# *UAV heading controller Using Reinforcement learning*

Stephen Kimathi,
Department of Electrical and Electronic Engineering
Dedan Kimathi University of Technology
Nyeri, Kenya
stephen.kimathi@dkut.ac.ke

Samuel Kang'ethe and Peter Kihato,
Department of Electrical and Electronic Engineering
Jomo Kenyatta University of Agriculture and Technology
Nairobi, Kenya
samuelkangethe7@gmail.com, kamitakhat1@gmail.com

***Abstract* - The control of heading of an Unmanned Aerial Vehicle is a vital operation. It is accomplished by employing a design of control algorithms that control its flying direction. The available autopilots exploit Proportional-Integral-Derivative (PID) based heading controllers. Here we propose an adaptive controller based on reinforcement learning. The heading controller will be designed in Matlab/Simulink for controlling a UAV in X-Plane test platform. Through this platform, the performance of the designed controller is compared with that of a well tuned PID controller using real time simulations. The results show that the proposed method performs better.**

**Keywords** - UAV, Reinforcement Learning, PID, X-Plane

## I. INTRODUCTION

UAVs have a broad range of applications that include surveillance, search and rescue [1], target tracking, digital mapping and weather observations. To accomplish these autonomous missions, it is essential to have a reliable heading control thus an autopilot system is an essential component. UAVs can be remotely controlled, semi-autonomous, autonomous or combining any of these. They present the future of aerial vehicles and are of great interest to the control engineering fraternity.

Due to the nonlinearity of the system dynamics and parameter uncertainty in UAVs, several control techniques including PID control [2], where two PID controllers were used in tandem, for the lateral and longitudinal motions. In [1] $H_\infty$ control strategy was used. Adaptive control strategies have been applied; fuzzy systems in [3], active disturbance rejection control, ADRC in [4]. In [5] an adaptive backstepping approach was used to obtain directional control of a fixed wing UAV, where the dynamics of the cross track error was derived using the lateral system equations of motion.

In this paper an adaptive control strategy based on reinforcement learning technique is presented. This is due to the high nonlinearity of the system dynamics associated with small flying aerial vehicles and lack of complete knowledge of vehicle dynamics for parameter estimation. Reinforcement learning explores actions from available courses of action and chooses the best course of action based on the reward it gets, hence suitable for this kind of application.

This rest of this paper is organized as follows: Section II presents the basics of UAV control, section III introduces reinforcement learning principles, and gives a brief overview of X-Plane test platform, section IV gives the design of the controller and experimental setup, section V provides the results and discussion and the conclusion is given in the last section.

## II. UAV BASICS

Fig. 1 shows that UAV has three principle axes of motion ($x$,$y$,$z$) from its centre of gravity [6].

The position control of the UAV is converted to its corresponding angular moments of rolling, pitching and yawing motions. The control of the roll ($\varphi$), pitch ($\theta$) and yaw ($\Psi$) angles in a fixed wing UAV are aided by the control surfaces such that

- Ailerons control the rolling

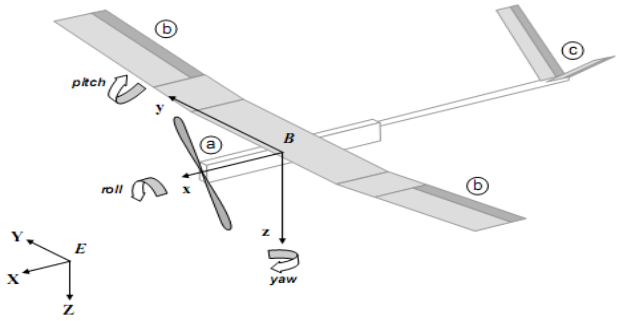


**Fig. 1: UAV axes of motion**

- Elevator control the pitching
- Rudder control the yawing movement,

And the engines throttle to control the engines power.

The derivation of equations of motion for fixed wing UAV are given in [7]. The lateral state space model decoupled from within the linear model is then used with

inputs of aileron and rudder to control the heading of an aircraft [8]. The decoupled lateral model state space equation is given as

$$\dot{\boldsymbol{x}} = \boldsymbol{A_{lat}x_{lat}} + \boldsymbol{B_{lat}u_{lat}} \quad (1)$$

Where $x_{lat}$ is the decoupled lateral state space model, $u_{lat}$ the control input(s), $A_{lat}$ is the state matrix and $B_{lat}$ the input matrix.

. The linear lateral state space model as given in [7] is

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\beta}} \\ \dot{\boldsymbol{r}} \\ \dot{\boldsymbol{\psi}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L_p} & \boldsymbol{L_\beta} & \boldsymbol{L_r} & \boldsymbol{0} \\ \boldsymbol{Y_p} & \boldsymbol{Y_\beta} & \boldsymbol{Y_r - 1} & \boldsymbol{mgcos\theta_e} \\ \boldsymbol{N_p} & \boldsymbol{N_\beta} & \boldsymbol{N_r} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\beta} \\ \boldsymbol{r} \\ \boldsymbol{\phi} \end{bmatrix} + \begin{bmatrix} \boldsymbol{L_{\delta a}} & \boldsymbol{L_{\delta r}} \\ \boldsymbol{Y_{\delta a}} & \boldsymbol{Y_{\delta r}} \\ \boldsymbol{N_{\delta a}} & \boldsymbol{N_{\delta r}} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta_a} \\ \boldsymbol{\delta_r} \end{bmatrix} \quad (2)$$

where $[\rho \ \beta \ r \ \varphi]^T$ as the state variables. $\rho$ is the roll rate, $\beta$ is the sideslip angle, $r$ is the yawing rate and $\varphi$ is the roll angle. The inputs consists of $\delta_a$, the aileron deflection and $\delta_r$, the rudder deflection.

## III. REINFORCEMENT LEARNING

Reinforcement learning, RL is learning what to do i.e. how to map situations to actions, so as to maximize some numerical reward. The learning agent is not told the correct actions; instead it explores the possible actions and remembers the reward it receives. The RL model consists of a set of environment states $s_t \in S$; a set of actions $a_t \in A$ that an agent can perform at each state, and as a consequence of its action, the agent receives a numerical reward $r_t$. At each time step, an agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's policy and denoted as $\pi_t$ which maximizes the cumulative reward of an agent over time as [9]

$$\boldsymbol{R} = \sum_{t=0}^{\infty} \boldsymbol{\gamma^t r_t} \quad (3)$$

where $0 < \gamma < 1$ is a discount factor, which discounts the value of future rewards.

### X-Plane

X-Plane is powerful flight simulator for personal computers. It is not a game but rather an engineering tool that can be used to predict the flying qualities of fixed and rotary wing aircraft with considerable accuracy. The accuracy of X-Plane makes it a useful tool to predict and test the performance of an aircraft and its characteristics. It has the capacity to send and receive data to and from other devices. This is achieved using the User Datagram Protocol, UDP [10].

## IV. CONTROLLER DESIGN

The main control objective is to obtain lateral-directional control in order to follow a desired reference heading. Reinforcement learning, *sarsa* algorithm is used to design the controller.

From the state space model presented above, the associated Ricatti coefficient, P is calculated. This Ricatti coefficient is used to calculate the

Cost function for each state-action pair using a simple Lyapunov function

$$\boldsymbol{Q(s,a)} = \boldsymbol{X^T} P \, \mathrm{X} \quad (4)$$

which was reformulated to include references as

$$\boldsymbol{Q(s,a) = (X-Z)^T P\,(X-Z)} \quad (5)$$

where, X are states and Z are references.

A reward function is calculated as the deviation of the target state from the desired state as in [11] which in this work is taken as the heading error.

$$\boldsymbol{r(s) = -C_1(\emptyset - \emptyset_{ref})} \quad (6)$$

The total value function is calculated, which is a sum of previous state-action value function, the current reward and the current state-action value function as

$$Q(s_1,a_1) \leftarrow Q(s_1,a_1) + \alpha[r_2 + \gamma Q(s_2,a_2) - Q(s_1,a_1)]$$

$$\boldsymbol{Q(s_2,a_2) \leftarrow Q(s_2,a_2) + \alpha[r_3 + \gamma Q(s_3,a_3) - Q(s_2,a_2)]} \quad (7)$$

$$\vdots$$

This is updated as the total value function for the next cycle of learning. According to [9], it is allowed to have a *one step gradient search* of the value function; exploitation for ease of real time implementation and less computational burden. This was achieved as the temporal difference which is evaluated as

$$\boldsymbol{\delta_t = r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)} \quad (8)$$

According to [12], an optimal control effort is given as $u^* = -\mathrm{KX}$ but according to differential games algorithm, this is expressed as

$$\boldsymbol{u^* = -\frac{1}{2} R^{-1} g^T \nabla V^*} \quad (9)$$

where $\nabla V^*$ is taken as the change in the optimal cost function. In this work, the temporal difference between successive cost functions is used as the reinforcement to the optimal control.

The control signals from a feed forward neural network are compared with this control signal. Then back propagation algorithm updates the feedforward neural network weights using back propagation. This means that we are correcting the error in the control deflection in the next control deflection through update of neural network weights thus slowly taking our deflections to the best available control effort in each consecutive cycle.

In this work the issue of exploration which is also central to RL alongside exploitation is not addressed. Here the RL controller only exploits the value function and new states are found by inference of favorable value functions.

## Implementation

The aerodynamic coefficients that constitute the values in the mathematical model as provided for in [7] are taken from [1] and [8] as $m = 1.9kg, b = 1.2m, g = 9.8\frac{m}{s^2}$, $S = 0.32\ m^2, \bar{c} = 0.3\ m, \rho = \frac{1.225\,kg}{m^3}, \frac{1}{\pi eAR} = 0.0815$

With the above parameters, the trim condition was obtained as:

$$A = \begin{bmatrix} -1.4000 & 0 & 0 & 9.4953 \\ -30.9000 & -12.8000 & 14.4000 & 0 \\ 1.4781 & -0.4480 & -6.080 & 0 \\ 0 & 1.0000 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0.7412 \\ 61.4000 & 12.4000 \\ -3.6700 & 15.0000 \\ 0 & 0 \end{bmatrix}$$

Simulations were carried out in Matlab/Simulink. The controller designed using the reinforcement learning method described above was compared with a well tuned PID controller. The two controllers were used for real time UAV control in X-Plane using the schematic shown in Fig. 6 below.

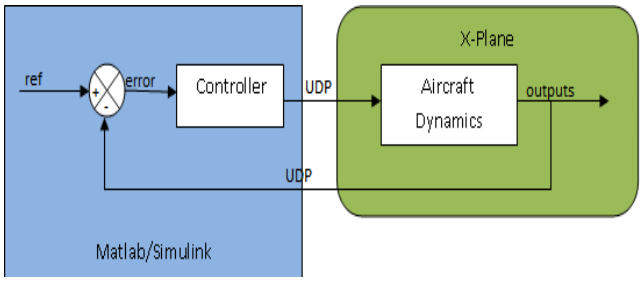


**Fig. 6: Software-In-the-Loop Simulation**

## V. RESULTS AND DISCUSSION

The state space model given above was used to design the controller in Matlab/Simulink as described in the preceding section. Initially, a single step heading reference was designed for an actual flight regime of 500 initial heading to 1000 final heading and the results were as below. Fig. 9 shows the PID response to the step reference. The rise time is 1.4007 seconds and the system does not settle within the bounds of 0.5% of final value that was specified. The percentage overshoot was calculated to be 6.3119%., which is close to that reported in [2] where a mathematical model was used.

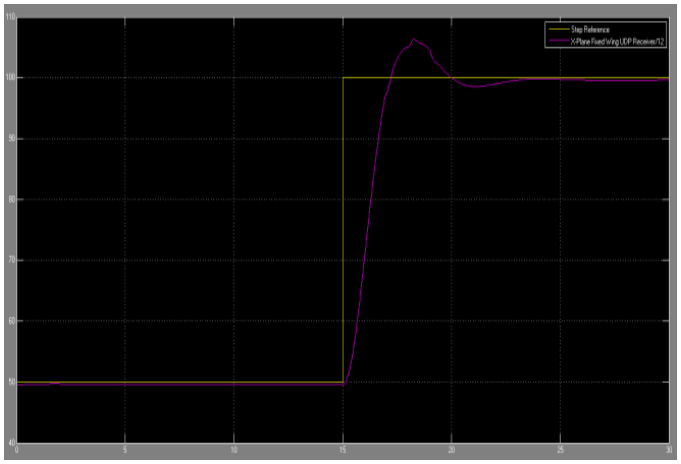

Fig. 9: PID controller response to a step heading reference

The rise time is 1.2663 seconds as compared to PID's 1.4007 seconds and the system settles after 21.371 seconds. The percentage overshoot is 3.1083% for the RL controller which is lesser than the PID's.

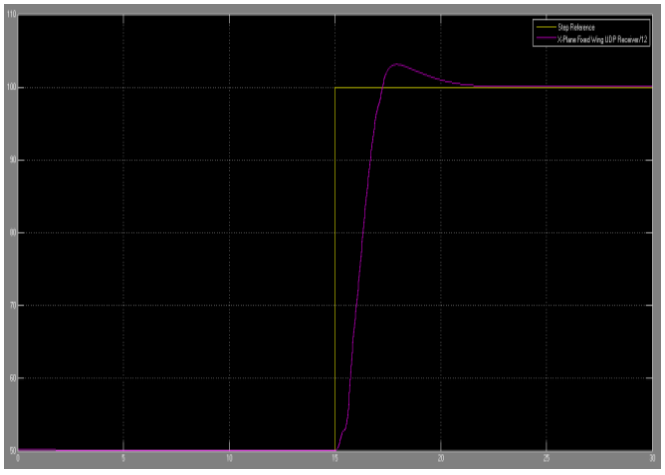

Fig. 10:RL controller response to a step heading

The next results are for another flight regime in X-Plane where the UAV is commanded from an initial heading of 400. The aircraft is then commanded to go to 800 and then 500 and so on. The Fig. 11 and Fig. 12 show the results where the heading in degrees is plotted on the vertical axis against time in seconds on the horizontal axis.

From Fig. 12, in the first 10 seconds the response of the RL controller is poor, as compared to that of the PID controller in Fig. 11; this is due to the fact that the artificial neural network weights are being continuously adjusted where initially big adjustments are expected, then the neural weights settle around the optimum weights. It should be noted that the initial weights were randomized. After 10 seconds the response stabilizes and follows the reference more robustly than the PID controller. It can also be seen that there is an overshoot on the first step heading change but due to adaptation that overshoot is eliminated in the consecutive step heading angle changes as is evident from the Fig. 12.

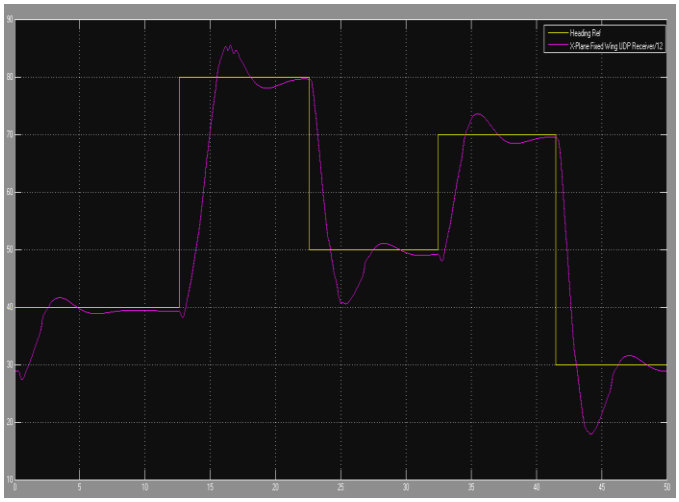

Fig. 11: PID controller response in X-Plane

Due to the poor initial tracking response of the RL controller, the data from Fig. 12 was used to train the designed controller neural network weights using the nonlinear autoregressive network with exogenous inputs, NARX in Matlab. The neural network weights achieved were set as initial weights in the designed RL controller and the simulation was ran again. The response was as

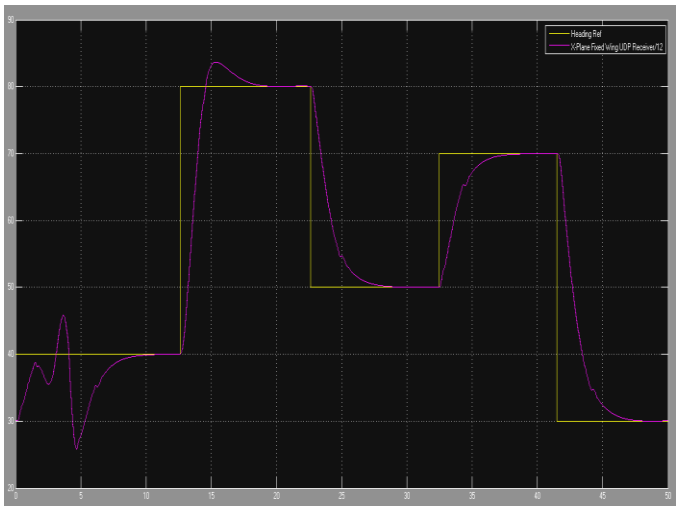

Fig. 12: RL controller response in X-Plane

shown in Fig. 13. There were some improvements on the initial tracking; there is less oscillations and overshoots during the initial stages as compared to Fig. 12 where the initial weights were randomly initialized.

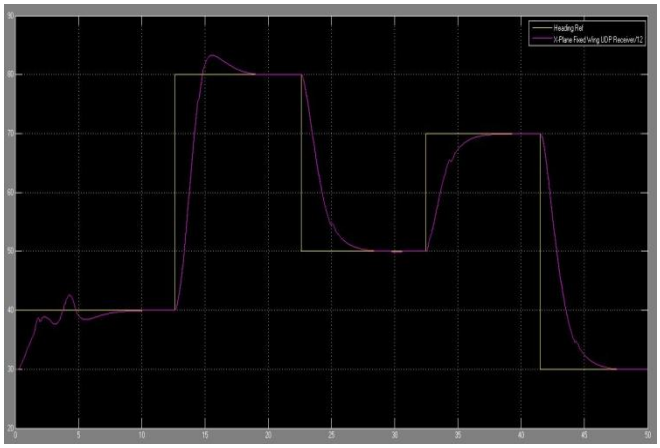

Fig. 2:RL controller response after training

## Conclusion

A reinforcement learning controller for heading control in a fixed wing UAV has been presented. As it has been shown using simulations in X-Plane, PID controller performs well to tracking the reference initially from any random position but its tracking response is poor, it overshoots in every step reference heading change and does not track the reference robustly. The RL controller produced better tracking results and shows the usability of this method.

## References


[1] H. Ferreira et.al, "Disturbance Rejection in a Fixed WIng UAV using nonlinear H-Infinty stste feedback," in *9th International Conference on Control and Automation*, Santiago, USA, 2011.

[2] A. Mansoor et. al, "Heading control of a FIxed wing UAV using Alternate control surfaces," *IEEE*, vol. Vol. 2, December 2012.

[3] D. Stojcsics, "Fuzzy controller for small size Unmanned Aerial Vehicles," in *10th IEEE Intl. Symposium on Applied machine Intelligence and Informatics*, Herl'any, Slovakia, 2012, pp. 91- 95.

[4] L. Jing-Mei and Z. Ke, "Design of Active Disturbance Rejection Controller for Autonomous Aerial Refuelling UAV," in *IEEE*, 2013.

[5] A. Brezoescu et. al, "Adaptive Trajectory following for a Fixed wing UAV in Presence of Crosswind," *Journal of Intelligent Robotic Systems*, vol. Vol. 69, pp. 257-271, 2013.

[6] A. Noth et. al, "Dynamic Modelling of Fixed Wing UAVs," Swiss Federal Institute of Technology, Zurich, Laboratoty Report 2006.

[7] R. Beard and T. McLain, *Small Unmanned Aircraft; Theory and Practice*, 1st ed. Princeton, New Jersey: Princeton University Press, 2012.

[8] Y. Paw, "Synthesis and Validation of Flight Control for UAV," University of Minnesota, Miinnesota, PhD Thesis Dec. 2009.

[9] R. Sutton and A. Barto, *Reinforcement Learning; An Introduction*, 1st ed. Massachusetts: MIT press, 2005.

[10] R. Lucio and O. Neusa, "UAV Autopilot Controllers Test Platform Using Matlab/Simulink and X-Plane," in *40th ASEE/IEEE Frontiers in Education Conference*, Washington, Dc, 2010, pp. 6262-6269.

[11] B. Haitham et. al, "Controller Design for Quadrotor UAVs using Reinforcement Learning," in *IEEE International Conference on Control Applications*, Yokohama, Japan, 2010, pp. 2130-2135.

[12] B. Anderson and J. Moore, *Optimal Control, Linear quadratic Methods*. London, UK: Prentice-Hall International, 1989.