Received: 10 May 2015 | Revised: 13 June 2015 | Accepted: 18 Jun 2015

DOI: 10.37917/ijeee.11.1.14 | Vol. 11| Issue 1 | June 2015

Open Access

Iraqi Journal for Electrical and Electronic Engineering

Original Article

Fuzzy Petri Net Controller for Quadrotor System using Particle Swam Optimization

Mohammed J. Mohammed
Electrical Engineering Department
University of Basrah
Basrah, Iraq
mdiraq90@gmail.com

Abduladhem A. Ali
Computer Engineering Department
University of Basrah
Basrah, Iraq
abduladhem@ieee.org

Mofeed T. Rashid
Electrical Engineering Department
University of Basrah
Basrah, Iraq
mofeed.t.rashid@ieee.org

Abstract: *In this paper, fuzzy Petri Net controller is used for Quadrotor system. The fuzzy Petrinet controller is arranged in the velocity PID form. The optimal values for the fuzzy Petri Net controller parameters have been achieved by using particle swarm optimization algorithm. In this paper, the reference trajectory is obtained from a reference model that can be designed to have the ideal required response of the Quadrotor, also using the quadrotor equations to find decoupling controller is first designed to reduce the effect of coupling between different inputs and outputs of quadrotor. The system performance has been measured by MATLAB. Simulation results showed that the FPN controller has a reasonable robustness against disturbances and good dynamic performance.*

Index Terms— ***Quadrotor; Fuzzy Petri Net (FPN); Particle Swarm Optimization (PSO); MATLAB simulation***

I. Introduction

A Quadrotor is aircraft that has four motors fixed in the body frame; two of these motors rotate in clockwise while the others rotate in anti-clockwise, the motion direction can be controlled by tuning the speed of motors [1, 2].

Petri nets are networks have great characteristics of combining a well-defined mathematical theory with a graphical representation of the dynamic behaviour of systems. The theoretical aspect of Petri nets allows precise modelling and analysis of system behaviour, at the same time, the graphical representation of Petri nets enable visualization of state changes of the modelled system [3, 5].

Therefore, Petri nets are familiar as one of sufficient and sound tool for description and analysis of concurrent, asynchronous and distributed dynamical system. The Fuzzy Petri net (FPN is expanded from a Petri net is a bidirectional graph that has place and transition nodes like the Petri net, however, in FPN a token incorporated with a place is associated with a real value between 0 and 1; a transition is associated with a certain factor (CF) between 0 and 1. Fuzzy Petri net is a promising modelling tool for expert systems. It have been used for many control applications [4, 7-10]. In this paper, Quadrotor control system using FPN is investigated. The parameters of the FPN controller are tuned using Particle Swarm Optimization (PSO). The mathematical modelling of Quadrotor is first discussed, the structure of the FPN controller and PSO algorithm for parameter tuning is then presented. The system simulation is finally carried out to compare the performances of the proposed controller with PID controller.

II. Quadrotor Modelling

The Quadrotor model is consisting of four rotors motorized by electrical motors and fixed at each corner of the + frame as shown in Fig. 1. The motors (M1 and M3) are rotating in the same direction (clockwise) while motors (M2 and M4) are rotating in the other direction (counter clockwise). The motion and direction of Quadrotor can be controlled by varying the speed of these motors [1, 11, and 12].

Linear acceleration in X-axis direction:

$$\ddot{x} = \frac{u_1(\sin\varphi\sin\phi + \cos\varphi\cos\phi)}{m_o} - k_1\dot{x} \qquad (1)$$

Linear acceleration in Y-axis direction:

$$\ddot{y} = \frac{u_1(\sin\varphi\sin\theta\cos\phi + \sin\varphi\sin\phi)}{m_o} - k_1\dot{y} \qquad (2)$$

Linear acceleration in Z-axis direction:

$$\ddot{z} = \frac{u_1(\sin\varphi\sin\phi)}{m_o} - g - k_3\dot{z} \qquad (3)$$

Rolling angular acceleration in the X- directions:

$$\ddot{\phi} = \dot{\theta}\dot{\varphi}I_x - \frac{J_r}{I_{xx}}\dot{\theta}\omega + \frac{l}{I_{xx}}u_2 - k_4\dot{\phi} \quad (4)$$

Pitching angular acceleration in the X- directions:

$$\ddot{\theta} = \dot{\theta}\dot{\varphi}I_y - \frac{J_r}{I_{yy}}\dot{\phi}\omega + \frac{l}{I_{yy}}u_3 - k_5\dot{\theta} \quad (5)$$

Yawing angular acceleration in the X- directions:

$$\ddot{\varphi} = \dot{\theta}\dot{\varphi}I_z + \frac{l}{I_{zz}}u_4 - k_6\dot{\phi} \quad (6)$$

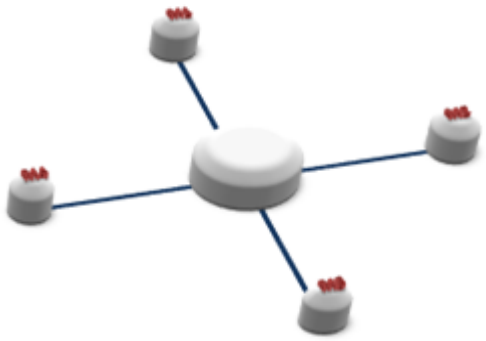


Fig.1: Top view of Quadrotor

Where J_r is moment of inertia of Quadrotor, I_{xx}, I_{yy} and I_{ZZ} are moment inertia of a Quadrotor for directions (X, Y and Z) [1,2]. The moment inertia of Quadrotor can be found by:

$$J_r = 2\left[\frac{ml^2}{4} + \frac{mh^2}{12}\right] + \frac{m_o R^2}{4} + \frac{m_o H^2}{12} \quad (7)$$

The moment inertia about X, Y and Z-axis are approximated by equations below:

$$I_{xx} = \frac{m\rho^2}{2} + \frac{mh^2}{2} + 2ml^2 + \frac{m_o R^2}{4} + \frac{m_o H^2}{12} \quad (8)$$

$$I_{yy} = \frac{m\rho^2}{2} + \frac{mh^2}{2} + 2ml^2 + \frac{m_o R^2}{4} + \frac{m_o H^2}{12} \quad (9)$$

$$I_{zz} = 4ml^2 + \frac{m_o R^2}{2} \quad (10)$$

Where m, h, ρ is mass, height and radius of the motors, and m_o, H, R is mass, height and radius of Quadrotor, l is radius of rotation and I_x, I_y, and I_z are described by

$$I_x = \frac{I_{yy} - I_{zz}}{I_{xx}} \quad (11)$$

$$I_y = \frac{I_{zz} - I_{xx}}{I_{yy}} \quad (12)$$

$$I_z = \frac{I_{xx} - I_{yy}}{I_{zz}} \quad (13)$$

Furthermore u_1, u_2, u_3, and u_4 are inputs of Quadrotor and ω is a disturbance in speed of Quadrotor that can be created by [2, 12]

$$u_1 = b\left(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2\right) \quad (14)$$

$$u_2 = b\left(\omega_4^2 - \omega_2^2\right) \quad (15)$$

$$u_3 = b\left(\omega_3^2 - \omega_1^2\right) \quad (16)$$

$$u_4 = d\left(\omega_2^2 + \omega_4^2 - \omega_1^2 + \omega_3^2\right) \quad (17)$$

$$\omega = \omega_2 + \omega_4 - \omega_1 - \omega_3 \quad (18)$$

Where $\omega_1, \omega_2, \omega_3$ and ω_4 are the speeds of motors of Quadrotor. The drag factor and thrust factor have been calculated in form:

$$b = c_T \rho A R^2 \quad (19)$$

$$d = c_Q \rho A R^3 \quad (20)$$

Limiter is used to simulate the saturation effect of motors in maximum speed after using square root to get S_1, S_2, S_3, and S_4 that researchers neglected its impact on Quadrotor.

$$w_1 = \begin{cases} 0 & if \quad S_1 > 1000 \\ S_1 & if \quad 0 < S_1 < 1000 \\ -1000 & if \quad S_1 < 0 \end{cases} \quad (21)$$

$$w_2 = \begin{cases} 0 & if \quad S_2 > 1000 \\ S_2 & if \quad 0 < S_2 < 1000 \\ -1000 & if \quad S_2 < 0 \end{cases} \quad (22)$$

$$w_3 = \begin{cases} 0 & if \quad S_3 > 1000 \\ S_3 & if \quad 0 < S_3 < 1000 \\ -1000 & if \quad S_3 < 0 \end{cases} \quad (23)$$

$$w_4 = \begin{cases} 0 & if \quad S_4 > 1000 \\ S_4 & if \quad 0 < S_4 < 1000 \\ -1000 & if \quad S_4 < 0 \end{cases} \quad (24)$$

Where b is thrust factor in hover (NS^2), d is drag factor in hover (NmS^2), c_T is the thrust coefficient, c_Q is drag coefficient.

III. Fuzzy Petri Net (FPN)

The structure of the proposed Fuzzy Petri Net is shown in Fig. (2). the network has the following three layers [8, 13]:

- Input layer composed of n input places.

- Transition layer composed of hidden transitions.
- Output layer consisting of m output places.

The input place is marked by the value of the feature. The transitions act as processing units. The firing depends on the parameters of transitions, which are the thresholds, and the parameters of the arcs (connections) are the weights. The marking of the output place reflects a level of membership of the pattern in the corresponding class.

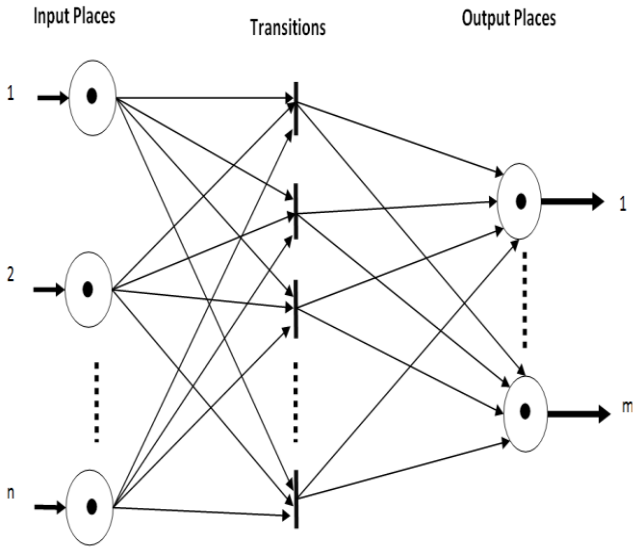


Figure (2): The Structure of FPN

The network specification is shown in Fig. (3) as follows [13]: P_j is the marking level of j-th input place produced by a triangular mapping function. The top of the triangular function is centred on the average point of the input values. The length of triangular base is calculated from the difference between the minimum and maximum values of the input. The height of the triangle is unity. This process keeps the input of the network within the period [0, 1]. This generalization of the Petri net will be in full agreement with the two-valued generic version of the Petri net.

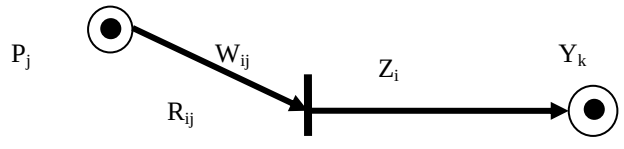


Figure (3): A section of the net outlines the notations.

$$P_j = f(input(j)) \tag{25}$$

Where f is a triangular mapping function which is shown in Fig. (4).

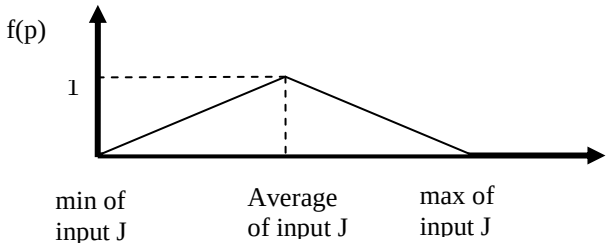


Figure (4): The triangular mapping function

W_{ij} is the weight between the i-th transition and the j-th input place, r_{ij} is a threshold level associated with the level of marking of the j-th input place and the i-th transition, Z_i is the activation level of i-th transition and defined as follows [15]:

$$f(x) = \begin{cases} \dfrac{x - \min(x)}{average(x) - \min(x)} & if \quad x < average(x) \\ \dfrac{\max(x) - x}{\max(x) - average(x)} & if \quad x > average(x) \\ 1 & if \quad x = average(x) \end{cases} \tag{26}$$

- Petri Net Layer

This layer is used to produce tokens that make use of competition laws for node firing as follows:

$$t_{ij} = \begin{cases} 1 & if \quad \mu_{ij} \geq d_{th} \\ 0 & f \quad \mu_{ij} < d_{th} \end{cases} \tag{27}$$

Where t_{ij} is the transition and d_{th} is the dynamic threshold that varies with error and can be tuned by the following equation [15]:

$$d_{th} = \frac{\alpha e^{(-\beta E)}}{1 + \alpha e^{(-\beta E)}} \tag{28}$$

Where α and β are positive constants that can be chosen randomly. It is clear that the larger the error is, the smaller the threshold is. If the error becomes large, the threshold values will be decreased to fire more rules for the current situation. Of course one can use a constant value for the threshold. It is important to mention that if the threshold value is chosen to be 0, then the FPN system will transformed to FNN system.

- Rule Layer

The output of each node is the product of its inputs and it is given by:

$$\phi_j = \begin{cases} \Pi_i^n \mu_{ij} & if \quad t_{ij} = 1 \\ 0 & if \quad t_{ij} = 0 \end{cases} \tag{29}$$

Where $\emptyset_j$ is the output of the j_{th} node of the rule layer; n is the number of crisp inputs.

- Output Layer

The output node calculates the total output y as a summation of the input signals as follows:

$$y = \sum_{j}^{ni} w_j \phi_j \tag{30}$$

Where the connection weights w_j is the output action strength associated with the j_{th} rule; n_i is the number of rules.

IV. CONTROL SYSTEM DESIGN

The PID controllers are still usable today to control Quadrotor systems. In general, PID controller is a direct controller placed in the feed forward path of the system, it receives the error signal and produces control command according to the value of the error signal. In this type of control systems, plant response should follow the response of a defined system.

In this paper, the basic concept of PN is used with fuzzy to create the presented FPN controller for Quadrotor control system. The proposed FPN control system is shown in Fig. (5).This control system, which is based on Model Following Control (MFC). MFC design is based on designing a controller such that the output of the plant follows a per specified trajectory defined by the response of a reference model. The FPN controller main task is to reduce the error between the plant and model output as defined by:

$$E_{ci} = \frac{1}{2} e_{ci} = \frac{1}{2} (y_{ri} - y_{pi})^2 \tag{31}$$

which is E_{ci} is the ith error square vector element, $e_{ci}$is the ith element of the output vector, $y_{ri}$is the i_{th} element of the reference output vector and $y_{pi}$is the ith element of the output vector.

- **Decupling controller**

To simplify the planning problem, the planning separately for each of the coordinates z, ϕ, θ, and φ, they are coupled in the acceleration constraint and the rotational control inputs. In order to allow decoupling, the residual dynamics, not directly controlled, is known as the internal dynamics. If the internal dynamics are stable, decupling controller is successful. This means that tracking cannot always be guaranteed for the original outputs of interest. Fig. 6 shows the details of decupling controller.

Decupling controller is found in from.

$$S_1^2 = \frac{v_1}{4b} - \frac{v_5}{2b} - \frac{v_4}{4d} \tag{32}$$

$$S_2^2 = \frac{v_1}{4b} - \frac{v_2}{2b} + \frac{v_4}{4d} \tag{33}$$

$$S_3^2 = \frac{v_1}{4b} + \frac{v_5}{2b} - \frac{v_4}{4d} \tag{34}$$

$$S_4^2 = \frac{v_1}{4b} + \frac{v_2}{2b} + \frac{v_4}{4d} \tag{35}$$

where $S_1^2, S_2^2, S_3^2, S_4^2$ are the output of the decupling controller.

V. LEARNING ALGORITHM OF FPN

The task of constructing the FPN is divided into two subtasks: structure learning and parameter learning. In the structure learning, the number of fuzzy rules, initial location of membership functions, and initial consequent parameters are chosen randomly. The parameter learning is used to tune the free parameters of the constructed network to its optimal values; the learning process is based on PSO. The PSO algorithm works by simultaneously maintaining several candidate solutions in the explore space. In each iteration of the algorithm, each candidate solution that can be thought of as a particle flies through the search space to find the maximum or minimum of the cost function (fitness value) [12, 13].

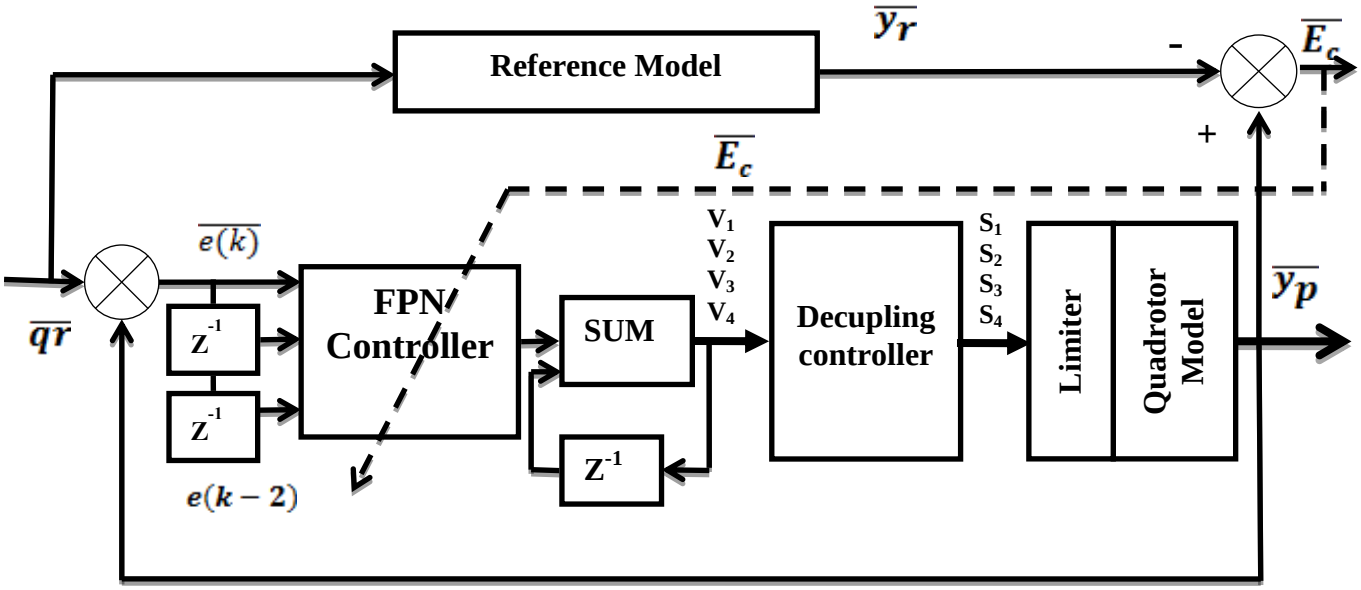


Figure (5) FPN forward control

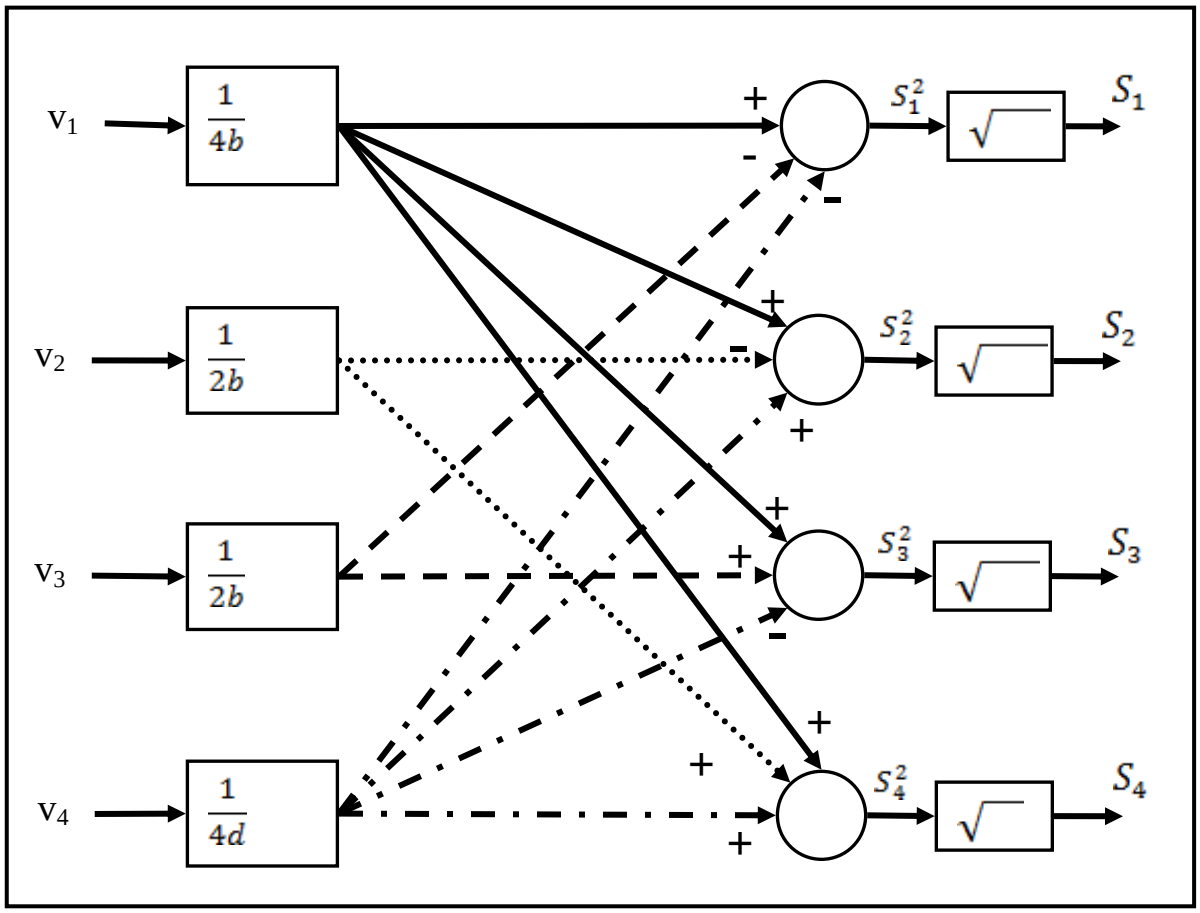


Figure (6) Decupling controller

Initially, the PSO algorithm chooses candidate solutions randomly within the search space. Each particle maintains its position, collected of the candidate result, its evaluated cost function, and its velocity. Additionally, it remembers the best fitness value it has achieved thus far during the operation of the algorithm, referred to as the individual best fitness, and the candidate solution that achieved this fitness, referred to as the individual best position.

Finally, the PSO algorithm maintains the best fitness value achieved among all particles in the swarm, called the global best fitness, and the candidate result that achieved this fitness, called the global best position. For a multidimensional problem, the velocity and position of each particle in the swarm are updated using the following equations: [14]

$$v_{j,g}(t+1) = \omega v_{j,g}(t) + c_1 r_1 \left[pdest_{i,g}(t) - x_{j,g}(t) \right] + c_2 r_{12} \left[gbest_{i,g}(t) - x_{j,g}(t) \right] \quad (36)$$

$$x_{j,g}(t+1) = x_{j,g}(t) + v_{j,g}(t+1) \quad (37)$$

where j=1, 2… n and g=1, 2… m, n is the number of particles in the swarm (or population), m is the number of members in a particle (dimension of problem), t is the current iteration number, $v_{j,g}(t)$ is the velocity of particle j(dimension g) at iteration t, ω is the inertia weight factor, $x_{j,g}(t)$ is the current position of particle j at iteration t,$pbest_{j,g}(t)$ is the individual best position of particle j until iteration t, $gbest_{i,g}(t)$ is the best particle in the swarm at iteration t.

The parameters ω, c_1, andc_2 are user-supplied coefficients. The values r_1 and r_2 ($0 \le r_1 \le 1$ and $0 \le r_2 \le 1$) are random values regenerated for each velocity update. Fig. (7)shows the flowchart of the PSO algorithm.

VI. CALIBRATION AND SIMULATION

The quadrotor controller has been simulated by MATLAB in order to investigate its performance. Fig. 7 shows the PSO flowchart for tunning FPN parametres.

Table 1 shows the list of various quantities used for calibrate Quadrotor model. Both FPN controller and PID controller is simulated to achieve the robostness of both controllers.

Fig.8 shows the paths of the Quadrotor in different positions under PID controller and FPN controller and measured the difference between two different methods of control.

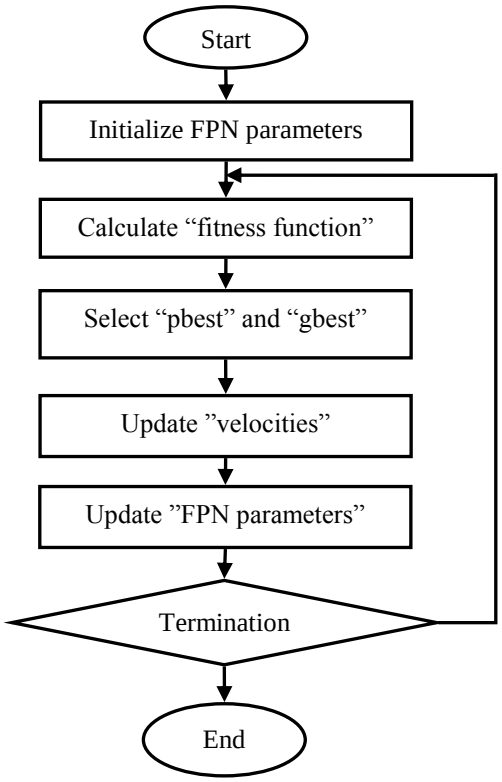


Figure (7): PSO flowchart for tuning FPN parameters[12]

Table1: Calibration Data for the Quadrotor

	Parameter Name	Symbol	Numerical Value	Unit
1	Rotational inertia along x-axis, y-axis	I_{xx},I_{yy}	0.0019	$Kg.m^2$
2	Rotational inertia along z-axis	I_{zz}	0.0033	$Kg.m^2$
3	Rotor inertia	J_r	0.0099	$Kg.m^2$
4	Motor mass	m	30	G
5	Total mass	m_o	1. 25	Kg
6	Pro area	A	0.3	m^2
7	Prop radius	R_p	0.15	M
8	Arm length	L	0.96	M
9	Air density	ρ_a	1.1	Kgm^{-3}
10	Height motor	h	0.02	m
11	Height Quad rotor	H	0.10	m
12	Drag constant	D	7.5E-7	$N.s^2$
13	Acceleration due to gravity	g	9.81	$m.s^{-2}$

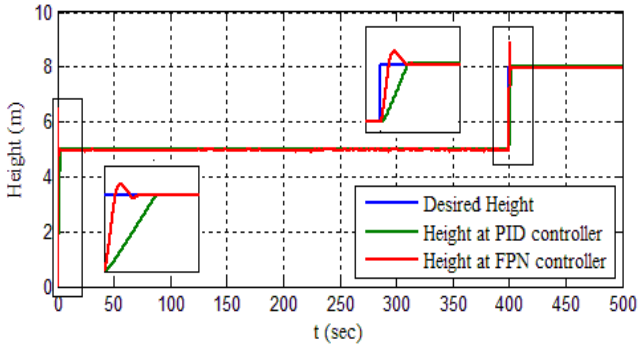
Fig.8a shows the height response at using PID controller and the height response at using FPN controller. in Fig. 8a the height change in t=0sec to 5m as step response, also in t=400sec the height to 8m at step response, the FPN controller faster than PID but also has small overshoot=14%, the stealing time in PID controller is 5sec and FPN controller 3.5 sec, the steady state error go to in both controllers.

Fig.8b shows the roll angle response at using PID controller and FPN controller, in Fig.8b the roll angle is changed in t=100sec to 0.4 rad (22.9^o) as step response. They have an overshoot, but overshoot in FPN controller is smaller and slower than over shoot in PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) as step response to show effect changing height and angle at same time.

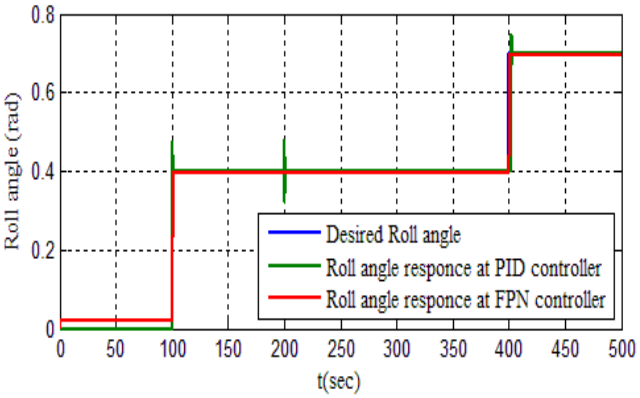
Fig. 8c shows the pitch angle response at using PID controller and using FPN controller, in Fig.8c the pitch angle change in t=200sec to 0.4 rad (22.9^o) as step response, PID controller has overshoot, FPN controller response does not have overshoot, but also the response slower than PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time.

Fig.8d shows the yaw angle response at using PID controller and using FPN controller in Fig.8d the yaw angle change in t=300sec to 0.4 rad (22.9^o) as step response, PID controller has overshoot, FPN controller response does not have overshoot, but also the response slower than PID controller, also in t = 400 sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time.

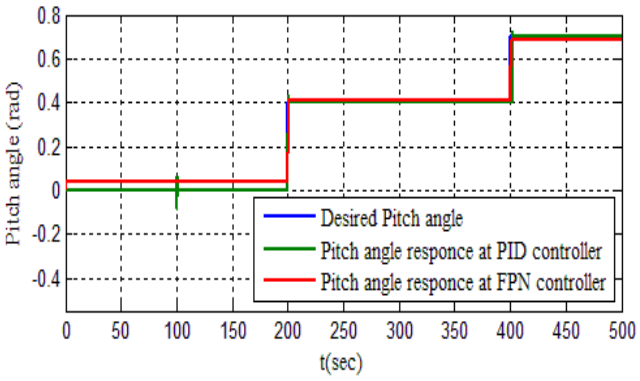
At using PID controller, when roll angle is changed from 0 rad to 0.4 rad, a small change is happed in pitch angle and yaw angle. When pitch angle is changed from 0rad to 0.4rad, a small change is happed in roll angle and yaw angle. When yaw angle is changed from 0rad to 0.4rad, nothing is happed to roll angle and pitch angle. At FPN controller, when any angle change, nothing is happened into other angles, as example when roll angle is changed, nothing is happened in pitch angle and yaw angle, FPN is better than PID controller.



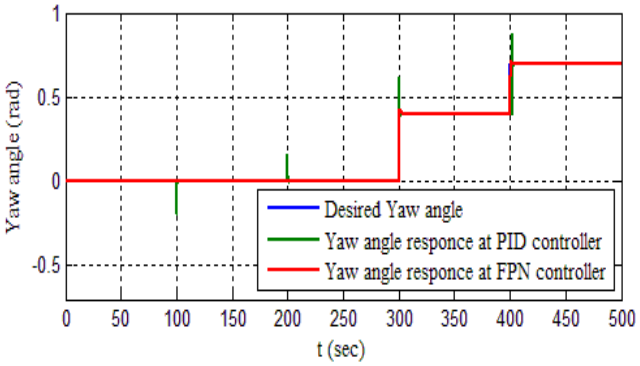
(a) The Trained Response After 1000 Epochs



(b) Roll angle response After 1000 Epochs



(c) Roll angle response After 1000 Epochs



(d) Roll angle response After 1000 Epochs

Figure (8): Response of Quadrotor for 1000 Epochs updating parameters

Fig.9 shows the quadrotor travelling with taking wind effect into consideration PID controller and FPN controller, the wind speed is 30 Km/H.

Fig.9a shows the height response at using PID controller and Fig.9b shows the height response at using FPN controller in two figures the height change in t=0sec to 5m as step response, also in t=400sec the height to 8m at step response, the FPN controller faster than PID but also has small overshoot=19%, the stealing time in PID controller is 7sec and FPN controller 5.5 sec.

Fig.9c shows the roll angle response at using PID controller and Fig.9d shows the roll angle response at using FPN controller in two figures the roll angle change in t=100sec to 0.4 rad (22.9^o) as step response, overshoot in FPN controller is smaller than over shoot in PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response same as in Fig.8. In additional to these small effect over PID controller and no effect in FPN controller, FPN controller is better than PID controller.

Fig.9e shows the pitch angle response at using PID controller and Fig.9f shows the pitch angle response at using FPN controller in two figures the pitch angle change in t=200sec to 0.4 rad (22.9^o) as step response, PID controller has overshoot, FPN controller response does not have over shoot, but also the response slower than PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time. In additional to these small effect over PID controller and FPN controller smaller but put steady start error 0.02 rad (0.8^o), FPN controller is better than PID controller.

Fig.9g shows the yaw angle response at using PID controller and Fig.9h shows the yaw angle response at using FPN controller in two figures the yaw angle change in t=300sec to 0.4 rad (22.9^o) as step response, PID controller has overshoot, FPN controller response does not have over shoot, but also the response slower than PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time. In additional to this small effect over PID controller and FPN controller smaller but put steady start error 0.03 rad (1.5^o), FPN controller is better than PID controller.

At using PID controller, when roll angle is changed from 0rad to 0.4rad, a small change is happed in pitch angle and yaw angle. When pitch angle is changed from 0rad to 0.4rad, a small change is happed in roll angle and yaw angle. When yaw angle is changed from 0rad to 0.4rad, nothing is happed to roll angle and pitch angle.

At FPN controller, when any angle change, nothing is happened into other angles, as example when roll angle is changed, nothing is happened in pitch angle and yaw angle, FPN is better than PID controller.

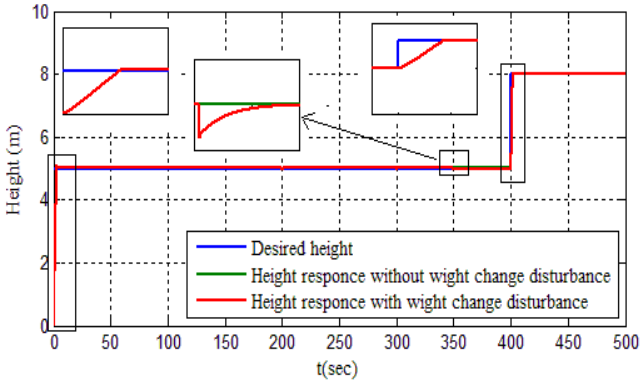
Fig.10 shows the quadrotor flight with taken changing weight disturbance (removing load) effect with PID controller and FPN controller, at t=350 sec the load has been left, the removing load is 0.6 kg .

Fig.10a shows the height response at using PID controller and Fig.10b shows the height response at using FPN controller in two figures the height change in t=0sec to 5m as step response , also in t=400sec the height to 8m at step response, the FPN controller faster than PID but also has small overshoot=11%, the stealing time in PID controller is 6sec and FPN controller 4.5 sec, the steady state error go to in both controllers, height PID response has small error at=350sec when the load is removed.

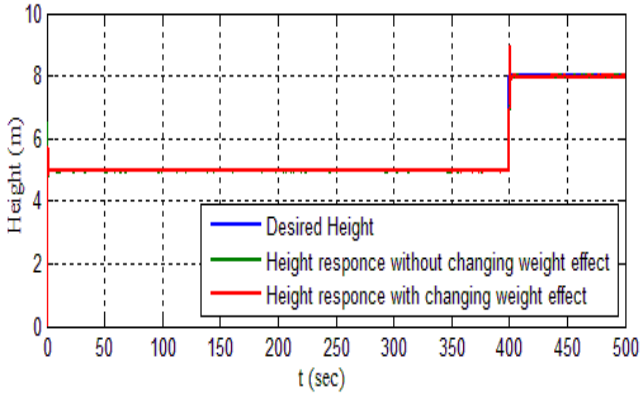
Fig.10c shows the roll angle response at using PID controller and Fig.10d shows the roll angle response at using FPN controller in two figures the roll angle change in t=100sec to 0.4 rad (22.9^o) as step response, they have an overshoot, overshoot in FPN controller is smaller and slower than over shoot in PID controller, also in t=400sec the roll angle to 0.7 rad (40.1_o) at step response to show effect changing height and angle at same time. In additional to these small effect over PID controller, FPN response has high state error 0.1 rad (5.7^o), PID controller is better than FPN controller.

Fig.10e shows the pitch angle response at using PID controller and Fig.10f shows the pitch angle response at using FPN controller in two figures the height change in t=200sec to 0.4 rad (22.9^o) as step response, PID controller has overshoot, FPN controller response does not have over shoot, but also the response slower than PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and

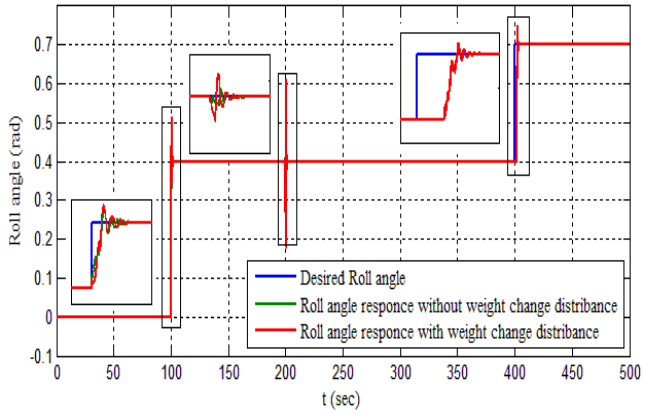
angle at same time. In additional to these small effect over PID controller and FPN controller has stealing time is 25sec, PID controller is better than FPN controller.



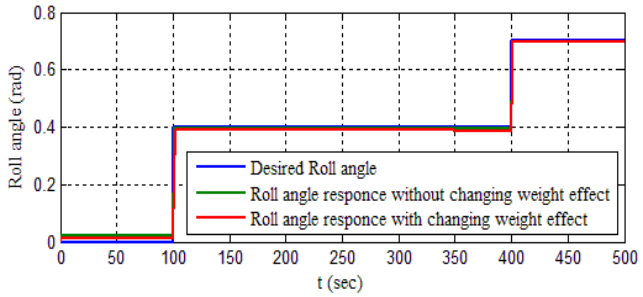
(a) Height at PID controller



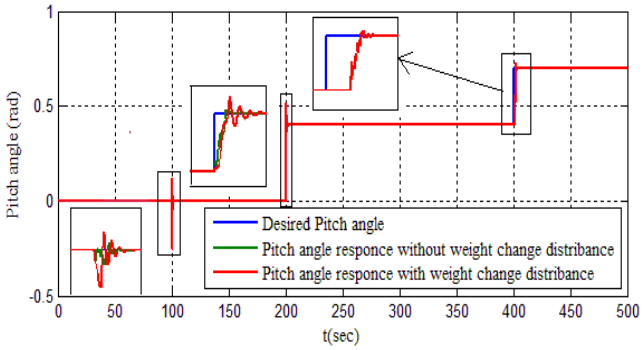
(b) Height at FPN controller



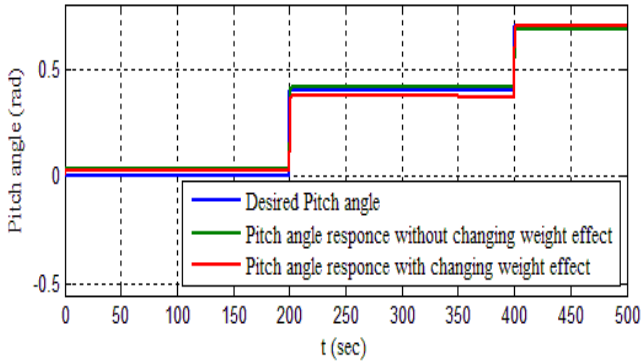
(c) Roll angle at PID controller



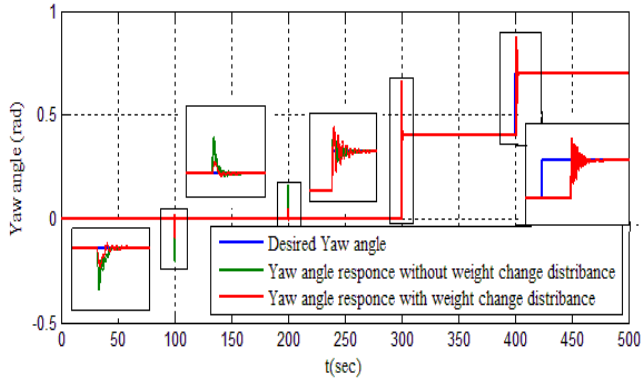
(d) Roll angle at FPN controller



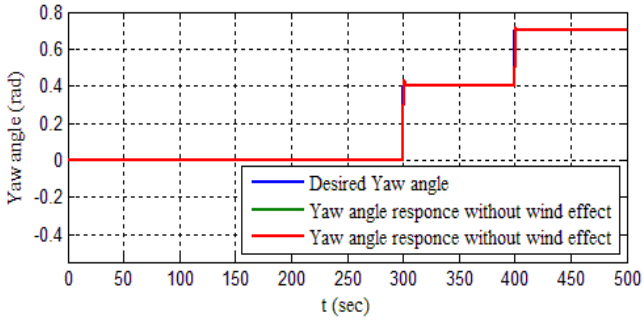
(e) Pitch angle at PID controller



(f)Pitch angle at FPN controller

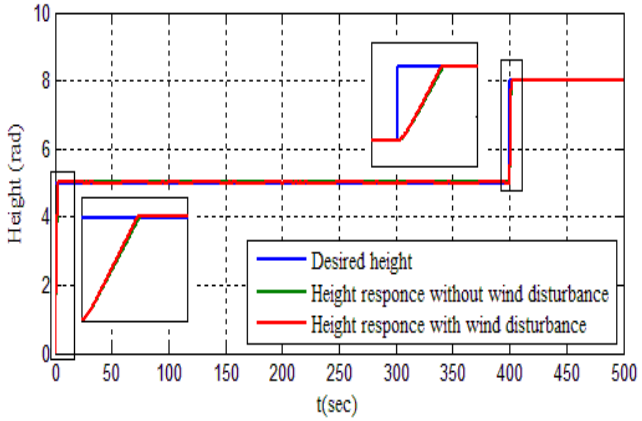


(g) Yaw angle at PID controller

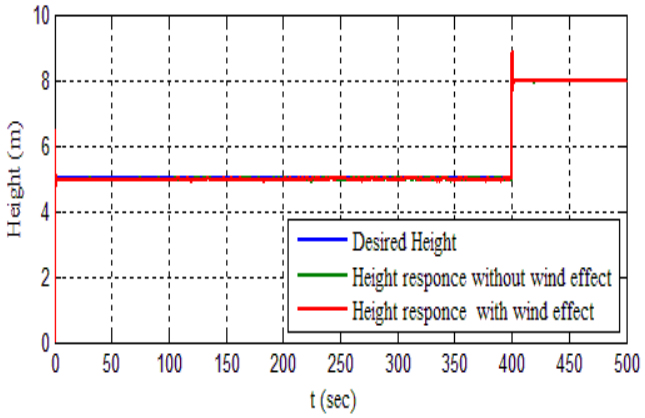


(h) Yaw angle at FPN controller

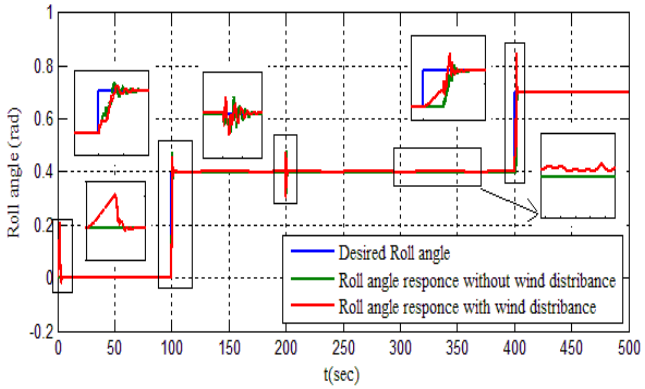
Figure (9): Response of Quadrotor at wind disturbance.



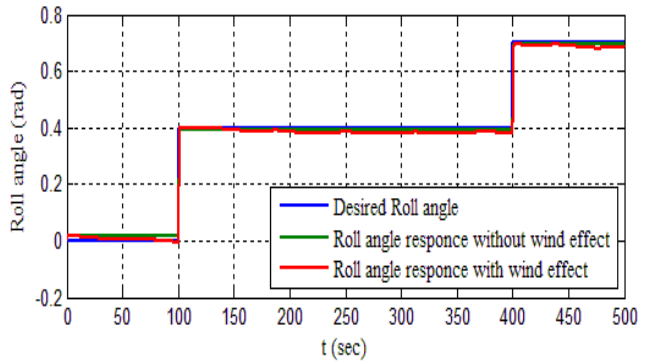
(a) Height at PID controller



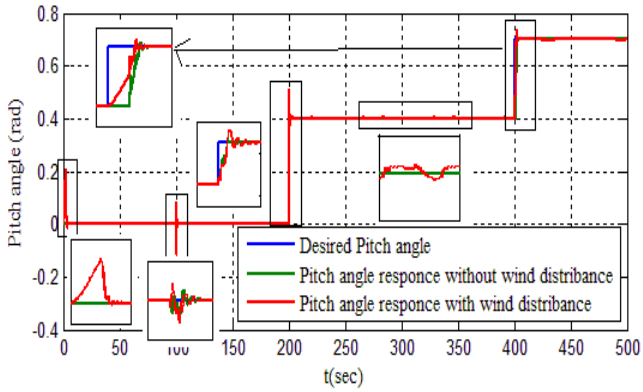
(b) Height at FPN controller



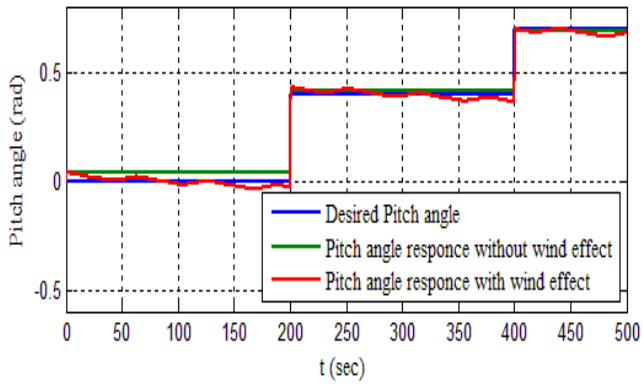
(c) Roll angle at PID controller



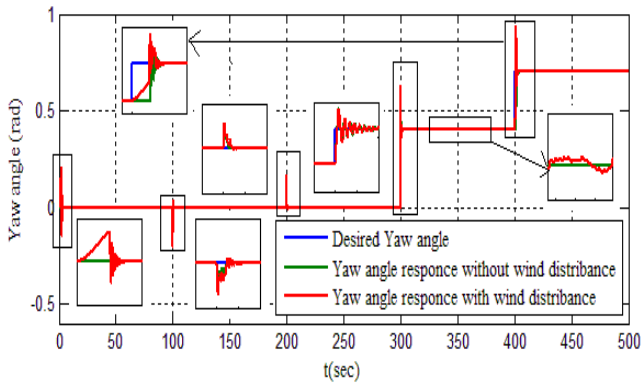
(d) Roll angle at FPN controller



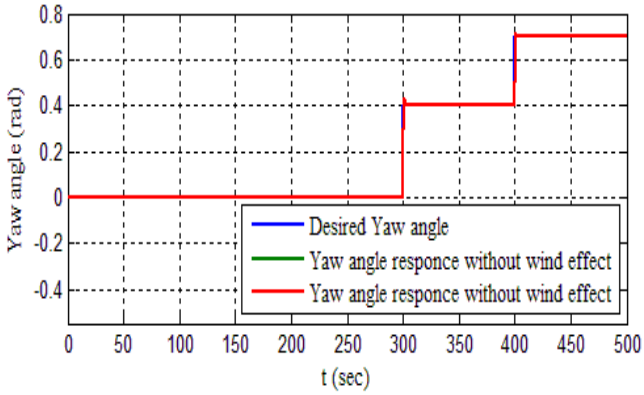
(e) Pitch angle at PID controller



(f)Pitch angle at FPN controller



(g) Yaw angle at PID controller



(h) Yaw angle at FPN controller

Figure (10): Responses of Quadrotor at changing weight disturbances.

Fig.10g shows the yaw angle response at using PID controller and Fig.10h shows the yaw angle response at using FPN controller in two figures the height change in t=300sec to 0.4 rad (22.9^o) as step response PID controller has high overshoot (63%), FPN controller response does not have over shoot, but also the response slower than PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time. FPN controller is better than PID controller. In additional to this small effect over PID controller and FPN controller smaller but put steady start error 0.03 rad (1.5^o), FPN controller is better than PID controller.

At using PID controller, when roll angle is changed from 0rad to 0.4rad, a small change is happed in pitch angle and yaw angle. When pitch angle is changed from 0rad to 0.4rad, a small change is happed in roll angle and yaw angle. When yaw angle is changed from 0rad to 0.4rad, nothing is happed to roll angle and pitch angle. At FPN controller, when any angle change, nothing is happened into other angles, as example when roll angle is changed, nothing is happened in pitch angle and yaw angle, FPN is better than PID controller.

Fig.11 the quadrotor travelling with all disturbances at same time wind effect and changing weight effect with PID controller and FPN controller, at t=350 sec the load has been left the wind speed is 30Km/H and removing weight is 0.6kg.

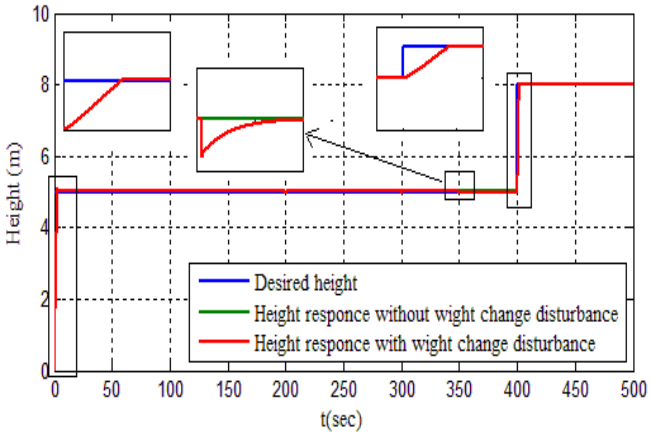
Fig.11a shows the height response at using PID controller and Fig.11b shows the height response at using FPN controller in two figures the height change in t=0sec to 5m as step response , also in t=400sec the height to 8m at step response, the FPN controller faster than PID but also has small overshoot=13%, the stealing time in PID controller is 6.6 sec and FPN controller 5.1 sec, the steady state error go to in both controllers, height PID response has small error at=350sec when the load is removed.

Fig.11c shows the roll angle response at using PID controller and Fig.11d shows the roll angle response at using FPN controller in two figures the roll angle change in t=100sec to 0.4 rad (22.9^o) as step response, they have an overshoot, overshoot in FPN controller is smaller and slower than over shoot in PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time. In additional to these small effect over PID controller, FPN response has high state error 0.08 rad (4.9^o), PID controller is better than FPN controller.

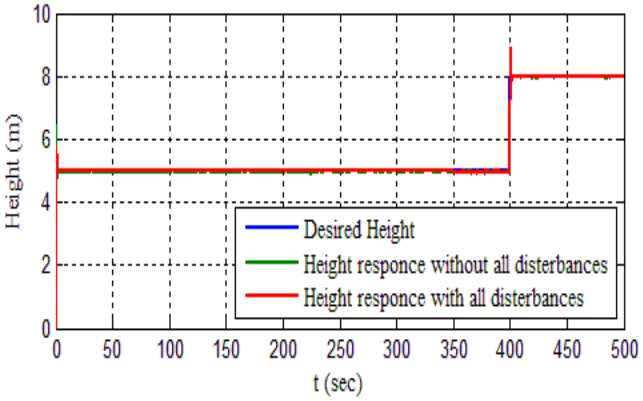
Fig.11e shows the pitch angle response at using PID controller and Fig.11f shows the pitch angle response at using FPN controller in two figures the pitch angle change in t=200sec to 0.4 rad (22.9^o) as step response PID controller has overshoot, FPN controller response does not have over shoot, but also the response slower than PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time. In additional to this small effect over PID controller and FPN controller has stealing time is 12 sec, PID controller is better than FPN controller.

Fig.11g shows the yaw angle response at using PID controller and Fig.11h shows the yaw angle response at using FPN controller in two figures the pitch angle change in t=300sec to 0.4 rad (22.9^o) as step response, PID controller has high overshoot (47%), FPN controller response does not have over shoot, but also the response slower than PID controller, also in t=400sec the roll angle to 0.7 rad (40.1^o) at step response to show effect changing height and angle at same time. FPN controller is better than PID controller.

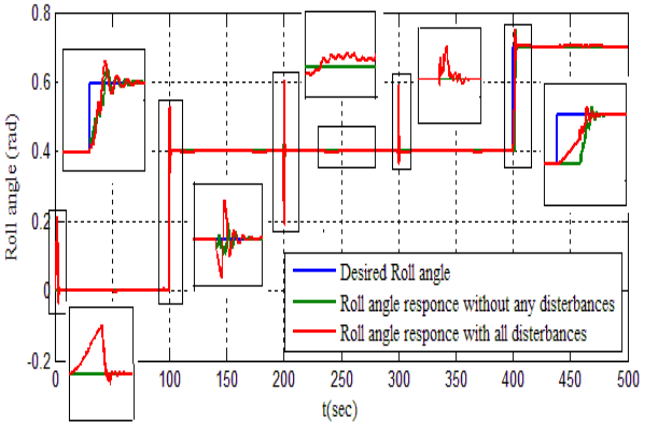
At using PID controller, when roll angle is changed from 0rad to 0.4rad, a small change is happed in pitch angle and yaw angle. When pitch angle is changed from 0rad to 0.4rad, a small change is happed in roll angle and yaw angle. When yaw angle is changed from 0rad to 0.4rad, nothing is happed to roll angle and pitch angle. At FPN controller, when any angle change, nothing is happened into other angles, as example when roll angle is changed, nothing is happened in pitch angle and yaw angle, FPN is better than PID controller.



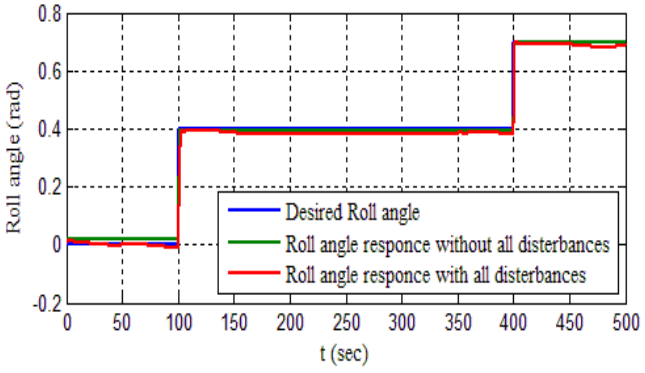
(a) Height at PID controller



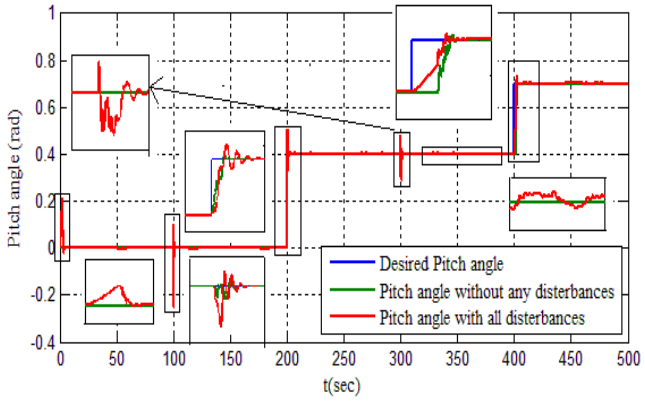
(b) Height at FPN controller



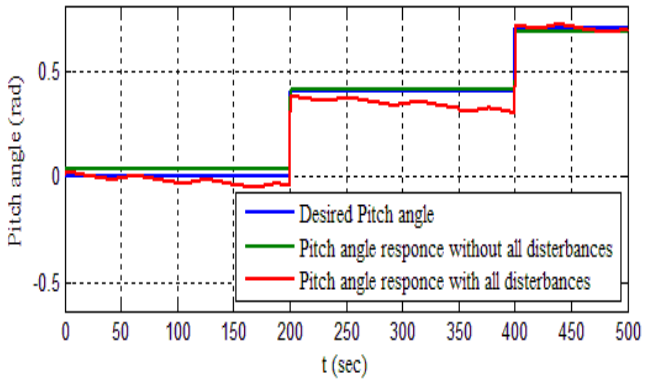
(c) Roll angle at PID controller



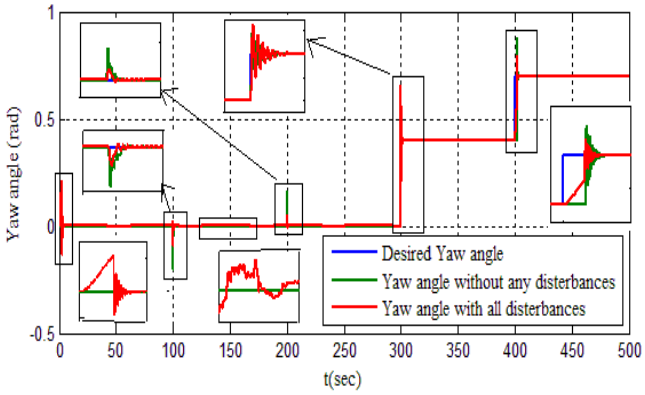
(d) Roll angle at FPN controller



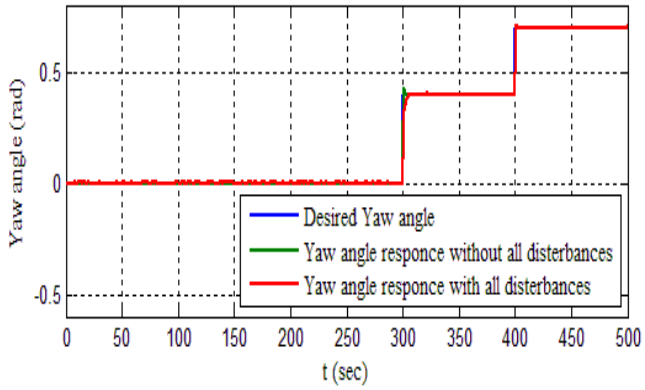
(e) Pitch angle at PID controller



(f)Pitch angle at PID controller



(g) Yaw angle at PID controller



(h) Yaw angle at FPN controller

Figure (11): Responses of Quadrotor at all disturbances.

VII. CONCLUSIONS

In this paper, FPN controller of the Quadrotor is proposed. The parameter learning is performed using PSO to ensure optimal path tracking. The optimal path in this paper is embedded in a reference model. The resulting design is FPN controller with MFC strategy. Simulation of proposed system shows that the designed controller is robust against disturbances. Also, the transient response of the developed controller is much better than conventional PID controller.

REFERENCES

[1] Castillo P., et.al ,"Modelling and Control of Mini-Flying Machines", Springer-Verlag, London, 2005.

[2] Bouabdallah, Samir, Andre Noth, and Roland Siegwart. "PID vs LQ control techniques applied to an indoor micro quadrotor." In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3, pp. 2451-2456. IEEE, 2004.

[3] Dydek, Zachary T., Anuradha M. Annaswamy, and Eugene Lavretsky. "Adaptive control of quadrotor UAVs: A design trade study with flight evaluations." Control Systems Technology, IEEE Transactions on 21, no. 4, pp.1400-1406, 2013.

[4] Li, Sen, Baokui Li, and QingboGeng. "Adaptive sliding mode control for quadrotor helicopters."In Control Conference (CCC), 2014 33rd Chinese, IEEE, pp. 71-76, 2014.

[5] Fakurian, Fardin, Mohammad BagherMenhaj, and AfshinMohammadi. "Design of a fuzzy controller by minimum controlling inputs for a quadrotor." In Robotics and Mechatronics (ICRoM), 2014 Second RSI/ISM International Conference on, IEEE, pp. 619-624, 2014.

[6] Alexis, Kostas, George Nikolakopoulos, and Anthony Tzes. "Experimental constrained optimal attitude control of a quadrotor subject to wind disturbances." International Journal of Control, Automation and Systems 12, no. 6, pp.1289-1302, 2014.

[7] Pounds, Paul EI, and Aaron M. Dollar. "Stability of Helicopters in Compliant Contact Under PD-PID Control." Robotics, IEEE Transactions on 30, no. 6, pp.1472-1486, 2014.

[8] Abed, Ali A., Abduladhem A. Ali, NaumanAslam, and Ali F. Marhoon. "A Robust Neural Fuzzy Petri Net Controller For A Temperature Control System."Procedia Computer Science 5, pp. 881-890, 2011.

[9] Yuan, Minghai, Zhiyong Dai, Shuo Cheng, and AiminJi. "Modeling of Mixed-Model Assembly System based on Agent Oriented Petri Net." Appl. Math 9, no. 1, pp. 369-375, 2015.

[10] Abed, Ali A., Abduladhem A. Ali, Ali F. Marhoon, and NaumanAslam. "A Field Bus Network With CAN Protocol And a Fuzzy Neural Petri Net Controller.", Basrah Journal of Science,31, no.2, pp. 86-102, 2013.

[11] Khatoon, Shahida, Mohammad Shahid, and Himanshu Chaudhary. "Dynamic modeling and stabilization of quadrotor using PID controller."In Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on, IEEE, pp. 746-750, 2014.

[12] Mohammed, M. J., M. T. Rashid, and A. A. Ali. "Design Optimal PID Controller for Quad Rotor System." International Journal of Computer Applications 106 (2014).

[13] Yogendra Kumar Soni and Rajesh Bhatt , 2013, PSO optimized reduced order PID Controller design". International Journal Of Advanced Resarch in Computer Engineering & Technology. Volume 2 Issue 3, ,pp 989-992, ISSN:2278-1323.

[14] Wai, Rong-Jong, and Chia-Chin Chu. "Robust petri fuzzy-neural-network control for linear induction motor drive." Industrial Electronics, IEEE Transactions on54, no. 1, pp. 177-189, 2007.

[15] Kumar, C. Agees, and N. Kesavan Nair. "NSGA-II based multiobjective PID controller tuning for speed control of DC motor drives." Int J Soft Computing 5, no. 3, pp. 83-87, 2010.