

Adaptive Path Tracking Mobile Robot Controller Based on Neural Networks and Novel Grass Root Optimization Algorithm

Prof. Dr. Hanan A. R. Akkar

University Of Technology/ Electrical Engineering Department, Baghdad, Iraq
E-mail: dr_hanauot@yahoo.com

Firas R. Mahdi

University Of Technology/ Electrical Engineering Department, Baghdad, Iraq
E-mail: firasrasool1980@gmail.com

Abstract—This paper proposes a novel metaheuristic optimization algorithm and suggests an adaptive artificial neural network controller that based on the proposed optimization algorithm. The purpose of the neural controller is to track desired proposed velocities and path trajectory with the minimum error, in the presence of mobile robot parameters time variation and system model uncertainties. The proposed controller consists of two sub-neural controllers; the kinematic neural feedback controller, and the dynamic neural feedback controller. The external feedback kinematic neural controller was responsible of generating the velocity tracking signals that track the mobile robot linear and angular velocities depending on the robot posture error, and the desired velocities. On the other hand, the internal dynamic neural controller has been used to enhance the mobile robot against parameters uncertainty, parameters time variation, and disturbance noise. However, the proposed grass root population-based metaheuristic optimization algorithm has been used to optimize the weights of the neural network to have the behavior of an adaptive nonlinear trajectory tracking controller of a differential drive wheeled mobile robot. The proposed controller shows a very good ability to prepare an appropriate dynamic control left and right torque signals to drive various mobile robot platforms using the same offline optimized weights. Grass root optimization algorithms have been used due to their unique characteristics especially, their derivative free, ability to optimize discretely and continuous nonlinear functions, and ability to escape of local minimum solutions.

Index Terms—Artificial Neural Networks, Metaheuristic algorithms, Grass Root Algorithm, Trajectory tracking controller, Wheeled mobile robot.

I. INTRODUCTION

Since the first launch of the Artificial Neural Networks (ANNs) and they have been used in many applications such as; image and signal processing [1], robotics control

[2], classification and clustering of data pattern sets [3]. The ANNs have the ability to approximate any nonlinear model by using parallel computation techniques [4]. One of the major topics related to the ANNs theory is the training process of the network weights. Many algorithms have been used for the learning purpose. However, one of the most famous algorithms in this field is the error back propagation algorithm which has been extensively used for the NNs training. Back propagation algorithm appears to be suffering from several problems such as easy being trapped into local minimum solution, and its low convergence speed [5]. Many papers have been proposed to develop the performance of the back propagation algorithm while other have just left this concept and migrate to another type of algorithms that called Metaheuristic Optimization Algorithms (MOA) [6,7] especially, in the training phase of the NNs.

The neural based controllers of robotic systems have been gained a great significance in the few recent years. These networks were recommended for their learning ability, adaptive behavior, and their high performance. An intelligent Mobile Robot (MR) has become an exciting choice for many scientific papers and industrial applications. Therefore, a lot of care have been spent to enhance controllers in the adaptive field and artificial intelligent.

This paper suggests an Adaptive Neural Controller (ANC), that's trained offline using MOA inspired by the grass plants reproduction and fibrous root system called Grass Root Optimization algorithm (GRO). The proposed ANC controller tracks the desired trajectory in the presence of parameters time variation and uncertainties using an offline fixed trained weight.

II. RELATED WORKS

Many researchers have written to solve the problem of nonlinear trajectory tracking of wheeled mobile robot controllers under the conception of non-holonomic constraints. In 1995 R. Fierro and F. L. Lewis [8] propose an expanding for Kanayama tracking rule. The expanding

integrates the kinematic controller with NN computed torque controller for the non-holonomic mobile robot. The proposed controller was designed to have the ability to deal with the basic non-holonomic MR and especially trajectory tracking problem, path following problem. In 2005 X. Jiang and et. al [9] proposed a mixed predictive and fuzzy logic mobile robot trajectory tracking controller. Fuzzy logic was intended to deal with the system nonlinear characteristic, while the predictive controller minimizes the time delay due to sensors low response, by predicting the position and direction of the MR. In 2012 R. M. Soto and et. al [10] used a hybrid evolutionary PSO and GA population-based optimization algorithm to design a fuzzy controller for the trajectory tracking of the MR. The hybrid evolutionary algorithm has been used to find the fuzzy membership function parameters. Both PSO and GA algorithms have to communicate with each other every predefined iteration number. In 2013 C. Raim Undez and A. Barreiro Blas [11] proposed an adaptive NN controller to compute the proportional torque besides another derivative to lead a non-holonomic MR through tracking problem. A dynamic inverse NN controller is responsible for the tracking trajectory, reduction of the obtained error and adaptation of the disturbance.

This work proposes a novel metaheuristic algorithm using it in optimization weights of ANNs that tracks the velocities and posture of a differential drive wheeled mobile robot in the presence of time variation of system parameters.

III. GRASS ROOT OPTIMIZATION ALGORITHM

Grass plants have three types of roots; primary roots, secondary roots, and hair roots. The first root grown in the soil is the primary root, which supplies the required energy and minerals to the first few generated leaves. Primary root is usually stayed active for only short time, till the secondary roots became active, then primary root dies and the new generated secondary roots take its function. The new initiated secondary roots develop new tillers and shoots and compete with other neighbor plant roots to absorb water and other minerals. On the other hand, hair roots are small branches, grow out of the epidermis of the secondary roots, they absorb almost all the required water and minerals. Grass plants have two major asexual reproduction methods. These methods usually depend on the stems that grow sideways; either under or above the ground. The first vegetative reproduction method in grass plants are the stolons, which are stems creep along the ground, while the second vegetative method is the rhizomes, which are stems grow below ground. Grass plants use both stolons and rhizomes to establish new grass culms. These stolons and rhizomes support the newly created grass plant until it became strong enough to live on its own [12].

Grasses perform global and local search to find better resources by reproduction of new grass plants using stolons and rhizomes and by developing their own root system. The generated new grasses and roots are

developed randomly, but when grass arrives at a place with high resources, it generates more secondary roots, rhizomes, and stolons. The GRO algorithm depends the same reproduction and development of grasses to execute the global and local search. The algorithm starts by initializing random grasses in the search space, evaluates their fitness, generates a new population that contains the best-obtained grass and modifies its secondary and hair roots to search for better resources. If grass plant trapped into local minimum position, it generates new grasses by stolons initiated from the best-obtained grass, and other survived grasses. However, new secondary roots with different deviation from the local minimum solution will be generated by rhizomes. These roots help the trapped plant to reach farther places and escape the local position. GRO algorithm begins with an initial random uniformly distributed population (\mathbf{p}) in the problem search space. The number of the initially generated grasses equal to the population size (ps). When iteration (i) begins, a new grass population (\mathbf{NP}) is created. \mathbf{NP} consists of three elements which are; the best grass obtained so far by the iteration process ($\mathbf{gb} = \min(F(\mathbf{p})) \in \mathbb{R}^{Dim}$) where F is the cost function, Dim is the problem dimension. The second element of \mathbf{NP} is a number (Gr) of grasses evaluated as in (1) and deviated from \mathbf{gb} by stolons (\mathbf{SG}) as in (2). These stolons deviated from the original grass with step size usually less than maximum upper limit vector \mathbf{ul} . The last element of \mathbf{NP} was a number of grasses equal to $(ps-Gr-1)$ deviated randomly from the survived grasses of the iteration process (\mathbf{SD}) as shown in (3).

$$Gr = \left(\frac{amse}{amse + minm} \right) \left(\frac{ps}{2} \right) \quad (1)$$

$$\mathbf{SG} = ones(Gr,1) * \mathbf{gb} + 2 * max(\mathbf{ul}) \dots \dots * (rand(Gr,1) - 0.5) * \mathbf{gb} \quad (2)$$

$$\mathbf{SD} = \mathbf{GS} + 2 * max(\mathbf{ul}) * \dots \dots (rand(ps - Gr - 1, 1) - 0.5) * \mathbf{ul} \quad (3)$$

where $amse$ is the average value of the \mathbf{mse} vector shown in (4). mse_i is the MSE value of i^{th} iteration given by (5), $minm$ is the minimum of \mathbf{mse} shown in (6). $Ones(a,1)$ is a column vector operator of a rows of ones. $rand(a,1)$ is a random column vector operator with a rows, its elements $\in [0,1]$. $max(\cdot)$ and $min(\cdot)$ represent operators that find the maximum and minimum element of a vector respectively. \mathbf{GS} is the $(ps-Gr-1)$ highest mse grasses retrieved from the last iteration process.

$$amse = \frac{1}{ps} \sum_{i=1}^{ps} mse_i \quad (4)$$

$$mse_i = \frac{1}{no} \sum_{j=1}^{no} (ds_j - y_j)^2 \quad (5)$$

$$\min m = \min(\mathbf{mse}) \quad (6)$$

$$\mathbf{mse} = \left[mse_1, \dots, mse_i, \dots, mse_{ps} \right]^T \quad (7)$$

ds_j, y_j are the desired and actual values of the j^{th} output. (8) shows the new population vector \mathbf{NP} .

$$\mathbf{NP} = \left[\mathbf{gb}^T \mathbf{SG}^T \mathbf{SD}^T \right]^T \quad (8)$$

Generated (\mathbf{NP}) will be bounded and checked to get the fittest obtained grass. If the fittest new generated grass is best than the old one then keep the fittest new grass as the best grass solution. Otherwise, calculate the absolute rate of decrease in \mathbf{mse} between the best grass obtained so far $bestm$ shown in (9) and the current iteration minimum \mathbf{mse} ($\min m$) shown in (6). If the rate is less or equal a tolerance value (tol) shown in condition (10) increase global stack (GC) counter by one till it reached its maximum value then move to the local search mechanism.

$$bestm = \min(\min m) \quad (9)$$

$$\left| \frac{\min m_j - bestm}{\min m_j} \leq tol \right| \quad (10)$$

$$\min m = [\min m_1, \dots, \min m_p, \dots, \min m_{il}]^T \quad (11)$$

The local search mechanism consists of two individual loops; secondary roots loop, and hair roots loop. Hair roots are considered equal to the dimensions of the objective function (Dim), and each secondary root generated by the \mathbf{gb} primary root represents a local candidate solution. The secondary roots number is represented by a random number usually less than Dim . Each single hair root modifies its location as in (12) for a repeated loops equal or less than secondary roots number (SC).

$$mgb_i^{k+1} = agb + gb_i^K + C2 * (rand - 0.5) \quad (12)$$

$$k=1,2,\dots,SC \text{ and } i=1,2,\dots,Dim.$$

$$agb = \frac{1}{Dim} \sum_{i=1}^{Dim} mgb_i \quad (13)$$

$$\mathbf{C} = [0.02, 0.02, 0.02, 0.2, 0.2, 2, 2, 2, 2, 15] \quad (14)$$

$$C_2 = C(1 + rand * 10) \quad (15)$$

Where \mathbf{mgb} is the locally modified \mathbf{gb} , agb is the average of the \mathbf{mgb} vector shown in (13), SC is the number of secondarily generated roots where ($0 \leq SC \leq Dim$), and \mathbf{C} is the searching step vector shown in (14). $C2$ represents the random element of \mathbf{C} chosen according

to the percentage repetition of \mathbf{C} elements as shown in (15). If the evaluated fitness of \mathbf{mgb} is less than $bestm$ then save \mathbf{gb} as \mathbf{mgb} . Otherwise, calculate the absolute rate of decrease in mean square error. If the rate is less than tol then increase local stack counter (CL) by one. When CL reached its maximum predefined value, break hair root loop and begin new secondary root loop. After each completed iteration check if the stopping condition (GE) is satisfied then stop the iteration. Otherwise, go to the next iteration until reached the mit (maximum iteration number) then stop. Table 1. illustrates the general proposed algorithm pseudo code.

Table 1. GRO pseudo code

```

1: Initialize  $mit, ps, Dim, ll, ul, GE, tol, GC, LC, SG, C, gb, minm, bestm, Gr, F, SD$ .
2: Initialize random grass population ( $\mathbf{P}$ ),  $p \in R^{Dim}$ .
3: Bound the initial population  $ll \leq p \leq ul, ul, ll \in R^{Dim}$ .
4: For  $i=1:ps$  // check for best fitness particle in the initial population
5:  $f = F(\mathbf{P})$  //  $F$  is a predefined objective function //
6: Calculate the  $mse(i)$  for  $f$  elements as in (5)
7: End For.
8: Sort the grass population ( $\mathbf{P}$ ) ascending according to step 6.
9: For  $it=1:mit$  // iteration and global search starting //
10: Evaluate  $C_2$  according to (15).
11: Generate the new population ( $\mathbf{NP}$ ) according to (8).
12: Bound  $\mathbf{NP} : ll \leq \mathbf{NP} \leq ul, \mathbf{NP}, ll, ul \in R^{Dim}$ .
13: For  $i=1:ps$  // check the  $\mathbf{NP}$  for the best fitness particle loop //
14:  $f_{new} = F(\mathbf{NP})$ 
15:  $Mse(i) = f_{new}$ 
16: End For
17:  $Minm = \min(Mse)$  // save minimum mean square error //
18: Index the grass with the  $Minm$  // index the position of the best particle in the population //
19: Evaluate  $Gr$  as in (1). //  $Gr$  represents number of stolons or deviated from  $\mathbf{gb}$  //
20: Evaluate  $\mathbf{SG}$  as in (2).
21: Evaluate the  $\mathbf{GS}$  from the ascending sorted  $p$ 
22: Evaluate  $\mathbf{SD}$  as in Equation (3).
23: If  $\min m < bestm$  then
24:  $bestm = \min m$ 
25:  $\mathbf{gb} = best$  indexed grass
26:  $GC = 0$ 
27: Else If condition (10) is true then
28: increase  $GC$  by 1
29: If  $GC$  is at its maximum then
30:  $GC = 0$  // Begin the local search //
31: For  $q=1$ : random integer less than  $Dim$ 
32:  $LC = 0$ 
33: For  $j=1:Dim$  // hair root loop //
34:  $\mathbf{mgb} = \mathbf{gb}$  // initial  $\mathbf{mgb}$  //
35: evaluate  $\mathbf{mgb}$  as in (12)
36:  $lmin = mean\ square\ error(\mathbf{mgb})$ 
37: If  $lmin < bestm$  then
38:  $bestm = lmin$ 
39:  $\mathbf{gb} = \mathbf{mgb}$ 
40:  $CL = 0$ 
41: Else If condition (10) with  $lmin$  instead of  $\min m$  is true then
42: Increase  $CL$  by one
43: Else
44:  $CL = 0$ 
45: End If
46: If  $CL$  is at its maximum then
    
```

```

47:   Break For j
48:   End If
49:   End For (j loop)
50:   End For (q loop)
51:   End If
52:   End If
53:   If bestm ≤ GE then
54:     break For (it loop)
55:   End If
56: End For (it loop)

```

A. GRO Optimizing NN Weights

The role of GRO algorithm is firstly; to bound and check the candidate solutions in population, modifies the solutions in an iteration process to get the best fitness solution. However, for supervised training NNs, the objective function is the Mean Square Error (MSE) function. Therefore, when the weights of the network are fully optimized, very minimum or zero MSE is obtained.

In order to make population-based metaheuristic GRO algorithm as a supervised training algorithm, we have considered all layer weights of the NNs as a grass vector in the population matrix. Each grass represents a candidate weight solution. The purpose of this weights vector is to minimize the objective MSE function of the NN. A number of weights in each grass vector represents the problem dimensions. Initial weights were created randomly, and the search space of the problem was limited to a minimum and maximum search space values.

Suppose the input data set represented by $\mathbf{x}(M, ni)$, and $\Phi(M, no)$ is the target teacher. Where $\mathbf{x} \in \mathbb{R}^{(M*ni)}$ and $\Phi \in \mathbb{R}^{(M*no)}$. M is the number of input patterns, ni and no are the number of input and output neurons respectively. If we suppose that l is the hidden neurons number, then NN dimension with bias connections is represented by (16) for single hidden layer.

$$N = (ni * l + l) + (l * no + no) \quad (16)$$

For multiple hidden layers suppose that l is a vector represents the layers and neurons as shown in (17). The network total dimension is represented by (18).

$$l = [l_0 l_1 \dots l_k \dots l_m], \text{ where } l_0 = ni, \text{ and } l_m = no \quad (17)$$

$$N = \sum_{k=0}^{m-1} (l_k + 1) l_{k+1} \quad (18)$$

Let the total weight vector $\mathbf{W}t \in \mathbb{R}^N$, then the layer weight vector could be represented by (19), where i represents the layer number and N_i shown in (20) represents each layer number of neurons.

$$\mathbf{W}t_i = [Wt_{N_{i-1}+1} Wt_{N_{i-1}+2} \dots Wt_{N_i}] \quad (19)$$

$$N_i = \begin{cases} (l_i + 1) l_{i+1}, & \text{for } i \geq 0 \\ 0, & \text{for } i < 0 \end{cases} \quad (20)$$

Approximately for most of the problems, a single hidden layer is usually sufficient to get the optimum solution. Two hidden layers are required sometimes for processing data with discontinuities and it rarely improves the performance of the networks. However, it may converge to a local minima solution when a large number of neurons is used. Therefore, there is no theoretical reason to use more than two hidden layers [13]. Hence, the network will have only two weights layer; input to hidden layer; and hidden to the output layer. For this purpose (19) will be rewritten as follows:

$$\mathbf{W}1 = [wt_1 \ wt_2 \ \dots \ wt_{ni*l+l}] \quad (21)$$

$$\mathbf{W}2 = [wt_{ni*l+l+1} \ wt_{ni*l+l+2} \ \dots \ wt_N] \quad (22)$$

$$\mathbf{B}i = [bi * \mathbf{ones}(M, 1)] \quad (23)$$

$$\mathbf{ones}(M, 1) = [1 \ 1 \ \dots \ 1_M]^T \quad (24)$$

Where $\mathbf{W}1$ and $\mathbf{W}2$ are the input to hidden weights layer and the hidden to output hidden layer respectively. Reshape $\mathbf{W}1$ and $\mathbf{W}2$ vectors into layer matrix form, then they will be written as follows:

$$\mathbf{\Omega} = \begin{bmatrix} wt_1 & wt_2 & \dots & wt_l \\ wt_{l+1} & wt_{l+2} & \dots & wt_{2*l} \\ \vdots & \vdots & \vdots & \vdots \\ wt_{ni*l+1} & wt_{ni*l+2} & \dots & wt_{l*(ni+1)} \end{bmatrix} \quad (25)$$

$$= \begin{bmatrix} \Omega_{11} & \Omega_{12} & \dots & \Omega_{1l} \\ \Omega_{21} & \Omega_{22} & \dots & \Omega_{2l} \\ \vdots & \vdots & \vdots & \vdots \\ \Omega_{(ni+1)1} & \Omega_{(ni+1)2} & \dots & \Omega_{(ni+1)l} \end{bmatrix}$$

$$\mathbf{\psi} = \begin{bmatrix} wt_{l*(ni+1)+1} & \dots & wt_{l*(ni+1)+no} \\ wt_{l*(ni+1)+no+1} & \vdots & wt_{l*(ni+1)+2no} \\ \vdots & \vdots & \vdots \\ wt_{l(ni+1+no)+1} & \dots & wt_{l(ni+1+no)+no} \end{bmatrix} \quad (26)$$

$$= \begin{bmatrix} \psi_{11} & \psi_{12} & \dots & \psi_{1no} \\ \psi_{21} & \psi_{22} & \dots & \psi_{2no} \\ \vdots & \vdots & \dots & \vdots \\ \psi_{(l+1)1} & \psi_{(l+1)2} & \dots & \psi_{(l+1)no} \end{bmatrix}$$

Where $\mathbf{\Omega} \in \mathbb{R}^{(ni+1)*l}$, and $\mathbf{\psi} \in \mathbb{R}^{(l+1)*no}$.

$$\mathbf{Xb} = [\mathbf{Bi} \ \mathbf{x}] \quad (27)$$

$\mathbf{Xb} \in \mathcal{R}^{Dim*(ni+1)}$ is the input pattern data set with input bias. The hidden layer output is shown in (28) for linear activation function.

$$\mathbf{h} = \mathbf{Xb} * \boldsymbol{\Omega} \quad (28)$$

Where $\mathbf{h} \in \mathcal{R}^{Dim*l}$. Adding bias to (28), another vector is constructed as in (29). The output of the network is shown in (30) for linear identity function.

$$\mathbf{H} = [\mathbf{Bi} \ \mathbf{h}] \quad (29)$$

$$\mathbf{y} = \mathbf{H} * \boldsymbol{\psi} \quad (30)$$

Where $\mathbf{H} \in \mathcal{R}^{Dim*(l+1)}$, $\mathbf{y} \in \mathcal{R}^{Dim*no}$

IV. DIFFERENTIAL DRIVE WHEELED MOBILE ROBOT

For the kinematic and dynamic MR modeling, suppose the model proposed by [14]. Fig. 1 shows a general differential drive wheeled MR geometrical structure, in which the point B represents the central point between the driver wheels, b is the straight distance between the center of gravity A and the wheel axis. The MR position is described by $\boldsymbol{\rho} = [X, Y, \vartheta]^T$, where X , and Y are the axis of point A. ϑ is the mobile robot steering rotation angle. The MR kinematic model is given by (31-33).

$$\dot{\boldsymbol{\rho}} = \mathbf{Q}(\boldsymbol{\rho})\boldsymbol{\Phi} \quad (31)$$

Where $\boldsymbol{\Phi} = [\Phi_r, \Phi_l]^T$, are the right and left wheels angular velocities.

$$\mathbf{Q}(\boldsymbol{\rho}) = \begin{bmatrix} \frac{r}{2} \cos(\vartheta) & \frac{r}{2} \cos(\vartheta) \\ \frac{r}{2} \sin(\vartheta) & \frac{a}{2} \sin(\vartheta) \\ \frac{r}{2} d & -\frac{r}{2} d \end{bmatrix} \quad (32)$$

$$\boldsymbol{\Phi} = \begin{bmatrix} \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \mathbf{A} \quad (33)$$

$\mathbf{A} = [v, w]^T$ represents the linear and angular velocities of the wheeled MR. Substituting (32) and (33) in (31) obtains the kinematic system model of the differential drive robot in terms of linear and angular velocities as shown in (34).

$$\dot{\boldsymbol{\rho}} = \begin{bmatrix} \cos(\vartheta) & 0 \\ \sin(\vartheta) & 0 \\ 0 & 1 \end{bmatrix} \mathbf{A} \quad (34)$$

For non-slipping and pure rolling condition, the non-holonomic constraint is shown in (35) [8].

$$\dot{X} \sin(\vartheta) - \dot{Y} \cos(\vartheta) = 0 \quad (35)$$

Let m be the mass of the robot platform, m_m the mass of the wheel and motor, I is the moment of inertia of the robot platform about the perpendicular axis over B, I_m is the wheel and motor moment of inertia around the wheel axis, and I_i is the wheel and motor moment of inertia around the wheel diameter. The dynamic robot model is represented by equations (36-41) as follows:

$$\mathbf{M}(\boldsymbol{\rho})\dot{\mathbf{A}} + \mathbf{N}(\boldsymbol{\rho}, \boldsymbol{\rho})\mathbf{A} = \mathbf{B}(\boldsymbol{\rho})\boldsymbol{\tau} \quad (36)$$

Where $\boldsymbol{\tau} = [\tau_r, \tau_l]$ stand for the right and left torques applied on the wheels, while \mathbf{M} , \mathbf{N} , \mathbf{B} are represented by:

$$\mathbf{M}(\boldsymbol{\rho}) = \begin{bmatrix} \frac{r^2}{4d^2}(m_t b^2 + I_t) + I_m & \frac{r^2}{4d^2}(m_t b^2 - I_t) \\ \frac{r^2}{4d^2}(m_t b^2 - I_t) & \frac{r^2}{4d^2}(m_t b^2 + I_t) + I_m \end{bmatrix} \quad (37)$$

$$\mathbf{N}(\boldsymbol{\rho}, \boldsymbol{\rho}) = \begin{bmatrix} 0 & \frac{r^2}{2d}(mbw) \\ -\frac{r^2}{2d}(mbw) & 0 \end{bmatrix} \quad (38)$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (39)$$

Where I_t and m_t represent the total moment of inertia and the total mobile robot mass as in (40-41).

$$I_t = mb^2 + I + 2m_m d^2 + 2I_i \quad (40)$$

$$m_t = m + 2m_m \quad (41)$$

V. THE PROPOSED ANC

This paper develops a trajectory tracking controller based on NNs that has been optimized using GRO algorithm for the differential drive MR of the Fig. 1. We have assumed that there is uncertainty in the dynamic system model. Furthermore, distance b , platform mass m , wheel radius r , and the platform width c_l are varying with time. After the NN training phase finished, the controller considered to be tough against parameters variation and

tracks the proposed trajectory at the dynamical and kinematic controller levels.

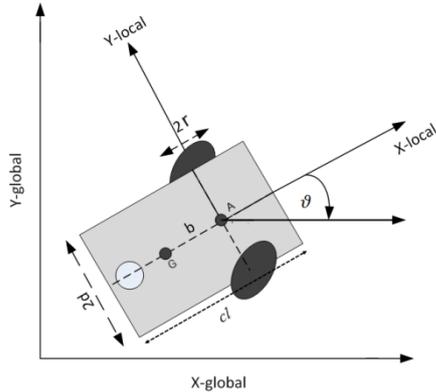


Fig.1. Differential Drive wheeled MR geometrical structure.

If supposed that there is a predefined desired trajectory given by (42).

$$\rho_d = \begin{bmatrix} \cos(\vartheta_d) & 0 \\ \sin(\vartheta_d) & 0 \\ 0 & 1 \end{bmatrix} A_d \quad (42)$$

Where $\rho_d = [X_d, Y_d, \vartheta_d]^T$, $A_d = [v_d, \dot{\vartheta}_d]$. The error between the desired and actual pose in the local robot frame are given by:

$$\begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} = T \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) & 0 \\ -\sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_d - X \\ Y_d - Y \\ \vartheta_d - \vartheta \end{bmatrix} \quad (43)$$

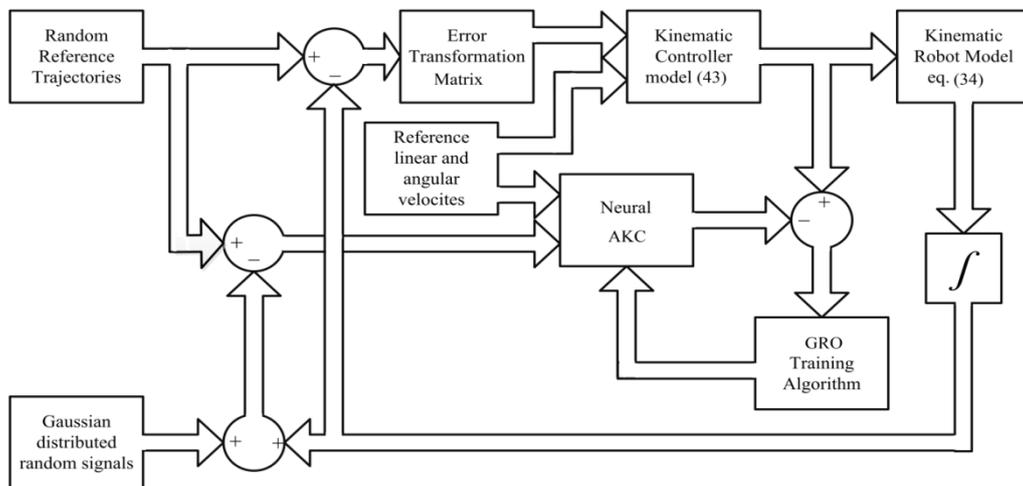


Fig.3. The KNC training phase.

The robot nominal values are proposed as follows; $r = 0.033$ m, $m = 0.575$ Kg, $c_l = 0.15$ m, $b = 0.04$ m. While for the training of the DNC purpose, we assumed that m varied between the values $[0.45, 1.2]$ Kg, distance b is varied in the interval of $[0.03, 0.1]$ m, c_l varied in between $[0.12, 0.2]$ m, r varies between $[0.03, 0.07]$ m.

The control inputs V and W which make E_1, E_2, E_3 converge to zero are given by (44) [15].

$$\begin{bmatrix} V \\ W \end{bmatrix} = \begin{bmatrix} \cos(E_3) & 0 \\ k_y E_2 & 1 \end{bmatrix} \begin{bmatrix} v_d \\ w_d \end{bmatrix} + \begin{bmatrix} k_x E_1 \\ k_\vartheta \sin(E_3) \end{bmatrix} \quad (44)$$

Where $k_x, k_y, k_\vartheta > 0$, represent the positive constants.

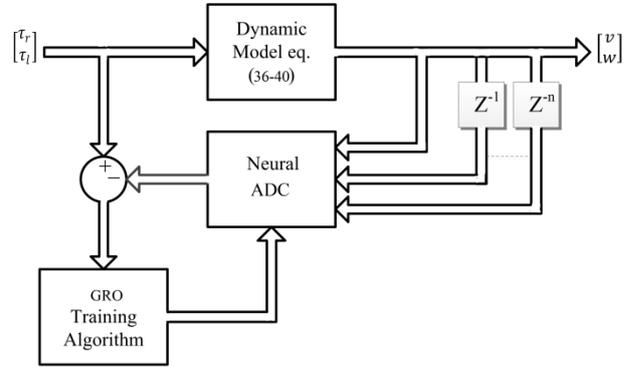


Fig.2. The DNC training phase

The DNC is designed to learn the collected input/output data from the dynamic model system (36-41) and learns the torque signals which transfers the MR from velocity at time (t) to upcoming ($t+1$) velocity. The DNC offers, after good training, an adaptive performance with fixed trained weights. However, the DNC has to be well trained for all possible parameters variation of the dynamical model especially, the values b, m, c_l , and r . Fig. 2 shows the DNC training phase block.

created, and a noisy data having zero mean and 0.1 variation level Gaussian distributed was added to the training data. The training input data of the KNC were the randomly created trajectory reference velocities and the error between the random trajectory and the kinematic model trajectory output plus noisy Gaussian distributed data. Fig. 3 shows the KNC training phase block.

10,000 samples data sets have been generated. These patterns data sets were divided into two sub-data; 5000 sets for training purpose, and 5000 sets for testing of the trained network. The used NNs architecture of both DNC and KNC consisted of single hidden layer with 4 hidden processing neurons and 2 output neurons. The DNC has 4 input neurons for the desired and real velocities, while KNC has 5 input neurons for the random reference velocities and the error posture with disturbance. The synaptic weights connections were randomly initialized between $[-1,+1]$. Linear identity activation function has been used for both input and hidden layers. 10 training cycles have been executed for both DNC and KNC. Each training cycle has 1000 maximum iterations (*mit*). 30

grass population (*ps*) were used. GRO algorithm stops iterations according to three proposed stopping conditions which are:

- ❖ If any grass converged to the predefined global error value, which is zero MSE in our case.
- ❖ When the testing MSE exceeds the training MSE by 10% of its value.
- ❖ When the algorithm completes the whole iteration cycles.

If GRO finished a training cycle without overfitting, the next cycle continues from the last reached point of the last valid cycle. The network weights will be the last known valid weights. The overall ANC structure is shown in Fig. 4. It consists of two feedback control loops; external feedback KNC controller that generates the control velocity signals that tracks the desired trajectories, and DNC internal control loop which is designed to improve the controller robustness against parameters uncertainty and time variations.

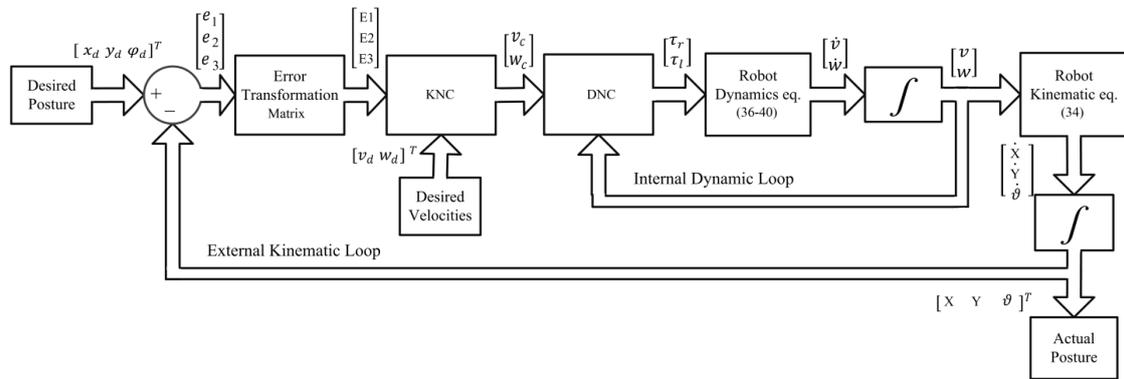


Fig.4. Overall ANC block diagram

Table 2. Dynamic parameters time variations.

Parameters	nominal values	Maximum values	Minimum values
b (m)	0.04 m	0.14 m	0 m
cl (m)	0.15 m	0.5 m	0.1 m
r (m)	0.033 m	0.093 m	0.023 m
m (Kg)	0.575Kg	3 Kg	0.25 Kg

For the purpose of testing the wheeled MR tracking performance, the mobile robot parameters have been assumed varying randomly as illustrated in Table 2. The MSE of the MR position and velocities are evaluated by (45) and (46).

$$Errp = \frac{1}{3} \sqrt{E_1^2 + E_2^2 + E_3^2} \quad (45)$$

$$Errv = \frac{1}{2} \sqrt{(v_d - v)^2 + (w_d - w)^2} \quad (46)$$

The desired velocities generated for the path tracking are illustrated in Table 3. The desired posture is found by integrating (34).

Table 3. velocities trajectories.

t	v_d	w_d
$0 \leq t \leq 157$	$0.05 \left(1 - \cos\left(\frac{2t}{10}\right) \right)$	$0.05 \left(1 - \cos\left(\frac{2t}{10}\right) \right)$
$157.1 \leq t \leq 300$	$0.05 \left(1 - \cos\left(\frac{2t}{10}\right) \right)$	$-0.05 \left(1 - \cos\left(\frac{2t}{10}\right) \right)$

VI. SIMULATION RESULTS

The GRO algorithm first training cycle for KNC and DNC are shown in Fig. 5 and Fig. 6 respectively. After 10 training cycles without overfitting of the KNC. The recorded maximum MSE was 0.0002 for the linear velocity and 0.4221 for the angular velocity. The average angular and linear velocities MSE was 0.2112 which is relatively high MSE, due to the large set of training input sets. The average testing MSE was 0.2042.

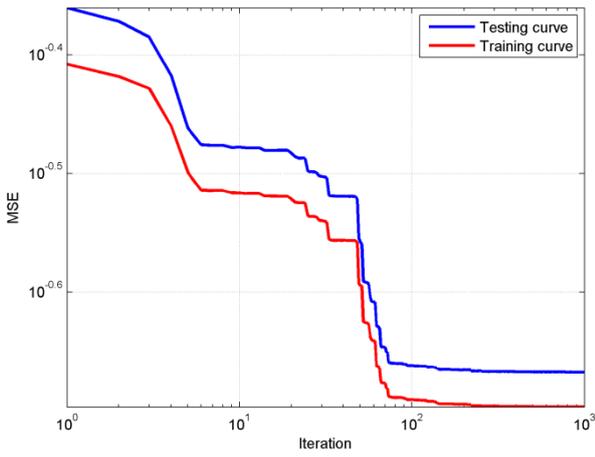


Fig.5. First epoch GRO training algorithm MSE for the KNC.

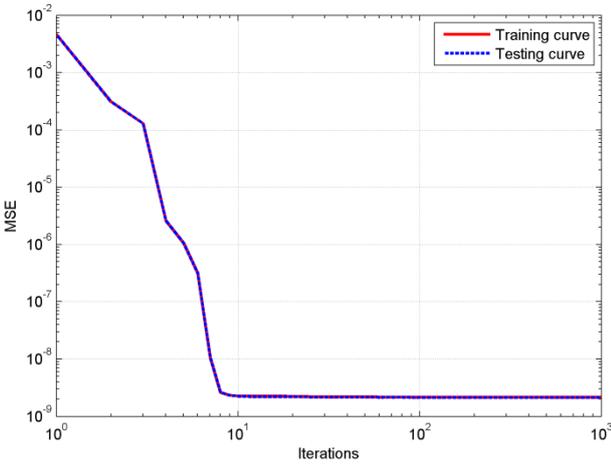


Fig.6. First epoch GRO training algorithm MSE for the DNC.

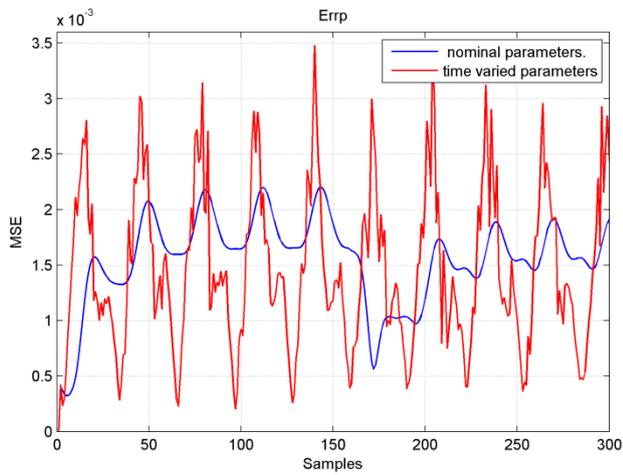


Fig.7. Mobile robot *Errp* MSE.

For the second DNC network, the recorded MSE was $2.33e-10$ for the left torque signal and $2.07e-10$ for the right torque signal. The average MSE of both torques was equal $2.199e-10$ with testing average MSE of $2.124e-10$. This very minimum value actually is due to the tiny torque signals that have a maximum value of 0.0001 N.m. The mobile robot position MSE (*Errp*) and velocities

MSE (*Errv*) were shown in Fig. 7 and Fig. 8. The velocity and position MSE of the MR with nominal parameters and time-varying parameters are less than $3.5e-03$. Finally, Fig. 9 shows the real and desired x-y trajectories tracking.

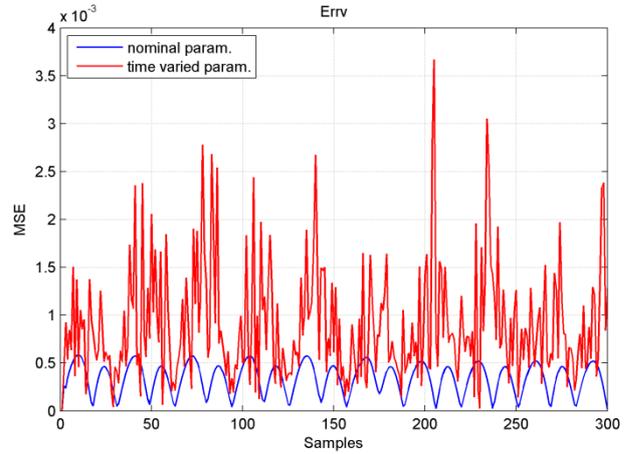


Fig.8. MOBILE ROBOT *Errv* MSE.

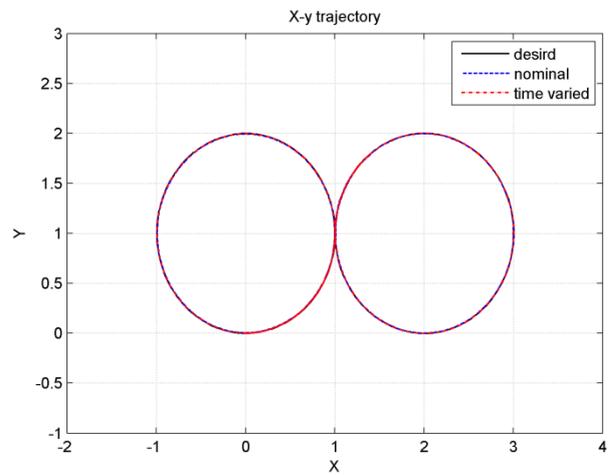


Fig.9. X-Y trajectory tracking.

VII. CONCLUSIONS

This paper proposes novel GRO algorithm optimizing an inverse ANC trajectory tracking wheeled MR. ANC consists of two subs NNs; KNC and DNC using internal and external feedback loops. The internal DNC feedback loop makes the robot more robust against parameters uncertainty and parameter time variations, while the external KNC responsible for tracking the desired angular and linear velocities of the MR and the posture x , y , and ϑ .

Both NNs trained using GRO algorithm. Applying 10 iteration cycles each with 1000 maximum iterations for both of networks. Fig.7 and Fig. 8 have shown the position and velocity MSE, from these figures we notice that the controller succeeds in tracking the MR under fixed nominal and time varying parameters. The ANC performance generally didn't effect by the parameters

variation even when some parameters increased by more than 300%. Therefore, this controller has shown a great performance against parameters variation with only 8 hidden neurons for both KNC and DNC without any overfitting during data training process using the GRO algorithm. Although we have used large training and testing datasets, there was no data overfitting, this is due to the used techniques of observing the training and testing data at the same time. Since the training was offline and have been carried only once (10 cycles) GRO processing time has not been taken into consideration.

REFERENCES

- [1] K. H. Teng, T. Wu, Z. Yang, C. H. Heng and X. Liu, "A 400-MHz wireless neural signal processing IC with 625 on-chip data reduction and reconfigurable BFSK/QPSK transmitter based on sequential injection locking," *2015 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Xiamen, pp. 1-4, 2015.
- [2] Neha Kapoor, Jyoti Ohri, "Evolutionary Optimized Neural Network (EONN) Based Motion Control of Manipulator", *International Journal of Intelligent Systems and Applications IJISA*, vol.6, no.12, pp.10-16, 2014.
- [3] Yevgeniy Bodyanskiy, Olena Vynokurova, Volodymyr Savvo, Tatiana Tverdokhlib, Pavlo Mulesa, "Hybrid Clustering-Classification Neural Network in the Medical Diagnostics of the Reactive Arthritis", *International Journal of Intelligent Systems and Applications (IJISA)*, Vol.8, No.8, pp.1-9, 2016.
- [4] R. Murugadoss and M. Ramakrishnan, "Universal approximation of nonlinear system predictions in sigmoid activation functions using artificial neural networks," *2014 IEEE International Conference on Computational Intelligence and Computing Research*, Coimbatore, pp. 1-6, 2014.
- [5] S. Chai and Y. Zhou, "A Study on How to Help Back-propagation Escape Local Minimum," *Third International Conference on Natural Computation (ICNC 2007)*, Haikou, pp. 64-68, 2007.
- [6] Koffka Khan, Ashok Sahai, "A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context", *International Journal of Intelligent Systems and Applications IJISA*, vol.4, no.7, pp.23-29, 2012.
- [7] Hanan A. R. Akkar, Firas R. Mahdi, "Training Artificial Neural Networks by PSO to Perform Digital Circuits Using Xilinx FPGA," *Eng. & Tech. Journal*, Vol. 29, No. 7 pp. 1329-1344, 2011.
- [8] R. Fierro and F. Lewis, "Control of a Nonholonomic Mobile Robot Using Neural Networks," *IEEE Transactions on neural networks*, vol. 9, pp. 589-600, 1998.
- [9] X. Jiang, Y. Motai and X. Zhu, "Predictive Fuzzy Logic Controller for Trajectory Tracking of a Mobile Robot," *Proceedings of the IEEE Midnight-Summer Workshop on Soft Computing in Industrial Applications. SMCia/05.*, pp. 29-32, 2005.
- [10] R. Martinez-Soto, O. Castillo, L. T. Aguilar and I. S. Baruch, "Bio-Inspired Optimization of Fuzzy Logic Controllers for Autonomous Mobile Robots," *Fuzzy Information Processing Society (NAFIPS), Annual Meeting of the North American*, Berkeley, CA, pp. 1-6, 2012.
- [11] Raimúndez and A. B. Blas, "Adaptive Tracking in Mobile Robots with Input-Output Linearization," *Industrial*

Electronics Society, IECON 39th Annual Conference of the IEEE, Vienna, pp. 3299-3304, 2013.

- [12] Stickler. Grass Growth and Development. *Texas Cooperative Extension*, Texas A&M University.
- [13] Gaurang Panchal, Amit Ganatra, Y P Kosta and Devyani Panchal, "Behaviour Analysis of Multilayer Perceptrons with Multiple Hidden Neurons and Hidden Layers," *International Journal of Computer Theory and Engineering*, Vol. 3, No. 2, April 2011.
- [14] T. Fukao, H. Nakagawa, "Adaptive Tracking Control of a Nonholonomic Mobile Robot," *IEEE transactions on Robotics and Automation*, vol. 16, pp. 609-615, 2000.
- [15] Kolmanovsky and N. H. McClamroch, "Developments in nonholonomic control problems," in *IEEE Control Systems*, vol. 15, no. 6, pp. 20-36, Dec 1995.

Authors' Profiles



Hanan A. R. Akkar received her Bachelor's Degree from the Electrical and Electronics Engineering Department at the University of Technology in 1988. She received her Master's degree and Ph.D. degree from the Electrical and Electronics Engineering Department at the University of Technology in 1994 and 1998,

respectively. She has been Professor in the Department of Electrical Engineering at the University of Technology in the field of ANN, FL, GA and swarm intelligent based on FPGA and electronic circuits. Currently she is Head of scientific committee in the Electrical Engineering Department at University of Technology.



Firas R. Mahdi received his Bachelor's Degree from the Electrical and Electronics Engineering Department at the University of Technology in 2004 Iraq, Baghdad. He received his Master's degree and Ph.D. degree from the same university, at the Electrical and Electronics Engineering Department in 2008 and 2017, respectively.

His major field of study was in the artificial intelligence, evolutionary algorithms, and robotics controllers.

How to site this paper: Hanan A. R. Akkar, Firas R. Mahdi, "Adaptive Path Tracking Mobile Robot Controller Based on Neural Networks and Novel Grass Root Optimization Algorithm", *International Journal of Intelligent Systems and Applications (IJISA)*, Vol.9, No.5, pp.1-9, 2017. DOI: 10.5815/ijisa.2017.05.01