Kalman Based Neural Network Analysis with Resampling Methods for Longitudinal Aerodynamic Coefficient Estimation

Murat MILLIDERE [1] and Huseyin Burak KURT [2]
Middle East Technical University, Ankara, Turkey
Turkish Aerospace, Ankara, Turkey

Hakan BALLI [3]
Turkish Aerospace, Ankara, Turkey

Omur UGUR [4]
Middle East Technical University, Ankara, Turkey

Accurateness of the approximated aerodynamic characteristics of an unstable aircraft is considerably significant in flight control system design or high-fidelity flight simulator development. Classical system identification technique consist of the equation and output error methods which are strongly influenced by data quality. Accordingly, unsatisfactory results may be arisen due to measurement or process noise. Consequently, most of the engineers rely on feed forward neural network in order to cope with those undesired drawbacks. But, noisy data dramatically degrades the performance of neural network as well; thus, the Kalman filter based backpropagation algorithm is proposed. Neural network approach has many parameters, including the hyperparameters, which have to be searched for an optimal result. To determine these optimal parameters, the genetic algorithm is used. In this paper, it is aimed to estimate longitudinal aerodynamic characteristics of highly maneuverable unstable aircraft with the engaged control system for flight simulation data. For this purpose, F-16 aircraft is modelled using the aerodynamic database derived from a low-speed wind tunnel test. Successful results show the effectiveness of the proposed method. To assess neural network approach performance, the most common resampling methods are used and compared in order to choose the best for each longitudinal aerodynamic coefficient.

Nomenclature

p	=	Roll Rate
q	=	Pitch Rate
r	=	Yaw Rate
p^*	=	Normalized Roll Rate
q^*	=	Normalized Pitch Rate
r^*	=	Normalized Yaw Rate
B	=	Reference Length for Lateral-Directional Coefficient
L	=	Reference Length for Longitudinal Coefficient
V	=	True Airspeed
a_x	=	x-axis Linear Acceleration

[1] Ph.D. Candidate, Department of Engineering Sciences
[2] Flight Dynamics, Simulation & Control Engineer
[3] Flight Dynamics, Simulation & Control Engineer
[4] Prof. Dr., Institute of Applied Mathematics

a_y = y-axis Linear Acceleration
a_z = z-axis Linear Acceleration
LEF = Leading Edge Flap
C_X = x-axis Aerodynamic Force Coefficient
C_Y = y-axis Aerodynamic Force Coefficient
C_Z = z-axis Aerodynamic Force Coefficient
C_l = x-axis Aerodynamic Moment Coefficient
C_m = y-axis Aerodynamic Moment Coefficient
C_n = z-axis Aerodynamic Moment Coefficient
α = Angle of Attack
β = Angle of Sideslip
δ_a = Aileron Deflection
δ_h = Horizontal Tail Deflection
δ_r = Rudder Deflection
δ_{LEF} = Leading Edge Flap Deflection
ANN = Artificial Neural Network
GA = Genetic Algorithm
SSE = Sum of Square Error
MSE = Mean Square Error
N = Number of samples in the data
n_{train} = Number of samples in the training data
n_{test} = Number of samples in the testing data

I. Introduction

System identification yields a complete breakdown of the various components contributing to the observed response and thereby provides an overall understanding of the flight vehicle's dynamics. For many applications, an aircraft can be assumed to be a rigid body whose motion is governed by the laws of Newtonian physics. System identification can be used to characterize the applied forces and moments acting on the aircraft arising from aerodynamics, inertial, gravitational, and propulsion. Typically, thrust forces and moments are obtained from ground tests, so aircraft system identification is applied to model the functional dependence of aerodynamic forces and moments on the aircraft motion and the control variables.

Aircraft system identification is primarily equipped with a mathematical description for the aerodynamic forces and moments in terms of relevant, measurable quantities such as control surface deflection, aircraft angular velocity, airspeed or Mach number, and orientation of the aircraft toward the relative wind [1]. Aerodynamic parameters quantify the functional dependence of the aerodynamic forces and moments on measurable quantities; and in general, the mathematical model is assumed to be parametric. Aircraft system identification can then be defined as a parameter estimation problem. Thus, the parameter estimation process consists of finding values of these unknown model parameters in the assumed model structure.

Parameter estimation includes a probability density function that describes the difference between the system model response and the measured system response. There are two common approaches in aircraft parameter estimation problem in literature; these are equation-error method and output-error method. These estimation approaches can be distinguished from each other. In the case of the equation-error method, non-state parameters such as force and moment coefficients, which are not integrated during the simulation, are determined without the knowledge of their past history [2]. In the case of the output-error method, system outputs are considered, such as the angle of attack, angular rates, etc., which are integrated during a simulation [2]. In recent years, researchers are interested in artificial neural networks and deep neural networks due to their successful achievements in a vast range of application areas: aircraft aerodynamic parameter identification [3-7] is also one such application.

The neural network is one of the most commonly used machine learning techniques. Artificial neural networks (ANN) are considered as an approximation to a general function and are assumed to be capable of approximating any continuous function with any desired accuracy by appropriate network architecture [7]. However, these neural networks are, in general, black-box models: basically, they have no physical understanding for designers. The neural network is a combination of the weight matrix, input vector, and bias vector [8]. In order to understand a neural network model and be able to use it, many parameters, hyperparameters, are to be considered and be tuned by the user. If the user doesn't have solid experience with the neural network, it is unlikely that the model is a good one; thence, it might take a long time in order to achieve a reasonable result via the neural network. To solve this problem, researchers consider a neural network as an optimization problem and solve for the parameters using, e.g., the genetic algorithm [9].

Although the neural network approach works well with the standard backpropagation algorithm, it is sensitive to the stochastic disturbances, and model performance decreases for time history data with noise. Therefore, the Kalman filter-based backpropagation algorithm is used to train the neural network and estimate the aerodynamic coefficients from aforementioned noisy time history data [1].

In system identification, the dataset is divided into a training, a validation, and a testing dataset. In this paper, the data is firstly split into two parts; the first part is used to train the aerodynamic model. The second split of data is used to test the aerodynamic model. There are many alternative approaches to validate generated model performance; the most common tool is the hold-out cross-validation approach. This approach randomly divides the training dataset into two groups: training and validation datasets. This approach has a potential drawback; the estimated test error rate can be highly variable depending on which observation dataset falls into the training dataset and which observation falls into the validation dataset. Thus, it is a good practice to apply other resampling methods as a refinement to overcome this drawback; some of them are K-fold cross-validation and bootstrap approaches. Resampling methods are an indispensable tool in modern statistics [10].

The flight test data used in this study is collected from a high-fidelity F-16 simulator. F-16 aircraft is modelled with controls for the leading-edge flap, flaperon, horizontal tail, rudder, and throttle. The stability augmentation system and control augmentation system has been developed using the aircraft model to carry out the maneuvers. Simulation data are collected in twelve different trim points with a combination of two different altitudes and six different speed settings that differ from 0.3 Mach to 0.6 Mach speed. Control inputs of the pilot are designed for short-period, phugoid, dutch-roll, and bank-to-bank maneuvers to excite the different modes of vehicle dynamic motion.

II. Aircraft Modeling, Simulation, and Data Gathering

A. Aircraft Modelling

The aircraft simulation model includes the primary model and the subsystem models (bare-airframe, actuator, engine, environment, sensors, and flight control) as seen in Fig. 1. The bare-airframe model consists of aerodynamics and the equation of motion models (EOM). The aerodynamic data of the aircraft are obtained from the study published in the 1979 NASA technical report prepared from wind tunnel tests [11]. In the EOM model, all forces and moments acting on body axes are summed, and then, equations of motion are solved. Actuators are modeled as a first-order system [12]. The flight control model includes a stability augmentation system (SAS) and control augmentation system (CAS). In highly maneuverable aircraft, CAS and SAS are needed to perform tasks such as precision tracking of targets [12]. Pitch and roll rate command systems are used to control the aircraft.

B. Optimized Maneuvers

The main idea behind system identification maneuvers that we describe in the sequel is to excite related modes of the aircraft motion independently and sufficiently. In general, while exciting a particular mode, excitation of other modes should be avoided. Input design for a short period, phugoid, Dutch roll, and bank-to-bank maneuvers are explained below and depicted in Fig. 2.

Short Period Mode (SP)

It is a multi-step 3-2-1-1 elevator input exciting the short period motion with variations in the angle of attack of about 4 degrees and 0.5 g in the vertical acceleration. It provides the most information to enable the estimation of derivatives of the vertical and pitching motion [1].

Dutch-Roll (DR)

The Dutch Roll maneuver provides information to enable the estimation of the derivatives of the lateral motion [1]. Such a maneuver provides maximum information on the frequency and damping of this oscillatory mode. It is excited by applying rudder inputs. Usually, several cycles of oscillations are recorded. The resulting maximum peak-to-peak variation in the angle of sideslip is typical of the order of ±4 degrees or 0.1g lateral acceleration. The Dutch roll as well as the bank-to-bank maneuvers described below are performed at different trim speeds because most of the lateral-direction derivatives depend on the angle of attack.

Bank-to-Bank (BTB)

Bank-to-Bank maneuver provides more additional information on lateral-direction derivatives related to roll rate and aileron deflection [1]. Aileron input is applied, which roll the aircraft from wings-level to 30 degrees bank on one side; this is followed by changing input and going smoothly to the wings-level and to opposite bank angle, and again to wings-level condition. The changes in aileron result in rapid variation in roll rate and acceleration.

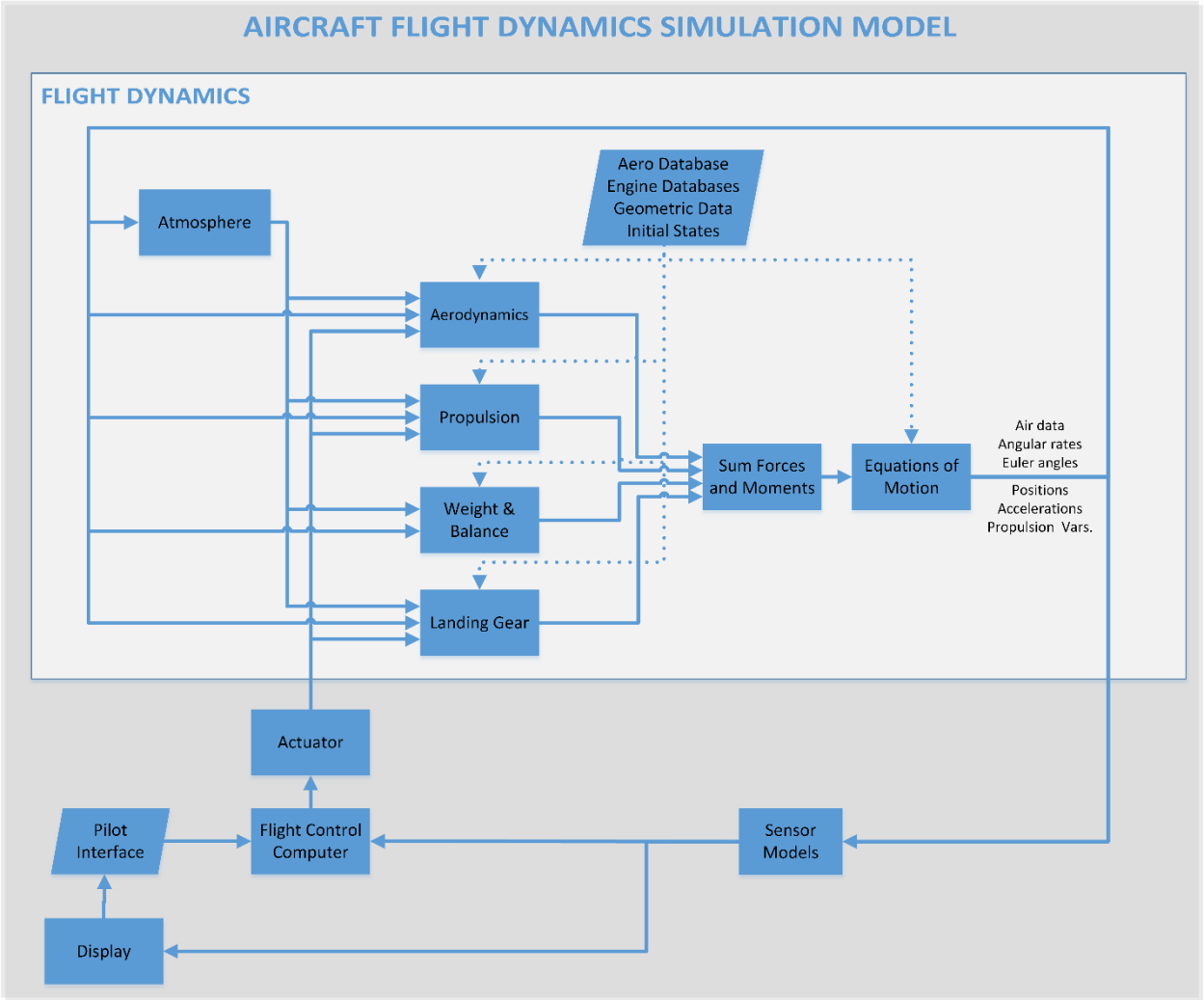


Figure 1. Aircraft model

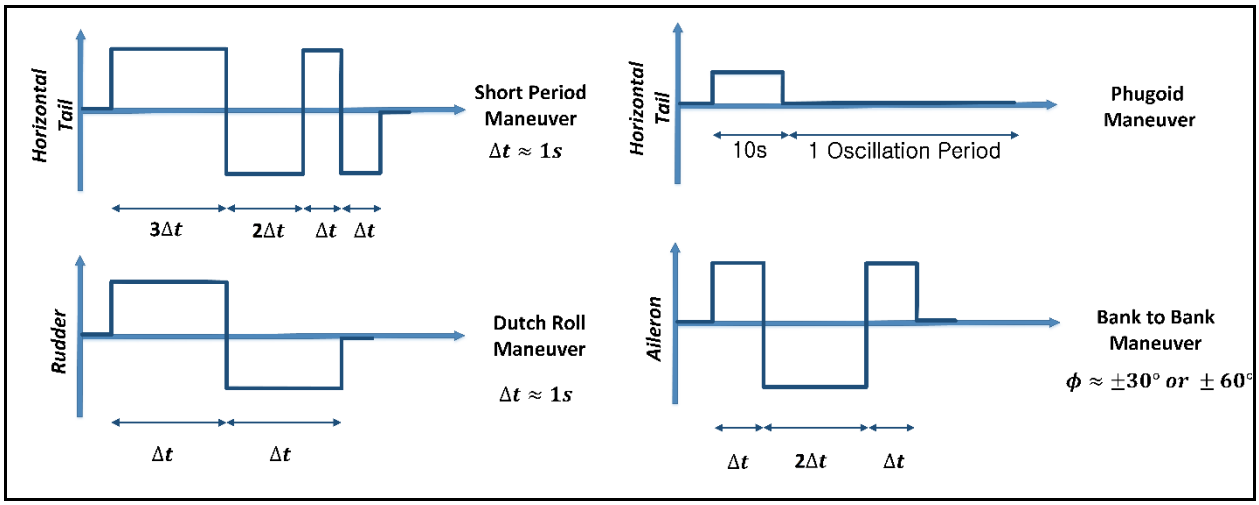


Figure 2. Control inputs for the short period, phugoid, dutch-roll, and bank-to-bank maneuvres

Control inputs of the pilot are applied for short-period, phugoid, dutch-roll, and bank-to-bank maneuvers to excite the different modes of vehicle dynamic motion. All test steps were started at level flight trim condition. Trims were not performed at idle or maximum power settings since there is no thrust effect on the aerodynamic coefficients. After the maneuver, the autopilot was activated to trim the aircraft. Flight test scope is shown in Fig. 3.

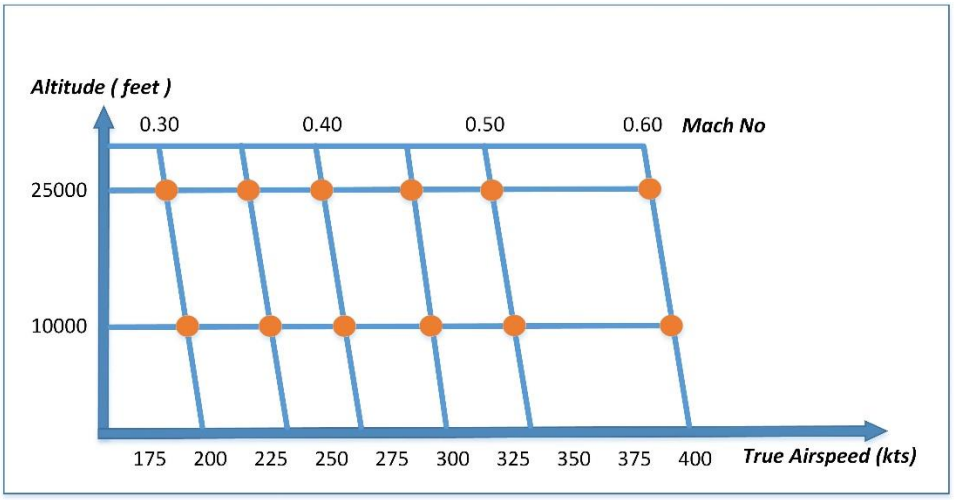


Figure 3. Flight test scope

For aerodynamic model extraction from the test data, a typical set of measurements required: (i) control surface deflections, (ii) linear accelerations, (iii) angular rates, (iv) attitude angles, (v) air data, (vi) static pressure, (vii) engine parameters, and finally, (viii) pilot forces are recorded [1-2]. A sampling frequency of 20-25 Hz is usually assumed to be sufficient for rigid-body aerodynamic model estimation [2]. Thereby, the simulation results are recorded at a 20 Hz sampling rate.

C. Flight Derived Aerodynamic Force and Moment Coefficients

A preprocessing step is required in the case of aerodynamic parameter identification as the aerodynamic forces and moments are not directly measured. However, those can be obtained from the measurements of the related variables such as linear accelerations, angular rates, mass properties, and other external forces and moments, as seen in Fig. 4 [13].

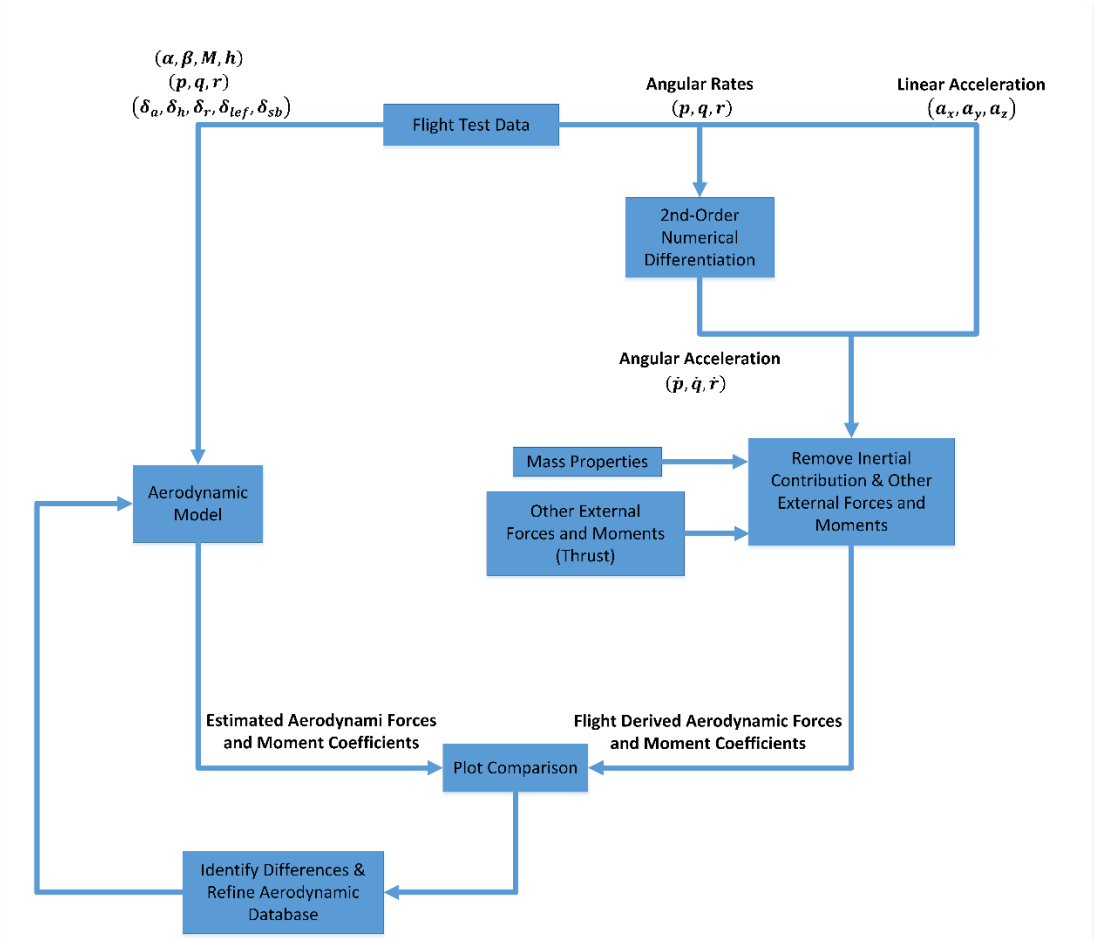


Figure 4. Flight derived aerodynamic force and moment coefficients

D. Aerodynamic Model Equations and Neural Network Dataset Preparation

Modeling the aircraft aerodynamics raises the fundamental question of what the mathematical structure of the model should be. Aerodynamic modeling provides a means of obtaining relationships between the three forces X, Y, Z along the three Cartesian coordinates and the moments l, m, n about these axes as functions of the linear translational motion variables u, v, w, rotational rates p, q, r, and control surface deflections [2]. For the system identification applied to an aircraft, it is more convenient to use non-dimensional derivatives of the non-dimensional aerodynamic force and moment coefficients [$C_X, C_Y, C_Z, C_l, C_m, C_n$]. These derivatives are obtained from the following relationships and expressed as a combination of aerodynamic derivatives [1].

Longitudinal aerodynamic coefficients when $param\ stands\ for\ X, Z, m$;

$$C_{param} = C_{param}(\alpha, \beta, q^*, \delta_h, \delta_{lef}) \tag{2.1}$$

Lateral aerodynamic coefficients when $param$ stands for Y, l, n;

$$C_{param} = C_{param}(\alpha, \beta, p^*, r^*, \delta_a, \delta_r, \delta_{lef}) \tag{2.2}$$

where normalized angular velocities are;

$$p^* = \frac{pb}{V}, \quad q^* = \frac{ql}{V}, \quad r^* = \frac{rb}{V} \tag{2.3}$$

Equation (2.1) and equation (2.2) can be written using non-dimensional derivatives for each non-dimensional aerodynamic force and moment coefficients as follows:

$$C_{param} = C_{param_0} + C_{param_\alpha}\alpha + C_{param_\beta}\beta + C_{param_{\delta_e}}\delta_e + C_{param_{q^*}}q^* + C_{param_{\delta_h}}\delta_h + \cdots \quad (2.4)$$

where $param$ stands for $[C_X, C_{Y,}\ C_Z, C_l, C_m, C_n]$.

In order to train the neural network, the recorded dataset is modified into specific format. The neural network needs input and output data for training. Therefore, the recorded values for $[\alpha, \beta, \delta_e, q^*, \delta_h, \ldots]$ are supplied as input and the corresponding values of C_{param} as output. Then, aerodynamic force and moment coefficient are obtained using mathematical equation and other properties as shown in Fig. 4. After obtaining aerodynamic force and moment coefficients, these values are given to the neural network and the aerodynamic model is therefore trained. Example of prepared input and output dataset is given in Table 1.

Dependencies of aerodynamic coefficients are generally taken as inputs. Note that the dependencies of aerodynamic coefficients are not same for each coefficient since each dimension have different influence on each aerodynamic coefficient. Therefore, the most effective dimensions taken for training process are shown in Table 2.

Table 1 Example of neural network time series dataset format

Time Step	Pitch Rate	AoA	AoS	Elevator Def.	LEF	AoA > 5 deg	AoA > 10 deg	Cx
1	0	0.1789	-0.0090	-0.0754	0.2676	0.0916	0.0044	0.0322
2	4.9847e-07	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
3	-5.3323e-06	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
4	-1.5072e-05	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
5	-2.4537e-05	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
6	-3.2099e-05	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
7	-3.7360e-05	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
8	-4.0519e-05	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
9	-4.1993e-05	0.1789	-0.0090	-0.0753	0.2676	0.0916	0.0044	0.0322
10	-4.2214e-05	0.1789	-0.0090	-0.0754	0.2676	0.0916	0.0044	0.0322
11	-4.1553e-05	0.1789	-0.0089	-0.0754	0.2676	0.0916	0.0044	0.0322
12	-4.0295e-05	0.1789	-0.0089	-0.0754	0.2676	0.0916	0.0044	0.0322
13	-3.8651e-05	0.1789	-0.0089	-0.0754	0.2676	0.0916	0.0044	0.0322
14	-3.6771e-05	0.1789	-0.0089	-0.0754	0.2676	0.0916	0.0043	0.0322
15	-3.4763e-05	0.1789	-0.0089	-0.0754	0.2676	0.0916	0.0043	0.0322
...	...	...	...	...	...	...	...	...

Table 2. Dependencies of aerodynamic coefficients

Coefficient	p	q	r	α	β	δ_h	δ_{lef}	δ_a	$\alpha^* \rightarrow \alpha > 5$	$\alpha^{**} \rightarrow \alpha > 10$
C_X	X			X	X	X	X		X	X
C_Y		X	X	X	X		X	X	X	X
C_Z	X			X	X	X	X		X	X
C_m	X			X	X	X	X		X	X
C_n		X	X	X	X		X	X	X	X
C_l		X	X	X	X		X	X	X	X

When the broad range of aircraft flight envelope is considered, these aerodynamics coefficients have nonlinear relationships with their dependent variables. The nonlinearity in model postulation can be expressed by using polynomial functions or using spline functions. Especially high degree polynomials show the problem of oscillation when the (equally-spaced) nodes of an interval are used in the interpolation. On the other hand, spline functions can avoid the disadvantages of the polynomial representation. They are defined only on the subintervals, and can approximate nonlinearities quite well. Besides, as seen in Table 1, the angle of attack range is divided and added as dependencies to neural network inputs for improving performances. Parameters dependent on the angle of attack in longitudinal coefficients are expressed using spline functions [2] in the form

$$(\alpha-\alpha_i)_+^m=\begin{cases}(\alpha-\alpha_i)^m & \alpha\geq\alpha_i\\ 0 & \alpha<\alpha_i\end{cases} \tag{2.5}$$

rather than the usual B-splines.

III. Neural Network Structure

Having prepared the input and the output datasets, we should find the weights of the nodes at the layers of a neural network structure. This process is referred to training the neural network. Backpropagation is the most commonly used method for this purpose; the essential idea of the approach is to observe the error between the real output values and the estimated output values by the network. Then, we perform, for instance, a gradient descent algorithm in order to decrease this error. The backpropagation algorithm is separated into two parts: a forward and a backward propagation.

A. Forward Propagation

In this procedure, the neural network makes the estimation of the output given the input values using the current weights, which are initially randomly initialized. This weight matrix and the bias vector are defined as follows:

W_1 and W_2, weight matrix between input-hidden layer and hidden-output layer, $(n_h\times n_u \; and \; n_y\times n_h)$

W_{1b} and W_{2b}, bias vector between input-hidden layer and hidden-output layer, $(n_h\times 1$ and $\; n_y\times 1)$

where n_h is the number of nodes in the hidden layer, n_u is number of nodes in the input layer, n_y is the number of nodes in the output layer.

Forward propagation steps are given in [1] as follows:

$$y_1=W_1u_0+W_{1b} \tag{3.1}$$

$$u_1=f(y_1) \tag{3.2}$$

$$f_i(y_1)=\tanh\left(\frac{\gamma_1}{2}y_1(i)\right)=\frac{1-e^{-\gamma_1 y_1(i)}}{1+e^{-\gamma_1 y_1(i)}} \text{ for } i=1,2,\dots,n_h \tag{3.3}$$

where γ_1 is the slope (gain) factor of hidden layer activation function. Similarly,

$$y_2=W_2u_1+W_{2b} \tag{3.4}$$

$$u_2=f(y_2) \tag{3.5}$$

$$f_i(y_2)=\tanh\left(\frac{\gamma_2}{2}y_2(i)\right)=\frac{1-e^{-\gamma_2 y_2(i)}}{1+e^{-\gamma_2 y_2(i)}} \text{ for } i=1,2,\dots,n_y \tag{3.6}$$

where γ_2 is the slope (gain) factor of hidden layer activation function.

B. Backpropagation with Momentum Term

The backpropagation algorithm is used with a momentum term to find an optimal weight matrix and bias vector. The backpropagation algorithm tries to decrease the associated and suitably chosen cost function with changing the weights and the bias.

Backpropagation steps are given as follows [1]:

$$E(k) = \frac{1}{2}[z(k) - u_2(k)]^T[z(k) - u_2(k)] = \frac{1}{2}e(k)^T e(k) \tag{3.7}$$

where E is the sum of squared errors (SSE), z is measured output values, and u_2 is estimated output values by the network and k is the index for the data.

$$W_2(k+1) = W_2(k) + \mu\left(-\frac{\partial E(k)}{\partial W_2}\right) \tag{3.8}$$

$$\frac{\partial E(k)}{\partial W_2} = -f'[y_2(k)][\, z(k) - u_2(k)]\, u_1^T(\mathrm{k}) \tag{3.9}$$

where μ is the so-called learning rate, $\frac{\partial E(k)}{\partial W_2}$ is the local gradients of the error cost function associated with W_2. Here, $f'[y_2(k)]$ is the derivative of the output node activation function defined by Eq. (3.6). Substituting Eq. (3.9) in Eq. (3.8) and denoting the result by e_{2b}, we obtain

$$e_{2b}(k) = f'[y_2(k)][\, z(k) - u_2(k)] \tag{3.10}$$

Weight update rule for the output layer with momentum term therefore can be written as

$$W_2(k+1) = W_2(k) + \mu e_{2b}(k)u_1^T(k) + \Omega[W_2(k) - W_2(k-1)] \tag{3.11}$$

Similarly, substituting Eq. (3.1) and Eq. (3.2) in Eq. (3.7) and considering the partial derivative with respect to W_1, we obtain

$$\frac{\partial E(k)}{\partial W_1} = -f'[y_1(k)]W_2^T e_{2b}(k)u_0^T(k) \tag{3.12}$$

where $f'[y_1(k)]$ is the derivative of the hidden layer activation function defined by Eq. (3.3). Thus, denoting the result e_{1b} it follows that

$$e_{1b}(k) = f'[y_1(k)]\, W_2^T e_{2b}(k) \tag{3.13}$$

Weight update rule for hidden layer with momentum term becomes

$$W_1(k+1) = W_1(k) + \mu e_{1b}(k)u_0^T(k) + \Omega[W_1(k) - W_1(k-1)] \tag{3.14}$$

To be specific, the derivative of Eq. (3.3) and Eq. (3.6) are given as follows:

$$f'(y_i) = \frac{\gamma_l}{2}\left[1 - \tanh^2\left(\frac{\gamma_l y_i}{2}\right)\right] = \frac{2\gamma_l e^{-\gamma_l y_i}}{(1+e^{-\gamma_l y_i})^2} \tag{3.15}$$

where $l = 1, 2$ is index for the input-hidden and output-output layers. These steps are recursively repeated for each training data indexed as $k = 1, 2, \ldots, n_{train}$, where n_{train} is the number of samples in the training dataset. At the end of the recursive loop, mean square error (MSE) is computed as a stopping criterion:

$$\text{MSE}_{Train} = \frac{1}{n_{train} n_y} \sum_{k=1}^{n_{train}} \sum_{j=1}^{n_y} [\, z_j(k) - u_j(k)]^2 \tag{3.16}$$

When MSE is sufficiently reduced, iteration is stopped.

C. Kalman based Backpropagation

In this algorithm, the overall training procedure similar to backpropagation with the momentum algorithm. There is a difference in the updating part. In this method, the updating part needs to calculate Kalman gains at each layer. This needs some computation time, so this method is slower than the standard backpropagation algorithm. Kalman based backpropagation steps, additional to standard backpropagation, are given [1] as follow;

The update equations for the output layer and hidden layer are given by

$$W_2(k+1) = W_2(k) + [d(k) - y_2(k)] K_2^T(k) \tag{3.17}$$

$$W_1(k+1) = W_1(k) + \mu e_{1b}(k) K_1^T(k) \tag{3.18}$$

where K_1 *and* K_2 are the Kalman gain vectors of the size $(n_h + 1, 1)$ and $(n_u + 1, 1)$ associated with hidden layer and output layer, respectively, and d is desired summation output and Kalman gains are given by

$$d(k) = \frac{1}{\gamma} \ln\left(\frac{1 + z(k)}{1 - z(k)}\right) \tag{3.19}$$

$$K_1(k) = \frac{D_1(k) u_0(k)}{\lambda_1 + u_0^T(k) D_1(k) u_0(k)} \tag{3.20}$$

$$K_2(k) = \frac{D_2(k) u_1(k)}{\lambda_2 + u_1^T(k) D_2(k) u_1(k)} \tag{3.21}$$

The matrices representing the inverse of the correlation matrices of the training data are;

$$D_1(k+1) = \frac{D_1(k) - K_1(k) u_0^T(k) D_1(k)}{\lambda_1} \tag{3.22}$$

$$D_2(k+1) = \frac{D_2(k) - K_2(k) u_1^T(k) D_2(k)}{\lambda_2} \tag{3.23}$$

where λ_1 and λ_2 denote the forgetting factors.

In general, Kalman based backpropagation approaches are less sensitive to stochastic disturbances and therefore lead to a nearly linear optimization problem having fewer local minima as well as faster convergence rates [1].

IV. Optimization Process with Genetic Algorithm

The neural network structure has many other parameters which have to be tuned for an optimal result. Researchers having substantial experience with neural networks might find the optimum parameters within a shorter period of time. However, those who don't have substantial experience with neural networks might spend some time in order to find the optimal parameter values. In addition, in this study, to find out the optimal values for the parameters of the neural network structure, we utilize the genetic algorithm [9].

In order to use genetic algorithm, firstly the decision of the parameters we optimize should be made: the number of hidden layers, slope factor of the hidden layer activation function, slope factor of the output layer activation function, learning rate parameter, momentum parameter, and initial random weights scale factor (randomly chosen initial weights are kept fixed) are optimized by the genetic algorithm.

Having decided the parameters to be optimized, the loss (cost) function should be decided, which will be minimized by the genetic algorithm: mean square error (MSE) between the estimated value and actual observed value of the aerodynamic coefficient is chosen as the cost function below.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(z_i - y_i)^2 \tag{4.1}$$

where n is number of samples in the dataset, z_i and y_i are the actual and the estimated values for the ith observation in the sample, respectively. Readers should refer to [9] for details of the genetic algorithm

V. Resampling Methods

To evaluate the performance of the feed-forward neural network on a given dataset, we need to measure how well its predictions match the observed data. In the regression setting, the most commonly used measure is the mean squared error:

$$\mathrm{MSE}_{Train} = \frac{1}{n_{train}}\sum_{i=1}^{n_{train}}(z_i - y_i)^2 \tag{5.1}$$

where n_{train} is the number of samples in the training dataset, z_i and y_i are the actual and the estimated values for the ith observation in the training set.

However, rather than how well the method works on the training data, one is most interested in how well the trained model works for the data in the testing dataset which has not been used to train the network at all. That is,

$$\mathrm{MSE}_{Test} = \frac{1}{n_{test}}\sum_{i=1}^{n_{test}}(z_i - y_i)^2 \tag{5.2}$$

should be computed for assessing of the model, where n_{test} is number of samples in the testing dataset, z_i and y_i are the actual and the estimated values for the ith observation in the testing dataset we have separated aside.

Adding many neural nodes to neural sets (increasing complexity), which decreases the training MSE, does not mean that it decreases the test MSE. When a given method yields a small training MSE but a large test MSE, it is called overfitting. The small changes in the training data can result in massive changes in the estimated model. In this case, we are modeling the random error rather than the pattern hidden in the data. This is not the desired case [10]. Such a situation is demonstrated in Fig. 5 [14]. We address resampling methods, which are indispensable tools in modern statistics, to mitigate this undesired case [10, 14].

The performance of the neural network is an important criterion to assess the optimized parameters. The available dataset is divided into training and testing dataset, firstly. The testing dataset, which is not seen in the identification phase, is used to assess the final performance at the end of the model identification. However, we should remark that in the identification phase, the training dataset can be used in different ways in order to assess the fitted model using various resampling methods, e.g., train-validation split, cross-validation sets, or random resamples. Use of such methods are illustrated in Fig. 6, in which the enclosed box indicates that several iterations may be required to obtain the finally trained model to be assessed by the testing dataset. Below we describe some of the commonly used

validation techniques in order to have assess how well the trained model might work on the testing dataset. For more information on the topic, we refer to [14].

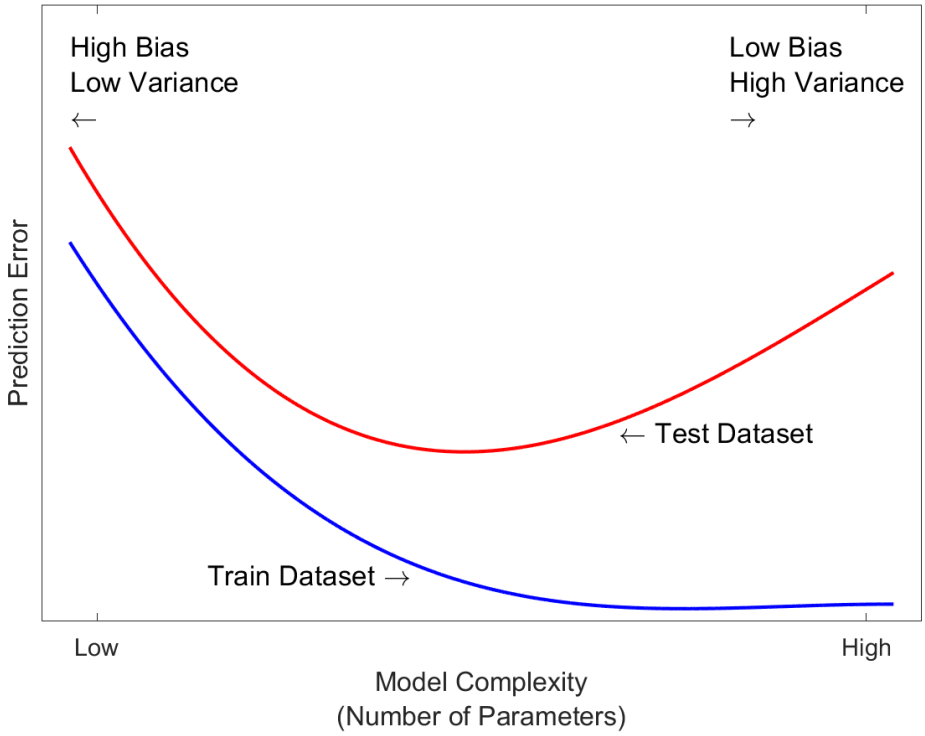


Figure 5. Test and training error as a function of model complexity

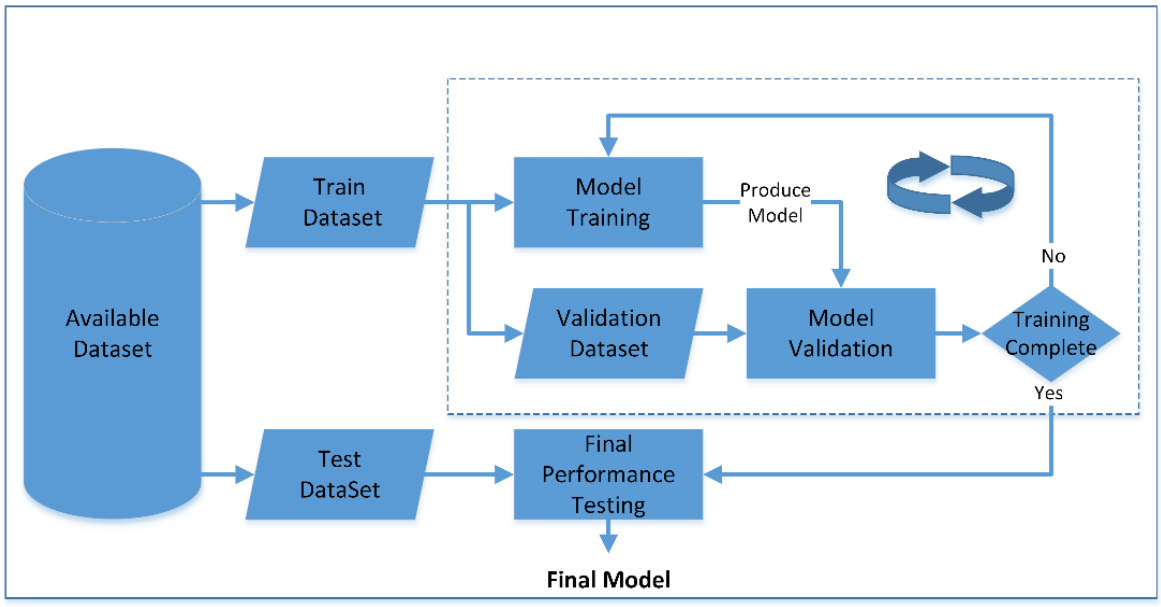


Figure 6. Model analysis engineering flowchart

A. Hold-out Cross-Validation

It is a very simple strategy, which is also named as cross-validation set approach. It involves randomly dividing the available training dataset into two parts: a training dataset, and a validation dataset, illustrated in Fig. 7.a. The model is fit on the newly generated training dataset, and the fitted model is used to predict the responses for the observations in the validation set. The trained model is then used for the testing dataset. Generally, the training data consists of %80 of the available data set. This simple approach is easy to implement, but it has a major drawback: testing MSE can be highly variable, depending on which observations are included in the training and which observations are included in the testing datasets.

B. K-fold Cross-Validation

This approach attempts to address the drawback of the hold-out cross-validation approach. This approach involves randomly dividing the set of observations into $\boldsymbol{k}$ groups, or folds, of approximately equal size, illustrated in Fig. 7.b. The first fold is treated as a validation data, and the remaining $\boldsymbol{k-1}$ folds are used to train the model. The mean square error for the validation set (first fold) is computed. This procedure is repeated $\boldsymbol{k}$ times; each time, a different group of observations is treated as a validation set. Hence the process results in $\boldsymbol{k}$ estimates of the validation MSEs. The $\boldsymbol{k}$-fold cross-validation error estimate is then computed by the average [10, 14]:

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{MSE}_{Val_i} \tag{5.3}$$

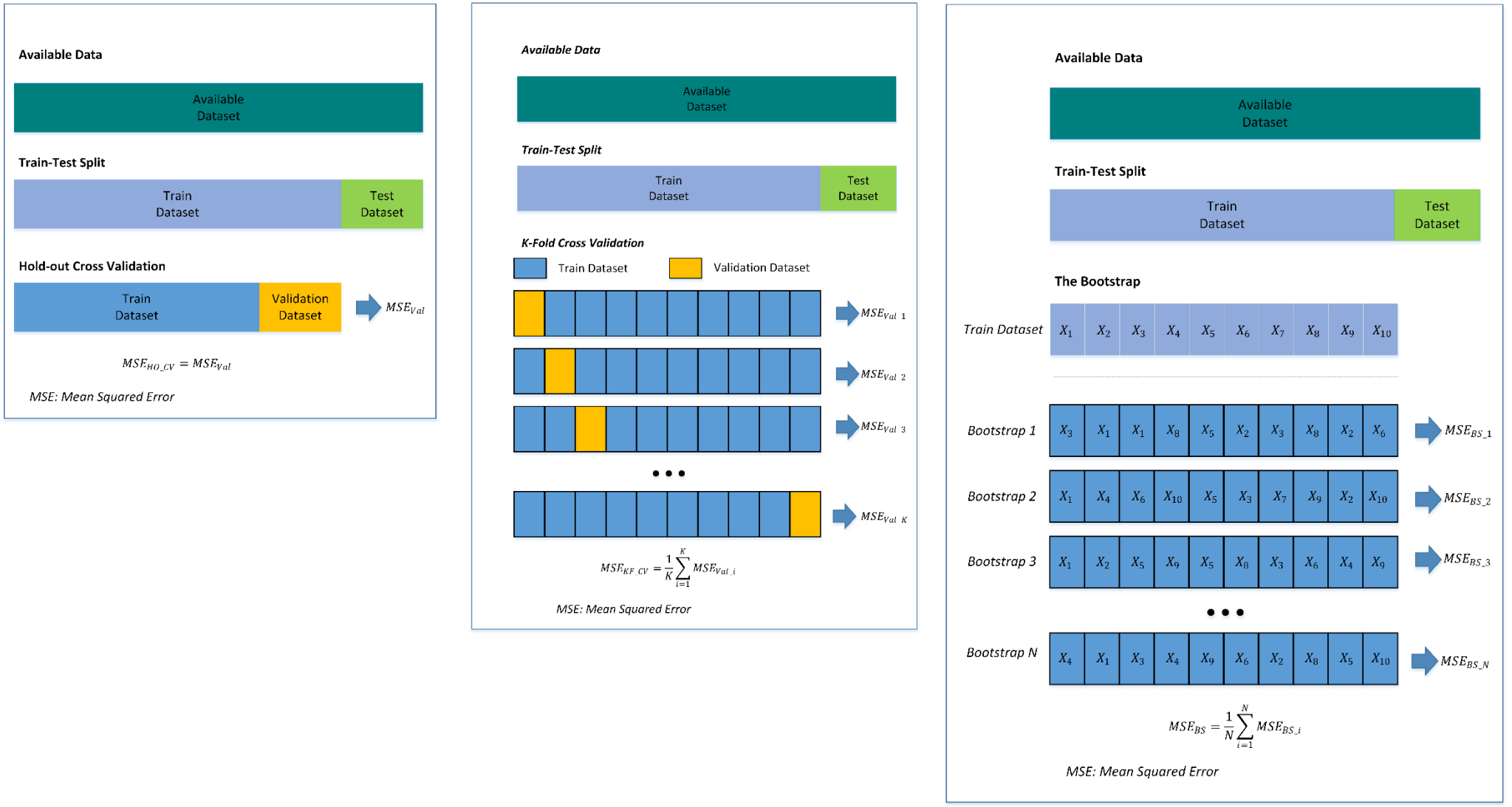
In contrast to the hold-out cross-validation approach, which produces different results when applied with different splits due to randomness in the train and validation splits, performing a $\boldsymbol{k}$-fold cross-validation is more robust in that respect. Generally, the number of folds, $\boldsymbol{k}$, to be used in this approach for a given $\boldsymbol{n}$ observations, as recommended in [15], is

$$k = \min\left\{\sqrt{n}, 10\right\} \tag{5.4}$$

C. Bootstrap

The bootstrap method is used to quantify the uncertainty associated with a given estimate by sampling a dataset with replacement, which means that the same observation can occur more than once in the bootstrap dataset. We obtain distinct data sets by repeatedly sampling observations from the original dataset, illustrated in Fig. 7.c. Having *N* randomly chosen bootstrap sets out of the training data, the model is trained to calculate each corresponding MSE. Hence, the overall MSE for the bootstrap approach the average is computed as an indicator of the testing error:

$$\mathrm{MSE}_{BS} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{MSE}_{BS_i} \tag{5.5}$$



(a) (b) (c)

Figure 7. Resampling Methods. (a) Hold-out Cross-Validation (b) K-fold Cross-Validation (c) Bootstrap

VI. Results and Conclusion

Neural network performance results are given in this section. The optimized parameters are found with the genetic algorithm for three different resampling methods. Optimized parameters are used to calculate the mean square errors for training, validation, and testing datasets.

A. Aerodynamic Coefficient Results

C_X Aerodynamic Coefficient

The resampling methods comparison on training, validation, and testing datasets are given in Fig. 8.

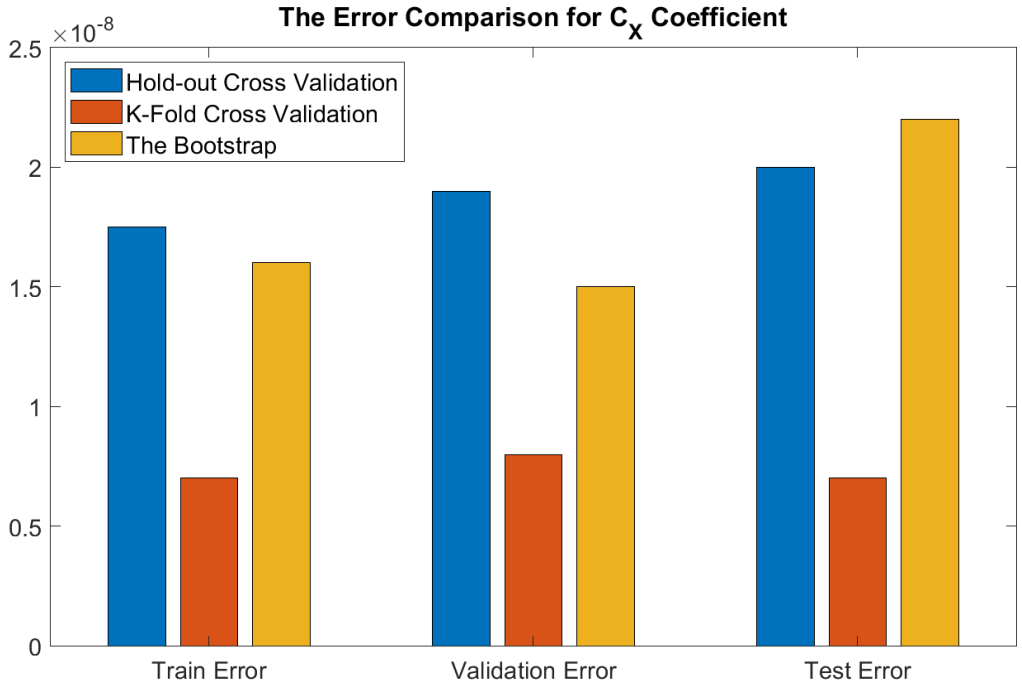


Figure 8. Resampling methods comparison on training, validation and testing datasets for C_X

Time history plots compare the time histories of flight measured and model estimated responses, which is a standard procedure to qualitatively evaluate the model fidelity. Any discrepancies in the match between the two responses often provide important clues to improve the model fidelity. The time history results for C_x in training dataset is given in Fig. 9. No discrepancy is observed in the plot.

Figure 9. Time history result for C_X in training dataset

Cross Plots of Residuals, the test of the residuals (error between the predicted and measured response) is a good indicator of the assumptions made. Flat spread of residual centered around zero is the ideal case whereas non-flat spread indicates that the model needs improvement with the variable observed.

Residual cross plot for C_X with respect to air data, pitch rate, and control surface deflections in training dataset are given in Fig. 10, and Fig. 11.

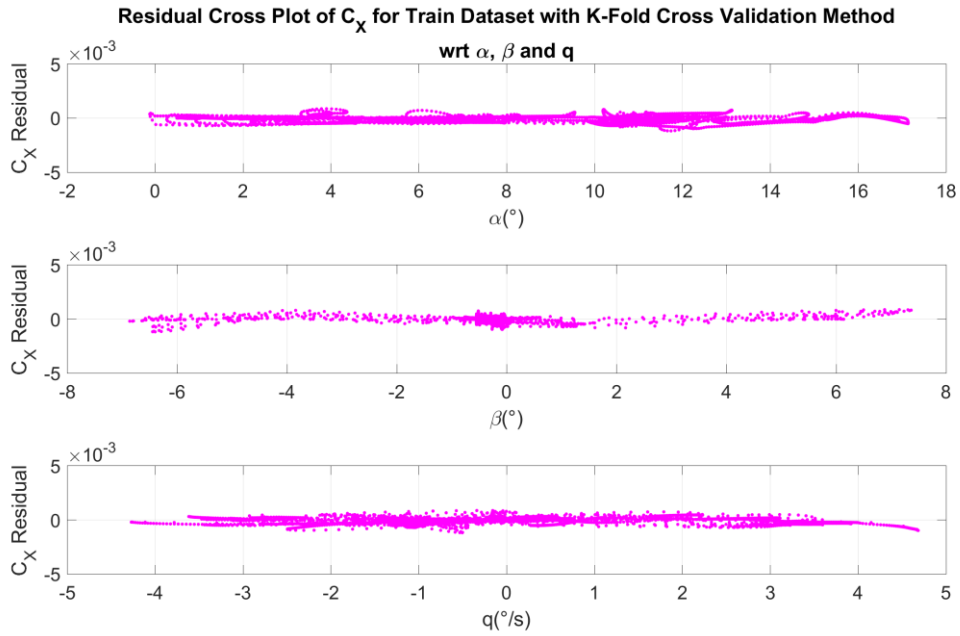


Figure 10. Residual cross plot, results for C_X with respect to α, β, and q in training dataset

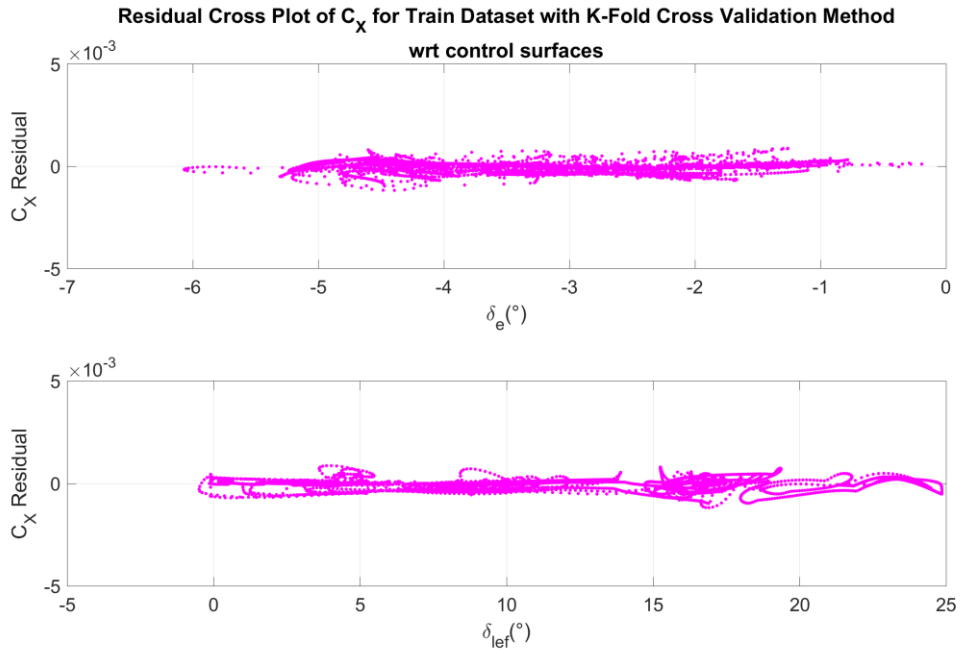


Figure 11. Residual cross plot results for C_X with respect to control deflections in train dataset

As shown in Fig. 10, cross plot of C_X is more dispersed in higher α values. With α is getting higher, residual error of model also is getting higher. However, residual error of cross plot is still in acceptable range.

In Fig. 10, higher β effect to residual error, can be seen in cross plot of C_X vs β. As seen in figure, higher β cause more residual error. However, error is still small and it is in acceptable range. In the same figure, residual cross plot of C_X is affected by aircraft pitch rate (q). Pitch rate effect have similar behavior with β effect. Both of their residual error is getting higher, when q/β is getting higher. However, there are more residual error, when we talk about pitch rate. As seen in figure, residual error is still in acceptable range.

In Fig. 11, residual cross plot of control surfaces can be seen. Although estimation error is still in acceptable range, cross plot of elevator deflection is not flat, when comparing others. Residual cross plot of leading-edge flap deflection is getting higher error, when leading edge flap deflection is getting higher value. But, still in acceptable range. This residual plot indicates that our trained model for C_X can estimate successfully.

As stated earlier, the results of training data are significant but the final performance of the aerodynamic model should be tested with testing data reserved and are not used in the identification phase. The time history results for C_X in testing dataset are shown in Fig. 12.

Test dataset time history results for C_X aerodynamic coefficient:

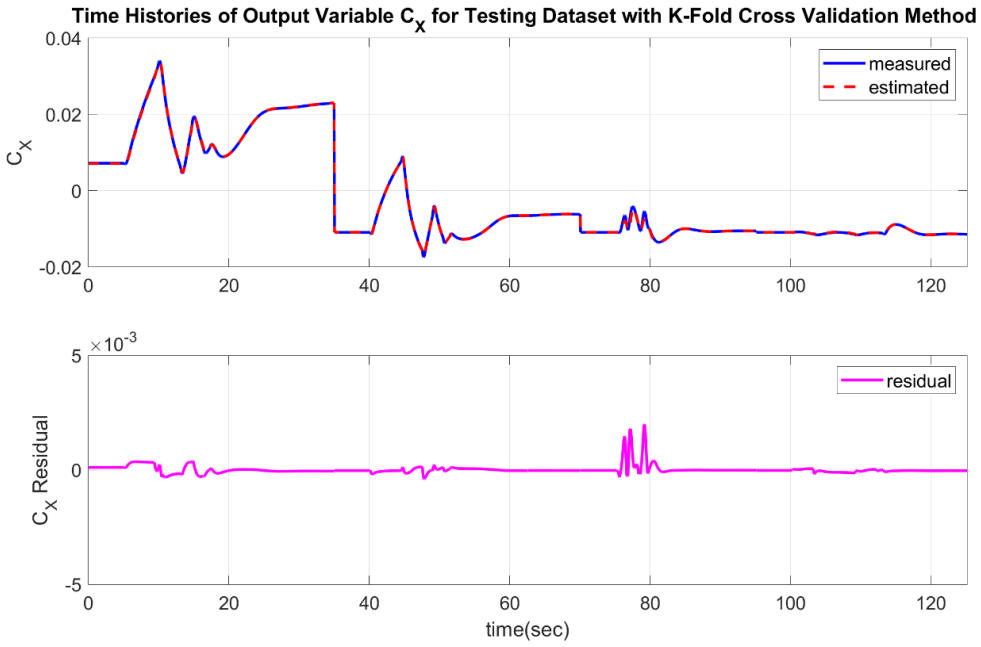


Figure 12. Time history result for C_X in the test dataset

In Fig. 12, time history of C_X variable for testing dataset is plotted. Measured and estimated data are almost the same. As seen in this figure, the residual error is so small that our aerodynamic coefficient model for C_X is acceptable.

C_Z Aerodynamic Coefficient

The resampling methods comparison on train, validation, and test datasets are given in Fig. 13.

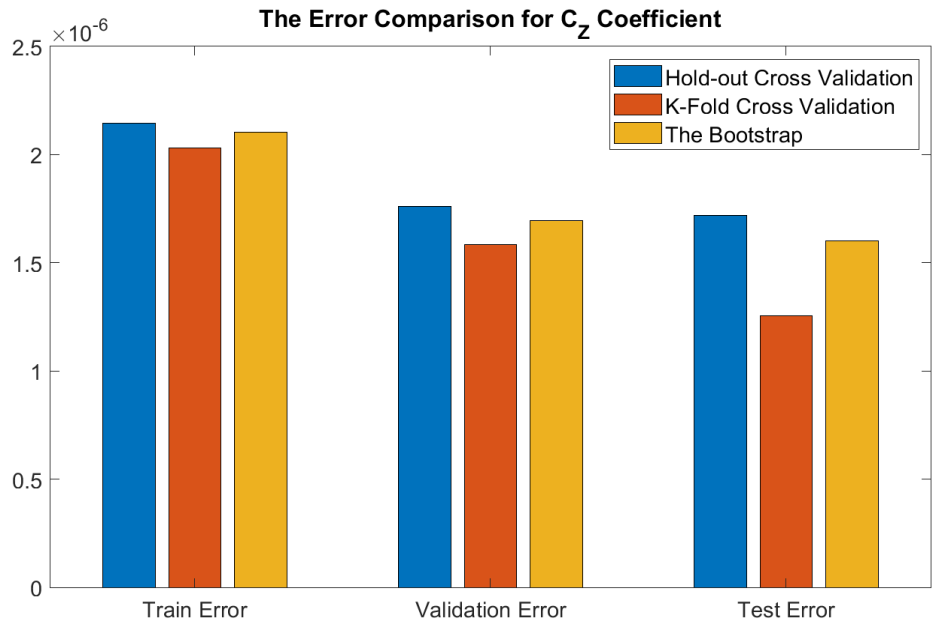


Figure 13. Resampling methods comparison on training, validation and testing datasets for C_Z

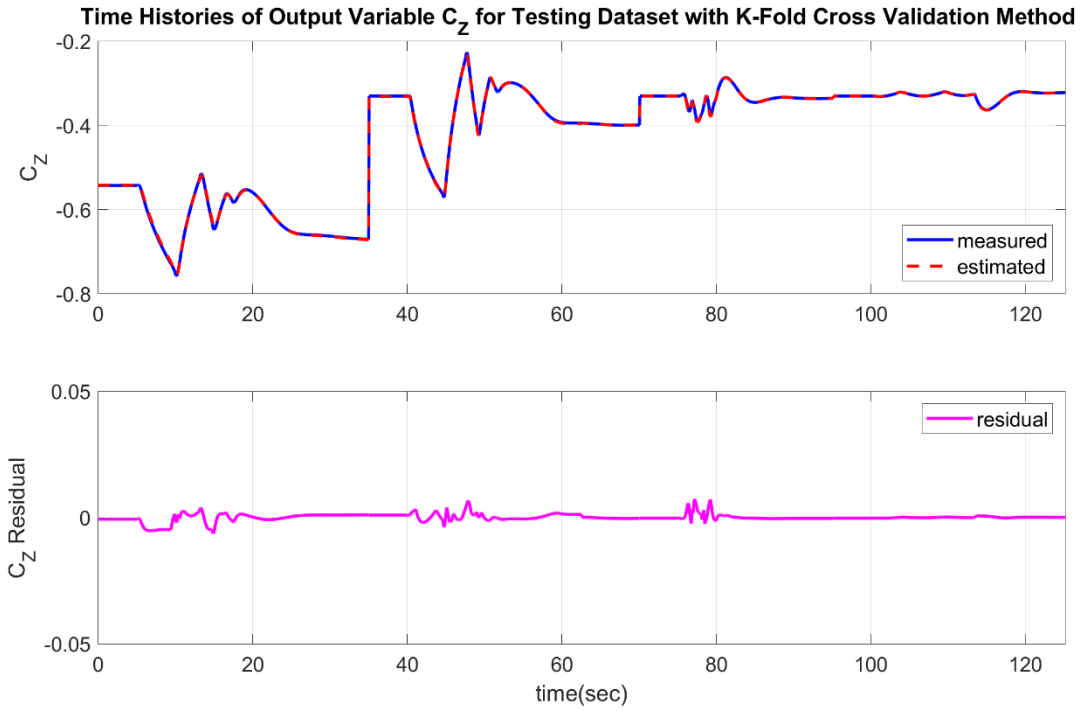


Figure 14. Time history result for C_Z in the testing dataset

C_m Aerodynamic Coefficient

The resampling methods comparison on train, validation, and test datasets are given in Fig. 15.

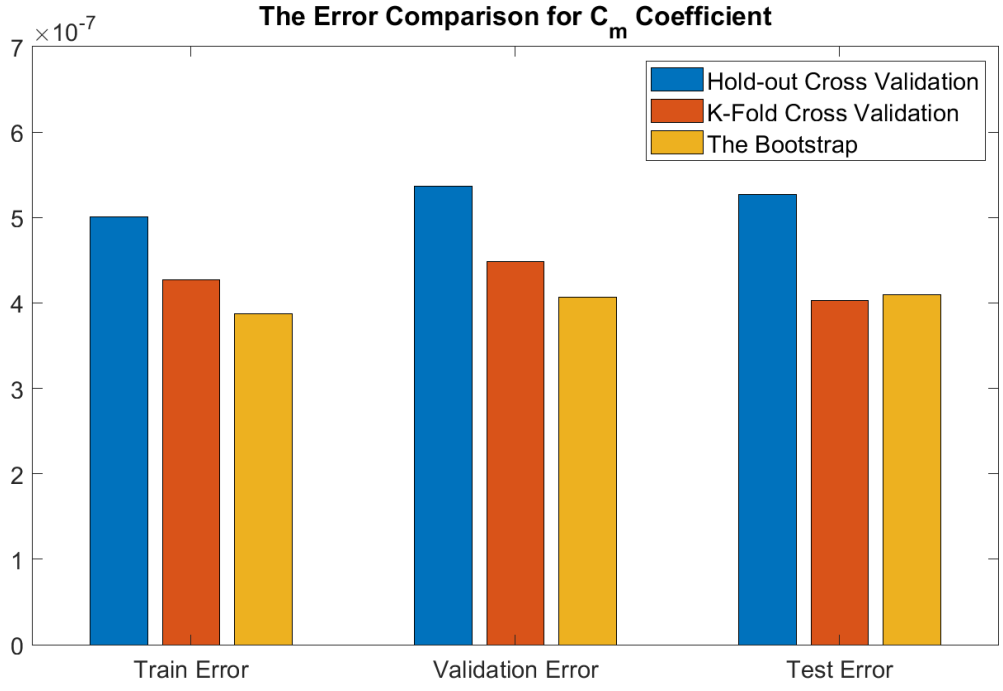


Figure 15. Resampling methods comparison on train, validation and test datasets for C_m

Figure 16. Time history result for C_m in the testing dataset

As seen in Fig. 8, Fig. 13, and Fig. 15, the *k*-fold cross-validation approach is superior to the other resampling approaches we used. As is also seen that the bootstrap approach gives a better performance, in general, than the hold-out cross-validation approach in the training and validation dataset; however, sometimes it is just the opposite in the testing dataset.

B. Proof of Match Results

The capability of identified model is determined by comparing the flight measured system responses with those predicted by the model. In flight vehicle applications terminology, this process is called proof-of-match; it is an important part of flight simulator certification and acceptance. In this proof-of-match process, the identified aerodynamic model is kept fixed [1].

To eliminate subjective evaluation of the match between measured system responses and model predicted outputs, FAA has specified guidelines in terms of tolerances for each variable, depending upon the nature of the validation test. As an example, Table 3 provides the definition of three tests, giving tolerances, flight conditions to be tested for each. For complete list of validation test, the reader is referred to [16, 17].

Longitudinal parameters matching is performed in two phases. The static part of aerodynamic coefficients is validated with Longitudinal Maneuvering Stability and Longitudinal Static Stability tests. Dynamic part of aerodynamic coefficient is validated with the Short-Period maneuver.

Table 3: FAA validation tests and tolerance values

TEST NUMBER	TITLE	PARAMETERS	TOLERANCES	FLIGHT CONDITION
2c(6)	Longitudinal Maneuvering Stability	ELEVATOR	±1° or ±10%	CRUISE APPROACH LANDING
2c(7)	Longitudinal Static Stability	ELEVATOR	±1° or ±10%	APPROACH
2c(10)	Short Period Dynamics	PITCH ANGLE PITCH RATE NORMAL ACCELERATION	±1.5° ±2°/Sec ±1g	CRUISE

Longitudinal Maneuvering Stability

The objective of this test is to demonstrate that the simulation of maneuvering stability conforms to the aircraft. The test is performed by establishing a steady-state condition at several intermediate bank angle up to the maximum angle. A critical factor for this test to obtain an accurate trim condition for steady turning flight for prescribed bank angles. Steady-state aircraft trim conditions for turning flight can be found using Newton-Raphson method [18]. The results are "snapshot" once the airplane has been stabilized at the required bank angle and at the trim airspeed (three different airspeed). Altitude is set to constant in each case.

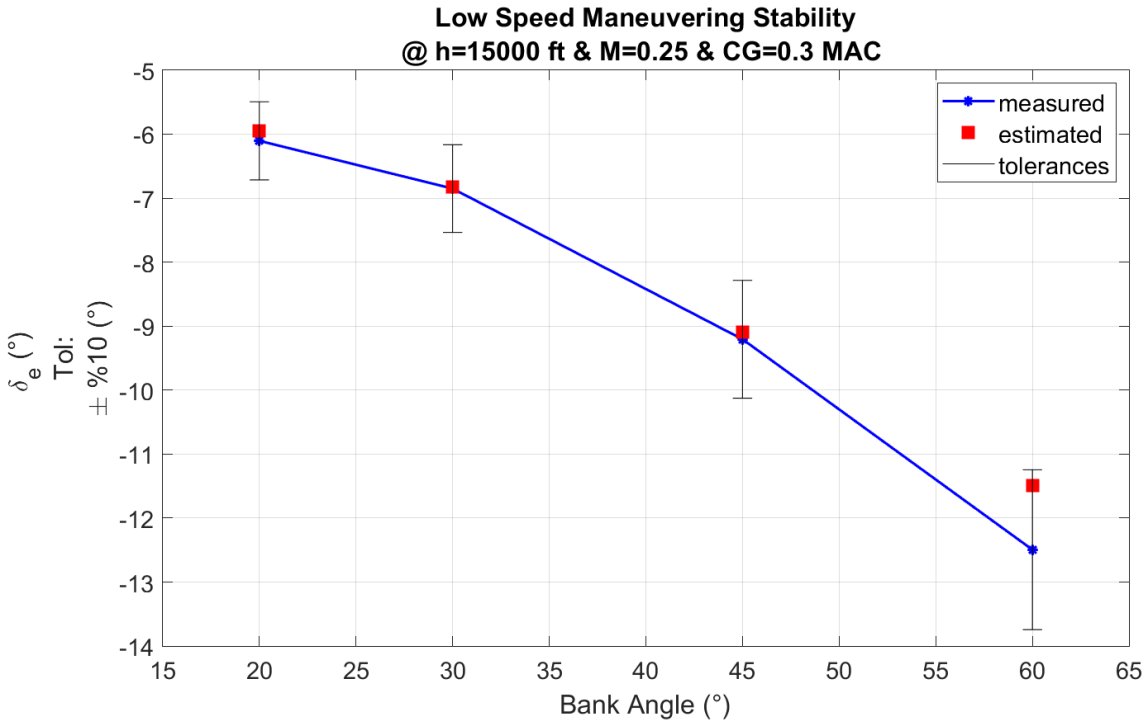


Figure 17. Low Speed Maneuvering Stability

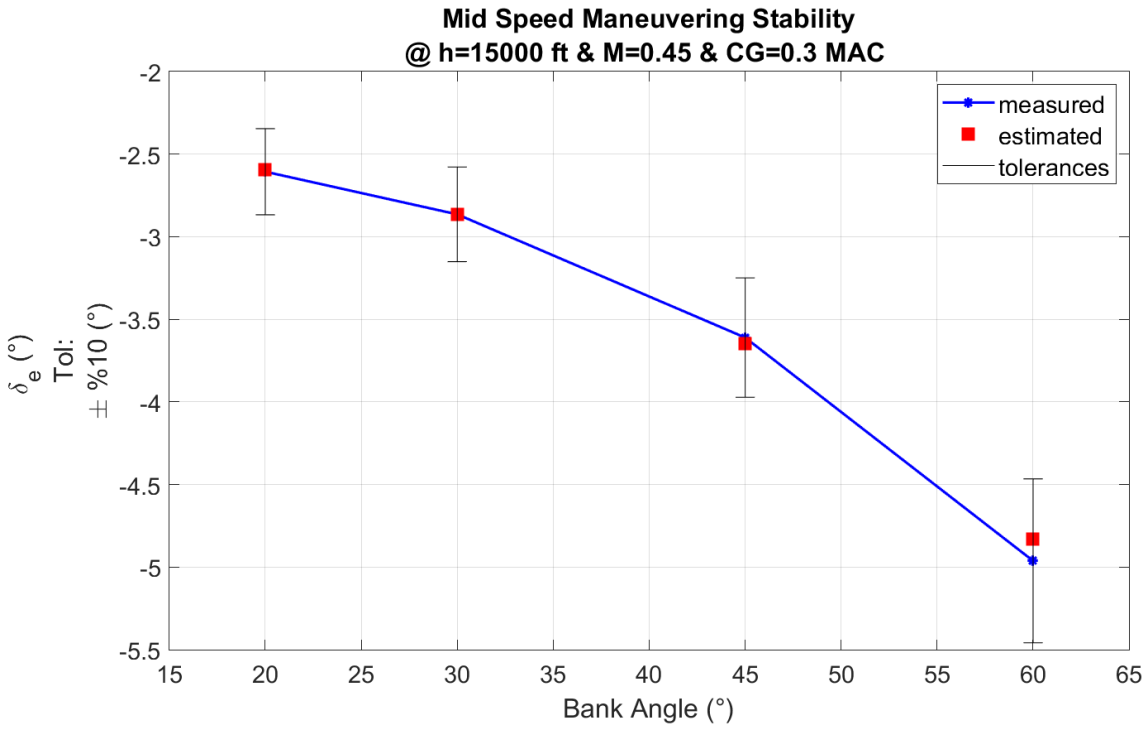


Figure 18. Mid Speed Maneuvering Stability

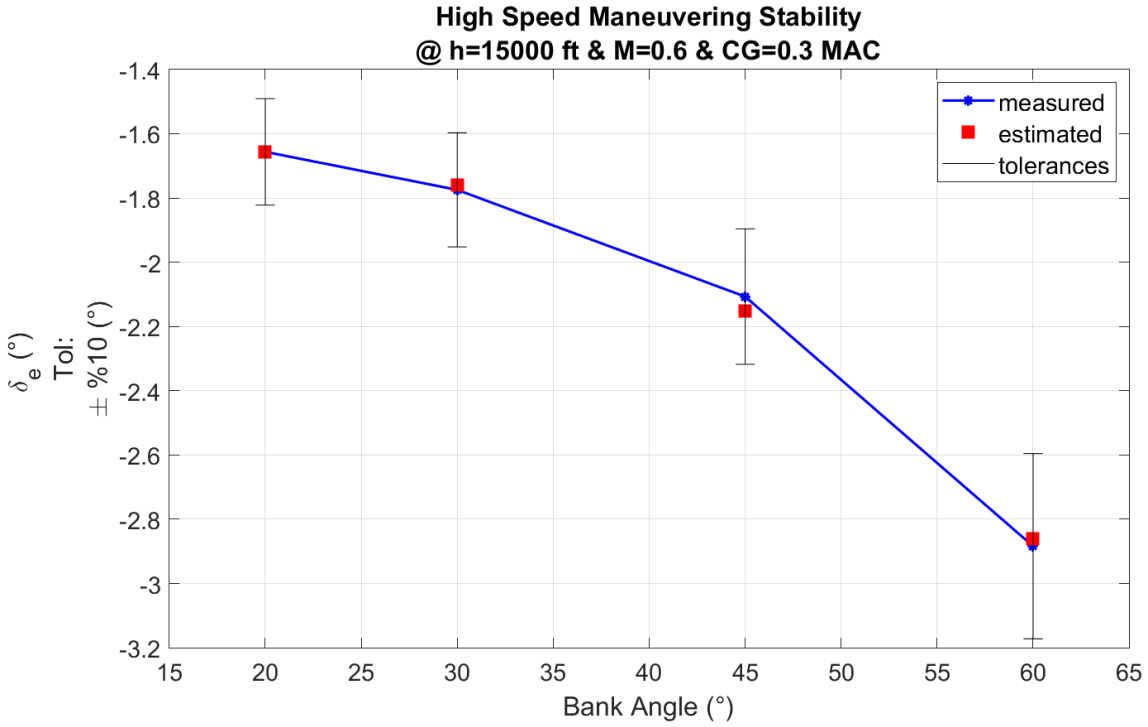


Figure 19. High Speed Maneuvering Stability

The tolerance bounds are lower and upper values which the estimated results are expected to lie between. As it seen in Fig. 17 - 19, the estimated results are in those tolerance bounds. It means our estimated coefficients are compatible the actual aircraft. The estimated results are closer to tolerance bounds, which means at low airspeed, the residual between the estimated and measures is a bit higher.

Longitudinal Static Stability

The objective of this test is to demonstrate that the simulator static longitudinal stability characteristics conform to the aircraft. Longitudinal control command is applied to the airplane in order to get a deviation from trimmed airspeed, and elevator deflection is used to maintain a steady-state condition at different speeds as shown in Fig. 20.

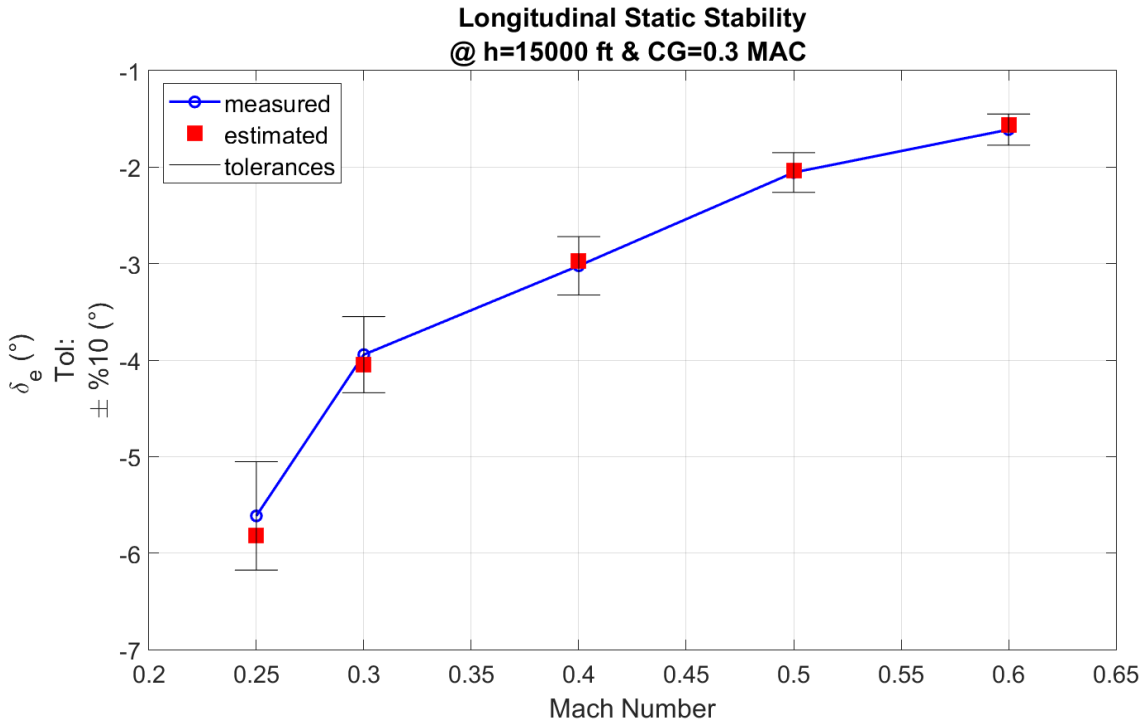


Figure 20. Longitudinal Static Stability

The estimated results are in tolerance bounds. It means our estimated coefficients are compatible with the actual aircraft. The estimated results are closer to tolerance bounds at low airspeed corresponding to high angle of attack values. As we already observed in cross plot of aerodynamic coefficients residual with respect to angle of attach, the residual between the estimated and measured values at high angle of attack is a bit higher.

Short Period Maneuver

Short period dynamics is evaluated by exciting the short period mode under the cruise condition by applying a brief (one second or less) longitudinal control input in one direction then allowing the airplane to freely respond.

The flight measurements with these tolerances define a band within which the model predicted response must lie. For the other variables, particularly those of the cross axis, a qualitative match which shows correct trends, is usually considered as adequate. The model adequacy is quite apparent from the figure. The result is fairly good as seen in Fig. 21.

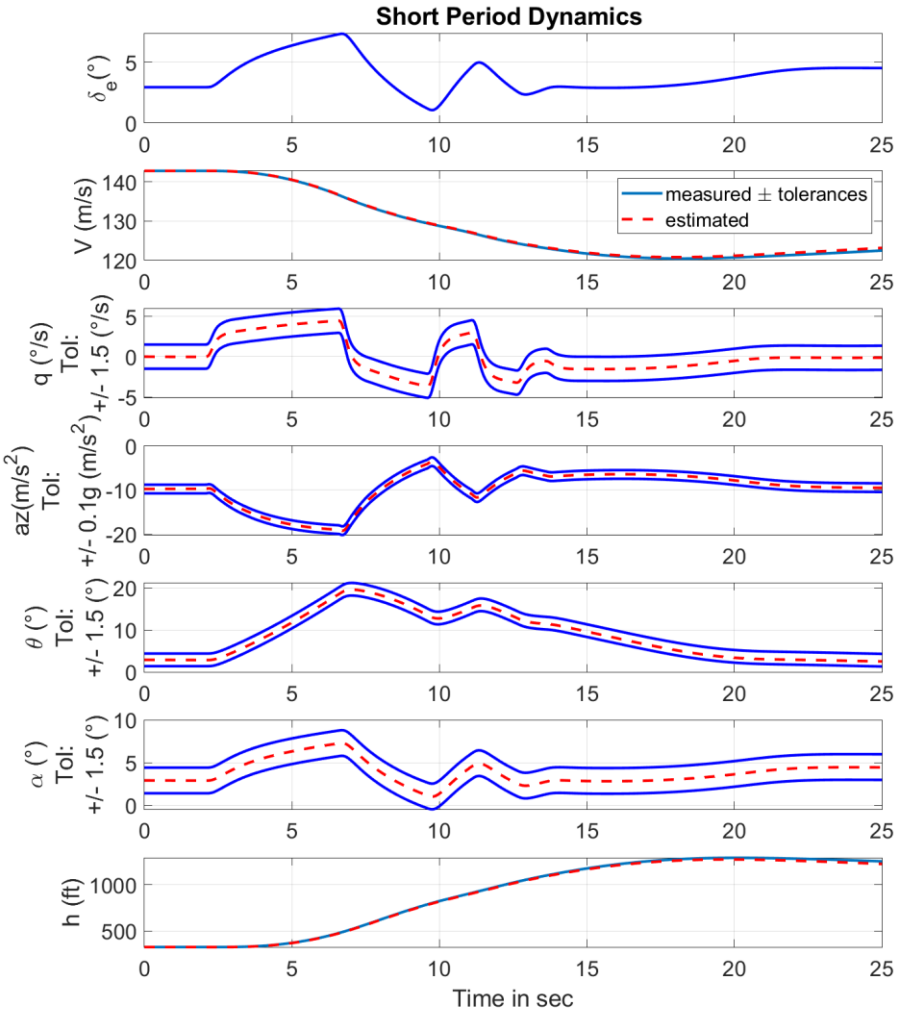


Figure 21. Short period dynamics

The solid lines are obtained from measured data plus/minus the tolerances specified in Table 3, and the dashed line shows the model predicted output, which is quite well within the allowed band for Level-D model fidelity.

References

[1] Jategaonkar, R.V., *Flight Vehicle System Identification: A Time-Domain Methodology*, 2nd ed., AIAA Process in Astronautics and Aeronautics, AIAA, Reston, VA, 2015.

[2] Klein, V., ve Morelli, E.A., *Aircraft System Identification: Theory and Practices*, AIAA Education Series, AIAA, Reston, VA, 2006.

[3] Hess, R.A. "On the use of back propagation with feed-forward neural networks for aerodynamic estimation problem", AIAA Flight Simulation and Technologies, Monterey, CA, 1993.

[4] Linse, D.J. and Stengel, R.F. "Identification of aerodynamic coefficient using computational neural networks," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 6, pp. 1018-1025, 1993.

[5] Youssef, H.M. and Juang, J.C. "Estimation of aerodynamic coefficient using neural networks", AIAA Flight Simulation and Technologies, Monterey, CA, 1993.

[6] Raol, J.R. and Jategaonkar, R.V. "Aircraft parameter estimation using recurrent neural networks – A critical appraisal", AIAA Atmospheric Flight Mechanics Conference, Baltimore, MD, 1995

[7] Raisinghani, S.C., Ghosh, A.K., Kalra, P.K. "Two new techniques for aircraft parameter estimation using neural networks", *The Aeronautical Journal*, Vol. 102, No. 1011, pp. 25-30, 1998.

[8] Singh, S., and Ghosh, A.K., "Estimation of lateral-directional parameters using neural networks based modified delta method", *Aeronautical Journal*, Vol. 111, No. 1124, pp. 659-667, 2007.

[9] Kurt, H.B., Millidere, M., Sezer, E., "Aerodynamic Database Simulation Implementation Based On Neural Network and Neural Network Parameter Selection Using Genetic Algorithms", 8th European Conference for Aeronautics and Space Sciences (EUCASS), 2019.

[10] James, G., Witten D., Hastie T., Tibshirani R., (2013), *Introduction to Statistical Learning with Applications in R*, Springer.

[11] Nguyen, L. and Ogburn, M., "Simulator Study of Stall/PostStall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability", NASA Technical Paper 1538, 1979.

[12] Stevens, B., Lewis, F., and Johnson, E., *Aircraft Control and Simulation Dynamics, Controls Design and Autonomous Systems*, 3 ed., Wiley-Blackwell, 2015.

[13] Millidere, M., Cakin, U., Yigit, T., "Generating Aerodynamic Database from Closed Loop Simulated Flight Data", 8th European Conference for Aeronautics and Space Sciences (EUCASS), 2019.

[14] Hastie, T., Tibshirani R., Friedman J., *The Elements of Statistical Learning – Data Mining, Inference, and Prediction*, Springer, 2009.

[15] Wisnowski, J., Simpson J., Montgomery D., and Runger G., "Resampling methods for variable selection in robust regression", *Journal of Computational Statistics and Data Analysis*, Vol. 43, No. 3, pp. 341-355, 2003

[16] Federal Aviation Administration (FAA), "14 CFR FAR Part 60, Requirements for the Evaluation, Qualification and Maintenance of Flight Simulation Training Devices", 2008.

[17] Royal Aeronautical Society (RAeS), "Aeroplane Flight Simulation Training Device Evaluation Handbook", Vol. 1, 4rd ed., 2009.

[18] Millidere, M., Karaman, U., Uslu, S., Kasnakoglu, C., and Çimen, T., "Newton-Raphson Methods in Aircraft Trim: A Comparative Study", AIAA Aviation Forum, Reno, NV, 2020, forthcoming.