Deep Residual Recurrent Neural Network for Aircraft Dynamics Simulation

Sriram Ranganathan
Electrical and Computer Engineering
University of Waterloo
sriram.ranganathan@uwaterloo.ca

***Abstract*—Capturing an aircraft system to model and simulate the flight trajectory is a profoundly intricate endeavour, owing to its higher-order complexity and inaccurate results. It is very much crucial for the safety of the aircraft and to control it properly. In the paper, the implementation of the physics-based Deep Residual Recurrent Neural Network (DR-RNN) for aircraft dynamics simulation. DR-RNN involves incorporating the physics of the aircraft system of interest with an unravelled recurrent neural network in the form of residues. The use of the DR-RNN is demonstrated by simulating the Boeing 747-100 aircraft's longitudinal trajectory. The results of training the DR-RNN and testing its extrapolating behaviour show that model's prediction is different from the ground truth values. Although the predicted results are different concerning the magnitude and phase observed in the trajectory, further fine-tuning of the hyper-parameters and the training will yield more accurately predicted trajectories. This paper culminates the results of various architectures of the Deep Residual Recurrent Neural Network extended from the foundation architecture mentioned in the previous literature. Also, this report includes a comprehensive study of the physics-informed model for aircraft dynamics simulation literature.**

1. Introduction

Flight simulation is a vital task in the field of aerospace and engineering. It aids the aircraft modelling engineers in designing and testing the design by being able to examine the characteristics of the flight throughout the envelope. It also aids the design changes by analyzing the stability of the aircraft under various trim conditions. Simulations are cost-friendly and environment-friendly ways to accurately test the impact of any new design modifications in the plane by allowing the design engineers and scientists to test virtually. A simulator aims to create a digital twin of the aircraft by inheriting the natural physics, impacts of environmental changes, and the control inputs the pilots provide. Furthermore, Flight simulations are used to assess the control and stability characteristics of the aircraft under different conditions. The simulators that offer faster and more accurate predictions can aid pilots in emergencies to oversee the flight conditions beforehand.

Developing fast and accurate simulators that predict the next state of the aircraft is a very complex task since many variables and control inputs govern it. There is a trade-off between developing an accurate and computationally efficient simulator. Hence model order reduction methods are used to create a simulator that is more computationally efficient and accurate [1]. There are two broad and main categories of model order reduction methods 1. Physics-based methods, and 2. Data-driven methods. Although data-driven strategies can perform well in interpolating, for extrapolating, the simulator needs to understand the physics that governs the aircraft's trajectory [1]. The data-driven models will not generalize throughout the entire envelope because the prediction results of the data-driven models are always in the domain of the training data. Also, including the entire flight envelope in training data is nearly impossible.

The physics-based models are preferred because the data-driven methods can become unstable when predicting the flight trajectory for a longer duration because of the lack of physics involved. Combining the physics of the aircraft system with a data-driven approach can help generalize the simulators throughout the flight envelope. In this paper, we try to implement a Deep Recurrent Residual Neural Network model with

different architectures for simulating Boeing 747-100 aircraft's states over a longer duration by incorporating the physics of the plane in the form of residues in the neural network.

2. Literature Review

In [2], authors introduced DR-RNN to efficiently reduce the complexity of non-linear dynamical systems. It is inspired by the line search methods which iteratively reduces the residue. In [2], authors have solved many numerical problems pertaining to dynamical systems and obtained highly accurate results.

There is previously existing literature that studies and reports on physics-informed neural networks for aircraft dynamics simulation. In [1] authors have proposed a novel physics-based architecture called deep residual recurrent neural network which performs next-state predictions. The authors of [1] have considered both longitudinal and latitudinal states with control input but only around trim conditions. The underlying architecture of the model incorporates physics using a residual function. The model produced results that closely follow the ground truth data generated by solving the latitudinal and longitudinal state space models using the RK-4 method. Although it produced better results around one trim condition, the model has to be tested in different conditions.

In [3], The authors present FlyNet, a physics-based nonlinear autoregressive exogenous neural network model architecture, as a new approach to aircraft modelling. Traditional output error methods capture aircraft models around a specific point in the flight envelope using a Taylor series approximation of the forces and moments of the aircraft. Flynet makes use of the neural network to model the non-linearity of the aircraft. The main contributions of the paper [3] are as follows:

- Introduction of autoregressive flynet across the flight envelope
- Demonstrating the behaviour of flynet by testing it out on the NRC's Bell 412HP data
- Comparison of the flynet model with the conventional model

It was observed that the flynet model outperformed the conventional model by showing better stability characteristics. However, the authors also mention that the flynet has to be tested out on multiple aircraft to better understand the generalizability.

In [4], physics-guided machine learning is introduced which mixes data-driven machine learning models with physics-based models. This integration enables the use of machine learning's data discovery capability while keeping valuable physics/domain expertise. The model incorporates RNN and MLP including the physics of the dynamical system. The proposed method produces models that are highly generalizable and capable of producing highly accurate results. The underlying physics of structural dynamics is embedded in the RNN architecture to guide the learning and prediction of structural dynamical responses, while the data-driven MLP can handle the unknown physical relationship. [4]

3. Aircraft Modelling (Boeing 747 - 100)

In this section, The aircraft force and moment equations are briefly reviewed. The aircraft state space equations around a stable condition are studied, which is used in further developing the DR-RNN model architecture.

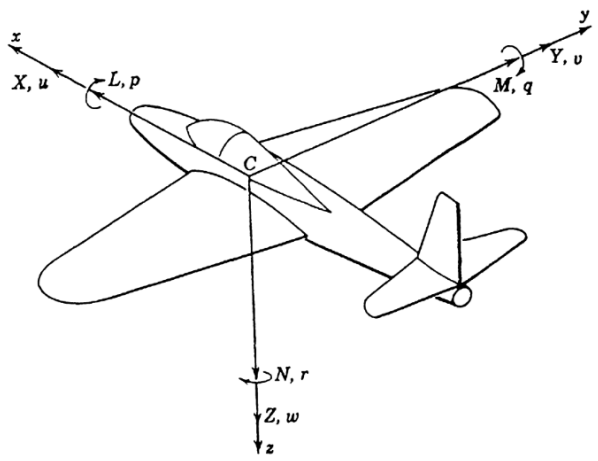

Figure 1. 6 degrees of freedom aircraft model [5]

Figure 1 shows the six degrees of freedom aircraft model. The equations of motion of the aircraft are

written as follows (as mentioned in [1]):

$$X = m(\dot{u} + qw - rv) \tag{1}$$

$$Y = m(\dot{v} + ru - pw) \tag{2}$$

$$Z = m(\dot{w} + pv - qu) \tag{3}$$

$$L = I_{xx}\dot{p} + I_{xz}\dot{r} + qr(I_{zz} - I_{yy}) + pqI_{xz} \tag{4}$$

$$M = I_{yy}\dot{q} + pr(I_{xx} - I_{zz}) + (r^2 - p^2)I_{xz} \tag{5}$$

$$N = I_{zz}\dot{r} + I_{xz}\dot{p} + pr(I_{yy} - I_{xx}) - qrI_{xz} \tag{6}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{7}$$

The variables mentioned in the equation 1 to equation 7 are described in Table 1. The variables that have a dot on top are the first derivative of the corresponding variable. For example, the variable $\dot{u}$ is the rate of change of linear velocity across the x - direction.

The model above can be linearized around a trim condition, which we will investigate in this study. Longitudinal trim is a flying envelope condition in which all of the forces of the longitudinal plane balance, and moments on a latitudinal plane are zero. In more practical terms, longitudinal trim is a stable situation in the flying envelope in which the aircraft maintains a specific altitude and speed without any elevator inputs from the pilot and any disturbances to the aircraft's present state are immediately recovered.

TABLE 1. VARIABLES AND THEIR DESCRIPTIONS

Variable	Variable Description
X	X-axis force
Y	Y-axis force
Z	Z-axis force
L	Rolling moments
M	Pitching moments
N	Yawing moment
u	Linear velocity across x direction
v	Linear velocity across y direction
w	Linear velocity across z direction
Φ	Roll angle
Θ	Pitch angle
Ψ	Yaw angle
p	Roll rate
q	Pitch rate
r	Yaw rate

At this trim condition, the flight is supposed to have a u_0, and a pitch angle θ_0. Aircraft is assumed to have zero angular velocity. $v_0 = p_0 = q_0 = r_0 = \phi_0 = 0$. Upon application of these longitudinal trim conditions to the equation (1) to (6) and followed by the removal of higher order terms, we get linearized state space equations of motion. The equation 8 represents the state space version of the longitudinal motion of the aircraft and equation 9 represents the state space version of the latitudinal motion of the aircraft [2]. Here $I_x^{'} = (I_{xx}I_{zz} - I_{xz}^2)/I_{zz}; I_z^{'} = (I_{xx}I_{zz} - I_{xz}^2)/I_x x; I_x^{'} z = I_x z/(I_{xx}I_{zz} - I_{xz}^2; \triangle\delta_e, \triangle\delta_t h, \delta_a, \delta_r$ are the elevator deflection, thrust deflection, aileron deflection and rudder changes of the airplane respectively. The notation of the force or moment with aircraft variable or control input corresponds to the stability or control derivative. In this study, we only consider the longitudinal motion for implementation with zero deflection in the control inputs.

$$\begin{bmatrix}\triangle\dot{u}\\ \dot{w}\\ \dot{q}\\ \triangle\dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{X_u}{m} & \frac{X_w}{m} & 0 & -g\cos(\theta_0)\\ \frac{Z_u}{m-Z_{\dot{w}}} & \frac{Z_w}{m-Z_{\dot{w}}} & \frac{Z_q+mu_0}{m-Z_{\dot{w}}} & \frac{-m\sin(\theta_0)}{m-Z_{\dot{w}}}\\ \frac{1}{I_{yy}}[M_u + \frac{M_{\dot{w}}Z_w}{m-Z_{\dot{w}}}] & \frac{1}{I_{yy}}[M_w + \frac{M_{\dot{w}}Z_w}{m-Z_{\dot{w}}}] & \frac{1}{I_{yy}}[M_q + \frac{M_{\dot{w}}(Z_q+mu_0)}{m-Z_{\dot{w}}}] & \frac{M_{\dot{w}}mg\sin(\theta_0)}{I_{yy}(m-Z_{\dot{w}})}\\ 0 & 0 & 1 & 0\end{bmatrix} \times \begin{bmatrix}\triangle u\\ w\\ q\\ \triangle\theta\end{bmatrix} + \begin{bmatrix}\frac{X_{\delta_e}}{m} & \frac{X_{\delta_{th}}}{m}\\ \frac{Z_{\delta_e}}{m-Z_{\dot{w}}} & \frac{Z_{\delta_{th}}}{m-Z_{\dot{w}}}\\ \frac{M_{\delta_e}}{I_{yy}} + \frac{M_{\dot{w}}Z_{\delta_e}}{I_{yy}(m-Z_{\dot{w}})} & \frac{M_{\delta_{th}}}{I_{yy}} + \frac{M_{\dot{w}}Z_{\delta_{th}}}{I_{yy}(m-Z_{\dot{w}})}\\ 0 & 0\end{bmatrix}\begin{bmatrix}\triangle\delta_e\\ \triangle\delta_{th}\end{bmatrix} \tag{8}$$

$$\begin{bmatrix}\dot{v}\\ \dot{p}\\ \dot{r}\\ \dot{\phi}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}\frac{Y_v}{m} & \frac{Y_p}{m} & (\frac{Y_r}{m} - u_0) & g\cos(\theta_0) & 0\\ \frac{L_v}{I_x^{'}} + I_{xz}^{'} & \frac{L_p}{I_x^{'}} + I_{xz}^{'}N_p & \frac{L_r}{I_x^{'}} + I_{xz}^{'}N_r & 0 & 0\\ I_{xz}^{'}L_v + \frac{N_p}{I_z^{'}} & I_{xz}^{'}L_p + \frac{N_p}{I_z^{'}} & I_{xz}^{'}L_r + \frac{N_r}{I_z^{'}} & 0 & 0\\ 0 & 1 & tan(\theta_0) & 0 & 0\\ 0 & 0 & sec(\theta_0) & 0 & 0\end{bmatrix} \times \begin{bmatrix}v\\ p\\ r\\ \phi\\ \psi\end{bmatrix} + \begin{bmatrix}\frac{Y_{\delta_a}}{m} & \frac{X_{\delta r}}{m}\\ \frac{L_{\delta_a}}{I_x^{'}} + I_{xz}^{'}N_{\delta_a} & \frac{L_{\delta_r}}{I_x^{'}} + I_{xz}^{'}N_{\delta_r}\\ I_{xz}^{'}L_{\delta a} + \frac{N_{\delta_a}}{I_z^{'}} & I_{xz}^{'}L_{\delta a} + \frac{N_{\delta_a}}{I_z^{'}}\\ 0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\delta_a\\ \delta_r\end{bmatrix} \tag{9}$$

Table 2, Table 3 gives the coefficient values of longitudinal, latitudinal stability and control derivatives, respectively. Table 4 shows the properties of Boeing

747 -100 at reference condition; here, the reference condition is the longitudinal trim condition. We choose a higher mass value to stabilize the system around the longitudinal trim condition. Since the parameters mentioned in [1] did not yield a stable converging system.

TABLE 2. LONGITUDINAL STABILITY AND CONTROL DERIVATIVES OF BOEING 747-100 AT REFERENCE FLIGHT CONDITION [1].

$\frac{\partial()}{\partial()}$	X (N)	Z (N)	M (N·m)
u (m/s)	-1.98×10^3	-2.595×10^4	1.593×10^4
w (m/s)	4.025×10^3	-9.030×10^4	-1.563×10^4
q (rad/s)	0	-4.524×10^5	-1.521×10^7
$\dot{w}$ (m/s^2)	0	1.909×10^3	-1.702×10^4
δ_e (rad)	-16.5299	-1.5794×10^6	-5.204×10^7
δ_{th}	849528	0	0

TABLE 3. LATERAL STABILITY AND CONTROL DERIVATIVES OF BOEING 747-100 AT REFERENCE FLIGHT CONDITION [1].

$\frac{\partial()}{\partial()}$	Y (N)	L (N·m)	N (N·m)
v (m/s)	-1.610×10^4	-3.062×10^5	2.131×10^5
p (rad/s)	0	-1.076×10^7	-1.330×10^6
r (rad/s)	0	9.925×10^6	-8.934×10^6
δ_a (rad)	0	-3.5323×10^6	-5.0945×10^4
δ_r (rad)	-4.9616×10^5	1.8013×10^6	-3.2457×10^7

TABLE 4. BOEING 747-100 PROPERTIES AT REFERENCE FLIGHT CONDITION [6].

Aircraft Properties	**Value**
Mass of the aircraft (W)	6×10^6 N
Wing area (S)	511.0 m^2
Standard mean chord of the wing (c)	8.234 m
Wing span (b)	59.64 m
I_{xx}	0.247×10^8 kgm^2
I_{yy}	0.449×10^8 kgm^2
I_{zz}	0.673×10^8 kgm^2
I_{xz}	-0.212×10^7 kgm^2
Reference flight velocity (u_0)	235.9 m/s
Air density (ρ)	0.3045 kg/m^3
θ_0	0

4. Model description - Deep Residual Recurrent Neural Network

The physics of any dynamical system can be represented as shown in the equation 10, it represents a general form. In equation 10, x represents the state vector of the system. In this study, the state vector constitutes states represented in the equation 8, shown in Equation 11.

$$\frac{dx}{dt} = f(t, x) \tag{10}$$

$$x = [\triangle u, w, q, \triangle \theta] \tag{11}$$

The solution of the equation 10 can be obtained using the Euler integration method as mentioned in the equation 12. The equation 12 gives the state at $t+1$. The variable h represents the time step size.

$$x_{t+1} = x_t + h \times f(t+1, x_{t+1}) \tag{12}$$

The residual between the state at t and state $t+1$ can be obtained using the equation 13.

$$r_{t+1} = x_{t+1} - x_t - h \times f(t+1, x_{t+1}) \tag{13}$$

In the equations 13, 12 f function corresponds to the longitudinal state function that is described in the equation 8. Hence the residual equation 13 can be rewritten as equation 14. A_{long} and B_{long} are the state and control input matrices of the state space models mentioned in equation 8.

$$r_{t+1} = x_{t+1} - x_t - h \times (A_{long} x_{t+1} + B_{long} \delta_{t+1}) \tag{14}$$

The DR-RNN input-output architecture is designed with the equations 15, 16, where k represents the layer number.

$$x_{t+1}^{(k)} = x_{t+1}^{(k-1)} - W \odot \tanh(U r_{t+1}^{(k)}), \quad \text{for } k = 1 \tag{15}$$

$$x_{t+1}^{(k)} = x_{t+1}^{(k-1)} - \frac{\eta_k}{\sqrt{G_k + \epsilon}} r_{t+1}^{(k)}, \quad \text{for } k > 1 \tag{16}$$

The residual $r_{t+1}^{(k)}$ is at time instant $t+1$ in the kth layer. The learnable weight parameters are W, U, η of the DR-RNN, and ϵ is a minimal quantity used to avoid division by zero. This report implements three different models with different shapes of W, U, η and discusses the results. The squared norm of the residual G_k is calculated as in equation 17 [2]. The $\odot$ symbol denotes element-wise multiplication [2].

$$G_k = \gamma ||r_{t+1}||^2 + \zeta G_{k-1} \tag{17}$$

The values of γ, ζ are set to 0.1, 0.9 respectively [2]. The DR-RNN iteratively minimizes the residual value by stacking up k layers. This model architecture can be seen as an extension to the Euler integration method but with non-linearity being learnt by the weight parameters during training which might provide

a cost-efficient way to predict the next state while being explicit in time [1]. The residual at $t = 0$ is calculated by substituting $x_{t+1} = x_t$ in the equation 14 [2].

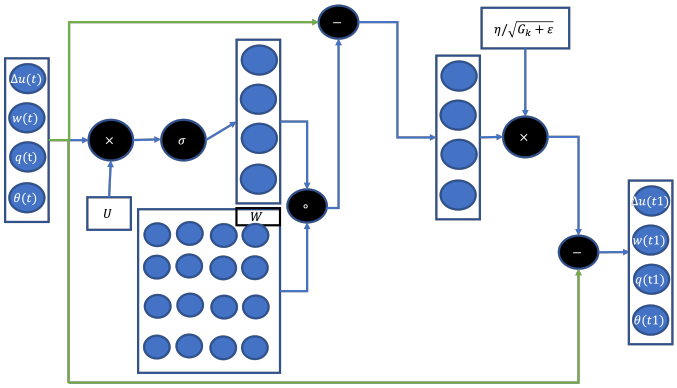

Figure 2. Model 2 Architecture

5. Implementation of DR-RNN

Three different architectures of the DR-RNN model are constructed and tested out. The main differences between the models are the definition of the weight parameters and the training strategy used to optimize the weights. Table 5 shows the different models and the number of learnable parameters used in different models. Figure 2 shows the pictorial representation of the Model 2.

TABLE 5. MODEL DESCRIPTIONS

Model Name	W	U	η	Layers
Model 1	4	1	1	2
Model 2	4	8	2	2
Model 3	256	512	2	2

5.1. Data generation

In this study, the Boeing 747-100 flight's reference condition altitude is set at 40,000 ft and u_0 is set at Mach 0.8. The data for training the models are generated by solving the equation 11 using LSODA numerical solver with random initial disturbances of the longitudinal states and no control input. For model 1, 1000 samples of 20s trajectories generated (step size $h = 0.1$)with random initial disturbance are used as the training(500) and validation(500) data. Model 2 uses a 500-second trajectory ($h = 0.1$) with random initial conditions, 1-degree elevator input and model 3 uses a 500-second trajectory ($h = 0.1$) generated with random initial disturbances as the training data. The generated data is normalized by dividing the $\triangle u, w$ by u_0 to reduce difficulties caused by the different ranges in the frequency scale. The longitudinal states in equation 11 after normalization can be rewritten as shown in equation 18.

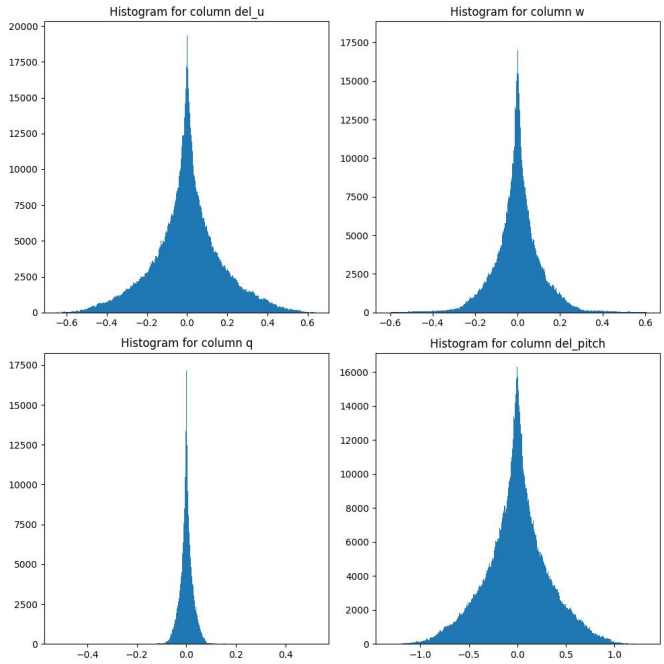

Figure 3. Training data histogram analysis

$$x = [\frac{\triangle u}{u_0}, \frac{w}{u_0}, q, \triangle\theta] \tag{18}$$

The histogram of the normalized data of all four longitudinal states is shown in figure 3. The states have values from a similar domain, which can be inferred from the histogram plot.

5.2. Training Methodology

We used different training strategies for three models. Model 1 used a training strategy as mentioned in [1], which used the loss function in equation 19. Given the $t = 0$ state from a sample in the training data, Model 1 predicts the entire trajectory, and loss is computed for the whole trajectory across all states. Then loss is back propagated to optimize the weights. The optimization of weight parameters was conducted

using the Adam optimizer. The model was trained for 800 epochs.

$$L(\theta) = \frac{1}{TSN} \sum_{s=1}^{S} \sum_{n}^{N} \sum_{t=1}^{T} (x_t^{pred} - x_t^{true}) \tag{19}$$

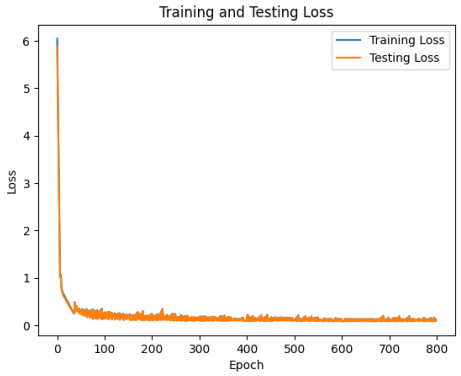

Figure 4. Model 1 loss

Model 2 and model 3 were trained using the teacher forcing method. The teacher-forcing method is used to speed up convergence for a sequence-to-sequence model. The task of predicting the next state while being explicit by time can be modelled as a sequence-to-sequence task. In this training method in every single time step of the training, the model is fed with the ground truth instead of the model's next state prediction. However, during the inference or prediction, the model uses its own prediction to predict the next state. Model 2 and Model 3 use mean square error as a loss function and Adam optimizer to optimize the weights. To train the models Tensorflow 2.0 was used to train the Model 2 and Model 3, and Model 1 was trained using the PyTorch. The reason for using PyTorch for Model 1 is because of its more computational freedom since the loss function used is not a traditional loss function. The learning rate used during the training of each model was 0.01. Models 2 and 3 each were trained for 300 epochs. The batch size used in the training the models 2 and 3 were 32 and for model 1 was 16 [1]. The loss vs epochs graph of models 1, and 2 is shown in figure 4 5. The model 2 training did not contain any validation data since the primary goal of the training was to understand the extrapolating behaviour of the model, that is, a model trained over one trajectory initiated with one initial condition can it reproduce the trajectory of different initial conditions over a long duration.

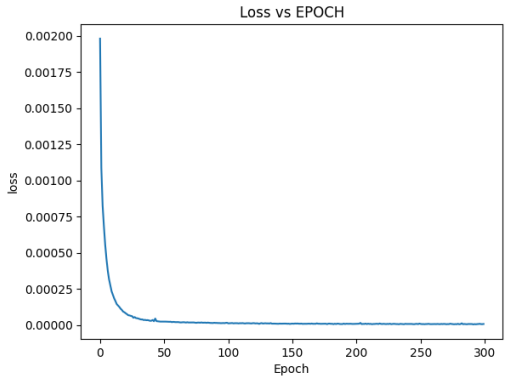

Figure 5. Model 2 loss

6. Results

The initial conditions used for testing model 1 were [0, 0, 0.3, 0], a ground truth comparison was conducted, and the model's predictions were found to be way off for even 20s from the ground truth as shown in figure 6. The same model tested over 800s with a similar initial condition yielded unconvincing results as shown in the 7. As you can see from the state trajectory graph simulated by model 1 with initial condition $q = 0.15$ is not satisfactory. Testing model 3 with the same initial condition with 1 deg elevator input produced a diverging trajectory for 500s with step size $0.1s$, predicting the system to be non-linear. But the aircraft system is empirically proven to be linearized in this longitudinal trim condition. The model 3 results are shown in figure 8, which are not satisfactory. The diverging behaviour can be explained by the higher number of weight parameters present in the model architecture. The 770 weight parameters combined with the physics of the aircraft should have introduced the non-linear diverging trajectory.

TABLE 6. INITIAL CONDITIONS AND RMSE ERROR FOR MODEL 3

case	Initial state	Control Inputs	RMSE Error
case1	$[0, 0, 0.15, 0]$	0	0.081305
case2	$[0, 0.13, 0.15, 0]$	0	0.076311
case3	$[0.11, 0.13, 0.15, 0.01]$	0	1.9876

Model 2 demonstrated a converging behaviour with multiple initial conditions. Table 6 shows the root mean squared error between the predicted and actual trajectory. The oscillatory behaviour of the system is captured, and the trajectory converges to the equi-

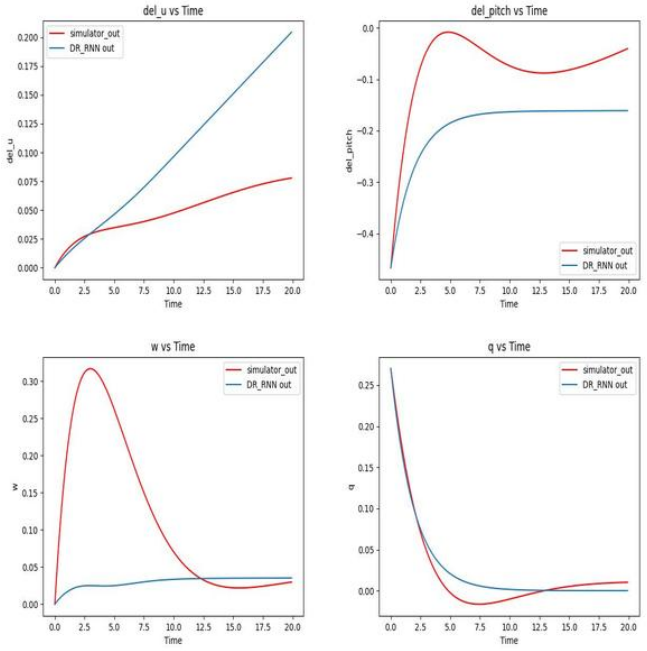

Figure 6. Model 1 validation

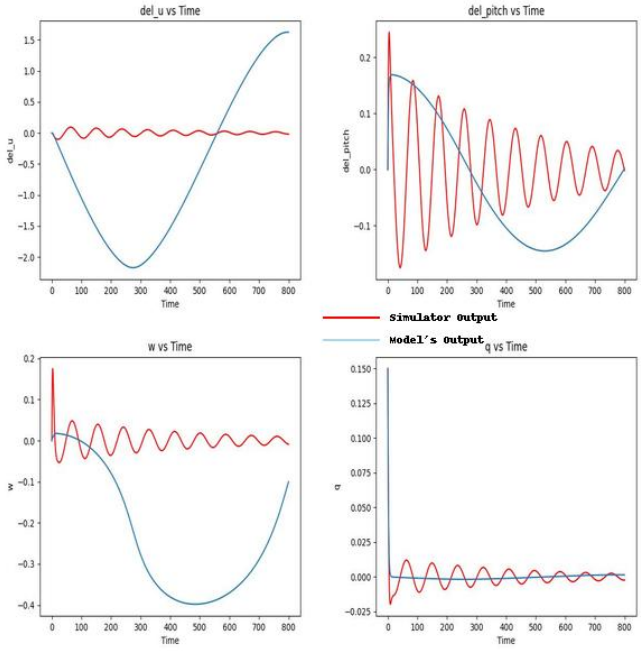

Figure 7. Model 1 800s trajectory prediction

librium position. Trajectory produced with the initial condition $[0.11, 0.13, 0.15, 0.01]$ corresponds to higher RMSE error because of the magnitude deviation in the $\triangle u$. Although the model produced very high RMSE error, this model architecture has outperformed the other two architectures presented because it captured the converging and reduced oscillatory behaviour. Also, the model performs very well with the initial conditions in which the model wasn't trained. The testing results of the three initial condition cases are shown in figure 9, 10, 11. Upon fine-tuning the model it can be made more accurate in terms of the phase and the magnitude of the predicted trajectory. Model 2 consumes around 70ms to produce a prediction for the next state.

7. Conclusion and Future Scope

In this report, we studied the DR-RNN and its shortcomings in understanding the physics of the aircraft. Although model 3 performed relatively better compared to other models, architectural and parameter level changes to make it a better-performing model. Model 1 replicates the DR-RNN model presented in the [1] but fails to produce the exact results. Seq2seq generative models that can quickly adapt to domain

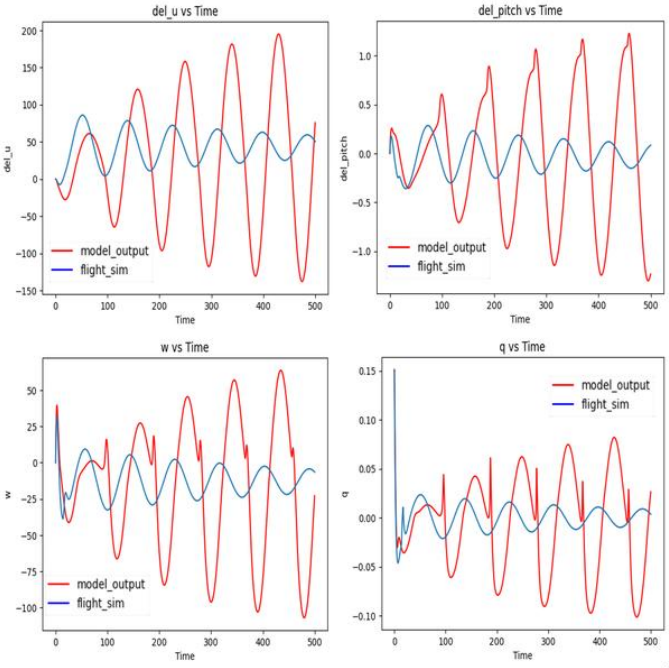

Figure 8. Model 3 500s trajectory prediction

changes can be explored in the future, which might pave the way for creating a digital twin of the aircraft. The digital twin can be used to test the plane, train the pilot and provide insights into the design of an aircraft.

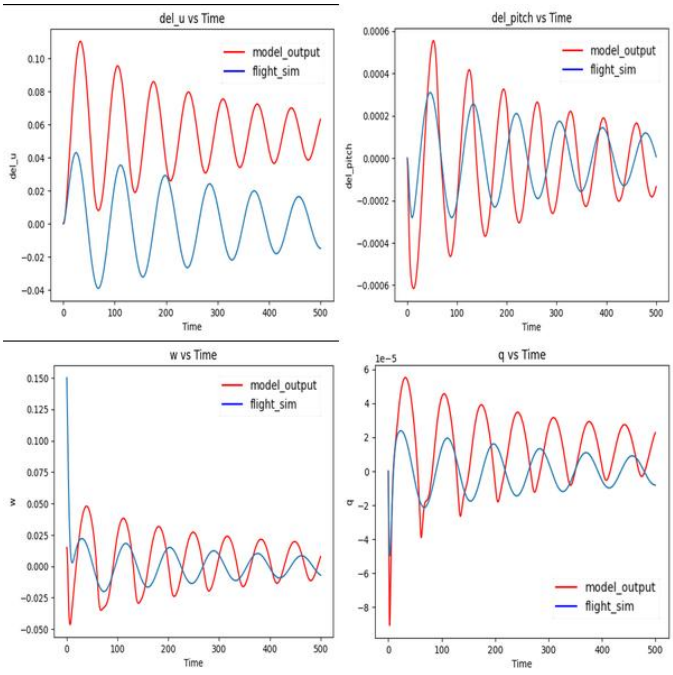

Figure 9. Model 2 500s trajectory prediction case 1

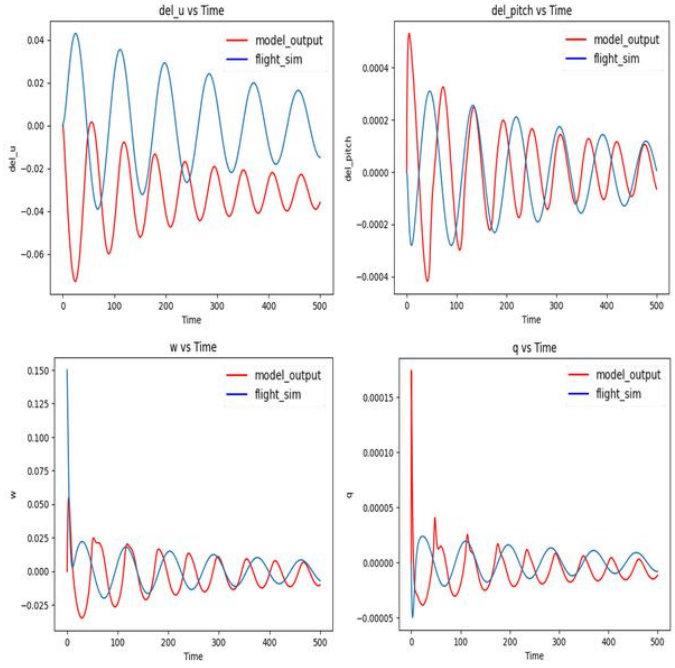

Figure 10. Model 2 500s trajectory prediction case 2

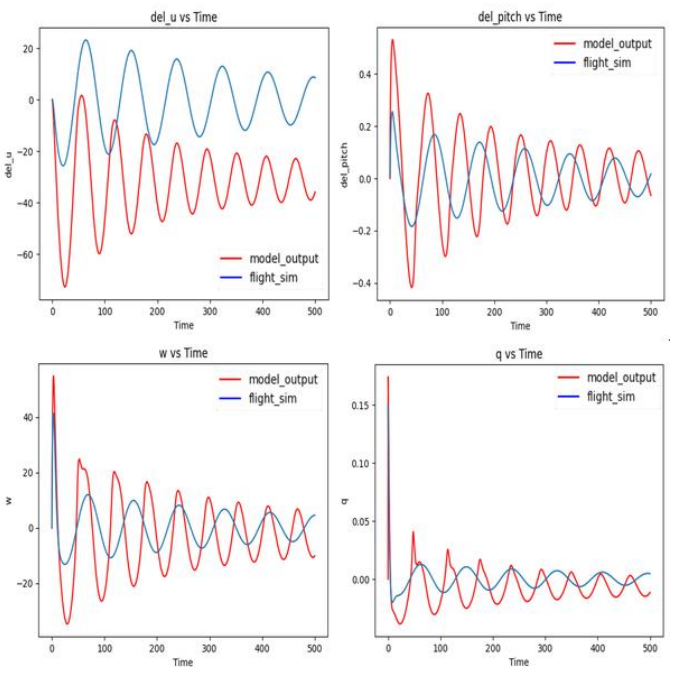

Figure 11. Model 2 500s trajectory prediction case 3

References

[1] Y. Yu, H. Yao, and Y. Liu, "Aircraft dynamics simulation using a novel physics-based learning method," *Aerospace Science and Technology*, vol. 87, pp. 254–264, 2019.

[2] J. N. Kani and A. H. Elsheikh, "Dr-rnn: A deep residual recurrent neural network for model reduction.," *CoRR*, vol. abs/1709.00939, 2017.

[3] T. Stachiw, A. Crain, and J. Ricciardi, "A physics-based neural network for flight dynamics modelling and simulation," *Advances in Modeling and Simulation in Engineering Sciences*, vol. 9, p. 13, 2022.

[4] Y. Yu, H. Yao, and Y. Liu, "Structural dynamics simulation using a novel physics-guided machine learning method," *Engineering Applications of Artificial Intelligence*, vol. 96, p. 103947, 2020.

[5] M. Pete, "Lecture 9: 6-dof equations of motion," MAE441 Spacecr. Aircr. Dyn. n.d.

[6] B. Etkin and L. D. Reid, *Dynamics of flight: stability and control*. John Wiley & Sons, 1995.