Optimal Fuzzy Logic Tuning of Dynamic Inversion Flight Controller

Alireza Abaspour¹, Mohammad Sadeghi² and Sayed Hosein Sadati³

Aerospace Department, Maleke_ ashtar University, Tehran, Iran.
²,³ Aerospace Department, Maleke_ ashtar university, Tehran, Iran.

Abstract

In this paper an Optimal fuzzy logic control (OFLC) law for a nonlinear control method is introduced. Dynamic inversion is a nonlinear control synthesis technique in which the inherent dynamics of a dynamical system are canceled out and replaced by desired dynamics, selected by the designer. The output of such an inner-loop controller is the control input, which produces the desired closed-loop response. The accuracy of control response in dynamic inversion method is depend on two gain which are locate before the outer loop and inner loop. This paper attempts to find a solution to tune these gains for a better control performance in an intelligent way. We used Particle Swarm Optimization (PSO) to optimize the membership functions’ (MFs) parameters of the proposed design. The distribution of the MFs is obtained by minimizing a nonlinear constrained multi-objective optimization problem where control errors are treated as competing objectives. The performance of the introduced control law is compared with classical dynamic inversion. The simulation results show that introduced method performs better than classical dynamic inversion. Moreover, the simulations show that the fuzzy pso dynamic inversion method that we proposed is robust on uncertainties of model parameters such as mass.

Keywords: Fuzzy logic, nonlinear model, dynamic inversion, control effort, Particle swarm optimization, uncertainties.

1. Introduction

Most airplanes must operate over a broad flight envelope, Which the aerodynamic characteristics within this envelope vary widely. A controller based on classical control
methodologies [1,2] such as root locus design, would require synthesis of many different designs at selected operating points along the flight path. These classical methodologies can be time consuming, thereby increasing both design time and cost during control system development. Additionally, gains need to be scheduled across the flight envelope. In contrast, dynamic inversion is a candidate methodology that seeks to eliminate gain scheduling through the inversion and cancellation of the inherent dynamics, by replacement with a set of user-selected desired dynamics. The need to gain schedule is thus reduced through the use of a high-fidelity, onboard aircraft model. For these reasons dynamic inversion is a promising candidate control design methodology aircraft with extensive flight envelopes.

Within the last decade dynamic inversion has become a popular methodology for aircraft flight controller design. Much of the literature has applied this design methodology to both the longitudinal and lateral/directional axes of high-performance aircraft, such as the F-117A[3] the F-18HARV [4] and other modified versions of the F-18[5] and the F-16 [6]. The control performance of Dynamic Inversion (DI) method is very sensitive to outer loop and inner loop gains. In this paper we used fuzzy logic optimal control (OFLC) to tune the outer loop gain in an intelligent way to obtain a better control performance. FLCS are developed to utilize human expert knowledge in controlling various systems. It is well known that while fuzzy rules are relatively easy to derive from human experts, the fuzzy MFs are difficult to obtain. Tuning of MFs is a time consuming and often frustrating exercise. To overcome these difficulties various techniques have been reported to automate the tuning process of MFs. An adaptive network based fuzzy inference system (ANFIS) was introduced [7], where an adaptive neural network was used to learn the mapping between the inputs and outputs and a Sugeno-type of fuzzy system could be generated based on the neural network. A quantum neural network was also used to learn the data space of a Tagaki-Sugeno FLC [8]. Genetic algorithm has been used in the automatic design of FLCS [9,10] in the areas of mobile robotics. In the current work, we introduce a fuzzy control law with rules obtained based upon the human experienced. Then, the MFs’ parameters of the introduced design will be optimized using PSO technique. In fact, PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy [11,12]. Based on their description, particle swarm optimization imitates human (or insects) social behavior. Individuals interact with one another while learning from their own experience, and gradually the population members move into better regions of the problem space. The swarm of PSO can be envisioned as multiple birds (particles) that search for the best food source (optimum) by using their inertia, their knowledge, and the knowledge of the swarm.

Single particles behave similarly because traditionally they share the same configuration. While searching for food, the birds are either scattered or go together before they locate the place where they can find the food. While the birds are searching for food from one place to another, there is always a bird that can smell the food very well and having the better food resource information, that is the bird is perceptible of the place where the food can be found, in Figure 1 you can see a flock of swarms using PSO.
Since they are transmitting the information, especially the good information at any time while searching the food from one place to another, conducted by the good information, the birds will eventually flock to the place where food can be found. Due to its simplicity in implementation, PSO has gained popularity in engineering applications, such as in image processing [13] and in system modeling [14]. A number of publications have also been reported in using PSO to automatically tune the FLC parameters [15,16,17,18]. These publications are focused on tuning the parameters involved in the TS-type fuzzy controllers. In general, the PSO is used to perform the learning tasks that are usually associated with the NN in the TS FLCs. Although there are research results in the area of automatic fuzzy MFs optimizing, most of them are in the area of TS type of fuzzy controllers. To the best of our knowledge, there is no report on using PSO for the Mamdani-type of fuzzy controller MFs tuning.

In this paper, we use PSO to automatically tune MFs of the proposed control law. The OFLCDI then is compared with classic DI. The paper is organized as follows:

in section II we proceed with a brief overview of mathematical model of aircraft, whereas dynamic inversion method explained in section III, and combination of fuzzy and pso explained in section IV. Then in section V we proceed with the numerical simulation of classic DI and OFLCDI, while the conclusions are provided in section VI.

2. Mathematical model of aircraft
In this paper we design a flight controller for a phantom fighter jet (known as F_4), the Aerodynamic coefficient obtained from [19]. In table 1 you see the specification of the airplane that we used. The inputs are delta aileron, delta elevator and delta rudder \([\delta_a, \delta_e, \delta_r]\). task of the controller is to track the commands of \(\alpha, \beta\) and \(\mu\) when aerodynamic model uncertainties exist. The body-fixed axes, nonlinear equations of motion for an aircraft over a flat Earth are given by [19].

\[
\dot{p} = \frac{I_x I_z - I_y^2}{I_z} \dot{\omega} + \frac{I_y (I_x - I_z) - I_w^2}{I_z} \dot{\tau}
\]  

(1)

\[
\dot{q} = \frac{1}{I_y} \{m_{\text{acc}} + pr (I_x - I_z) + I_w (r^2 - p^2)\}
\]  

(2)
\[
\dot{\mathbf{x}} = \begin{bmatrix}
(L_x - I_y + I_z)I_{xw} - r + I_z(I_y - I_z) + I_{z2}\nI_yI_z - I_{z2}\n\end{bmatrix} q + \\
\frac{I_{w1}I_{w2} + I_{w2}I_{w1}}{I_yI_z - I_{z2}} p
\]
(3)

\[\dot{\beta} = p \sin \alpha - r \cos \alpha + \frac{1}{m_v} [m, g \sin \gamma \sin \mu] + \]
(4)

\[\dot{\alpha} = - (p \cos \alpha + r \sin \alpha) \tan \beta + \]
(5)

\[\dot{\mu} = \frac{1}{\cos \beta} (p \cos \alpha + r \sin \alpha) - \]
(6)

\[\dot{Y} = \frac{1}{m_v} [L \cos \mu - mg \cos \gamma - Y \sin \mu \cos \beta] + \]
(7)

\[\dot{V} = \frac{1}{m} [-D + Y \sin \beta - mg \sin \gamma + T \cos \beta \cos \alpha] \]
(8)

In above equations: F is aerodynamic force about the body-fixed frame, I is moment of inertia, L, M, N are aerodynamic rolling, pitching, yawing moment, \(p,q,r\) are roll, pitch, yaw rate about the body-fixed frame, \(\bar{q}\) is dynamic pressure, T is thrust, \(V\) is velocity, \(\alpha, \beta, \mu\) is angle of attack, sideslip angle, bank angle, and \(\gamma\) is flight path angle. It is assumed that the aerodynamic forces and moments are expressed as functions of angle of attack, sideslip angle, bank angle, angular rates, and control surface deflection.

3. Dynamic Inversion
In this section, we present a feedback linearization technique known as Dynamic Inversion. Dynamic inversion is a control design methodology that uses a feedback signal to cancel inherent dynamics and simultaneously achieve a specified desired dynamic response [20].

Consider a general time-invariant nonlinear system modeled by the ordinary differential equations.

\[
\dot{x} = f(x,u,t) \\
y = H(x,u)
\]
Optimal Fuzzy Logic Tuning of Dynamic Inversion Flight Controller

There are two separate differential equations: a set of slow dynamics \( \dot{x} \) and a set of fast dynamics \( \dot{y} \).

\[
\begin{align*}
\dot{x} &= f(x) + g(x)y \\
\dot{y} &= h(x, y) + k(x, y)u
\end{align*}
\] (10)

**Figure 2**: Two-timescale inversion of slow dynamics.

**TABLE 1**: Phantom specification and aerodynamic coefficient in cruise flight[9]

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Value</th>
<th>unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wing Span</td>
<td>38.7</td>
<td>Ft</td>
<td>B</td>
</tr>
<tr>
<td>Wing Area</td>
<td>530</td>
<td>ft²</td>
<td>S</td>
</tr>
<tr>
<td>Chord Length</td>
<td>1.6</td>
<td>ft</td>
<td>c</td>
</tr>
<tr>
<td>Xcg</td>
<td>0.29</td>
<td>ft</td>
<td>Xcg</td>
</tr>
<tr>
<td>Weight</td>
<td>39000</td>
<td>lbs</td>
<td>m</td>
</tr>
<tr>
<td>X axis Inertia moment</td>
<td>25000</td>
<td>lbs. ft²</td>
<td>Iₓₓ</td>
</tr>
<tr>
<td>Y axis Inertia moment</td>
<td>122200</td>
<td>lbs. ft²</td>
<td>Iᵧᵧ</td>
</tr>
<tr>
<td>Z axis Inertia moment</td>
<td>139800</td>
<td>lbs. ft²</td>
<td>Izz</td>
</tr>
<tr>
<td>XZ palneinenriatia moment</td>
<td>2200</td>
<td>lbs. ft²</td>
<td>Iₓz</td>
</tr>
</tbody>
</table>

If the system is affine in the controls, then solving explicitly for the control vector yields.

\[
y = g(x)^{-1}[\dot{x} - f(x)]
\] (11)

Replacement of the inherent dynamics with the desired dynamics results in the control that will produce the desired dynamics.

\[
y = g(x)^{-1}[\dot{x}_d - f(x)]
\] (12)
Substituting for the linear aircraft dynamics represented as \( \dot{x} = Ax + Bu \) yields a set of slow dynamic equations for the rotational variables \( x_i = [\beta, \alpha, \mu] \) and a set of fast dynamic equations for the rotational rate variables \( x_z = [p, q, r] \). The rate variables now form the input for the slow dynamics, while the actual control surface commands form the inputs for the rate dynamics. Inverting the slow and fast differential equations yields the two dynamic-inversion control laws for the outer dynamic inversion loop and inner dynamic inversion loop, whose block diagram is shown in Figure 2.

The structure of the two-timescale controller is shown in Figure 3. In each feedback loop, control laws \( x^d \) and \( u \) are designed separately.

![FIGURE 3: Structure of the two-timescale controller](image)

### 3.1 Inner loop control for the fast variables

When designing a flight control system with the two-timescale assumption, the inner-loop controller is designed to control the fast states \( x_z \) using the control input \( u \), where the desired values of the fast states \( x^d \) are given by the outer loop. Now for using dynamic inversion based on (1-8), for the fast differential equation we have:

\[
\begin{bmatrix}
\dot{p} \\
\dot{q} \\
\dot{r}
\end{bmatrix} =
\begin{bmatrix}
f_p(\bar{x}) \\
f_q(\bar{x}) \\
f_r(\bar{x})
\end{bmatrix} + g(x)
\begin{bmatrix}
\delta_p \\
\delta_q \\
\delta_r
\end{bmatrix}
\]

(13)

With considering (13) the controller of inner loop yields:

\[
\begin{bmatrix}
\delta_p \\
\delta_q \\
\delta_r
\end{bmatrix} = g^{-1}(\bar{x})
\begin{bmatrix}
\dot{p}_d \\
\dot{q}_d \\
\dot{r}_d
\end{bmatrix}
\]

(14)

While the desired angular rates defined with following equation:

\[
\begin{bmatrix}
\dot{p}_d \\
\dot{q}_d \\
\dot{r}_d
\end{bmatrix} =
\begin{bmatrix}
\omega_p & 0 & 0 \\
0 & \omega_q & 0 \\
0 & 0 & \omega_r
\end{bmatrix}
\begin{bmatrix}
p - p_d \\
q - q_d \\
r - r_d
\end{bmatrix}
\]

(15)
Where $\omega_p, \omega_q$ and $\omega_r$ that shown in figure1 by $k_z$, are inner loop control gains which chosen by designer to obtain desire performance, and the subscript $c$ denotes the commands.

3.2 Outer loop control for the slow variables

In the outer loop, the controller is designed to control the slow states $x_1$, and the output of outer loop used as inner loop commands. The inner-loop controller neglects the transient responses of the fast states $x_2$. It assumes that the fast states track their commanded values instantaneously and that the control surface deflection has no effect on the outer-loop dynamics.

For using dynamic inversion based on (1-8) for the slow variables we have:

$$\begin{bmatrix}
\dot{\beta} \\
\dot{\alpha} \\
\dot{\mu}
\end{bmatrix} = \begin{bmatrix}
f_p(\bar{x}) \\
f_q(\bar{x}) \\
f_r(\bar{x})
\end{bmatrix} + g_{c1}(\bar{x}_c) \begin{bmatrix}
p \\
q \\
r
\end{bmatrix} + g_{c2}(\bar{x}_c)\bar{\mu}$$

(16)

Computing the relation between $p_c, q_c, r_c$ and main control surfaces is difficult, so we neglect the small term of $g_{c2}(\bar{x}_c)$. With considering (16) the controller of outer loop yields.

$$\begin{bmatrix}
\dot{\beta} \\
\dot{\alpha} \\
\dot{\mu}
\end{bmatrix} \approx \begin{bmatrix}
f_p(\bar{x}) \\
f_q(\bar{x}) \\
f_r(\bar{x})
\end{bmatrix} + g_{c1}(\bar{x}_c) \begin{bmatrix}
p \\
q \\
r
\end{bmatrix}$$

(17)

While the desire angular variables defined with following equation.

$$\begin{bmatrix}
\beta_d \\
\alpha_d \\
\mu_d
\end{bmatrix} = \begin{bmatrix}
\omega_p & 0 & 0 \\
0 & \omega_u & 0 \\
0 & 0 & \omega_\mu
\end{bmatrix} \begin{bmatrix}
\beta - \beta \\
\alpha - \alpha \\
\mu - \mu
\end{bmatrix}$$

(18)

Where $\omega_p, \omega_u$ and $\omega_\mu$ that shown in figure1 by $k_i$, are outer loop control gains which chosen by designer to obtain desire performance and $\beta_c, \alpha_c, \mu_c$ are pilot's commands.

By using (17), the output of outer loop derived as following equation.

$$\begin{bmatrix}
p_c \\
q_c \\
r_c
\end{bmatrix} = g^{-1}_{c1}(\bar{x}_c) \begin{bmatrix}
\dot{\beta}_d \\
\dot{\alpha}_d \\
\dot{\mu}_d
\end{bmatrix} - \begin{bmatrix}
f_p(\bar{x}) \\
f_q(\bar{x}) \\
f_r(\bar{x})
\end{bmatrix}$$

(19)
4. Fuzzy PSO control
Each FLC has the structure as shown in the following:

![Figure 4: The structure of FLC.](image)

The important components of a FLC are the Fuzzifier, the Inference engine, the Fuzzy Knowledge base, and the Defuzzifier. According to Figure 4, the Fuzzifier converts the crisp input to a linguistic variable using the MFs stored in the fuzzy knowledge base. By using If-Then type fuzzy rules the Inference engine converts the fuzzy input to a fuzzy output. The Defuzzifier converts the fuzzy output of the inference engine to crisp one. In our design the centre of area (CoA) method, which supplies defuzzified output with better continuity is used for defuzzification. In general, CoA method with the output is calculated as:

$$u^* = \frac{\int um_o(u)du}{\int m_o(u)du}$$  \hspace{1cm} (20)

Where $u$ is the output variable, $o$ is the output fuzzy set and $m_o$ is the MFs of the output fuzzy set. Minimum Mamdani (AND method), the most popular inference engine, is chosen to obtain the best possible conclusion. This type of inference engine allows easy and effective computation and it is appropriate for the real time control application [21].

The starting point of a fuzzy controller design is to choose the number and the shape of the MFs for input and output variables. Our fuzzy controller is similar to PD controller. It has two inputs ($e, \dot{e}$) and single output ($o$). So that, three groups of MFs are chosen for the three corresponding variables and. Each group has 7 triangular MFs as shown in Figure 5. It has been found that using complex forms of MFs cannot bring any advantage over the triangular ones, where this kind of MFs gives faster response [22].

In fact, appropriate number and shapes of membership functions are usually result of different compromises among contradicting factors, such as accuracy, hardware and computation complexities. The ones, we are mostly concerned in this work, are relative accuracy, computation time, and complexity.
Before defining the rules we normalize the input and output data of the controller, the normalization procedures are required to transform performance ratings with different data measurement units into a decision matrix with compatible unit. So that, the variables, $e$, $\dot{e}$ and $C$ are quantized and normalized within $[-1, 1]$, according to the following Equation:

$$X_{\text{norm}} = \frac{X}{X_{\text{max}}}$$  \hspace{1cm} (21)

Where, the variable has to be normalized to its maximum value. The maximum values of the controller’s variables can be obtained from the engineering experience of designer and simulation results. As the absolute maximum value of $e$ and $\dot{e}$ are always between 0 and 1(radian), there is no need to normalize and denormalize them, but for $C$ we use scaling factor of 20 to denormalize it. Each normalized variable then is replaced by a set of linguistic values as shown in Table 2.

<table>
<thead>
<tr>
<th>$e$, $\dot{e}$, $C$</th>
<th>LN</th>
<th>MN</th>
<th>SN</th>
<th>ZE</th>
<th>SP</th>
<th>MP</th>
<th>BP</th>
</tr>
</thead>
</table>

Where, the linguistic values {LN, MN – LP} are abbreviations of {Large Negative, Medium Negative… Large Positive} respectively as shown in Figure 3. Now, we can obtain the rules based upon this paper conception.

### 4.1 Defining the rule

While model is running, we have two parameter to control, the error of controller which showed by $e$ and its variation ratio showed by $\dot{e}$. As you can see in table 3, because our controller is similar to PD controller, the relation between $e$ and $\dot{e}$ defined by a sum assumption.
TABLE 3: The entire Rules of FLCDI.

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>NM</th>
<th>NS</th>
<th>ZE</th>
<th>PS</th>
<th>PM</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>NB</td>
<td>NB</td>
<td>NM</td>
<td>NM</td>
<td>NS</td>
<td>NS</td>
<td>ZE</td>
</tr>
<tr>
<td>NM</td>
<td>NB</td>
<td>NM</td>
<td>NM</td>
<td>NS</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
</tr>
<tr>
<td>NS</td>
<td>NM</td>
<td>NM</td>
<td>NS</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
</tr>
<tr>
<td>ZE</td>
<td>NM</td>
<td>NS</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
<td>PM</td>
</tr>
<tr>
<td>PS</td>
<td>NS</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
<td>PM</td>
<td>PM</td>
</tr>
<tr>
<td>PM</td>
<td>NS</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
<td>PM</td>
<td>PM</td>
<td>PB</td>
</tr>
<tr>
<td>PB</td>
<td>ZE</td>
<td>PS</td>
<td>PS</td>
<td>PM</td>
<td>PM</td>
<td>PB</td>
<td>PB</td>
</tr>
</tbody>
</table>

4.2 PSO

The PSO is a population based stochastic optimization technique, consists of a swarm of particles flying through the search space. Every individual in the swarm contains parameters for position and velocity. The position of each particle represents a potential solution to the optimization problem. The dynamic of the swarm is governed by a set of rules that modify the velocity of each particle according to the experience of the particle and its neighbors depending on the social network structure within the swarm. By adding a velocity to the current position, the position of each particle is modified. As the particles move around the space, different fitness values are given to the particles at different locations according to how the current positions of particles satisfy the objective. At the iteration each particle keeps track of its personal best position. Depending on the social network structure of the swarm, the global best position, and/or the local best position, is used to influence the swarm dynamic. After a number of iterations, the particles will eventually cluster around the area where fittest solutions are. The swarm behavior is influenced by the number of particles, the neighborhood size, the inertia weight, the maximum velocity, and the acceleration calculation that modifies the velocity. The larger the number of particles in the swarm, the more likely the swarm will converge on the global optimum, because the social information exchange is increased. The influence of the current velocity on the new velocity can be controlled by the inertia weight. The influence of the particle’s experience and that of its neighbor is governed by the acceleration calculation. The acceleration limits the trajectory of the particle oscillation. The new velocity is limited by the given maximum velocity to prevent particles from moving too fast in the space.

In particular, the velocity associated with each particle in PSO is calculated as:

\[
\begin{align*}
v_i(k+1) &= w.v_i(k) + c_1.r_1(k)(x_g - x_i(k)) \\
&\quad + c_2.r_2(k)(x^g_i - x_i(k))
\end{align*}
\]

Where \( w \) is the momentum or inertia weight of the particle, \( v_i(k) \) is the velocity of the particle \( i \) at time step \( k \), \( x_g \) is the global best performing particle up to time step \( k \)
in the entire population, $x^p_i$ is the best experience particle has had up to time step $k$, $x^i_k$ is the current location of particle $i$, and $c_1, c_2$ are constants usually equal each to other, $r_1, r_2$ are random numbers within $[0, 1]$ those represent random fiction[23]. To limit the searching space is limited to be within a certain range of $v_{\text{min}} \leq v_i \leq v_{\text{max}}$. The new location of particle can be calculated as:

$$x_i(k+1) = x_i(k) + v_i(k+1)$$

(23)

The evaluation of the particle performance is based on a problem specific objective function that decides the ‘closeness’ of the particle to the optimal solution. With Figure 6, the optimization process is started with random initial values then the object function is calculated. The first positions are automatically the best values. Based on (22 and 23) the PSO updates both velocity and position vectors. Object function is calculated again, if the new value of the object function is smaller than the old one, the corresponding position vector is replaced by the old one, else it remained. The process is re-run until a termination criterion, such as a limit on the number of iterations or satisfactory results, is reached.

**Figure 6:** Flowchart of PSO.
4.3 Optimization of the MFs
Each triangular MFs is determined by three parameters such as a, b and c, where a, c locate the "feet" of the triangle and b locates the peak (see Figure 5). Anyway the triangular MF has the form:

\[ f_{\text{tri}}(x,a,b,c) = \begin{cases} 
0, & x \leq a \\
\frac{x-a}{b-a}, & a \leq x \leq b \\
\frac{c-x}{c-b}, & b \leq x \leq c \\
0, & c \leq x
\end{cases} \]  \hspace{1cm} (24)

The corresponding parameters have to satisfy the inequality Since there are 14 MFs in the inputs and 7 MFs in the output, in addition, each MF has its three own parameters. In total we have 63 parameters for optimization, so that, a vector of dimension particles is adopted in the optimization process. The population is set P=50 as vectors, while the total searching iterations is set to be I=30. With the help of [23], the inertia weight \( w \) was set to be 0.9 decreased linearly to 0.4 and the weighting factors \( r_1, r_2 \) were set to be 0.5 for both, while \( r_1, r_2 \) are randomized within [0, 1]. Therefore, it is possible to use PSO as a global optimization search method to find a set of such parameters that will produce the best control performance of the FLC. Throughout the optimization process we try to minimize the miss distance, the object function is defined as:

\[ J = \int x^T Q x dt \]  \hspace{1cm} (25)

In (25) \( x \) is a vector which contain the parameter that should be optimized and \( Q \) is a diagonal matrix which denotes the limitation and importance of errors.

\[ \begin{bmatrix} E_{\text{out}} & E_{\text{obj}} & E_{\text{av}} \end{bmatrix} \]

\[ Q = 10^{\text{diag}}([6 /40 1/400]) \]  \hspace{1cm} (26)

\( Q \) is a parameters which chosen by designer according to system’s dynamics and the desired control behavior. For the optimization process, we assumed the following data:

One of the important factors in the simulation process is usually the integration time-step. This is normally chosen based on nature of the problem or experience. Here, we use a time step equal to 0.01 second, mainly because a typical airplane-gyro gyrates around 100 cycles per second.

5. Numerical Simulation
In this paper we used the combination of dynamic inversion control and fuzzy logic control and PSO as we described. To find the advantages and disadvantages of this
new method we should compare the result of this new method with classical dynamic inversion to a certain command. Since step input is a common input and clearly shows the performance of controller, we selected step input for our analysis and the following command values of $\alpha, \beta$ and $\mu$ are applied to the aircraft in a steady state level flight of $V=0.9$ Mach, $h=1000ft$:

**TABLE 4. Input commands.**

<table>
<thead>
<tr>
<th>Command(degree)</th>
<th>Time(s)</th>
<th>$0 \leq t &lt; 0.5$</th>
<th>$0.5 \leq t &lt; 5$</th>
<th>$t \geq 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td></td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>$\beta$</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\mu$</td>
<td></td>
<td>0</td>
<td>50</td>
<td>0</td>
</tr>
</tbody>
</table>

To make the step input similar as real inputs, we used a command filter to smooth the edges, you can see the filter structure in figure 7. As you can see in figure 7 the output of filter depends on two parameter which selected by designer, and we selected $\omega=2$, $\xi=1$ to make the input like real pilot commands.

**FIGURE 7:** Command filter structure.

For a better comparison between OFLDI and DI method we considered 30 percent uncertainties which is usually happens because of fuel consumption. It is assumed the center of gravity(c.g) is fixed in mass uncertainties. The MFs of the inputs and the output were tuned by PSO algorithm and shown in figure 8.

**FIGURE 8:** Tuned membership functions.
In Figure 9 you can see the comparison of tracking command of $\alpha$, $\beta$ and $\mu$ between OFLDI and classic dynamic inversion in presence of mass uncertainties and without it, this figure denote that there is no clear difference in command tracking between these methods. But when we calculate the tracking cost ($\int e^2 dt$) in Figure 12, it’s clearly showed that OFLDI has quite less tracking cost in presence of mass uncertainties.

Control effort is important factor in designing controller, which is in direct relation with control deflection ($\int u^2 dt$). So beside the saturation limits caused by limitation of actuators, control deflection is a really important factor in designing controller. According to Figure 10 OFLDI method comparing to DI method has less control deflection, more over it has better robustness facing mass uncertainties. In Figure 11 you can see the differences in control effort between these two control methods, from this figure it is clear that using OFLDI pay less control effort and it is lead to saving more energy.
FIGURE 10: Comparison of aileron deflection with uncertainties and without it.
6. Conclusions
A controller for a six-degree-of-freedom nonlinear flight model is proposed. The Dynamic Inversion (DI) controller is used to track the $\alpha$, $\beta$ and $\mu$ commands with the assumption that the aerodynamic characteristics are fully understood. The controller gains which affect the control performance directly tuned by OFL method, so we named this method OFLDI. In simulations 30 percent of mass uncertainties with assumption of fixed c.g considered. In numerical simulation it is shown that, the OFLDI controller in a same command tracking situation has better performance in presence of mass uncertainties, it has less control deflection, less control effort and less Tracking cost. So according to simulation result, it's demonstrated that OFLDI method performs better than DI method.

References
Optimal Fuzzy Logic Tuning of Dynamic Inversion Flight Controller


