International Journal of Computational Intelligence in Control
Vol. 12 No. 2 (July-December, 2020)

EVOLUTIONARY COMPUTATION AND DSP BASED INTELLIGENT AIRCRAFT LANDING CONTROL

Jih-Gau Juang*, Hou-Kai Chiou and Chia-Ling Lee

Department of Communications, Navigation and Control Engineering
National Taiwan Ocean University, Keelung 20224, Taiwan, R. O. C.

ABSTRACT: This paper presents several digital signal processor (DSP) based intelligent controllers to aircraft automatic landing system. PID control law is adopted in the intelligent controller design. Fuzzy cerebellar model articulation controller (FCMAC) is utilized to compensate for the PID control signal. Control gains are selected by evolutionary computation. Tracking performance of preset landing path and adaptive capability to different disturbances are demonstrated through hardware simulations. Different evolution methods, Adewuya crossover, arithmetical crossover, average crossover, convex crossover, and blend crossover, are utilized to analyze the performance on optimal parameter search. Hardware implementation of this intelligent controller is performed by a DSP board with VisSim platform. This study also compares different CMACs to improve the performance of conventional ALS. The atmospheric disturbances affect not only flying qualities of an aircraft but also flight safety. The proposed intelligent controllers can successfully expand the controllable conditions in severe wind disturbances.

Keywords: PID Control, Evolutionary Computation, DSP, Automatic Landing Control, Fuzzy CMAC, Turbulence.

1. INTRODUCTION

Evolutionary computation (EC) is the general term for several computational techniques on the evolution of biological life in the natural world. In computer science evolutionary computation is a subfield of artificial intelligence involving combinatorial optimization problems. Evolutionary computation includes genetic algorithm (GA), evolutionary programming (EP), evolution strategies (ES), genetic programming (GP), and classifier systems (CS). Despite being developed independently over several decades in these subfields of evolutionary computation, most of these techniques are similar in spirit, but differ in the details of their implementation and the nature of the particular problem to which they have been applied. The GA is search procedures based on the mechanics of natural genetics and natural selection. Compared to GA, the EP typically only uses the mutation process without using the binary-coded and reproduction (or crossover) processes. The efficiency of ES also depends on the size of mutation strength. The GP is that one kind by operators according to the basic function of the definition of the working purpose, on the basis of limiting conditions of questions, through evolutionary computation to generate optimal procedure modeling automatically. The difference between GA and GP is that the type of GA is by way of coding and the type of GP is by way of procedure. The CS is a kind of adaptive rule-based system, which can study complicated rule from the change of external environment and adjust progressively to strengthen own inherent knowledge. The most widely used form of evolutionary computation is the generic algorithm. The GA was proposed by John Holland in 1962 [1], which is an optimization and search technique based on the principles of genetics and natural selection. In 1975, Holland mentioned the most basic principle of GA in "Adaptation in Natural and Artificial System" [2]. In the same year, De Jong showed the usefulness of the GA for function optimization and made the first concerted effort to find optimized GA parameters.

Corresponding author: E-mail: jgjuang@mail.ntou.edu.tw

Whereas the GA generally only involves techniques of implementing mechanisms, such as reproduction, crossover, mutation, fittest function, etc. Via reproduction, crossover, and mutation steps, the GA can generate next generation to reach the purpose of evolution. According to level of fitness value, the GA retains the fine one and eliminates the inferior populations. Therefore, it is widely used to solve optimal problems recently. It can search many points at the same time and is not apt to fall into local optimal solution. The GA has been used for a wide range of applications, as well as for specific applications focused on a specific requirement. So far many new improving methods focus on making GA more efficient and increasing parameter searching ability widely. Here we put focus on crossover principle of different ECs. We utilize five crossover principles [3-5] with intelligent controllers under wind disturbances to search optimal control gains, and compare the differences between these principles.

According to Boeing's report [6], 67% of the accidents by primary cause are due to human factors and 5% are attributed to weather factors. By phase of flight, 47% accidents are during final approach or landing. It is therefore desirable to develop an intelligent automatic landing system (ALS) that expands the operational envelope to include safer responses under a wider range of conditions. In this study, robustness of the proposed controller is obtained by choosing optimal control gains that allow a wide range of disturbances to the controller. The goal of this paper is to show that the proposed intelligent ALS can relieve human operators and guide the aircraft to a safe landing in severe turbulence environments. This study first uses conventional aircraft automatic landing control system that uses PID controller with EC as the adjustment mechanism to improve the performance of conventional ALS and guide the aircraft to a safe landing. Wind disturbances are also implemented into the flight simulations. Many researchers have applied intelligent concepts to the problem of intelligent landing control [7-8], but these intelligent concepts are not adaptive to various wind disturbance conditions. In past decades, most of the improvements in the ALS system have been on the guidance instruments, such as GNSS Integrity Beacons, Global Positioning System, Microwave Landing System, and Automatic Land Position Sensor [9-12]. By using improvement calculation methods and high accuracy instruments, these systems provide more accurate flight data to the ALS to make the landing smoother. However, these research do not include weather factors such as wind turbulences. Recently, intelligent concepts such as neural networks, fuzzy system, genetic algorithm, and hybrid systems have applied to flight control to increase the flight controller's adaptive capability to different environments [13-17]. This paper uses type-2 FCMAC [18-20] to improve the performance of conventional ALS. Comparisons of conventional CMAC [21] and conventional (type-1) FCMAC [22] are also given. The performance of the intelligent ALS under severe environment can be improved by the advantages of the CMAC which include local generalization and rapid learning process. Meanwhile, this study also utilizes the VisSim software and TI C2000 Rapid Prototyper to develop an embedded control system that uses a DSP controller. Thus, hardware-in-the-loop control can be achieved.

2. LANDING SYSTEM

At the aircraft landing phase, the pilot descends from the cruise altitude to an altitude of approximately 1200 feet 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 Figure 1. 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 +5 degrees. 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 2ft/sec so that the landing gear may be able to dissipate the energy of the impact at landing. The

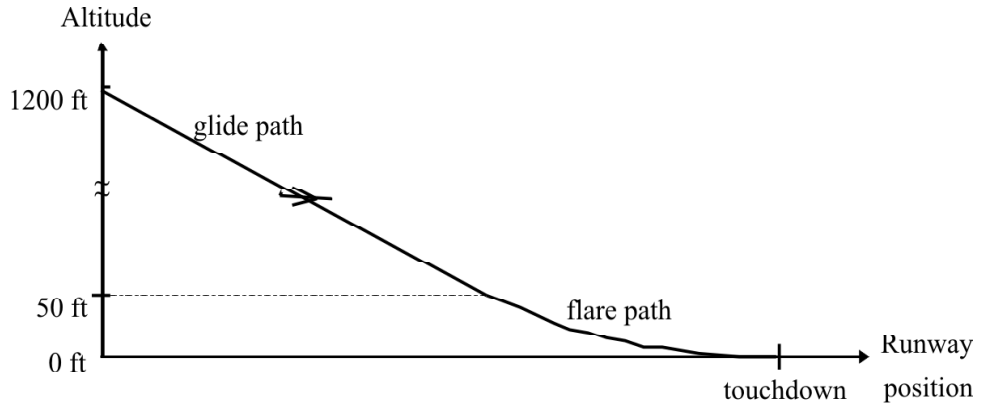

Figure 1: Glide Path and Flare Path

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.

A simplified model of a commercial aircraft that moves only in the longitudinal and vertical plane is used in the simulations for implementation ease [13]. The motion equations of the aircraft are given as follows:

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

$$\dot{w} = Z_u(u-u_g) + Z_w(w-w_g) + \left(Z_q - \frac{\pi}{180}U_0\right)\Delta q - g\left(\frac{\pi}{180}\right)\sin(\gamma_0)\theta_T \tag{2}$$

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

$$\dot{\theta} = q \tag{4}$$

$$\dot{h} = -w + \frac{\pi}{180}U_0\theta \tag{5}$$

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 models by von Karman and Dryden are mostly used for aircraft response studies. In this study the Dryden form [13] was used for its demonstration ease. The model is given by :

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

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

where $u_{gc} = -u_{wind510}\left[1+\frac{\ln(h/510)}{\ln(51)}\right]$, $a_u = \frac{U_o}{L_u}$, $L_w = h$, $a_u = \frac{U_o}{L_u}$, $a_w = \frac{U_o}{L_w}$, $b_w = \frac{U_o}{L_w\sqrt{3}}$, $L_u = 100\text{h}^{1/3}$ for $h > 320$, $L_u = 600$ for $h \le 230$,

$\sigma_w = 0.2|u_{gc}|(0.5 + 0.00098 \times h)$ for $0 \le h \le 500$, $\sigma_w = 0.2|u_{gc}|$ for $h > 500$.

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). Figure 2 shows a turbulence profile with a wind speed of 30 ft/sec at 510 ft altitude.

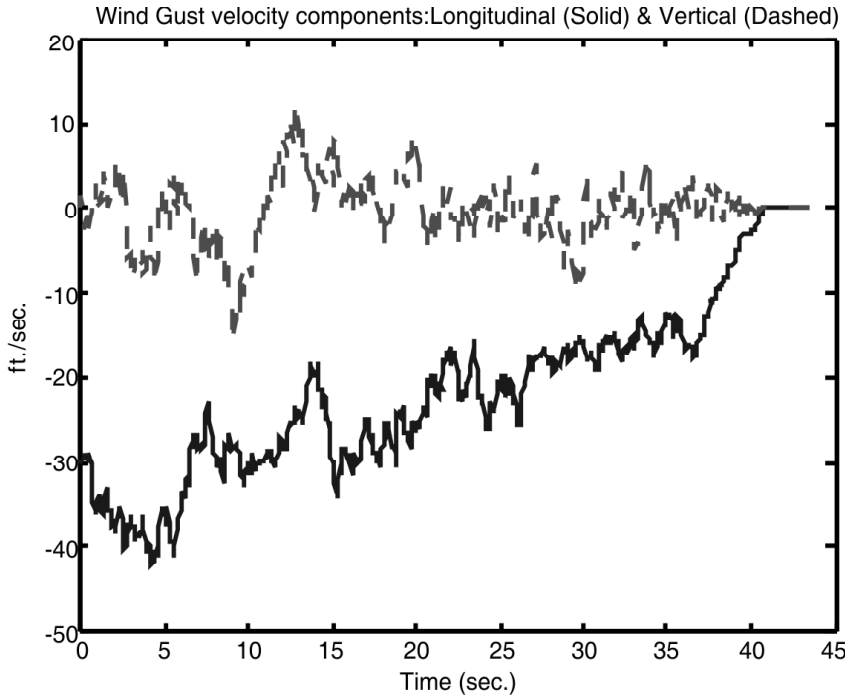

Figure 2: Turbulence Profile

3. CONTROL SCHEME

Conventional aircraft landing system uses PID-type control, as shown in Figure 3. Controller inputs consist of altitude and altitude rate commands along with aircraft altitude and altitude rate. The pitch command θ_c is obtained from the PID controller. Then, the pitch autopilot is controlled by pitch command. The pitch autopilot is shown in Figure 4. 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 the CMAC compensator, the proposed controller can overcome these disadvantages. It uses a traditional PID controller to stabilize the system and train the CMAC to provide precise control. The original gains of PID controller are adjusted based on experiences, what it provides are tolerable solutions, not desired solutions. The CMAC can effectively meliorate these conditions.

The overall control scheme is described in Figure 5, in which the control signal U is the sum of the PID controller output and the CMAC output. The inputs for the CMAC and PID controller are: altitude, altitude command, altitude rate, and altitude rate command. In each time step k, the CMAC involves a recall process and a learning process. In the recall process, it uses the desired system output of the next time step and the actual system output as the address to generate the control signal U_{CMAC}. In the learning process, the control signal of the pitch autopilot,

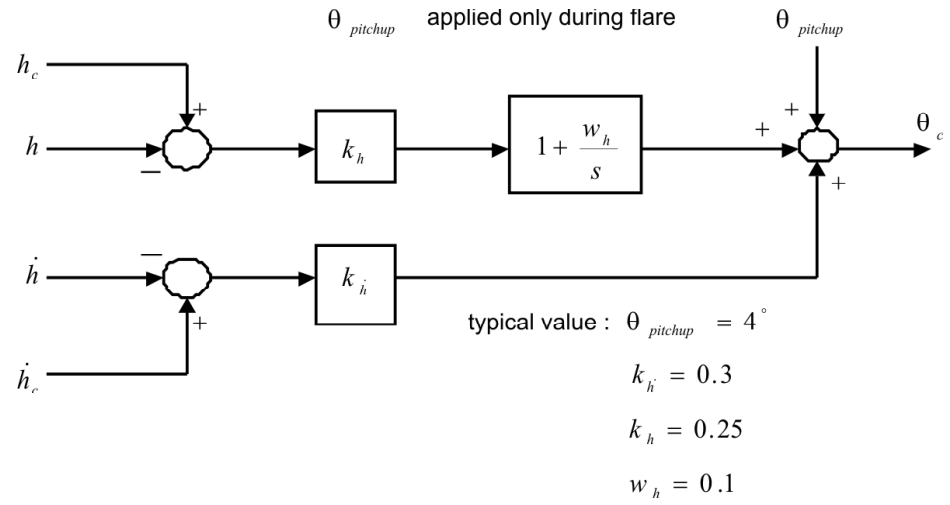

Figure 3: PID-controller

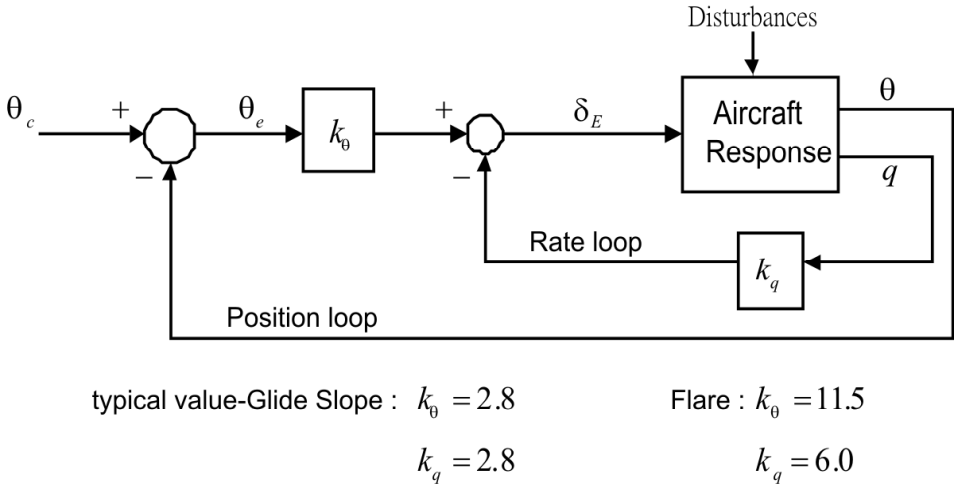

Figure 4: Pitch Autopilot

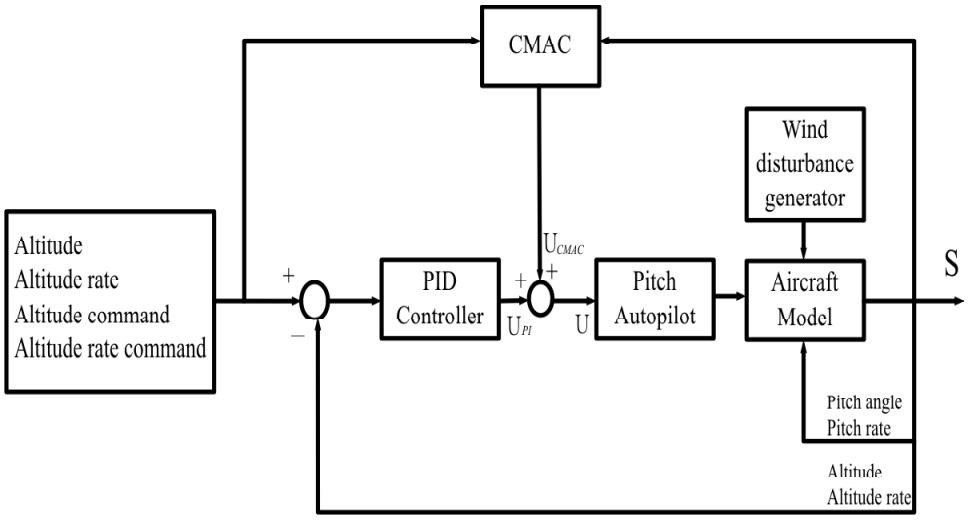

Figure 5: The CMAC Control Scheme

U, is treated as a desired output. It is used to modify the weights of CMAC stored at location which is addressed by the actual system output and the system output of the next time step. The output of the CMAC is the compensation for pitch command. When the wind turbulence is too strong, the ALS can not control the aircraft to land safely. Here we use CMAC, type-1 FCMAC, and type-2 FCMAC control schemes to improve the ability of turbulence resistance of the ALS.

3.1. Cerebellar Model Articulation Controller (CMAC)

CMAC is a type of artificial neural network proposed in the literatures [21]. It could be considered as an associative memory learning structure based on the performance of the cerebellum of human being. The function of CMAC is alike to a lookup-table technique which represents complex and nonlinear systems. And the fundamental concept of CMAC is to store information into overlapping regions in an associative approach so that stored information can easily be recalled using less storage space (memory cell). The structure of CMAC is shown in Figure 6. Manipulation of the CMAC divides the algorithm into two segments.

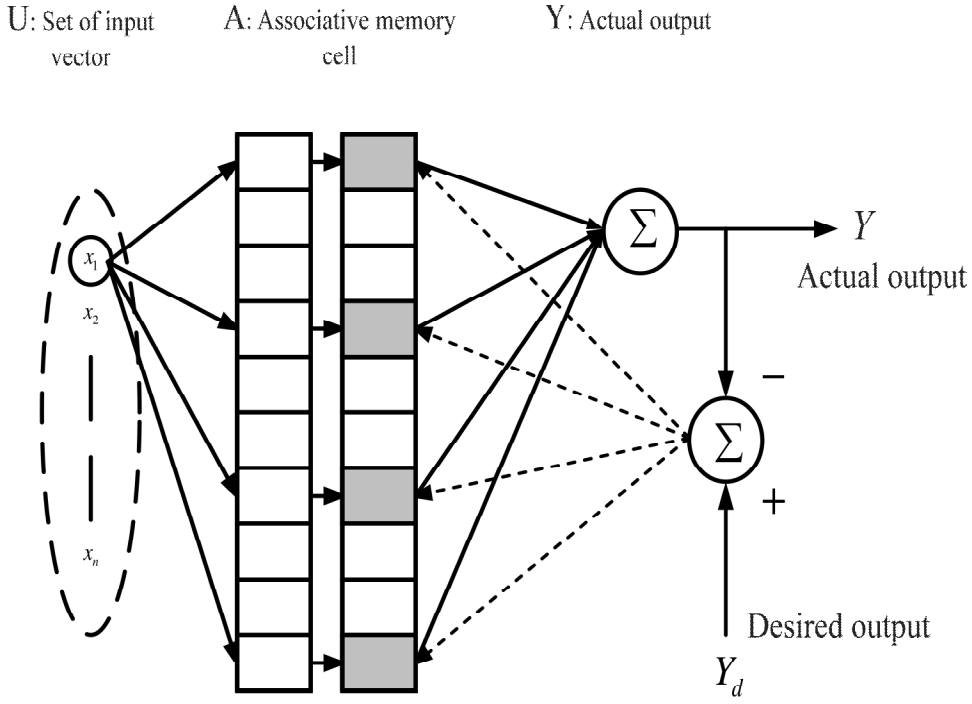

Figure 6: The Conceptual Diagram of CMAC

The first is an output generating stage. The output of CMAC can be obtained by the mapping process $U \rightarrow A \rightarrow Y$, where A stands for the M-dimensional memory cell, the $a \in A \subset R_M$ is the binary associative vector, as an address indexes in coherence with the input vector x. Let the input x address N ($N < M$) memory cells; the mapping $A \rightarrow Y$ represents the chosen weights that stored in memory cells are added together to compute the output as:

$$y(x) = \sum_{j=1}^{N} w_j a_j(x) \tag{8}$$

where w_j is the weight of the j^{th} storage hypercube and $a_j(x)$ is a binary factor indicating whether the j^{th} storage hypercube is addressed by the input x. The second is the stage of network learning in the CMAC, it is to update the addressed weights of memory cells according to the error between the desired output and the real output. Its weight updating rule is:

$$w_j^{(i)} = w_j^{(i-1)} + \frac{\alpha}{m}\left(y_d - \sum_{j=1}^{N} w_j^{(i-1)} a_j \right) \tag{9}$$

where y_d is the desired output, m is the number of addressed memory cells, α is the learning rate.

When it processes input vector of the CMAC, it simply divides it into certain blocks. The relation between input vector with these blocks is simply a crisp relation. The relation between the input condition and the association intensity is simply "activated" or "not activated". Further, an important identity of the CMAC is local generalization that derived from where nearby input vectors have some overlapping vicinity and then share some associative memory cells.

3.2. Type-1 FCMAC

The structure of type-1 FCMAC is shown in Figure 7. FCMAC is a kind of associative memory network. Not only has it faster self-learning rate than normal neural network by quantities with a few adjustments of memory weights, but also it has good local generalization ability. The function of FCMAC is similar to a look-up table, and the output of CMAC is figured from a linear combination of weights which are stored in memory. The concept of FCMAC is to store data (knowledge) into overlapped storage hypercubes (remembering region) in an associative manner such that the stored data can easily be recalled. Two kinds of operations are included in the FCMAC, one is calculating the output result and the other is learning and adjusting the weight. The output of FCMAC can be obtained by the mapping process $X \rightarrow S \rightarrow C \rightarrow W \rightarrow Y$ as follows.

Step 1: Quantization ($X \rightarrow S$): X is n-dimension input space. For the given $x = [x_1\ x_2, \ldots, x_n]^T$, $s = [s_1\ s_2, \ldots, s_n]^T$ represents the quantization vector of x. It is specified the corresponding state of each input variable before the fuzzification.

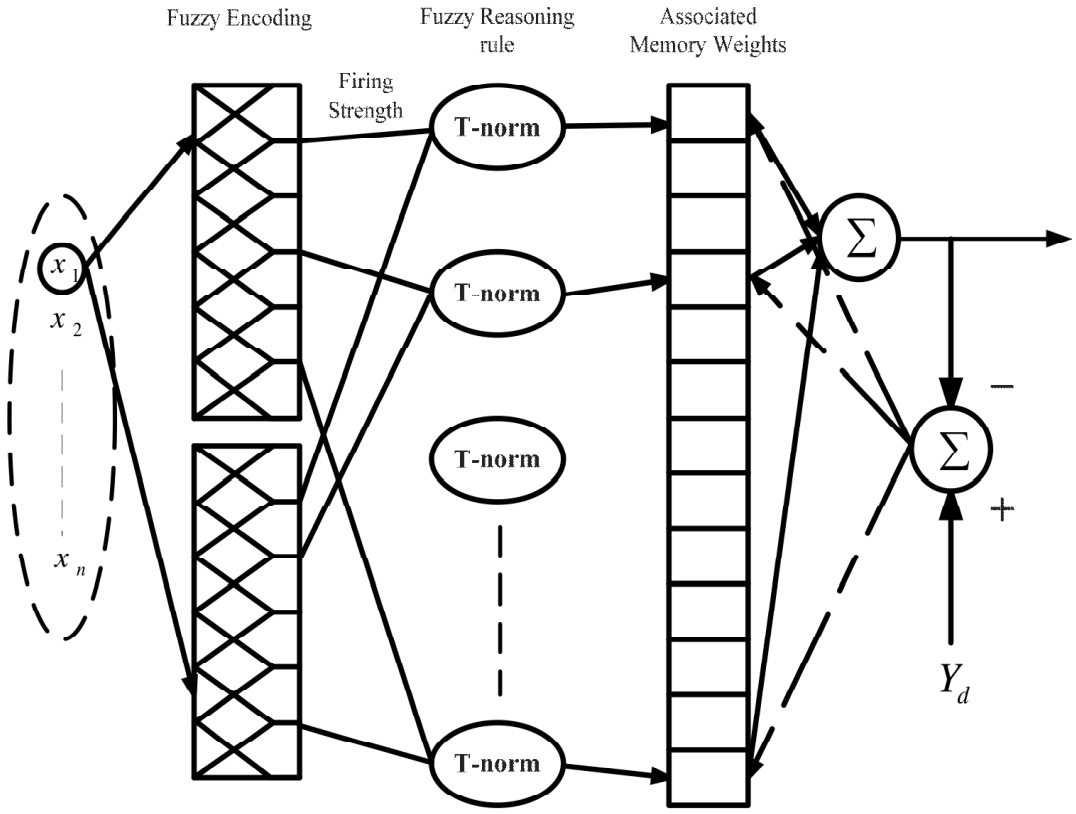

Figure 7: Conventional FCMAC Structure

Step 2: Associative Mapping segment ($S \rightarrow C$): It is to fuzzify the quantization vector which is quantized from x. FCMAC uses the fuzzification method of the fuzzy theorem as its addressing scheme. After the input vector is being fuzzified, the input state values are transformed to "firing strength", which is based on corresponding membership functions.

Step 3: Memory Weight Mapping ($C \rightarrow W$): After fuzzifying block regions, the i^{th} rule's firing strength in FCMAC could be computed as:

$$C_j(x) = c_{j1}(x_1) * c_{j2}(x_2) * \ldots c_{jn}(x_n) = \prod_{i=1}^{n} c_{j1}(x_i) \tag{10}$$

where $c_{j_i}(x_i)$ is the j^{th} membership function of the i^{th} input vector and n is the number of total states. The asterisk "*" denotes a fuzzy t-norm operator. And there are several kinds of t-norms such as the max, min and product operators. We choose the product inference method as the t-norm operator because it is easy to implement.

Step 4: Output generation with memory weight learning ($W \rightarrow Y$): Due to partial proportional fuzzy rules and existent overlap situation, more than one fuzzy rules are fired simultaneously. The consequences of multi-rules are merged by a defuzzification process. The defuzzification approach we applied is to sum assigned weights of the activated fuzzy rules on their firing strengths, denoted as $C_j(x)$. The output of network is,

$$y = \sum_{j=1}^{N} (w_j C_j(x) / \sum_{i=1}^{N} C_i(x)) \tag{11}$$

The work on learning of FCMAC is to update the memory weight according to the error between the desired output and the actual output. The weight update rule for FCMAC is as follows [22]:

$$w_j^{(i)} = w_j^{(i-1)} + \frac{\alpha}{m}(y_d - y) C_j(x) / \sum_{i=1}^{N} C_i(x) \tag{12}$$

where α is the learning rate, m is the size of floor (called generalization), y_d is the desired output.

3.3. Type-2 Fuzzy CMAC

The type-2 fuzzy theorem is utilized into CMAC structure in order to promote more accurate resolution than conventional FCMAC. The mapping procedure of type-2 FCMAC is similar to conventional FCMAC. The diagram structure of type-2 FCMAC is shown in Figure 8. Each phase of mapping is described as follows. The X is an n-dimensional input space, as shown in Figure 9. For the given $X = [x_1, x_2 \ldots. x_n]$, $S = [s_1, s_2 \ldots. s_n]$ represents the quantization vector of x. It is specified the corresponding state of each input variable before the fuzzification. Type-2 FCMAC uses the interval type-2 fuzzification method of the fuzzy theorem as its addressing scheme. After the input vector to the interval type-2 fuzzy set is being fuzzified, the input state values are transformed to upper firing strength and lower firing strength, which is based on corresponding interval type-2 membership functions. We choose the product inference method as the t-norm operator. The j^{th} rule's upper firing strength $\overline{c}^j$ and lower firing strength firing strength $\underline{c}^j$ in type-2 FCMAC could be computed as:

$$\overline{c}^j(x) = \overline{c}_{j_1}(x_1) * \overline{c}_{j_2}(x_2) * \ldots\ldots \overline{c}_{j_n}(x_n) = \prod_{i=1}^{n} \overline{c}_{j_i}(x_i) \tag{13}$$

$$\underline{c}^j(x) = \underline{c}_{j_1}(x_1) * \underline{c}_{j_2}(x_2) * \ldots\ldots \underline{c}_{j_n}(x_n) = \prod_{i=1}^{n} \underline{c}_{j_i}(x_i) \tag{14}$$

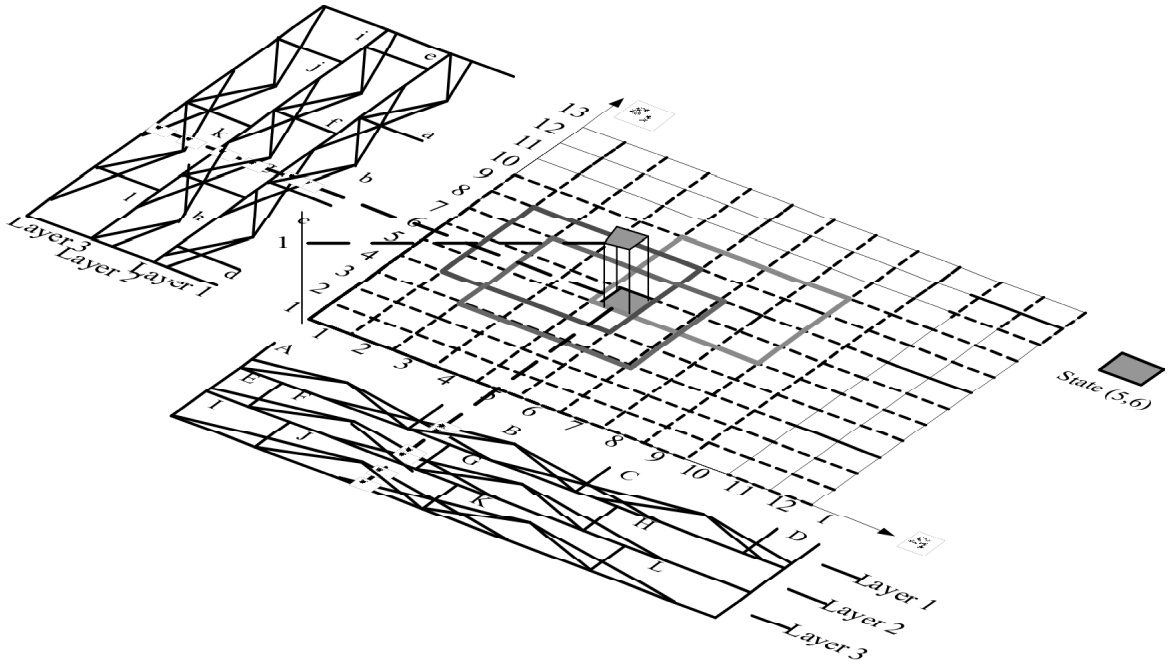

Figure 8: Diagram of type-2 FCMAC in 3-D

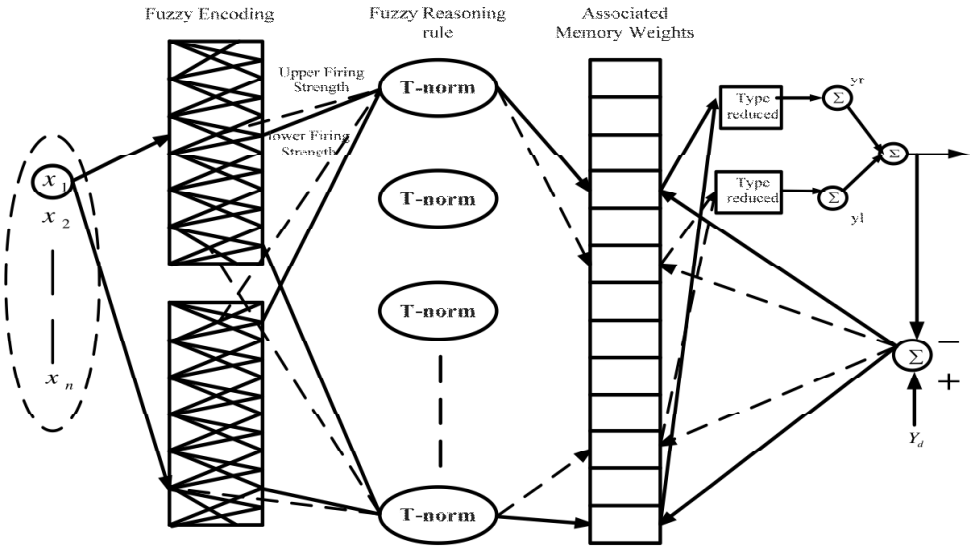

Figure 9: Architecture of type-2 FCMAC Network

The type-reduced set of the type-2 FCMAC using the center of sets type reduction :

$$y_{\cos} = [y_l, y_r] = \int_{w^1 \in [\underline{w}^1, \overline{w}^1]} \cdots \int_{w^N \in [\underline{w}^M, \overline{w}^N]} \cdots \int_{c^1 \in [\underline{c}^1, \overline{c}^1]} \cdots \int_{c^M \in [\underline{c}^M, \overline{c}^N]} 1 \Big/ \frac{\sum_{j=1}^{n} c^j w^j}{\sum_{j=1}^{n} c^j} . \quad (15)$$

It is an interval type-1 set determined by its left and right end points y_l and y_r, which can be written as follows [20]:

$$y_r = \frac{\sum_{j=1}^{N} \overline{c}^j \overline{w}^j}{\sum_{j=1}^{N} \overline{c}^j} = \frac{\sum_{j=1}^{R} \underline{c}^j \overline{w}^j + \sum_{j=R+1}^{N} \overline{c}^j \overline{w}^j}{\sum_{j=1}^{R} \underline{c}^j + \sum_{j=R+1}^{N} \overline{c}^j} \tag{16}$$

$$y_l = \frac{\sum_{j=1}^{N} \underline{c}^j \underline{w}^j}{\sum_{j=1}^{N} \underline{c}^j} = \frac{\sum_{j=1}^{L} \overline{c}^j \underline{w}^j + \sum_{j=L+1}^{N} \underline{c}^j \underline{w}^j}{\sum_{j=1}^{L} \overline{c}^j + \sum_{j=L+1}^{N} \underline{c}^j} \tag{17}$$

$\overline{w}$ and $\underline{w}$ are the corresponding weights of $\overline{c}$ and $\underline{c}$, respectively. L and R can be obtained from [20]:

Step 1: Assume that the pre-computed $\overline{w}^j$ are arranged in ascending order, i.e.,

$$\overline{w}^1 \le \overline{w}^2 \le \ldots\ldots\ldots \le \overline{w}^N$$

Step 2: Compute y_r by initially setting $\overline{c}^j = (\overline{c}^j + \underline{c}^j)/2$ for $j = 1\ldots\ldots N$ and let $y_r' = y_r$

Step 3: Find $R(1 \le R \le N-1)$ such that $\overline{w}^R \le y_r' \le \overline{w}^{R+1}$

Step 4: Compute y_r with $\overline{c}^j = \underline{c}^j$ for $j \le R$ and $\overline{c}^j = \overline{c}^j$ for $j > R$ and let $y_r'' = y_r$

Step 5: If $y_r'' \ne y_r$ then go to step 6. If $y_r'' = y_r'$ then stop and set $y_r'' \equiv y_r'$

Step 6: Set $y_r' = y_r''$ and return to Step 3.

The procedure for computing y_l is very similar to the one just given for y_r. In Step 3 find $L(1 \le L \le N-1)$ such that $\underline{w}^L \le y_l' \le \underline{w}^{L+1}$. Additionally, in Step 2 compute y_l initially setting $\underline{c}^j = (\overline{c}^j + \underline{c}^j)/2$ for $j = 1\ldots\ldots N$ and in Step 4 compute y_l with $\underline{c}^j = \overline{c}^j$ for $j \le L$ and $\underline{c}^j = \underline{c}^j$ for $j > L$.

The defuzzified output is simply the average

$$y = y_r + y_l \tag{18}$$

The work on learning of type-2 FCMAC is to update the memory weight according to the error between the desired output and the actual output. The learning rule for type-2 FCMAC is as follow:

$$\overline{w}_j^{(i)} = \overline{w}_j^{(i-1)} + \frac{\alpha}{m}(y_d - y)\overline{c}_j(x) / \sum_{j=1}^{N} \overline{c}_j(x) \tag{19}$$

$$\underline{w}_j^{(i)} = \underline{w}_j^{(i-1)} + \frac{\alpha}{m}(y_d - y)\underline{c}_j(x) / \sum_{j=1}^{N} \underline{c}_j(x) \tag{20}$$

where α is the learning rate, m is the size of floor (called generalization).

4. OPTIMAL CONTROL PARAMETERS

The GA's good properties do not stem from the use of bit strings. GA based on real number representation is called real-valued genetic algorithm, which would seem particularly natural

when optimization problems with variables in continuous search spaces are tackled. A chromosome is a vector of floating point numbers whose size is kept the same as the length of the vector, which is a solution to the problem. The operation mechanisms, reproduction, crossover, and mutation are operated by real number also. In recent years, a lot of researchers have done much improvement for crossover and mutation, in order to expect GA to have better performance. In this study, five crossover methods are applied to the evolution. They are Adewuya crossover, arithmetical crossover, average crossover, convex crossover, and blend crossover. In addition to utilize GA with these crossover methods to search the control parameters of the pitch autopilot, we also compare the differences among these five methods.

4.1. Adewuya Crossover Method

In reproduction stage, we used roulette wheel selection to choose better parents, which is according to the fitness function of populations. For each generation, the reproduction operator chooses populations that are placed into a mating pool, which is used as the basis for creating the next generation. Then, enter the next stage, crossover. The crossover in a GA is an important process. The first method we used is proposed by Adewuya [3]. The process is divided into three steps, as shown below.

Step 1: Randomly choose a gene from each individual of a matching pair in parent generation, $P_{m\alpha}$ and $P_{n\alpha}$, as crossover site.

$$pattern_1 = \left[p_{m1} \quad p_{m2} \quad ..... \quad p_{m\alpha} \quad ..... \quad p_{ms} \right] \tag{21}$$

$$pattern_2 = \left[p_{n1} \quad p_{n2} \quad ..... \quad p_{n\alpha} \quad ..... \quad p_{ns} \right] \tag{22}$$

Step 2: Calculate new values of these selected genes as follow, where β is a random number and $0 \le \beta \le 1$.

$$p_{new1} = (1-\beta) \cdot p_{m\alpha} + \beta \cdot p_{n\alpha} \tag{23}$$

$$p_{new2} = \beta \cdot p_{m\alpha} + (1-\beta) \cdot p_{n\alpha} \tag{24}$$

Step 3: Replace $P_{\mathrm{m}\alpha}$ and $P_{n\alpha}$ with P_{new1} and P_{new2}, respectively. The genes in the right side of the crossover site exchange with each other, which will obtain new offspring.

$$Newpattern_1 = \left[p_{m1} \quad p_{m2} \quad ..... \quad p_{new1} \quad ..... \quad p_{ns} \right] \tag{25}$$

$$Newpattern_2 = \left[p_{n1} \quad p_{n2} \quad ..... \quad p_{new2} \quad ..... \quad p_{ms} \right] \tag{26}$$

Finally an important process is the mutation, which permits the introduction of extra variability into the population. We pick out a population randomly, and change their gene information, but the new offspring must be in the range established after adding gene information. We use real number mutation process as follow

$$x_{new} = x_{old} + s \cdot rand\_noise \tag{27}$$

where s is the random value between 0 to 1. The fitness function that was used in this control scheme is:

Fitness = number of successful landing with different turbulence strengths. (28)

4.2. Arithmetical Crossover Method

The second one we used is the arithmetical crossover [4]. The reproduction and mutation that we used are the same as in Section 4.1. The arithmetical crossover makes the mating pair pull away or get closer. The process is shown below.

$$\text{Pull away} \quad \begin{aligned} x_1^{'} &= x_1 + \sigma \cdot (x_1 - x_2) \\ x_2^{'} &= x_2 - \sigma \cdot (x_1 - x_2) \end{aligned} \tag{29}$$

$$\text{Get closer} \quad \begin{aligned} x_1^{'} &= x_1 + \sigma \cdot (x_2 - x_1) \\ x_2^{'} &= x_2 - \sigma \cdot (x_2 - x_1) \end{aligned} \tag{30}$$

where x_1 and x_2 are the parents, $x_1^{'}$ and $x_2^{'}$ are the new offspring, and σ is a random and positive small real value. In addition, we can also use either (29) or (30) with $-1 < \sigma < 1$, which can determine pull away or get closer by the sign of σ.

4.3. Average Crossover Method

The third one we used is the average crossover. The reproduction and mutation that we used are the same as in Section 4.1. The average crossover uses a simplified model with (23) and (24) where β is 1/2. It can be obtained as follow

$$p_{new} = \frac{1}{2} \cdot (p_{m\alpha} + p_{n\alpha}) \tag{31}$$

where $P_{m\alpha}$ and $P_{n\alpha}$ are the parents, P_{new} is the new offspring.

4.4. Convex Crossover Method

The fourth one we used is the convex crossover. The reproduction and mutation that we used are the same as in Section 4.1. The convex crossover is shown below.

$$x_{new} = \gamma \cdot x_j + (1 - \gamma) \cdot x_k \tag{32}$$

where x_j and x_k are the parents, x_{new} is the new offspring, γ is a random and small real value.

4.5. Blend Crossover Method

The fifth one we used is the blend crossover. The blend crossover (BLX-α) was proposed by Eshelman and Schaffer [5]. It is a prominent crossover operator for GA, and excels in optimization of a number of standard separable functions with multimodality. The BLX-α crossover generates offspring as following process.

$$X_i^1 = \min(x_i^1, x_i^2) - \alpha \cdot d_i \tag{33}$$

$$X_i^2 = \max(x_i^1, x_i^2) + \alpha \cdot d_i \tag{34}$$

$$d_i = \left| x_i^1 - x_i^2 \right| \tag{35}$$

where x^1 and x^2 are chosen randomly from the population, x_i^1 and x_i^2 are the i-th elements of x^1 and x^2, respectively. The value of each element x_i^c of the offspring vector x^c is uniformly sampled from the interval $\left[X_i^1, X_i^2 \right]$. α is a positive parameter, which is suggested to be 0.5.

5. HARDWARE REALIZATION

VisSim is a Windows-based program for the modeling and simulation of complex nonlinear dynamic systems [23]. VisSim combines an intuitive drag & drop block diagram interface with a powerful simulation engine. The visual block diagram interface offers a direct method for constructing, modifying and maintaining system models. The simulation engine provides fast

and accurate solutions for linear, nonlinear, continuous time, discrete time, time varying and hybrid system designs. In here, we build a aircraft dynamic model under VisSim software, and realize the conventional PID controller by the same manner. Then, the intelligent controller design using C language and its realization by DSP are presented.

Since the invention of the transistor and integrated circuit, digital signal processing functions have been implemented on many hardware platforms ranging from special-purpose architectures to general-purpose computers. It was not until all of the functionality (arithmetic, addressing, control, I/O, data storage, control storage) could be realized on a single chip that DSP could become an alternative to analog signal processing for the wide span of applications that we see today. In this study, we use TI TMS320LF2407 chip to perform our task. The 2407A devices offer the enhanced TMS320DSP architectural design of the C2xx core CPU for low-cost, low-power, and high-performance processing capabilities. Moreover, it offers suitable array of memory sizes and peripherals tailored to meet the specific performance points required by various applications. The TMS320LF2407 operates at 40 MHz (40 MIPS), has 4 to16 PWM output channels and has serial communication capabilities. In addition, the TMS320LF2407 contains a 10-bits analog-to-digital converter (ADC) having a minimum conversion time of 500 ns that offers up to 16 channels of analog input. Furthermore, the working process and externals of the eZdspTMLF2407A board are shown in Figure 10 and Figure 11, respectively.

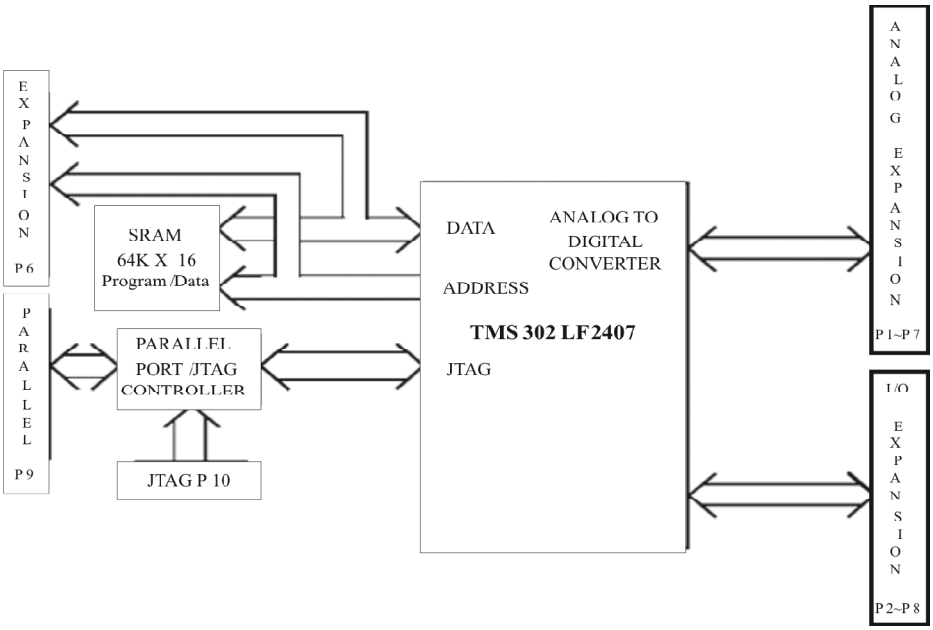

Figure 10: Working Process of the eZdspTMLF2407A board

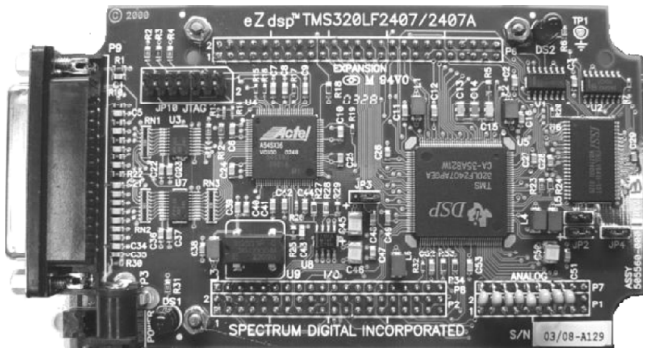

Figure 11: Externals of the eZdspTMLF2407A board

There are three basic steps in the development of a DSP algorithm: (1). Create the system you want to execute on the target DSP; (2). Generate the C source code from the system; (3). Compile and link the C source code to produce an executable file. If step 1 performed using available blocks form VisSim software, thus, steps 2 and 3 are automatically performed by VisSim. The core of VS-ECD2407 is TI TMS320LF2407A, which is a 16 bits fixed-point DSP. While designing the controller of an aircraft, it must consider the problem of the fixed point. Because molds of VisSim are all floating-point operation, we must match VisSim/Fixed-Point software to design molds of fixed-point flight controller. Figure 12 shows DSP development of the procedure entirely, which is divided into the following steps.

Step 1: In VisSim major software, the fixed-point controller molds of an aircraft are designed by VisSim/TI C2000 Rapid Prototype and VisSim/Fixed Point.

Step 2: CCStudio can make fixed-point controller molds to do compiling, analyzing, debugging, and demonstrating, and generate *.c code and *.out code finally.

Step 3: Generate DSP controller molds which include *.out code, and replace original fixed-point controller molds.

Step 4: Download *.out code to TI TMS320LF2407A embedded flash memory by JTAG.

Step 5: Utilize DSP controller to control automatic landing system and show real-time relevant flight behavior.

Through above-mentioned DSP controller's procedure of development, whole real-time DSP hardware in-the-loop mode is shown in Figure 13. From VisSim development platform, it will transmit information, altitude, altitude rate, altitude and altitude rate commands to the VS-ECD2407 via connecting a JTAG between the computer and the VS-ECD2407. After DSP processing, it passes the pitch command and adjusts the angle of elevator back to pitch autopilot in VisSim via JTAG., and enable an aircraft to follow landing trajectory to land. The advantages of DSP are fast operation, powerful instruction, fixed addressing ability at a high speed, parallel process, etc. These can improve processing speed and accuracy greatly, such that it can be applied to real-time control. In the simulation selection items of the VisSim, one can choose "Run in Real Time", and while designing fixed-point DSP controller, choose the exchange frequency of the datum between DSP and PC properly, then real-time control can be performed.

6. SIMULATION RESULTS

The aircraft starts the initial states of the ALS as follows: the flight height is 500 ft, the horizontal position before touching the ground is 9240 ft, the flight angle is -3 degrees, and the speed of the aircraft is 234.7 ft/sec. Successful touchdown landing conditions are defined as follows:

(1) $-3 \le \dot{h}_{TD} \le -1$ (ft/sec) (2) $-300 \le x_{TD}(T) \le 1000$ (ft)

(3) $200 \le V_{TD}(T) \le 270$ (ft/sec) (4) $-10 \le \theta_{TD}(T) \le 5$ (degrees)

where T is the time at touchdown, $\dot{h}_{TD}$ is vertical speed, x_{TD} is the horizontal position, V_{TD} is the horizontal speed, and θ_{TD} is the pitch angle.

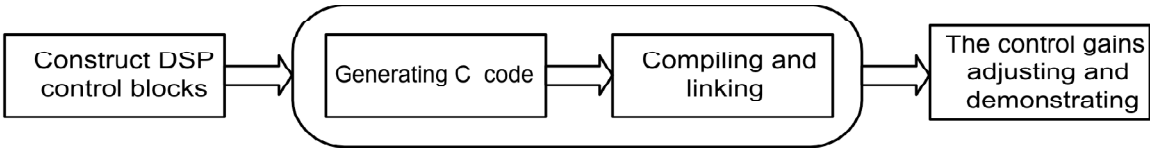

Figure 12: The Flow Chart of VisSim/DSP Procedure Development

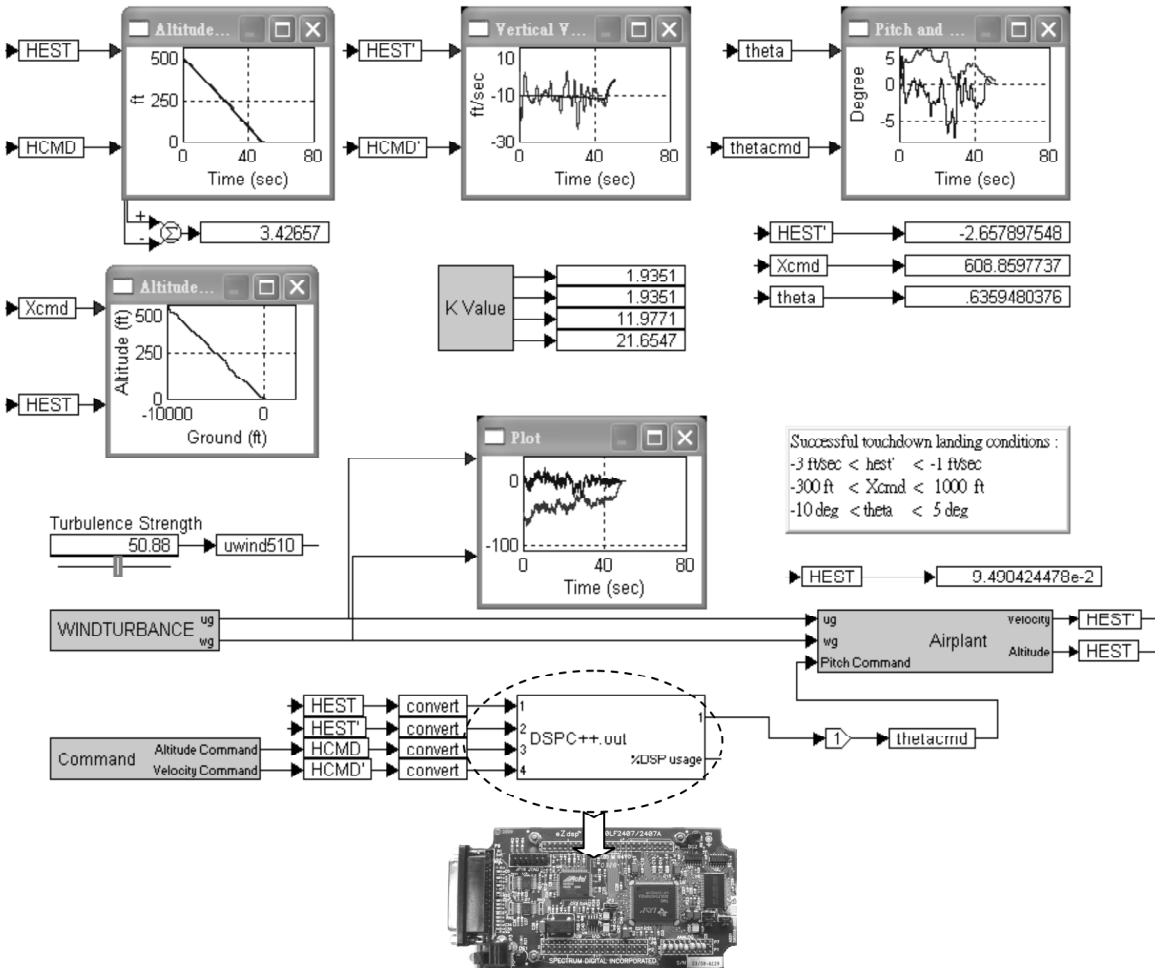

Figure 13:DSP Hardware in-the-loop Mode

6.1. Conventional PID Controller

Table 1 shows the results from using different wind turbulence speeds. The conventional PID controller with original control gains can only successfully guide an aircraft flying through wind speeds of 0 ft/sec to 30 ft/sec [13]. If the wind speed is higher than 30 ft/sec, the ALS will be unable to guide an aircraft to land safely. An aircraft is safe to land in the wind turbulence speed at 30 ft/sec, as shown in Figure 14 to Figure 17.

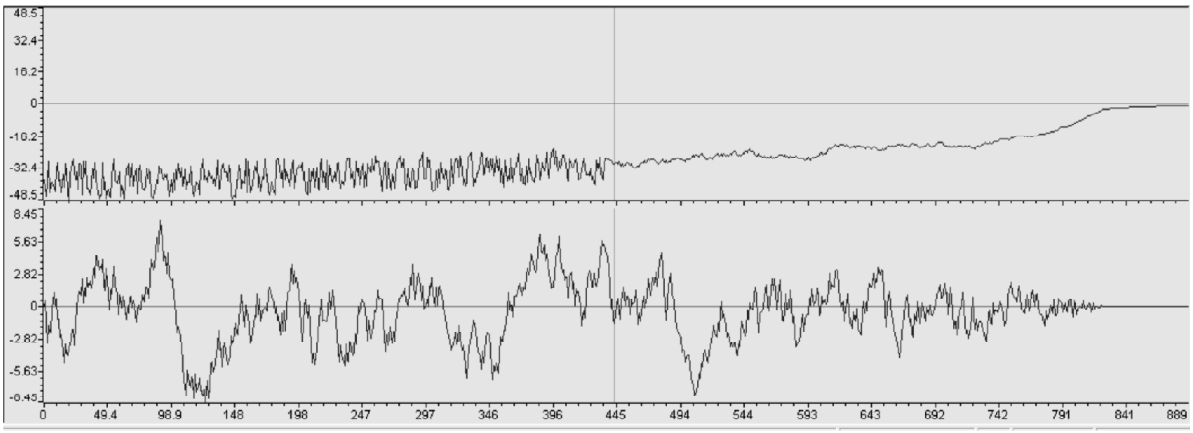

Figure 14: Turbulence Profile (30 ft/sec)

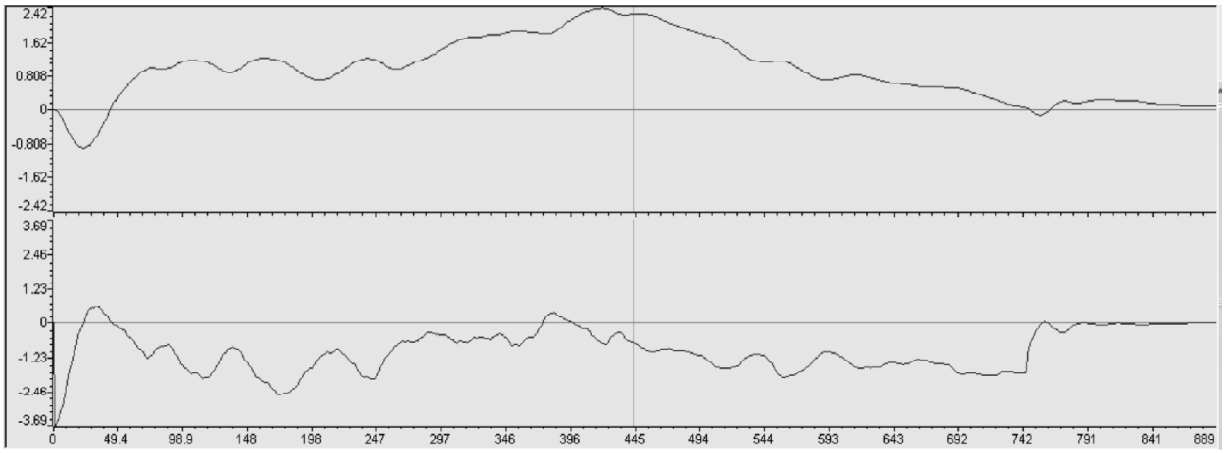

Figure 15: Aircraft pitch (top) and Command (bottom)

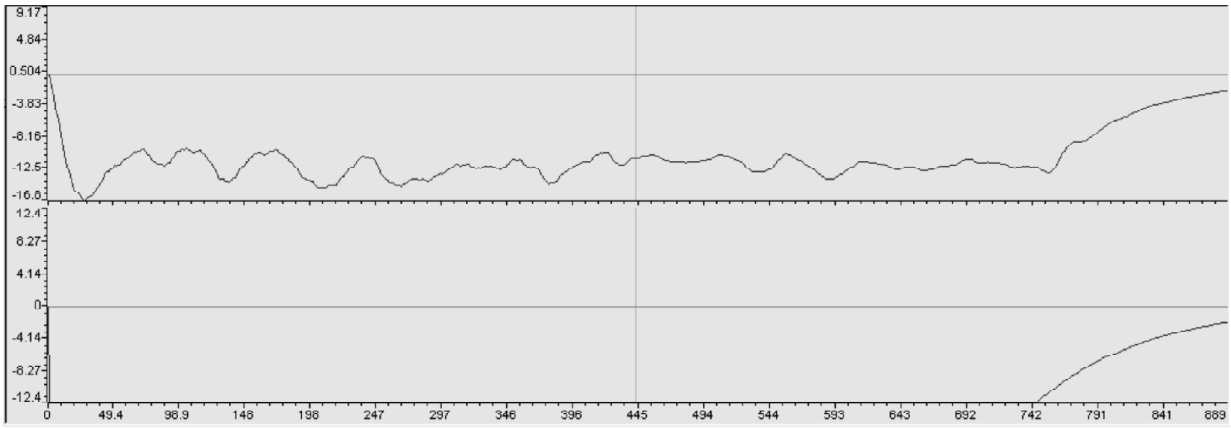

Figure 16: Vertical Velocity (top) and Command

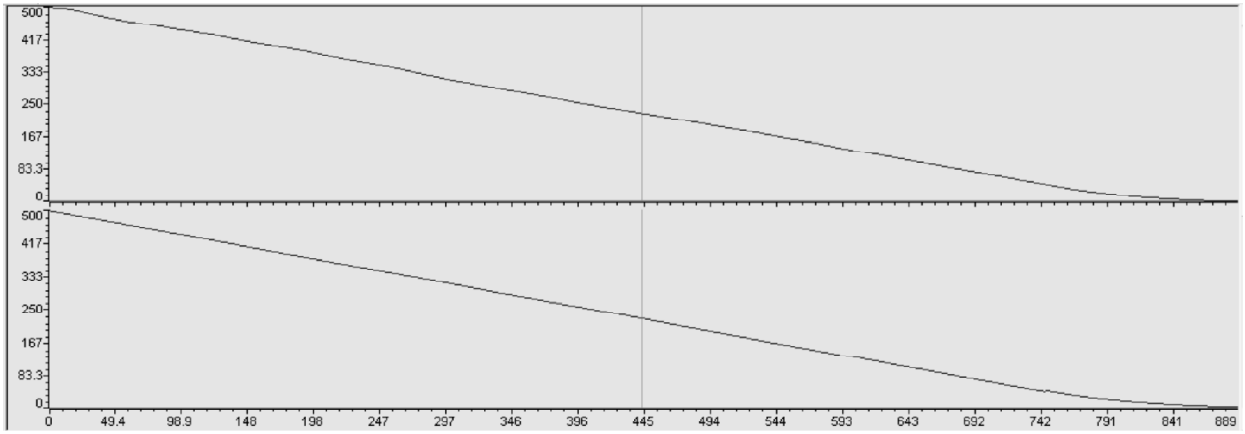

Figure 17: Aircraft Altitude (top) and Command

6.2. Different CMAC Controllers

The results from using CMAC controller are shown in Table 2. The type-2 FCMAC control scheme can successfully guide the aircraft flying through wind speeds of 0 ft/sec to 100 ft/sec while the type-1 FCMAC can only reach 90 ft/sec [22], as shown in Table 3. Table 4 shows the results from using type-2 FCMAC. The situations at wind turbulence 100 ft/sec are that the pitch angle is 2.05 degrees, vertical speed is -2.21 ft/sec, horizontal velocity is 234.68 ft/sec, and horizontal position at touchdown is 937 ft.

Table 1
The Results from using Conventional PID Controller (Control Gains: $K1$ = 2.8, $K2$ = 2.8, $K3$ = 11.5, $K4$ = 6.0)

Wind speed	*Landing point (ft)*	*Aircraft vertical speed (ft/sec)*	*Pitch angle (degree)*
0	797	-2.83	-1.41
10	910	-2.55	-0.85
20	809	-2.38	-0.59
30	844	-2.19	-0.17

Table 2
Results from using CMAC Controller

Wind speed	*Landing point (ft)*	*Aircraft vertical speed (ft/sec)*	*Pitch angle (degree)*
0	854	-2.55	-0.96
10	762	-2.76	-0.93
20	774	-2.51	-0.61
30	844	-2.72	-0.41
40	691	-1.93	0.21
50	586	-2.26	0.87
58	844	-2.58	0.98

Table 3
Results from using type-1 FCMAC Control

Wind speed	*Landing point (ft)*	*Aircraft vertical speed (ft/sec)*	*Pitch angle (degree)*
10	797	-2.83	-1.41
30	938	-1.54	-0.58
50	891	-2.13	0.47
70	691	-2.21	1.41
90	926	-1.99	1.34

Table 4
The Results from using type-2 FCMAC Control

Wind speed	*Landing point (ft)*	*Aircraft vertical speed (ft/sec)*	*Pitch angle (degree)*
20	855	-2.51	-0.58
40	726	-2.44	0.03
60	996	-2.00	0.50
80	890	-1.71	1.39
100	937	-2.21	2.05

Figure 18. Evolutionary learning process of the CMAC control scheme

Table 5
Fixed Number of Generations

	Adewuya crossover	*Arithmetical crossover*	*Average crossover*	*Convex crossover*	*Blend crossover*
CPU timesRGA/Total	2.04%	1.61%	1.51%	1.75%	1.62%
Error (%)	9.28%	9.04%	9.06%	9.00%	9.20%
Max intensity (ft/sec)	70	70	65	70	75

6.3. Evolutionary Computation Based Controller

In evolutionary computation, regardless of binary-coded type or real-valued type, many improved methods were proposed to make the evolution more perfect with better parameter search ability. In this section, we put focus on crossover method of real-valued genetic algorithm. First of all, the relevant initial parameters of the real-valued genetic algorithm are fixed, and evolution procedures which are reproduction and mutation are all unanimous. The flow chart of a complete evolution is shown in Figure 18. We utilize five crossover principles to search optimal control

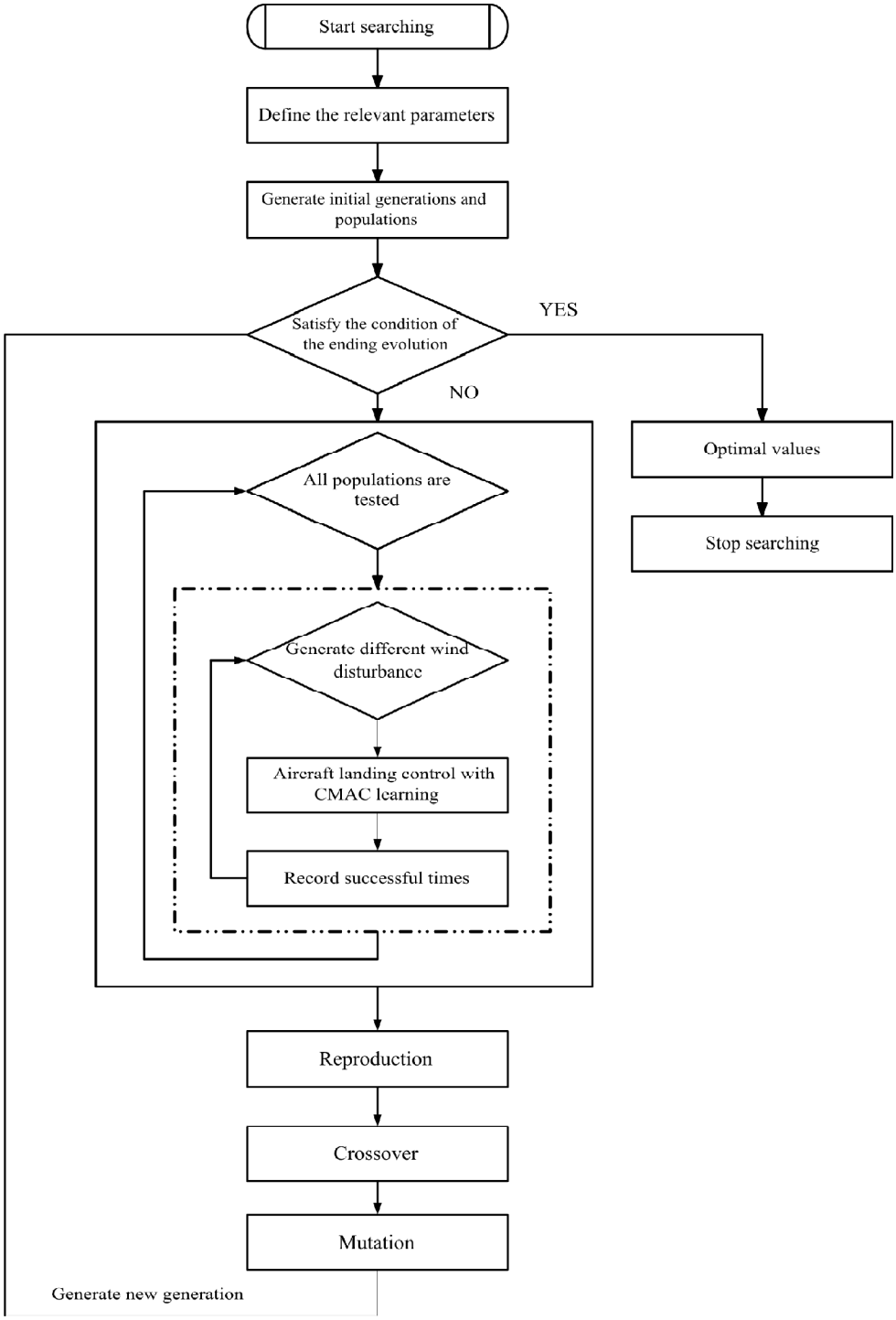

Figure 18: Evolutionary Learning Process of the CMAC Control Scheme

gains separately which an aircraft uses the CMAC controller under wind turbulence, and compare differences between these methods. We fixed number of generation of evolution to calculate execution time (CPU time) that each crossover method requires.

Table 5 and Table 6 show the comparison results of using different crossover methods. All cases are tested by 10 independent runs. The results shown in Table 5 and Table 6 are averaged values of these independent runs. We fixed the number of generation to 10 for comparing the CPU time, as shown in Table 5. Table 6 shows the required number of generation until optimal control gains are obtained. From Table 5, the CPU time of the average crossover is the least. From the comparison of the error, which is the difference between altitude command and actual altitude under wind turbulence speeds at 70 ft/sec, the least is the convex crossover. From Table 6, the Adewuya crossover can overcome the most intense of wind turbulence to 80 ft/sec, and does not need more generations in convergence. In fact, the CPU time of these five crossover methods are similar, and it only takes a little bit of portion out of total execution time. Furthermore, execution speed of new developed computer increases continuously. The execution time of the evolution is relative to the speed of CPU and the size of RAM. Therefore, the difference lies in the parameters that are searched. Moreover, Table 5 and Table 6 show comparison results of percentage of CPU time, which has slight difference. We carry out other application software during searching optimal control gains. The result of sharing a part of calculation resources causes differences. In addition, by the change within unit time, wind turbulence is violent than wind shear. Searching optimal control gains under wind turbulence environment is more difficult, and needs more generations. Since Adewuya crossover method has best performance than others, it is used in FCMAC control scheme. Tables 7 and 8 show the results from using type-1 FCMAC and type-2 FCMAC by Adewuya crossover method, respectively. Type-2 fuzzy CMAC with optimal control gains can guide the aircraft overcome turbulence to 114 ft/sec, as shown in Figures 19 to 22.

Table 6
Required Number of Generations

	Adewuya crossover	*Arithmetical crossover*	*Average crossover*	*Convex crossover*	*Blend crossover*
CPU timesRGA / Total	1.84%	1.63%	1.61%	1.43%	1.94%
Error (%)	9.21%	9.27%	9.09%	9.10%	9.20%
Max intensity (ft/sec)	80	75	75	75	75
Required generations	23	18	35	31	25

Table 7
The Results from using type-1 FCMAC with Optimal Control Gains in TMS320C6713 DSP Board (Optimal Control Gains: $K1$ =2.5409, $K2$=6.8029, $K3$=10.7398, $K4$=13.4932)

Wind speed	*Landing point (ft)*	*Aircraft vertical speed (ft/sec)*	*Pitch angle (degree)*
10	785.2276	-2.6398	-1.0054
20	738.2920	-2.5844	-0.6968
30	861.7598	-2.4173	-0.2652
40	693.1877	-2.1031	0.3418
50	814.1143	-2.5232	0.3842
60	820.4292	-1.8753	1.0607
70	798.7611	-1.9576	1.3403
80	738.2920	-1.6419	2.0949
90	961.2381	-1.5932	1.6522
100	849.5021	-1.4839	2.0451
105	736.9072	-1.6998	2.5472

Table 8
The Results from using type-2 FCMAC with Optimal Control Gains in TMS320C6713 DSP board
($K1$ = 2.0738; $K2$ = 2.0738; $K3$ = 8.4802; $K4$ = 14.8921)

Wind speed	*Landing point (ft)*	*Aircraft vertical speed (ft/sec)*	*Pitch angle (degree)*
10	713.4394	-2.7149	-1.0181
20	658.6411	-2.5085	-0.5884
30	689.9717	-2.5961	-0.3657
40	796.1005	-2.6263	0.0399
50	867.3648	-1.7812	0.6581
60	926.0343	-1.6648	0.7523
70	937.7682	-1.5684	1.0462
80	738.2920	-2.1779	1.3978
90	722.0064	-2.7847	2.5582
100	867.3648	-1.6653	2.1311
110	703.6141	-2.5363	2.2396
114	838.8158	-1.8290	2.9149

7. CONCLUSIONS

The purpose of this paper is to investigate the use of evolution computation and DSP with CMACs in aircraft automatic landing system. Current flight control law is adopted in the intelligent controller design. The proposed controllers are implemented in a DSP. Tracking performance and adaptive capability are demonstrated through hardware simulations. By using PID, CMAC, type-1 FCMAC, and type-2 FCMAC, the wind speed of turbulence limits are 30, 58, 90, and 100 ft/sec, respectively. While the adaptive neural network controller and fuzzy neural nework controller can overcome up to 75 ft/s [24-25], and the recurrent neural network controller can overcome to 60ft/sec [26]. In this study, optimal control gains CMAC control scheme can reach 80 ft/sec, type-1 FCMAC can reach 105 ft/sec, and the type-2 FCMAC can reach 114 ft/sec. The proposed controllers have better performance than previous works. The intelligent controller can be used to replace the conventional controller. The proposed intelligent control scheme can act as an experienced pilot and guide the aircraft to a safe landing in severe wind turbulence environment.

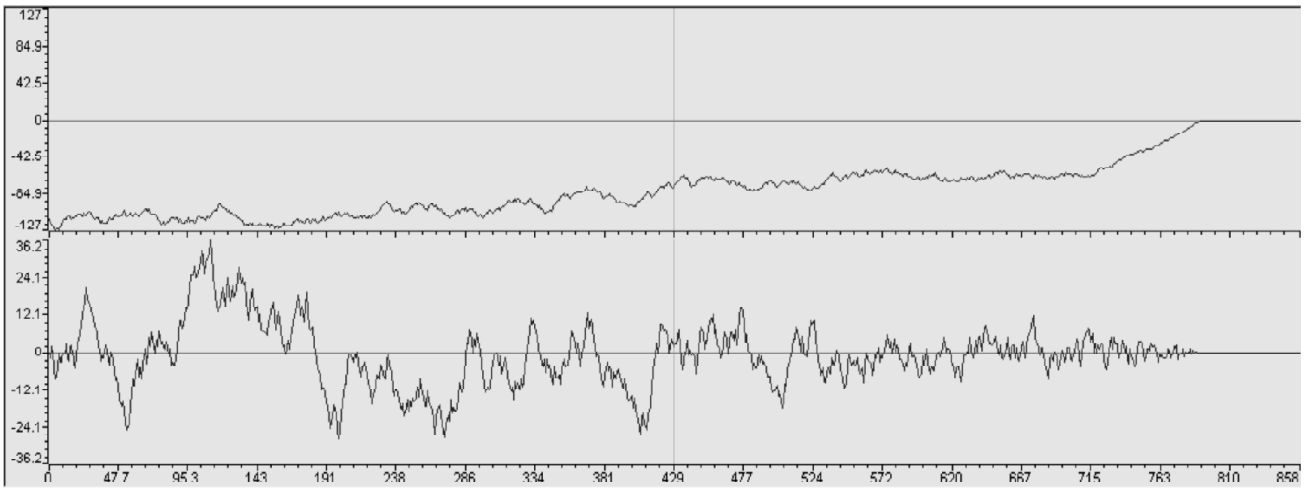

Figure 19: Turbulence Profile (100 ft/sec)

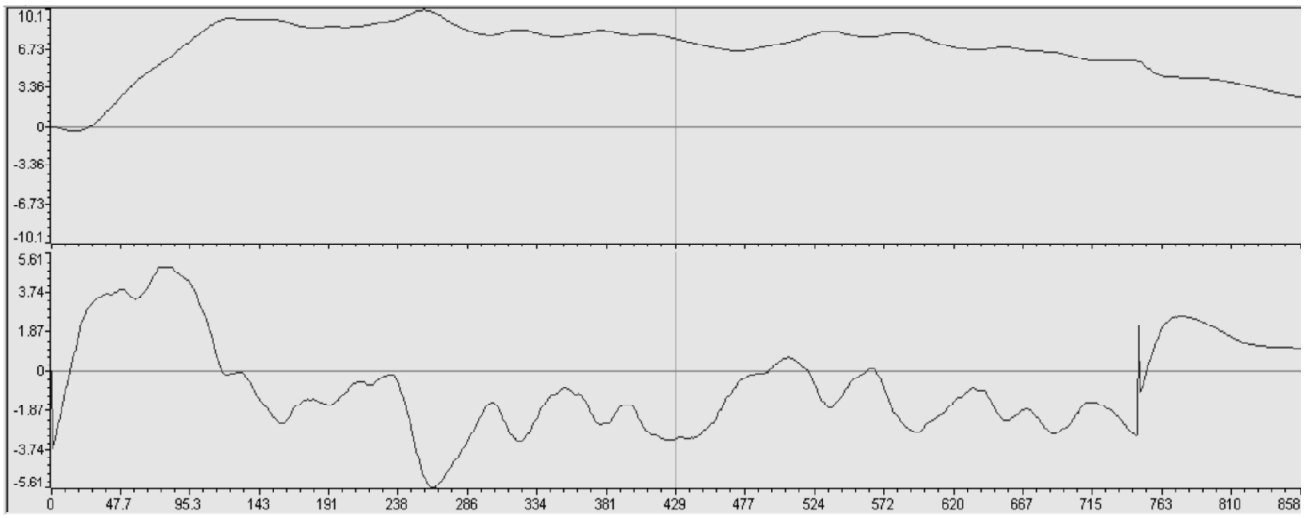

Figure 20: Aircraft Pitch (top) and Command (bottom)

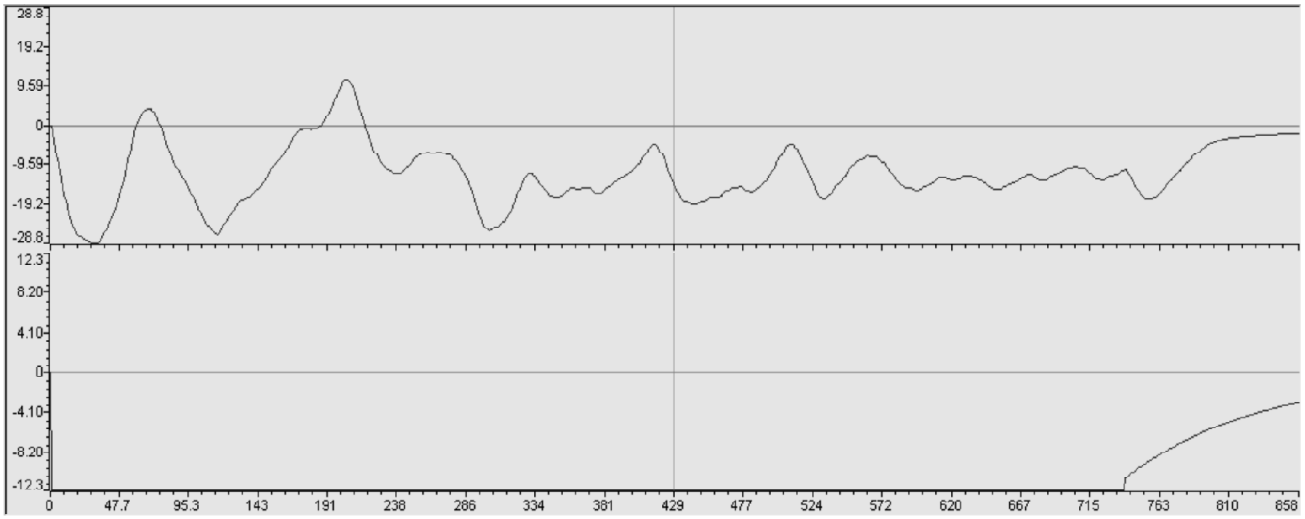

Figure 21: Vertical Velocity (top) and Command

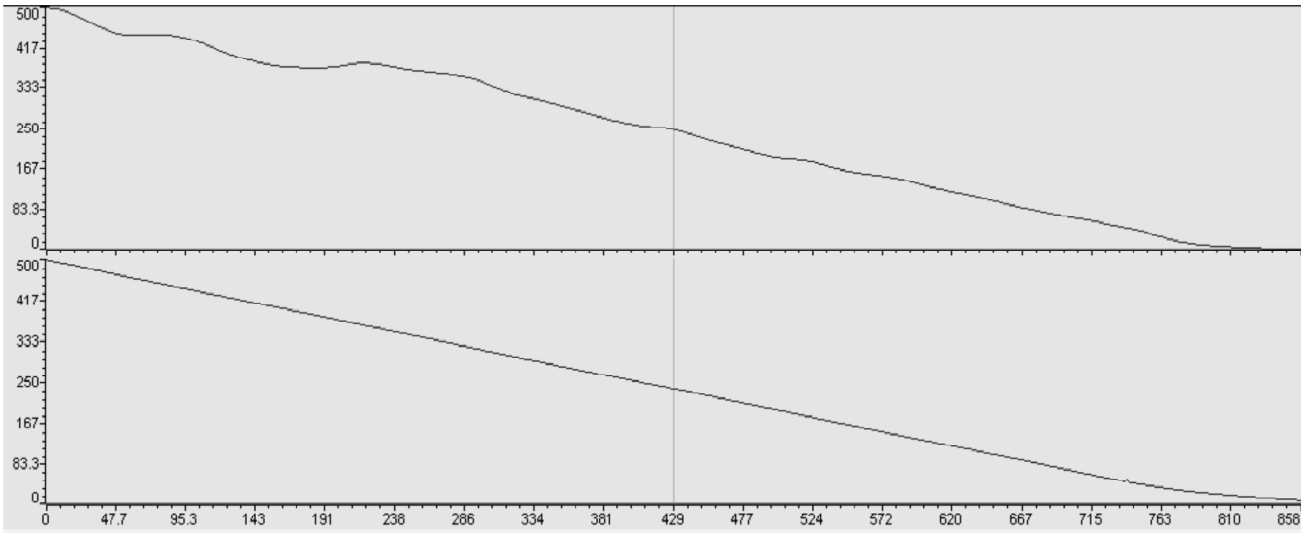

Figure 22: Aircraft Altitude (top) and Command

References

[1] Holland, J. H.: Outline for a Logical Theory of Adaptive Systems. *Journal of the Association for Computing Machinery*, **3**, 297-314, 1962.

[2] Holland, J. H.: Adaptation in Natural and Artificial Systems. Ann Arbor, MI, University of Michigan Press, 1975.

[3] Adewuya, A. A.: New Methods in Genetic Search with Real-valued Chromosomes. M. S. Thesis, Dept. of Mechanical Engineering, Massachusetts Institute of Technology, 1996.

[4] Michalewicz, Z.: Genetic Algorithm + Data Structure = Evolution Programs. New York, Springer-Verlag, 1992.

[5] Eshelman, L. J., Schaffer, J. D.: Real-coded Genetic Algorithms and Interval-schemata. *Foundations of Genetic Algorithms* **2**, 187-202, 1993.

[6] Boeing Publication: Statistical Summary of Commercial Jet Airplane Accidents. Worldwide Operations 1959-1999, 2000.

[7] Ionita, S., Sofron, E.: The Fuzzy Model for Aircraft Landing Control. Proc. AFSS International Conference on Fuzzy Systems, pp. 47-54, 2002.

[8] Nho, K., Agarwal, R. K.: Automatic Landing System Design Using Fuzzy Logic. *Journal of Guidance, Control, and Dynamics*. **23**, 298-304, 2000.

[9] Cohen, C. E. *et al.*: Automatic Landing of a 737 Using GNSS Integrity Beacons, Proc. ISPA, 1995.

[10] DDC-I: Advanced Auto Landing System from Swiss Federal Aircraft Factory, Real-Time Journal, Sprint, 1995.

[11] Asai, S., *et al.*: Development of Flight Control System for Automatic Landing Flight Experiment, *Mitsubishi Heavy Industries Technical Review*, **34**(3), 1997.

[12] Kaufmann, D. N., McNally, B. D.: Flight Test Evaluation of the Stanford University and United Airlines Differential GPS Category III Auotmatic Landing System, NASA Technical Memorandum 110355, June 1995.

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

[14] Juang, J. G., Chin, K. C.: Intelligent Landing Control Based on Neural-Fuzzy-GA Hybrid System, *Proc. IEEE International Joint Conference on Neural Networks*, **3**, 1781-1786, 2004.

[15] Chaturvedi, D. K., Chauhan, R., Kalra, P. K.: Application of Generalized Neural Network for Aircraft Landing Control System, Soft Computing, **6**, 441-118, 2002.

[16] Izadi, H., Pakmehr, M., Sadati, N.: Optimal Neuro-Controller in Longitudinal Autolanding of a Commercial Jet Transport, *Proc. IEEE International Conference on Control Applications*, CD-000202, pp. 1-6, 2003.

[17] Iiguni, Y., Akiyoshi, H., Adachi, N.: An Intelligent Landing System Based on Human Skill Model, *IEEE Transactions on Aerospace and Electronic Systems*, **34**(3), 877-882, 1998.

[18] Liu, Z., Zhang, Y., Wang, Y.: A Type-2 Fuzzy Switching Control System for Biped Robots, *IEEE Transactions on Systems, Man, and Cybernetics—Part C*, **37**(6), 1202-1213, 2007.

[19] Wang, C. H., Cheng, C. S., Lee, T. T.: Dynamical Optimal Training for Interval Type-2 Fuzzy Neural Network, *IEEE Transactions on Systems, Man, and Cybernetics—Part C*, **34**(3), 1462-1477, 2004.

[20] Liang, Q., Mendel, J.: Interval Type-2 Fuzzy Logic Systems: Theory and Design, *IEEE Transactions on Fuzzy Systems*, **8**(5), 535–550, 2000.

[21] Albus, J. S.: A New Approach to Manipulator Control: the Cerebellar Model Articulation Control (CMAC), *ASME Journal of Dynamic Systems, Measurement, and Control*, **97**, 220-227, 1975.

[22] Juang, J. G., Lin, W. P.: Aircraft Landing Control Based on CMAC and GA Techniques, Proc. of IFAC WC 2008, P1730, Aug. 2008.

[23] Visual Solutions, Inc.: VisSim User's Guide-Version 5.0, 2002.

[24] Juang, J.G., Chien, L.H., Lin, F.: Automatic Landing Control System Design Using Adaptive Neural Network and Its Hardware Realization, *IEEE Systems Journal.*, **5**(2), 266-277, 2011.

[25] Juang, J. G., Chin, K. C.: Intelligent Landing Control Based on Neural-Fuzzy-GA Hybrid System, *Proc. of IEEE International Joint Conference on Neural Networks*, **3**, 1781-1786, 2004.

[26] Juang, J. G., Chiou, H. K., Chien, L. H.: Analysis and Comparison of Aircraft Landing Control Using Recurrent Neural Networks and Genetic Algorithms Approaches, *Neurocomputing*, **71**, 3224-3238, 2008.