Fuzzy Q Learning based UAV Autopilot

Rajneesh Sharma
Netaji Subhas Institute of Technology, New Delhi, India
Email: rajneesh496@gmail.com

***Abstract*— Navigation and control of an unmanned aerial vehicle (UAV) is a challenging problem and could be framed as a Reinforcement Learning (RL) task. Herein, we propose to use reinforcement learning for designing a UAV autopilot based on the Fuzzy Q Learning (FQL) approach. Proposed control scheme envisages an amalgamation of proportional (P) control that stabilizes the UAV and an action triggering Fuzzy Inference system (FIS) control that learns the correct control action to achieve the desired flight trajectory for a UAV flight. We test the proposed RL based UAV control for three cases: (i) Altitude control (ii) Trajectory Tracking, and (iii) Reconnaissance flight of a UAV. Results demonstrate the viability and effectiveness of a UAV autopilot designed using FQL.**

Keywords- Reinforcement Learning, FQL, UAV

I. INTRODUCTION

Our Objective is to demonstrate the viability of RL [1] in general and FQL [2] in particular, for autonomous navigation and control of small UAVs [3]. We propose an FQL based adaptive autopilot for UAVs that is conceptually simple yet computationally efficient and requires no model of the aircraft.

In modern times UAVs are increasingly being used to perform tasks considered dangerous/repetitive or both. A UAV offers reliable, low cost and safe alternative to manned aircrafts for tasks such as spying missions or patrolling potentially dangerous enemy terrains. They can also be used to locate and destroy enemy hideouts/terrorist camps (more relevant in modern times). Civilian applications include search and rescue, weather related information gathering or police patrolling. Their small size and low cost coupled with the fact that they are relative undetectable by radars make them an ideal choice for military applications. We propose to apply reinforcement learning for designing an autopilot for UAVs. The designed autopilot successfully learns to take-off, fly along a designated flight path and then land the UAV.

Reinforcement learning techniques mimic human learning in terms of being direct, simple to apply and adaptive in nature [4]. In a standard RL based setting one usually stores the *Q*-values [5] in a look-up table. However, this type of methodology becomes highly computationally cumbersome even for small problems or in some cases becomes computationally intractable, e.g., for a continuous state-space. Herein, we use FIS for generalizing over the state space and for speeding up learning or in other words to tackle the so called 'Curse of dimensionality'. Fuzzy logic [6] is a mathematical approach which copies humans in learning and thinking. Fuzzy logic finds use in generic function approximation and for generating continuous control actions. Glorennec and Jouffe [7] have advocated the use of a group of fuzzy rules which act as an agent that produce continuous actions. Fuzzy Q-learning (FQL) produces an action by firing some rules and cooperating.

In this paper, we propose to use FQL for autonomous navigation and control of a UAV. Section II describes the FQL approach, Section III elaborates on the proposed FQL based UAV control scheme, Section IV describes the simulation results and section V concludes the paper outlining future scope of work.

II. FUZZY Q LEARNING

A Reinforcement learning based controller interacts with the system it aims to control and learns a control policy by repeated interactions. The system state is $x^k \in X$ at time step *k, X* being the state space, the controller being allowed to take a finite set of decisions, $u(x^k) \in U(x^k)$ which decide the next state. The variable *k* represents time instants at each of which the controller has to make a decision or take action where $k = 0,1,2, \ldots..$.

Fuzzy Q-learning is a form of reinforcement learning method to tune FIS conclusions. FQL is an adaptation of Watkin's Q-learning [5] for FIS, wherein both the actions and *Q*-functions that store experiential information are inferred from fuzzy rules. FIS scheme depends on the number of variables in the state vector, *n* and the number of the fuzzy sets used to sense each variable which ultimately decide the number of fuzzy rules. There can be various types of membership functions, e.g., triangular, trapezoidal, gaussian etc, which type of membership function to use is highly problem dependent and sometimes context dependent. The total number of rules *N* in the FIS rule base is given by: $N = \prod_{s=1}^{n} N_L(s)$ where $N_L(s)$ = number of labels for variable *s*.

Use of rule based learning allows one to use the proposed scheme on continuous state and action apace problems. FQL has been used extensively as an RL method that adjusts/ tunes only the conclusion part of fuzzy rules in an incremental manner, for reasons specified in [7]. The controller structure is based on a Takagi Sugeno kind of

fuzzy control scheme with crisp consequents for each rule that are learned online using a reinforcement learning algorithm, which, in our case, is the Q learning algorithm.

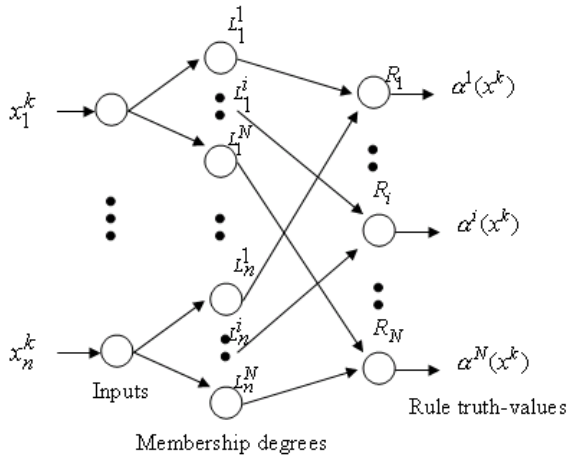

Fig. 1. FIS A structure: Fuzzy Q control scheme

For a sensed input vector $x^k = \{x_1^k, x_2^k, ..., x_n^k\}$, we find truth-value of rule $R_i : \alpha_i(x^k)$ which is the rule antecedent part of FIS (1) or the FIS A (Figure 1). For each fuzzy rule we have m possible discrete control actions $U = \{u_1, u_2, ..., u_m\}$, and a q parameter $q(i,u_i)$ associated with each control action. FQL builds an FIS with competing actions in each rule $i \in N$ designated as R_i :

$$R_i: \text{ If } x_1^k \text{ is } L_1^i \text{ and.....and } x_n^k \text{ is } L_n^i \quad \text{then } u = u_1 \text{ with } q(i,1) \text{ or } u = u_2 \text{ with } q(i,2) \text{ ..... or } u = u_m \text{ with } q(i,m) \quad (1)$$

where L_s^i = linguistic term of input variable x_s^k in rule R_i with membership function $\mu_{L_s^i}$.

Tunable parameters represented by vector q select actions, which maximize discounted sum of reinforcements received by the controller. Reinforcement learning controller can choose one action u_i from the controller action set $U = \{u_1, u_2, ..., u_m\}$, under each rule R_i . FQL algorithm uses a simplified Takagi-Sugeno FIS with N rules; therefore, the inferred global continuous action $u(x^k)$ at state x^k (defuzzified output) for an input vector x^k and rule truth-values $(\alpha_i(x^k))_{i=1}^N$ is:

$$u(x^k) = \frac{\sum_{i=1}^{N} \alpha_i(x^k) u_i}{\sum_{i=1}^{N} \alpha_i(x^k)}; u_i \in U \quad (2)$$

where u_i being the action under rule R_i .

In Fuzzy Q Learning, optimal action $u_i \in U$ under each rule R_i is the one that maximizes the q value, i.e., $q(i,u_i^*) = \max_{b \le m} q(i,b)$ and the maximizing action u_i^* is given by $q(i,u_i^*) = \arg\max_{b \le m} q(i,b)$ also referred to as the greedy action selection in RL literature. In reinforcement learning jargon, one usually selects actions that are optimal or greedy but there is a need to try new or untried actions so as to improve on action selection for enhanced performance. This is usually achieved by using an exploration/exploitation policy (EEP). One way of doing it called the Boltzmann exploration [1], but FQL uses a pseudo-stochastic or a combined directed/non-directed exploration [1]. The EEP action $u_i^\dagger$ is the action selected in rule R_i using EEP startegy, i.e., $u_i^\dagger = \varepsilon$-Greedy u_i , ε-Greedy corresponds to EEP policy while u_i^* is the maximizing action. Exploration is introduced in the selection of local actions and local probability distributions, under each rule. Action $u^\dagger(x^k)$ is random with probability ε and is defined as:

$$u_i^\dagger(x^k) = \varepsilon\text{-Greedy } u_i(x^k) = \begin{cases} u_i^r(x^k) \text{ with probability } \varepsilon \\ u_i^*(x^k) \text{ otherwise} \end{cases} \quad (3)$$

$u_i^*(x^k)$ is the greedy action, and $u_i^r(x^k)$ is a random action chosen from amongst the available control actions, under a rule R_i . Q-value for the inferred action $u(x^k)$ is:

$$Q(x^k, u(x^k)) = \frac{\sum_{i=1}^{N} \alpha_i(x^k) q(i,u_i^\dagger)}{\sum_{i=1}^{N} \alpha_i(x^k)} \quad (4)$$

and the value of state x^k (global target value) is:

$$V(x^k) = \frac{\sum_{i=1}^{N} \alpha_i(x^k) q(i, u_i^*)}{\sum_{i=1}^{N} \alpha_i(x^k)} \quad (5)$$

With action $u(x^k)$,a transition of state occurs as $x^k \xrightarrow{r} x^{k+1}$ where r is the reinforcement received by the controller. This transition information is used for calculating *temporal difference* (TD) [1] approximation error as: $\Delta Q = r + \gamma V(x^{k+1}) - Q(x^k, u(x^k))$ and q parameter values are updated as:

$$q(i, u_i^\dagger) \leftarrow q(i, u_i^\dagger) + \eta \Delta Q \frac{\alpha_i(x^k)}{\sum_{i=1}^{N} \alpha_i(x^k)} \quad (6)$$

$0 \le \gamma < 1$ is discount factor which discounts the effect of future rewards on current decisions, and η is called the learning-rate parameter [1]

III. FQL BASED UAV AUTOPILOT

We consider UAV control and navigation without a detailed flight environment [8]. The position of UAV, at any given moment is available via a GPS receiver. The heading angle and heading speed of the UAV are given by [8]:

$$\begin{aligned} \dot{v} &= -\frac{1}{\tau_v} v + \frac{1}{\tau_v} v_c \\ \ddot{\theta} &= \frac{1}{\tau_\theta} \dot{\theta} + \frac{1}{\tau_\theta} \theta_c \end{aligned} \quad (7)$$

where v_c and θ_c are command velocity and heading angle of autopilot. τ_v and τ_θ are time constants of autopilot.

As in [8], we reconstruct the UAV system equations in two dimension coordinates by defining:

$$\begin{aligned} x &= l \cos\theta \\ y &= l \sin\theta \end{aligned} \quad (8)$$

where x and y are horizontal and vertical coordinates of the UAV with respect to the origin of flight and $\dot{l} = v$. The corresponding scheme has been shown in Figure 2.

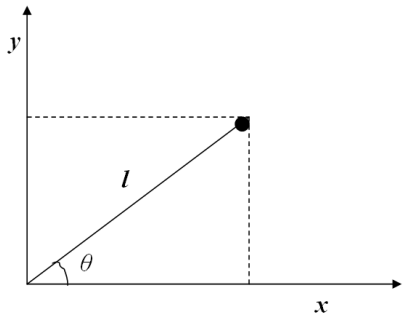

Fig. 2. UAV in Two dimensions

Differentiating (8), we get

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & -l\sin\theta \\ \sin\theta & l\cos\theta \end{pmatrix} \begin{pmatrix} v \\ \dot{\theta} \end{pmatrix} \quad (9)$$

Control objective is to design control input such that actual flight path tracks the desired trajectory. We simulate the dynamics of UAV by Euler's method using a sample time $T = 0.1$ sec.

At any given instant *t*, the position of UAV (*x* (*t*), *y* (*t*)) is given by the GPS. We define the UAV state as $(e_x(t), \Delta e_x(t), e_y(t), \Delta e_y(t))$ where $e_x(t) = x_d(t) - x(t)$ is the horizontal tracking error, $\Delta e_x(t) = e_x(t) - e_x(t-1)$ is change in horizontal tracking error, $e_y(t) = y_d(t) - y(t)$ is vertical tracking error, $\Delta e_y(t) = e_y(t) - e_y(t-1)$ is change in vertical tracking error. The state vector acts as input for the fuzzy Q learning controller. We take speed of UAV (*v*) and the angle of attack (θ) as the outputs of the FQL controllers (one module each for UAV speed and angle of attack).

In FQL, the antecedent part (1) of each rule is fuzzy. We use Gaussian membership function to fuzzify each input variable. We use two FQL controller modules one each for speed and angle of attack. The inputs for the FQL controller (speed) are $(e_x(t), \Delta e_x(t))$ while inputs for the FQL controller (angle of attack) are $(e_y(t), \Delta e_y(t))$. We partition each variable into three fuzzy subsets, i.e., $e_x(t)$ and $\Delta e_x(t)$ are divided into three subsets thereby generating nine rules for each FQL controller. Total number of rules for the UAV controller is thus eighteen. Membership degree of input variables is computed using

$$\mu_{l_p}(x_j) = \mathrm{e}^{\frac{-(x_j - x_j^{l_p})^2}{2\sigma_j^2}} ; l_p = 1,2,3; \quad j = 1,2,3,4; \quad (10)$$

i.e, Gaussian membership function, where l_p are fuzzy labels for variable *j*, $(x_1 = e_x(t), x_2 = \Delta e_x(t), x_3 = e_y(t), x_4 = \Delta e_y(t))$. Fuzzy label

centers are defined as $x_j^{l_p} = a_j + b_j(l_p - 1)$ with $a_1 = -15, a_2 = -24, a_3 = -60, a_4 = -20$, $b_1 = 15, b_2 = 24, b_3 = 60, b_4 = 20$, and widths defined by $\sigma_1 = 30, \sigma_2 = 48, \sigma_3 = 120, \sigma_4 = 40$.

The current state vector x^k is matched with the fuuzy sets laid over state variables for each controller module for generating rule strength values $\alpha_i(x^k)$. Greedy action selection is done for each rule, u_i^* based on the current q values. Next step is to produce what may be called a global continuous action $u^*(x^k)$. We use the q values that correspond to the greedy action to form the global target value $V(x^k)$ as per (5).

EEP strategy is implemented as per (3) to generate $u_i^\dagger(x^k)$ which are used to generate the global Q value $Q(x^k, u(x^k))$ as per (4). To the global continuous action $u^*(x^k)$ we add a conventional proportional action [9] $u_p = K_p(e)$ to generate the overall control action $u = u^* + u_p$ which is applied to the UAV. The current coordinates of the UAV are sensed by the GPS sensor and compared with the desired coordinates (as per the desired flight path) to generate an error signal. This positional error signal is used to generate a reinforcement signal as:

$$r = -10 \text{ if } abs(e^k) < abs(e^{k+1}) \text{ or error increases} \\ = +10 \text{ if } abs(e^k) > abs(e^{k+1}) \quad (11)$$

Next, using this reward or reinforcement signal, we calculate the TD error [1] ΔQ. Finally, the q values are updated using (6). We use one FQL controller for speed (v) and another one for the heading angle θ .Overall structure of FQL based autopilot is shown in Figure 3.

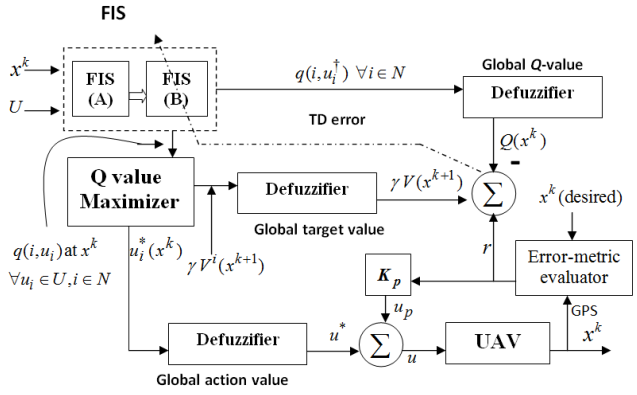

Fig. 3. UAV Autopilot based on FQL

IV. Simulation Results

We use the proposed FQL based autopilot on the simulated UAV (9). The simulation parameters for RL are: discount factor $\gamma = 0.9$, learning rate $\eta = 0.01$, explore parameter $\varepsilon = 0.2$ and other simulation parameters are: k_p : $k_{px} = 19$, $k_{py} = 0.2$, initial state $x^0 = (0.1, 0.1, 0.2, 0.05)^t$, action set for speed FQL controller (-20 0 +20) m/s, action set for the heading angle FQL controller (-10 0 +10) degrees. These parameter settings were obtained after a comprehensive trial and error procedure. The results depict FQL based autopilot's performance after 5 trials. We take three case scenarios:

Altitude control

In this case the UAV has to take off and attain a desired altitude and maintain the altitude during the flight. The UAV takes off and reaches an altitude of y = 500 m at a horizontal position x = 1000 m and then maintains the altitude for the rest of the flight. Figure 4 show the performance of the FQL based UAV for the altitude control task. As can be seen, that except for the initial take off, UAV is able to maintain the desired flight altitude of 500m.

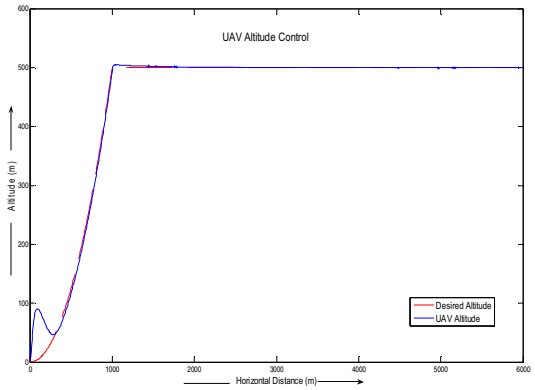

Fig. 4. Altitude control of the FQL based UAV

Trajectory control

In the next task, the UAV has to take off and reach a desired altitude y = 500m at horizontal position x = 1000m and then follow a desired flight path. We have used sine waves with different amplitudes at different horizontal positions for simulating the trajectory. Figure 5 shows the performance of the UAV for the trajectory control task. Here again except for the initial take off the UAV is able to closely follow the desired trajectory.

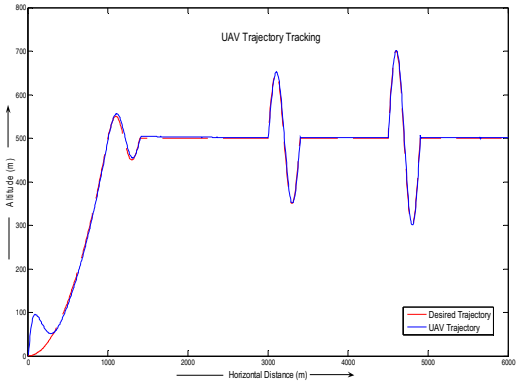

Fig. 5. Trajectory control of the FQL based UAV

Reconnaissance flight control

Our final task involves the UAV on a reconnaissance mission wherein the UAV has to take off and reach a flight level of y = 550m at horizontal position x = 1000m. Then the UAV must maintain the flight level of 550m from x = 1000m to x = 6000m and then the UAV returns to the take-off point. Figure 6 shows the performance of the FQL based UAV for this task. The UAV is able to complete the reconnaissance mission quite closely.

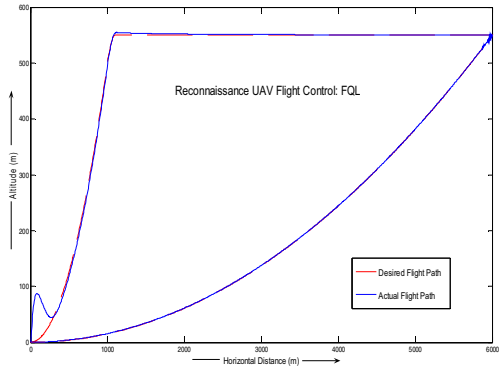

Fig. 6. Reconnaissance flight of the FQL based UAV

V. Conclusions and Future Scope of Work

This paper presents the design of a UAV autopilot based on FQL. The results show the effectiveness of a UAV autopilot based on a reinforcement learning scheme that uses fuzzy systems. It is seen that the UAV is able to maintain desired altitude, track a desired trajectory and even carry out a precise reconnaissance mission. In future we would apply the proposed approach on a real UAV and then onto a group of UAVs using the Decentralized Partially Observable Markov Decision Process (Dec-POMDPs) framework [10].

References

[1] R. S. Sutton, and A.G. Barto, *Reinforcement learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[2] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Trans. Systems, Man, and Cybernet.—Part C: Applications and Reviews*, vol. 28, no. 3, pp. 338-355, August 1998.

[3] L. Doitsidis, K. P. Valavanis, N. C. Tsourveloudis, and M. Kontitsis, "A Framework for Fuzzy Logic Based UAV Navigation and Control", *Proc. of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans, LA, April 2004.

[4] R.S. Sutton, A.G. Barto, and R.J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems Magazine*, vol. 12, no.2, pp. 19-22, 1992.

[5] C. J. C. H. Watkins, and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.

[6] T. J. Ross. *Fuzzy Logic with Engineering Applications*. Wiley & Sons Inc. 2nd edition, 2004.

[7] P. Y. Glorennec, and L. Jouffe, "Fuzzy Q-learning," in *Proc. 6th IEEE Conf. Fuzzy Systems*, Barcelona, Spain, vol. 2, July 1-5, 1997, pp. 659-662.

[8] Tao Dong, X. H. Liao, R. Zhang, Zhao Sun and Y. D. Song, "Path Tracking and Obstacle Avoidance of UAVs - Fuzzy Logic Approach", Fuzz IEEE 2005 Reno, Nevada, USA, May 22-25, 2005

[9] Rajneesh Sharma, and M. Gopal, "A Markov Game Adaptive Fuzzy Controller for Robot Manipulators," *IEEE Transactions on Fuzzy Systems* vol. 16, issue 1, pp. 171 – 186, Feb. 2008.

[10] Rajneesh Sharma, and Matthijs T. J. Spaan, " Bayesian Game Based Fuzzy Reinforcement Learning Control for Decentralized POMDPs ", *IEEE Transactions on Computational Intelligence in AI and Games,* vol. 4, issue 4, pp. 309-328, December 2012.