System Identification Summary

In this summary, we examine several systems identification techniques. What is system identification? How do techniques like the Kalman filter, the least-squares method, the maximum likelihood method and the two-step method work? And how is system identification related to aircraft? Let's find out.

1 The basics of aircraft system identification

1.1 Aircraft model identification

Let's suppose that we are developing an airplane. In the later stages of developement, it is important to know the dynamics of the aircraft as precisely as possible. Flight tests are often used for this. On-board flight test instrumentation provides us with data, which we should then use to model the dynamics of the aircraft. This is called **aircraft model identification** (AMI) with flight test data, and will be the subject of this summary. Once an aircraft model has been obtained, it can be used for aircraft guidance and flight control design.

The aircraft model is split up into a dynamic model and a kinematic model, each having both a translational and rotational part. The kinematic model is precisely known. The dynamic model is, however, more difficult to obtain. To find it, we use Newton's second law, involving forces and moments. But finding these forces and moments can be very difficult.

Conventionally, AMI is done offline. That is, the flight data is analysed after the flight. But recently, newer online AMI methods have been developed. In such methods the aircraft model is continuously updated during the flight. These online methods can be applied very well in **adaptive flight control**. (In adaptive flight control, the controller adapts itself to the dynamics of the aircraft, even when these dynamics change due to, for example, aircraft damage.)

1.2 State and parameter estimation

A linear aircraft model has the form

$$\dot{\mathbf{x}}(t) = A(\theta,t)\mathbf{x}(t) + B(\theta,t)\mathbf{u}(t) + G(\theta,t)\mathbf{w}(t), \tag{1.1}$$

$$\dot{\mathbf{y}}(t) = C(\theta,t)\mathbf{x}(t) + D(\theta,t)\mathbf{u}(t) + \mathbf{v}(t), \tag{1.2}$$

where θ is a vector of **aircraft parameters**, $\mathbf{w}$ is the **system noise** and $\mathbf{v}$ is the **measurement noise**. Both $\mathbf{w}$ and $\mathbf{v}$ are assumed to be white noise, having covariance matrices $Q(\theta,t)$ and $R(\theta,t)$, respectively.

In flight tests, we often try to measure the state $\mathbf{x}$. From this, we then determine/estimate the aircraft parameters θ. Such a problem is called a **parameter estimation** problem.

Conversely, it also often occurs that we know the aircraft model, including the parameters θ. However, we do not know the state of the aircraft. (Measuring it directly may be impossible due to measurement noise.) This problem of determining the state is called a **state estimation** problem.

Finally, it may also occur that we know neither the state $\mathbf{x}$ or the parameters θ. We are now dealing with a **joint state and parameter estimation** problem.

2 Filtering, prediction and smoothing

We will now examine the problem of state estimation in more detail. There are three ways of state estimation, depending on which data is available. In **prediction**, only past measurements are available.

In **filtering**, past and present measurements are available. Finally, in **smoothing**, past, present and future measurements are available.

2.1 Filtering

We'll start by examining the problem of filtering. (Later on we examine prediction and smoothing.) To be more precise, we'll look at the **Kalman filter** (KF). We will only do this for discrete-time systems. (Measurement data is usually discrete as well.) The system thus has the form

$$\begin{aligned} \underline{\mathbf{x}}(x+1) &= \Phi(x)(k)\underline{\mathbf{x}}(k) + \Psi(k)\mathbf{u}(k) + \Gamma(k)\underline{\mathbf{w_d}}(k), & (2.1)\\ \underline{\mathbf{y}}(k) &= C(k)\underline{\mathbf{x}}(k) + D(k)\mathbf{u}(k) + \underline{\mathbf{v}}(k), & (2.2) \end{aligned}$$

where $\underline{\mathbf{w_d}}$ now is the discrete version of the system noise $\mathbf{w}$. (It has covariance matrix Q_d.) Since we don't know the state $\mathbf{x}$, we also treat it as a random variable $\underline{\mathbf{x}}$. This also means that $\underline{\mathbf{x}}$ has a mean and a covariance matrix, respectively defined as

$$E\left\{\underline{\mathbf{x}}(k+1)\right\} = \hat{\underline{\mathbf{x}}}(k+1|k), \tag{2.3}$$

$$E\left\{(\underline{\mathbf{x}}(k+1) - \hat{\underline{\mathbf{x}}}(k+1|k))(\underline{\mathbf{x}}(k+1) - \hat{\underline{\mathbf{x}}}(k+1|k))^T\right\} = P(k+1|k). \tag{2.4}$$

By the way, the addition '$|k$' indicates that the expectations are taken, given all data available up to time k.

What we now would like to do is make a prediction of $\underline{\mathbf{x}}(k+1)$, given all data up to time $k+1$. (Remember, we're doing filtering.) This prediction should be such that it minimizes the cost function

$$\begin{aligned} J &= \frac{1}{2}\left(\underline{\mathbf{x}}(k+1) - \hat{\underline{\mathbf{x}}}(k+1|k)\right)^T P^{-1}(k+1|k)\left(\underline{\mathbf{x}}(k+1) - \hat{\underline{\mathbf{x}}}(k+1|k)\right) + & (2.5)\\ &\quad + \frac{1}{2}\left(\underline{\mathbf{y}}(k+1) - C(k+1)\underline{\mathbf{x}}(k+1)\right)^T R^{-1}(k+1)\left(\underline{\mathbf{y}}(k+1) - C(k+1)\underline{\mathbf{x}}(k+1)\right). & (2.6) \end{aligned}$$

The best estimate of $\underline{\mathbf{x}}(k+1)$ is denoted as $\hat{\underline{\mathbf{x}}}(k+1|k+1)$. To find it, we set $\partial J/\partial \mathbf{x} = \mathbf{0}$. After working out a lot of equations, the Kalman filter is derived.

2.2 The Kalman filter

Let's suppose that we know the mean $\hat{\underline{\mathbf{x}}}(k|k)$ and the covariance matrix $P(k|k)$ at time k, how do we find the data at time $k+1$? First, we can find the estimates of the data at time $k+1$ given the data at time k. We will have

$$\hat{\underline{\mathbf{x}}}(k+1|k) = \Phi(k)\hat{\underline{\mathbf{x}}}(k|k) + \Psi(k)\mathbf{u}(k), \tag{2.7}$$

$$P(k+1|k) = \Phi(k)P(k|k)\Phi^T(k) + \Gamma(k)Q_d(k)\Gamma^T(k). \tag{2.8}$$

Of course, if we use the data at time $k+1$ as well, we can get a better approximation of $\hat{\underline{\mathbf{x}}}(k+1)$. If we assume (without loss of generality) that $D = 0$, this approximation is found using

$$\hat{\underline{\mathbf{x}}}(k+1|k+1) = \hat{\underline{\mathbf{x}}}(k+1|k) + K(x+1)\left(\underline{\mathbf{y}}(k+1) - C(k+1)\hat{\underline{\mathbf{x}}}(k+1|k)\right). \tag{2.9}$$

By doing this, the covariance matrix $P(k+1|k+1)$ becomes

$$P(k+1|k+1) = \left(I - K(k+1)C(k+1)\right)P(k+1|k). \tag{2.10}$$

The question remains, which **Kalman gain matrix** $K(k+1)$ should we use? We want to use the matrix $K(k+1)$ that minimizes J. By using this fact, we can eventually find that

$$K(k+1) = P(k+1|k)C^T(k+1)\left(C(k+1)P(k+1|k)C^T(k+1) + R(k+1)\right)^{-1}. \tag{2.11}$$

(Note that you first have to find $K(k+1)$, before finding $\hat{\underline{\mathbf{x}}}(k+1|k+1)$ or $P(k+1|k+1)$.)

2.3 Prediction

Let's examine prediction. This time, we have data up to time k. So we know $\hat{\underline{\mathbf{x}}}(k|k)$ and $P(k|k)$. To find $\hat{\underline{\mathbf{x}}}$ and P at later times, we can make use of the simple recursive relations

$$\hat{\underline{\mathbf{x}}}(k+i+1|k) = \Phi(k+i)\hat{\underline{\mathbf{x}}}(k+i|k) + \Psi(k+i)\mathbf{u}(k+i), \tag{2.12}$$

$$P(k+i+1|k) = \Phi(k+i)P(k+i|k)\Phi^T(k+i) + \Gamma(k+i)Q(k+i)\Gamma^T(k+i). \tag{2.13}$$

There is one evident downside to prediction. Since $Q(k)$ is a positive definite matrix (for every time k), the above relation will mean that the covariance matrix $P(k+i|k)$ will grow with i. In other words, the further ahead you predict in time, the bigger your uncertainties are. Kind of makes sense, doesn't it?

There is also another problem. In the above relation, we see the future input $\mathbf{u}(k+i)$. Often we don't know this input yet. To solve this, we can simply assume that $\mathbf{u}(k+i) = \mathbf{u}(k)$ for all i.

2.4 Single stage smoothing

In smoothing, we can make a distinction between single stage and multistage smoothing. We'll start by examining single stage smoothing. Here, we have estimates $\hat{\underline{\mathbf{x}}}(0|0)$ and $\hat{\underline{x}}(1|1)$, as well as measurements $\underline{\mathbf{y}}(0)$ and $\underline{\mathbf{y}}(1)$. We now want to find a better estimate of $\hat{\underline{\mathbf{x}}}(0)$ using our data. This estimate is then denoted by $\hat{\underline{\mathbf{x}}}(0|1)$. It can be found using

$$\hat{\underline{\mathbf{x}}}(0|1) = \hat{\underline{\mathbf{x}}}(0|0) + P(0|0)\Phi^T(0)C^T(1)R^{-1}(1)\left(\underline{\mathbf{y}}(1) - C(1)\hat{\underline{\mathbf{x}}}(1|1)\right). \tag{2.14}$$

We can also use the data to find an estimate for the system noise $\underline{\mathbf{w}}(0)$ at time 0. We then use

$$\hat{\underline{\mathbf{w}}}(0|1) = \hat{\underline{\mathbf{w}}}(0|0) + Q(0)\Gamma^T(0)C^T(1)R^{-1}(1)\left(\underline{\mathbf{y}}(1) - C(1)\hat{\underline{\mathbf{x}}}(1|1)\right). \tag{2.15}$$

Note that, since $\underline{\mathbf{w}}$ is white noise, we have $\hat{\underline{\mathbf{w}}}(0|0) = 0$. However, we don't have $\hat{\underline{\mathbf{w}}}(0|1) = 0$. This might seem strange – white noise not having a zero mean. But $\hat{\underline{\mathbf{w}}}(0|1)$ actually is the expected value of $\hat{\underline{\mathbf{w}}}(0)$, given the measurements up to time 1. Of course, using the measured value of $\underline{\mathbf{y}}(1)$, you're probably able to derive some indication on what value $\hat{\underline{\mathbf{w}}}(0)$ had.

Alternatively, we can write

$$\hat{\underline{\mathbf{x}}}(0|1) = \hat{\underline{\mathbf{x}}}(0|0) + C_0\left(\hat{\underline{\mathbf{x}}}(1|0) - \hat{\underline{\mathbf{x}}}(1|1)\right), \qquad \text{with} \qquad C_0 = P(0|0)\Phi^T(0)P^{-1}(1|0), \tag{2.16}$$

$$\hat{\underline{\mathbf{w}}}(0|1) = \hat{\underline{\mathbf{w}}}(0|0) + B_0\left(\hat{\underline{\mathbf{x}}}(1|0) - \hat{\underline{\mathbf{x}}}(1|1)\right), \qquad \text{with} \qquad B_0 = Q(0|0)\Gamma^T(0)P^{-1}(1|0). \tag{2.17}$$

The resulting covariance matrices of $\underline{\mathbf{x}}$ and $\underline{\mathbf{w}}$ are now given by

$$P(0|1) = P(0|0) - C_0\left(P(1|0) - P(1|1)\right)C_0^T, \tag{2.18}$$

$$Q(0|1) = Q(0|0) - B_0\left(P(1|0) - P(1|1)\right)B_0^T. \tag{2.19}$$

Note that we generally have $Q(0|0) = Q(0)$. (At time 0, you can't have received any information on $Q(0)$ yet.) Also note that we don't necessarily have to do these steps for time $k = 0$. The above steps can be performed for any time k. (Just replace 0 by k and 1 by $(k+1)$.)

2.5 Multistage (Kalman) smoothing

Now suppose that we have measurements $\mathbf{y}$ for times 0 to N. We want to use all these measurements to improve our estimate of $\hat{\underline{\mathbf{x}}}(k)$ at some time k. This process is called multistage smoothing. The resulting value will be denoted by $\hat{\underline{\mathbf{x}}}(k|N)$.

We assume that the values of $\hat{\underline{\mathbf{x}}}(k|k)$ and $P(k|k)$ are already known for all times k (with $0 \leq k \leq N$). If not, then these values have to still be determined using a basic Kalman filtering, starting from time 0, going step by step to time N.

To find $\hat{\underline{\mathbf{x}}}(k|N)$ and $\hat{\underline{\mathbf{w}}}(k|N)$, we can use the recursive relations

$$\hat{\underline{\mathbf{x}}}(k|N) = \hat{\underline{\mathbf{x}}}(k|k) - C_k\left(\hat{\underline{\mathbf{x}}}(k+1|k) - \hat{\underline{\mathbf{x}}}(k+1|N)\right), \quad \text{with} \quad C_k = P(k|k)\Phi^T(k)P^{-1}(k+1|k), \tag{2.20}$$

$$\hat{\underline{\mathbf{w}}}(k|N) = \hat{\underline{\mathbf{w}}}(k|k) - B_k\left(\hat{\underline{\mathbf{x}}}(k+1|k) - \hat{\underline{\mathbf{x}}}(k+1|N)\right), \quad \text{with} \quad B_k = Q(k|k)\Gamma^T(k)P^{-1}(k+1|k). \tag{2.21}$$

(Note that the matrix $Q(k|k)$ here again means the same as $Q(k)$.) The covariance matrices of these two parameters are then given by

$$P(k|N) = P(k|k) - C_k\left(P(k+1|k) - P(k+1|N)\right)C_k^T, \tag{2.22}$$

$$Q(k|N) = Q(k|k) - B_k\left(P(k+1|k) - P(k+1|N)\right)B_k^T. \tag{2.23}$$

It is interesting to see that (contrary to the Kalman filter) we are now going back in time. So at every point in time, we are updating our data using information gained in all the future times.

3 The extended Kalman filter

3.1 Linearizing and discretizing a continuous nonlinear system

The Kalman filter only works for linear systems. In case we run into a nonlinear system, we need to extend the Kalman filter. In this part, we'll examine how that works. So, let's examine a continuous nonlinear system, like

$$\dot{\underline{\mathbf{x}}}(t) = \mathbf{f}(\underline{\mathbf{x}}(t), \mathbf{u}(t), t) + G(\underline{\mathbf{x}}(t), t)\underline{\mathbf{w}}(t), \tag{3.1}$$

$$\underline{\mathbf{y}}(t) = \mathbf{h}(\underline{\mathbf{x}}(t), \mathbf{u}(t), t) + \underline{\mathbf{v}}(t). \tag{3.2}$$

At a certain time t_k, the expected state is denoted as $\hat{\underline{\mathbf{x}}}(k|k)$. This state is also called the **nominal state** $\underline{\mathbf{x}}^*(t_k)$. (So per definition $\hat{\underline{\mathbf{x}}}(k|k) = \underline{\mathbf{x}}^*(t_k)$.) We will linearize the system about it. The deviations from the nominal values $\underline{\mathbf{x}}^*(t)$, $\mathbf{u}^*(t)$ and $\underline{\mathbf{y}}^*(t)$ are denoted as

$$\delta\underline{\mathbf{x}}(t) = \underline{\mathbf{x}}(t) - \underline{\mathbf{x}}^*(t), \qquad \delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}^*(t) \qquad \text{and} \qquad \delta\underline{\mathbf{y}}(t) = \underline{\mathbf{y}}(t) - \underline{\mathbf{y}}^*(t). \tag{3.3}$$

We can now linearize the system about the nominal state. The linearized system will take the form of

$$\delta\dot{\underline{\mathbf{x}}}(t) = F_x(\underline{\mathbf{x}}^*(t), \mathbf{u}^*(t), t)\delta\underline{\mathbf{x}}(t) + G(\underline{\mathbf{x}}^*, t)\underline{\mathbf{w}}(t) \tag{3.4}$$

$$\delta\underline{\mathbf{y}}(t) = H_x(\underline{\mathbf{x}}^*(t), \mathbf{u}^*(t), t)\delta\underline{\mathbf{x}}(t) + \underline{\mathbf{v}}(t). \tag{3.5}$$

Here, the **Jacobians** F_x and H_x are defined as the derivatives of $\mathbf{f}$ and $\mathbf{h}$ at the nominal state, so

$$F_x(\underline{\mathbf{x}}^*(t), \mathbf{u}^*(t), t) = \left.\frac{\partial}{\partial\underline{\mathbf{x}}}\mathbf{f}(\underline{\mathbf{x}}(t), \mathbf{u}(t), t)\right|_{\underline{\mathbf{x}}(t)=\underline{\mathbf{x}}^*(t), \mathbf{u}(t)=\mathbf{u}^*(t)} \tag{3.6}$$

and similarly for H_x. Also, we have made the assumption that $G(\underline{\mathbf{x}}(t), t) \approx G(\underline{\mathbf{x}}^*(t), t)$. This linear system can then be discretized. We wind up with

$$\delta\underline{\mathbf{x}}(k+1) = \Phi(k+1, k; *)\delta\underline{\mathbf{x}}(k) + \Gamma(k+1, k; *)\underline{\mathbf{w}}_\mathbf{d}(k), \tag{3.7}$$

$$\delta\underline{\mathbf{y}}(k) = H_x(k; *)\delta\underline{\mathbf{x}}(k) + \underline{\mathbf{v}}(k+1), \tag{3.8}$$

where the **discrete system matrix** for the nominal state $\Phi(k+1, k; *)$ can be found using

$$\Phi(k+1, k; *) = e^{(t_{k+1}-t_k)F_x} = \sum_{n=0}^{\infty}\frac{(t_{k+1}-t_k)^n F_x^n}{n!}. \tag{3.9}$$

(The star $*$ means that the matrix is derived from a linearization about the nominal state $\underline{\mathbf{x}}^*$.) We can then apply the **Extended Kalman Filter** (EKF) to this discrete linear system.

3.2 Application of the Kalman filter

Let's assume that we have an estimate of $\underline{\hat{\mathbf{x}}}(k|k)$ at time k. We also know the corresponding covariance matrix $P(k|k)$. We can now predict the values of these two parameters at time $k+1$. This is done using

$$\underline{\hat{\mathbf{x}}}(k+1|k) = \underline{\hat{\mathbf{x}}}(k|k) + \int_{t_k}^{t_{k+1}} \mathbf{f}(\underline{\hat{\mathbf{x}}}(t|t_k), \mathbf{u}^*(t), t)\, dt, \tag{3.10}$$

$$P(k+1|k) = \Phi(k+1,k;*)P(k|k)\Phi^T(k+1,k;*) + \Gamma(k+1,k;*)Q_d(k+1,k;*)\Gamma(k+1,k;*). \tag{3.11}$$

Of course, using measurements from time $k+1$, we can update the estimate and its covariance matrix. This is done according to

$$\underline{\hat{\mathbf{x}}}(k+1|k+1) = \underline{\hat{\mathbf{x}}}(k+1|k) + K(k+1;*)\left(\underline{\mathbf{y}}(k+1) - \mathbf{h}(\underline{\hat{\mathbf{x}}}(k+1|k), \mathbf{u}^*(k+1), k+1)\right), \tag{3.12}$$

$$P(k+1|k+1;*) = (I - K(k+1;*)H_x(k+1;*))\, P(k+1|k;*). \tag{3.13}$$

Again the question remains which Kalman matrix $K(k+1;*)$ is the best. This time, it's the matrix

$$K(k+1;*) = P(k+1|k;*)H_x^T(k+1;*)\left(H_x(k+1;*)P(k+1|k;*)H_x^T(k+1;*) + R(k+1)\right)^{-1}. \tag{3.14}$$

3.3 The iterated extended Kalman filter

The EKF has a big downside. It uses linearization. But if the actual state is far away from the expected state, the linearization might not apply. The model may diverge. To reduce this problem, the **Iterated Extended Kalman Filter** (IEKF) can be used.

The basic idea of the Kalman filter is to update $\underline{\hat{\mathbf{x}}}(k+1|k)$ to find $\underline{\hat{\mathbf{x}}}(k+1|k+1)$. Previously, we linearized about the nominal state $\underline{\mathbf{x}}^*(k+1) = \underline{\hat{\mathbf{x}}}(k+1|k)$ to do this compensation. In the IEKF, we do the same, except that we do it multiple times. But in every iteration we use a different nominal state $\underline{\mathbf{x}}^*(k+1)$.

We start with the nominal state $\underline{\mathbf{x}}_\mathbf{1}^*(k+1) = \underline{\hat{\mathbf{x}}}(k+1|k)$. (For simplicity of notation, we write $\underline{\mathbf{x}}_\mathbf{i}^*(\mathbf{k+1})$ as $\underline{\eta}_\mathbf{i}$.) We then update $\underline{\eta}_\mathbf{i} = \underline{\mathbf{x}}_\mathbf{i}^*(k+1)$ recursively, through

$$\underline{\eta}_{\mathbf{i+1}} = \underline{\hat{\mathbf{x}}}(k+1|k) + K(k+1;\underline{\eta}_\mathbf{i})\left(\underline{\mathbf{y}}(k+1) - \mathbf{h}(\underline{\eta}_\mathbf{i}, \mathbf{u}(k+1), k+1) - H_x(k+1;\underline{\eta}_\mathbf{i})(\underline{\hat{\mathbf{x}}}(k+1|k) - \underline{\eta}_\mathbf{i})\right). \tag{3.15}$$

The term on the right might be surprising, since it didn't occur in the previous equation for $\underline{\hat{\mathbf{x}}}(k+1|k+1)$. This is because, in the first iteration, we have $\eta_\mathbf{i} = \underline{\hat{\mathbf{x}}}(k+1|k)$, so the term dropped out.

After several iterations, the nominal state $\eta_\mathbf{i}$ doesn't change much anymore. In fact, we can use as stopping condition

$$\frac{|\eta_\mathbf{l} - \eta_{\mathbf{l-1}}|}{|\eta_{\mathbf{l-1}}|} \leq \varepsilon, \tag{3.16}$$

for a certain ε. The number of iterations is then denoted by l. When the above condition is reached, we set $\underline{\hat{\mathbf{x}}}(k+1|k+1)$ and $P(k+1|k+1)$ as

$$\underline{\hat{\mathbf{x}}}(k+1|k+1) = \eta_\mathbf{l}, \tag{3.17}$$

$$P(k+1|k+1) = (I - K(k+1;\eta_\mathbf{l})H_x(k+1;\eta_\mathbf{l}))\, P(k+1|k;\eta_\mathbf{l})\, (I - K(k+1;\eta_\mathbf{l})H_x(k+1;\eta_\mathbf{l}))^T + K(k+1;\eta_\mathbf{l})R(k+1)K^T(k+1;\eta_\mathbf{l}). \tag{3.18}$$

In case of significant nonlinearities, this version of the Kalman filter gives a more exact approximation of $\underline{\hat{\mathbf{x}}}(k+1|k+1)$ than the original version.

4 The least-squares method

4.1 Least-squares definitions

In this part, we'll examine the **least-squares method**. This is a form of off-line parameter estimation. Suppose that we have a set of measurements of the form

$$\underline{\mathbf{y}}(k) = \mathbf{f}(k, \mathbf{p}) + \underline{\mathbf{e}}(k), \tag{4.1}$$

where $\underline{\mathbf{y}}$ is the **measurement vector**, $\mathbf{f}$ is a known vector function, $\underline{\mathbf{e}}(k)$ is the unknown **error vector** and k is the measurement number. Our goal is to find the **parameter vector p** such that the model $\mathbf{f}(k, \mathbf{p})$ matches the measurements $\underline{\mathbf{y}}(k)$ as well as possible.

Let's say that we already have some estimate $\hat{\mathbf{p}}$ of the parameter vector $\mathbf{p}$. The least-squares method can only be applied to linear functions. So we apply linearization to the set of measurements about our estimate $\hat{\mathbf{p}}$. This turns the above relation into

$$\underline{\mathbf{y}}(k) = \mathbf{f}(k, \hat{\mathbf{p}}) + \left(\frac{\partial f(k, \mathbf{p})}{\partial \mathbf{p}} \right)_{\mathbf{p}=\hat{\mathbf{p}}} (\mathbf{p} - \hat{\mathbf{p}}) + \underline{\mathbf{e}}(k) = \mathbf{f}(k, \hat{\mathbf{p}}) + h(k, \hat{\mathbf{p}})\delta\mathbf{p} + \underline{\mathbf{e}}(k), \tag{4.2}$$

where we have defined $h(k, \hat{\mathbf{p}})$ as the derivative $\partial f / \partial \mathbf{p}$ at $\mathbf{p} = \hat{\mathbf{p}}$ and $\delta\mathbf{p} = (\mathbf{p} - \hat{\mathbf{p}})$. (Note that $h(k, \hat{\mathbf{p}})$ is in fact a matrix.) The **residual** of the provisional fit $\mathbf{f}(k, \hat{\mathbf{p}})$ is now denoted by

$$\underline{\mathbf{z}}(k) = \underline{\mathbf{y}}(k) - \mathbf{f}(k, \hat{\mathbf{p}}) = h(k, \hat{\mathbf{p}})\delta\mathbf{p} + \underline{\mathbf{e}}(k). \tag{4.3}$$

Of course, we don't have just one measurement. We have $k = 1 \ldots m$ measurements. With all these measurements, we can assemble

$$\underline{\mathbf{Z}} = \begin{bmatrix} \underline{\mathbf{z}}(1) \\ \underline{\mathbf{z}}(2) \\ \vdots \\ \underline{\mathbf{z}}(m) \end{bmatrix}, \qquad H = \begin{bmatrix} H(1) \\ H(2) \\ \vdots \\ H(m) \end{bmatrix} \qquad \text{and} \qquad \underline{\mathbf{E}} = \begin{bmatrix} \underline{\mathbf{e}}(1) \\ \underline{\mathbf{e}}(2) \\ \vdots \\ \underline{\mathbf{e}}(m) \end{bmatrix}. \tag{4.4}$$

We thus have $\underline{\mathbf{Z}} = H\delta\mathbf{q} + \underline{\mathbf{E}}$. (By the way, don't confuse the assembled matrix H with the individual matrices $H(k)$.)

4.2 The least-squares solution

In the least-squares problem, we would like to find the value of $\mathbf{p}$ (or similarly, $\delta\mathbf{p}$) which gives the best fit. That is, the value which would minimize the **cost function**

$$V(\mathbf{p}) = \frac{1}{2} \sum_{i=1}^{m} \underline{\mathbf{z}}^T(i) W^{-1}(i) \underline{\mathbf{z}}(i) = \frac{1}{2} \underline{\mathbf{Z}}^T W^{-1} \underline{\mathbf{Z}}, \tag{4.5}$$

with m the number of measurements. Also, the matrices $W(i)$ are symmetric **weight matrices**. Assemble them diagonally to find the assembled matrix $W = \text{diag}(W(1), \ldots, W(m))$. (Often, we select $W(i) = E\left(\underline{\mathbf{z}}(i)\underline{\mathbf{z}}^T(i)\right)$ and thus $W = E\left(\underline{\mathbf{Z}}\underline{\mathbf{Z}}^T\right)$.) Differentiation could now show that the optimal value for $\delta\mathbf{p}$ satisfies

$$H^T W^{-1} H \delta\mathbf{p} = H^T W^{-1} \underline{\mathbf{Z}}, \qquad \text{so,} \qquad \delta\mathbf{p} = (H^T W^{-1} H)^{-1} H^T W^{-1} \underline{\mathbf{Z}}. \tag{4.6}$$

The covariance matrix of this parameter estimation error gives an indication of the accuracy of our estimate. This matrix equals

$$E\left(\delta\mathbf{p}\delta\mathbf{p}^T\right) = (H^T W^{-1} H)^{-1} H^T W^{-1} E\left(\underline{\mathbf{Z}}\underline{\mathbf{Z}}^T\right) W^{-1} H (H^T W^{-1} H)^{-1} = (H^T W^{-1} H)^{-1}. \tag{4.7}$$

Of course, this whole estimate is based on the linearization of $\mathbf{f}(k, \mathbf{p})$ about $\hat{\mathbf{p}}$. In case there are non-linearities, multiple iterations might be necessary. If we do this, then during every iteration we linearize $\mathbf{f}(k, \mathbf{p})$ about a new point $\hat{\mathbf{p}}_\mathbf{i}$. This gives a new value for $\delta\mathbf{p}$. We then update our parameter estimate according to

$$\hat{\mathbf{p}}_{\mathbf{i+1}} = \hat{\mathbf{p}}_\mathbf{i} + \delta\mathbf{p}. \tag{4.8}$$

After several iterations the algorithm will converge. And, unless the algorithm gets stuck in a local minimum, this converged value $\hat{\mathbf{p}}_\mathbf{l}$ is the optimal value for $\mathbf{p}$.

4.3 Recursive least-squares

Previously, we have considered the least-squares method as an offline method. We just had a batch of information and needed to derive the parameter vector $\mathbf{p}$. Now we'll try to expand it to an online method. However, for simplicity, we'll only examine the linear problem,

$$\underline{\mathbf{y}}(k) = H(k)\mathbf{p} + \underline{\mathbf{e}}(k). \tag{4.9}$$

(Note that, because the model is linear, we have $\underline{\mathbf{z}}(k) = \underline{\mathbf{e}}(k)$.) Let's suppose that we have data up to time k. The parameter estimate is thus given by

$$\hat{\mathbf{p}}(k) = \left(\sum_{i=1}^{k} H^T(i) W^{-1}(i) H(i) \right)^{-1} \sum_{i=1}^{k} H^T W^{-1}(i) \underline{\mathbf{y}}(i). \tag{4.10}$$

(Note that this is essentially the same equation as equation (4.6).) Now let's suppose we obtain the measurement vector $\underline{\mathbf{y}}(k+1)$ at time $k+1$. It would be nice if we could find $\hat{\mathbf{p}}(k+1)$ from $\hat{\mathbf{p}}(k)$, without having to perform the whole summation from time $k = 1$ all over again. To do this, we first define

$$R(k) = \sum_{i=1}^{k} H^T(i) W^{-1}(i) H(i). \tag{4.11}$$

It can then be shown that

$$\hat{\mathbf{p}}(k+1) = \hat{\mathbf{p}}(k) + R^{-1}(k+1) H^T(k+1) W^{-1}(i+1) \left(\underline{\mathbf{y}}(k+1) - H(k+1)\hat{\mathbf{p}}(k) \right). \tag{4.12}$$

Note that the term between brackets $(\underline{\mathbf{y}} - H\hat{\mathbf{p}})$ can, roughly speaking, be seen as the error in the predicted output. This is left-multiplied by $R^{-1}H^T W^{-1}$ to find the error in the parameter vector.

The question remains, how do we find $R^{-1}(k+1)$? We could use

$$R^{-1}(k+1) = \left(R(k) + H^T(k+1) W^{-1}(k+1) H(k+1) \right)^{-1}, \tag{4.13}$$

but this requires the inversion of a matrix of size $q \times q$ (with q the size of the parameter vector $\mathbf{p}$). An alternative is to use

$$R^{-1}(k+1) = \left(I - R^{-1}(k) H^T(k+1) W^{-1}(k+1) \left(H(k+1) R^{-1}(k) H^T(k+1) + W \right)^{-1} H(t) \right) R^{-1}(k). \tag{4.14}$$

This requires the inversion of a matrix of size $n \times n$ (with n the size of the measurement vector $\mathbf{y}$). Which option is best depends on q and n.

5 The maximum likelihood method

5.1 The idea behind the maximum likelihood method

We have seen a method for state estimation (the Kalman filter) and parameter estimation (the least-squares method). Now we'll examine a method that can do both: the **maximum likelihood (ML)**

method. For this reason, we examine a general system of the form

$$\dot{\underline{\mathbf{x}}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \theta) + G(\mathbf{x}(t))\underline{\mathbf{w}}(t), \tag{5.1}$$

$$\underline{\mathbf{y}}(k) = \mathbf{h}(\underline{\mathbf{x}}(t_k), \mathbf{u}(t_k), \theta) + \underline{\mathbf{v}}(k). \tag{5.2}$$

It is assumed that measurements are only taken at certain times t_k. Also, we assume that the covariance matrices $Q(\theta)$ and $R(\theta)$, of the noises $\underline{\mathbf{w}}$ and $\underline{\mathbf{v_k}}$, respectively, both depend on the parameter vector θ, but that they do not vary with time t.

We define the **prediction error vector** $\hat{\underline{\varepsilon}}(k+1|k)$ at time $k+1$, given the data at time k, as the difference between the measured output and the expected output

$$\hat{\underline{\varepsilon}}(k+1|k;\theta) = \underline{\mathbf{y}}(k+1) - \hat{\underline{\mathbf{y}}}(k+1|k;\theta), \qquad \text{with} \qquad \hat{\underline{\mathbf{y}}}(k+1|k;\theta) = \mathbf{h}(\hat{\underline{\mathbf{x}}}(k+1|k), \mathbf{u}(k+1), \theta). \tag{5.3}$$

We denote the probability that this error vector occurs, given a parameter vector θ, as $p(\hat{\underline{\varepsilon}}(k+1|k;\theta)|\theta)$, or more briefly, as $p(\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)|\theta)$. The function p is known as the **probability density function**. We generally assume that it is Gaussian. So,

$$p(\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)) = ((2\pi)^q \det(\Lambda_{k+1,k}(\theta)))^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)^T)\Lambda^{-1}_{k+1,k}(\theta)\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta))\right). \tag{5.4}$$

But of course, we don't have only the measurements at time $k+1$. We have measurements from time $k=1$ to some time $k=m$. We assume that all these measurements are independent. In this case, the probability that all errors $\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)$ (with $k = 0 \ldots m-1$) occurred equals

$$p(\hat{\underline{\varepsilon}}_{\mathbf{1,0}}(\theta), \ldots, \hat{\underline{\varepsilon}}_{\mathbf{m,m-1}}(\theta)) = \prod_{k=0}^{m-1} p(\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)). \tag{5.5}$$

In the maximum likelihood method, we choose our parameter vector θ such that the actual situation is the most likely situation that could occur. In other words, we choose θ such that $p(\hat{\underline{\varepsilon}}_{\mathbf{1,0}}(\theta), \ldots, \hat{\underline{\varepsilon}}_{\mathbf{m,m-1}}(\theta))$ is maximized. This is is equivalent to using (and minimizing) the cost function

$$V_m(\theta) = -\log\left(\prod_{k=0}^{m-1} \left(\det(\Lambda_{k+1,k}(\theta))\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)^T)\Lambda^{-1}_{k+1,k}(\theta)\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)\right)\right) \tag{5.6}$$

$$= \frac{1}{2}\sum_{k=0}^{m-1}\left(\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta)^T\Lambda^{-1}_{k+1,k}(\theta)\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\theta) + \ln\left(\det(\Lambda_{k+1,k}(\theta))\right)\right). \tag{5.7}$$

Note that, in the above equation, we have dropped away the constant $(2\pi)^q$ terms. When minimizing a function, constant values in the function don't matter, so they can be removed.

5.2 Minimizing the cost function

A difficult question is, how do we minimize the cost function V_m? This is in fact done iteratively. At some point, we have an estimate $\theta_{\mathbf{i}}$. We then find a new (and hopefully improved) estimate $\theta_{\mathbf{i+1}}$ using

$$\hat{\theta}_{\mathbf{i+1}} = \hat{\theta}_{\mathbf{i}} + \alpha_i \xi_{\mathbf{i}}. \tag{5.8}$$

Here, α_i is a constant scalar value, similar to a learning rate. The difficulty lies in choosing the direction of adaptation $\xi_{\mathbf{i}}$. There are several ways to do this. In the **Newton-Raphson iteration method**, we use

$$\xi_{\mathbf{i}} = -\left(\frac{\partial^2 V_m(\theta)}{\partial\theta\partial\theta^T}\right)^{-1} \left.\frac{\partial V_m(\theta)}{\partial\theta}\right|_{\theta=\hat{\theta}_{\mathbf{i}}}. \tag{5.9}$$

The problem is that the second partial derivative of V_m may not always be invertible. As a solution to this, we could use the expectation matrix $E(\partial^2 V_m(\theta)/\partial\theta\partial\theta^T)$ (also known as the **Fisher information matrix**) and invert that instead. We are then actually using the **Gauss-Newton iteration method**.

Finding the Fisher information matrix can be quite difficult though. The reason lies in the error vectors $\hat{\underline{\varepsilon}}$. They ought to be derived from the (extended) Kalman filter. This is computationally quite a hard problem. As a solution, we could also ignore the noise and assume that the state equations are deterministic. This significantly simplifies the problem. The resulting method is called the **output error method**.

5.3 The recursive maximum likelihood method

The above offline method can be turned into an online method, called the **recursive maximum likelihood (RML) method**. The idea behind this is actually quite simple. At some point in time t_k, we are simply iterating using equation (5.8), just like in the normal ML method. But then, at time t_{k+1}, we get a new measurement $\underline{\mathbf{y}}(k+1)$. We add this measurement to our data, and use the new data set in our iterations.

The difficulty arises in updating the partial derivatives $\partial V_k/\partial\theta$ and $\partial^2 V_k(\theta)/\partial\theta\partial\theta^T$, such that they include the new measurement $\underline{\mathbf{y}}(k+1)$. There are methods to do this. However, explaining these methods would be too complicated for a summary. Instead, we'll just give you the equations.

Let's denote θ as the last found value for the parameter vector, with data up to time k. We then have

$$\left.\frac{\partial V_{k+1}(\theta)}{\partial\theta}\right|_{\theta=\hat{\theta}} = \left.\frac{\partial V_k(\theta)}{\partial\theta}\right|_{\theta=\hat{\theta}} + \frac{\partial\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\hat{\theta})^T}{\partial\theta}\Lambda^{-1}_{k+1,k}(\hat{\theta})\hat{\underline{\varepsilon}}_{\mathbf{k+1}}(\hat{\theta}) + \frac{1}{2}\left(\eta_{\mathbf{k+1}}(\hat{\theta}) - \mu_{\mathbf{k+1}}(\hat{\theta})\right), \tag{5.10}$$

$$\left(\frac{\partial^2 V_{k+1}(\theta)}{\partial\theta\partial\theta^T}\right) = \left(\frac{\partial^2 V_k(\theta)}{\partial\theta\partial\theta^T}\right) + \frac{\partial\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\hat{\theta})^T}{\partial\theta}\Lambda^{-1}_{k+1,k}(\hat{\theta})\frac{\partial\hat{\underline{\varepsilon}}_{\mathbf{k+1,k}}(\hat{\theta})}{\partial\theta} + \frac{1}{2}\eta_{\mathbf{k+1}}(\hat{\theta})\eta_{\mathbf{k+1}}(\hat{\theta})^T. \tag{5.11}$$

The terms $\eta_{\mathbf{k+1}}(\hat{\theta})$ and $\mu_{\mathbf{k+1}}(\hat{\theta})$ are, respectively, defined as

$$\eta_{\mathbf{k+1}}(\theta) = \begin{bmatrix} \mathrm{tr}\left(\Lambda^{-1}_{k+1,k}(\theta)\frac{\partial\Lambda_{k+1,k}(\theta)}{\partial\theta_1}\right) \\ \vdots \\ \mathrm{tr}\left(\Lambda^{-1}_{k+1,k}(\theta)\frac{\partial\Lambda_{k+1,k}(\theta)}{\partial\theta_q}\right) \end{bmatrix}, \tag{5.12}$$

$$\mu_{\mathbf{k+1}}(\theta) = \begin{bmatrix} \hat{\underline{\varepsilon}}_{k+1,k}(\theta)^T\Lambda^{-1}_{k+1,k}(\theta)\frac{\partial\Lambda_{k+1,k}(\theta)}{\partial\theta_1}\Lambda^{-1}_{k+1,k}(\theta)\hat{\underline{\varepsilon}}_{k+1,k}(\theta) \\ \vdots \\ \hat{\underline{\varepsilon}}_{k+1,k}(\theta)^T\Lambda^{-1}_{k+1,k}(\theta)\frac{\partial\Lambda_{k+1,k}(\theta)}{\partial\theta_q}\Lambda^{-1}_{k+1,k}(\theta)\hat{\underline{\varepsilon}}_{k+1,k}(\theta) \end{bmatrix}. \tag{5.13}$$

There are several further possibilities to simplify the (recursive) maximum likelihood method. However, these methods are all too extensive for this summary, so we won't discuss them here.

6 System identification applied to an aircraft

6.1 The input, output, state and parameters

It is time to see how we can apply system identification techniques to an aircraft. The input and state of an aircraft are

$$\mathbf{u} = \begin{bmatrix}\delta_e & \delta_a & \delta_r & T_c\end{bmatrix}^T, \tag{6.1}$$

$$\mathbf{x} = \begin{bmatrix}u & v & w & x & y & z & p & q & r & \phi & \theta & \psi\end{bmatrix}^T. \tag{6.2}$$

But what about the output? In an aircraft, we have several measurement devices. There is the **Inertial Measurement Unit** (IMU), which measures the accelerations $A_x = X/m$, $A_y = Y/m$ and $A_z = Z/m$, as well as the angular rates p, q and r. (It does this by integrating angular accelerations.) Next to this, there are air data measurements, which provide V, α and β, and there is the GPS system. If we put it all together, we will have

$$\begin{aligned}\mathbf{y} &= [A_{x_m} \quad A_{y_m} \quad A_{z_m} \quad p_m \quad q_m \quad r_m \quad x_{m_{GPS}} \quad y_{m_{GPS}} \quad z_{m_{GPS}} \\ &\quad u_{m_{GPS}} \quad v_{m_{GPS}} \quad w_{m_{GPS}} \quad \phi_{m_{GPS}} \quad \theta_{m_{GPS}} \quad \psi_{m_{GPS}} \quad V_m \quad \alpha_m \quad \beta_m]^T . \end{aligned} \tag{6.3}$$

The subscript m indicates that these are not actual values but measured values. A noise will be present in the signals. (So $A_{x_m} = A_x + v_{A_x}$, and similarly for the other parameters.)

Finally, there are the parameters that need to be estimated. These parameters include the IMU biases λ_x, λ_y, λ_z, λ_p, λ_q and λ_r, defined such that

$$A_{x_m} = A_x + \lambda_x + w_x, \qquad A_{y_m} = A_y + \lambda_y + w_y, \qquad A_{z_m} = A_z + \lambda_z + w_z, \tag{6.4}$$

$$p_m = p + \lambda_p + w_p, \qquad q_m = q + \lambda_q + w_q, \qquad r_m = r + \lambda_r + w_r. \tag{6.5}$$

Other parameters are the wind velocities W_{x_E}, W_{y_E} and W_{z_E}. (These velocities are expressed in the Earth-based reference frame.) And there are the aerodynamic parameters, like C_{X_0}, C_{X_α}, $C_{X_\alpha^2}$, C_{X_q}, $C_{X_{\delta_e}}$ and so on. (For a complete overview of all aerodynamic parameters, see relation (6.25).)

6.2 The aircraft model

To apply system identification techniques, we need some sort of model of the aircraft. This model has the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \theta), \tag{6.6}$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \theta). \tag{6.7}$$

So let's see what we can calculate with our state. First of all, we can calculate the force and moment coefficients, using

$$C_X = C_{X_0} + C_{X_\alpha}\alpha + C_{X_\alpha^2}\alpha^2 + C_{X_q}\frac{q\bar{c}}{V} + C_{X_{\delta_e}}\delta_e + C_{X_{T_c}}T_c, \tag{6.8}$$

$$C_Y = C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{pb}{2V} + C_{Y_r}\frac{rb}{2V} + C_{X_{\delta_a}}\delta_a + C_{X_{\delta_r}}\delta_r, \tag{6.9}$$

$$C_Z = C_{Z_0} + C_{Z_\alpha}\alpha + C_{Z_q}\frac{q\bar{c}}{V} + C_{Z_{\delta_e}}\delta_e, \tag{6.10}$$

$$C_l = C_{l_0} + C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r, \tag{6.11}$$

$$C_m = C_{m_0} + C_{l_\alpha}\alpha + C_{m_q}\frac{q\bar{c}}{V} + C_{m_{\delta_e}}\delta_e, \tag{6.12}$$

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r. \tag{6.13}$$

We can use these coefficients to calculate the actual forces and moments acting on the aircraft. This is done according to

$$X = \frac{1}{2}\rho V^2 S C_x, \qquad Y = \tfrac{1}{2}\rho V^2 S C_y, \qquad Z = \frac{1}{2}\rho V^2 S C_z, \tag{6.14}$$

$$L = \frac{1}{2}\rho V^2 S b C_l, \qquad M = \tfrac{1}{2}\rho V^2 S \bar{c} C_m, \qquad N = \frac{1}{2}\rho V^2 S b C_n. \tag{6.15}$$

The derivatives of u, v and w can now be found using

$$\dot{u} = \frac{X}{m} - qw + rv - g\sin\theta, \tag{6.16}$$

$$\dot{v} = \frac{Y}{m} - ru + pw + g\cos\theta\sin\phi, \tag{6.17}$$

$$\dot{w} = \frac{Z}{m} - pv + qu + g\cos\theta\cos\phi. \tag{6.18}$$

The derivatives of p, q and r satisfy $\mathbf{M} = I\dot{\omega} + \omega \times I\omega$ or, equivalently,

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix} I_x & 0 & -I_{xz}\\ 0 & I_y & 0\\ -I_{zx} & 0 & I_z\end{bmatrix}^{-1}\begin{bmatrix}L\\ M\\ N\end{bmatrix} - \begin{bmatrix} I_x & 0 & -I_{xz}\\ 0 & I_y & 0\\ -I_{zx} & 0 & I_z\end{bmatrix}^{-1}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \times \begin{bmatrix} I_x & 0 & -I_{xz}\\ 0 & I_y & 0\\ -I_{zx} & 0 & I_z\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}. \tag{6.19}$$

We can find $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ using the kinematic equations for rotational motion

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta, \tag{6.20}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi, \tag{6.21}$$

$$\dot{\psi} = q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta}. \tag{6.22}$$

Finally, we can find the derivatives of the position coordinates using

$$\begin{bmatrix}\dot{x}_E\\ \dot{y}_E\\ \dot{z}_E\end{bmatrix} = \begin{bmatrix}\cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}u\\ v\\ w\end{bmatrix} + \begin{bmatrix}W_{x_E}\\ W_{y_E}\\ W_{z_E}\end{bmatrix}. \tag{6.23}$$

(The above matrix L_{EB} transforms the velocities from the body reference frame to the Earth-based reference frame.) The model above is now ready to be used in a state/parameter estimation method.

6.3 The two-step method

Finding both the state and the parameters of a system simultaneously can be difficult. For an aircraft, there is a technique which makes this a bit easier. It is called the **two-step method**.

Remember that we want to know both the state values of $\mathbf{x}$ and the parameter values of θ. We'll now move the wind values W and the bias values λ from θ to $\mathbf{x}$, giving

$$\mathbf{x} = \begin{bmatrix}u & v & w & x & y & z & \phi & \theta & \psi & \lambda_x & \lambda_y & \lambda_z & \lambda_p & \lambda_q & \lambda_r & W_{x_E} & W_{y_E} & W_{z_E}\end{bmatrix}^T, \tag{6.24}$$

$$\theta = \begin{bmatrix}C_{X_0} & C_{X_\alpha} & C_{X_{\alpha^2}} & C_{X_q} & C_{X_{\delta_e}} & C_{X_{T_c}} & C_{Y_0} & C_{Y_\beta} & C_{Y_p} & C_{Y_r} & C_{Y_{\delta_a}} & C_{Y_{\delta_r}}\\ C_{Z_0} & C_{z_\alpha} & C_{Z_q} & C_{Z_{\delta_e}} & C_{l_0} & C_{l_\beta} & C_{l_p} & C_{l_r} & C_{l_{\delta_a}} & C_{l_{\delta_r}}\\ C_{m_0} & C_{m_\alpha} & C_{m_q} & C_{m_{\delta_e}} & C_{n_0} & C_{n_\beta} & C_{n_p} & C_{n_r} & C_{n_{\delta_a}} & C_{n_{\delta_r}}\end{bmatrix}^T. \tag{6.25}$$

In the two-step method, we assume that with all our measurement data, we can already estimate the state $\mathbf{x}$. So we don't need to use our aerodynamic model, with all its difficult coefficients, for that. Simply by combining IMU data, GPS data and air measurement data, we can derive the state of the aircraft with a state estimator like the Kalman filter. This is called the **first step** of the two-step method.

Once we know the state of the aircraft, we can start to estimate the aerodynamic parameters θ. (This is the **second step**.) To do this, we first predict the aerodynamic forces and moments X, Y, Z, L, M and N that were present. This is done based on the estimated state of the aircraft. We can then use equations (6.8) to (6.13) with the least-squares method to derive the aircraft parameters.

So we see that the two-step method splits the state and parameter estimation problem up into two steps, significantly simplifying the computations. This makes the two-step method a very useful method.