear Equations," Englewood Cliffs, NJ: Prentice-Hall, 1996.

[36] M. T. Hagan and M. B. Menhaj, "Training feedforward network with the Marquardt algorithm," IEEE Trans. Neu. Net., vol. 5, no. 6, pp. 989-993, Nov. 1994.

[37] R. Fletcher, "Practical Methods of Optimization," 2nd ed., Chichester: Wiley & Sons, 1987.

[38] K. Levenberg, "A method for the Solution of Certain Non-Linear," Problems in Least Squares", Quart. ofAppl Math., vol. 2, no. 2, pp. 164 - 168, 1944. https://doi.org/10.1090/qam/10666

[39] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," J. Soc. for Ind. & Appl. Math., vol. 11, no. 2, pp. 431-441, 1963. https://doi.org/10.1137/0111030

[40] J. Hertz, A. Krough and R. G. Palmer, "An Introduction to the Theory of Neural Computation," Lecture Notes, vol. 1, Redwood City, California: Addison-Wesley, 1991.

[41] J. T. Spooner, M. Maggiore, R. Ordóñez and K. M. Passion, "Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques," New York: John Wiley & Sons, 2002.

[42] O. M. Omidvar and D. L. Elliot, "Neural systems for control," Academic Press, February, 1997. Available: http://www.isr.umd.edu/~delliot/NeuralSystemsForControl.pdf

[43] A. Visioli, "Practical PID Control," London: Springer-Verlag Ltd., 2006.

[44] L. Wang, S. Chai, D. Yoo, L. Gan and K. Ng, "PID and Predictive Control of Electrical Drives and Power Converters using MATLAB®/SIMULINK®," IEEE John Wiley & Sons Singapore Pte. Ltd, 2015.

[45] J. Sjöberg and L. Ljung, "Overtraining, regularization, and searching for minimum in neural networks," Int. J. of Cont., vol. 62: 1391-1408, 1995.

[46] The MathWorks Inc.., "Parallel Computing Toolbox™," March, 2025. The MathWorks, Inc. 3 Apple Hill Drive, Natick, MA 0170-2098, USA. Available: https://www.mathworks.com/products/parallel-computing.html

[47] V. A. Akpan, D. Chasapis and G. D. Hassapis, "FPGA Implementation of Neural Network-Based AGPC for Nonlinear F-16 Aircraft Auto-Pilot Control: Part 1 - Modeling, Synthesis, Verification and FPGA-in-the-Loop Co-Sim," American J. of Emb. Sys. & Appl., vol. 9, no. 1, pp. 6-36 pages, 2022. Available: https://www.sciencepg.com/journal/paperinfo?journalid=236&doi=10.11648/j.ajesa.20220901.13

[48] V. A. Akpan, D. Chasapis and G. D. Hassapis, "FPGA Implementation of Neural Network-Based AGPC for Nonlinear F-16 Aircraft Auto-Pilot Control: Part 2 - Implementation of Embedded PowerPC™440 with AGPC," American J. of Emb. Sys. & Appl., vol. 9, no. 2, pp. 37-65, 2022. Available: https://www.sciencepublishinggroup.com/journal/paperinfo?journalid=236&doi=10.11648/j.ajesa.20220902.11

Biography

Vincent Andrew Akpan obtained his B. Sc. (Hons) degree in Physics from Delta State University, Abraka, Delta State, Nigeria in 1997; a Master of technology (M. Tech) degree in Electronic Measurement & Instrumentation from The Federal University of Technology, Akure, Ondo State, Nigeria in 2003; and a Ph.D degree in Electrical & Computer Engineering from Aristotle University of Thessaloniki, Thessaloniki, Greece in 2011. He is currently a Professor with the Department of Biomedical Engineering, School of Electrical Systems Engineering, The Federal University of Technology (FUTA), Akure, Nigeria. He was Head of the Department of Biomedical Technology, FUTA between 2017 and 2024. His research interest is in artificial intelligence, advanced electronic instrumentation, intelligent adaptive control, automation and embedded systems engineering. He is the co-author of a book and has authored and/or co-authored more than 90 articles in refereed journals and conference proceedings. He is a reviewer of several local and international journals as well as conference proceedings. Prof. Akpan was one among two Nigerian recipients of the 2005 Bilateral Education Agreement (BEA) scholarship tenable in Greece for a P.hD programme. He has won several research grants. Prof. Akpan is a member of several local and international professional societies.

Ahmed-Ade Fatai obtained his Higher National Diploma (HND) in Physics Electronics from The Federal Polytechnic, Idah, Kogi State, Nigeria in 2010; Postgraduate Diploma (PGD) in Physics Electronics from the Federal University of Technology, Akure, Odo State, Nigeria in 2013; and a Master of Technology (M. Tech.) degree in Electronic Measurement and Instrumentation from The Federal University of Technology, Akure, Nigeria in 2019. He is currently a Principal Technologist with the Department of Physics Electronics, The Federal University, Lokoja, Kogi State, Nigeria. He is currently working towards his Ph.D degree in Electronic Measurement and Instrumentation with The Department of Physics Electronics, The Federal University of Technology, Akure, Nigeria. His research interest is in instrumentation, intelligent adaptive control and embedded systems engineering. He has authored and co-authored more 7 articles in refereed journals and conference proceedings. Mr. Fatai is a member of the Nigeria Institute of Science Laboratory Technologist (NISLT).

Olugbenga Kayode Ogidan received a B. Eng (Hons) degree in Electrical and Electronics Engineering from the University of Ado Ekiti, Ekiti State, Nigeria in 2001; a Master of Electrical and Electronics Engineering degree from The Federal University of Technology, Akure, Ondo State, Nigeria in 2010; and Ph.D degree in Electrical Engineering from the Cape Peninsula University of Technology (CPUT), South Africa in 2015. He is currently an Associate Professor with the Department of Electrical and Electronics Engineering, Faculty of Engineering, Elizade University, Ilara-Mokin, Ondo State, Nigeria. His research interests include Smart Irrigation System, Internet of Things (IoT), Real-time systems, Networked Control Systems and Renewable Energy. He is currently the Acting Director, Entrepreneurship Development Centre, Elizade University, Ilara-Mokin, Ondo State, Nigeria. His research tagged "*Solar Powered Smart Irrigation System*" received first place award nationwide in the Nigerian Academy of Engineering Inaugural Innovation Challenge in 2024. Dr. Ogidan is a recipient of two national patent awards from the Federal Government of Nigeria. Dr. Ogidan is one of the two African scholars to receive the Climate Action Grant from the Association of Commonwealth Universities, UK in 2021. He is a registered member of the Council for the Regulation of Engineering in Nigeria (COREN) and the Computer Professional Registration Council of Nigeria (CPN).

Research Field

Vincent Andrew Akpan: artificial intelligence, automation, electronic instrumentation, industrial networked control systems, intelligent adaptive control systems, real-time embedded systems, robotics.

Ahmed-Ade Fatai: electronic communication, electronic instrumentation, intelligent adaptive control, real-time embedded systems.

Olugbenga Kayode Ogidan: electronic instrumentation, Internet of things (IoT), networked control systems, real-time systems, renewable systems, solar powered smart irrigation systems.