Aircraft Control with Neural Networks

Akif Altun

akif.altun@roketsan.com.tr

Prof. Dr. Mehmet Önder Efe

onderefe@ieee.org

Abstract **– In this study, a controller is designed by using neural networks. Performance of this controller is compared with an another controller which is designed by using classical control methods. They are both tested with the same flight scenario to analyse their capabilities. This comparison provides comprehension about the dynamical capabilities of the neural networks as a controller, so that in further studies, robustness against model uncertainties and sudden unplanned airframe changes may be achieved with a neurocontroller that is subjected to online training.**

1. INTRODUCTION

Using neural networks in control systems is a worthy area to study. However, in aeronautics, usually it is not prefered. The reason of this situation is that, the output of the neural networks may not be explicit for every possible condition due to its complex structure. The neural networks are universal approximators and there is always a risk of overfitting. That means the system can produce undesired outputs even in the predefined boundaries that involves the training data. On the other hand, reliability standards are very high in aviation industry, so it is more difficult to certify such systems. In this study, artificial neural networks is used to control an aircraft to investigate its capabilities. Other than this neurocontroller, a second controller is designed with classical control methods to compare their performances.

To do this comparison, a simulation environment is prepared. The simulation runs the equations of motion with 6 degrees of freedom. These equations can be found in reference [4]. It models numerous kinematic and dynamic parameters along with the engagement area properties like speed of sound, air density, etc. Those parameteres are needed to carry out the specified flight scenario.

2. SIMULATION

The simulation is made in Matlab/Simulink environment. In the simulation, motion of the aircraft is modeled using the flight mechanics equations. These equations involve all the major dynamics of the aircraft in three dimensional space corresponding to pitch plane, yaw plane and roll plane. The simulation solves the corresponding differential equations of motion and so that, position, velocity and acceleration of the aircraft is obtained at every time step.

To solve the differential equations of motion, aircraft's dynamical, inertial and geometrical properties must be known. In this study, an anti-tank missile called "Javelin" is used to work with. Table 1 shows some of the major geometrical and inertial properties of the Javelin which are used in the simulation. Figure 1 shows a drawing of the Javelin.

Mass	CG	I_xx	I_yy	I_zz	Length	Diameter
10.15 kg	0.446 m	0.0231 kg/m^2	0.914 kg/m^2	0.914 kg/m^2	1.1 m	0.127 m

Table 1 – Some inertial and geometrical properties

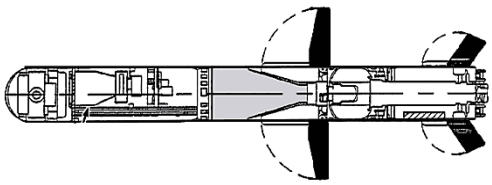

Figure 1 – Javelin missile

For expressing the dynamical characteristics of the missile, an aerodynamic database is needed to use in the simulation so that the dynamical differential equations can be solved. To create an aerodynamic database, MISSILE DATCOM is used. The process is explained in detail in reference [2]. Once the geometrical and inertial properties are known, for all three axes, force and moment derivatives with respect to concerned dynamic and kinematic properties can be extracted by using this software. In the flight scenario, the altitude change is very small and inertial properties of the missile are constant. Thus, the aerodynamic database is extracted by concerning only the Mach number change.

In simulation, the booster of the missile is not modeled. Thus, the boost phase of the flight is not simulated. The flight of the missile is started in the air at 120 m altitude and at 0.6 Mach.

978-1-7281-0521-5/19/$31.00 ©2019 IEEE

2.1. Flight Scenario

Flight scenario consists of two parts: altitude hold and the guidance. In the first part of the flight, it is expected that the autopilot keeps the missile`s altitude constant. After that, the guidance phase starts and missile heads to some target on the ground. The time of switching to the guidance mode and the location of the target is pre-specified.

2.2. Guidance and Autopilot

To control the missile and follow a trajectory according to the flight scenario, guidance and autopilot algorithms are added to the simulation.

In this study, the missile is controlled by using three distinct autopilots, each of them controls the pitch plane, yaw plane and roll plane dynamics, seperately. Pitch and yaw autopilots control the corresponding component of acceleration commands. On the other hand, the roll autopilot controls the roll angle of the missile and always tries to keep this angle at zero because the missile is using skid-to-turn principle to maneuver. Also, all three autopilots are responsible for stabilizing the aircraft. In this part of this study, classical control methods are used. Autopilots have full-state feedback structure and designed by using pole placement techniques.

Guidance algorithm produces the acceleration commands which are the inputs of the autopilots. It needs velocity and position information of the missile and the target, to use them with some guidance law. By doing that, it calculates the acceleration that missile must have, to be able to hit the target. The calculation is done vectorially, so that, the components of the acceleration command vector can be send to the seperate autopilots.

In the design stage of the autopilot and guidance algorithms, miss distance and flight time are tried to be minimized as the performance requirements.

3. NEURAL NETWORKS

Once the simulation is ready, the neurocontroller can be designed because to design a neural network, a training data is required. Here, the training data is extracted from the simulation.

The neurocontroller is trained offline by using the training data. However, in simulation, it is planned that the neurocontroller will continue to train with online learning. That means, an emulator, that will be used as a model of the system, is needed in the simulation. The emulator will provide the required information for online learning of the neurocontroller. To obtain such an emulator, a second network is designed and it will also continue to train with online learning after the same offline learning procedure. In the simulation, autopilot structure will be replaced with this neurocontroller and emulator couple. The structure of the new control logic is shown in Figure 2.

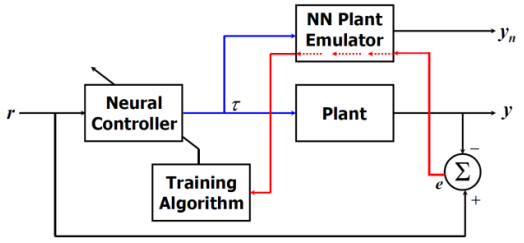

Figure 2 – Control logic of the new control system designed by using neural networks

Both of the networks are using Levenberg-Marquardt algorithm to update their parameters. The update rule of the Levenberg-Marquardt algorithm as follows:

$$w_{t+1} = w_t - \left(J_t^T J_t + \mu_t I\right)^{-1} J_t^T E_t$$

where, μ is the stepsize, I is an identity matrix.

$$E = \begin{bmatrix} e_1^1 & \dots & e_K^1 & e_1^2 & \dots & e_K^2 & \dots & e_1^P & \dots & e_K^P \end{bmatrix}^T$$

$$J = \begin{bmatrix} \frac{\partial e_1^1}{\partial w_1} & \frac{\partial e_1^1}{\partial w_2} & \dots & \frac{\partial e_1^1}{\partial w_N} \\ \frac{\partial e_2^1}{\partial w_1} & \frac{\partial e_2^1}{\partial w_2} & \dots & \frac{\partial e_2^1}{\partial w_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_K^1}{\partial w_1} & \frac{\partial e_K^1}{\partial w_2} & \dots & \frac{\partial e_K^1}{\partial w_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_1^P}{\partial w_1} & \frac{\partial e_1^P}{\partial w_2} & \dots & \frac{\partial e_1^P}{\partial w_N} \\ \frac{\partial e_2^P}{\partial w_1} & \frac{\partial e_2^P}{\partial w_2} & \dots & \frac{\partial e_2^P}{\partial w_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e_K^P}{\partial w_1} & \frac{\partial e_K^P}{\partial w_2} & \dots & \frac{\partial e_K^P}{\partial w_3} \end{bmatrix}$$

In the above equations, K is the output number of the network and P is the pair number of the training data.

3.1. Neurocontroller

The neurocontroller is a neural network structure that tries to estimate the control system behaviour of the missile. In this neural network 4-10-1 configuration is used. There are 4 neurons in the input layer, 10 neurons in the hidden layer and 1 neuron in the output layer. The inputs are, the acceleration command, current values of the acceleration of the missile and the Mach number and lastly the previous step value of the actuation command. The ouput parameter of the neural network is the current value of the actuation command.

The reason why the previous step value of the actuation command is chosen as one of the inputs of the neural network is that, with a that kind of input, the nonlinearity of the input-output relationship is reflected better to the network. Without this modification, the performance of the network is inadequate to estimate the controller`s output.

In the offline training, these input and output parameters are used as the training data in pairs. To train the controller online, derivative of the error between the acceleration command and the current value of the acceleration of missile, e_a, with respect to the actuation command is used. This derivative, $\frac{\partial e_a}{\partial \delta_{com}}$, and the error value, e_a, is used for updating the weight parameters of the neurocontroller. In the offline training, this procedure was done with the error between the desired value and the output of the network, as usual.

As a result, $\frac{\partial e_a}{\partial \delta_{com}}$ appears as a multiplier with the Jacobian expression in the update rule. Besides, e_a is used as the error expression, in the same update rule.

The training data is obtained from the simulation while the autopilot, the one that is designed with classical control techniques, is driving the missile. To get a wide range of information about the controller dynamics, acceleration command is given to the system manually. The acceleration command is shown in Figure 3.

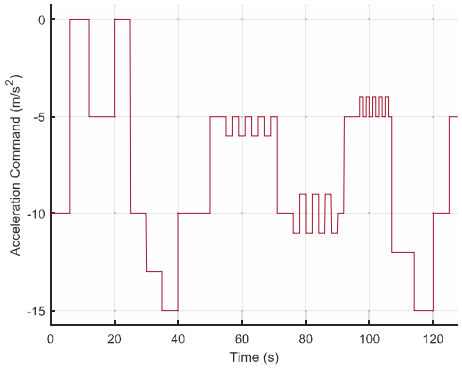

Figure 3 – Acceleration command (gravity compansated)

The training data was normalized before training the network. With a stepsize of 1, and 50 step iteration, the neural network is trained. After the training, the corresponding actuation command of the system and the response of the neural network for that is shown in Figure 4.

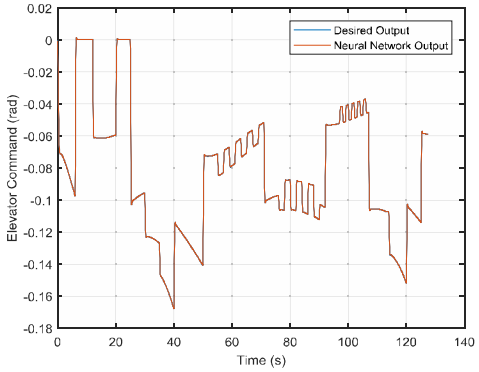

Figure 4 – Desired output vs. neurocontroller output

As seen in the Figure 4, curves are not exactly overlapping each other and there are small differences on the edges. However, it can be said that the designed neural network is capable enough to represent the controller dynamics.

The same procedure that carried out for the pitch autopilot, is applied for the yaw and roll autopilots to control the corresponding dynamics with neural networks, either.

3.2. Emulator

Emulator is a neural network structure that is modeling the system dynamics. It gets the necessary inputs and tries to estimate the system output. In this neural network, 3-10-1 configuration is used. There are 3 neurons in the input layer, 10 neurons in the hidden layer and 1 neuron in the

output layer. The inputs are, actuation command, acceleration of the missile and the Mach number. In this part, all the inputs belong to the previous step of the simulation so that the emulator can estimate the current value of the system output. The system output is the current value of the acceleartion of the missile.

In the offline training, these input and output parameters are used as the training data in pairs. In the online training, the procedure goes the same way, as opposed to the neurocontroller design stage. Because this time, the desired output value, the current acceleration value of the missile, is available and can be obtained from the simulation. Thus, the method used for calculating the error value and updating the weight parameters will not differ from the offline training.

The training data for the emulator is obtained by the same way as the neurocontroller. The only difference is that, the alignments and the sortings of the training data arranged according to the neural structure of the emulator. Other than that, the rest of the procedure is the same. The same acceleration command is used for training the emulator. After training, the actual acceleration of the system and the response of the network for that is shown in Figure 5.

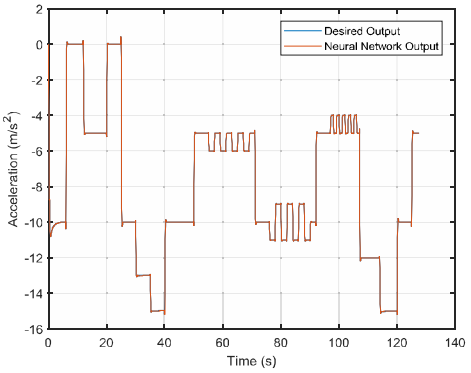

Figure 5 - Desired output vs. emulator output

As seen in Figure 5, curves are almost on top of each other so it can be said that this network is ready to use in the simulation to represent the system dynamics.

The same procedure that carried out for the pitch plane dynamics, is applied for the yaw and roll planes to model the corresponding dynamics with neural networks, either.

4. FLIGHT PERFORMANCE

After designing the neurocontroller and the emulator, those neural networks are tested in the simulation environment. Same flight scenario is used for both the classical controller case and the neurocontroller case. In this scenario, simulation starts with 0.6 Mach initial speed in the direction of x-axis of the body frame at 120 m altitude. Initially, all the orientation angles, angle rates and the other components of velocity vector are zero. Also, the missile initial position is [0,0,120] with respect to an initial frame of reference that is chosen for this problem to work with.

In those conditions, missile is expected to hold its altitude for the first 5 seconds. After that, guidance algorithm becomes active and leads the missile to a target at the ground. The position of the target with respect to the same frame of reference is [2000,0,0] in meters.

Following graphics show the performances of controllers according to the simulation results:

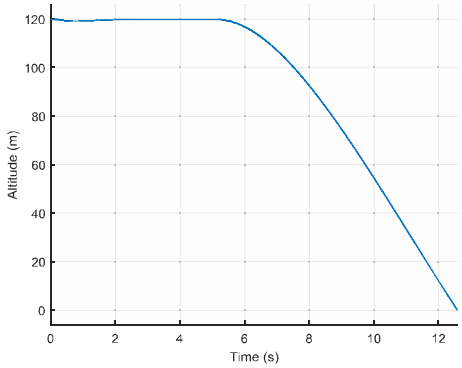

Figure 6 – Altitude vs. time for the classical controller (miss distance is 0.005 m)

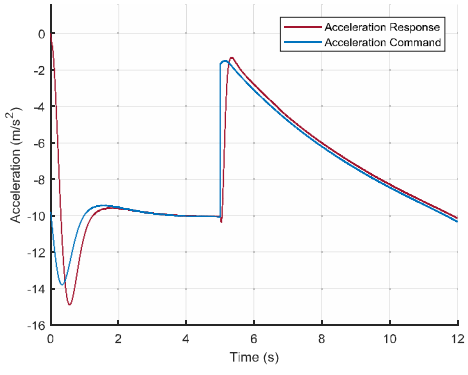

Figure 7 – Acceleration command & response for the classical controller

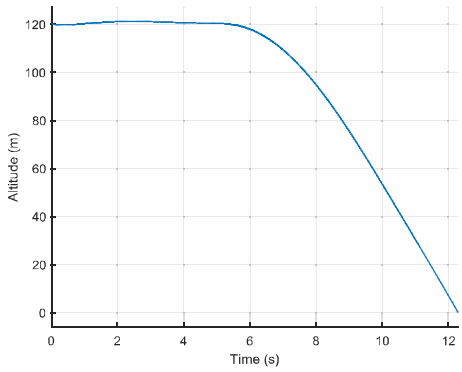

Figure 8 - Altitude vs. time for the neurocontroller (miss distance is 0.125 m)

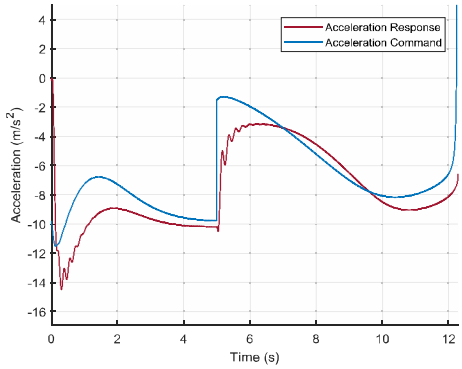

Figure 9 - Acceleration command & response for the neurocontroller

5. CONCLUSIONS AND DISCUSSONS

When the performances are investigated, it can be said that the neurocontroller is successful in the sense of mission completion. As expected, it holds the altitude for the first 5 seconds and then it leads the missile to the target with a miss distance of 0.125 meters. As a result it can be said that, a neurocontroller is capable of tracking acceleration commands and providing the dynamical stability of the missile during the flight.

However, there is a deficiency of the performance of the neurocontroller that causes to miss distance of the neurocontroller to be worse than the miss distance of the classical controller. It can be seen that, when there is a sharp change in the acceleration command, the acceleration response of the missile is oscillating. Although it converges to the desired value after a short time, this oscillatory behaviour is not desired for transient performance concerns. It is a disadvantage when it is compared to the classical controller. To solve this problem, the angular velocity in the pitch plane, q, is also added to the inputs of the neurocontroller. In that configuration, performance is improved but not enough to be considered as a solution.

Other than that, the sharp changes of the acceleration command can be filtered to prevent those oscillations but this is not an optimum solution to the problem.

6. FURTHER STUDIES

In the previous part of the study, it was shown that the network has the capability to replace the controller and handle the nonlinear flight dynamics of the missile. By observing this performance that meets the requirements, it can be said that, a neurocontroller which is adaptive to the airframe changes with online learning is a topic worth to study further.

In this study, two separate aerodynamic databases are created to model a situation that includes rupture of two wings. In this part, the simulation starts with the regular airframe properties. Then, in the middle of the flight, two wings of the missile break and the corresponding aerodynamic database is started to use. The aim of this part of the study is that, the neurocontroller is expected to learn the new airframe model and started to create actuator commands according to this new situation but it is not possible with the current structure of the network. It needs some modification to overcome the sudden nonlinear changes of the plant and update its parameters according to this new condition.

7. REFERENCES

[1] Harris, J., & Slegers, N. (2009). Performance of a Fire and Forget Anti-tank Missile with a Damaged Wing. *Mathematical and Computer Modelling*,292-305.

[2] Blake, W. B. (1998). Missile Datcom User's Manual - 1997 Fortran 90 Revision. *Wright-Patterson Air Force Base*.

[3] Efe, M. Ö. (2017). Artificial Neural Networks. *Lecture Notes*.

[4] Etkin, B. (2012). *Dynamics of Atmospheric Flight*. Dover Publications.

[5] Haykin, S. (1994). *Neural Networks - A Comprehensive Foundation*. New Jersey: Macmillan College Printing Company.