Ohanes Dadian, Subodh Bhandari, and Amar Raheja *Cal Poly, Pomona, CA, 91768*

[[1]](#footnote-1)

A Recurrent Neural Network for Nonlinear Control of a Fixed-Wing UAV

*Abstract* — Application of echo state network (ESN) for the nonlinear control of a fixed-wing unmanned aerial vehicle (UAV) is presented. The data required for the network training is generated using a validated flight dynamics model of the UAV. The ESN is used for both offline and online training. The offline training realizes the inversion required for the feedback linearlization. The online training is then used to reduce the inversion error that results due to the modeling deficiencies. The data required for online training is obtained in real-time from the FlightGear flight simulator while the UAV is flying in simulation. Results show that the ESN performs very well for the nonlinear control of the UAV with significantly less computational time and simplicity.

# Introduction

As UAVs grow in popularity, there is a need for flying in a wider flight envelope. This requires an increased level of autonomy, which in turn requires the vehicles to have nonlinear flight controllers to account for the changing flight conditions and nonlinearity due to other reasons.

The UAV sector is the fastest growing sector of the aerospace industry. UAV’s provide cheaper solution to dull, dirty, and dangerous missions. They do not require a pilot to be on board, and thus pose no threat to human operators. This opens up to the ability to perform many civilian and military missions at significantly reduced cost and danger. This includes search and rescue, wildfire detection, and battlefield damage assessment. With the ever increasing importance placed upon on unmanned missions, it has become more important for these vehicles to fly in wider flight envelopes. This requires the underlying flight controller to be nonlinear and exhibit highly adaptive behavior in order to stay stable in changing environments. Many techniques have been used for the design on nonlinear flight control system. Most popular method is dynamics inversion [1], [2]. However, dynamic inversion requires accurate knowledge of flight dynamics and any error in the model can cause instability [3], [4]. Recently, much attention has been given to the application of neural networks for flight control, due to their ability to approximate any nonlinear function to a high degree using input/output pair collected during flight tests. Many work exist on the application of neural networks to flight control [5-7]. Most of these work, however, use feedforward network such as multi-layer perceptrons (MLP) and radial-basis functions (RBF) networks [8], [5], [9]. However, to the authors’ knowledge, no work exist on the application of recurrent neural network to the design of flight controllers. The distinct advantages of recurrent neural networks are faster processing time and simplicity. This paper presents the application of echo state network (ESN), a class of recurrent neural network, to the design of flight controllers for a fixed-wing UAV. ESN can be trained by methods such as recursive least-squares in a one-shot fashion, without requiring back-propagation and recursive passes through the training set [10]. The overall goal of the ongoing research at Cal Poly Pomona is the development, validation in simulation, and flight testing of the neural network based adaptive controllers for UAV’s. The project is funded by National Science Foundation.

The paper is organized as follows. Second section talks about the twin-engine UAV that is being used for this research and the avionics system that is under development for the implementation of neural network based controllers. Nonlinear flight dynamics and feedback linearlization is discussed in Section III along with the validated nonlinear flight dynamics model for the airplane. Section IV talks about the neural networks, and how it is used for the nonlinear control. Application of ESN for the identification and control of the twin-engine UAV is presented in Section V, followed by conclusion in the last section.

# aerial vehicle platform

The airplane being used for this research is a twin-engine airplane from Unmanned Systems Incorporated (USI). Figure 1 shows the airplane in flight. The DA 50 gasoline engine-powered high-wing airplane has wing span of 134 inches and is 95 inches long. The empty weight of the airplane is 42 lbs. with 25 lbs. of payload capacity. The airplane is currently equipped with Piccolo II autopilot that the Department of Aerospace Engineering at Cal Poly Pomona has been using for autonomous flights [11].The airplane has been flight tested for data collection. The collected flight data was used for the development and validation of linearized and nonlinear flight dynamics model for the vehicle. The airplane will be equipped with the custom avionics system [12] for the implementation of the controllers being presented in this paper and controllers designed using other techniques.



## Fig. 1. Twin-engine UAV

## Avionics Architecture

At the heart of the twin engine aircraft is our open avionics architecture [12]. For the custom flight controllers such as those based neural networks, a custom and open source avionics system is required. Most of the commercial off-the-shelf avionics systems are closed source [11]. The avionics system is designed to meet the requirements of various research projects related to flight controls. These projects require autonomous waypoint navigation, state estimation, wireless data links, communication with operator control units, and payload support. The avionics system supports an open controller architecture for changes to the Guidance, Navigation, and Control (GN&C).

# aircraft nonlinear equations

The dynamics model for aircraft is derived using the nonlinear equations of motion that describe the input-output relationships for the aircraft. For the rigid body dynamics of the airplane, there are three force and three moment equations as given below [13], [14].

 (1)

 (2)

 (3)

 (4)

 (5)

 (6)

where *c1* through *c8*are the functions of moments of inertias [15]. In the above equations, *u*, *v*, and *w* are the translational velocities, *p*, *q*, and *r* are the angular rates, *L*, *M*, and *N*, and *Fx*, *Fy*, and *Fz* are the three moments and three forces respectively, *m* is the aircraft mass, *ϕ* is the roll attitude angle *θ* is the pitch attitude angle, g is the acceleration due to gravity.

These equations with some reformulation can be written as the functions of angle of attack (*α*), sideslip angle (*β*), translation velocities, angular rates, and control surface deflection angles: *δa* (aileron), *δe* (elevator), and *δr* (rudder). For example, the roll acceleration equation can be written as:

 (7)

where M is the Mach number. In general, above equation can written as

The general nonlinear equation can be written in the following form:

 (8)

For the aircraft problem, *x* is the state vector and *δ* is the control vector. The required suitable input can be estimated by inverting the above using dynamic inversion technique as given below

 (9)

where *x*d is the desired dynamics.

There are three several disadvantages of dynamic inversion [3], [5]. Due to uncertainty in the aerodynamic model, there will be an error in the inversion process. The number of states and controls should be identical, otherwise dynamic inversion cannot be used, as the g matrix must be square so as to be invertible. Dynamic inversion cannot be applied to non-minimum phase system. Due to this limitation a neural network approach can be applied [16] as discussed below to realize this inversion, i.e., the network can be used to represent nonlinear transformation for feedback linearization, eliminating the need for dynamic inversion. The neural network uses the input/output pairs for the training. The data required for the training is generated from a validated flight dynamics model for the airplane.

## Aircraft Nonlinear Flight Dynamics Model

A nonlinear flight dynamics model for the twin-engine airplane was developed using Athena Vortex Lattice (AVL) software [17] as well as the System Identification Program for Aircraft (SIDPAC) software [18]. SIDPAC software uses regression and maximum likelihood methods for the determinations of model parameters [18]. The identified model was verified using flight data. The airplane was flown for doublet inputs in the aileron, elevator, and rudder. Figure 2 shows the comparison of the flight data with the simulation results for roll and yaw rates. It is seen that the model response matches well with the flight data. There are some discrepancies. Further refinement of the flight dynamics model is underway.

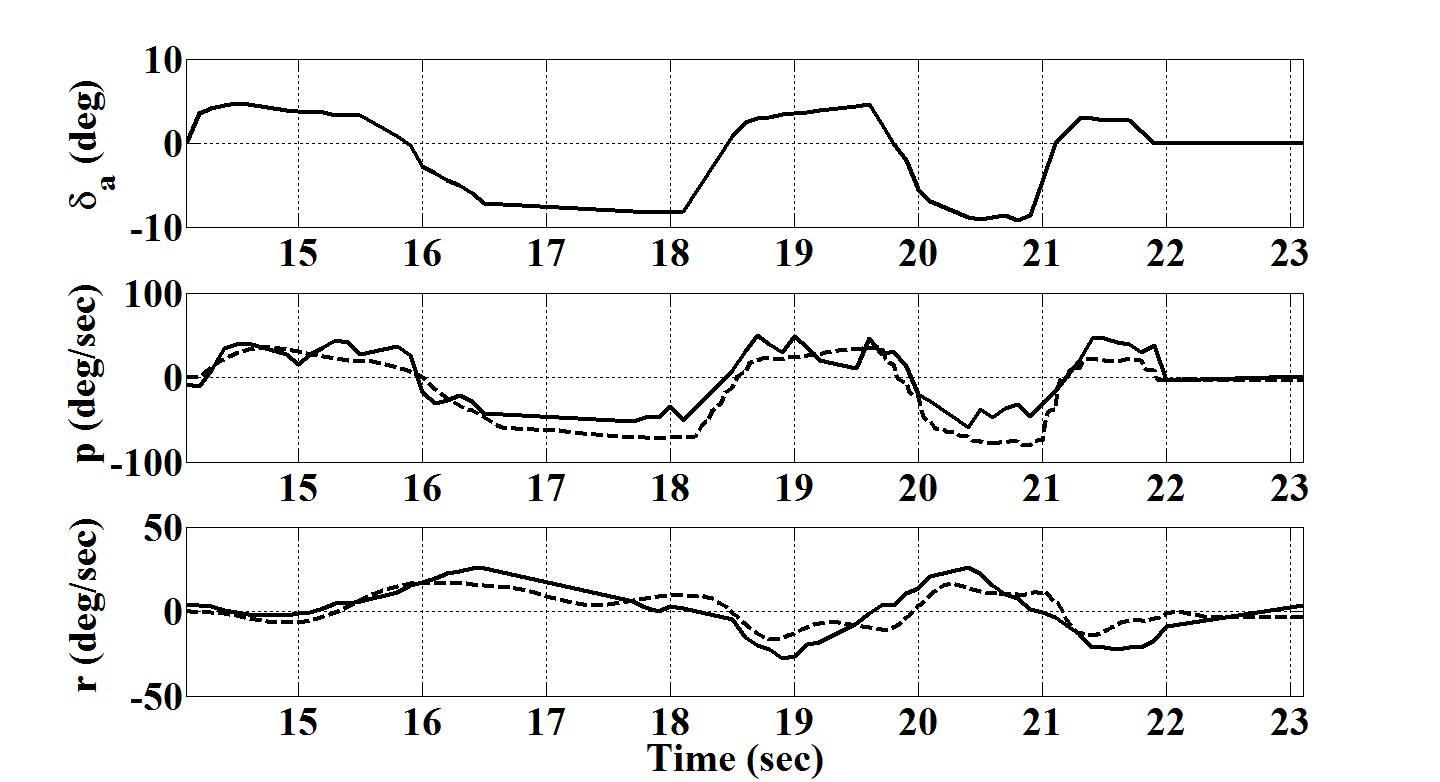


Fig. 2. Flight data (solid) vs. model response for roll and yaw rates

Similar results were obtained for pitch responses as shown in Figure 3.

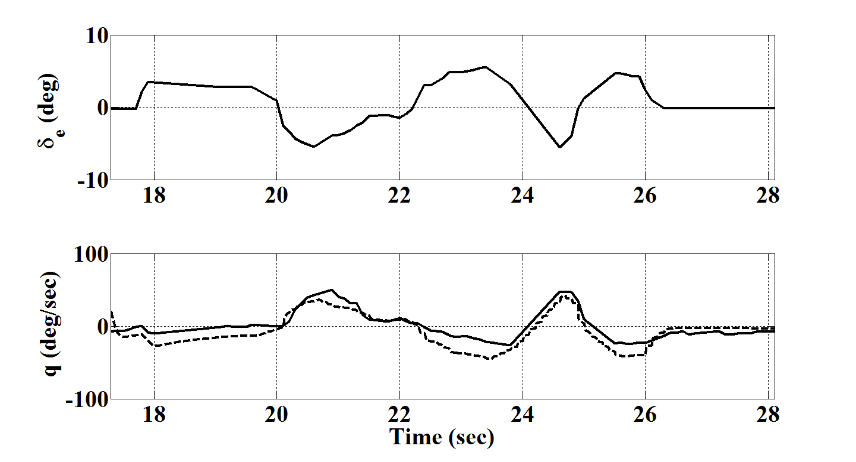


Fig. 3. Flight data (solid) vs. simulation results for pitch rate

Using the validated flight dynamics model, a model for the airplane was created in the FlightGear Flight Simulator as shown in the figure below.



Fig. 4. Twin-engine model in FlightGear simulator

As discussed below, the data required for the neural network training was generated using the FlightGear model of the airplane.

# nonlinear control using neural networks

An artificial neural network is a non-linear and parallel computational model that contains interconnected neurons which can perform computations significantly faster than that of linearly programmed computer. The neurons are interconnected, forming a directed graph. Modeled after the mammalian brain, the network builds up knowledge of its environment through training, as it is parametric by design to no a-priori knowledge. Current knowledge is weighted in the form of signal strength between the neurons, providing critical path to desired output. This process is accomplished through a learning algorithm.

ANNs can approximate virtually any function with accuracy based on the size of the available sample data, without a priori assumptions of the specific functional form [19]. The functional form is derived directly from the example data, i.e. through learning. Literature has shown that neural networks function as highly nonlinear dynamical systems and offer distinct advantages over conventional linear controllers in terms of desired performance [19]. There are many different classifications of neural networks. Each classification is geared toward solving certain problems such as prediction, pattern recognition, image analysis, and adaptive control [20].

The most popular among these classifications is the feedforward network. This model encapsulates neurons within a series of layers. This includes an input, hidden, and output layers. Both the input and output layer are linear, while the hidden layers are nonlinear. A model may have single to multiple hidden layers, depending on the design. The model is unidirectional with the input layer flowing through the output layer. Hence, the feedforward moniker.

The neurons in the hidden layers are activated using an activation function, usually a logistic sigmoid. Its nonlinear behavior allows it to perform high-order approximations within a close delta to the desired result. The neurons in the output layer exhibit the overall response to the network by the activations which occur in the hidden layer.

The output signal generated from neuron activation is used in adjusting the connection weights between the neurons. This effectively builds a least cost path to the desired output in the model; in turn generating a properly fitted model [21]. This is desirable for many applications including handwriting, target recognition, and flight control. Performing machine learning for aircraft control would be beneficial for the complete control of the aircraft in the entire flight envelope. However, as discussed above, this paper discusses the application of recurrent neural networks for nonlinear flights control. The following section discusses why this model is a choice candidate for such a problem.

## Recurrent Neural Network

Recurrent neural networks (RNN) differ from feed-forward networks through the inclusion of a feedback loop. This gives the model an inherently directed cycle between for the series of neurons. This brings the behavior of the system closer to that of the mammalian brain, as it relies heavily on feedback. The change offers nonlinear dynamic and temporal behavior through the addition of time delay units feeding into the input signals from the output neurons, as shown in Figure 5. This in turn provides compatibility with classical control laws.

The design allows for improved training capability, as the multiplexed result of the output signal and input signal shift the internal neurons closer to the designed teacher signal. This also results in continuous excitation rather than discrete, as seen in feedforward models.

The output signal is fed back into the network, exciting the neurons for the generation of the next state. There has been much interest in using RNNs for technical applications where it is necessary to map sequences of input and output pairs, with or without a teacher. Many research efforts are averse to using RNNs due to their complex behavior, difficult implementation, and high computational requirement. Despite this fact, they tend to have large dynamical memory and highly adaptable computational capabilities. This tends to provide flight control researchers with a bit of a paradox in term of employing them. One model of this class, echo state network, stands out to rid us of this problem [10].

In addition to this, it provides the ability to store current knowledge by way of associative memories. In turn, this provides a means for backpropagation over time using weights throughout all epochs. This provides a complete temporal pattern for deciding connection weights.

The memory in the hidden layer may be modeled as a state space. The neurons make up the state, charged by the time delay propagation of the output neurons. A time delay induced network can be seen in Figure 5. The activation function *a(t)* is computed using the weight matrix w, input signal *p(t)*, and scaling coefficients *I* and *i*. The input signal, shifted by coefficient i, is multiplexed with the current knowledge shifted by coefficient I.

Effects of the environment are applied to the model through the input sources (signals). Order is determined by the number of time-delay units.

Training may be completed using supervised or reinforcement learning. With supervised learning, with the teacher signal being fed into the input signal used in activation of the hidden neurons. Reinforcement learning, on the other hand provides a fitness function instead of a teacher signal.

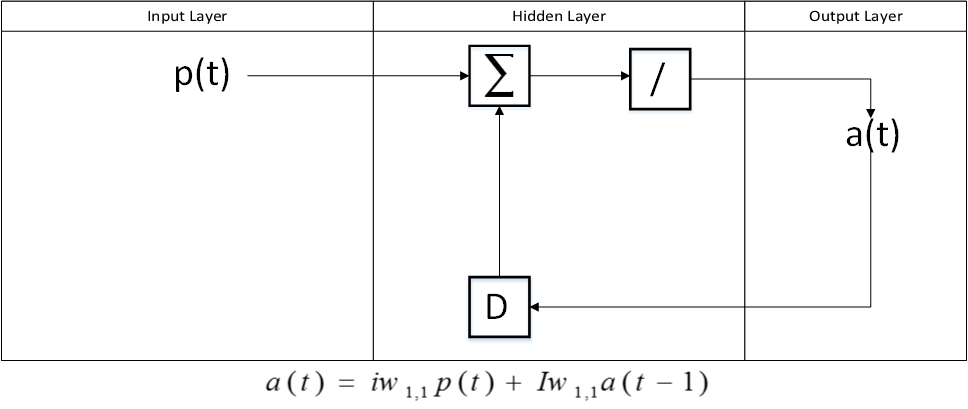


Fig. 5. Recurrent Neural Network with Time Delay

## Echo State Network

ining availableThere’s theH for training[24]is quite large as compared to our current multilayer perceptron.

Elman network, which has a three layer design and weighted context units in the middle layer to provide better signal adjustment. However, the context units may lead to bloat in the nonlinear state space.

After much investigation, the choice became clear to us. We adopted to move forward with the Echo State Network (ESN) for the purposes of our research and development. The ESN, developed by Herbert Jaeger at Jacobs University in 2004, provided a novel reservoir based approach combined with supervised learning. The offered an interesting take on neural computing which we haven’t seen before. Very practical and computationally inexpensive by design, ESNs provides a means to generate signals based on current knowledge throughout the model’s history. Utilizing a leak rate for this current knowledge in the hidden layer. This behavior in the hidden layer is the reason it has been coined reservoir computing. At the heart of its simplistic design is the sparse connectivity between its neurons. This allows for a leaner model and less complex computations in generating the output signal. These design choices provide an overall architecture that is simpler and easier to implement than the aforementioned models.[23]

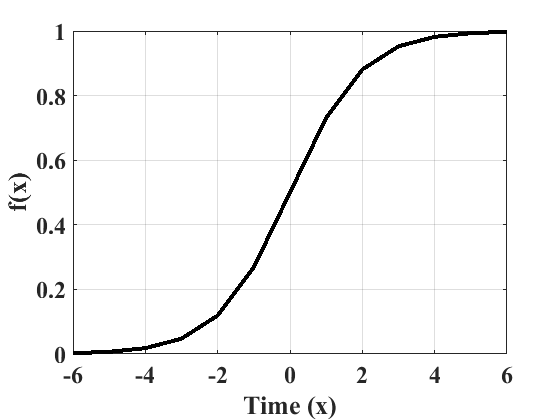


Fig. 6. Logistic Sigmoid

The hidden layer was deemed a reservoir due to the behavior of the current knowledge leaking over time. As with the other recurrent models, the output neurons are driven back into the model. The ESN activates its neurons using a hyperbolic tangent function given below [21]:

 (10)

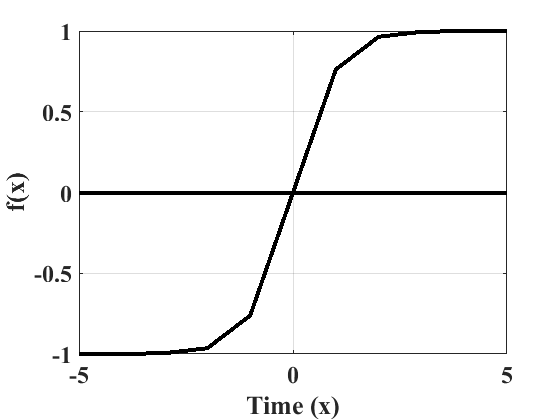


Fig. 7. Hyperbolic tangent

The ESN then performs Wiener-Hopf based linear regression (equations 10-12) during the read-out phase of the output. The output signals are compared against the teacher to determine if the model is fitted or if further training is necessary.

(11)

where *f* consists of hyperbolic tangent function, *Win* is the weight of the network input *u*(*n*), *Wfb*is output feedback matrix, *x*(*n*) is the reservoir state, and *y*(*n*) is the network output. The extended system state that is a concatenation of input and reservoir states is given by

(12)

The output of the network is then obtained as given in the following equation:

(13)

where *g* is an out output activation function and *Wout* is the matrix of output weights, which is given by

(14)

Where,

(15)

and

where *d*(*n*) is the desired outputs.

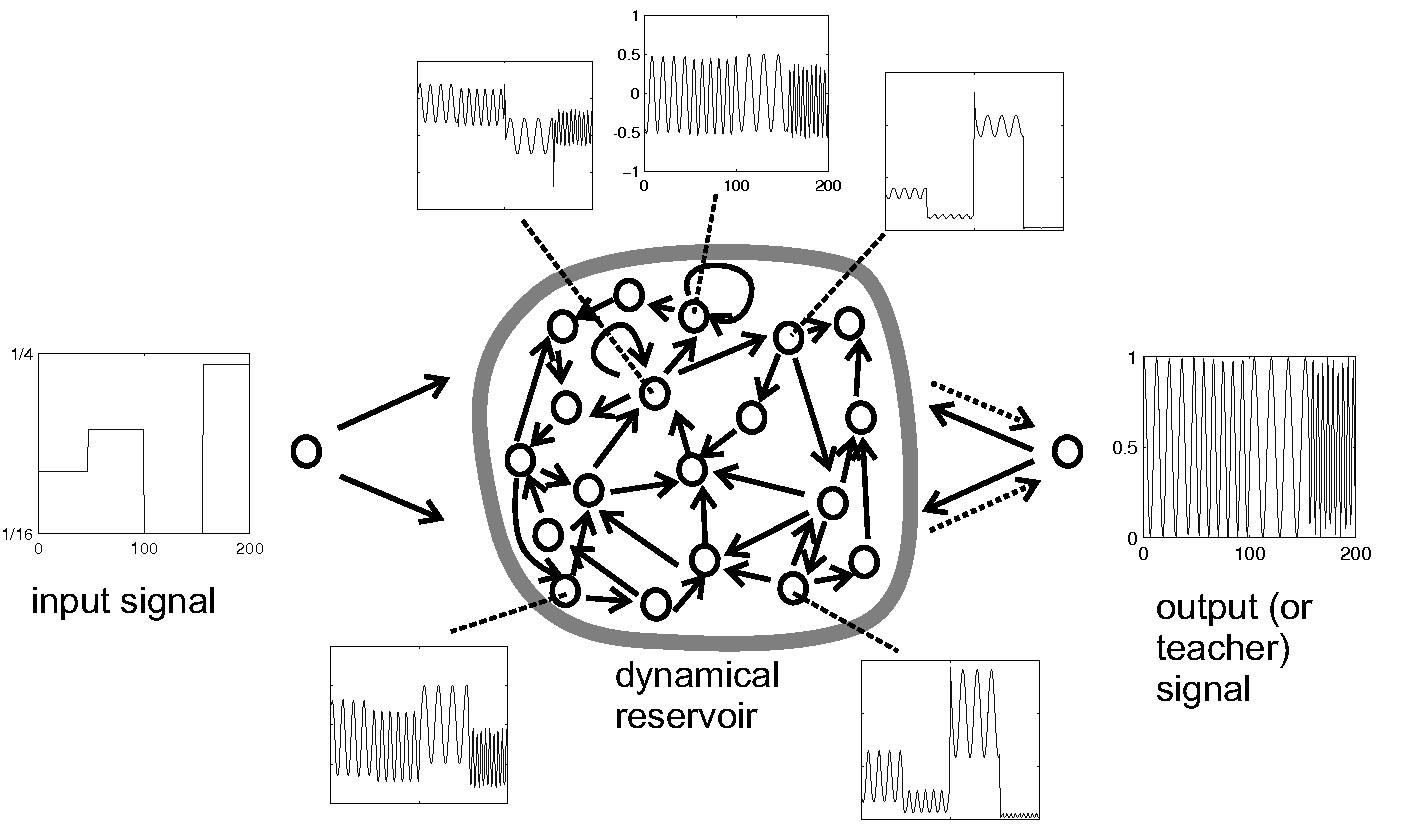


Fig. 8. Echo state network

## As the training data is fed into the echo state network, the network activates and harvests the neuron states in its reservoir. The neurons are used to generate the output signal, which is then compared to the target. The target, as mentioned previously, is used to train the model’s output signal. One instance of training comprised of driving the network with roll acceleration and emits aileron as output.

For performing Proportional-Integral-Derivative (PID) tuning, we have to model the adaptive behavior necessary for performing the required control techniques.

(17)

Function u(t) describes the control structure. Parameter upd defines the Proportional-Derivative, while uad defines the output signal from the ESN which provides adaptive control. The Proportional-Derivative is used to model system response, while adaptive control is used to perform error compensation.

upd is described as the following.

(18)

The x variables are the commanded position and velocity. While the k variables are the proportional and derivative control gains. Error handling is taken care of by the following formula.

(19)

*e* (20)

(21)

(22)

The delta of error e is what determines the strength of output signal. The reference term for the model’s error delta can be determined by the following.

(23)

P is a 2x2 matrix defining the stability behavior derived from the Lyapunov equation. More elaboration is given in section G.

## Error Computation

To determine whether the output signal of the network is close enough to the target signal, a delta of the two signals needs to be calculated [10]. To achieve this, root mean square formula as given below is used.

(24)

This acts as the global rate of error for the entire network, and allows the determination of whether the model is properly fitted and whether or not to continue training.

# identification and control of twin-engine airplane using neural networks

Both offline and online training for the twin-engine airplane was performed. The data required for the training was obtained from the FlightGear model of the airplane is based on the validated nonlinear dynamics of the airplane as discussed above. The simulated data instead of real flight data was used because of the fact that the network training requires a large set of data for effective and accurate.

The generated data is separated into two categories, training and validation. The training data was used to generate the input signal, while the validation data was used to generate the teacher signal. These signals include but not limited to roll rate/acceleration, aileron deflection, sideslip angle, and yaw rate.

The prototype was modeled in MATLAB version of ESN within the Simulink environment, while the experimental model was developed in C++. The C++ implementation supports Unix-based operating systems and Windows. It is currently being integrated into to the avionics system as an application running on a QNX Neutrino partition. The matrix algebra is handled by the Eigen2.0 C++ library. The extensibility of this library allows for a well-defined hierarchy of the matrix- and vector-based behavior of signals and weight matrices in the network

Both offline and online training were performed. The offline training realizes the inversion required for the feedback linearization [5], [16]. The online training is then performed to iron out modeling errors. Both training methods use Weiner-Hopf method for linear regression. This allows for generation of the output weights, which is then used to generate the output signal. The output signal is fed back into the network as a means of learning. The feedback loop eliminates the requirement of backpropagation of weights [16], as the output signal is used in calculating the next current state.

## Offline Training

Offline training utilizes the data generated from the FlightGear model of the aircraft. The Training continues until the output signal comes with an acceptable delta with the teacher; 0.01%. At this point, it is properly fitted and ready for online training. Figure 10 shows the implementation of the offline trained ESN model.

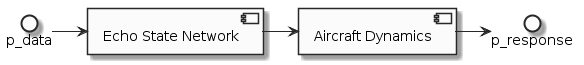


Fig. 9. Implementation of offline ESN model

In order to train our neural network to perform flight control, we require flight data to generate the input signal. For the purposes of this research, we decided upon using simulated data gathered from our flight dynamics model of the TwinEngine aircraft running in FlightGear. Results from aileron deflection was used.

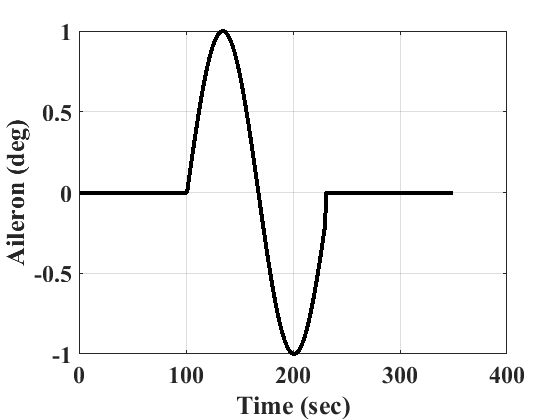


Fig. 10. Sinusoidal Input of Aileron Deflection

## Offline Training

In order to ensure that the neural network is flight ready, the model must be properly fitted with accurate flight data. Flight parameter read outs were recorded from flight tests and simulations. Flight tests were conducted with the Twin Engine aircraft, while simulations were done with the dynamics model of the same aircraft.

The data was formulated as a sinusoidal input, as shown in Figure 10.

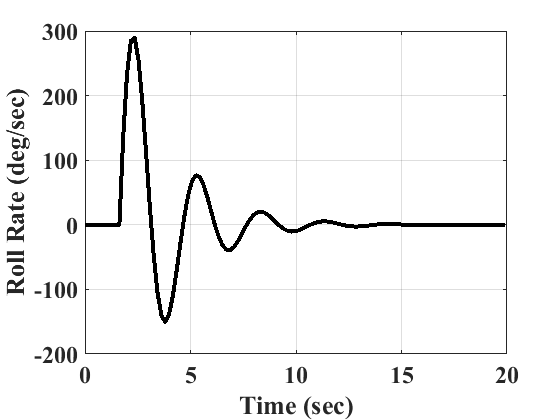


Fig. 11. Roll Rate Output

## Online Training

For the online training process, the data used was the real-time data from the FlightGear simulator while the airplane was flying in simulation. The data was transmitted to the ESN from the FlightGear simulator using the user datagram protocol (UDP). This was also useful in testing the controller performance in software-in-the-loop simulation. Figure 12 shows the implementation of online ESN network.

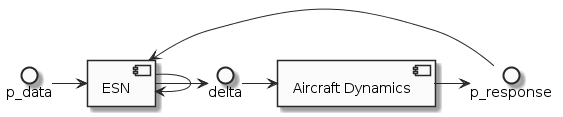


Fig. 12. Implementation of online ESN model

Figure 13 shows the response of offline trained ESN compared to the simulated flight data for roll rate. It can be seen that there is excellent correlation between the two. At the beginning, there are some discrepancies, but the error decreases as the time increases.

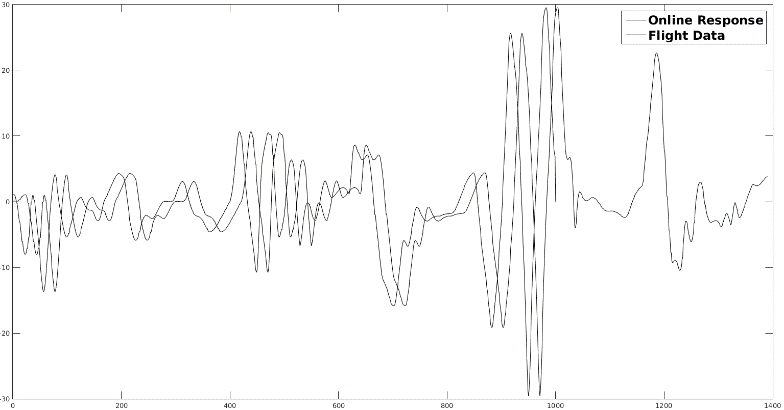


Fig. 13. ESN offline model response vs. flight data (roll rate)

Figure 14 shows the response of online trained ESN compared to the simulated flight data as well as the response of the offline trained ESN for roll rate. It can be seen that there is excellent correlation between the two. The error between two has largely disappeared.

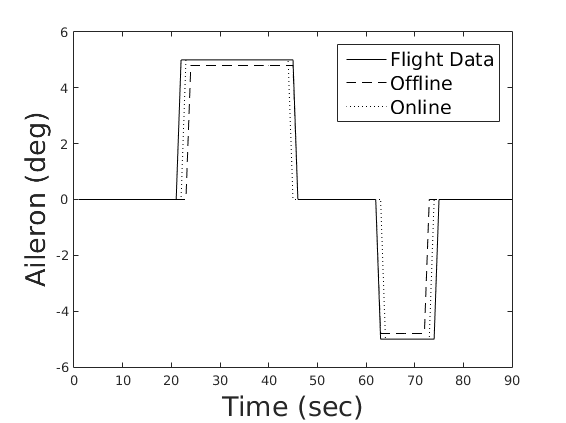


Fig. 14. ESN model responses vs. flight data (roll rate)

## Implementation of the Trained Networks in Closed-Loop

Figure 15 shows the implementation of the trained networks in a closed-loop. The figure also shows proportional-integral-derivavtive (PID) controller in the loop. The PID controller is used to shape the closed-loop response as well as to guarantee the stability.

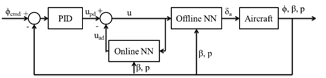


Fig. 15. Implementation of neural networks in closed-loop

Figure 16 shows the closed-loop response of the airplane to a unit step command in bank angle for both offline only and offline plus the online networks. It is seen that the closed-loop response is superior with both the networks compared to the response with only the offline network.

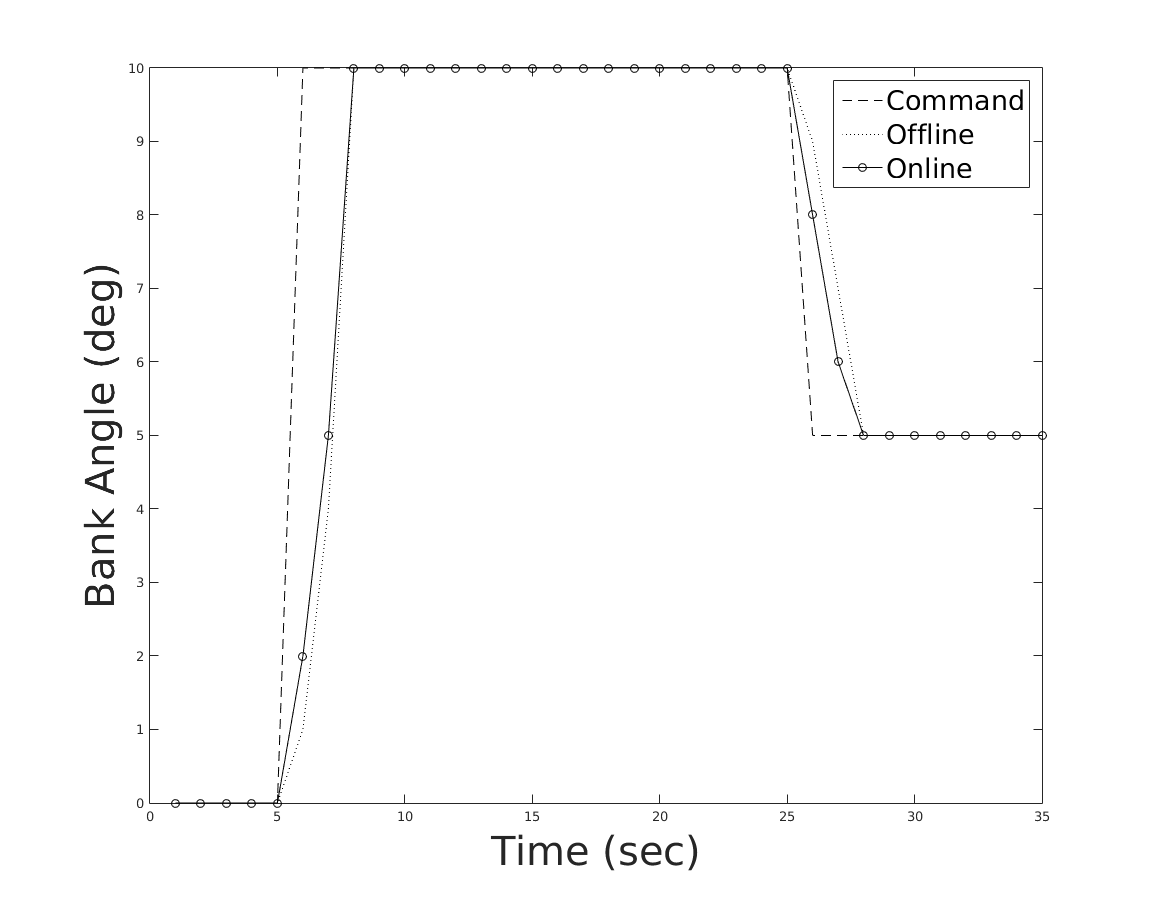


Fig. 16. Closed-loop response for bank angle

## Lyapunov Stability

One way to determine that the controller designed above will guarantee stability is to use Lapunov’s stability theorem [5]. It is a means to provide analysis of asymptotic stability for nonlinear systems. Stability is determined by how close the equations are to the point of equilibrium. If the equations start out at a point of equilibrium, , stay near then is lyapunov stable. In order to apply the Lyapunov’s theorem, the above equations will need to be reformulated. Lyuponov’ Direct Method can be utilized in order to derive the candidate Lyuponov function.

The following formula describes an autonomous nonlinear system, much like our flight controller, , where denotes a vector of the system state. D is a set containing the origin. There is a function f where is continuous on D. If f has equilibrium against , as just described, then . With that being the case, the following assertions are proposed to be true.

1. If for every , there exists, a , such that if , then for every there is .
2. The equilibrium of this system is asymptotically stable if it is lyapunov stable and there exists and , then .
3. The equilibrium system is exponentially stable if it exponentially stable and there exists , such that if , then , for all .
4. The trajectory of x is attractive if for that are close enough and attractrive.

Once the candidate Lyapunov function is determined, that can be utilized to assess the self-stabilization of the controllers designed above. This is done through witnessing global asymptotic behavior during minima. This would in turn prove little to no bursting in the model response.

The Lyapunov function is determined using the model’s reference error. This is due to the fact that the error term is used to determine the fitness of the model during training. Equation x is used to find the reference error. It may be reformulated as the following equation.

(25)

A is what is called a Hurwitz Matrix which models stability. Reformulating equation 24 as the following provides us with a Lyapunov solution.

(26)

# conclusion

The paper showed that a recurrent neural network can successfully be used for the identification and nonlinear control of flight vehicles. Echo state network was used for both offline and online training using the input/output pairs for the twin-engine UAV. The offline training realizes the inversion required for feedback linearization. The data required for the offline training was generated using the FlightGear model of the aircraft that is based on a validated nonlinear dynamics model of the aircraft. The online training is then performed to reduce the inversion errors. Simulation results show that with the inclusion of online trained network, the performance of the controller significantly improves for both open-loop and closed-loop responses. Because of the recurrent behavior, the ESN facilitates adaptability with more ease than a feedforward network the ESN model provides shorter bursts while staying Lyapunov stable. It was also found that the error due to offline training was less compared to other networks. The final paper will show more results including the controllers for pitch and yaw-axes. The final paper will also show the proof of Lyapunov stability for the controllers.

Future work will involve testing the ESN model in Hardware-in-the-Loop (HIL) simulation. If found satisfactory in these simulations, the controller will be tested in flight. The data required for the online training will be actual flight data. Also, the controllers will include adaptive elements for increased robustness in the presence of modeling errors, disturbance, and noise.

Acknowledgment

This research was supported by National Science Foundation under the Award No. 1102382. The authors would like to thank Hovig Yaralian and Luis Andrade for their help with flight testing, data collection, and development and validation of the nonlinear flight dynamics model of the UAV. The authors would also like to thank Mr. Jim Cesari, the Aerospace Engineering Department technician, for his help with flight testing of the UAV.

references

1. D. J. Bugajski, et al., “A Dynamic Inversion Based Control Law with Application to the High Angle of Attack Research Vehicle,” *Proceedings of AIAA Guidance, Navigation, and Control Conference,* 1990.
2. F. Holzapfel and G. Sachs, “Dynamic Inversion Based Control Concept with Application to an Unmanned Aerial Vehicle,” *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, RI, Aug. 16-19, 2004.
3. B. Mroton, et al., “Stability of Dynamic Inversion Control Laws Applied to Non-linear Aircraft Pitch-Axis Models”, *Institute for Mathematics and its Applications*, University of Minnesota.
4. J. S. Brinker and K. A. Wise, “Stability and Flying Qualities Robustness of a Dynamic Inversion Aircraft Control Law,” *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 6, 1996.
5. B. S. Kim and A. J. Kalise, “Non-linear Flight Control Using Neural Networks,” AIAA Journal of Guidance, Control, and Dynamics, vol. 20, no. 1, pp. 26-33, 1997.
6. M. R. Napolitano and M. Kincheloe, “On-line Learning Neural-Network Controllers for Autopilot Systems,” Journal of Guidance, Control, and Dynamics, vol. 33, no. 6, Nov.-Dec., 1995.
7. D. Shin and Y. Kim, “Reconfigurable Flight Control System Design using Adaptive Neural Networks,” IEEE Transactions on Control Systems Technology, vol. 12, no. 1, pp. 87-100, 2004.
8. S. Bhandari, A. Raheja, et al., “Nonlinear Control of UAVs using Multi-Layer Perceptrons with Off-Line and On-Line Learning,” Proceedings of *American Control Conference*, Portland, OR, 4-6 June 2014.
9. C. M. Ha, “Neural Networks Approach to AIAA Aircraft Control Design Challenge,” *Journal of Control, Guidance, and Dynamics*, Vol. 18, No. 4, July-August 1995.
10. D. Prokhorov, “Echo State Networks: Appeal and Challenges,” *IEEE International Joint Conference on Neural Networks*, Montreal, Canada, 31 July-4 Aug. 2005.
11. N. Anderson, S. Bhandari, et al., “Flight-Testing of a UAV Aircraft for Autonomous Operation using Piccolo II Autopilot”, AIAA Atmospheric Flight Mechanics Conference, Honolulu, HI, Aug. 18-21, 2008.
12. S. Bhandari, O. Dadian, A. Bettadapura, et al., “Avionics System for UAV Flight Controls Research,” AIAA Infotech@Aerospace Conference, Boston, MA, 19-22 Aug. 2013.
13. J. Roskam, Airplane Flight Dynamics and Automatic Flight Controls, Kansas: DAR Corporation, 1998.
14. R. C. Nelson, Flight Stability and Automatic Control, Boston: McGraw Hill Higher Education, 1998.
15. M. Sadraey and R. Colgren, “Robust Non-linear Controller Design for a Complete UAV Mission,” AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO, Aug. 21-24, 2006.
16. J. F. Haffner, N. T. Meyrer, et al., “A Multi-layer Perceptron Replaces a Feedback Linearization Controller in a Non-linear Servomechanism,” IEEE International Joint Conference on Neural Networks, Anchorage, AK, May, 1998.
17. E. Morelli., “System IDentification Programs for AirCraft (SIDPAC),” *AIAA Atmospheric Flight Conference Proceedings*, Monterey, CA, Aug. 5-8, 2002.
18. M. Rose, S. Bhandari, et al, “Development and Validation of Flight Dynamics Model of a UAV Airplane,” AIAA Infotech@Aerospace 2012, Garden Grove, CA, June 19-21, 2012.
19. K. S. Narendra and K. Parthasarathy, “Identification and Control of Dynamical Systems Using Neural Networks,” IEEE Transactions on Neural Networks, vol. 1, no. 1, pp. 4–27, 1990.
20. T. Lee and Y. Kim, “Non-linear Adaptive Flight Control using Backstepping and Neural Networks Controller,” Journal of Guidance, Control, and Dynamics, vol. 24, no. 4, July-Aug. 2001.
21. S. Bhandari, A. Raheja, D. Tang, et al., “Modeling and Control of UAVs using Neural Networks,” AIAA Modeling and Simulation Technologies Conference, Minneapolis, MN, Aug. 13-16, 2012.
22. Haykin, S., “Neural Networks and Learning Machines”, Upper Saddle River, New Jersey: Pearson, 2009.
23. Maass, T. Natschläger, and H. Markram. “A fresh look at real-time computation in generic recurrent neural circuits.” Technical report, Institute for Theoretical Computer Science, TU Graz, 2002
24. Rául Rojas (1996). “Neural networks: a systematic introduction. Springer”. p. 336.
25. Forssell, Lindskog, Combining Semi-Physical and Neural Network Modeling: An Example of Its Usefulness
26. Araki, M. "PID Control"
27. Lukosevicious, Mantas. “A Practical Guide to Applying Echo State Networks”, Jacobs University, 2012.

1. Graduate Researcher, Department of Computer Science, oadadian@cpp.edu.

   Professor, Department of Aerospace Engineering, AIAA Senior Member, sbhandari@cpp.edu.

   Professor, Department of Computer Science, raheja@cpp.edu. [↑](#footnote-ref-1)