Controlling the Angle of Attack of an Aircraft Using Genetic Algorithm Based Flight Controller

S. Swain, P. S Khuntia

Abstract—In this paper, the unstable angle of attack of a FOXTROT aircraft is controlled by using Genetic Algorithm based flight controller and the result is compared with the conventional techniques like Tyreus-Luyben (TL), Ziegler-Nichols (ZN) and Interpolation Rule (IR) for tuning the PID controller. In addition, the performance indices like Mean Square Error (MSE), Integral Square Error (ISE), and Integral Absolute Time Error (IATE) etc. are improved by using Genetic Algorithm. It was established that the error by using GA is very less as compared to the conventional techniques thereby improving the performance indices of the dynamic system.

Keywords—Angle of Attack, Genetic Algorithm, Performance Indices, PID Controller.

I. INTRODUCTION


In this paper, Genetic Algorithm is applied for controlling the angle of attack and improving the performance indices of the system and finally the results are compared with the conventional techniques like ZN, TL, and IR. It was established that GA gives excellent results and improves the performance indices as compared to the conventional methods like Tyreus-Luyben (TL), Ziegler-Nichols (ZN) and Interpolation Rule (IR) for tuning the PID controller.

II. BLOCK DIAGRAM OF ANGLE OF ATTACK

The block diagram of the system with disturbance shown in Fig. 1 describes the feedback control system of angle of attack of an aircraft.

![Block diagram of angle of attack control system](image-url)

Fig. 1 Block diagram of angle of attack control system: $\delta_E(s)$ = The deflection of elevator as commanded by the pilot; $\alpha$ = The actual angle of attack of the aircraft; $G(s)$ = The open loop transfer function between the deflection of the elevator; $\delta_E$ and the angle of attack $\alpha$; $C(s)$ = The PI controller to be designed (tuned); $G(s)$ = The transfer function of the disturbance which occurs after some interval of the given input $\delta_E$. 

Mr. S. Swain is with the Electrical Engineering of Synergy Institute of Technology, Bhubaneswar, Odisha, India (phone: 0671-2356150; fax: 0671-2356150; e-mail: swaisnubash@gmail.com).

Dr. P. S. Khuntia is with GMRIT, Rajam, Andhra Pradesh, India. He is now with the Department of Electronics and Telecommunication (e-mail: parthsarathi_K@yahoo.com).
III. RELATION BETWEEN ANGLE OF ATTACK AND ELEVATOR DEFORMATION

Angle of attack [17] of an aircraft specifies the angle between the chord line of the wing of a fixed-wing aircraft and the vector representing the relative motion between the aircraft and the atmosphere. The angle of attack is controlled by the deflection in control surface (elevator). The description of angle of attack is shown in Fig. 2.

![Fig. 2 Description of angle of attack](image_url)

The short period approximation [17] consists of assuming that any variations in speed of the aircraft (U) which arise in airspeed as a result of control surface deflection, atmospheric turbulence or just aircraft motion, are so small that any terms in the equation of motion involving ‘U’ are negligible. In other words, the approximation assumes that short period transients are of sufficiently short duration that speed of the aircraft $U_0$ remain essentially constant, i.e., $U = 0$. Thus, the equations of longitudinal motion in terms of stability may now be written as:

$$w = Z_w w + U_0 q + Z_{q} \delta_E$$  \hspace{1cm} (1)

$$q = M_w w + M_q q + M_{q} \delta_E$$ \hspace{1cm} (2)

If the state vector for short period motion is now defined as

$$x = \begin{bmatrix} w \\ q \end{bmatrix}$$

and the control vector ‘u’ is taken as the elevator deflection $\delta_E$, then (1) and (2) may be written as a state equation:

$$\dot{x} = Ax + Bu$$ \hspace{1cm} (3)

In (3), the values of $A$ and $B$ are

$$A = \begin{bmatrix} Z_w & U_0 \\ (M_w + M_{q} Z_w) & (M_q + U_0 M_w) \end{bmatrix}$$

$$B = \begin{bmatrix} Z_{q} \\ M_{q} \end{bmatrix}$$

If $sI - A$ is a Hurwitz matrix, then $\Delta_p(s) = \text{det}[sI - A]$ and $\alpha(s)$ are the transfer functions of longitudinal dynamics of FOXTROT aircraft.

Using the stability derivatives of longitudinal dynamics of FOXTROT as shown in Table I and substituting the values in (4), the transfer function $G(s)$ between $\delta_E$ and $\alpha$ for flight condition-1 is given by

$$G(s) = \frac{2Z_{0}\alpha_0}{s(2.0302s^2 + 102.8)} = \frac{3.604s + 182.5}{s^2 + 0.901s + 0.5633}$$

A. Stability Derivatives of Longitudinal Dynamics of FOXTROT Aircraft

The stability derivatives of longitudinal dynamics of FOXTROT aircraft is shown in Table I. By using stability derivatives the transfer function for a particular flight condition can be determined.

B. Transfer Functions for Flight Condition-1

Using the stability derivatives of longitudinal dynamics of FOXTROT as shown in Table I and substituting the values in (6), the transfer function $G(s)$ between $\delta_E$ and $\alpha$ for flight condition-1 is given by

$$G(s) = \frac{2.0302s^2 + 102.8}{s^2 + 0.901s + 0.5633} = \frac{3.604s + 182.5}{s^2 + 1.7752s + 1.5798s + 1}$$
IV. CONVENTIONAL METHODS FOR PID CONTROLLER TUNING

A. Ziegler-Nichols (ZN) Method

There are two versions of ZN method. Out of which one version depends on the reaction curve and the other version depends on ultimate gain $K_u$ and the ultimate period $P_u$. Here, the 2nd version is considered. The values of $K_p$, $T_i$ and $T_d$ for 2nd version is shown in Table II.

<table>
<thead>
<tr>
<th>Stability Derivatives</th>
<th>Flight Conditions (FC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_0$ [ms$^{-1}$]</td>
<td>FC-1</td>
</tr>
<tr>
<td>$X_u$</td>
<td>-0.012</td>
</tr>
<tr>
<td>$X_u$</td>
<td>0.14</td>
</tr>
<tr>
<td>$Z_u$</td>
<td>-0.117</td>
</tr>
<tr>
<td>$Z_u$</td>
<td>-0.452</td>
</tr>
<tr>
<td>$Z_\nu$</td>
<td>-0.76</td>
</tr>
<tr>
<td>$M_u$</td>
<td>0.0024</td>
</tr>
<tr>
<td>$M_u$</td>
<td>-0.006</td>
</tr>
<tr>
<td>$M_u$</td>
<td>-0.002</td>
</tr>
<tr>
<td>$X_{dc}$</td>
<td>1.83</td>
</tr>
<tr>
<td>$Z_{dc}$</td>
<td>-2.03</td>
</tr>
<tr>
<td>$M_{dc}$</td>
<td>-1.46</td>
</tr>
</tbody>
</table>

B. Tyreus-Luyben (TL) Method

A ZN classical tuning constants are aggressive and oscillate. This results in a controller not very robust to model imprecision. Therefore, TL tuning constants are used for minor oscillations and robustness. The values of $K_p$, $T_i$ and $T_d$ for TL is shown in Table III.

<table>
<thead>
<tr>
<th>PID Type</th>
<th>$K_p$</th>
<th>$T_i$</th>
<th>$T_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>$0.45k_u$</td>
<td>$\frac{P_u}{1.2}$</td>
<td>0</td>
</tr>
<tr>
<td>PID</td>
<td>$0.6k_u$</td>
<td>$\frac{P_u}{2}$</td>
<td>$\frac{P_u}{8}$</td>
</tr>
</tbody>
</table>

C. Interpolation Rule

Let the controller be of the form

$$C(s) = K_c + \frac{K_i}{s} + K_ds$$

where $K_c$, $K_i$ and $K_d$ are proportional, integral and derivative constants. The tuning parameter for an under damped second order process is as:

$$K_c = \max \{A, X\}, \quad X = B \; \text{for} \; \zeta \geq 1$$

$$K_i = \max \{A, X\}, \quad X = B \; \text{for} \; \zeta \geq 1$$

$$K_d = \text{Either A, B or C}$$

where $A$, $B$, $B$ and $C$ for proportional, integral and derivative controllers are given in Table IV.

<p>| Table IV: Values of $A$, $B$, $B$ and $C$ for Calculation of $K_c$, $K_i$ and $K_d$ |</p>
<table>
<thead>
<tr>
<th>Calculation of $K_c$</th>
<th>Calculation of $K_i$</th>
<th>Calculation of $K_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$\frac{2\zeta}{k'(\tau \theta + \theta)\tau_0}$</td>
<td>$\frac{1}{k'(\tau \theta + \theta)}$</td>
</tr>
<tr>
<td>$B$</td>
<td>$\frac{1+4(\tau \theta + \theta) + (\zeta + \sqrt{\zeta^2 - 1})}{\tau_0} \frac{1}{k'(\tau \theta + \theta)}$</td>
<td></td>
</tr>
<tr>
<td>$C$</td>
<td>$\frac{1}{2k'(\tau \theta + \theta)}$</td>
<td>$\frac{1}{16k'(\tau \theta + \theta)}$</td>
</tr>
</tbody>
</table>

$k = \frac{k}{\tau_0}$, where $k$ is the gain $\tau_0 = \frac{1}{\omega_n}$ where $\omega_n$ = Natural frequency of oscillation; $\zeta$ = Damping ratio; $\tau_c$ = The controller tuning parameter; $\theta = \tau_0(1.5 + 0.5\zeta)(0.6)^\theta$ = The delay angle; $a = \tau_0^2$; $B$ is obtained by setting $\sqrt{\zeta^2 - 1} = 0$ in $B$.

D. Simulation Result

Time domain simulations of controller output ‘u’ and process output ‘y’ for ZN, TL and IR Techniques for the processes result from Flight Condition-1 of FOXTROT aircraft with setpoint and disturbances are shown in Figs. 3 and 4.
one represents a solution of the problem which performance is evaluated by a fitness function.

In this chapter, a PID controller is evolved by using a fitness function that quantifies closed-loop performance and is evaluated by repeated simulations. Each chromosome represents a point in the search space of the genetic algorithm. The term “phenotype” from biology to refer to the whole structure of the controller that is to be evolved. Hence, in this case the phenotype is given by

$$K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{e(t)}{dt}$$

where $e = \delta_k - \alpha$ = the error input to the PID controller; $\delta_k$ = The reference input; $\alpha$ = The output of the plant

A. The Population of Individuals

A population is a set of candidate solutions (chromosomes). A notation is used for representing a whole set of individuals i.e. a population. Let $k$ denote the generation number and $\theta^j(k)$ be a single parameter at time $k$. The superscript $j$ refers to the $j^{th}$ chromosome. A trait in a biological system is most often thought of as a property of the phenotype, like hair or eye color, whiles in our systems i.e. in the PID controller the genes, which are digits of parameters, are “expressed” as traits of the phenotype. Hence, the strings of genes are expressed as traits in the phenotype. Again the subscript i on $\theta^i(k)$ refers to the $i^{th}$ trait on the $j^{th}$ chromosome. Suppose the chromosome $j$ is composed of $p$ of these parameters (traits).

Let $\theta^j(k) = [\theta^1(k), \theta^2(k), ... , \theta^p(k)]^T$ be the $j^{th}$ chromosome. The population of individuals at time $k$ is given by $P(k) = \{\theta^j(k) : j = 1, 2, ..., S \}$ where $S$= the number of individuals in the population.

B. Genetic Algorithm operators

Basically, GA consists of three main operators. These are Selection, Crossover, and Mutation. The application of these three basic operators allows the creation of new individuals which may be better than their parents. This algorithm is repeated for many generations and finally stops when reaching individuals that represent the optimum solution to the problem.

1. Selection

The selection operator selects chromosomes from the current generation to be parents for the next generation. In this method, a few good chromosomes are used for creating new offspring in every iteration. Then some bad chromosomes are removed and the new offspring is placed in their places. The rest of population migrates to the next generation without going through selection process.

There are many ways to perform selection, but by far the most common one used in practice is fitness-proportionate selection.
1.1 Fitness Proportionate Selection

In this case, an individual is selected (the $i$th chromosome) for mating by letting each $m'(k)$ be equal to $\theta'(k) \in P(k)$ with probability

$$p_i = \frac{\theta'(k)}{\sum_j \theta'(k)} \quad (7)$$

Here the analogy of spinning a unit circumference roulette wheel is used where the wheel is divided like a pie into $S$ regions where the $r$th region is associated with the $r$th individual of $P(k)$. Each pie-shaped region has a portion of the circumference that is given by $p_i$ in (7). You spin the wheel, and if the pointer points at region $i$ when the wheel stops, then you place $\theta_i$ into the mating pool $M(k)$. You spin the wheel $S$ times so that $S$ elements end up in the mating pool and the population size stays constant. Clearly, individuals who are more fit will end up with more copies in the mating pool. Hence, chromosomes with larger-than-average fitness will embody a greater portion of the next generation. At the same time, due to the probabilistic nature of the selection process, it is possible that some relatively unfit individuals may end up in the mating pool $M(k)$.

2. Reproduction Phase, Crossover

The crossover/reproduction operator computes two offspring for each parent pair given from the selection operator. The crossover operator is used to create new solutions from the existing solutions available in the mating pool after applying selection operator. This operator exchanges the gene information between the solutions in the mating pool. The most popular crossover selects any two strings randomly from the mating pool and some portion of the strings is exchanged between the strings. The selection point is selected randomly. A probability of crossover is also introduced in order to give freedom to an individual solution string to determine whether the solution would go for crossover or not.

Crossover is the process of combining (mixing) chromosomes. The crossover operation operates on the mating pool $M(k)$ by “mating” different individuals. First, the crossover probability $p_c$ (usually chosen to be near 1 since, when mating occurs in biological systems, genetic material is certainly swapped between the parents) is specified. Here the crossover probability is 0.8. There are many types of crossover (i.e., ways to swap genetic material on chromosomes), but the simplest one is “single-point” crossover.

2.1 Single-Point Crossover

The procedural steps for single-point crossover are:

Step1. Randomly pair off the individuals in the mating pool $M(k)$. There are many ways to do this. For instance, pick up each individual from the mating pool and then randomly select a different individual for it to mate or mate all the individuals with the ones that are right next to each other. In this approach, if there are an odd number of individuals in $M(k)$, then you could simply take the last individual and pair it off with another individual who has already been paired off or you could pair it off with the individual with the highest fitness.

Step2. Consider the chromosome pair $\theta'$, $\theta$ that was formed in Step-1. Generate a random number $r \in [0, 1]$. If $r < p_c$, then cross over $\theta'$ and $\theta$. To cross over these chromosomes, select a “cross site” at random and exchange all the digits to the right of the cross site of one string with those of the other. This process is shown in Fig. 5. In this example, the cross site is position 5 on the string and hence, swap the last eight digits between the two strings. Clearly, the cross site is a random number that is greater than or equal to 1, and less than or equal to the number of digits in the string minus 1.

![Cross site](image)

Fig. 5 Crossover operation

(b) If $r > p_c$, then not cross over and hence do not modify the strings and go to the mutation operation.

Step3. Repeat Step-2 for each pair of strings in the mating pool $M(k)$.

3. Reproduction Phase, Mutation

Mutations are global searches. A probability of mutation is again predetermined before the algorithm is started which is applied to each individual bit of each offspring chromosome to determine if it is to be inverted. Mutation changes the structure of the string by changing the value of a bit chosen at random. Mutation is the occasional introduction of new features in to the solution strings of the population pool to maintain diversity in the population. Mutation operator changes a 1 to 0 or vice-versa, with a mutation probability. Here the mutation probability is of 0.08. The mutation probability is generally kept low for steady convergence.

Like crossover, mutation modifies the mating pool. The operation of mutation is normally performed on the elements in the mating pool after crossover has been performed. The biological analog of our mutation operation is the random mutation of genetic material. Again, there are many ways to perform mutations. The most common methods of mutation are described below.

3.1 Gene Mutations

To perform mutation in the computer, first choose a mutation probability $p_m$. With probability $p_m$, change (mutate) each gene location on each chromosome randomly to a member of the number system being used.
For example, in a base-2 genetic algorithm, we can mutate
\[
1010111 \rightarrow 1011111
\]
where the fourth bit was mutated to one. For a base-10 number system, you would simply pick a number at random to replace a digit, if you are going to mutate a digit location. Usually, the mutation probability is chosen to be quite small (e.g., less than 0.01), since this will help guarantee that all the individuals in the mating pool are not mutated, so that any search progress that was made is lost i.e. it is kept relatively low to avoid degradation to exhaustive search via a random walk in the search space).

C. Computational Procedure of GA

The computational procedure for GA is stated below:

Step 1. Define the GA parameters (e.g. crossover and mutation probability, population size, termination parameters).

Step 2. Define the initial population \( P(0) \).

Step 3. For \( k = 1 \) to \( N_{\text{ga}} \) (main loop for producing generations). Compute the fitness function for each individual.

- Selection: From \( P(k) \), form \( M(k) \), the mating pool at iteration \( k \), using fitness-proportionate selection.

- Reproduction: For each individual in \( M(k) \), select another individual in \( M(k) \), mate the two via crossover with probability \( p_c \), and mutate each gene position with probability \( p_m \). Take the \( S \) children produced by this process, and put the children in \( P(k + 1) \), the next generation.

Step 4. Next \( k \) (i.e. return to step 3).

Step 5. Provide results.

D. Flow Chart of GA

The flow chart of Genetic Algorithm is shown in Fig. 6.

E. Block diagram of GA Based PID Controller

In this work, a genetic PID controller is designed to control the angle of attack of FOXTROT aircraft. The tuning of PID controller is governed by system nonlinearities and continuous parameter variations. The PID controller parameters \( K_p, K_i \) and \( K_d \) can be tuned with automatic tuning rules rather than conventional tuning rules such as Ziegler-Nichols, Cohen-Coon, Tyerus-Luyben etc. for better performances in many applications. Considerable revival of PID controller in the last decade is due to the introduction of automatic tuning and new approaches to automatically optimizing the performance of a dynamic system. A control problem is likely to have a unique optimum solution for the problem if it is specified by strict constraints. On the other hand, if some system specifications are left free to vary in a certain range several solutions might be possible for a control problem.

The use of GA tuning method as a powerful tool in the controller parameter design is implemented here as single objective function optimization. Most realistic optimization problems, particularly those in design require the simultaneous optimization of more than one objective function. Fig. 7 shows the block diagram of a GA based PID controller for controlling the angle of attack of an aircraft. Here the controlling parameters are \( K_p, K_i, \) and \( K_d \).

F. Simulation Results

The Matlab 7.1 is used for simulation work. In this simulation, the value of \( K_p, K_i \), and \( K_d \) is calculated for 100 generations and the values for IAE are shown in Fig. 8. Similarly, the values of \( K_p, K_i \), and \( K_d \) for MSE and for IATE are calculated 100 generations and shown in Figs. 9 and 10.
where the control error, 

\[ \text{IAE} = \int_0^\infty |e(t)| dt, \quad \text{MSE} = \frac{1}{T} \int_0^T e^2(t) dt, \quad \text{IATE} = \int_0^\infty t |e(t)| dt \]

with the control error, \( e = y - \hat{y} \).

The performance indices of various soft computing techniques like GA, FMRLC and RBFNC are compared with conventional techniques like ZN, TL and IR and the results are given in Table V.

### Table V: Performance Indices of a Dynamic System

<table>
<thead>
<tr>
<th>Techniques</th>
<th>MSE</th>
<th>IAE</th>
<th>IATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZN</td>
<td>0.1311</td>
<td>57.3971</td>
<td>63.0637</td>
</tr>
<tr>
<td>TL</td>
<td>0.1256</td>
<td>53.4471</td>
<td>50.6712</td>
</tr>
<tr>
<td>IR</td>
<td>0.0570</td>
<td>27.3914</td>
<td>1.9471</td>
</tr>
<tr>
<td>GA</td>
<td>0.0015</td>
<td>0.1585</td>
<td>0.0190</td>
</tr>
</tbody>
</table>

It is clear from Table V that the performance indices for using GA is improved i.e. it is very less as compared to the conventional methods like ZN, TL and IR.

### REFERENCES


[18] Kelvin M. Passino, “Biomimicry for optimization, control and automation”, Department of Electrical and Computer Engineering, 416 Dreese Laboratories, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210, USA.

Mr. S. Swain was born on 21st May 1970 in Odisha, India. He completed his ME degree in Power Systm Engg. in 2001 from VSSUT, Burla, Odisha India. Now he is continuing his Ph.D degree in BPUT, Rourkela, Odisha, India. His major field of study is Control System analysis and design.
Mr. Swain is a life member in ISTE. He is also a senior member in Institution of Engineers, India.

Dr. P. S. Khuntia was born on 06th Oct. 1969 in Odisha, India. He completed his ME degree in Automatic Control System and Robotics from MS University, Varodara in 1999. He has completed his Ph.D degree from ISM Dhanbad in 2011, India. His major field of study is Intelligent Control, Digital Signal Processing and Soft Computing.

He is now working as Professor in GMRIT, Rajam, Srikakulam, Andhra Pradesh. He has published many papers in National and International Journals. His current research interest is Advanced Control Theory and its application.

Dr. Khuntia is a life member in ISTE. He is also a senior member in Institution of Engineers, India.