Flying Vehicle Longitudinal Controller Design via Prey-Predator Pigeon-Inspired Optimization

Mostafa S. Mohamed, Haibin Duan, *Senior Member, IEEE*, Li Fu
School of Automation Science and Electrical Engineering Beihang University, 100083
Beijing, China
mostafasaad43@gmail.com

***Abstract*— Prey-Predator Pigeon-Inspired Optimization (PPPIO) is a new bio-inspired swarm intelligence algorithm which combines the standard PIO algorithm and the prey predator strategy to improve the optimal solution obtained from the PIO algorithm. PIO algorithm can easily trap into a local optimal solution, which is the main defect that limits its further application. To overcome this defect, a Prey-Predator PIO algorithm is proposed. This paper addresses both PIO and PPPIO in finding the optimal values for control system gains of tactical missile longitudinal autopilot. The control system gains are calculated at first the classical control techniques and then both PIO and PPPIO algorithms are utilized to find out optimal values for these gains which improves system performance and stability margins. Simulation is used to declare the efficiency of each algorithm.**

Keywords— longitudinal autopilot; classical control; missile; Prey-Predator Pigeon-Inspired Optimization (PPPIO); Pigeon-Inspired Optimization (PIO), Proportion-Integral (PI), parameter adjustment; Flying Vehicle.

I. INTRODUCTION

An autopilot is a closed-loop flight control system which is a minor loop inside the main guidance loop [1], [2]. The missile longitudinal autopilot must perform the followings:

- Provide the required missile normal acceleration response which ordered by guidance loop.
- Stabilize or damp the airframe longitudinal acceleration oscillations.
- Reduce the missile performance sensitivity to disturbance inputs over the missile's flight envelope.

Longitudinal autopilot (acceleration controller) in tactical missiles is the heart of a missile autopilot. It consists of body rate and acceleration feedback with Integral controller to provide zero steady-state error, suitable stability margins and performance characteristics [3], [4]. The goal of this paper is a comparison of different autopilot gains calculation methods for a tactical missile. The first method is gains calculation using the classical control techniques. The others are based on both Pigeon-Inspired Optimization (PIO) and Prey-Predator Pigeon-Inspired Optimization (PPPIO) algorithms. The gains of controller in classical control are calculated using root locus and stability analysis whereas both PIO and PPPIO algorithms will tune the values of the controller gains using iteration to reach the optimum desired response. The response should include minimum rise time with no overshoot to guarantee that the normal acceleration does not exceed its desired value. Simulation is created for the system to verify the performance of the designed systems. The remainders of this paper are organized as follows. Section II introduces the mathematical model of the plant.Subsequently, Section III describes classical control technique used for the PI controller design and Section IV describes PIO & PPPIO algorithms for design. Simulation results and analysis are given in section V. Concluding remarks are contained in section VI.

II. MODEL

The linear equations needed for control system design is derived using the small perturbation method from the nonlinear model [5], [6]. Considering first-order lag actuator of transfer function (60/(*s*+60)) and unity gain rate-gyro and accelerometer (as the dynamic characteristics of rate-gyro and accelerometer is very high with respect to missile dynamics) [4], the state-space of normal acceleration autopilot yields:

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\\ \dot{\eta}\end{bmatrix}=\begin{bmatrix} Z_w & 1+\frac{Z_q}{u_0} & \frac{Z_\eta}{u_0}\\ M_\alpha & M_q & M_\eta\\ 0 & 0 & -60\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \eta\end{bmatrix}+\begin{bmatrix}0\\ 0\\ 60\end{bmatrix}u_\eta \tag{1}$$

$$\begin{bmatrix}a_n\\ q\end{bmatrix}=\begin{bmatrix} l_pM_\alpha - Z_\alpha & l_pM_q - Z_q & l_pM_\eta - Z_\eta\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \eta\end{bmatrix}$$

where α is the angle of attack, q is the pitch rate, η is the elevator deflection angle, a_n is the normal acceleration of missile, u_η is the actuator control signal, Z_w, Z_q, Z_η are normal force derivative w.r.t. vertical velocity, pitch rate and elevator deflection angle respectively, M_w, M_q, M_η are pitching moment derivative w.r.t. vertical velocity, pitch rate and elevator deflection angle respectively and u_0 is the undisturbed velocity.

III. CLASSICAL CONTROL TECHNIQUE

The task of the control system is to produce the necessary normal force and maneuver the missile (change the direction of the missile velocity vector) quickly and efficiently as a result of guidance signals [1]. The design point needed at which the controller design will be performed has the state-space model:

978-1-5386-2726-6/17/$31.00 ©2017 IEEE

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\\ \dot{\eta}\end{bmatrix}=\begin{bmatrix}-0.2253 & 0.9977 & -0.00218\\ -20.9856 & -0.4221 & -0.47785\\ 0 & 0 & -60\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \eta\end{bmatrix}+\begin{bmatrix}0\\ 0\\ 60\end{bmatrix}u_\eta \qquad (2)$$

$$\begin{bmatrix}a_n\\ q\end{bmatrix}=\begin{bmatrix}3.645 & -0.01628 & -0.03332\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \eta\end{bmatrix}$$

The autopilot is designed to control the normal acceleration, which its input is a desired normal acceleration *(a_{nc} [g])* command sent from the guidance-loop. The complete normal acceleration control system block diagram is shown in Fig. 1, where the inner loop is the pitch damper system to ensure adequate damping ratio.

From (2), the transfer function of the inner-loop can be deduced as:

$$\frac{q}{e_q}=\frac{-28.671s-3.7147}{s^3+60.6s^2+59.9s+1261.9} \qquad (3)$$

The transfer function has three poles, one real (-60) and two imaginary poles. The two short period complex poles (-0.3237±4.575i) have low damping (ζ_{sp} = 0.0705) and move on the imaginary axis due to increase of K_q and then intersecting on the real axis, one of them moves to the short period zero (-0.13) and the other pole intersects with actuator pole (-60) and moves on the imaginary axis towards infinity. By increasing the value of K_q, the damping ratio of the short period mode increases about 0.7 where the value of (K_q = 11.8) at (ζ_{sp}=0.706) [7], [8]. Then the state-space model will be:

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\\ \dot{\eta}\end{bmatrix}=\begin{bmatrix}-0.2253 & 0.9977 & -0.00218\\ -20.9856 & -0.4221 & -0.47785\\ 0 & 708 & -60\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \eta\end{bmatrix}+\begin{bmatrix}0\\ 0\\ 60\end{bmatrix}u_\eta \qquad (4)$$

$$\begin{bmatrix}a_n\end{bmatrix}=\begin{bmatrix}3.645 & -0.01628 & -0.03332\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \eta\end{bmatrix}$$

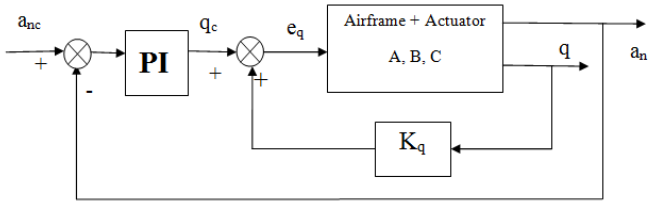

Fig. 1. Normal acceleration control system block diagram

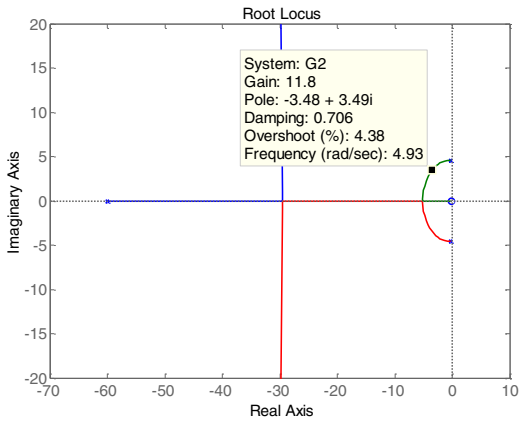

Fig. 2. Root locus of inner-loop

Utilizing integrator controller in the outer-loop as shown in (5) will eliminate the steady state error in normal acceleration.

$$\frac{q_c}{e_{a_n}}=\frac{-K_a}{s} \qquad (5)$$

And the forward path transfer function for the outer-loop is:

$$\frac{a_n}{q_c}=\frac{-2.01s^2-1.312s-146.7}{s^3+60.65s^2+398.2s+1306} \qquad (6)$$

Using *(K_a = 1)*, the dynamic characteristics are shown in Table 1. Increasing of the value of K_a will decrease rise time whereas the overshoot appears and if the overshoot needed to be eliminated, the rise time will increase. The value of K_a is increased and the characteristics are calculated till acceptable requirements reached at (K_a = 11.5) at which the overshoot is lower and the rise time is lower as possible. The results are shown in TABLE I. and shown in Fig. 3.

TABLE I. PERFORMANCE CHARACTERISTICS

Ka	GM [dB]	PM [°]	t_r [sec]	t_s [sec]	M.O. [%]
1	38.9	88.1	26.1	34.2	0
11.5	17.7	68.2	1.04	1.62	1

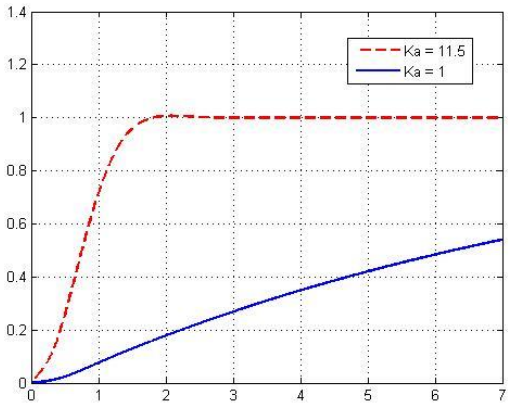

Fig. 3. Normal acceleration step response

IV. Prey-Predator Pigeon-Inspired Optimization(PPPIO) Algorithm

A. Pigeon-Inspired Optimization (PIO) algorithm for PI Controller

Pigeon-Inspired Optimization (PIO) algorithm is a swarm intelligence algorithm inspired by the homing behaviors of pigeons [9],[10],[11]. Homing pigeons can easily find their homes by using three homing tools: magnetic field, sun and landmarks. Pigeons can sense the Eearth field by using magneto receptions to shape the map in their mind. Their compasses to adjust the directions rely on the altitude of the sun. As getting near to the destination, they depend less on magnetic particles and the sun and use landmarks to locate themselves. If they are familiar to the landmarks, they will fly straightly to the destination. If they are unfamiliar with the landmarks, they will follow the pigeons which are familiar with the landmarks.

Map and compass operator model is presented based on magnetic field and the sun. The rules are defined with the position X_i and the velocity V_i of pigeon i, and the positions and velocities in a D-dimension search space are updated in each iteration. The new position X_i and velocity V_i of pigeon i at the *t-th* iteration can be calculated by (7) and (8).

$$V_i(t) = V_i(t\text{-}1).e^{-Rt} + rand.(X_g(t) - X_i(t-1)) \quad (7)$$

$$X_i(t) = X_i(t-1) + V_i(t) \quad (8)$$

where R is the map and compass factor, rand is a random number, and X_g is the current global best position, and which can be obtained by comparing all the positions among all the pigeons.

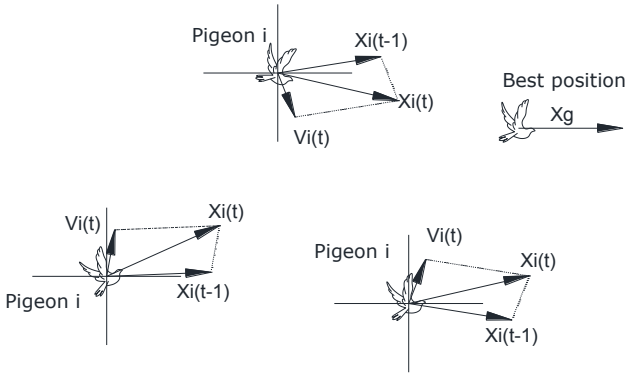

Fig. 4. Map and compass operator model of PIO

As shown in Fig. 4, the best positions of all pigeons are guaranteed by using map and compass. Each pigeon can adjust its flying direction by following the pigeon which has best position according to (7). The former flying direction $X_i(t\text{-}1)$, which has relation to $V_i(t\text{-}1).e^{-Rt}$ in (8). The vector sum of these two arrows is its next flying direction.

landmark operator model is presented based on landmarks. In this operator, the number of pigeons is decreased by half in every generation. The pigeons are still far from the destination, and they are unfamiliar the landmarks. Let $X_c(t)$ be the center of half of pigeon's position at the *t-th* iteration, and suppose every pigeon can fly straight to the destination X_c. The position updating rule of for pigeon i at the *t-th* iteration can be given by:

$$N_{pl}(t) = \frac{N_p(t-1)}{2} \quad (9)$$

$$X_c = \frac{\sum_{N_{pl}} X_i(t).fitness(X_i(t))}{\sum_{N_{pl}} fitness(X_i(t))} \quad (10)$$

$$X_i(t) = X_i(t-1) + rand.(X_c(t) - X_i(t-1)) \quad (11)$$

where $fitness(X_i(t))$ is the quality of the pigeon individual. For the minimum optimization problems, we can choose:

$$fitness(X_i(t)) = \begin{cases} \frac{1}{f_{\min}(X_i(t)) + \varepsilon} & \text{for minimization problems} \\ f_{\max}(X_i(t)) & \text{for maximization problems} \end{cases}$$

For each individual pigeon, the optimal position of the Nc-*th* iteration can be denoted with X_p , and X_p = *min* (X_{i1}, X_{i2} , ..., X_{iNc}).

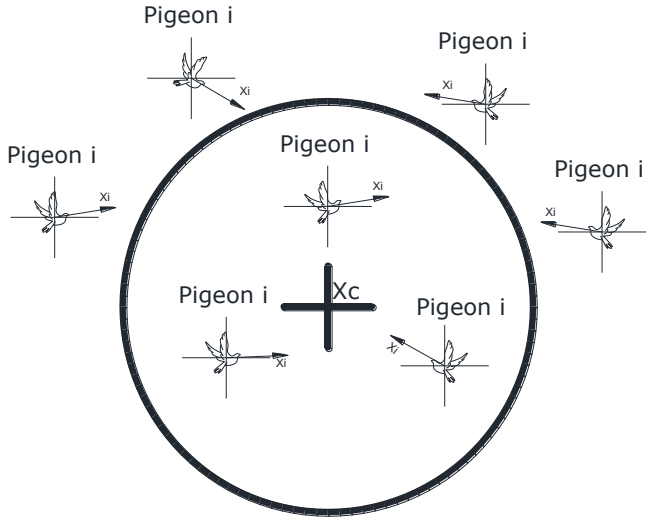

Fig. 5. Landmark operator model

As shown in Fig. 5, the center of all pigeons X_c is their destination in each iteration. Half of all the pigeons (the pigeons out of the circle) that are far from their destination will follow the pigeons that are close to their destination, which also means that two pigeons may be at the same position. The pigeons that are close to their destination (the pigeons in the circle) will fly to their destination very quickly. PIO implementation procedure is as follow:

Step1: Utilizing the model used in (2) and the control structure as in Fig. 1.

Step 2: Initialize parameters of PIO algorithm, with ($D = 6$), the pigeons number ($Np = 60$) , map and compass factor ($R=6$) , ($rand = 0.3$), the number of iteration ($Nc_1 \max$) and ($N_{c2} \max$) for two operators.

Step3: Set each pigeon with a randomized velocity and position. Compare the fitness of each pigeon and find the current best solution Xg which has the minimum fitness.

Step4: Operate map and compass operator. Firstly, the velocity and position of every pigeon is updated by (7) and (8). Then we compare all the pigeon's fitness and find the new best solution Xg.

Step5: If $Nc > {}_{1\max} Nc$, stop the map and compass operator and operate next operator. Otherwise, go to step 4.

Step6: Rank all pigeons according their fitness values. Half of pigeons whose fitness functions are low will follow those pigeons with high fitness according to (9). We then find the center of all pigeons Xc according to (10), and this center is the desirable destination. All pigeons will fly to the destination by adjusting their flying direction according to (11). Next, store the best solution parameters.

Step7: If $Nc > Nc_{2\max}$, stop the landmark operator, and output the results. If not, go to step 6.

B. Prey-Predator Pigeon-Inspired Optimization(PPPIO) for PI Controller

The basic pigeon optimization model can be improved using two aspects (Prey and Predator) in order to improve its feasibility and accuracy in solving the problem of optimizing the value of controller gains. Predatory behavior is one of the most common phenomena in nature, and many optimization algorithms are inspired by the Prey-Predator strategy from ecology [14],[15]. In nature, predators hunt prey to guarantee their own survival, while the preys need to be able to run away from predators. On the other hand, predators help to control the prey population while creating pressure in the prey population [16],[17].

The basic PIO uses two independent iterations, and the two operators act in different loops, respectively. In PPPIO, the work of the two operators is merged into an iterative loop through the navigation tool transition factor tr. The specific update is as follows:

$$N(t) = N(t-1) - N_{dec} \tag{12}$$

$$V_i(t) = V_i(t-1)e^{-Rt} + rand\ tr.(1 - \frac{\ln(t)}{\ln(t_{\max})}).(X_{gbest}(t) - X_i(t-1)) + rand\ tr.(\frac{\ln(t)}{\ln(t_{\max})}).(X_{center}(t) - X_i(t-1)) \tag{13}$$

$$X_i(t) = X_i(t-1) + V_i(t) \tag{14}$$

where N_{dec} is the number of pigeons discarded in each iteration and t_{max} is the maximum number of iterations.

The behavior of predator and prey pigeons will be described as the predator pigeons will chase the center of the preys' swarm; they look like chasing preys as shown in Fig. 6. And preys escape from predators in multidimensional solution space. After taking a trade-off between predation risk and their energy, escaping particles would take different escaping behaviors. This helps the particles avoid the local optimal solutions and find the global optimal solution [9].

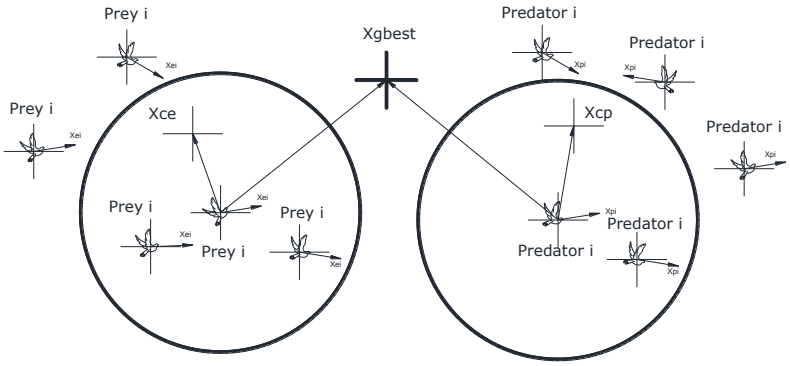

Fig. 6. Schematic diagram of predator and prey

When the distance between the prey pigeons and the PP pigeons is less than the flight initiation distance (FID), its escape speed will depend on the energy state (ie, fitness value). The greater energy, the stronger escape ability. If the distance between the prey and the predator ($distance_j$) is less than the distance of the escape, the prey needs to be mutated with different position after mutation. For each predator i, the speed and position are updated as follows:

$$N_p(t) = N_p(t-1) - N_{p\_dec} \tag{15}$$

$$V_{pi}(t) = V_{pi}(t-1)e^{-Rt} + rand\ tr.(1 - \frac{\ln(t)}{\ln(t_{\max})}).(X_{pgbest}(t) - X_{pi}(t-1)) + rand\ tr.(\frac{\ln(t)}{\ln(t_{\max})}).(X_{p\_center}(t) - X_{pi}(t-1)) \tag{16}$$

$$X_{pi}(t) = X_{pi}(t-1) + V_{pi}(t) \tag{17}$$

for each prey i, if ($distance_j > FID$), the speed and position for prey are updated as follows:

$$N_e(t) = N_e(t-1) - N_{e\_dec} \tag{18}$$

$$V_{ei}(t) = V_{ei}(t-1)e^{-Rt} + rand\ tr.(1 - \frac{\ln(t)}{\ln(t_{\max})}).(X_{egbest}(t) - X_{ei}(t-1)) + rand\ tr.(\frac{\ln(t)}{\ln(t_{\max})}).(X_{e\_center}(t) - X_{ei}(t-1)) + rand.pr\ sign(D_e - dis\tan ce_j).E_{ei}(t-1).X_{\max}.(1 - P_{ei}(t-1)) \tag{19}$$

$$X_{ei}(t) = X_{ei}(t-1) + V_{ei}(t) \tag{20}$$

where ($distance_j$) is the distance between prey and nearest predator, X_{max} is the maximum value for parameters in X_{ei}, pr is the predator impact factor.

$$sign(D_e - distance_j) = 1 : D_e > distance_j$$
$$= 0 : D_e \le distance_j \quad (21)$$

In order to calculate the fourth term in (19), the follows will be defined:

1) Alert distance D_e which reflects the ability of the prey to guard against predator and its value varies with the size of the population and the density of the population.

$$D_e = FID(1 + \frac{N_p}{\rho . N_p}) \quad (22)$$

where $\rho = \frac{N_p + N_e}{X_{span}}$ Indicates the local density of the current population, X_{span} represents the span for parameters in X_{ei}.

2) The energy state E_{ei} refers to the current preying state of the prey, expressed as the ratio of the fitness of the prey to the mean fitness of the prey pigeons

$$E_{ei}(t) = \frac{fitness_{ei}(t)}{fitness_{e\_avg}(t)} \quad (23)$$

3) Predation probability P_{ei} indicates the probability that prey i is being preyed for a certain period of time, ie:

$$P_{ei} = e^{-\alpha_i k (\frac{t + t_{max}}{t_{max}})} \quad (24)$$

where $\alpha_i = e^{\frac{-distance_j \cdot \beta}{N_p}}$ Indicates the probability that the prey i will meet the predator, β indicates the control parameter, k represents the probability that the predator will attack the prey pigeon (fixed at 1). PPPIO implementation procedure is as follow:

Step1: Utilizing the model used in (2) and the control structure as in Fig. 1 but with PI controller in forward path of normal acceleration loop.

Step 2: Initialize parameters of PIO algorithm, with ($D = 3$), predators number ($Np = 30$) , preys number ($Ne = 60$) , map and compass factor ($R=6$) , ($rand = 0.3$), the number of iteration (Ncmax = 50) and (Np_dec , Ne_dec), transition factor (tr =2), probability of attack (k=0.9), impact factor (pr = 1).

Step3: Set each pigeon with a randomized velocity and position. Compare the fitness of each pigeon and find the current best solution X_g and center X_c for both predators and preys.

Step4: Firstly, the velocity and position of every pigeon is updated by using (15) to (20). Then we compare all the pigeon's fitness and find the new best solution X_{gbest} and the center X_c.

Step5: from the minimum fitness function for predators f_{gpbest} and minimum fitness function for preys f_{gebest}, the minimum is selected to be the best fitness f_{gbest} and its solution X_{gbest}.

Step6: If $Nc >$ 1max Nc , stop the iteration and output the results. Otherwise, go to step 4.

C. Fitness function calculation for PIO & PPPIO Algorithm

The control law needed to operate control action is:

$$u = k_q q - (k_p + k_i s)e_{a_n} \quad (25)$$

So, using the time integration of the square of absolute error value as the minimum objective function. In order to prevent the control energy to be too high, add the square of control input to the objective function and square of pitch rate. The optimal parameter selection is as follows:

$$J = \int_0^\infty k_1 q^2 + k_1 u^2 + k_3 e_{a_n}^2 dt \quad (26)$$

where k_1, k_2, k_3 represents the weights. Adding the term concerning rise time to make response faster. In order to avoid overshoot, the punishing function is adopted. Once overshoot appears, it is taken into the optimal index. So the optimal indexes are as follows:

$$J = \int_0^\infty k_1 q^2 + k_1 u^2 + k_3 e_{a_n}^2 dt + k_4 t_r : e(t) < 0 \quad (27)$$
$$= \int_0^\infty k_1 q^2 + k_1 u^2 + k_3 e_{a_n}^2 + k_5 \cdot |e_{a_n}(t)| dt + k_4 t_r : e(t) \ge 0$$

where t_r is the rise time, k_4, k_5 are also weights. Setting the weights to be [k_1=1, k_2=0.1, k_3=4, k_4=100, k_5=4].

V. RESULTS AND ANALYSIS

The optimum values for controller gains (k_q, k_p, k_i) obtained from PSO, PIO, PPPIO algorithms and classical technique with the final value of optimization function J calculated (from 0 to 5 seconds) are shown in TABLE II. .

TABLE II. ACQUIRED VALUES FOR CONTROLLER GAINS

Algorithm	Classical control techniques	PSO	PIO	PPPIO
k_q	11.8	20	20	29.5212
k_p	0	12	12	29.0049
k_i	11.5	22	22	29.5848
J	113.8272	107.5209	107.5209	85.9468

Fig. 7 shows the optimization process of function J. Simulating of the different forms of control gains with the system, the response of normal acceleration is obtained as shown in Fig. 8.

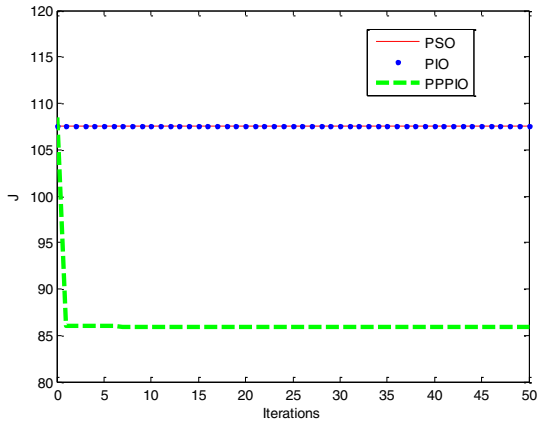

Fig. 7. Optimization process with function J

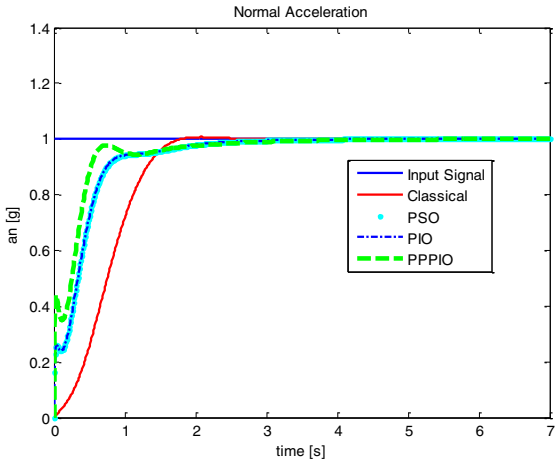

Fig. 8. Step response for system parameters

From the above figures it is obvious that PSO, PIO and PPPIO algorithms have no overshoot and made a faster response rather than classical control even though the later achieved lower settling time than the others. PSO, PIO and PPPIO algorithm optimized the gains so that made no overshoot in normal acceleration response and decreased rise time and improved the stability margins. PPPIO algorithm improved the performance of normal acceleration, whether in performance characteristics (minimum rise time) or stability margins. The performance and stability characteristics for all forms are shown in TABLE III. .

TABLE III. PERFORMANCE AND STABILITY CHARACTERISTICS

Algorithm	**Classical control techniques**	**PSO**	**PIO**	**PPPIO**
rise time[sec]	1.04	0.74	0.74	0.51
settling time [sec]	1.62	2.05	2.05	2.235
Overshoot [%]	1	0	0	0
Gain margin [dB]	17.7	∞	∞	∞
Phase margin [°]	68.2	85.2	85.2	70.9

VI. CONCLUSION

In this paper, PSO, PIO and PPPIO algrithms are used to evaluate the optimum values for longitudinal autopilot gains which revealed that there is other optimal solution rather than designed by classical techniques improves the system characteristics. The solutions obtained from PSO, PIO and PPPIO algorithms made an improvement in system respone except for settling time. Rather than PPPIO made the best improvements in system characteristics (rise time, stability margins).

REFERENCES

[1] John H. Blacklock, “Automatic Control of Aircraft and Missile”, 2nd edition, John Wiley & Sons, Inc., (1991).

[2] P. Garnell, D.J. East, "Guided Weapons and Control Systems", Pergamon Press, 1980.

[3] George M. Siouris. Missile Guidance and Control Systems. Springer, New york, 2003.

[4] Peter H. Zipfel," Modeling and Simulation of Aerospace Vehicle Dynamics", 2nd edition, AIAA, Inc., 2007.

[5] Farhan A. Faruqi , Thanh Lan Vu,” Mathematical Models for a Missile Autopilot Design”, Weapons Systems Division,Systems Sciences Laboratory,DSTO-TN-0449,(2002).

[6] Peter H. Zipfel," Modeling and Simulation of Aerospace Vehicle Dynamics", 2nd edition, AIAA, Inc., 2007.

[7] Benjamin C. Kuo, Farid Golnaraghi, “Automatic Control Systems”, 9th ed., John Wiley & Sons, Inc., (2010).

[8] Brian L. Stevens, Frank L. Lewis, “Aircraft Control and Simulation”, John Wiley & Sons, Inc., 1992.

[9] H. Duan, Pei Li, “Bio-inspired Computation in Unmanned Aerial Vehicles:, Springer-Verlag Berlin Heidelberg 2014.

[10] H. Duan, S. Shao, B. Su, L. Zhang, “New development thoughts on the Bio-inspired intelligence based control for unmanned combat aerial vehicle,” Science China technological sciences, vol. 53, no.8, pp. 2025–2031, 2010.

[11] H. Duan, P. Qiao, “Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning,” International Journal of Intelligent Computing and Cybernetics Vol. 7 No. 1, 2014, pp. 24-37.

[12] C. Mora, C. Davison, J. Wild, M. Walker, “Magentorecrption and its trigeminal medition in the homing pigeon,” Nature, vol. 432, no. 7016, pp. 508-511, 2004.

[13] H. Sun, H. Duan, “PID Controller Design Based on Prey-Predator Pigeon-Inspired Optimization Algorithm,” 2014 IEEE International Conference on Mechatronics and Automation, Tianjin, 2014, pp. 1416-1421.

[14] W. Zhu, H. Duan, “Chaotic predator-prey biogeography-based optimization approach for UCAV path planning,” Aerospace science and technology, pp. 153-161, 2014.

[15] Duan H B, Yu Y X, Zhao Z Y., “Parameters identification of UCAV flight control system based on predator-prey particle swarm optimization”, Sci China Inf Sci, 2013, 56: 012202(12), doi: 10.1007/s11432-012-4754-9

[16] Duan H B, Qiu H X, Fan Y M. “Unmanned aerial vehicle close formation cooperative control based on predatory escaping pigeon-inspired optimization” (in Chinese). Sci Sin Tech, 2015, 45: pp. 559–572, doi: 10.1360/N092015-00125

[17] H. Duan, Pei Li, Yaxiang Yu, “A Predator-prey Particle Swarm Optimization Approach to Multiple UCAV Air Combat Modeled by Dynamic Game Theory”, IEEE/CAA Journal of Automatica Sinca, VOL. 2, NO. 1, 2015, pp. 11-18.