Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080146

Applications of Resource Allocation Network and Optimization Algorithms to Automatic Landing Control

Jih-Gau Juang[1,*], *Cheng-Yen Yu*[1] *and Chih-Min Lin*[2]

[1] Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University, Keelung 202, Taiwan
[2] Department of Electrical Engineering, Yuan Ze University, Chungli 320, Tao-Yuan, Taiwan

Received: 3 Jul. 2013, Revised: 14 Nov. 2013, Accepted: 16 Nov. 2013
Published online: 1 Jan. 2014

Abstract: This paper presents four optimization algorithms: Bacterial Foraging Optimization (BFO), Particle Swarm Optimization (PSO), Chaos Particle Swarm Optimization (CPSO) and Bacterial Swarm Optimization (BSO), and a neural network compensator: Resource Allocation Network (RAN) to aircraft automatic landing control design. When wind disturbance is beyond the originally scheduled flight condition, the aircraft automatic landing system can not be used in such environment during serious wind speed changes. The proposed intelligent control scheme can enhance the PID control performance of the autopilot and guide the aircraft to a safe landing in difficult environment.

Keywords: Resource Allocation Network, PID control, BFO, PSO, CPSO, BSO

1 Introduction

On March 1, 2008, at Hamburg airport, a Lufthansa Airbus A320 tried to land in crosswind condition, which exceeded the limit for the aircraft, and made the left wing touch ground. The pilots then performed a go around and successfully saved the aircraft from crashing. The atmospheric disturbance causes the problem of flight safety and reduces the flying quality. An accident survey [1] of 1,843 aircraft accidents from 1950 through 2009 categorized the causes to be. Weather was a contributing factor, the percentage of weather and weather related to total accidents is 28%. The first automatic landing system (ALS) was developed in England in 1965. Since then, most aircraft have had this system installed. The ALS relies on the instrument landing system (ILS) to guide the aircraft into the proper altitude, position, and approaching angle during the landing phase. According to Federal Aviation Administration (FAA) regulations [2], environmental conditions considered in the determination of dispersion limits are: headwinds up to 25 knots, tailwinds up to 10 knots, crosswinds up to 15 knots, moderate turbulence, and wind shear of 8 knots per 100 feet from 200 feet to touchdown.

Conventional automatic landing systems, which utilize PID controller in control system design, can provide a smooth landing that is essential to the comfort of passengers. However, these systems work only within a specified operational safety envelope. When the conditions, such as turbulence or wind shear, are beyond the envelope, they often cannot be used. Most conventional control laws of the ALS are based on the gain scheduling method [3]. Control parameters are preset for different flight conditions within a specified safety envelope, which is relative defined by FAA regulation. When the conditions, such as turbulence, are beyond the envelope, the ALS is disabled and manual operation is engaged. An inexperience pilot may not be able to guide the aircraft safely. Therefore it is desirable to develop an intelligent ALS that expands the operational envelope to include more safe responses under a wider range of conditions. The goal of this study is to show that the proposed intelligent ALS can relieve human operation and guide the aircraft to a safe landing in wind-disturbance environment. In recent years, intelligent control is more and more popular in control engineering applications. Many intelligent concepts have been applied into various scientific and engineering researches. There are also obvious achievements in flight control domain such as neural networks, fuzzy systems, and evolutionary computation [4]- [9].

[*] Corresponding author e-mail: jgjuang@mail.ntou.edu.tw

PID control has been applied to controller design for decades. It is the most used controller in engineering applications. Control gains of the PID can be tuned by many techniques. In recent years, evolutionary computation is the most used one. This study applies artificial life models to modeling and simulating life-like phenomena for aircraft automatic landing controller design. This paper is mainly based on the following four algorithms to adjust the control parameters of pitch autopilot, they are Bacterial Foraging Optimization (BFO) [10], Particle Swarm Optimization (PSO) [11], Chaos Particle Swarm Optimization (CPSO) [12] and Bacterial Swarm Optimization (BSO) [13]. Control scheme is based on the PID-RAN controller. It uses a traditional PID controller to stabilize the system and train the RAN [14] to provide precise control. The gains of PID controller are adjusted based on the artificial life models.

2 Model Description

At the aircraft landing phase, the pilot descends from the cruise altitude to an altitude of approximately 1200 ft above the ground. The pilot then positions the aircraft so that the aircraft is on a heading towards the runway centerline. When the aircraft approaches the outer airport marker, which is about 4 nautical miles from the runway, the glide path signal is intercepted, as shown in Fig. 1 [15]. As the airplane descends along the glide path, its pitch, attitude, and speed must be controlled. The descent rate is about 10 ft/sec and the pitch angle is between -5 to +5degrees. Finally, as the airplane descends 20 to 70 feet above the ground, the glide path control system is disengaged and a flare maneuver is executed. The vertical descent rate is decreased to 2 ft/sec so that the landing gear may be able to dissipate the energy of the impact at landing. The pitch angle of the airplane is then adjusted, between 0 to 5 degrees for most aircraft, which allows a soft touchdown on the runway surface.

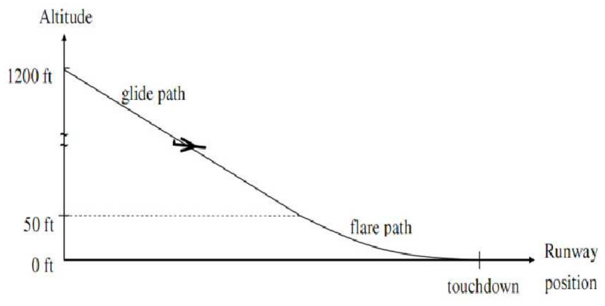


Fig. 1: Glide path and flare path

A simplified model of a commercial aircraft that moves only in the longitudinal and vertical plane is used in the simulations for implementation ease [7]. The motion equations of the aircraft are given as follows:

$$\Delta\dot{u} = X_u(\Delta u - u_g) + X_w(\Delta w - w_g) + X_q\Delta q \\ -g(\frac{\pi}{180})\cos(\gamma_0)\Delta\theta + Z_E\delta_E + Z_T\delta_T \quad (1)$$

$$\Delta\dot{w} = Z_u(\Delta u - u_g) + Z_w(\Delta w - w_g) + (Z_q - \frac{\pi}{180}U_0)\Delta q \\ -g(\frac{\pi}{180})\sin(\gamma_0)\Delta\theta + Z_E\delta_E + Z_T\delta_T \quad (2)$$

$$\Delta\dot{q} = M_u(\Delta u - u_g) + M_w(\Delta w - w_g) + M_q\Delta q \\ +M_E\delta_E + M_T\delta_T \quad (3)$$

$$\Delta\dot{\theta} = \Delta q \quad (4)$$

$$\Delta\dot{h} = -\Delta w + \frac{\pi}{180}U_0\Delta\theta \quad (5)$$

where u is the aircraft longitudinal velocity (ft/sec), w is the aircraft vertical velocity (ft/sec), q is the pitch rate (rate/sec), θ is the pitch angle (deg), h is the aircraft altitude (ft), δ_E is the incremental elevator angle (deg),δ_T is the throttle setting (ft/sec), γ_o is the flight path angle (-3deg),and g is the gravity (32.2 ft/sec^2). The parameters X_i ,Z_i and M_i are the stability and control derivatives.

To make the ALS more intelligent, reliable wind profiles are necessary. Two spectral turbulence forms modeled by von Karman and Dryden are mostly used for aircraft response studies. In this study the Dryden form [7] was used for its demonstration ease. The model is given by:

$$u_g = u_{gc} + N(0,1)\sqrt{\frac{1}{\Delta t}}\left(\frac{\sigma_u\sqrt{2a_u}}{s+a_u}\right) \quad (6)$$

$$w_g = N(0,1)\sqrt{\frac{1}{\Delta t}}\left(\frac{\sigma_w\sqrt{3a_w}(s+b_w)}{(s+a_u)^2}\right) \quad (7)$$

where

$$u_{gc} = -u_{wind510}[1+\frac{\ln(h/510)}{\ln(51)}], a_u = \frac{U_o}{L_u}, L_w = h,$$

$$a_w = \frac{U_o}{L_w}, b_w = \frac{U_o}{L_w\sqrt{3}}, L_u = 100h^{1/3} \text{ for } h > 230,$$

$$L_u = 600 \text{ for } h \le 230, \sigma_u = 0.2\left|u_{gc}\right| \text{ for } h > 500,$$

$$\sigma_w = 0.2\left|u_{gc}\right|(0.5+0.00098h) \text{ for } 0 \le h \le 500.$$

The parameters are: u_g is the horizontal wind velocity (ft/sec), w_g is the vertical wind velocity (ft/sec), u_0 is the nominal aircraft speed (ft/sec), $u_{wind510}$ is the wind speed at 510 ft altitude, L_u and L_w are scale lengths (ft), σ_u and σ_w are RMS values of turbulence velocity (ft/sec), Δt is the simulation time step (sec), $N(0,1)$ is the Gaussian white noise with zero mean and unity standards deviation, u_{gc} is the constant component of u_g, and h is the aircraft altitude (ft). Fig. 2 shows a turbulence profile with a wind speed of 30 ft/sec at 510 ft altitude.

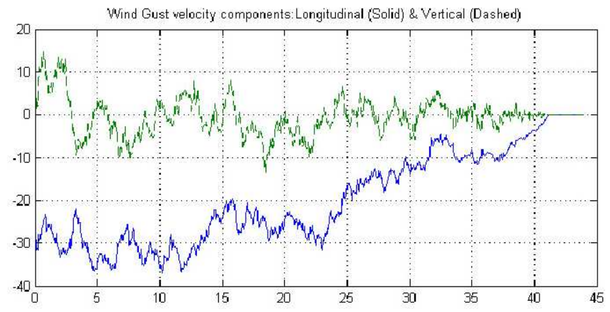


Fig. 2: Turbulence profile

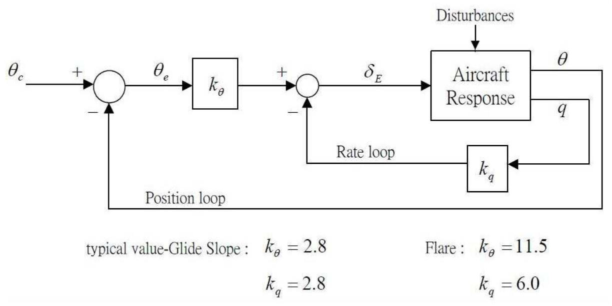


Fig. 4: Pitch autopilot

3 Control System

PID controller is a simplified structure of an aircraft landing controller as shown in Fig. 3. Its inputs consist of altitude and altitude rate commands along with aircraft altitude and altitude rate. Via aircraft landing controller we can obtain the pitch command θ_c. Then, the pitch autopilot is controlled by pitch command. The pitch autopilot is shown in Fig. 4. Detail descriptions can be found in [7]. In order to enable aircraft to land more steady when an aircraft arrives to the flare path, a constant pitch angle will be added to the controller. In general, the PID controller is simple and effective but there are some drawbacks such as apparent overshoot and sensitive to external noise and disturbance. When severe turbulence is encountered the PID controller may not be able to guide the aircraft to land safely. With RAN compensator the proposed controller can overcome these disadvantages. It uses a traditional PID controller to stabilize the system and train the RAN to provide precise control. The gains of PID controller are adjusted based on experiences, what it provides are tolerable solutions, not desired solutions. The RAN can effectively meliorate these conditions.

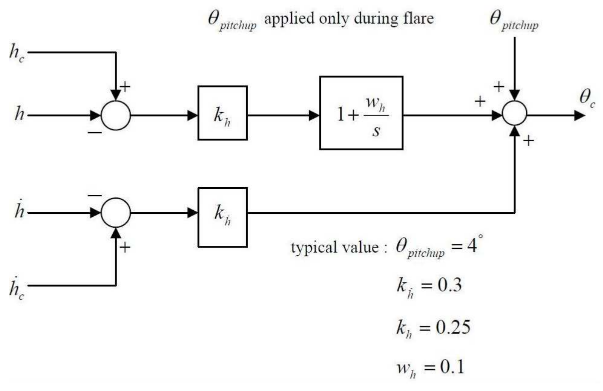

$\theta_{pitchup}$ applied only during flare
typical value : $\theta_{pitchup} = 4^\circ$
$k_h = 0.3$
$k_{\dot h} = 0.25$
$w_h = 0.1$

Fig. 3: PID-controller

Fig. 5 shows the application of the intelligent system to the ALS. The four inputs of the aircraft are altitude, altitude rate, altitude command, and altitude rate command and also are the inputs for the PID controller and the intelligent system. The input of pitch autopilot U, which is the control signal of aircraft model, is the summation of the PID controller output U_{PID} and the intelligent system output U_{IS}. The conventional PID controller is used to stabilize the aircraft and to help intelligent system in learning process, then the intelligent system improves the performance of the intelligent controller in severe wind disturbance condition. The adjust-gains mean using the optimization algorithm to replace the original PID value after repeated training.

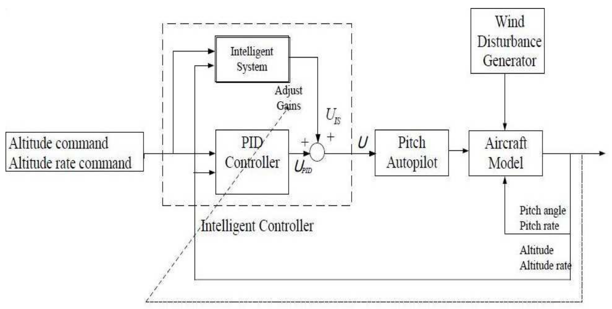


Fig. 5: The intelligent control scheme

At each sampling instant k, the function of the intelligent system includes two phases, which are recall process and learning process. First, the intelligent system will utilize $Y_d(k+1)$ and $Y(k)$ to address the corresponding weights in order to generate an output U_{IS} in the recall process, where the $Y(k)$ is the output of the dynamic aircraft model at sampling instant k and the $Y_d(k+1)$ represents the desired dynamic aircraft model output at the next time step, as shown in Fig. 6. U_{IS} in the recall process is taken to be an calculation of the demanded control signal U. And then it is integrated with the output of PID controller U_{PID} to form the demanded control signal U. In the learning process, as shown in Fig. 7, U is obtained in the recall process and regarded as the desired output. The error obtained from $U - U_{IS}$ is used to update the corresponding weights that are stored at location $Y(k)$ and $Y(k+1)$. The error will converge after several iterations, then the intelligent system can compensate for the PID controller.

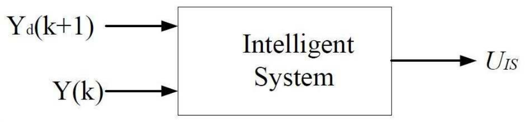


Fig. 6: The control process of intelligent system

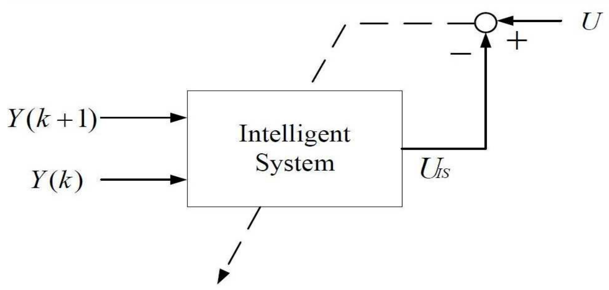


Fig. 7: The learning process of intelligent system

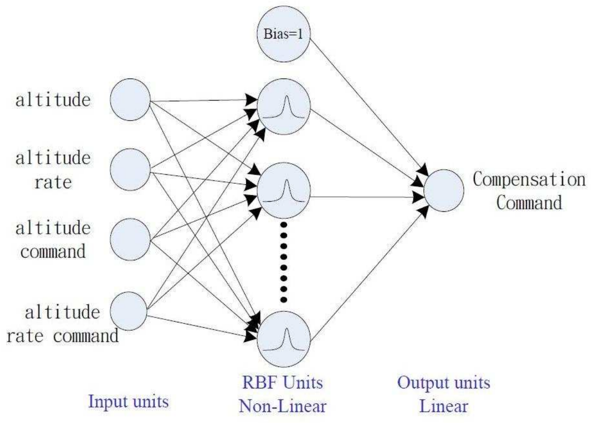


Fig. 8: Structure of RAN [16]

The RAN, as shown in Fig. 8, uses a sequential learning algorithm. The output of the RAN algorithm has the following form:

$$\begin{aligned} y(n) &= \sum_{j=1}^{HN} \varphi_j\left(\underline{x}(n)\right) w_j(n) + \theta(n) = \sum_{j=0}^{HN} w_j(n)\varphi_j\left(\underline{x}(n)\right) \\ &= W_H^T(n)\varphi_H\left(\underline{x}(n)\right) \end{aligned} \tag{8}$$

where HN is the number of hidden units, $\varphi_j\left(\underline{x}(n)\right)$is the response of the jth hidden neuron to the input vector $\underline{x}$,$\varphi_j(x_0) = 1$, and $\theta = w_0\varphi_0$ is the bias term. The activation function of the neuron is a Gaussian function.

$$\varphi_j(\underline{x}) = \exp\left(-\frac{\left\|\underline{x} - \underline{m}_j\right\|^2}{2\sigma_j^2}\right) \qquad \varphi(\cdot) \in [0 \sim 1] \tag{9}$$

where $\underline{m}_j = (m_1^j, m_2^j, \ldots, m_p^j)$ is the center of the jth activation function, and σ_j is the width of the jth Gaussian function. $W_H \in R^{(HN+1)\times 1}$is the weight vector from the hidden layer to the output layer.

The learning process of RAN involves allocation of new hidden units as well as adjustment of network parameters. The network begins with no hidden units. As observations are received, the network grows, using them as new hidden units. The following two criteria must be met for an observation (x_0, y_0) to be added as a new hidden unit to the network:

$$\left\|\underline{x} - \underline{m}_j\right\| > \varepsilon(n) \tag{10}$$

$$e(n) = d(n) - y(n) > e_{min} \tag{11}$$

where $\underline{m}_j$ is the center of the hidden unit which is the closest to $\underline{x}(n)$, $\varepsilon(n)$ and e_{min} are thresholds to be selected appropriately. When a new hidden unit is added to the network, the parameters associated with the unit are

$$W_{j+1} = e(n) \tag{12}$$

$$\underline{m}_{j+1} = \underline{x}(n) \tag{13}$$

$$\sigma_{j+1} = K\left\|\underline{x}(n) - \underline{m}_j\right\| \tag{14}$$

K is an overlap factor, determining the overlap of the responses of the hidden units in the input space. When the observation (x_0, y_0) does not meet the criteria for adding a new hidden unit, the network parameters W_H use the back-propagation algorithm to the updating law as follows:

$$\begin{aligned} w_j(n+1) &= w_j(n) - \eta_w \frac{\partial E(n)}{\partial w_j(n)} \\ &= w_j(n) + \eta_w (d(n) - y(n))\varphi_j(\underline{x}(n)) \end{aligned} \tag{15}$$

$$
\begin{aligned}
\underline{m}_j(n+1) &= \underline{m}_j(n) - \eta_m \frac{\partial E(n)}{\partial \underline{m}_j(n)} \\
&= \underline{m}_j(n) + \eta_m (d(n) - y(n)) w_j(n) \varphi_j(\underline{x}(n)) \\
&\quad \frac{1}{\sigma_j^2} (\underline{x}(n) - \underline{m}_j(n)) \qquad (16)
\end{aligned}
$$

$$
\begin{aligned}
\sigma_j(n+1) &= \sigma_j(n) - \eta_\sigma \frac{\partial E(n)}{\partial \sigma_j(n)} \\
&= \sigma_j(n) + \eta_\sigma (d(n) - y(n)) w_j(n) \varphi_j(\underline{x}(n)) \\
&\quad \frac{1}{\sigma_j^3} \left\| \underline{x}(n) - \underline{m}_j(n) \right\|^2 \qquad (17)
\end{aligned}
$$

where η is the learning rate.

4 Optimization Algorithms

The proposed control scheme still needs a PID controller to provide primary control signals to the pitch autopilot. Instead of trial-and-error, PID control gains are tuned by different artificial life models as follows.

A. BFO

The bacterial foraging system consists of four principal mechanisms, namely chemotaxis, swarming, reproduction, and elimination dispersal [10].
(i) Chemotaxis
This process simulates the movement of an *Escherichia coli* (*E.coli*) cell through swimming and tumbling via flagella. Biologically, an *E.coli* bacterium can move in two different ways. It can swim for a period of time in the same direction, or it may tumble, and alternate between these two modes of operation for the entire lifetime. Suppose $\theta i(j,k,l)$ represents ith bacterium at jth chemotactic, kth reproductive and lth elimination dispersal step. $C(i)$ is the size of the step taken in the random direction specified by the tumble (run length unit). Then in computational chemotaxis the movement of the bacterium may be represented by

$$
\theta^i(j+1,k,l) = \theta^j(j,k,l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad (18)
$$

where Δ indicates a vector in the random direction whose elements lie in [-1, 1].
(ii) Swarming
An interesting group behavior has been observed for several motile species of bacteria including *E.coli* and *Salmonella Typhimurium* (*S. typhimurium)*, where intricate and stable spatiotemporal patterns (swarms) are formed in a semisolid nutrient medium. A group of *E.coli* cells arrange themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemoeffecter. The cells, when stimulated by a high level of *succinate*, release an attractant *aspertate*, which helps them to aggregate into groups and thus move as concentric patterns of swarms with high bacterial density. The cell-to-cell signaling in *E. coli* swarm may be represented by the following equation:

$$
\begin{aligned}
J_{cc}(\theta, P(j,k,l)) &= \sum_{i=1}^{s} J_{cc}(\theta, \theta^i(j,k,l)) \\
&= \sum_{i=1}^{s} [-d_{attractant} \exp(-\omega_{attractant} \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2)] \\
&+ \sum_{i=1}^{s} [-h_{repellant} \exp(-\omega_{repellant} \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2)] \qquad (19)
\end{aligned}
$$

where $J_{cc}(\theta, P(j,k,l))$ is the objective function value to be added to the actual objective function (to be minimized) to present a time-varying objective function, S is the total number of bacteria, p is the number of variables to be optimized that are present in each bacterium, and $\theta = [\theta 1, \theta 2, ..., \theta p]^T$ is a point in the p-dimensional search domain. $d_{attractant}$, $\omega_{attractant}$, $h_{repellant}$, $\omega_{repellant}$ are different coefficients that should be chosen properly.
(iii) Reproduction
The least healthy bacteria eventually die while each of the healthier bacteria (those yielding lower value of the objective function) asexually split into two bacteria, which are then placed in the same location. This keeps the swarm size constant.
(iv) Elimination and Dispersal
Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons:e.g., a significant local rise of temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. Events can take place in such a fashion that all the bacteria in a region are killed or a group is dispersed into a new location. To simulate this phenomenon in BFO, some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

B. PSO

The main steps in the particle swarm optimization process are described as follows [11]:
(i) Initialize a population (array) of particles with random positions and velocities in the problem space.
(ii) Calculate the fitness function and set the values to the *pbest* for each particle, and set the best value of all the particles to *gbest*.
(iii) Change the velocity and position of the particle according to equations (20) and (21), respectively:

$$
\begin{aligned}
v_{id}^{(k+1)} &= w \cdot v_{id}^{(k)} + c_1 \cdot rand1 \cdot (pbest_{id} - x_{id}^{(k)}) + \\
&\quad c_2 \cdot rand2 \cdot (gbest_d - x_{id}^{(k)}) \qquad (20)
\end{aligned}
$$

$$
x_{id}^{(k+1)} = x_{id}^{(k)} + v_{id}^{(k+1)} \qquad (21)
$$

(iv) Calculate the fitness again and compare the particle's fitness evaluation with the particle's *pbest*. If the current value is better than *pbest*, then set the *pbest* value equal to the current value, and the *pbest* location equal to the current location in d-dimensional space.
(v) Compare the fitness evaluation with the population's overall previous best. If the current value is better than *gbest*, then reset *gbest* to the current particle's array index and value.
(vi) Loop to step (iii) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).
The definition of the parameters are
$v_{id}^{(k)}$: velocity of individual *i*at iteration *k*, where
$v_d^{min} \le v_{id}^{(k)} \le v_d^{max}$
w: inertia weight factor,
c_1, c_2: acceleration constant,
$rand1, rand2$: uniform random number between 0 and 1,
$x_{id}^{(k)}$: current position of individual *i* at iteration *k* ,
$pbest_i$: *pbest* of individual *i*,
gbest: *gbest* of the group.

C. CPSO

The random movement which is obtained by the definite motion equation is called chaotic motion. The variable that presents chaotic state is called chaotic variable that is very sensitive to initial value. The chaotic variable has the characteristic of traversing all states according to the determination formula from any point (except for fixed-point) starting. The Logistic equation is a typical chaotic map system, its expression is [12]:

$$X_{n+1} = \mu X_n(1-X_n), n = 1,2,3,...,\\ 0 \le \mu \le 4, X_n \in [0,1] \tag{22}$$

where, μ is a control parameter. According to chaotic variables with ergodicity and randomness having the enhancement population's search ability, based on chaotic ideas, some domestic and foreign scholars have proposed many kinds of improvement chaos particle swarm optimization (CPSO) algorithms. These improvement algorithm's fundamental mode is:
(i) using chaotic variables to initialize particle's position and velocity in order to increase population's diversity and ergodicity;
(ii) using early-maturing judgment mechanisms to monitor population's evolution situation, and randomly generating an initial value to replace original swarm in order to reinitialize the particle's velocity.
Since these chaotic variables in CPSO algorithm has not involved in the parameters such as inertia weight w and random numbers c_1 and c_2 of PSO algorithm, so these parameter choices will have a very tremendous influence to optimize performance of CPSO algorithm. Especially, when the w value is larger, which will help PSO to jump out the local optimum, and when the w value is smaller, which will is advantageous to CPSO algorithm convergence.

D. BSO

In what follows we briefly outline the new BSO algorithm step by step [13].
(i) Initialize parameters *n, N, NC, NS, Nre, Ned, Ped, C(i)(i =1,2,. . . ,N)*, ϕ^i.
Where,
n: Dimension of the search space,
N: The number of bacteria in the population,
NC: No. of Chemo-tactic steps,
Nre : The number of reproduction steps,
Ned : The number of elimination-dispersal events,
Ped : Elimination-dispersal with probability,
$C(i)$:The size of the step taken in the random direction specified by the tumble.
ω : The inertia weight.
C1: Swarm Confidence.
$\underline{\theta}(i,j,k)$: Position vector of the *i*th bacterium, in *j*th chemotactic step, and *k*th reproduction.
$\underline{V}_i$: Velocity vector of the *i*th bacterium.
(ii) Update the following:
J(i,j,k) : Cost or fitness value of the *i*th bacterium in the *j*th chemo-taxis, and *k*th reproduction loop.
$\underline{\theta}_{g\_best}$: Position vector of the best position found by all bacteria.
Jbest (i,j,k) : Fitness of the best position found so far.
(iii) Reproduction loop: $k = k+1$
(iv) Chemotaxis loop: $j = j+1$
[substep_a] For i =1,2,. . . ,*N*, take a chemotactic step for bacterium *i*as follows.
[substep_b] Compute fitness function,J (i,j,k).
[substep_c] Let *Jlast*=J (i,j,k) to save this value since we may find a better cost via a run.
[substep_d]Tumble: generate a random vector $\Delta(i) \in R^n$ with each element $\Delta_m(i)$, m =1,2,..., *p*, a random number on [-1, 1].
[substep_e] Move :Let

$$\theta(i,j+1,k) = \theta(i,j,k) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{23}$$

[substep_f] Compute $J(i,\ j+1,\ k)$.
[substep_g] Swim: we consider only the *i*th bacterium is swimming while the others are not moving then.
i) Let m =0 (counter for swim length).
ii) While m< *s N*(if have not climbed down too long).
•Let $m = m+1$.
•If $J(i,\ j+1,\ k)$ <*Jlast* (if doing better), let $Jlast = J(i,j+1,k)$ and let $\theta(i,j+1,k)$ equal to eq.(23) to compute the new $J(i,j+1,k)$ as we did in [substep_f]
•Else, let $m = N_s$. This is the end of the while statement.
(v) Mutation with PSO operator
For $i = 1,2,\ldots,S$

Update the $\underline{\theta}_{g\_best}$ and $Jbest(i,j,k)$
Update position and velocity of the dth coordinate of the ith bacterium according to the following rule:

$$V_{id}^{new} = \omega \cdot V_{id}^{new} + C_1 \cdot \varphi_1 \cdot (\underline{\theta}_{g\_best_d} - \theta_d^{old}(i,j+1,k))$$

$$\theta_d^{new}(i,j+1,k) = \theta_d^{old}(i,j+1,k) + V_{id}^{new} \qquad (24)$$

(vi) Let S_r=S/2.
The S_r bacteria with highest cost function (J) values die and the other half of bacteria population with the best values split (and the copies that are made are placed at the same location as their parent).
(vii) If $k<Nre$, go to step (i). We have not reached the specified number of reproduction steps. So we start the next generation in the chemo-taxis loop.

5 Simulations

Different combinational control schemes are tested. Fig. 9 to Fig. 12 show the results of using BSO algorithm with PID-RAN controller in the wind turbulence speed at 121 ft/sec.

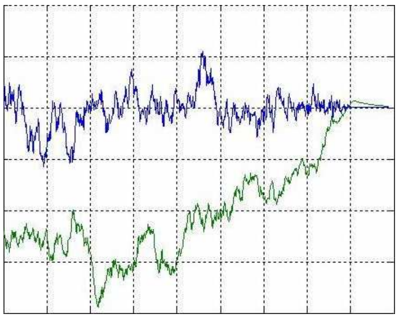


Fig. 9: Turbulence profile (121 ft/sec)

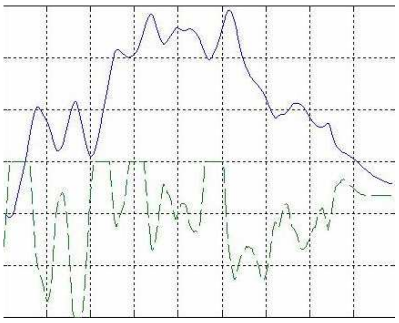


Fig. 10: Aircraft pitch and pitch command

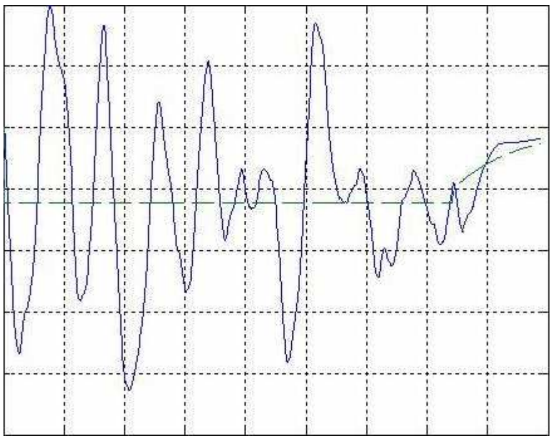


Fig. 11: Aircraft vertical velocity and command

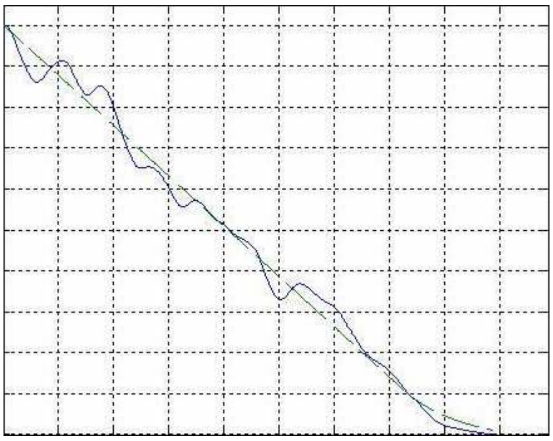


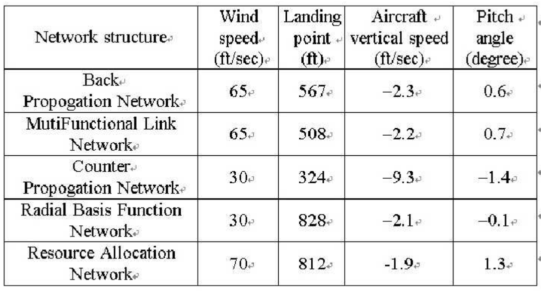
Fig. 12: Aircraft altitude and command

Table I shows the maximum wind speed that the proposed automatic landing controller can overcome by using different optimization algorithms with the PID-RAN controller in turbulence condition. Compared to previous works [16], as shown in Table II, the proposed control scheme has better performance than the one use only neural network controller without compensation and fine tune to the control gains.

Table I. The results from using pid-ran controller with different optimization algorithms

Optimization Algorithms	Wind speed (ft/sec)	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)	Number of hidden units
BFO	81	632	-1.6	2.5	139
PSO	234	620	-1.0	3.5	146
CPSO	102	433	-1.9	2.2	158
BSO	121	879	-1.7	2.7	168

Table II. Results from using different neural network controllers [16]



Network structure	Wind speed (ft/sec)	Landing point (ft)	Aircraft vertical speed (ft/sec)	Pitch angle (degree)
Back Propogation Network	65	567	−2.3	0.6
MutiFunctional Link Network	65	508	−2.2	0.7
Counter Propogation Network	30	324	−9.3	−1.4
Radial Basis Function Network	30	828	−2.1	−0.1
Resource Allocation Network	70	812	-1.9	1.3

6 Conclusion

The purpose of this paper is to investigate the use of optimization algorithms with PID-RAN in aircraft automatic landing system and to make the automatic landing system more intelligent. Current flight control law is adopted in the intelligent controller design. Tracking performance and adaptive capability are demonstrated through software simulations. For the safe landing of an aircraft using a conventional PID controller or a RAN controller the wind speed of turbulence limits are 30ft/sec and 70 ft/sec, respectively [16]. In this study, a well-trained BSO with PID-RAN control scheme can reach 148 ft/sec. The proposed controllers have better performance than previous works, and it can act as an experienced pilot and guide the aircraft to a safe landing in severe wind turbulence environment.

References

[1] Aircraft accident statistic http://www.planecrashinfo.com/cause.htm

[2] Federal Aviation Administration, Automatic Landing Systems, **AC**, 20-57A (1971).

[3] H. Buschek and A. J. Calise, Uncertainty Modeling and Fixed-Order Controller Design for Hypersonic Vehicle Model, Journal of Guidance, Control, and Dynamics, **20**, 42-48 (1997).

[4] M.G. Cooper, Genetic Design of Rule-Based Fuzzy Controllers, Ph.D. dissertation, University of California, Los Angeles, (1995).

[5] Y. Iiguni, H. Akiyoshi, N. Adachi, An Intelligent Landing System Based on Human Skill Model, IEEE Transactions on Aerospace and Electronic Systems, **34**, 877-882 (1998).

[6] H. Izadi, M. Pakmehr, N. Sadati, Optimal Neural-Controller in Longitudinal Autolanding of a Commercial Jet Transport, Proc. IEEE International Conference on Control Applications, CD-000202, Istanbul, Turkey, 1-6 (2003).

[7] C. C. Jorgensen, and C. Schley, A Neural Network Baseline Problem for Control of Aircraft Flare and Touchdown, Neural Networks for Control, 403-425 (1991).

[8] J. G. Juang, and J. Z. Chio, Fuzzy Modeling Control for Aircraft Automatic Landing System, International Journal of Systems Science, **36**, 77-87 (2005).

[9] S. M. B. Malaek, N. Sadati, H. Izadi, M. Pakmehr, Intelligent Autolanding Controller Design Using Neural Networks and Fuzzy Logic, Proc. IEEE 5th Control Conference, **1**, 365-373 (2004).

[10] K. M. Passino, Biomimicry of Bacterial Foraging for Distributed Optimization and Control, IEEE Control Systems Magazine, 52-67 (2002).

[11] J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, **4**, 1942-1948 (1995).

[12] H. J. Meng, P. Zheng, R. Y. Wu, X. J. Hao, Z. Xie, A Hybrid Particle Swarm Algorithm with Embedded Chaotic Search, Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems, Singapore, 3-3 December, (2004).

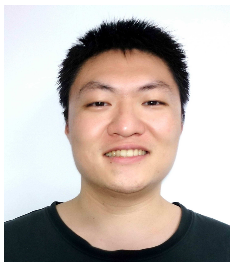
[13] A. Biswas, S. Dasgupta1, S. Das, A. Abraham, Synergy of PSO and Bacterial Foraging OptimizationA Comparative Study on Numerical Benchmarks, Innovations in Hybrid Intelligent Systems, ASC, **44**, 255-263 (2007).

[14] J. Platt, A Resource Allocating Network for Function Interpolation, Neural Computing, **3**, 213-225 (1991).

[15] Flight Safety Foundation-Taiwan, http://www.flightsafety.org.tw/news1.php?Code=1&main_id=3&pages=5

[16] J. G. Juang, L. H. Chien, F. Lin, Automatic Landing Control System Design Using Adaptive Neural Network and Its Hardware Realization, IEEE Systems Journal, **5**, 266-277 (2011).

Jih-Gau Juang received the B.S. degree in control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1981, and the M.S. and Ph.D. degrees in electrical engineering from University of Missouri, Columbia, USA, in 1989 and 1998, respectively. Since 1999, he has been with the National Taiwan Ocean University, Keelung, Taiwan, where he is currently a Professor of the Department of Communications, Navigation and Control Engineering and General-Secretary of the university. His research interests include intelligent control, adaptive control, robotics, and artificial intelligence.



Cheng-Yen Yu received the B.S. degree in electrical engineering from National Taiwan Ocean University, Keelung, Taiwan, in 2009, and the M.S. degree in communications, navigation and control engineering from National Taiwan Ocean University, Keelung, Taiwan, in 2012. Since 2012, he has been with the Pegatron Corporation, New Taipei City, Taiwan, where he is currently a senior engineer.

Chih-Min Lin received the B.S. and M.S. degrees in control engineering and the Ph.D. degree in electronics engineering from National Chiao Tung University, Taiwan, in 1981, 1983, and 1986, respectively. He joined the faculty of the Department of Electrical Engineering, Yuan Ze University in 1993, and is currently the Dean and a Chair-Professor of Electrical and Communication College. He also serves as an Associate Editor of IEEE Trans. Systems, Man, and Cybernetics, Part B; Asian Journal of Control; and International Journal of Fuzzy Systems. He is now a Board of Governor member of IEEE Systems, Man, and Cybernetics Society. His research interests include fuzzy neural network, cerebellar model articulation controller, and intelligent control systems. He is an IEEE Fellow and IET Fellow and has published 112 journal papers and 150 conference papers.