AE4320 System Identification of Aerospace Vehicles

Assignment (2019)

Aerodynamic Model Identification using the Two-Step Approach

Author: Abishek Narasimhan *Student Number:* 4788613

March 15, 2020

AE4320 Assignment:

Aerodynamic Model Identification using the Two-Step Approach

Assignment description:
This project concerns the aircraft aerodynamic model identification using flight test data with the two-step approach learnt in the lecture. Accurate estimation of aircraft position, airspeed body components and attitude with a GPS/IMU/Airdata integrated system for a short range flying region with wind is performed in the first step of the two-step approach applying the Extended or Iterated Extended Kalman Filters EKF/IEKF). The sensor integration structure is defined by using tightly coupled IMU and loosely coupled GPS/Airdata. The IMU will measure the specific force and rotational rate vectors (raw data) of the aircraft, the GPS receiver will provide aircraft position, ground speed earth components and attitude (processed data), and the airdata sensors will offer processed true airspeed, angle of attack, and side slip angle of the aircraft. After the aircraft flight trajectory has been estimated, the aerodynamic model identification will be performed in the second step.

The entire assignment consists of 4 parts. Additional information necessary to complete the assignment is provided in Appendix A.

Part 1: Report introduction (13 points)

You are asked to write a brief introduction (max 2 pages) for your report which should contain the following items:

1. A discussion on the relevance of System Identification to aircraft control & simulation. (**3 points**)
2. A brief explanation of the working principles of state estimation techniques and parameter estimation methods. (**5 points**)
3. A brief explanation of the working principles of advanced system identification methods such as neural networks and multivariate splines. (**5 points**)

Part 2: Data pre-processing (17 points)

You are asked to pre-process the given data into a form that is usable in the state estimation part:

1. Generate aircraft position using simulated airspeed components and assumed wind velocity. Clearly present the results. (**5 points**)
2. Generate GPS, IMU and airdata measurements using Eqs. (5) and (6) using the flight data provided. (**5 points**)
3. Design an integrated GPS/IMU/Airdata navigation system using the accelerometers, rate gyros, GPS receiver and airdata sensors by constructing a navigation model with 18 states (position, velocity, attitude and IMU biases and wind components), 6 inputs and 12 measurements. (**7 points**)

Part 3: Kalman Filter (35 points)

You are asked to design and use an extended or iterated extended Kalman filter on the pre-processed data. Use the information provided in Appendix A in this part.

1. Create an Extended or Iterated Extended Kalman filters (EKF/IEKF) for this aircraft flight path reconstruction problem. Clearly present in the report the different steps in your EKF/IEKF. (**5 points**)
2. Clearly show that the Kalman filter estimates for the accelerometer and gyro biases are correct. (**10 points**)
3. Use your EKF/IEKF to estimate all aircraft states. Clearly show the difference between the 'raw' sensor measurements and the filtered measurements. (**5 points**)
4. Clearly demonstrate how the estimates of the aircrafts states and IMU sensor biases changes if standard deviations for the true airspeed, angle of attack and side slip angle measurement noises are increased by a factor of 10 to 1.0 m/s, 1.0 degrees and 1.0 degrees, respectively. (**5 points**)
5. Prove (numerically) that your Kalman filter functions properly by discussing convergence and filter innovation. (**10 points**)

Part 4: Aerodynamic Model Identification (35 points)

You are asked to identify and validate a simple aerodynamic model using the estimated states and estimated forces and moments from the state estimation part. Use the information provided in Appendix A in this part.

1. Compute the aerodynamic forces and moments from the trajectory and sensor parameters found using the EKF/IEKF. Clearly indicate how you performed the numerical differentiation to calculate the angular acceleration components required for the moment calculations. (**5 points**)
2. Formulate a least squares parameter estimator for the parameters of the aerodynamic model given in the assignment. Clearly indicate the structure of your linear regression model and estimator. (**5 points**)
3. Estimate the parameters of your aerodynamic model. Clearly indicate how (and why) you used the different data files in this process. (**5 points**)
4. Validate your aerodynamic model by using the estimated model output and measured data. Provide a model residual analysis. Additionally, prove the statistical quality of your parameter estimates. Which parameter(s) are you most/less sure about? (**10 points**)
5. Provide an indication which model terms are most dominant, and which have the least influence. Introduce an alternative model structure and compare its validation with the validation given in 4.2. (**10 points**)

Contents

List of Figures

List of Tables

1 | Report Introduction

System Identification is a mathematical framework that is built with statistical methods to study the effect of input-output behaviour of the system or evaluate the dynamics of a system from a set of measured data and creating a mathematical abstraction of the system itself. It is possible because of the conjugal relationship between the physical world and the theoretical knowledge on it is working. When the theoretical model is understood well, based on the data that is measured, various aspects of the system that is not physically measured is now observable because of their mathematical relationships. Like in this assignment with the data that is available from in-flight avionics: accelerometer, gyroscopes, GPS and airdata sensors - the external wind that is incident on the plane can now be quite accurately observed because of the mathematical relationships between all the measured data.

The relevance of system identification and statistical modelling on aircraft flight control and simulation comes in many folds—the first being the ability to describe the unknown input-output relationship of a said system accurately. The idea behind this is a mathematical abstraction that approximates the complicated mathematical relationship or formulation that is acting on any said system. Like for instance, the working of a Brushless DC Motor is quite complicated, and motor controllers ensure the motor can spin at the required commanded RPM (Rotations Per Second) on a set of external disturbances acting on them. The complex interaction of the motor controller and the physical world acts as though they are analogous to water heating to a specific temperature set on a geyser in any given weather to an ignorant observer. This ability to be able to mathematical abstract a complex dynamics of a system stems in the very root of system identification and approximating complex systems to design flight simulators that train professional pilots in "almost" a realistic scenario on various safety-critical flight procedures.

The second is when a physical system is well understood theoretically; then, their real-world performance can be defined quantitatively with a given set of assumptions. The idea behind this is, a theoretical framework usually describes a class of systems and determine their performance qualitatively. However, on involving system-specific constants, the performance of the system can now be defined quantitatively. This is similar to how, the equations of motion that describe the motion of a biplane also collectively describe the motion of any given plane on the basis of strict aerodynamic assumptions. But however, to quantitatively describe accurate models of one subsonic beach plane from that of another, the model and aerodynamic parameters have to be known. The ability to measure this plane specific constants from the flight data is also an important part of system identification and is used in flight simulator models.

The third is the ability to design optimal control methods based on the knowledge of a given system. Many advanced control methods like Nonlinear dynamics Inversion (NDI), Incremental Nonlinear Dynamics Inversion (INDI) and Incremental Nonlinear Control Allocation (INCA) rely heavily on the accurate model of the system. Sometimes when the model is so accurately known and when all required variables can be measured or observed—then based on the inconsistency in the physical measurements to that of the theoretical predictions. In some cases, the error can also be traced back to detect a fault within the physical system. This field of study is known as Fault Detection and Diagnosis (FDD).

The fourth is being able to accurately know or measure physical aircraft states from the aircraft sensor fusion technology; the data measured from flight sensors are noisy and often biased due to very physical properties. Say, if a pilot were to rely on air data alone entirely, then his estimate of aircraft position alone would be drifting to that of the external wind and gusts that is incident on the plane. The sensor fusion and state estimation algorithms designed by statistical models rely on the ability of the sensors to accurately determine the state of the aircraft from listening to data from different noisy and biased sources. Needless to say, as accurate as the measurements get, the better is the estimate of aircraft state say position with respect to earth, on a higher margin of certainty.

On a general level, the basis of state or parameter estimation in the time domain relies on a fundamental idea of making the expected value to match that of the measured value. This error is often associated with an error function or a cost function that is designed by the user. Then error estimated based on this cost function is used to update the model until the estimated model consistently matches the measured system within a small degree of uncertainty. The function that performs this model update is called the update function.

Care is often taken to ensure convexity of this cost function with the minimum value of the curve corresponds to the most optimal value of the parameter that produces the least error over consistent measurements. The idea is that the convex functions have positive and negative slopes around the minimum. Moreover, the algorithms work based on taking small steps down the slope until the minimum is reached. It, however, gets problematic when the

cost functions have many regions of convexity, then the most optimal value is determined at "Global" minima and the one where the update function is stuck while not estimating the optimal value is called as the "Local" minima. This method of descending to optimal value is called as Gradient Descent(GD).

One of the most popular of all cost functions is the square of the error. Many optimal estimation strategies for approximating the non-linear model is linearised to be able to use this cost function, and the most basic description is called the Ordinary Least Squares (OLS) estimation. However, it is assumed that the estimation is based around data with white noise, but more advanced variants are derived based on updating the models based on various energy of noise and are called as Weighted Least Squares (WLS) and Generalised Least Squares (GLS). A digital version of Ordinary Least Squares is more computationally efficient to compute estimations over a large amount of data is called Recursive Least Squares (RLS).

Traditionally, to perform system identification a theoretical knowledge was always required or at least a practical knowledge was required on the working of the system to generalise a given performance. But however, modern methods like Neural Networks (NN) can be used to approximate the mathematical relationships by given set of functions. The assumption of approximating a mathematical function with a many combination of a given set of functions that produce outputs between 0 or 1 and is often compared to that of the working of a biological neural networks, hence gaining the name. These functions are often described in layers and each functional unit is called a neuron and a layer has a set of neurons that are connected to a set in the adjacent layers.

A cost function is now designed and to reduce the error from the neural network model. The cost is then used to update the model and the update function propagates through the model by exploiting a fundamental property called as a chain rule. But however these models should be used with caution as the extrapolations no longer guarantee the performance as that to within the working domain where the model has the least error. And it however not known of the required number of layers and neurons that are required to produce the best estimate of the system. It is crucial that the model is also verified for the over fitting the data, as extrapolating a over fitted model may no longer yield a desired performance.

Sometimes on using the Least Squares Estimator, it gets difficult to model higher order dynamics and also the number of higher order regressors required becomes uncertain. And thus on such data sets, multi-variable splines come in handy as they chop down the dataset to smaller resolutions and the OLS is fit locally on smaller intervals. The resolutions of the data can be in many dimensions. This essentially means OLS is put on smaller domains and the smoothing function is used to have better continuity on fit data. This way when new data-sets get available the regions of the model where the data set is available gets a better fit but it has no global impact on the parameter fit.

In this assignment, a Aerodynamic Model Identification is performed using the Two-Step Approach. In the first step accurate estimation of aircraft position, airspeed body components and attitude with a GPS/IMU/Airdata integrated system for a short range flying with wind is performed. The sensor integration is performed with a tightly coupled IMU data with a loosely coupled GPS/Airdata. In the second step aircraft model identification will be performed linear regression with Ordinary Least Squares (OLS) estimator. This assignment is part of the course AE4320, System Identification of Aerospace Vehicles (2019).

2 | Data pre-processing

2.1 | Generate Aircraft Position with simulated Windspeed Components

The wind speed was assumed to be (10, 6, 1) m/2

$$V_e = V_{gps} + V_{wind}$$

$$X_e = \int_0^t V_e dt = X_e[k] = X_e[k-1] + V_e[k] * (t[k] - t[k-1])$$

2.1.1 Generate GPS, IMU, Airdata Measurements

GPS Measurement

$$\begin{aligned}
&x_{GPS} = x \\
&x_{GPS_m}(t_i) = x_{GPS}(t_i) + \underline{\nu}_x(t_i) \\
&y_{GPS} = y \\
&y_{GPS_m}(t_i) = y_{GPS}(t_i) + \underline{\nu}_y(t_i) \\
&z_{GPS} = z \\
&z_{GPS_m}(t_i) = z_{GPS}(t_i) + \underline{\nu}_z(t_i) \\
&u_{GPS} = [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\cos\Psi - (v\cos\phi - w\sin\phi)\sin\Psi + W_{x_E} \\
&u_{GPS_m}(t_i) = u_{GPS}(t_i) + \underline{\nu}_u(t_i) \\
&v_{GPS} = [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\sin\Psi + (v\cos\phi - w\sin\phi)\cos\Psi + +W_{y_E} \\
&v_{GPS_m}(t_i) = v_{GPS}(t_i) + \underline{\nu}_v(t_i) \\
&w_{GPS} = -u\sin\theta + (v\sin\phi + w\cos\phi)\cos\theta + +W_{z_E} \\
&u_{GPS_m}(t_i) = u_{GPS}(t_i) + \underline{\nu}_u(t_i) \\
&\phi_{GPS} = \phi \\
&\phi_{GPS_m}(t_i) = \phi_{GPS}(t_i) + \underline{\nu}_\phi(t_i) \\
&\theta_{GPS} = \theta \\
&\theta_{GPS_m}(t_i) = \theta_{GPS}(t_i) + \underline{\nu}_\theta(t_i) \\
&\psi_{GPS} = \psi \\
&\psi_{GPS_m}(t_i) = \psi_{GPS}(t_i) + \underline{\nu}_\psi(t_i)
\end{aligned} \tag{2.1}$$

The flight data is generated with the following noise and wind:

$$\begin{aligned}
&\sigma_{\underline{\nu}_x} = 10\ m, \quad \sigma_{\underline{\nu}_u} = 0.1\ m/s, \quad \sigma_{\underline{\nu}_\phi} = 0.1\ deg, \quad W_{x_E} = 10\ m/s \\
&\sigma_{\underline{\nu}_y} = 10\ m, \quad \sigma_{\underline{\nu}_v} = 0.1\ m/s, \quad \sigma_{\underline{\nu}_\theta} = 0.1\ deg, \quad W_{y_E} = 6\ m/s \\
&\sigma_{\underline{\nu}_z} = 10\ m, \quad \sigma_{\underline{\nu}_w} = 0.1\ m/s, \quad \sigma_{\underline{\nu}_\psi} = 0.1\ deg, \quad W_{z_E} = 1\ m/s
\end{aligned}$$

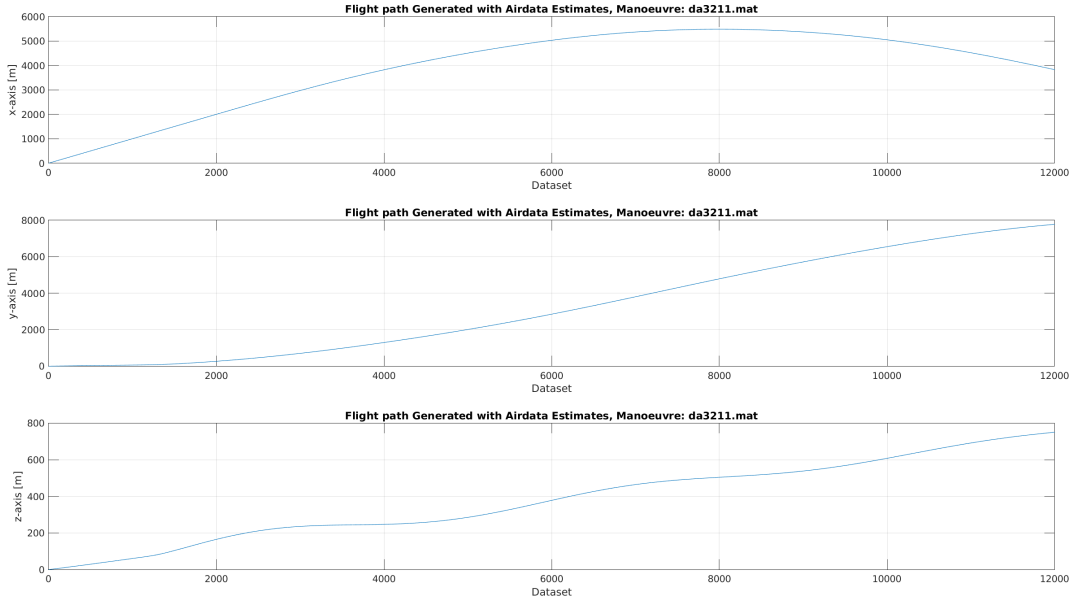

Figure 2.1: Regenerated Flight path with external wind. Dataset: *da*3211.*mat*

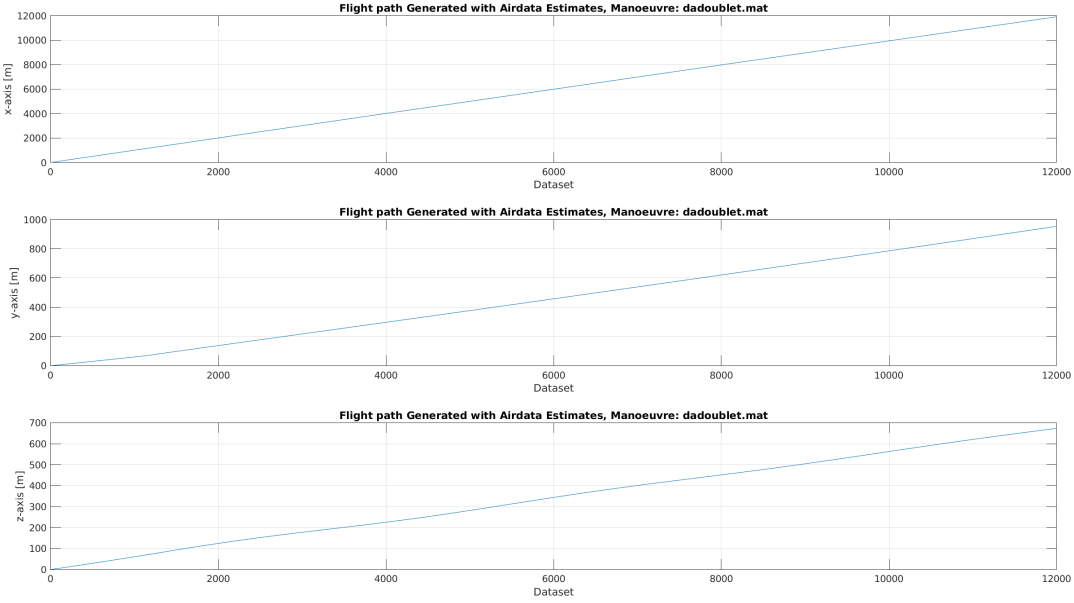

Figure 2.2: Regenerated Flight path with external wind. Dataset: *dadoublet.mat*

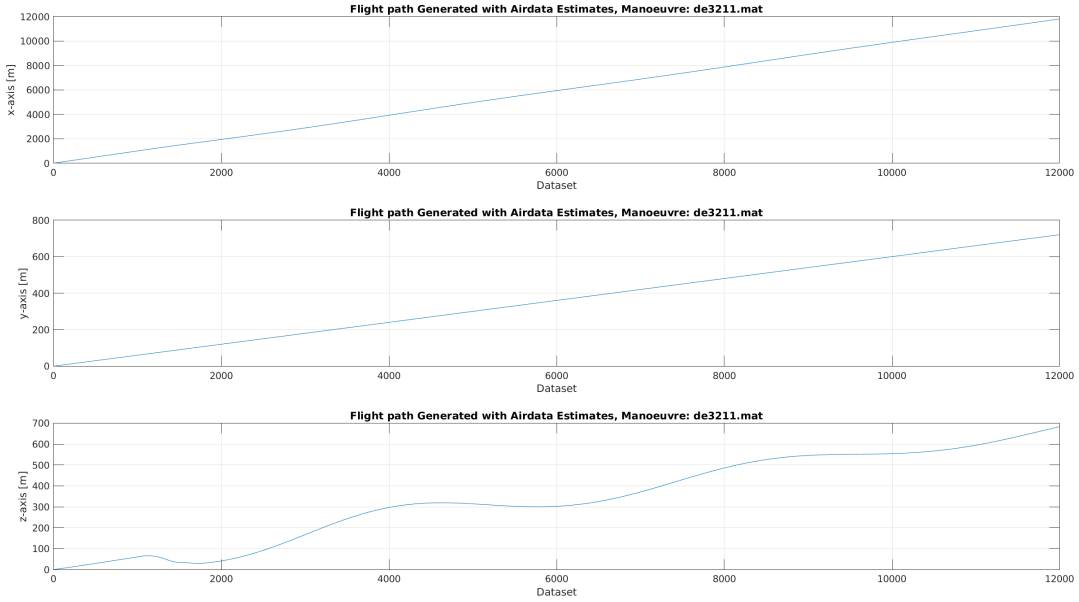

Figure 2.3: Regenerated Flight path with external wind. Dataset: *de*3211.*mat*

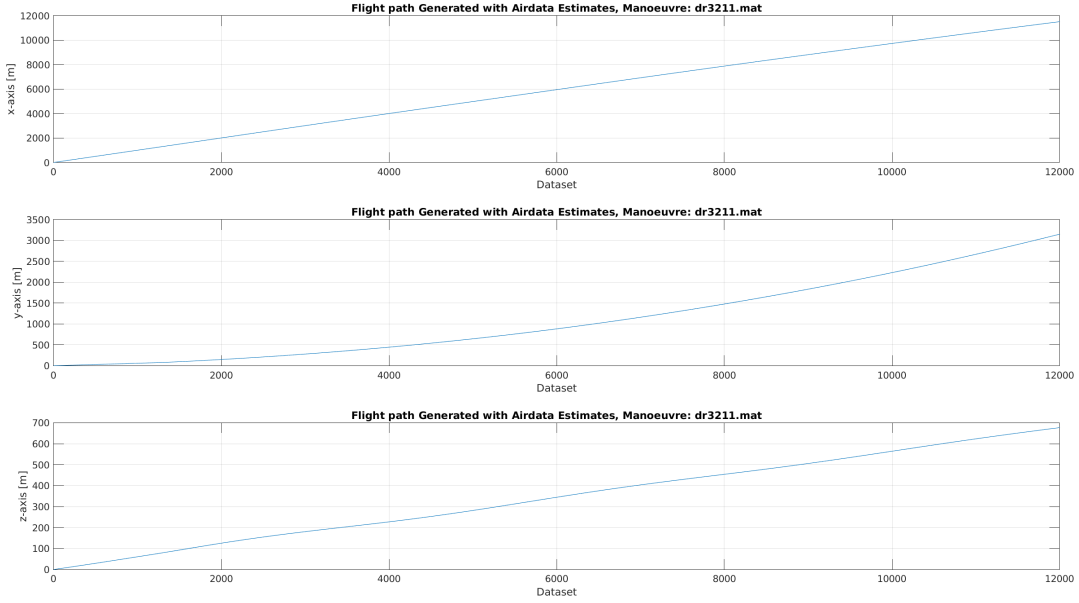

Figure 2.4: Regenerated Flight path with external wind. Dataset: *dr*3211.*mat*

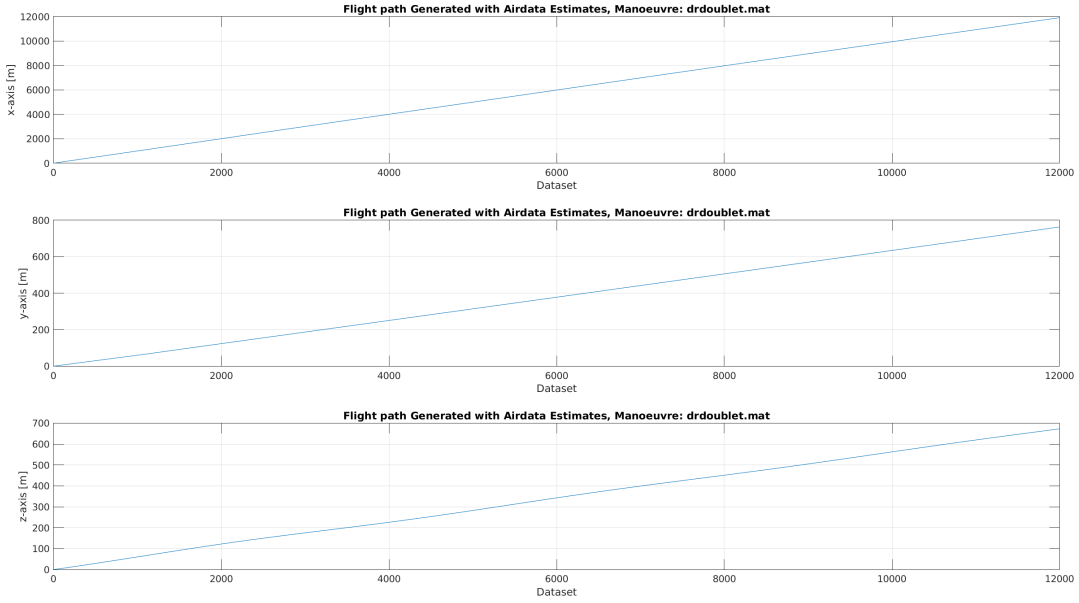

Figure 2.5: Regenerated Flight path with external wind. Dataset: *drdoublet.mat*

IMU Measurement

$$\begin{aligned} A_{x_m} &= A_x + \lambda_x + \underline{w}_x \\ A_{y_m} &= A_y + \lambda_y + \underline{w}_y \\ A_{z_m} &= A_z + \lambda_z + \underline{w}_z \\ p_m &= p_x + \lambda_p + \underline{w}_p \\ q_m &= q_x + \lambda_q + \underline{w}_q \\ r_m &= r_x + \lambda_r + \underline{w}_r \end{aligned} \tag{2.2}$$

Complete IMU model is generated with the following bias and noise of standard deviation:

$$\begin{aligned} \lambda_x &= 0.001\ m/s^2, \quad \sigma_{\underline{w}_x} = 0.001\ m/s^2 \\ \lambda_y &= 0.001\ m/s^2, \quad \sigma_{\underline{w}_y} = 0.001\ m/s^2 \\ \lambda_z &= 0.001\ m/s^2, \quad \sigma_{\underline{w}_z} = 0.001\ m/s^2 \\ \lambda_p &= 0.001\ deg/s, \quad \sigma_{\underline{w}_p} = 0.001\ deg/s \\ \lambda_q &= 0.001\ deg/s, \quad \sigma_{\underline{w}_q} = 0.001\ deg/s \\ \lambda_r &= 0.001\ deg/s, \quad \sigma_{\underline{w}_r} = 0.001\ deg/s \end{aligned}$$

Airdata Measurement

$$\begin{aligned} V_{tas} &= \sqrt{u^2 + v^2 + w^2} \\ V_{tas_m} &= V_{tas}(t_i) + \underline{\nu}_v(t_i) \\ \alpha &= \arctan \frac{w}{u} \\ \alpha_m &= \alpha(t_i) + \underline{\nu}_\alpha(t_i) \\ \beta &= \arctan \frac{v}{\sqrt{u^2 + w^2}} \\ \beta_m &= \beta(t_i) + \underline{\nu}_\beta(t_i) \end{aligned} \tag{2.3}$$

The following noise was used:

$$\sigma_{\underline{\nu}_v} = 0.1m/s, \quad \sigma_{\underline{\nu}_\alpha} = 0.1deg, \quad \sigma_{\underline{\nu}_\beta} = 0.1deg$$

Noise Model

$$\begin{aligned}
&E\{\underline{w}(t_i)\} = 0 \\
&E\{\underline{w}(t_i)\underline{w}^T(t_i)\} = Q\delta_{ij}; \quad Q = diag(\sigma^2_{\underline{w}_x}, \sigma^2_{\underline{w}_y}, \sigma^2_{\underline{w}_z}, \sigma^2_{\underline{w}_p}, \sigma^2_{\underline{w}_q}, \sigma^2_{\underline{w}_r}) \\
&E\{\underline{\nu}(t_i)\} = 0 \\
&E\{\underline{\nu}(t_i)\underline{\nu}^T(t_i)\} = R\delta_{ij}; \quad R = diag(\sigma^2_{\underline{\nu}_u}, \sigma^2_{\underline{\nu}_v}, \sigma^2_{\underline{\nu}_w}, \sigma^2_{\underline{\nu}_x}, \sigma^2_{\underline{\nu}_y}, \sigma^2_{\underline{\nu}_z}, \sigma^2_{\underline{\nu}_\phi}, \sigma^2_{\underline{\nu}_\theta}, \sigma^2_{\underline{\nu}_\psi}, \sigma^2_{\underline{\nu}_v}, \sigma^2_{\underline{\nu}_\alpha}, \sigma^2_{\underline{\nu}_\beta})
\end{aligned} \tag{2.4}$$

Assume the input and output noise are not correlated: $E\{\underline{w}(t_i)\underline{\nu}^T(t_i)\} = 0$

Here, $[u, v, w]$ measure velocities along $[x, y, z]$ axis in body reference frame. $\underline{\nu}$ represents noise in output measurement and $\underline{w}$ measures noise in input measurement. The subscript m represents generated measurement data to be used in the EKF.

2.2 | Design Integrated State Equations for Kalman Filter Design

Tha state equations can be written as:

$$\begin{aligned}
\dot{u} &= (A_{x_m} - \lambda_x - \underline{w}_x) - g\sin\theta + (r_m - \lambda_r - \underline{w}_r)v - (q_m - \lambda_q - \underline{w}_q)w \\
\dot{v} &= (A_{y_m} - \lambda_y - \underline{w}_y) - g\cos\theta\sin\phi + (p_m - \lambda_p - \underline{w}_p)w - (r_m - \lambda_r - \underline{w}_r)u \\
\dot{w} &= (A_{z_m} - \lambda_z - \underline{w}_z) - g\cos\theta\cos\phi + (q_m - \lambda_q - \underline{w}_q)u - (p_m - \lambda_p - \underline{w}_p)v \\
\dot{x} &= [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\cos\Psi - (v\cos\phi - w\sin\phi)\sin\Psi + W_{x_E} \\
\dot{y} &= [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\sin\Psi + (v\cos\phi - w\sin\phi)\cos\Psi + W_{y_E} \\
\dot{z} &= -u\sin\theta + (v\sin\phi + w\cos\phi)\cos\theta + W_{z_E} \\
\dot{\phi} &= (p_m - \lambda_p - \underline{w}_p) + (q_m - \lambda_q - \underline{w}_q)\sin\phi\tan\theta + (r_m - \lambda_r - \underline{w}_r)\cos\phi\tan\theta \\
\dot{\theta} &= (q_m - \lambda_q - \underline{w}_q)\cos\phi - (r_m - \lambda_r - \underline{w}_r)\sin\phi \\
\dot{\Psi} &= (q_m - \lambda_q - \underline{w}_q)\frac{\sin\phi}{\cos\theta} + (r_m - \lambda_r - \underline{w}_r)\frac{\cos\phi}{\cos\theta} \\
\dot{\lambda}_x &= 0 \\
\dot{\lambda}_y &= 0 \\
\dot{\lambda}_z &= 0 \\
\dot{\lambda}_p &= 0 \\
\dot{\lambda}_q &= 0 \\
\dot{\lambda}_r &= 0 \\
\dot{W}_{x_E} &= 0 \\
\dot{W}_{y_E} &= 0 \\
\dot{W}_{z_E} &= 0
\end{aligned} \tag{2.5}$$

State Vector: $X = [u, v, w, x, y, z, \phi, \theta, \psi, \lambda_x, \lambda_y, \lambda_z, \lambda_p, \lambda_q, \lambda_r, W_{x_E}, W_{y_E}, W_{z_E}]$

The observation could be formulated as:

$$
\begin{aligned}
u_{GPS_m} &= [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\cos\Psi - (v\cos\phi - w\sin\phi)\sin\Psi + W_{x_E} + \nu_u \\
v_{GPS_m} &= [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\sin\Psi + (v\cos\phi - w\sin\phi)\cos\Psi + W_{y_E} + \nu_v \\
w_{GPS_m} &= -u\sin\theta + (v\sin\phi + w\cos\phi)\cos\theta + W_{z_E} + \nu_w \\
x_{GPS_m} &= x + \nu_x \\
y_{GPS_m} &= y + \nu_y \\
z_{GPS_m} &= z + \nu_z \\
\phi_{GPS_m} &= \phi + \nu_\phi \\
\theta_{GPS_m} &= \theta + \nu_\theta \\
\psi_{GPS_m} &= \psi + \nu_\psi \\
V_{tas_m} &= \sqrt{u^2 + v^2 + w^2} + \nu_v \\
\alpha_m &= \arctan\frac{w}{u} + \nu_\alpha \\
\beta_m &= \arctan\frac{v}{\sqrt{u^2 + w^2}} + \nu_\beta
\end{aligned}
\tag{2.6}
$$

The above equations 2.5, 2.6 can be written in form of:

$$
\begin{aligned}
\dot{\vec{x}} &= f[\vec{x}, \vec{u}_m] + G[\vec{x}]\underline{\vec{w}} \\
\underline{z}_m &= h[\vec{x}, \vec{u}_m] + \underline{\vec{\nu}}
\end{aligned}
\tag{2.7}
$$

where:

$$
\begin{aligned}
\vec{x} &= [u, v, w, x, y, z, \phi, \theta, \psi, \lambda_x, \lambda_y, \lambda_z, \lambda_p, \lambda_q, \lambda_r, W_{x_E}, W_{y_E}, W_{z_E}]^T \\
\vec{u}_m &= [A_{x_m}, A_{y_m}, A_{z_m}, p_m, q_m, r_m]^T \\
\underline{\vec{w}} &= [\underline{w}_x, \underline{w}_y, \underline{w}_z, \underline{w}_p, \underline{w}_q, \underline{w}_r]^T \\
\vec{z}_m &= [u_{GPS_m}, v_{GPS_m}, w_{GPS_m}, x_{GPS_m}, y_{GPS_m}, z_{GPS_m}, \phi_{GPS_m}, \theta_{GPS_m}, \psi_{GPS_m}, V_{tas_m}, \alpha_m, \beta_m]^T \\
\underline{\vec{\nu}} &= [\underline{\nu}_u, \underline{\nu}_v, \underline{\nu}_w, \underline{\nu}_x, \underline{\nu}_y, \underline{\nu}_z, \underline{\nu}_\phi, \underline{\nu}_\theta, \underline{\nu}_\psi, \underline{\nu}_v, \underline{\nu}_\alpha, \underline{\nu}_\beta]^T
\end{aligned}
$$

and so

$$
f[\vec{x}, \vec{u}_m] = \begin{bmatrix}
(A_{x_m} - \lambda_x) - g\sin\theta + (r_m - \lambda_r -)v - (q_m - \lambda_q)w \\
(A_{y_m} - \lambda_y) - g\cos\theta\sin\phi + (p_m - \lambda_p)w - (r_m - \lambda_r)u \\
(A_{z_m} - \lambda_z) - g\cos\theta\cos\phi + (q_m - \lambda_q)u - (p_m - \lambda_p)v \\
[u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\cos\Psi - (v\cos\phi - w\sin\phi)\sin\Psi + W_{x_E} \\
[u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\sin\Psi + (v\cos\phi - w\sin\phi)\cos\Psi + W_{y_E} \\
-u\sin\theta + (v\sin\phi + w\cos\phi)\cos\theta + W_{z_E} \\
(p_m - \lambda_p) + (q_m - \lambda_q)\sin\phi\tan\theta + (r_m - \lambda_r)\cos\phi\tan\theta \\
(q_m - \lambda_q)\cos\phi - (r_m - \lambda_r)\sin\phi \\
(q_m - \lambda_q)\frac{\sin\phi}{\cos\theta} + (r_m - \lambda_r)\frac{\cos\phi}{\cos\theta} \\
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}
$$

$$h[\vec{x}, \vec{u}_m] = \begin{bmatrix} [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\cos\Psi - (v\cos\phi - w\sin\phi)\sin\Psi + W_{x_E} \\ [u\cos\theta + (v\sin\phi + w\cos\phi)\sin\theta]\sin\Psi + (v\cos\phi - w\sin\phi)\cos\Psi + W_{y_E} \\ -u\sin\theta + (v\sin\phi + w\cos\phi)\cos\theta + W_{z_E} \\ x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ \sqrt{u^2 + v^2 + w^2} \\ \arctan\frac{w}{u} \\ \arctan\frac{v}{\sqrt{u^2+w^2}} \end{bmatrix}$$

$$G[\vec{x}] = \begin{bmatrix} -1 & 0 & 0 & 0 & w & -v \\ 0 & -1 & 0 & -w & 0 & u \\ 0 & 0 & -1 & v & -u & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -\sin(\phi)\tan(\theta) & -\cos(\phi)\tan(\theta) \\ 0 & 0 & 0 & 0 & -\cos(\phi) & \sin(\phi) \\ 0 & 0 & 0 & 0 & -\frac{\sin(\phi)}{\cos(\theta)} & -\frac{\cos(\phi)}{\cos(\theta)} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3 | Kalman Filter

3.1 | Implementation of Extended Kalman Filter

One Step-ahead Prediction

$$\vec{x}_{k+1|k} = \vec{x}_{k|k} + \int_{t=t_k}^{t_{k+1}} f[\vec{x}(t), u_m(t), t]dt \tag{3.1}$$

Range-Kutta 4 (Numerical Integration):

$$\begin{aligned}
&\vec{x}_{k+1|k} = \vec{x}_{k|k} + \frac{1}{6}(l_1 + l_2 + l_3 + l_4) \\
&h = t_{k+1} - t_k \\
&l_1 = hf[\vec{x_{k|k}}, u_m(t_k), t_k] \\
&l_2 = hf[\vec{x_{k|k}} + \frac{l_1}{2}, u_m(t_k), t_k + \frac{h}{2}] \\
&l_3 = hf[\vec{x_{k|k}} + \frac{l_2}{2}, u_m(t_k), t_k + \frac{h}{2}] \\
&l_4 = hf[\vec{x_{k|k}} + l_3, u_m(t_k), t_k + h]
\end{aligned}$$

Jacobian Calculation

$$\begin{aligned}
\mathbf{F}_x(\bullet) &= \frac{\delta}{\delta \vec{x}} f[\vec{x_{k|k}}, u_m(t_k), t_k] \\
\mathbf{B}_u(\bullet) &= \frac{\delta}{\delta \vec{u}_m} f[\vec{x_{k|k}}, u_m(t_k), t_k] \\
\mathbf{H}_x(\bullet) &= \frac{\delta}{\delta \vec{x}} h[\vec{x_{k|k}}, u_m(t_k), t_k]
\end{aligned} \tag{3.2}$$

The jacobians were generated using the symbolic toolbox $MATLAB$ with given state equations written in symbolic variables.

State Transition and Input Matrix Discretisation

$$\begin{aligned}
[\Phi_{k+1|k}(\bullet), \Psi_{k+1|k}(\bullet)] &= c2d(\mathbf{F}_x(\bullet), \mathbf{B}_u(\bullet), (t_{k+1} - t_k)] \\
[\Phi_{k+1|k}(\bullet), \Gamma_{k+1|k}(\bullet)] &= c2d(\mathbf{F}_x(\bullet), \mathbf{G}_x(\bullet), (t_{k+1} - t_k)]
\end{aligned} \tag{3.3}$$

$MATLAB$ c2d function was used to discretise the ssystem with "Zero Order Hold" method.

Covariance Matrix of State Prediction Error

$$\mathbf{P}_{k+1|k}(\bullet) = \Phi_{k+1|k}(\bullet)\mathbf{P}_{k|k}\Phi^T_{k+1|k}(\bullet) + \Gamma_{k+1|k}(\bullet)\mathbf{Q}(\bullet)\Gamma^T_{k+1|k}(\bullet) \tag{3.4}$$

Kalman Gain Calculation

$$\mathbf{K}_{k+1}(\bullet) = \mathbf{P}_{k+1|k}(\bullet)\mathbf{H}^T_x(\bullet)[\mathbf{H}_x(\bullet)\mathbf{P}_{k+1|k}(\bullet)\mathbf{H}^T_x(\bullet) + \mathbf{R}]^{-1} \tag{3.5}$$

Measurement Update

$$\vec{x}_{k+1|k+1} = \vec{x}_{k+1|k} + \mathbf{K}_{k+1}(\bullet)[z_{k+1} - h[\vec{x}_{k+1|k}, u_m(t_{k+1}), t_{k+1}]] \tag{3.6}$$

Covariance Error matrix of State Estimation Error

$$\mathbf{P}_{k+1|k+1} = [\mathbf{I} - [\mathbf{K}_{k+1}(\bullet)\mathbf{H}_x(\bullet)]\mathbf{P}_{k+1|k}(\bullet) \tag{3.7}$$

3.2 | Estimation Results

3.2.1 Sensor bias and wind Estimates

Refer the following figures: Bias Estimate: A.2, B.2, C.2, D.2, E.2 Wind Estimate: A.3, B.3, C.3, D.3, E.3

3.2.2 Measured Data and Filtered Data

Refer the following: EKF: A.1, B.1, C.1, D.1, E.1

3.2.3 State and Bias Estimate after adding noise in Airdata Sensors

Bias Estimate: A.6, B.6, C.6, D.6, E.6 Wind Estimate: A.7, B.7, C.7, D.7, E.7 EKF: A.5, B.5, C.5, D.5, E.5

3.3 | Proof of Convergence

The residuals where estimated,

$$Residual = Measured\ Data - Filtered\ Data \tag{3.8}$$

If the Kalman filter converged then the data required from the measured values is extracted. The residuals merely remain as the white noise that was initially added. So, two tests are conduced:

- Check if Residual is a Gaussian Distribution (with 5% significance).
- Check if the Residual belongs to the same distribution to that of the initially added noise (with 5% significance).

The above tests would have to be performed after the initial data and from when the Kalman filter is assumed to have converged. So the analysis is performed after the first 20[s]. And as a added bonus, ksdensity test was performed over the noise that was added to the system initially and to the residuals and they overlayed plots can be found in figures: A.4, B.4, C.4, D.4, E.4. All of the above mentioned plots were Gaussian and can also be verified by the test done in the next section.

Gaussian Distribution Check

Check for Gaussian Distribution is done by: Kolmogorov-Smirnov test - Marsaglia Method. [1]

Mean, Variance Similarity test

Ztest- The mean and variance used for testing was that of the noise that was initially added to the system. This was done because it is not possible to generate a perfect white noise.
The result of tests are given in tables: A.1, B.1, C.1, D.1, E.1
And it clearly be seen that when the mean is not fully accurate then the plot in the figures are slightly biased from the noise that was input to the system.

4 | Aerodynamic Model Identification

4.1 | Calculation of Aerodynamic Forces and Moments

$$\begin{aligned} A_x &= A_{x_m} - \lambda_x - \underline{w}_x \\ A_y &= A_{y_m} - \lambda_y - \underline{w}_y \\ A_z &= A_{z_m} - \lambda_z - \underline{w}_z \\ p_x &= p_m - \lambda_p - \underline{w}_p \\ q_x &= q_m - \lambda_q - \underline{w}_q \\ r_x &= r_m - \lambda_r - \underline{w}_r \end{aligned} \tag{4.1}$$

$\lambda_x, \lambda_y, \lambda_z, \lambda_p, \lambda_q, \lambda_r$ is directly estimated as a state in the Kalman filter. And the white noise added at in put is already removed.

$$\begin{aligned} C_x &= \frac{X}{1/2\rho V^2 S} = \frac{mA_x}{1/2\rho V^2 S} \\ C_y &= \frac{Y}{1/2\rho V^2 S} = \frac{mA_y}{1/2\rho V^2 S} \\ C_z &= \frac{Z}{1/2\rho V^2 S} = \frac{mA_z}{1/2\rho V^2 S} \\ C_l &= \frac{L}{1/2\rho V^2 Sb} = \frac{\dot{p}I_{xx} + qr(I_{zz} - I_{yy}) - (pq + \dot{r})I_{xz}}{1/2\rho V^2 Sb} \\ C_m &= \frac{M}{1/2\rho V^2 Sc} = \frac{\dot{q}I_{yy} + rp(I_{xx} - I_{zz}) + (p^2 - r^2)I_{xz}}{1/2\rho V^2 Sc} \\ C_n &= \frac{N}{1/2\rho V^2 Sb} = \frac{\dot{r}I_{zz} + pq(I_{yy} - I_{xx}) + (qr - \dot{p})I_{xz}}{1/2\rho V^2 Sb} \end{aligned} \tag{4.2}$$

Angular acceleration is calculated based on 2 point formula as:

$$\begin{aligned} \dot{p}(t) &= \frac{p(t+dt) - p(t-dt)}{2dt} \\ \dot{q}(t) &= \frac{q(t+dt) - q(t-dt)}{2dt} \\ \dot{r}(t) &= \frac{r(t+dt) - r(t-dt)}{2dt} \end{aligned} \tag{4.3}$$

4.2 | Ordinary Least Squares(OLS) Estimator

Force and Moments Model

$$\begin{aligned} C_x &= C_{x_0} + C_{x_\alpha}\alpha + C_{x_{\alpha^2}}\alpha^2 + C_{x_q}\frac{qc}{V} + C_{x_{\delta_e}}\delta_e + C_{x_{T_c}}T_c \\ C_y &= C_{y_0} + C_{y_\beta}\beta + C_{y_p}\frac{pb}{2V} + C_{y_r}\frac{rb}{2V} + C_{y_{\delta_a}}\delta_a + C_{y_{\delta_r}}\delta_r \\ C_z &= C_{z_0} + C_{z_\alpha}\alpha + C_{z_q}\frac{qc}{V} + C_{z_{\delta_e}}\delta_e + C_{z_{T_c}}T_c \\ C_l &= C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \\ C_m &= C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{qc}{V} + C_{m_{\delta_e}}\delta_e + C_{m_{T_c}}T_c \\ C_n &= C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \end{aligned} \tag{4.4}$$

And they are rewritten as follows:

$$
\begin{aligned}
C_x &= [1, \alpha, \alpha^2, \frac{qc}{V}, \delta_e T_c][C_{x_0}, C_{x_\alpha}, C_{x_{\alpha^2}}, C_{x_q}, C_{x_{\delta_e}} C_{x_{T_c}}]^T \\
C_y &= [1, \beta, \frac{pb}{2V}, \frac{rb}{2V}, \delta_a, \delta_r][C_{y_0}, C_{y_\beta}, C_{y_p}, C_{y_r}, C_{y_{\delta_a}}, C_{y_{\delta_r}}]^T \\
C_z &= [1, \alpha, \frac{qc}{V}, \delta_e T_c][C_{z_0}, C_{z_\alpha}, C_{z_q}, C_{z_{\delta_e}} C_{z_{T_c}}]^T \\
C_l &= [1, \beta, \frac{pb}{2V}, \frac{rb}{2V}, \delta_a, \delta_r][C_{l_0}, C_{l_\beta}, C_{l_p}, C_{l_r}, C_{l_{\delta_a}}, C_{l_{\delta_r}}]^T \\
C_m &= [1, \alpha, \frac{qc}{V}, \delta_e T_c][C_{m_0}, C_{m_\alpha}, C_{m_q}, C_{m_{\delta_e}} C_{m_{T_c}}]^T \\
C_n &= [1, \beta, \frac{pb}{2V}, \frac{rb}{2V}, \delta_a, \delta_r][C_{n_0}, C_{n_\beta}, C_{n_p}, C_{n_r}, C_{n_{\delta_a}}, C_{n_{\delta_r}}]^T
\end{aligned}
\tag{4.5}
$$

All the above Equations are in the form of: $y = A(x)\hat{\theta}_{OLS} + \epsilon$
And y represents the model output and are the values calculated form the previous section and $A(x)$ is the values of the independent variables, $\hat{\theta}_{OLS}$ is the model parameters with ϵ as the residual. $\hat{\theta}_{OLS}$ can be calculated as:

$$\hat{\theta}_{OLS} = (A^T(x)A(x))^{-1}A^T(x)y + \epsilon \tag{4.6}$$

4.3 | Parameter Estimation and Residual Analysis

Data Set Management

The data given is for deflection of elevator, airelons and rudder. The deflection of elevator actuates the longitudinal dynamics and lateral dynamics is actuated by the lateral dynamics.

- Longitudinal dynamics: *de*3211.*mat*
- Lateral dynamics: *da*3211.*mat*, *dadoublet.mat*, *dr*3211.*mat*, *drdoublet.mat*

This distinction was made because the elevator dynamics was completely independent from that of the ailerons and rudders in lateral dynamics.

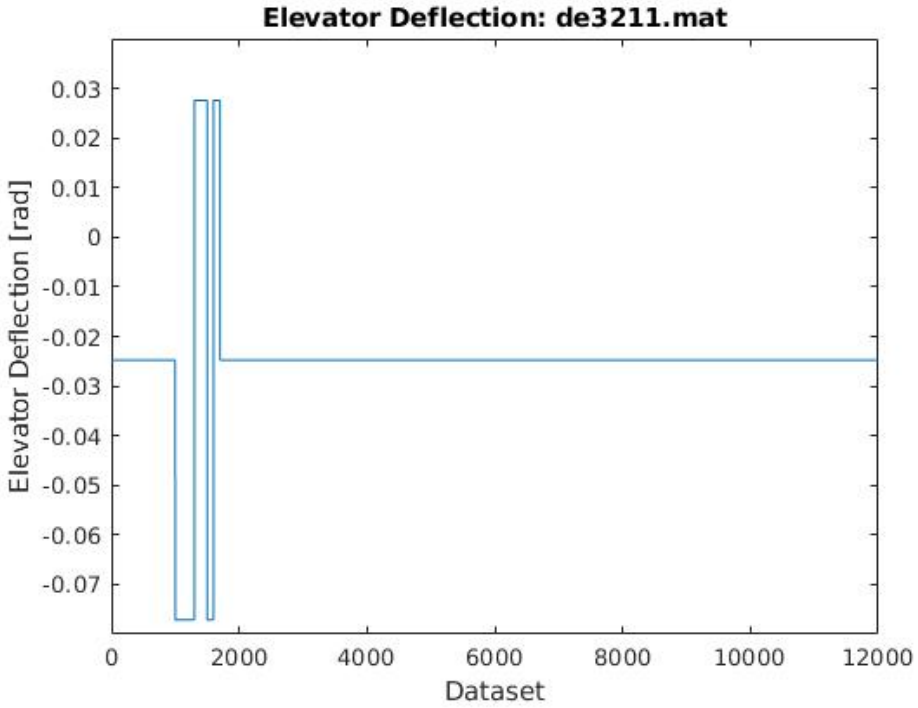

Figure 4.1: Elevator Deflection. Dataset: *de*3211.*mat*

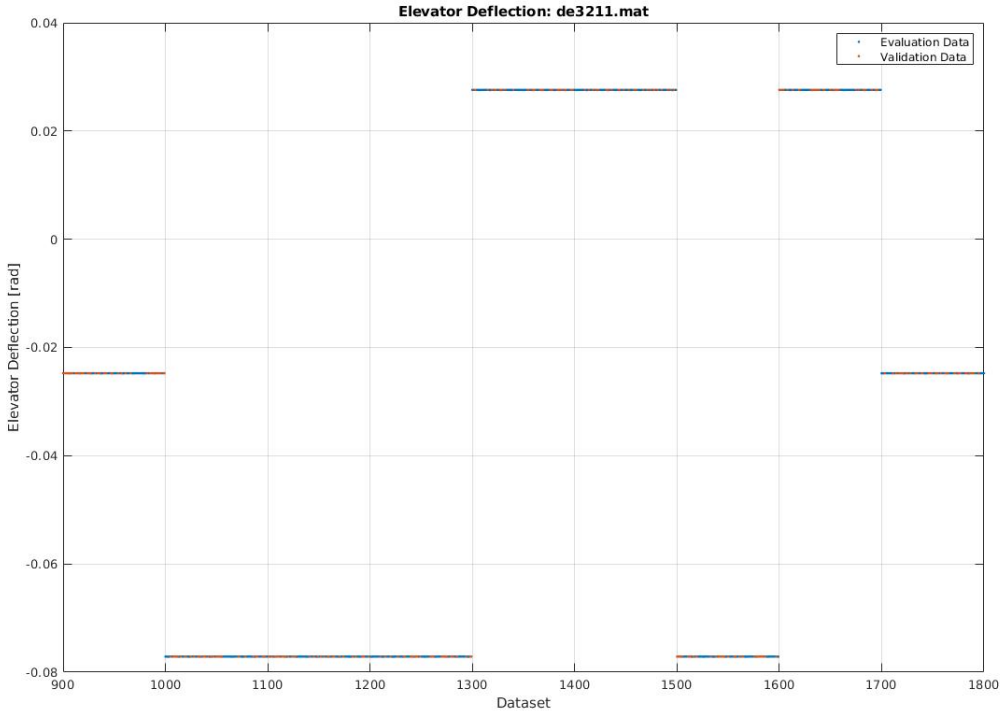

Figure 4.2: Evaluation and Validation dataset. Dataset: *de*3211.*mat*

All datasets are taken between datset t = 9[s] and t = 18 [s]. This is where the most excitation occurs and out of which 70% of the data is taken for evaluation and the remaining 30% is taken for validation. The data is also

normalised to ensure the model doesn't overfit few variables from the others..
Normalisation is done as:

$$\begin{aligned} max\_A(x) &= max(A(x),' columns') \\ max\_y &= max(y) \\ A(x)_n &= A(x)/max\_A(x) \\ y_n &= y/max\_y \\ \hat{\theta}_{OLS_n} &= (A_n^T(x)A_n(x))^{-1}A_n^T(x)y_n + \epsilon \\ \hat{\theta}_{OLS} &= \hat{\theta}_{OLS_n} * \frac{max\_y}{max\_A(x)} \end{aligned} \tag{4.7}$$

Parameter Estimation

The parameters were estimated based on the OLS formulation mentioned above. The residuals are calculated as follows:

$$\begin{aligned} Residual &= y - A(x)\hat{\theta}_{OLS} \\ \epsilon &= mean(Residual) \\ \sigma &= var(Residual) \end{aligned} \tag{4.8}$$

The mean and variance were calculated with the matlab inbuilt funcions and calculation of variance also correct for mean along the dataset. Additionally, R^2 analysis was performed over the dataset. Higher the value of R^2 it is assumed that the knowledge of the parameter is better known. R^2 is calculated as:

$$\begin{aligned} R^2 &= \frac{Variance\ explained\ by\ Parameters\ in\ Dataset}{Variance\ in\ Dataset} \\ R^2 &= 1 - \frac{Variance\ \underline{not}\ explained\ by\ Parameters\ in\ Dataset}{Variance\ in\ Dataset} \\ R^2 &= 1 - \frac{var(Residual)}{var(Dataset)} \end{aligned} \tag{4.9}$$

To check for over fitting the data, the R^2 analysis is only performed on "Validation Data" as shown in 4.2. The value of R^2 is between [0,1].
Additionally covariance of each estimated parameter was estimated as follows:

$$\begin{aligned} P_{Cx} &= diag(cov(A_{Cx}^T(x)A_{Cx}(x))\sigma_{Cx}^2) \\ P_{Cy} &= diag(cov(A_{Cy}^T(x)A_{Cy}(x))\sigma_{Cy}^2) \\ P_{Cz} &= diag(cov(A_{Cz}^T(x)A_{Cz}(x))\sigma_{Cz}^2) \\ P_{Cl} &= diag(cov(A_{Cl}^T(x)A_{Cl}(x))\sigma_{Cl}^2) \\ P_{Cm} &= diag(cov(A_{Cm}^T(x)A_{Cm}(x))\sigma_{Cm}^2) \\ P_{Cn} &= diag(cov(A_{Cn}^T(x)A_{Cn}(x))\sigma_{Cn}^2) \end{aligned} \tag{4.10}$$

And the diagonal elements in the matrix P_i is the variance of each estimated parameter.

C_x									
	C_{x_0}	C_{x_α}	$C_{x_{\alpha^2}}$	C_{x_q}	$C_{x_{\delta_e}}$	$C_{x_{T_c}}$	ϵ	σ^2	R^2
Values	-0.0231	0.0938	3.0826	-0.1205	0.0161	0.03	$3.7203e^{-5}$	$1.7271e^{-6}$	0.9969
Variance	0.0207	$8.51e^{-6}$	$8.622e^{-7}$	$4.85e^{-10}$	$2.013e^{-5}$	0.0031			

C_z								
	C_{z_0}	C_{z_α}	C_{z_q}	$C_{z_{\delta_e}}$	$C_{z_{T_c}}$	ϵ	σ^2	R^2
Values	1.3368	-4.9318	-3.1745	-0.2804	-3.739	$3.564e^{-4}$	$7.0426e^{-5}$	0.9988
Variance	0.972	0.0041	$7.453e^{-8}$	0.0011	0.1447			

C_m								
	C_{m_0}	C_{m_α}	C_{m_q}	$C_{m_{\delta_e}}$	$C_{m_{T_c}}$	ϵ	σ^2	R^2
Values	0.3394	-0.6582	-3.5219	-0.7876	-0.828	$-1.4044e^{-4}$	$9.5334e^{-5}$	0.9074
Variance	1.3158	0.0056	$1.0089e^{-7}$	0.0015	0.1958			

C_y									
	C_{y_0}	C_{y_β}	C_{y_p}	C_{y_r}	$C_{y_{\delta_a}}$	$C_{y_{\delta_r}}$	ϵ	σ^2	R^2
Values	$-5.25e^{-4}$	-0.56	-0.0154	0.476	-0.0403	0.116	$-2.89e^{-5}$	$-2.54e^{-6}$	0.9553
Variance	0.4971	$4.164e^{-6}$	$3.153e^{-7}$	$1.147e^{-6}$	$7.578e^{-7}$	$2.4657e^{-6}$			

C_l									
	C_{l_0}	C_{l_β}	C_{l_p}	C_{l_r}	$C_{l_{\delta_a}}$	$C_{l_{\delta_r}}$	ϵ	σ^2	R^2
Values	$-2e^{-6}$	-0.1073	-0.0498	0.1752	-0.0963	0.0437	$-1.708e^{-5}$	$9.12e^{-7}$	0.8142
Variance	0.1785	$1.4956e^{-6}$	$1.132e^{-7}$	$4.119e^{-7}$	$2.722e^{-7}$	$8.8556e^{-7}$			

C_n									
	C_{n_0}	C_{n_β}	C_{n_p}	C_{n_r}	$C_{n_{\delta_a}}$	$C_{n_{\delta_r}}$	ϵ	σ^2	R^2
Values	$3.47e^{-4}$	0.1811	0.0846	-0.2920	0.1672	-0.0727	$2.687e^{-5}$	$2.675e^{-6}$	0.8144
Variance	0.5234	$4.3852e^{-6}$	$3.321e^{-7}$	$1.207e^{-6}$	$7.9807e^{-7}$	$2.596e^{-6}$			

Best Linear Unbiased Estimates (BLUE)

By performing the analysis shown above, a strong co variance is observed between the thrust parameter and a the bias term. Then it was discovered that the thrust value was kept constant throughout the duration of the log. And thus $A(x)$ matrix became rank deficient and so the coefficient of thrust cannot be accurately estimated with this estimator. And given the very low errors in estimation $\epsilon \approx 0$, it suggests the estimator is in accordance to BLUE criteria.
Since the higher R^2 tells accuracy,

- Longitudinal Dynamics: $C_z(R^2 = 0.9988)$ is most accurately known. Followed by $C_x(R^2 = 0.9969)$ and $C_m(R^2 = 0.9074)$.
- Lateral Dynamics: $C_y(R^2 = 0.9553)$ is most accurately known. Followed by $C_n(R^2 = 0.8144)$ and $C_l(R^2 = 0.8142)$.

Out of which for longitudinal dynamics,

- Thrust Coefficients cannot be estimated due to lack of excitation
- Offset terms have high variances due to the lack of excitation on thrust and hence they are less certain. And also the Kalman term never converged completely in z axis along the dataset, that also increases the uncertainity in the offset term.
- The coefficient of q terms are most accurate in longitudinal dynamics because of their very small variances.
- Interestingly the coefficient of α terms are less accurately known in longitudinal dynamics. This could be due to low p-value in C.1 for α state and BLUE estimators are ideally designed for Gaussian based distribution. And the *alpha* terms have a trim setting which the estimator finds by a stong correlation with the bias terms.
- The elevator term is also less likely known and that could be due to the elevator operating from the trim setting and perhaps centering of the data is required to distinctively relate elevator dynamics from the bias term. This was also observed as high covariance between elevator term and the bias term.

And for lateral dynamics,

- The bias coefficients are least likely known.
- Airelons and rudder data was already centered and thus the coefficients of them are very accurately known along with the coefficients of p term.

- The coefficients of the sideslip angle terms are less accurately known.
- Despite having lower R^2 values the data's relative accuracy in lateral channel is higher than that of the longitudinal channel.

Parameter	ϵ	σ^2	R^2
C_x	$3.7203e^{-5}$	$1.7271e^{-6}$	0.9969
C_z	$3.564e^{-4}$	$7.0426e^{-5}$	0.9988
C_m	$-1.4044e^{-4}$	$9.5334e^{-5}$	0.9074
C_y	$-2.89e^{-5}$	$-2.54e^{-6}$	0.9553
C_l	$-1.708e^{-5}$	$9.12e^{-7}$	0.8142
C_n	$2.687e^{-5}$	$2.675e^{-6}$	0.8144

4.4 | Alternate Model Structure

$$
\begin{aligned}
C_x &= C_{x_{bias}} + C_{x_\alpha}(\alpha - \alpha_{trim}) + C_{x_{\alpha^2}}(\alpha^2 - \alpha_{trim}^2) + C_{x_q}\frac{qc}{V} + C_{x_{\delta_e}}(\delta_e - \delta_{e_{trim}}) \\
C_y &= C_{y_{bias}} + C_{y_\beta}(\beta - \beta_{trim}) + C_{y_p}\frac{pb}{2V} + C_{y_r}\frac{rb}{2V} + C_{y_{\delta_a}}(\delta_a - \delta_{a_{trim}}) + C_{y_{\delta_r}}(\delta_r - \delta_{r_{trim}}) \\
C_z &= C_{z_{bias}} + C_{z_\alpha}(\alpha - \alpha_{trim}) + C_{z_q}\frac{qc}{V} + C_{z_{\delta_e}}(\delta_e - \delta_{e_{trim}}) \\
C_l &= C_{l_{bias}} + C_{l_\beta}(\beta - \beta_{trim}) + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_a}}(\delta_a - \delta_{a_{trim}}) + C_{l_{\delta_r}}(\delta_r - \delta_{r_{trim}}) \\
C_m &= C_{m_{bias}} + C_{m_\alpha}(\alpha - \alpha_{trim}) + C_{m_q}\frac{qc}{V} + C_{m_{\delta_e}}(\delta_e - \delta_{e_{trim}}) \\
C_n &= C_{n_{bias}} + C_{n_\beta}(\beta - \beta_{trim}) + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_a}}(\delta_a - \delta_{a_{trim}}) + C_{n_{\delta_r}}(\delta_r - \delta_{r_{trim}})
\end{aligned}
\tag{4.11}
$$

C_x									
	C_{x_0}	C_{x_α}	$C_{x_{\alpha^2}}$	C_{x_q}	$C_{x_{\delta_e}}$	$C_{x_{T_c}}$	ϵ	σ^2	R^2
Values		0.927	3.0886	-0.0986	0.0156		$9.61e^{-7}$	$1.69e^{-6}$	0.9971
Variance		$3.75e^{-6}$	$3.57e^{-7}$	$8.65e^{-10}$	$8.72e^{-7}$				

C_z								
	C_{z_0}	C_{z_α}	C_{z_q}	$C_{z_{\delta_e}}$	$C_{z_{T_c}}$	ϵ	σ^2	R^2
Values		-4.9237	-3.2053	-0.2673		$5.05e^{-5}$	$8.24e^{-5}$	0.9986
Variance		$2.27e^{-4}$	$5.26e^{-8}$	$5.51e^{-5}$				

C_m								
	C_{m_0}	C_{m_α}	C_{m_q}	$C_{m_{\delta_e}}$	$C_{m_{T_c}}$	ϵ	σ^2	R^2
Values		-0.6450	-3.7075	-0.7778		5.28^{-4}	1.33^{-4}	0.8906
Variance		$3.68e^{-4}$	$8.51e^{-8}$	$8.91e^{-5}$				

Thus by centering the data, the variances of the longitudinal dynamics are much much better and the parameters in C_z and C_m states are much better known.
The bias terms used in eqn 4.11, $C_{x_{bias}}, C_{y_{bias}}, C_{z_{bias}}, C_{l_{bias}}, C_{m_{bias}}, C_{n_{bias}}$ are added for the convenience of OLS estimation and to remove the bias in estimates. They are <u>not</u>. the same as $C_{x_0}, C_{y_0}, C_{z_0}, C_{l_0}, C_{m_0}, C_{n_0}$.
It was not required to redo the analysis for lateral states as it was found that the data in β, δ_a, δ_r where already centered.

Dominant Parameters

Since the parameters are centered and normalised all the values in y, $A(x)$ lie between [1,-1]. And thus based on equation 4.7, on looking into the values of $\hat{\theta}_{OLS_n}$, the dominant and second most dominant parameters are underlined.

C_{x_n}					
$C_{x_{0_n}}$	$C_{x_{\alpha_n}}$	$C_{x_{\underline{\alpha_n^2}}}$	$C_{x_{q_n}}$	$C_{x_{\delta_{e_n}}}$	$C_{x_{T_{c_n}}}$
	0.1341	$\underline{0.7983}$	-0.0183	0.0153	

C_{z_n}				
$C_{z_{0_n}}$	$C_{z_{\underline{\alpha_n}}}$	$C_{z_{q_n}}$	$C_{z_{\delta_{e_n}}}$	$C_{z_{T_{c_n}}}$
	$\underline{-0.9502}$	0.0475	0.0378	

C_{m_n}				
$C_{m_{0_n}}$	$C_{m_{\underline{\alpha_n}}}$	$C_{m_{q_n}}$	$C_{m_{\underline{\delta_{e_n}}}}$	$C_{m_{T_{c_n}}}$
	$\underline{-0.4290}$	-0.1768	$\underline{-0.3609}$	

C_{y_n}					
$C_{y_{0_n}}$	$C_{y_{\underline{\beta_n}}}$	$C_{y_{p_n}}$	$C_{y_{r_n}}$	$C_{y_{\delta_{a_n}}}$	$C_{y_{\delta_{r_n}}}$
	$\underline{-1.0880}$	-0.0527	0.4406	-0.0838	0.2453

C_{l_n}					
$C_{l_{0_n}}$	$C_{l_{\underline{\beta_n}}}$	$C_{l_{p_n}}$	$C_{l_{r_n}}$	$C_{l_{\underline{\delta_{a_n}}}}$	$C_{l_{\delta_{r_n}}}$
	$\underline{-0.3053}$	-0.2730	0.2367	$\underline{-0.2995}$	0.1345

C_{n_n}					
$C_{n_{0_n}}$	$C_{n_{\underline{\beta_n}}}$	$C_{n_{p_n}}$	$C_{n_{r_n}}$	$C_{n_{\underline{\delta_{a_n}}}}$	$C_{n_{\delta_{r_n}}}$
	$\underline{-0.3114}$	0.2798	-0.2382	$\underline{0.3141}$	0.1352

Appendices

A | Estimation Results, Dataset: *da*3211

A.1 | Measured Data and Filtered Data

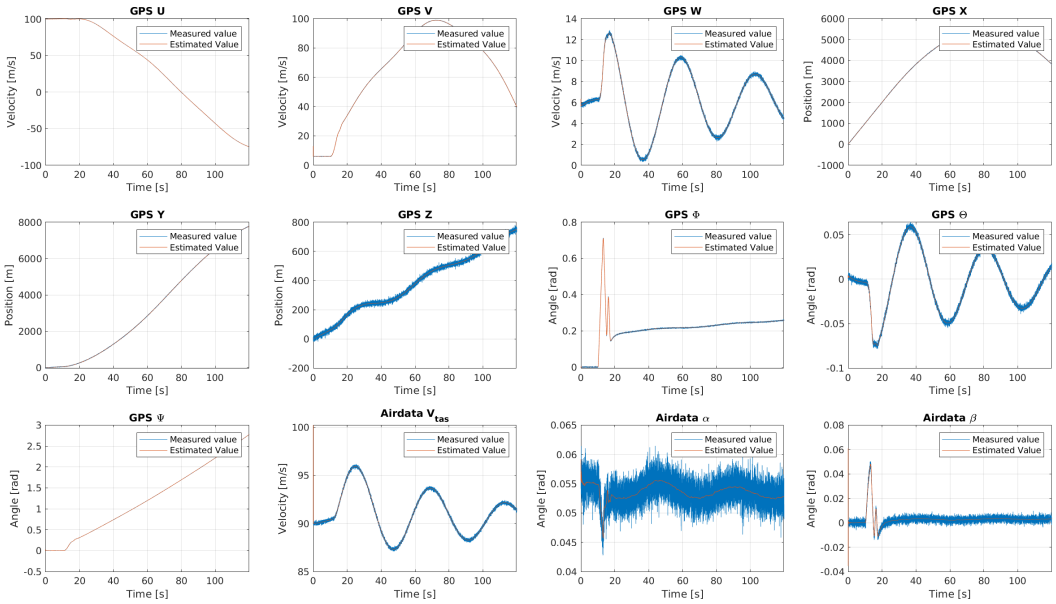

Figure A.1: Raw data and converged result. dataset: *da*3211.*mat*

A.2 | Bias Estimate

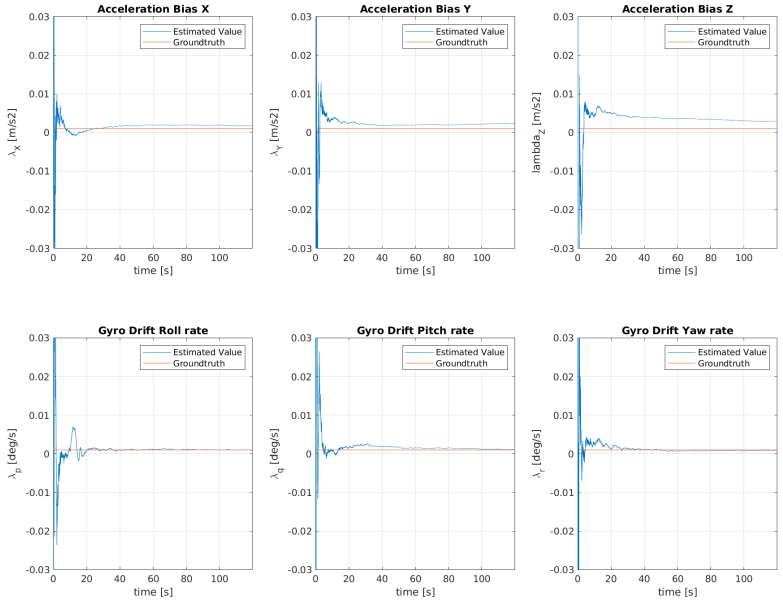

Figure A.2: Bias Estimate. Dataset: *da*3211.*mat*

A.3 | Wind Estimate

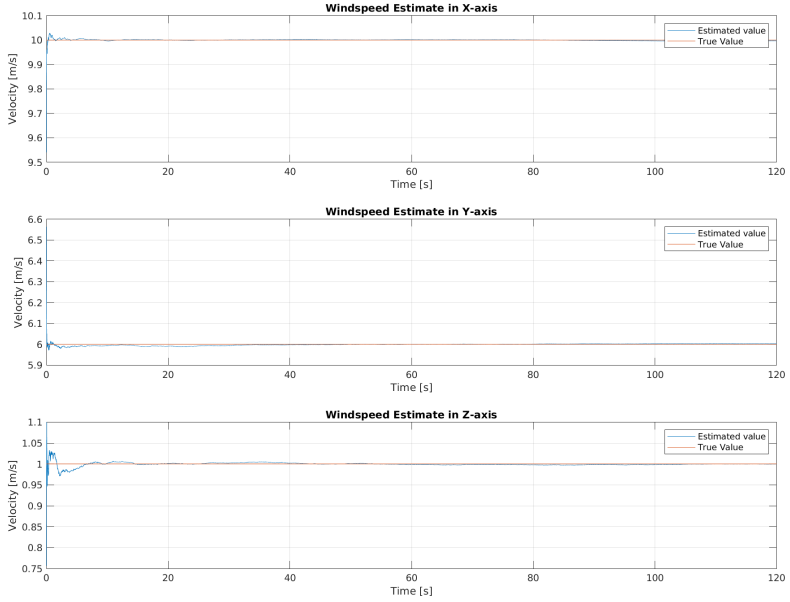

Figure A.3: Wind Estimate. Dataset: *da*3211.*mat*

A.4 | Convergence

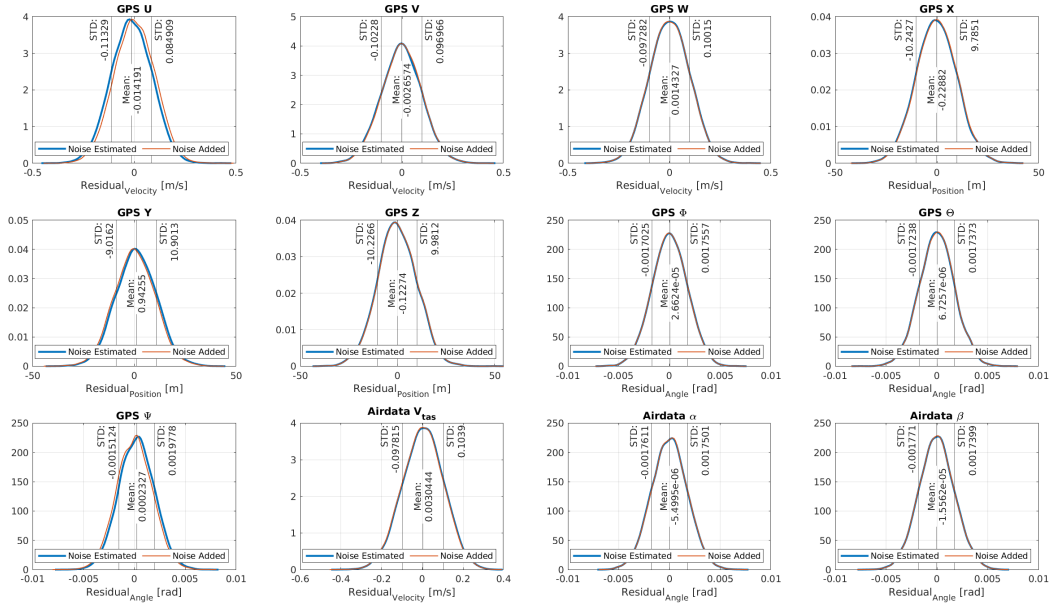

Figure A.4: Convergence. Dataset: *da*3211.*mat*

Table A.1: Convergence-*da*3211.*mat*

Convergence test				
Output Measurement	Gaussian test (KS Test)	pvalue	Ztest	pvalue
GPS U	Gaussian	0.75	false	2.2e-79
GPS V	Gaussian	0.87	true	0.69133
GPS W	Gaussian	0.53	false	2.25e-8
GPS X	Gaussian	0.98	true	0.07990
GPS Y	Gaussian	0.95	false	2.6e-21
GPS Z	Gaussian	0.92	true	0.18525
GPS Φ	Gaussian	0.69	false	2.65e-5
GPS Θ	Gaussian	0.91	false	0.027
GPS Ψ	Gaussian	0.38	false	4.9e-55
Airdata Vtas	Gaussian	0.68	false	0.02223
Airdata α	Gaussian	0.90	false	0.00544
Airdata β	Gaussian	0.91	true	0.26358

A.5 | Adding more noise to Airdata

The noise was increased such that the standard deviation increased by factor of 10.

Extra Noise: Measured Data and Filtered Data

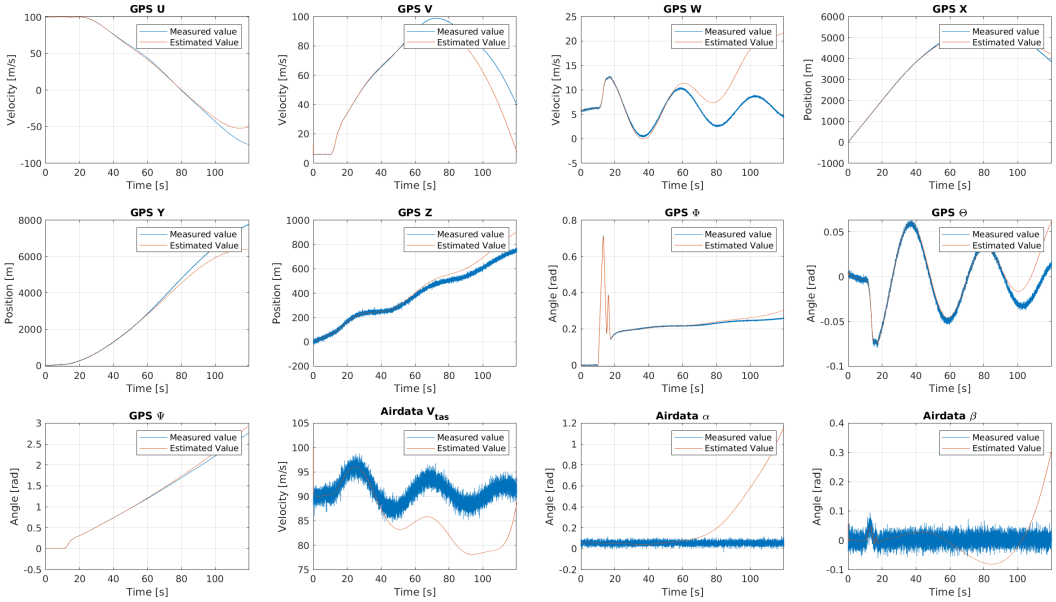

Figure A.5: Raw data and converged result after adding extra noise in airdata sensors. dataset: *da*3211.*mat*

Extra Noise: Bias Estimate

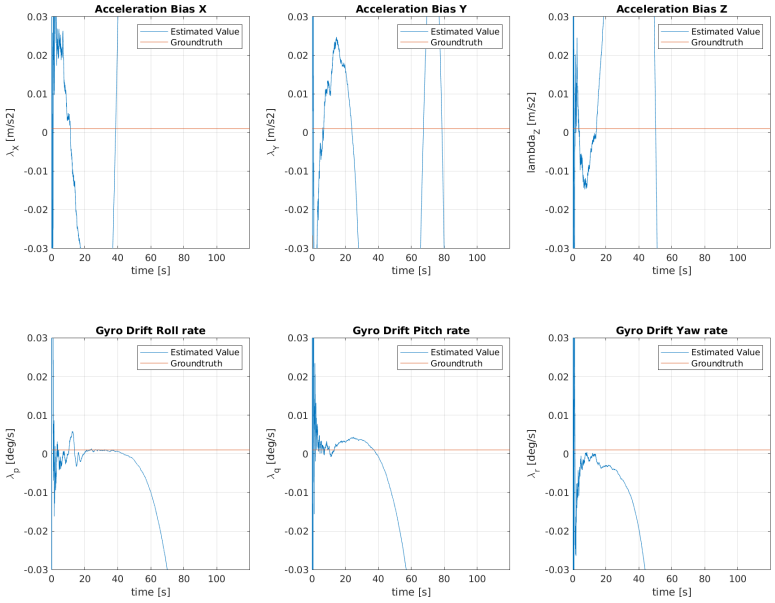

Figure A.6: Bias Estimate after adding extra noise in airdata sensors. Dataset: *da*3211.*mat*

Extra Noise: Wind Estimate

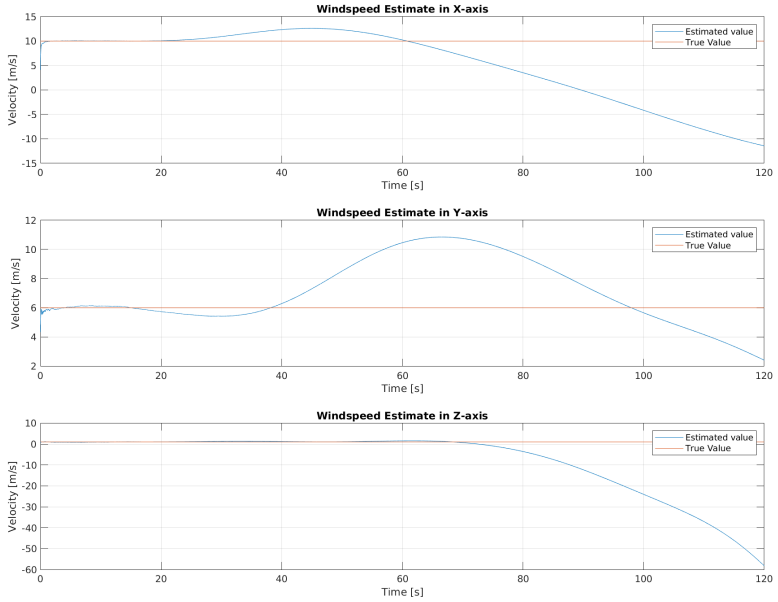

Figure A.7: Wind Estimate after adding extra noise in airdata sensors. Dataset: *da*3211.*mat*

B | Estimation Results, Dataset: *dadoublet*

B.1 | Measured Data and Filtered Data

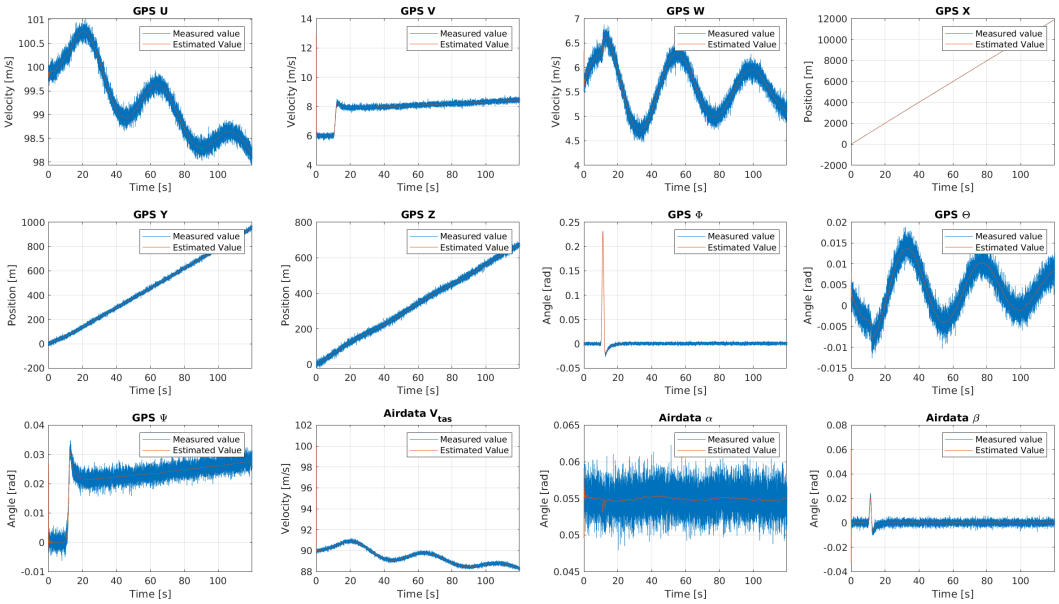

Figure B.1: Raw data and converged result. dataset: *dadoublet.mat*

B.2 | Bias Estimate

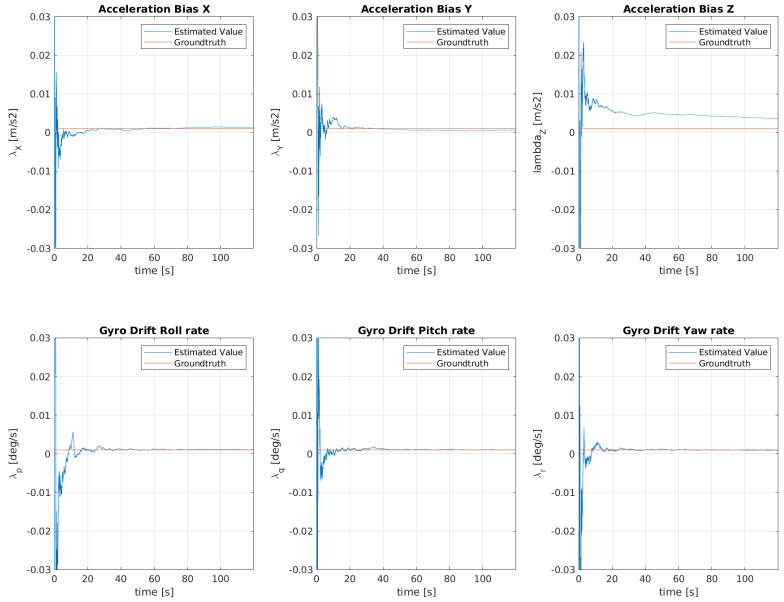

Figure B.2: Bias Estimate. Dataset: *dadoublet.mat*

B.3 | Wind Estimate

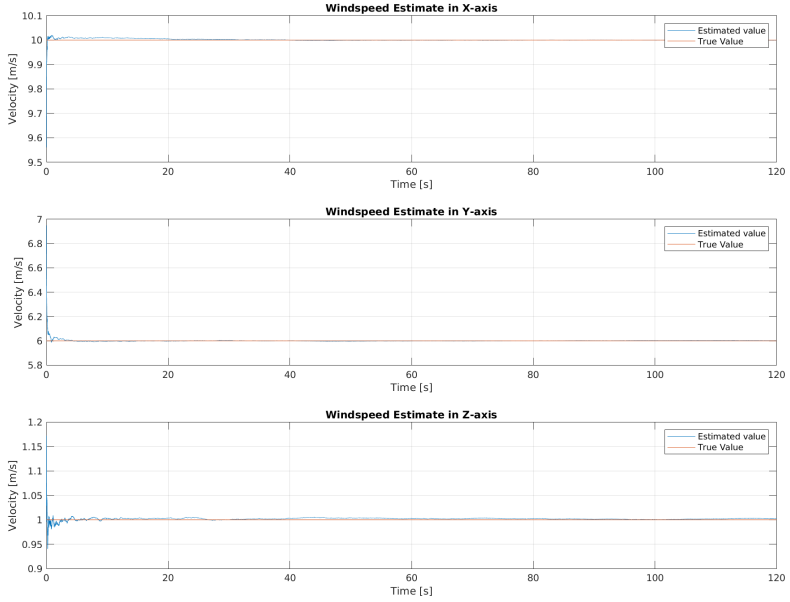

Figure B.3: Wind Estimate. Dataset: *dadoublet.mat*

B.4 | Convergence

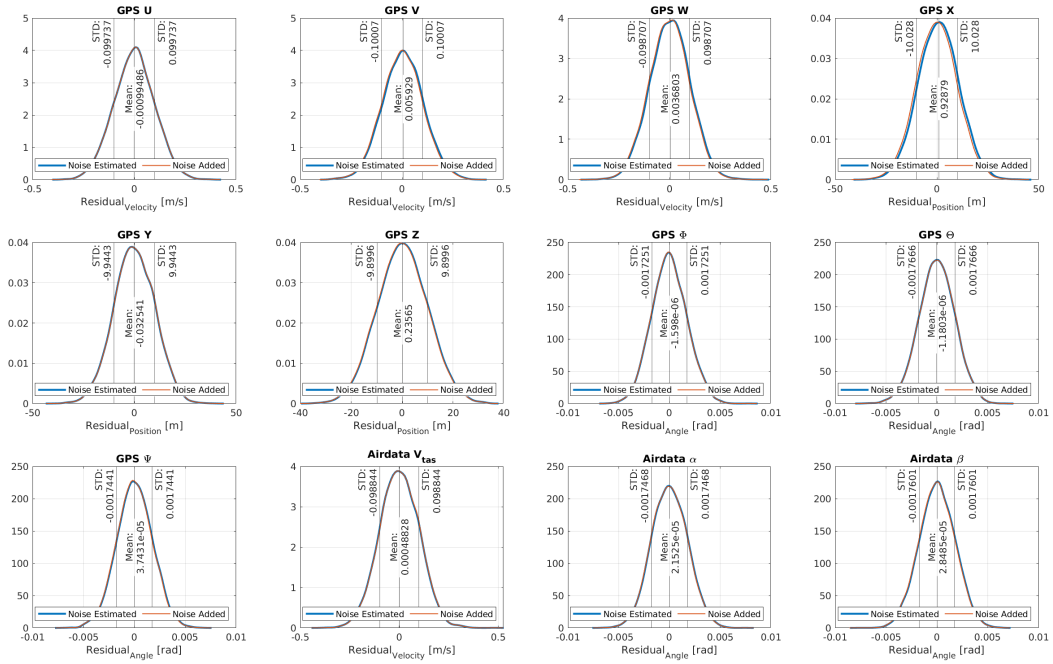

Figure B.4: Convergence. Dataset: *dadoublet.mat*

Table B.1: Convergence-*dadoublet.mat*

Convergence test				
Output Measurement	Gaussian test (KS Test)	pvalue	Ztest	pvalue
GPS U	Gaussian	0.87	true	0.798
GPS V	Gaussian	0.97	true	0.0716
GPS W	Gaussian	0.72	false	6.1e-11
GPS X	Gaussian	0.98	false	4.9e-18
GPS Y	Gaussian	0.86	true	0.2547
GPS Z	Gaussian	0.99	true	0.58072
GPS Φ	Gaussian	0.90	true	0.0604
GPS Θ	Gaussian	0.84	true	0.57
GPS Ψ	Gaussian	0.97	true	0.28286
Airdata Vtas	Gaussian	0.91	true	0.80551
Airdata α	Gaussian	0.85	false	1.483e-7
Airdata β	Gaussian	0.99	true	0.47055

B.5 | Adding more noise to Airdata

The noise was increased such that the standard deviation increased by factor of 10.

Extra Noise: Measured Data and Filtered Data

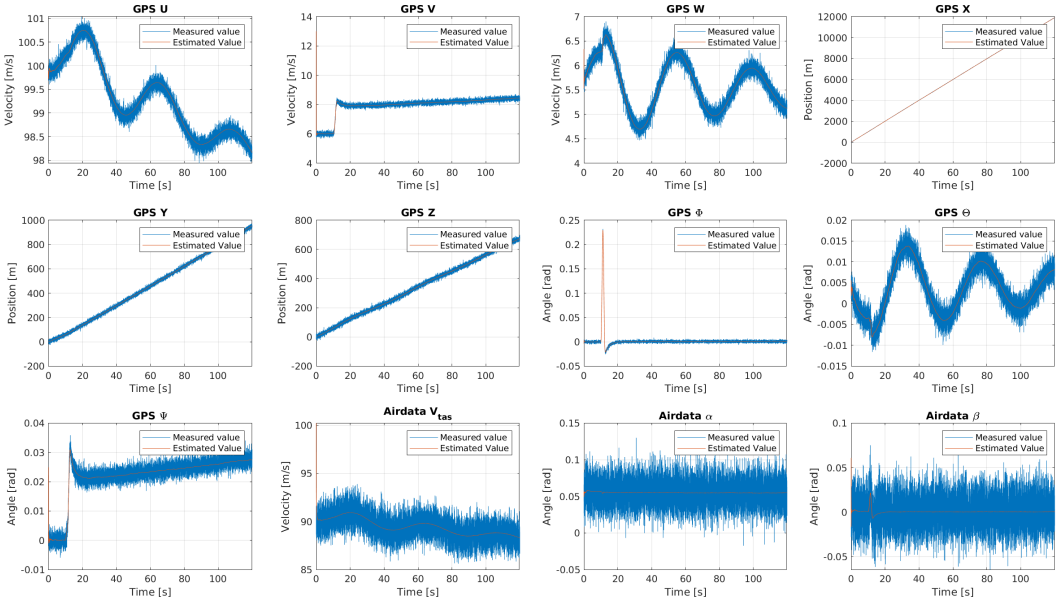

Figure B.5: Raw data and converged result after adding extra noise in airdata sensors. dataset: *dadoublet.mat*

Extra Noise: Bias Estimate

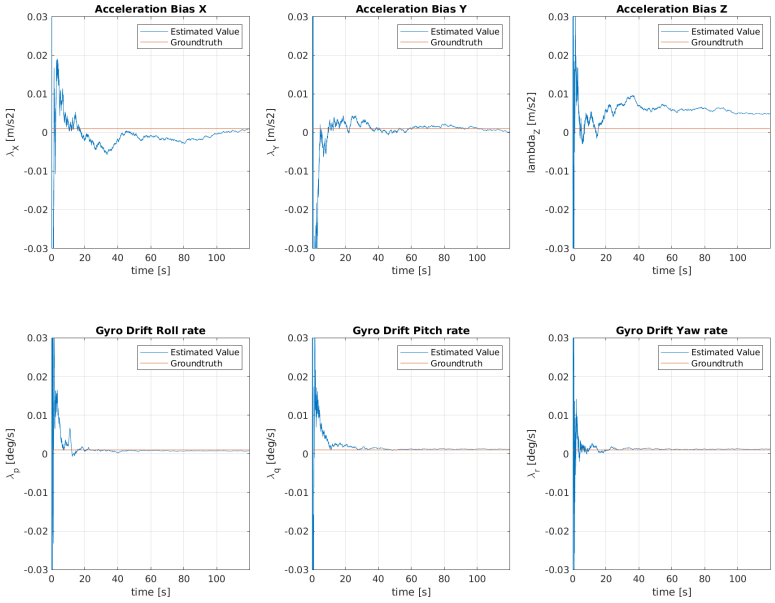

Figure B.6: Bias Estimate after adding extra noise in airdata sensors. Dataset: *dadoublet.mat*

Extra Noise: Wind Estimate

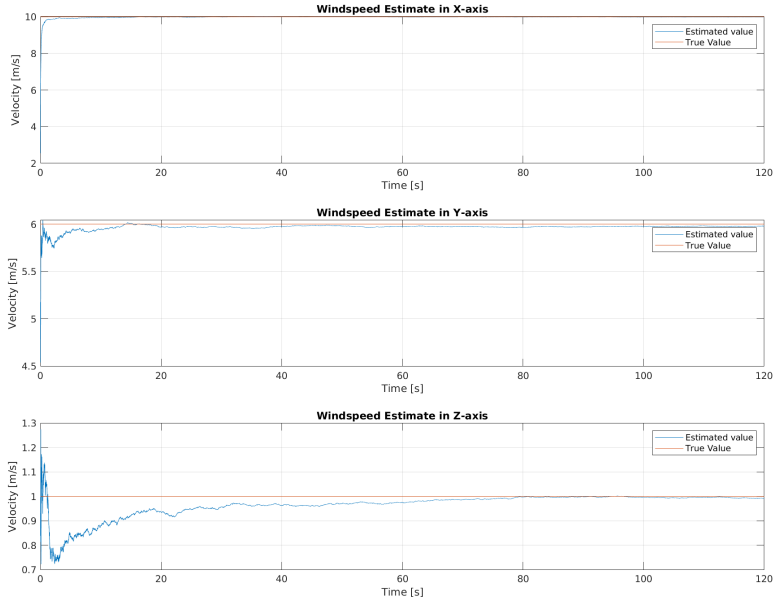

Figure B.7: Wind Estimate after adding extra noise in airdata sensors. Dataset: *dadoublet.mat*

C | Estimation Results, Dataset: *de*3211

C.1 | Measured Data and Filtered Data

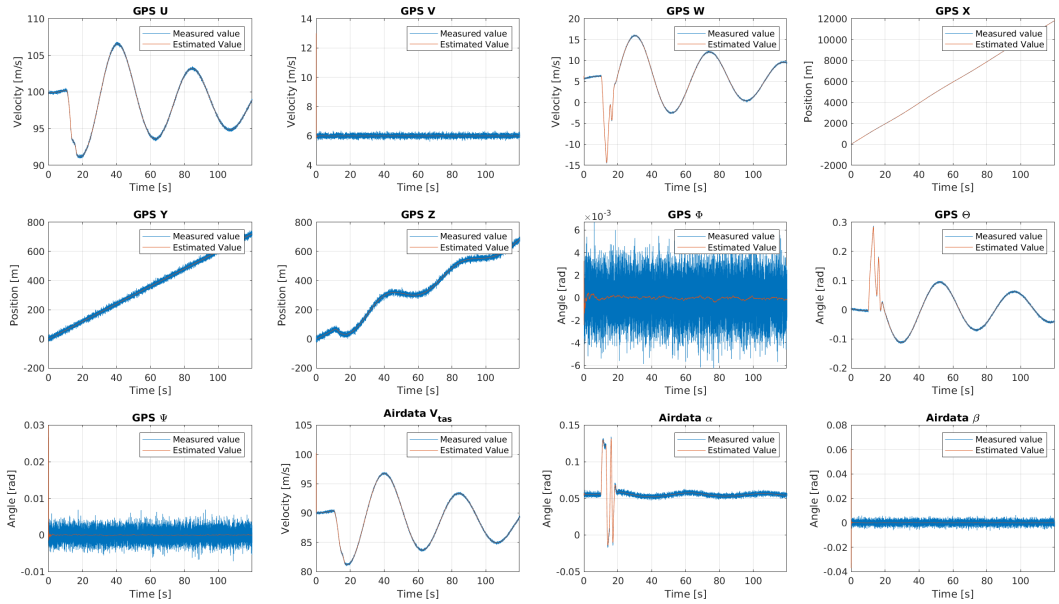

Figure C.1: Raw data and converged result. dataset: *de*3211.*mat*

C.2 | Bias Estimate

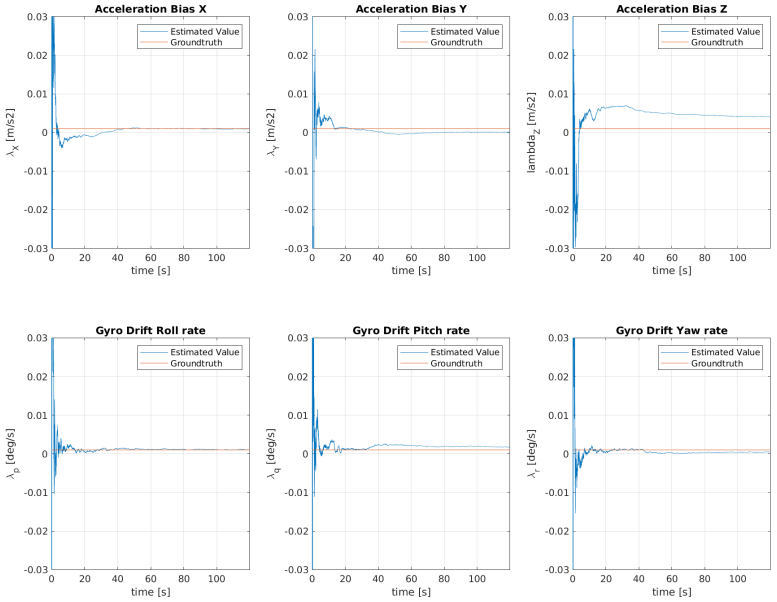

Figure C.2: Bias Estimate. Dataset: *de*3211.*mat*

C.3 | Wind Estimate

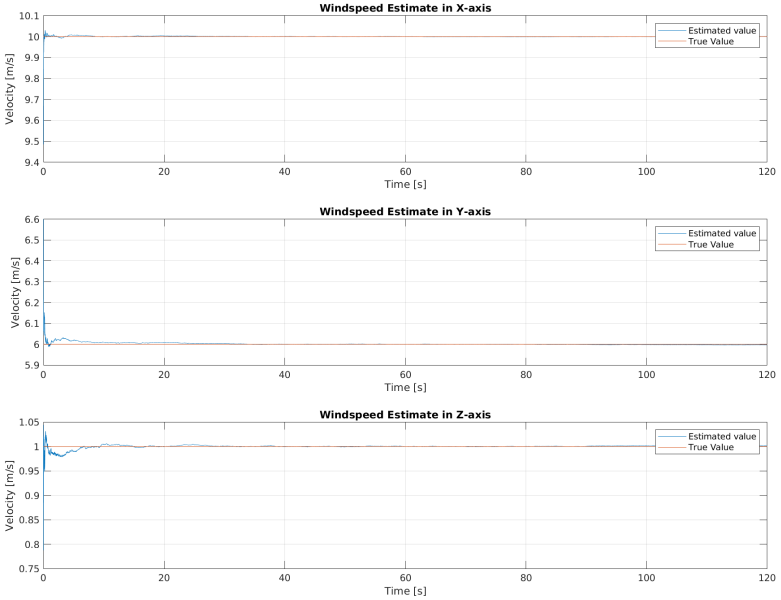

Figure C.3: Wind Estimate. Dataset: *de*3211.*mat*

C.4 | Convergence

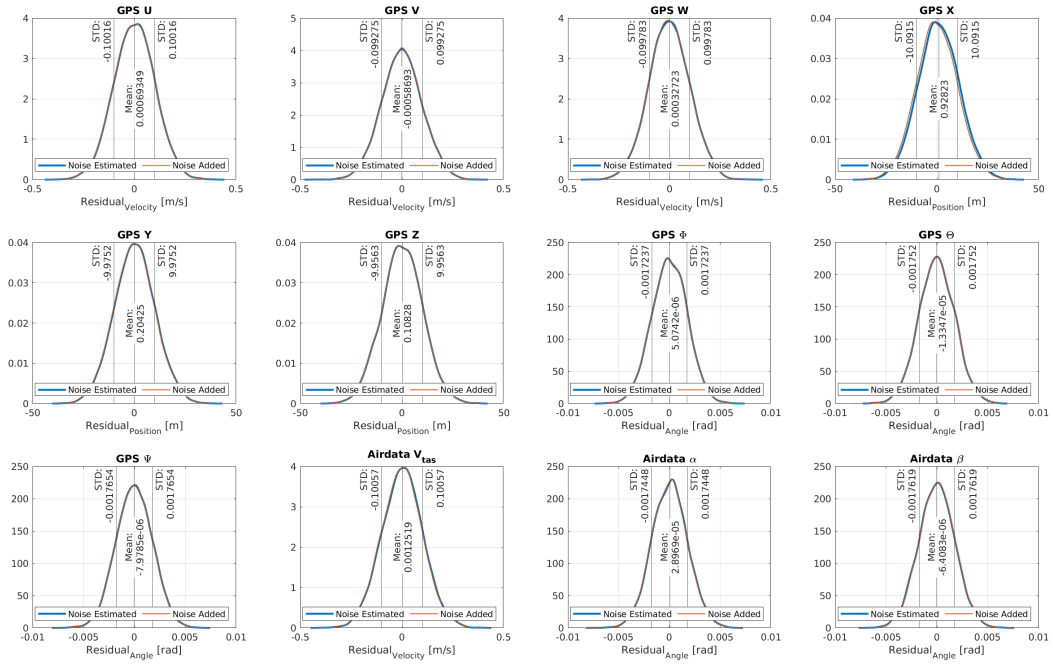

Figure C.4: Convergence. Dataset: *de*3211.*mat*

Table C.1: Convergence-*de*3211.*mat*

Convergence test				
Output Measurement	Gaussian test (KS Test)	pvalue	Ztest	pvalue
GPS U	Gaussian	0.89	false	0.036
GPS V	Gaussian	0.81	true	0.486
GPS W	Gaussian	0.37	true	0.976
GPS X	Gaussian	0.96	false	3.76e-17
GPS Y	Gaussian	0.64	true	0.01
GPS Z	Gaussian	0.83	false	0.23
GPS Φ	Gaussian	0.47	true	0.0107
GPS Θ	Gaussian	0.98	true	0.7773
GPS Ψ	Gaussian	0.33	true	0.76553
Airdata Vtas	Gaussian	0.85	true	0.1203
Airdata α	Gaussian	0.27	true	0.4428
Airdata β	Gaussian	0.76	true	0.12178

C.5 | Adding more noise to Airdata

The noise was increased such that the standard deviation increased by factor of 10.

Extra Noise: Measured Data and Filtered Data

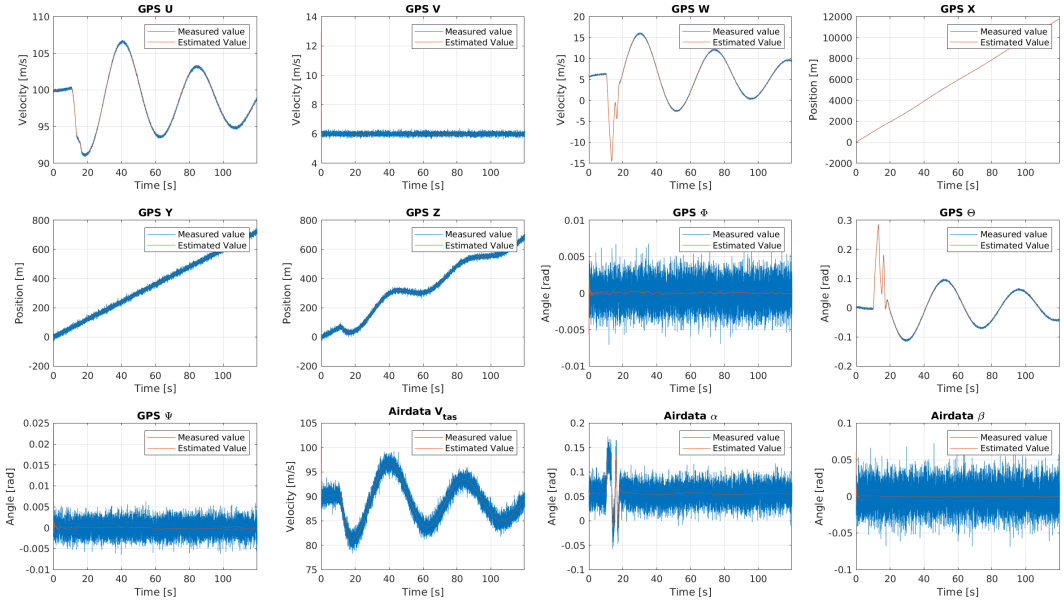

Figure C.5: Raw data and converged result after adding extra noise in airdata sensors. dataset: *de*3211.*mat*

Extra Noise: Bias Estimate

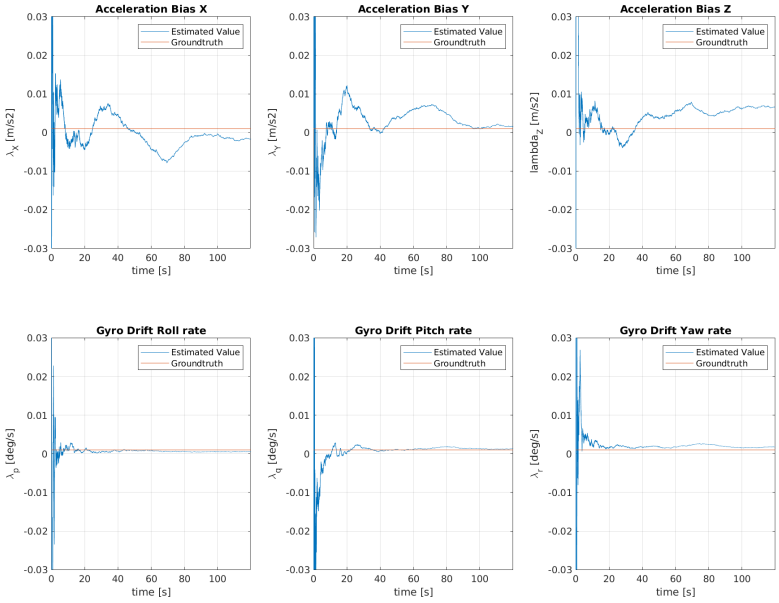

Figure C.6: Bias Estimate after adding extra noise in airdata sensors. Dataset: *de*3211.*mat*

Extra Noise: Wind Estimate

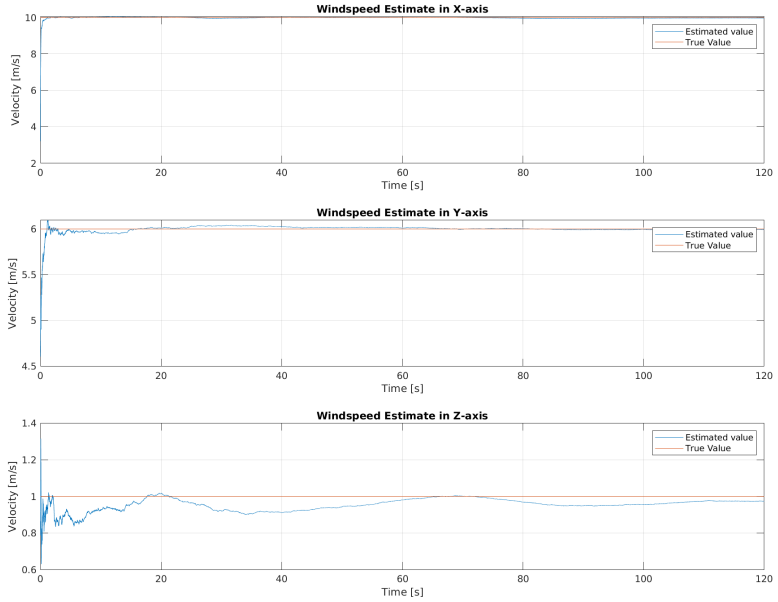

Figure C.7: Wind Estimate after adding extra noise in airdata sensors. Dataset: *de*3211.*mat*

D | Estimation Results, Dataset: $dr3211$

D.1 | Measured Data and Filtered Data

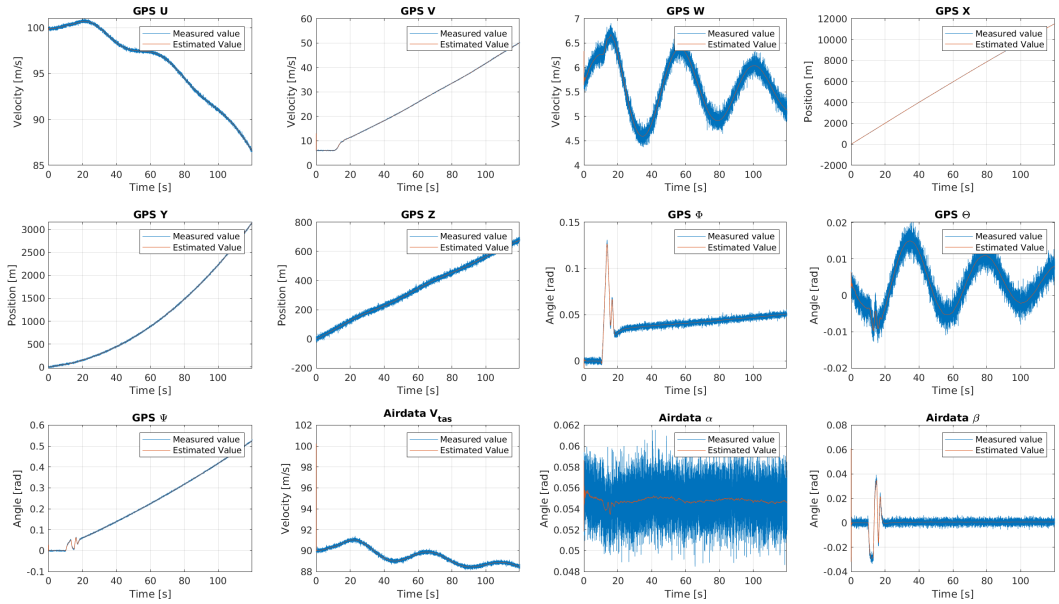

Figure D.1: Raw data and converged result. dataset: $dr3211.mat$

D.2 | Bias Estimate

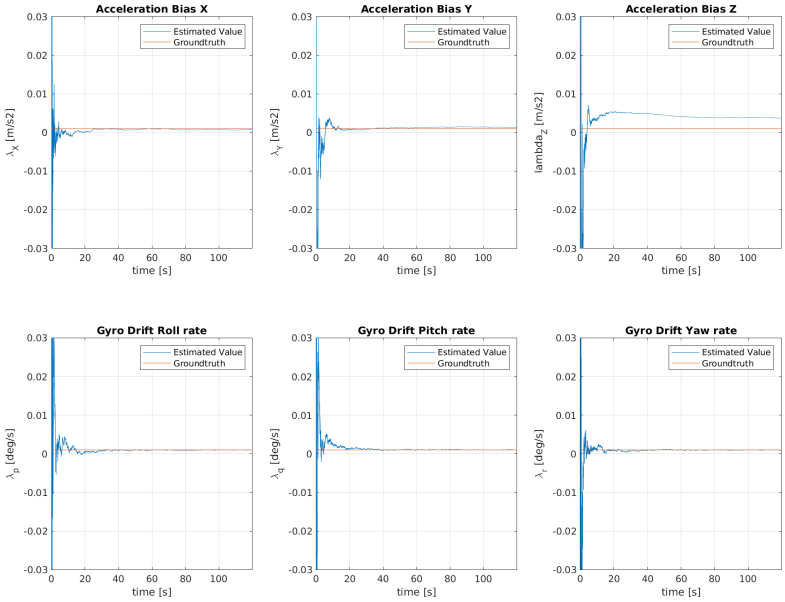

Figure D.2: Bias Estimate. Dataset: *dr*3211.*mat*

D.3 | Wind Estimate

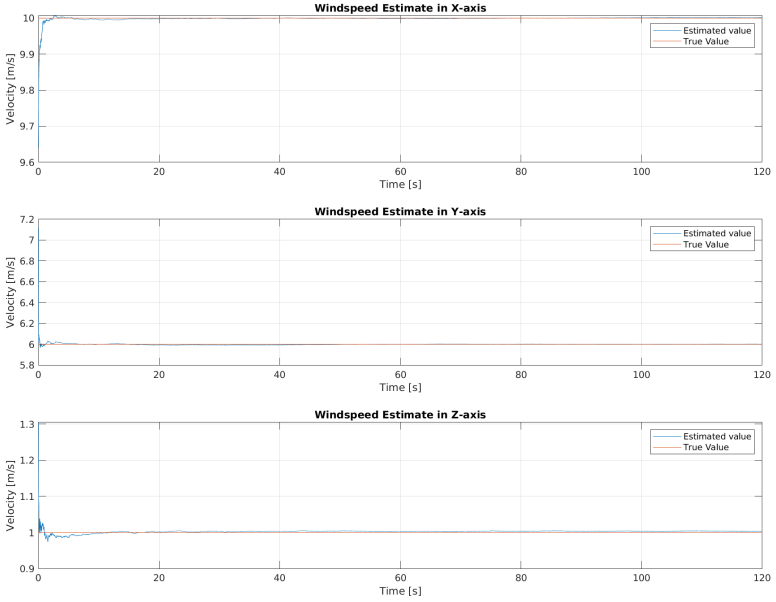

Figure D.3: Wind Estimate. Dataset: *dr*3211.*mat*

D.4 | Convergence

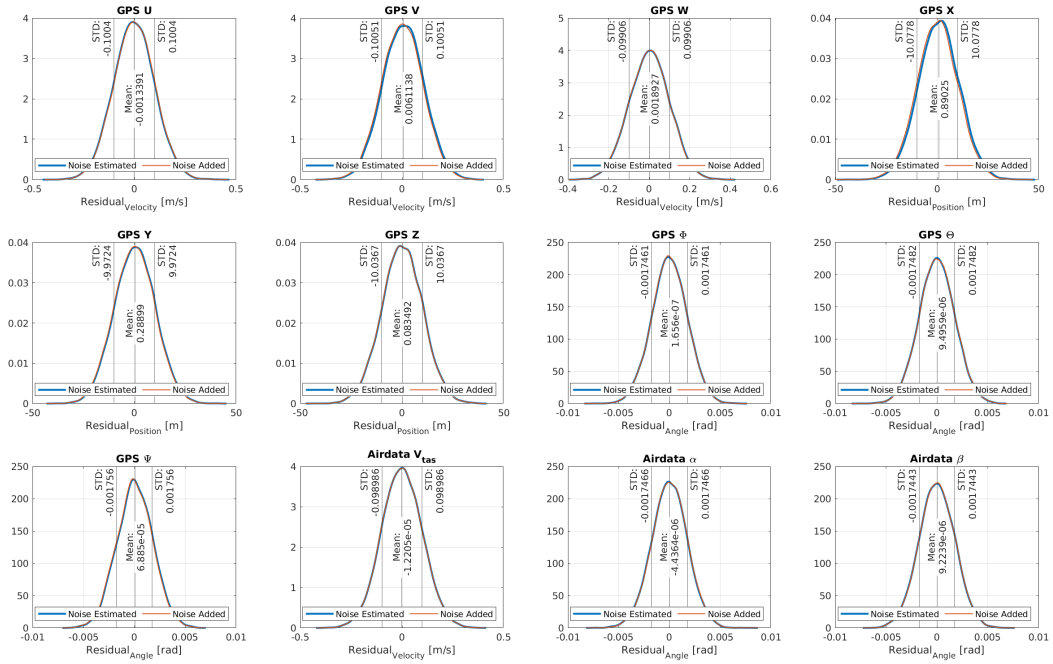

Figure D.4: Convergence. Dataset: *dr*3211.*mat*

Table D.1: Convergence-*dr*3211.*mat*

Convergence test				
Output Measurement	Gaussian test (KS Test)	pvalue	Ztest	pvalue
GPS U	Gaussian	0.95	true	0.16634
GPS V	Gaussian	0.99	false	0.0125
GPS W	Gaussian	0.57	false	4.9e-5
GPS X	Gaussian	0.58	false	1.3e21
GPS Y	Gaussian	0.86	true	0.5527
GPS Z	Gaussian	0.86	true	0.817
GPS Φ	Gaussian	0.37	true	0.206
GPS Θ	Gaussian	0.98	true	0.0985
GPS Ψ	Gaussian	0.86	true	0.75
Airdata Vtas	Gaussian	0.96	true	0.722
Airdata α	Gaussian	0.21	true	0.923
Airdata β	Gaussian	0.23	false	0.0002952

D.5 | Adding more noise to Airdata

The noise was increased such that the standard deviation increased by factor of 10.

Extra Noise: Measured Data and Filtered Data

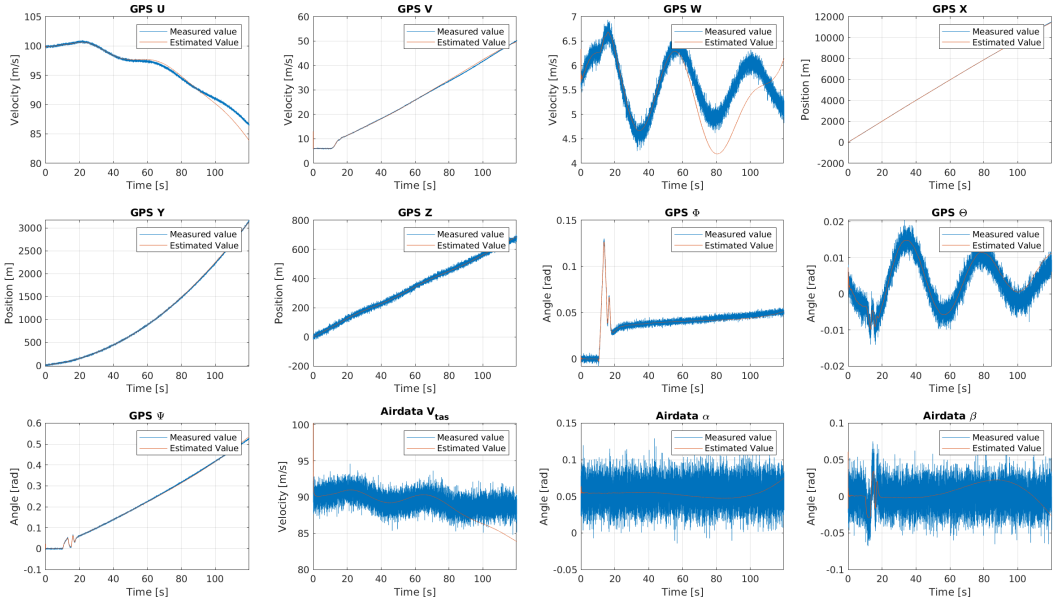

Figure D.5: Raw data and converged result after adding extra noise in airdata sensors. dataset: *dr*3211.*mat*

Extra Noise: Bias Estimate

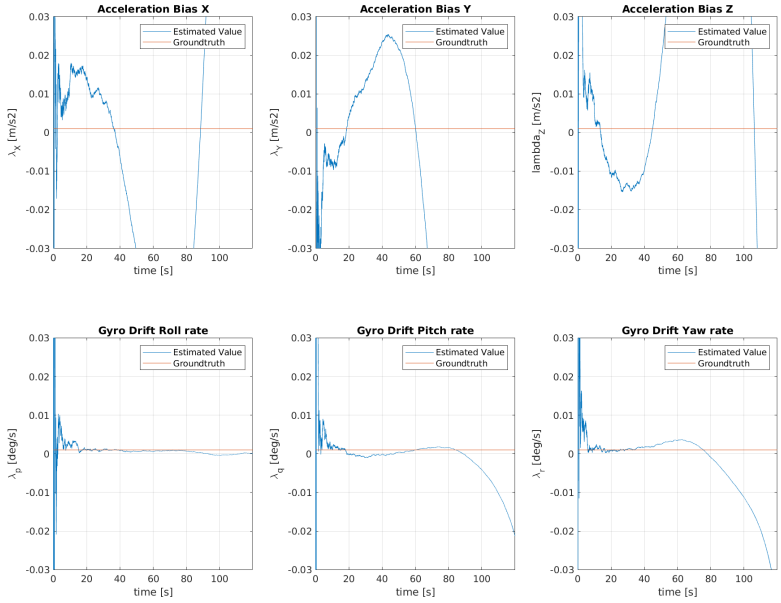

Figure D.6: Bias Estimate after adding extra noise in airdata sensors. Dataset: *dr*3211.*mat*

Extra Noise: Wind Estimate

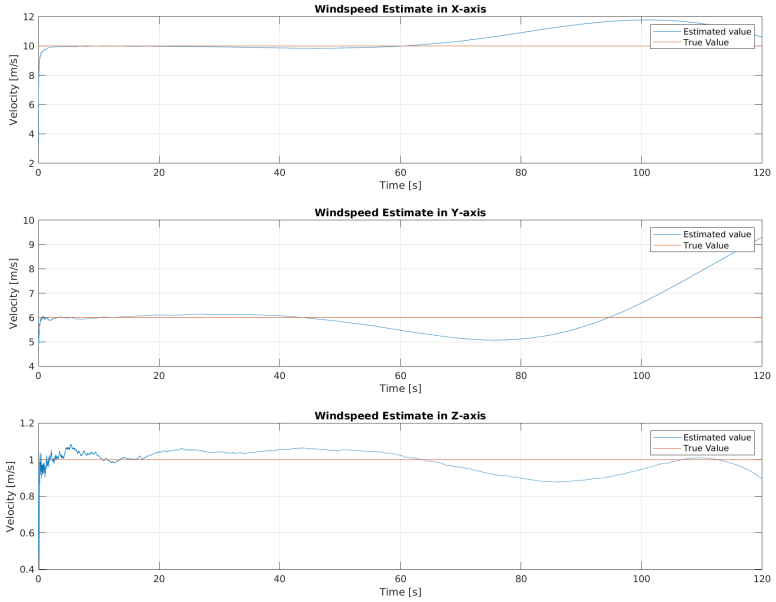

Figure D.7: Wind Estimate after adding extra noise in airdata sensors. Dataset: *dr*3211.*mat*

E | Estimation Results, Dataset: *drdoublet*

E.1 | Measured Data and Filtered Data

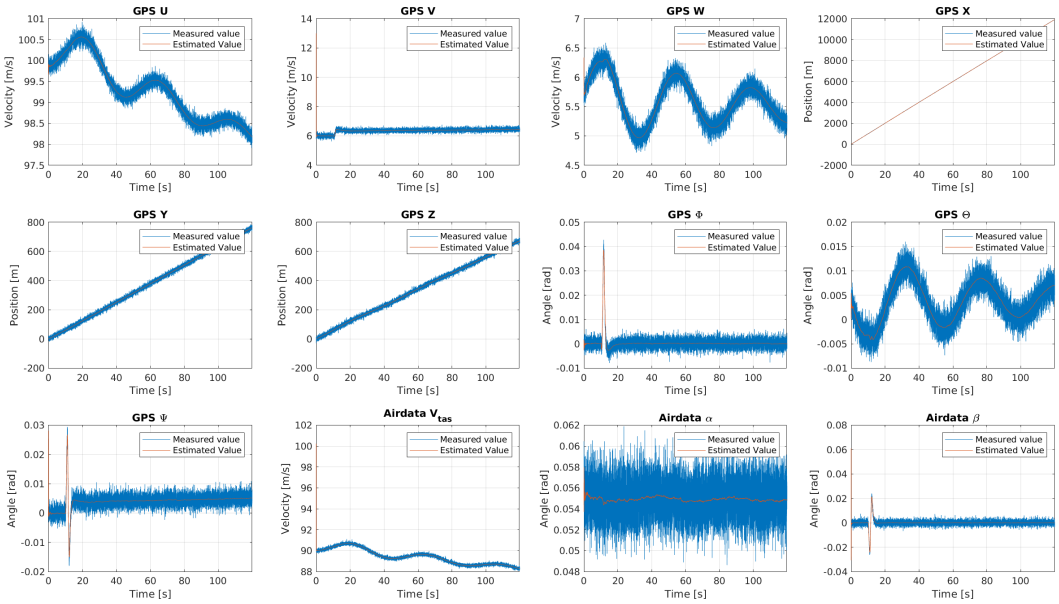

Figure E.1: Raw data and converged result. dataset: *drdoublet.mat*

E.2 | Bias Estimate

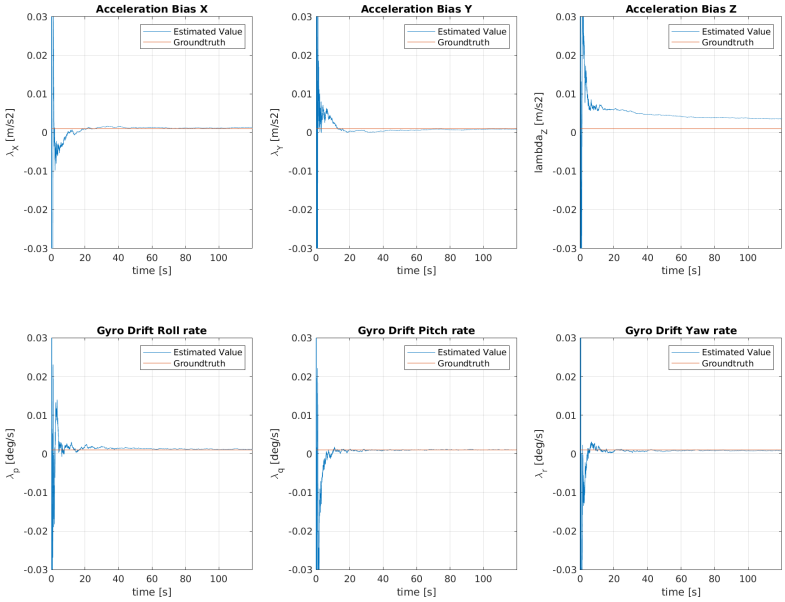

Figure E.2: Bias Estimate. Dataset: *drdoublet.mat*

E.3 | Wind Estimate

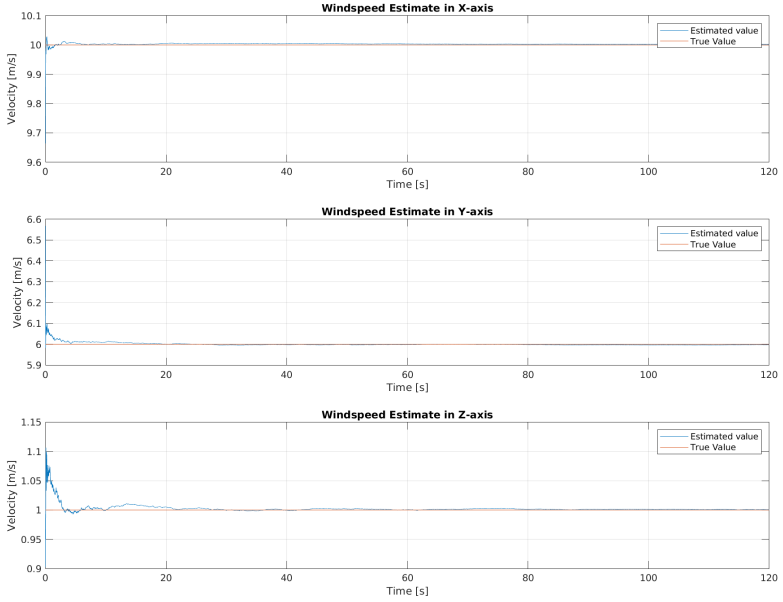

Figure E.3: Wind Estimate. Dataset: *drdoublet.mat*

E.4 | Convergence

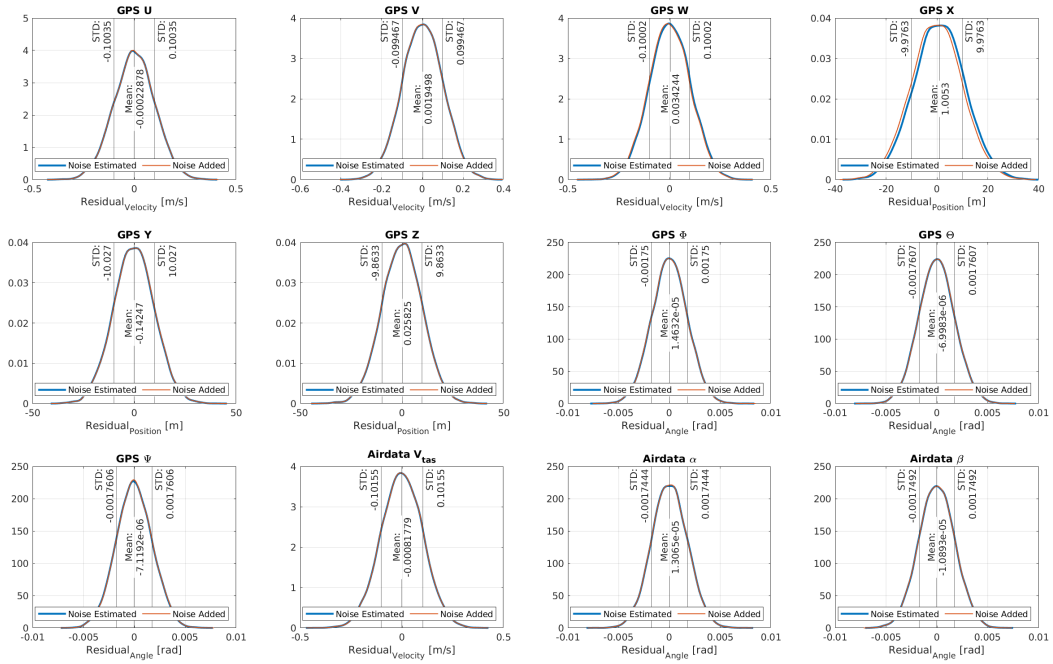

Figure E.4: Convergence. Dataset: *drdoublet.mat*

Table E.1: Convergence-*drdoublet.mat*

Convergence test				
Output Measurement	Gaussian test (KS Test)	pvalue	Ztest	pvalue
GPS U	Gaussian	0.95	true	0.20442
GPS V	Gaussian	0.76	false	0.00026
GPS W	Gaussian	0.71	false	0.00086
GPS X	Gaussian	0.98	false	4.0e-21
GPS Y	Gaussian	0.49	false	1.96e-6
GPS Z	Gaussian	0.75	true	0.35064
GPS Φ	Gaussian	0.98	true	0.77455
GPS Θ	Gaussian	0.84	false	0.00076
GPS Ψ	Gaussian	0.57	false	0.01088
Airdata Vtas	Gaussian	0.58	false	0.00397
Airdata α	Gaussian	0.61	true	0.43599
Airdata β	Gaussian	0.33	true	0.63189

E.5 | Adding more noise to Airdata

The noise was increased such that the standard deviation increased by factor of 10.

Extra Noise: Measured Data and Filtered Data

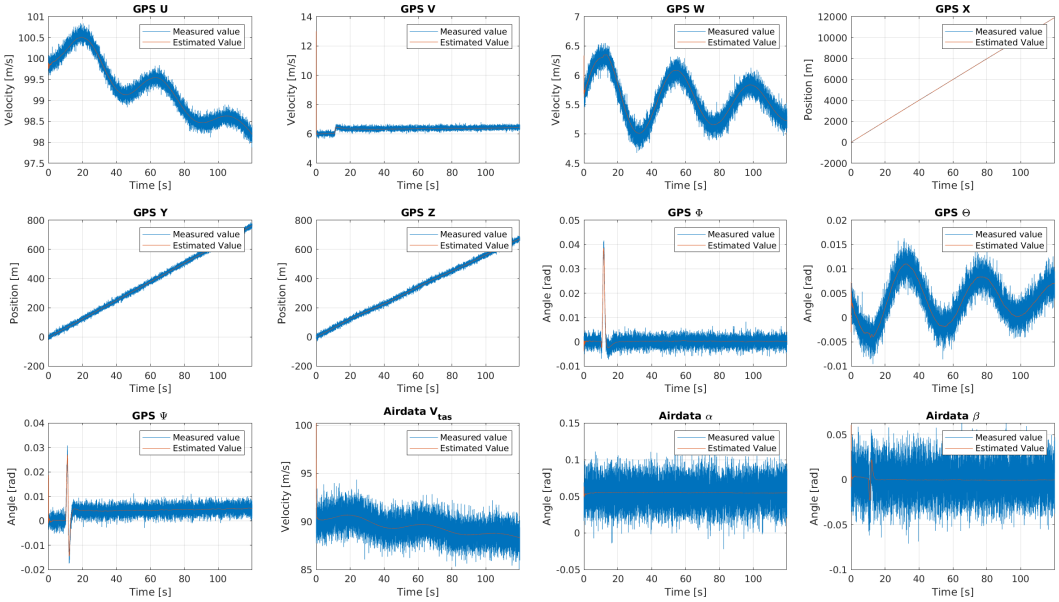

Figure E.5: Raw data and converged result after adding extra noise in airdata sensors. dataset: *drdoublet.mat*

Extra Noise: Bias Estimate

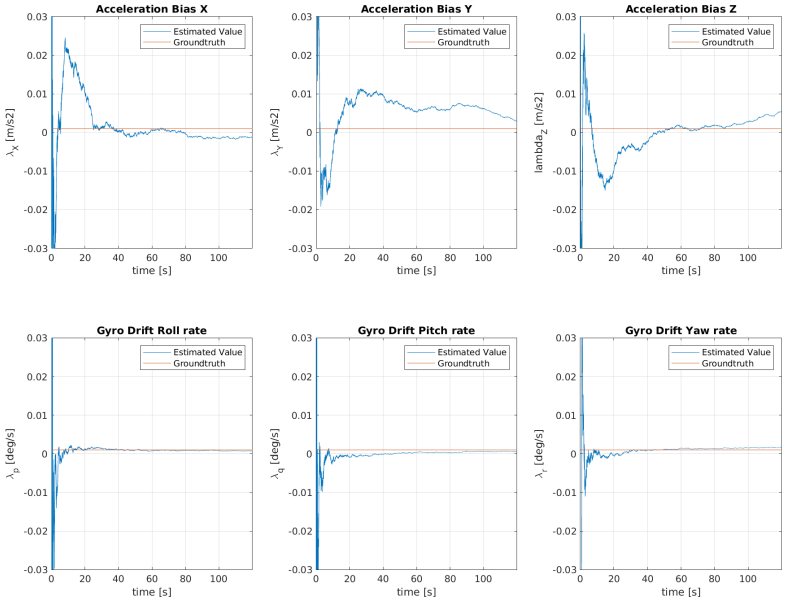

Figure E.6: Bias Estimate after adding extra noise in airdata sensors. Dataset: *drdoublet.mat*

Extra Noise: Wind Estimate

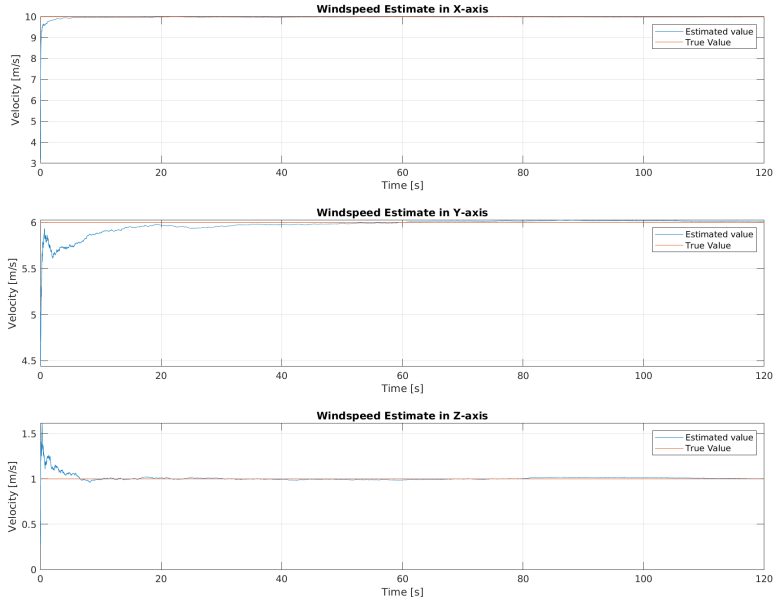

Figure E.7: Wind Estimate after adding extra noise in airdata sensors. Dataset: *drdoublet.mat*

References

[1] Metin Öner and Ipek Deveci Kocakoç. "JMASM 49: A compilation of some popular goodness of fit tests for normal distribution: Their algorithms and MATLAB codes (MATLAB)". In: *Journal of Modern Applied Statistical Methods* 16 (Dec. 2017), pp. 547–575. DOI: 10.22237/jmasm/1509496200.