PixHawk - Dynamic Model and Sensor Fusion

Zurbriggen Florian

February 8, 2010

Contents

Abstract

This is the abstract

Part I

Dynamic Model

Chapter 1

Mechanical Frame

The used mechanical frame in the Pixhawk unit is based on a coaxial helicopter (figure 1.1). Both rotors are aligned at the same axis, have the same diameter and rotate in opposing direction to compensate the drag torques. The absolute values of force (thrust) and drag torque are driven by one electro motor on each rotor. These are called fixed pitch rotor systems, because there is no collective way to change the blade pitch angle. The sum of both thrust forces is used to lift the helicopter and the difference of the drag torques is used for yaw motions.

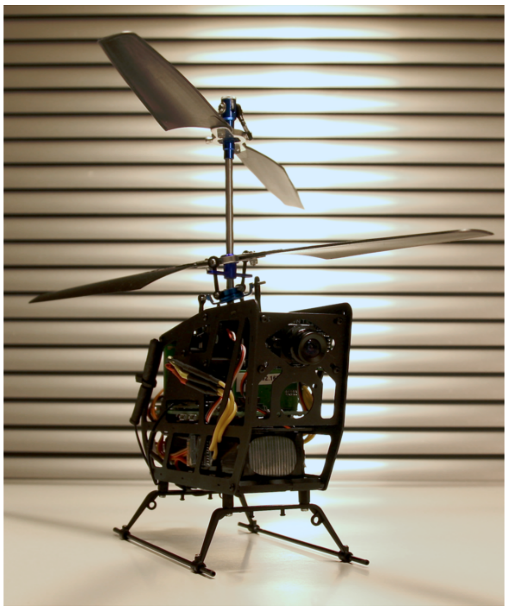


Figure 1.1: the used Pixhawk 1.0 unit

The thrust- and drag-vector of the upper rotor are fixed aligned in direction of rotor shaft. The initially mounted stabilizer is removed due to agility improvement. The direction of the thrust- and drag-vector of the lower rotor depends on the longitudinal flapping angle a and the lateral flapping angle b (see chapter 4). The flapping angles are given by the orientation of the cyclic swash plate, which can be controlled via two servos. By changing the direction of the thrust- and drag-vector of the lower rotor, one can influence the orientation of the helicopter.

Chapter 2

Overview of the System and Subsystems

The main system is divided into 3 subsystems: the dynamics of the rotor blades together with the two drive systems (dynamics rotor systems), the dynamics of the position and the dynamics of the orientation/Euler-angles (see figure 2.1).

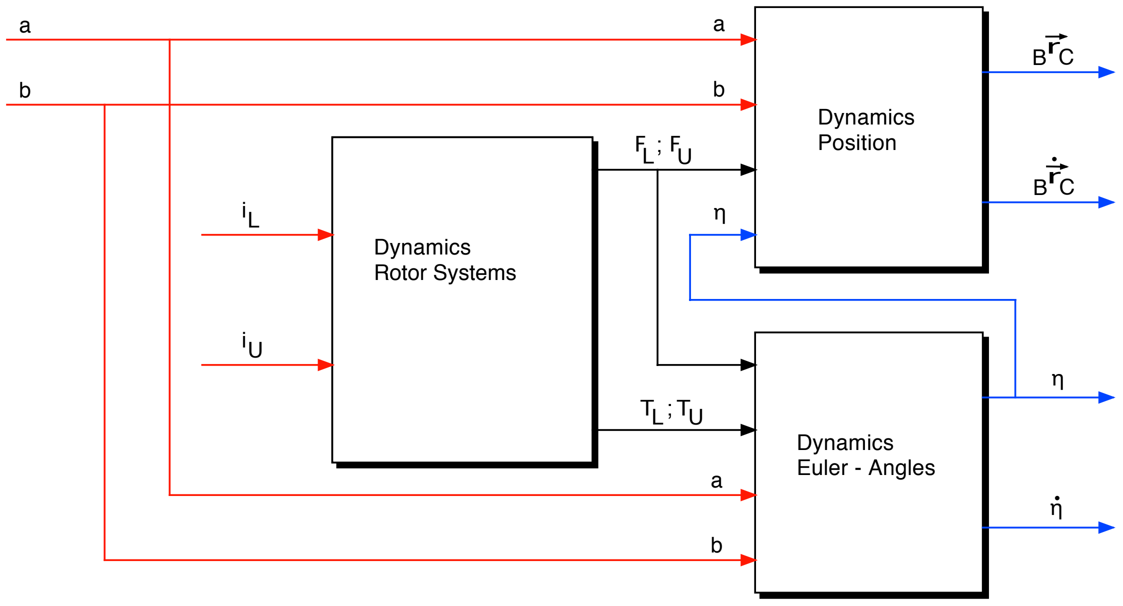


Figure 2.1: the different subsystems

The following signals are used:

a	longitudinal flapping angle
b	lateral flapping angle
i_L	input-/set-value of the lower rotor's drive system
i_U	input-/set-value of the upper rotor's drive system
F_L	absolute value of the lower rotor's thrust force

F_U	absolute value of the upper rotor's thrust force
T_L	absolute value of the lower rotor's drag torque
T_U	absolute value of the upper rotor's drag torque
${}_B\vec{r}_C$	the coordinates of the helicopter's cog (center of gravity) (see chapter 3)
${}_B\dot{\vec{r}}_C$	the time derivative of the cog's coordinates
η	the orientation of the helicopter containing the Euler angles (see chapter 3)
$\dot{\eta}$	the derivatives of the Euler angles

The dynamics of the rotor blades and drive systems:

inputs:	i_L, i_U
states:	F_L, F_U, T_L, T_U
outputs:	F_L, F_U, T_L, T_U

The dynamics of the position:

inputs:	a, b, F_L, F_U, η
states:	${}_B\vec{r}_C$, ${}_B\dot{\vec{r}}_C$
outputs:	${}_B\vec{r}_C$, ${}_B\dot{\vec{r}}_C$

The dynamics of the orientation:

inputs:	a, b, F_L, F_U, T_L, T_U
states:	η, $\dot{\eta}$
outputs:	η, $\dot{\eta}$

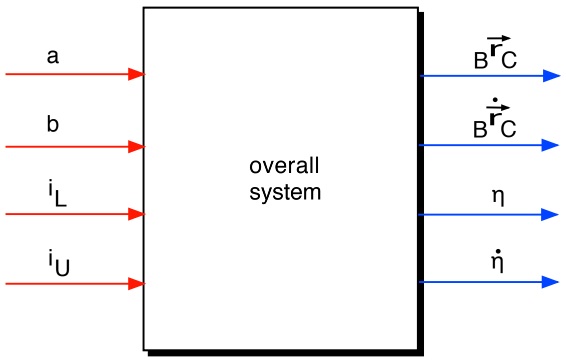


Figure 2.2: the overall system

The overall system (figure 2.2):

inputs:	a, b, i_L, i_U
states:	F_L, F_U, T_L, T_U, ${}_B\vec{r}_C$, ${}_B\dot{\vec{r}}_C$, η, $\dot{\eta}$
outputs:	${}_B\vec{r}_C$, ${}_B\dot{\vec{r}}_C$, η, $\dot{\eta}$

Remark 1. *In the derivation of the dynamic system, it is assumed, that every value of* ${}_B\vec{r}_C$*,* ${}_B\dot{\vec{r}}_C$*,* η *and* $\dot{\eta}$ *are given ("measured"). But in practice, e.g. for control applications, this won't be the case. This usually depends on the different sensors, sensor fusion etc.*

Chapter 3

Coordinate-Frames and -Transformations

There are basically two coordinate frames used to describe the dynamics of the Pixhawk helicopter. The first is the inertial frame I, which is fixed to the earth and the second is the body frame B, which is fixed to the helicopter. Both are right hand coordinate systems. A convention is used which is common in aviation problems. The origin of the body frame B is placed at the center of gravity (COG) of the helicopter. The coordinate x_B is pointing ahead in course direction, z_B is pointing downward and y_B is set according to the right hand rule. At the beginning, the inertial frame I and the body frame B are lying on each other. Figure 3.1 shows the coordinate frames. The index of the coordinate frame in which something is expressed is written on the left side. For example:

- ${}_I\vec{c}$ is some vector $\vec{c}$ expressed in the inertial frame I
- ${}_B\vec{c}$ is the same vector expressed in the body frame B

6 variables are needed to describe the body frame B in reference to the inertial frame I. First the vector ${}_I\vec{r}_{IB}$ pointing from the origin of I to the origin of B expressed in frame I:

$$ {}_I\vec{r}_{IB} = \begin{pmatrix} {}_Ix_{IB} \\ {}_Iy_{IB} \\ {}_Iz_{IB} \end{pmatrix} \tag{3.1}$$

In the following, the vector ${}_I\vec{r}_{IB}$ is defined in a short form as $\vec{r}_C$ representing the position of the center of gravity (COG) of the flying unit:

$$\vec{r}_C = \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} := {}_I\vec{r}_{IB} \tag{3.2}$$

Remark 2. *It is assumed that the helicopter is perfectly balanced in a way, that the center of gravity lies on the rotor shaft or its projection. I.e. the center of gravity lies on the z_B-axis of the body frame.*

Remark 3. *Instead of (3.2), ${}_B\vec{r}_C = {}_B\vec{r}_{IB}$ is mostly used in the description of the dynamics.*

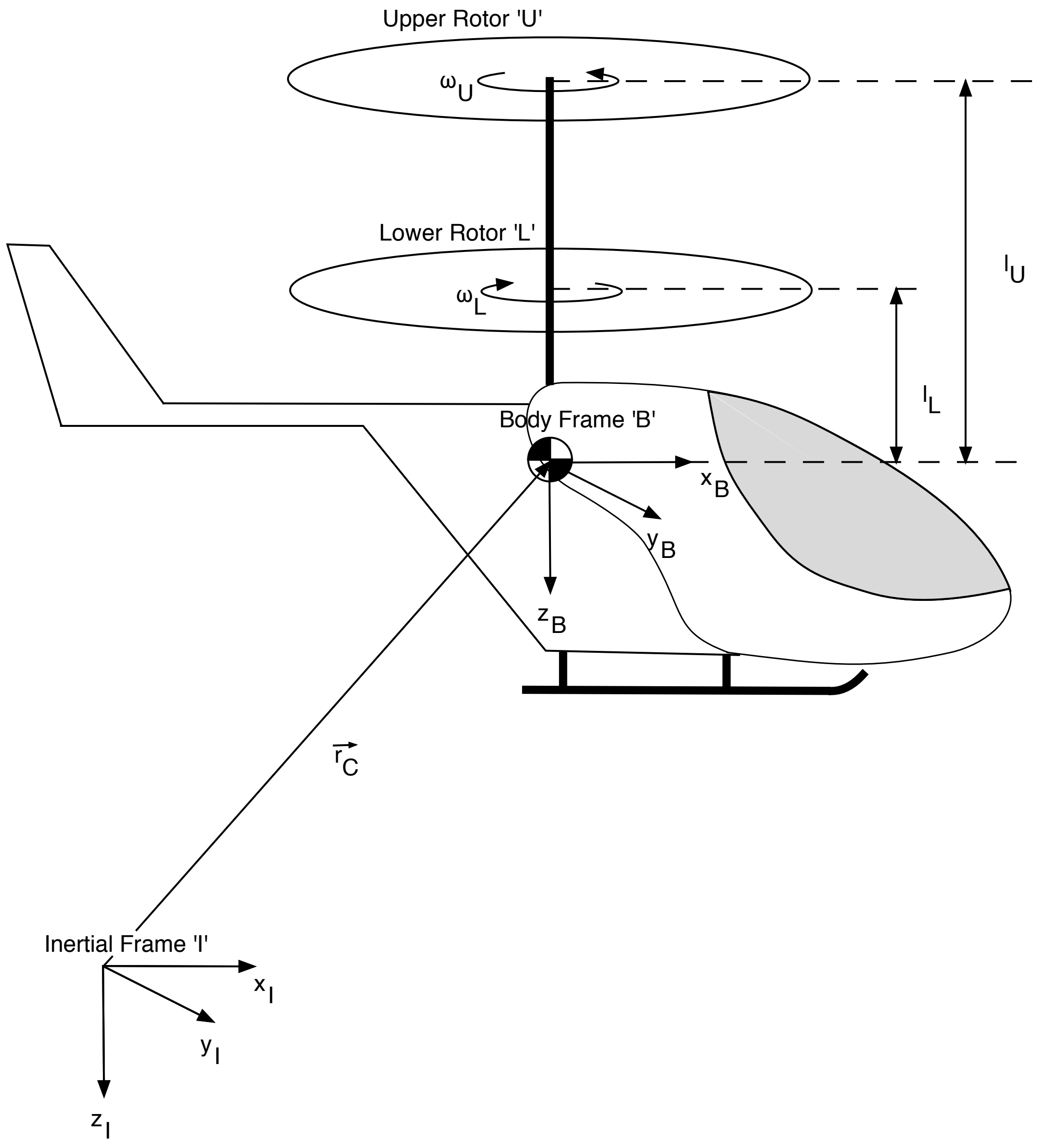


Figure 3.1: the main coordinate frames of the system

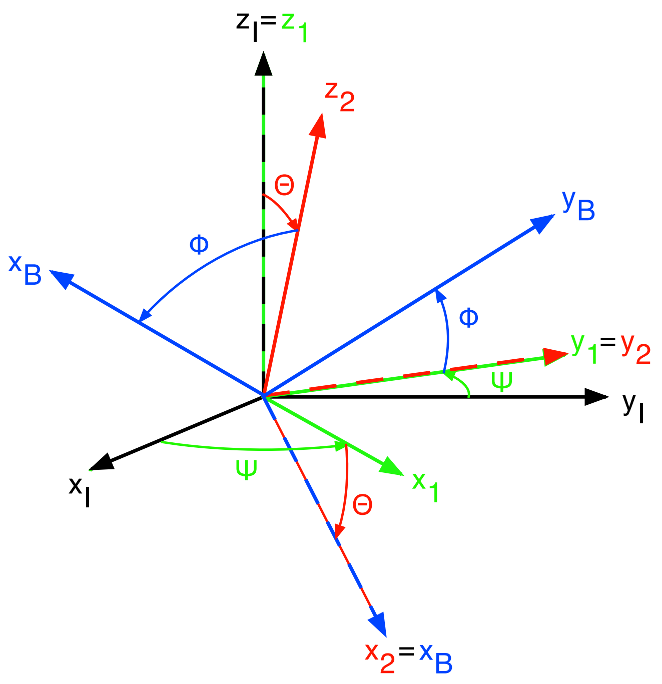


Figure 3.2: elemental rotations about the Euler angles

In addition to the translation $\vec{r}_C$ of frame B, there is also some information about the orientation needed. The orientation is described by the angle vector $\eta = (\Psi, \Theta, \Phi)^T$ (see figure 3.2).

The different angles are named as:

- Ψ: *yaw* angle
- Θ: *pitch* angle
- Φ: *roll* angle

The following notation for coordinate frame transformations is used:

$$A_{IB} \qquad \text{transformation from coordinate frame } B \text{ to } I \tag{3.3}$$

One can calculate the representation of a vector $\vec{c}$ in the coordinate frame I $(= {}_I\vec{c})$ out of its representation ${}_B\vec{c}$ in frame B the following way:

$${}_I\vec{c} = A_{IB} \cdot {}_B\vec{c} \tag{3.4}$$

blablablab

The inverse transformation A_{BI} can be calculated from A_{IB}:

$$A_{BI} = {A_{IB}}^{-1} = {A_{IB}}^{T} \tag{3.5}$$

Connecting different transformations in series can be done as follows:

$$ {}_I\vec{c} = A_{IB} \cdot {}_B\vec{c} = A_{IB} \cdot A_{BC} \cdot {}_C\vec{c} = \dots \quad \Rightarrow \quad A_{IN} = A_{IB} \cdot A_{BC} \cdot \dots \cdot A_{KN} \tag{3.6} $$

The orientation is introduced as TaitBryan angles, a special form of the Euler angles. This convention is very used in aerospace applications. The transformation A_{IB} consists of 3 elemental rotations, based on figure 3.2:

1. Coordinate frame (x_I, y_I, z_I) is rotated around the z_I-axis about the angle Ψ (*yaw*) $\Rightarrow$ (x_1, y_1, z_1):
$$ A_{I1} = \begin{pmatrix} \cos\Psi & -\sin\Psi & 0 \\ \sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.7} $$
2. Coordinate frame (x_1, y_1, z_1) is rotated around the y_1-axis about the angle Θ (*pitch*) $\Rightarrow$ (x_2, y_2, z_2):
$$ A_{12} = \begin{pmatrix} \cos\Theta & 0 & \sin\Theta \\ 0 & 1 & 0 \\ -\sin\Theta & 0 & \cos\Theta \end{pmatrix} \tag{3.8} $$
3. Coordinate frame (x_2, y_2, z_2) is rotated around the x_2-axis about the angle Φ (*roll*) $\Rightarrow$ (x_B, y_B, z_B):
$$ A_{2B} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & -\sin\Phi \\ 0 & \sin\Phi & \cos\Phi \end{pmatrix} \tag{3.9} $$

It is possible to find A_{IB} using the rule for connections in series (3.6) together with equations (3.7), (3.8) and (3.9):

$$ A_{IB} = A_{I1} \cdot A_{12} \cdot A_{2B} = A_{I1} \cdot A_{1B} $$

$$ A_{1B} = A_{12} \cdot A_{2B} = \begin{pmatrix} \cos\Theta & 0 & \sin\Theta \\ 0 & 1 & 0 \\ -\sin\Theta & 0 & \cos\Theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & -\sin\Phi \\ 0 & \sin\Phi & \cos\Phi \end{pmatrix} = $$
$$ \begin{pmatrix} \cos\Theta & \sin\Phi\sin\Theta & \cos\Phi\sin\Theta \\ 0 & \cos\Phi & -\sin\Phi \\ -\sin\Theta & \cos\Theta\sin\Phi & \cos\Phi\cos\Theta \end{pmatrix} $$

$$ A_{IB} = A_{I1} \cdot A_{1B} = \begin{pmatrix} \cos\Psi & -\sin\Psi & 0 \\ \sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\Theta & \sin\Phi\sin\Theta & \cos\Phi\sin\Theta \\ 0 & \cos\Phi & -\sin\Phi \\ -\sin\Theta & \cos\Theta\sin\Phi & \cos\Phi\cos\Theta \end{pmatrix} = $$
$$ A_{IB} = \begin{pmatrix} \cos\Psi\cos\Theta & \cos\Psi\sin\Phi\sin\Theta - \cos\Phi\sin\Psi & \sin\Phi\sin\Psi + \cos\Phi\cos\Psi\sin\Theta \\ \cos\Theta\sin\Psi & \cos\Phi\cos\Psi + \sin\Phi\sin\Psi\sin\Theta & \cos\Phi\sin\Psi\sin\Theta - \cos\Psi\sin\Phi \\ -\sin\Theta & \cos\Theta\sin\Phi & \cos\Phi\cos\Theta \end{pmatrix} \tag{3.10} $$

Chapter 4

The Flapping Dynamics of the Lower Rotor

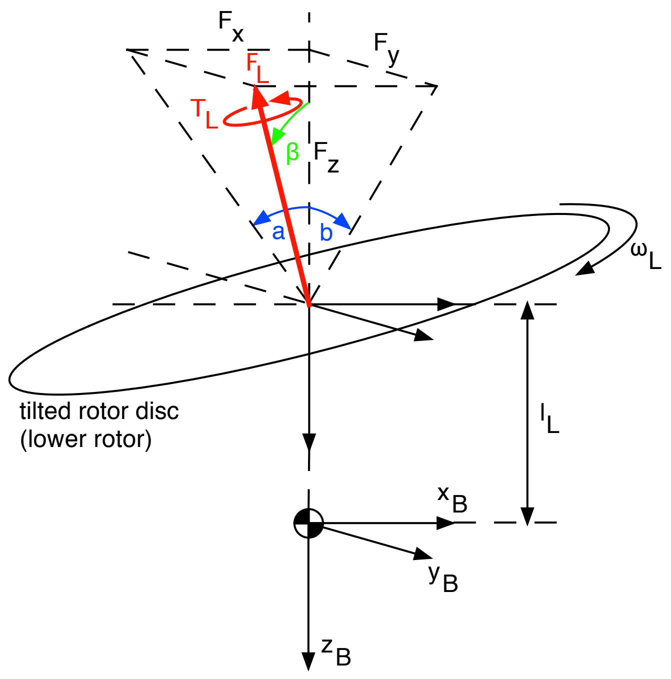


Figure 4.1: flapping dynamics of the lower rotor

The direction of the thrust- and drag-vector of the lower rotor is given by the flapping angle β, which is the angle between the normal vector of the tilted rotor disk and the rotor shaft axis (see figure 4.1). As said above, the longitudinal flapping angle is called a and the lateral one b.

Given a and b, one can calculate the direction of the force- and torque vectors. The derivation of the transformation is done with the example of the lower thrust vector ${}_B\vec{F}_L$. The aim is to find a function with the vector ${}_B\vec{F}_L$, the lower thrust vector expressed in frame B, as output, given a, b and the absolute value of force F_L:

$$
{}_B\vec{F}_L = \begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = f\left(a, b, F_L\right) \tag{4.1}
$$

The first step is to express β with a and b using the Pythagorean theorem and the definition of tangent (see figure 4.1):

$$
\begin{aligned}
\tan^2\beta = \frac{F_x^2 + F_y^2}{F_z^2} &= \\
\frac{F_x^2}{F_z^2} + \frac{F_y^2}{F_z^2} &= \\
\tan^2\beta = \tan^2 a + \tan^2 b
\end{aligned}
$$

Using the relations $\tan\alpha = \frac{\sin\alpha}{\cos\alpha}$ and $\sin^2\alpha + \cos^2\alpha = 1$ leads to:

$$
\begin{aligned}
\frac{\sin^2\beta}{\cos^2\beta} &= \frac{\sin^2 a}{\cos^2 a} + \frac{\sin^2 b}{\cos^2 b} \\
\frac{1-\cos^2\beta}{\cos^2\beta} &= \frac{\sin^2 a}{\cos^2 a} + \frac{\sin^2 b}{\cos^2 b}
\end{aligned}
$$

Now solving this for $\cos\beta$:

$$
\begin{aligned}
1 - \cos^2\beta = \cos^2\beta\left(\frac{\sin^2 a}{\cos^2 a} + \frac{\sin^2 b}{\cos^2 b}\right) \\
1 = \cos^2\beta\left(\frac{\sin^2 a}{\cos^2 a} + \frac{\sin^2 b}{\cos^2 b} + 1\right) &= \\
\cos^2\beta\frac{\sin^2 a\cos^2 b + \cos^2 a\sin^2 b + \cos^2 a\cos^2 b}{\cos^2 a\cos^2 b} &= \\
\cos^2\beta\frac{\sin^2 a\cos^2 b + \cos^2 a(\sin^2 b + \cos^2 b)}{\cos^2 a\cos^2 b} &= \\
\cos^2\beta\frac{\sin^2 a(1-\sin^2 b) + \cos^2 a}{\cos^2 a\cos^2 b} &= \\
\cos^2\beta\frac{\sin^2 a - \sin^2 a\sin^2 b + \cos^2 a}{\cos^2 a\cos^2 b} &= \\
\cos^2\beta\frac{1 - \sin^2 a\sin^2 b}{\cos^2 a\cos^2 b}
\end{aligned}
$$

$$
\Rightarrow \cos\beta = \frac{\cos a\cos b}{\sqrt{1 - \sin^2 a\sin^2 b}} \tag{4.2}
$$

F_z can directly be obtained by the projection of F_L on the x_B-axis together with equation (4.2):

$$
\cos\beta = \frac{-F_z}{F_L}
$$

$$\Rightarrow F_z = -\cos\beta F_L = \frac{-\cos a \cos b}{\sqrt{1-\sin^2 a \sin^2 b}} F_L \tag{4.3}$$

Using the definitions of $\tan a$ and $\tan b$ with the result in equation (4.3) leads to:

$$\tan a = \frac{F_x}{F_z}$$

$$\Rightarrow F_x = F_z \tan a = F_z \frac{\sin a}{\cos a} = \frac{-\sin a \cos b}{\sqrt{1-\sin^2 a \sin^2 b}} F_L \tag{4.4}$$

$$\tan b = \frac{F_y}{-F_z}$$

$$\Rightarrow F_y = -F_z \tan b = F_z \frac{\sin b}{\cos b} = \frac{\cos a \sin b}{\sqrt{1-\sin^2 a \sin^2 b}} F_L \tag{4.5}$$

The transformation of the thrust vector can be written in the following compact form:

$$\boxed{{}_B\vec{F_L} = G(a,b) F_L} \tag{4.6}$$

Were $G(a,b)$ is:

$$G(a,b) = \frac{1}{\sqrt{1-\sin^2 a \sin^2 b}} \begin{pmatrix} -\sin a \cos b \\ \cos a \sin b \\ -\cos a \cos b \end{pmatrix} \tag{4.7}$$

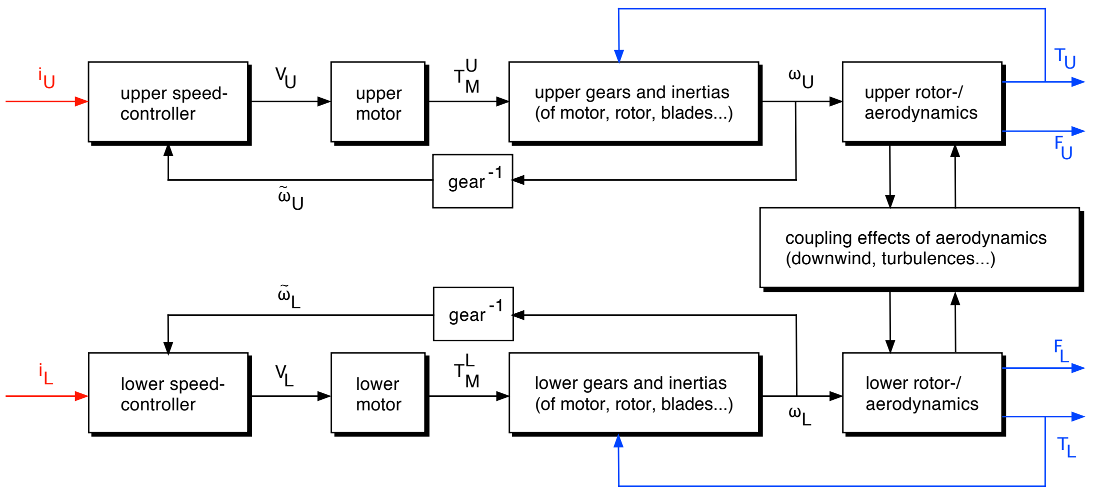
The lower rotor is turning clockwise (figures 3.1 and 4.1), which means, that the drag torque T_L acts counterclockwise and the associated vector ${}_B\vec{T_L}$ has the same direction as ${}_B\vec{F_L}$:

$$\boxed{{}_B\vec{T_L} = G(a,b) \cdot T_L} \tag{4.8}$$

Chapter 5

Dynamics: Drive System and Rotor Blades

Instead of modelling the dynamics of the brushless speedcontrollers together with the brushless motors, the rotor system with gear transmission ratio, blades etc. and the corresponding fluid dynamic effects, a much simpler way is chosen. The reason is the high order model one would get if the drive systems are modelled. A possible way to model it is shown in figure 5.1. There would be at least 9 states, which means that the shown system would finally have around order 9 too.



i_U
upper speed-controller
V_U
upper motor
T_M^U
upper gears and inertias (of motor, rotor, blades...)
ω_U
upper rotor-/ aerodynamics
T_U
F_U
gear $^{-1}$
$\tilde{\omega}_U$
coupling effects of aerodynamics (downwind, turbulences...)
$\tilde{\omega}_L$
gear $^{-1}$
i_L
lower speed-controller
V_L
lower motor
T_M^L
lower gears and inertias (of motor, rotor, blades...)
ω_L
lower rotor-/ aerodynamics
F_L
T_L

Figure 5.1: one possibility to model the drive systems

Figure 5.2 shows another way to derive a model of the drive systems. One assumes a black box model with the corresponding inputs and outputs. The order of the black box model and the different parameters are estimated with the help of some step responses, assuming linear systems.

The main effort is to build up a proper measurement setup (figure 5.3), which allows to analyze the system from its inputs to outputs. Figure 5.3 shows the way it is done with the Pixhawk unit. The helicopter is mounted on a one-axis force and torque sensor unit, which is

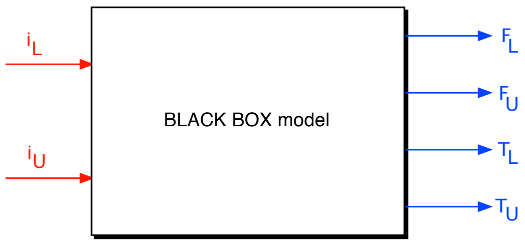


Figure 5.2: blackbox approximation of the drive systems

again mounted on the ground with enough distance, so that the ground effects (turbulences...) are minimized. It is helpful to align the sensor along the rotor axis of the helicopter, because this can minimize swinging effects and therefore noisy disturbances, although the measured vertical results shouldn't change in theory. But the generated torque in the horizontal plane due to misalignment leads to swinging effects.

5.1 force-torque-measurements of the fixed pixhawk

The main idea is to derive a preferably simple model, but enough complex to catch the behaviour around important operating points. In the case of controller task, this would be for example the equilibrium or some trajectory.

Remark 4. *The model will just have positive outputs, i.e. the absolute values of the forces and torques. The signs will be considered in chapter 6.*

Remark 5. *The speed controller input doesn't have any physical unit, but it can be interpreted as the percental fraction of the maximum possible input divided by 100.*

$$[i_L] = [i_U] = \frac{\%\ of\ max.\ controller\ input}{100}$$

5.2 first order low pass element

A first order element has the following transfer function:

$$P(s) = \frac{k}{1 + T \cdot s} \tag{5.1}$$

Where k is the steady state gain and T is the time constant. Figure 5.4 shows the step response of a first order element with unity step as input. k is the steady state factor from input to output. T gives information about how fast the system reacts on a input change. In the case of a step input for example, T defines the slope of the response at the beginning of the step input.

5.3 From controller-inputs to forces

5.3.1 Identification of one single rotor system via step response

Figure 5.5 shows the step response of the lower rotor. The step size of the input is 0.6. Comparing this with figure 5.4, the system can be interpreted as a first order system. The transfer function

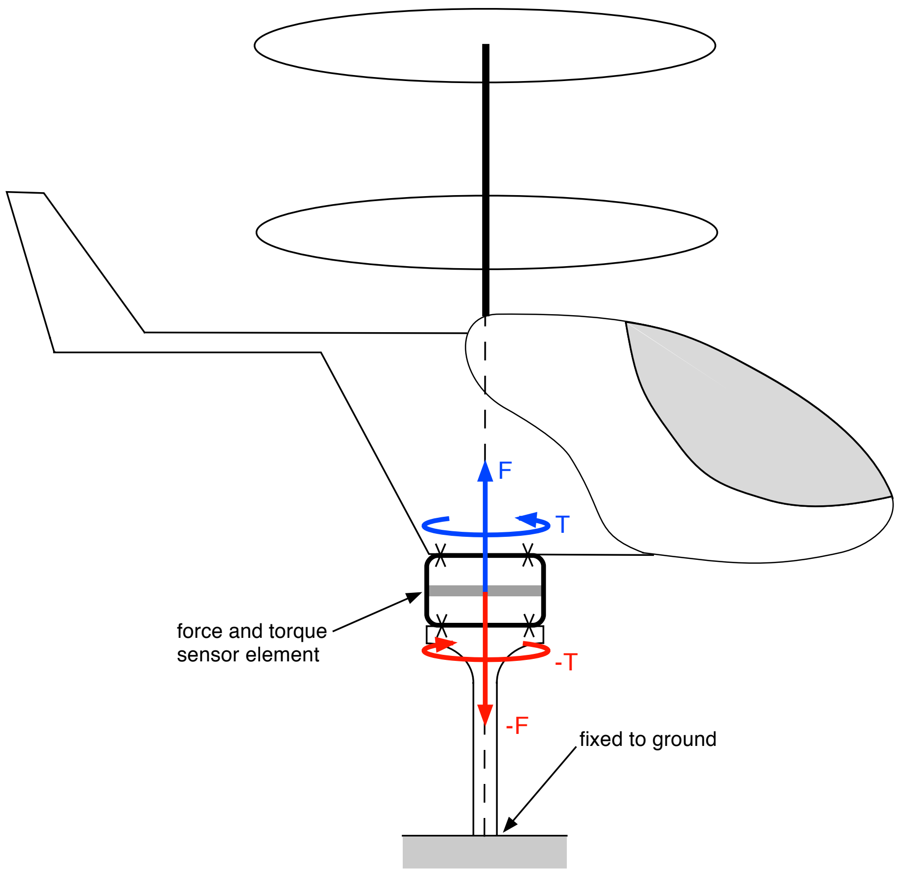


Figure 5.3: measurement setup for the analysis

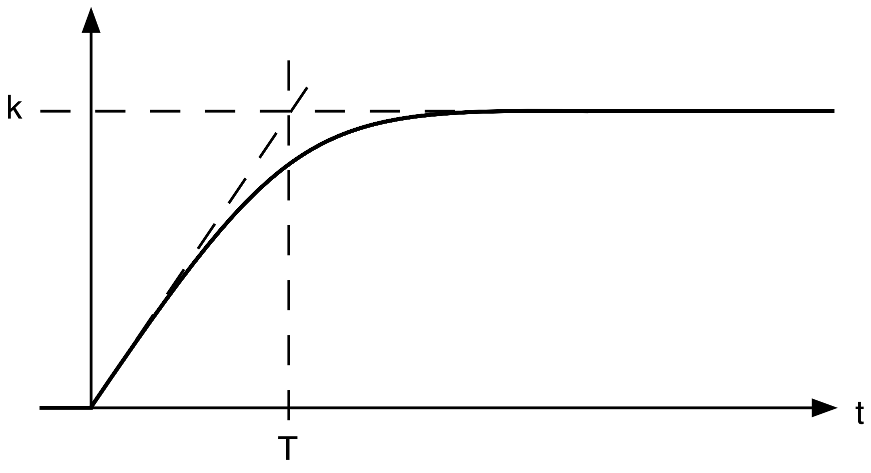


Figure 5.4: step response of first order element with unity input

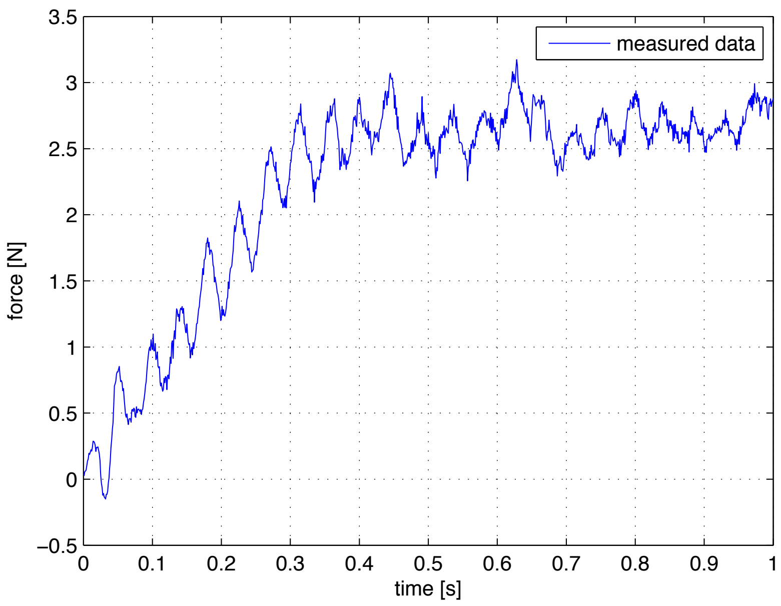


Figure 5.5: step response of the lower rotor with 0.6 as step size of the input

from input to force of the lower drive system is defined as:

$$P_L^F(s) = \frac{k_L^F}{1 + T_L^F \cdot s} \tag{5.2}$$

The index L (respectively U) stands for the lower rotor (respectively upper rotor) and F (respectively T) stands for force (respectively torque) as output. The next step is to find the parameters k_L^F and T_L^F. k_L^F can be calculated in the Laplace domain. Be k_{st} the step size of the input. A step function is also known as Heaviside function, which is defined as:

$$h(t-\delta) = \begin{cases} 0 & \forall t \leq \delta \\ 1 & \forall t > \delta \end{cases} \tag{5.3}$$

The measurement data can be switched along the time axis, such that δ is zero. The input to the test bench is:

$$i_L(t) = k_{st} \cdot h(t)$$

A general transfer function $\Sigma(s)$ in the frequency domain is defined as:

$$\Sigma(s) = \frac{Y(s)}{U(s)} \tag{5.4}$$

Where $U(s)$ is the input and $Y(s)$ the output expressed in the frequency domain. The Laplace transformation of the Heaviside function (5.3) for $\delta = 0$ is:

$$H(s) = \frac{1}{s} \tag{5.5}$$

A step with factor k_{st} can be written, due to linearity of the Laplace transformation, as:

$$\mathcal{L}\{k_{st}h(t)\} = k_{st}\mathcal{L}\{h(t)\} = k_{st}H(s) = \frac{k_{st}}{s} \tag{5.6}$$

The step response Y_{sr} with factor k_{st} in the frequency domain can be derived with equations (5.2), (5.4) and (5.6):

$$Y_{sr}(s) = \Sigma(s)U(s) = P_L^F(s)k_{st}H(s) = \frac{k_L^F}{1 + T_L^F \cdot s}\frac{k_{st}}{s} \tag{5.7}$$

To find the value k_L^F, the *final value theorem* (5.8) is needed:

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0^+} s \cdot F(s) \tag{5.8}$$

Using the theorem (5.8) together with the step response (5.7) leads to the final value y_{fin} the step response will reach in the case of convergence:

$$\begin{aligned} y_{fin} = \lim_{t\to\infty} y_{sr}(t) = \lim_{s\to 0^+} s \cdot Y_{sr}(s) &= \\ \lim_{s\to 0^+} s \cdot \frac{k_L^F}{1 + T_L^F \cdot s}\frac{k_{st}}{s} &= \\ \lim_{s\to 0^+} \frac{k_L^F}{1 + T_L^F \cdot s} k_{st} & \end{aligned}$$

$$\Rightarrow y_{fin} = k_L^F \cdot k_{st} \quad \Leftrightarrow \quad k_L^F = \frac{y_{fin}}{k_{st}} \tag{5.9}$$

Remark 6. *This result can also be seen by looking at figure 5.4, considering that a first order element is linear.*

In practice, the final value of a step response y_{fin} can be found over the mean value of the steady state output $y_{sr}(t >> 0)$. n should be chosen in a way, so that the step response is already steady state at t_n.

$$y_{fin} \approx \frac{\sum_{i=n>>1}^{N} y_{sr}(t_i)}{N - n + 1} \tag{5.10}$$

The factor k_L^F can now be approximated as (equations (5.9) and (5.10)):

$$k_L^F \approx \frac{\sum_{i=n>>1}^{N} y_{sr}(t_i)}{(N - n + 1) \cdot k_{st}} \tag{5.11}$$

In the case of the Pixhawk helicopter, the resulting value for k_L^F is:

$$k_L^F \approx 4,4N \tag{5.12}$$

There are many possibilities to find a value for T_L^F. One way is to define first a cost function $V(T_L^F)$:

$$V(T_L^F) = \sum_{i=1}^{N} \left(y_{sr}(t_i) - \hat{y}_{sr}(t_i)\right)^2 \tag{5.13}$$

Where $y_{sr}(t_i)$ is a measurement point and $\hat{y}_{sr}(t_i)$ is the corresponding estimated output based on the first order approach (5.2). The main idea will be to look for a T_L^F, which minimizes (locally) this cost function.

$$T_{L\ opt}^F = \arg\min_{T_L^F} V(T_L^F) \tag{5.14}$$

To find a specific $\hat{y}_{sr}(t_i)$, the step response (5.7) has to be transformed from the frequency domain back to the time domain. This will be done in two steps:

1. Partial fraction decomposition of (5.7), so that the linearity of the Laplace transformation (5.15) can be used.

$$\mathcal{L}\{a_1 f_1(t) + a_2 f_2(t)\} = a_1\mathcal{L}\{f_1(t)\} + a_2\mathcal{L}\{f_2(t)\} = a_1 F_1(s) + a_2 F_2(s) \tag{5.15}$$

2. The function in the time domain can now be derived by exchanging the single fractions of the partial fraction decomposition with the corresponding terms in time domain, which are standard elements and can be looked up in tables.

Equation (5.7) as partial fraction decomposition:

$$Y_{sr}(s) = \frac{k_L^F \cdot k_{st}}{\left(1 + T_L^F \cdot s\right) s} = \frac{A}{s} + \frac{B}{\left(1 + T_L^F \cdot s\right)}$$

Multiplying both sides with $\left(1 + T_L^F \cdot s\right) s$:

$$k_L^F \cdot k_{st} = A\left(1 + T_L^F \cdot s\right) + B \cdot s \tag{5.16}$$

This equation should hold for every s. Replacing s in (5.16) with $s = 0$ leads to the factor A:

$$A = k_L^F \cdot k_{st}$$

And replacing s in (5.16) with $s = -\frac{1}{T_L^F}$ leads to the factor B:

$$k_L^F \cdot k_{st} = -B\frac{1}{T_L^F}$$
$$\Rightarrow B = -k_L^F \cdot k_{st} \cdot T_L^F$$

Therefore the partial fraction decomposition of equation (5.7) is:

$$Y_{sr}(s) = k_L^F \cdot k_{st}\left(\frac{1}{s} - \frac{1}{s + \frac{1}{T_L^F}}\right) \tag{5.17}$$

For the inverse Laplace transformation of (5.17), the following two basic inverses are needed:

$$\mathcal{L}^{-1}\left\{\frac{1}{s}\right\} = h(t) \qquad \mathcal{L}^{-1}\left\{\frac{1}{s+a}\right\} = e^{-at} \tag{5.18}$$

The first one is the Heaviside function as already seen above. With equations (5.15), (5.17) and (5.18), the step response (5.7) can be written in the time domain as:

$$y_{sr}(t) = k_L^F \cdot k_{st} \left(1 - e^{-\frac{1}{T_L^F} t}\right), \qquad t \geq 0 \tag{5.19}$$

The problem (5.14) is a least square problem and can now be solved as explained in chapter 7:

$$T_L^F \approx 0.17s \tag{5.20}$$

In summary, the resulting transfer function from the lower input i_L to the absolute value of the lower force F_L is:

$$\boxed{\begin{array}{c} P_L^F(s) = \dfrac{k_L^F}{1 + T_L^F \cdot s} \\ k_L^F \approx 4,4N \qquad T_L^F \approx 0.17s \end{array}} \tag{5.21}$$

Figures 5.6 and 5.7 show the step responses of this first order approximation and the corresponding measured data on the real plant.

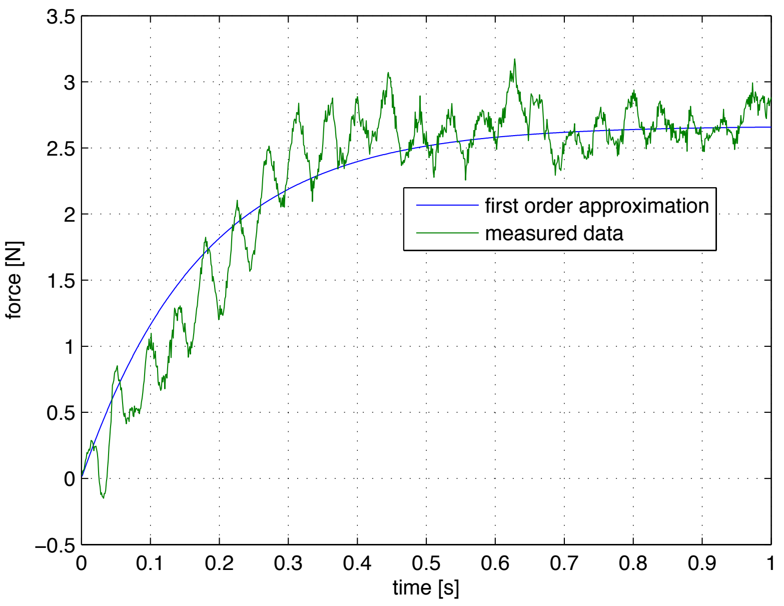


Figure 5.6: step response of the lower rotor with 0.6 as step size of the input

Figure 5.8 shows the step response of the upper rotor system compared with the same first order approximation as used for the lower rotor (equation (5.21)). The transfer function P_U^F has i_U as input and F_U as output.

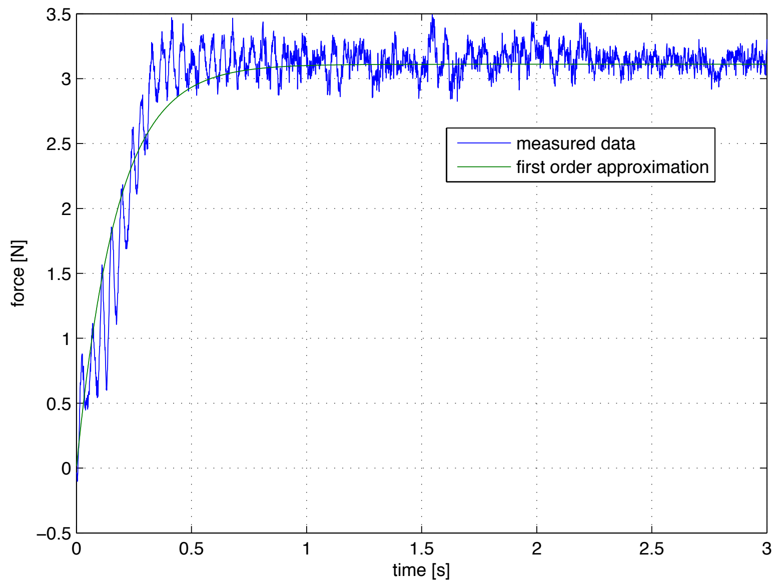


Figure 5.7: step response of the lower rotor with 0.8 as step size of the input

$$\boxed{\begin{array}{c} P_U^F(s) = \dfrac{k_U^F}{1 + T_U^F \cdot s} \\ k_U^F \approx 4,4N \qquad T_U^F \approx 0.17s \end{array}} \tag{5.22}$$

5.3.2 Coupling effects of both rotors running simultaniously

Figure 5.9 shows the step response of both driving systems together, once for the real plant and once if both transfer functions $P_L^F(s)$ and $P_U^F(s)$ are just added. When both rotors run simultaneously, some cross coupling effects should therefore be considered. To keep a possibly simple system, the following assumptions were made:

1. The lower rotor is just influenced by the downwind of the upper rotor.
2. The upper rotor is not influenced by the lower one.
3. In the case of both rotors running, the transfer function $P_L^F(s)$ of the lower rotor is multiplied with a wind constant k_w.

Remark 7. *The third assumption holds just in the case, where both rotors rotate with more or less the same rate, which holds for linear controllers around an equilibrium. Figure 5.9 besides shows, that the cross coupling influence is not enough big to demand higher order or non-linear descriptions.*

k_w can be found the same way as k_L^F was found in chapter 5.3.1. The total force output in the frequency domain is given as:

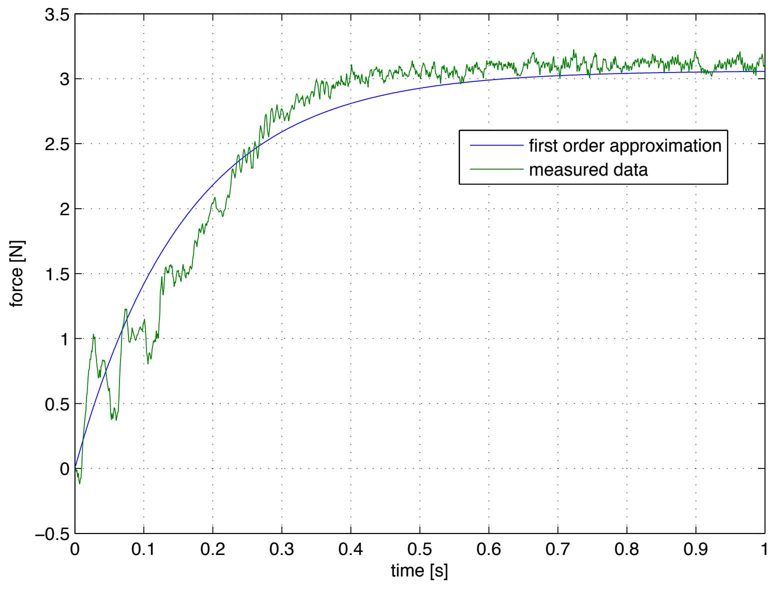


Figure 5.8: step response of the upper rotor with 0.8 as step size of the input

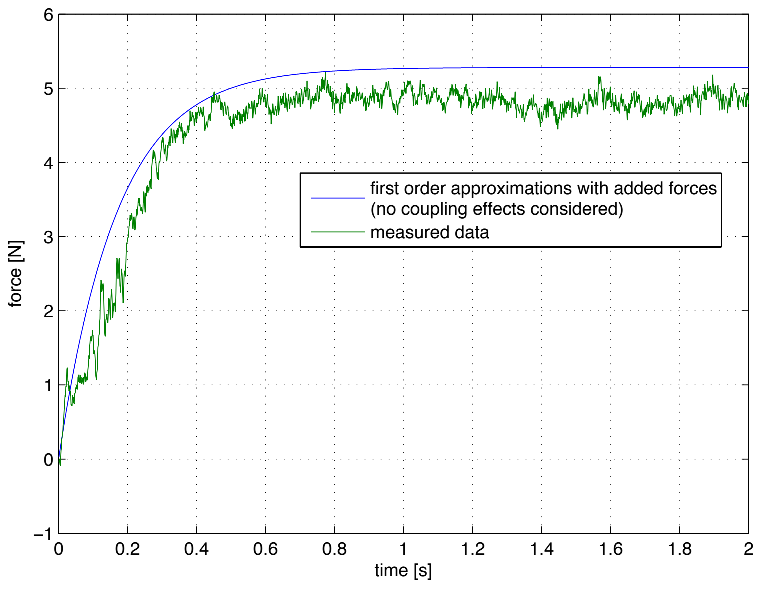


Figure 5.9: step response of both rotors with 0.6 as step size of both inputs

$$\boxed{Y_{tot}(s) = k_w \cdot P_L^F(s) \cdot U_L(s) + P_U^F(s) \cdot U_U(s)} \tag{5.23}$$

$U_L(s)$ is $i_L(t)$, the input of the lower speedcontroller, expressed in the frequency domain and $U_L(s)$ is $i_L(t)$ expressed in the frequency domain. In the case where both inputs are steps with equal step size k_{st}, the inputs can be written as:

$$U_L(s) = U_U(s) = k_{st} \cdot H(s) = \frac{k_{st}}{s} \tag{5.24}$$

Equation (5.23) together with equation (5.24) leads to the step response in the frequency domain:

$$Y_{sr}(s) = k_w \cdot P_L^F(s) \cdot \frac{k_{st}}{s} + P_U^F(s) \cdot \frac{k_{st}}{s} \tag{5.25}$$

With the same steps as used for the derivation of equation (5.9), one can get the final value y_{fin} the step response will get for $t \rightarrow \infty$:

$$y_{fin} = k_w \cdot k_L^F \cdot k_{st} + k_U^F \cdot k_{st} \tag{5.26}$$

This can be rearranged to find value for k_w:

$$k_w = \frac{y_{fin} - k_U^F \cdot k_{st}}{k_L^F \cdot k_{st}} \tag{5.27}$$

Together with practical approximation (5.10) for y_{fin}, k_w can be calculated as:

$$k_w \approx \frac{\sum\limits_{i=n>>1}^{N} y_{sr}(t_i) - k_U^F \cdot k_{st} \cdot (N-n+1)}{k_L^F \cdot k_{st} \cdot (N-n+1)} \tag{5.28}$$

The resulting k_w is:

$$\boxed{k_w \approx 0.84} \tag{5.29}$$

Figure 5.10 shows the same measurement data as shown in figure 5.9, but this time compared with the approximation including the factor k_w. Figure 5.11 shows the resulting force when both rotor systems have a ramp as input.

5.4 from controller-inputs to torques

5.4.1 Identification of one single rotor system via step response

Figure 5.12 shows the step response of the upper rotor system when the torque is measured. It does not look like a first order system, it looks more like a higher order or non-linear system. It is assumed, that this effects just appear when starting at zero rotation speed and for high input changes. One reason could be the drag-hinge of the rotor blade. Nevertheless, the approximation is also done by a first order element. It is furthermore assumed, that the time constant T_U^T (and T_L^T) is the same as the one for the transfer functions of the forces:

$$T_L^T = T_U^T = T_L^F = T_U^F = 0.17s \tag{5.30}$$

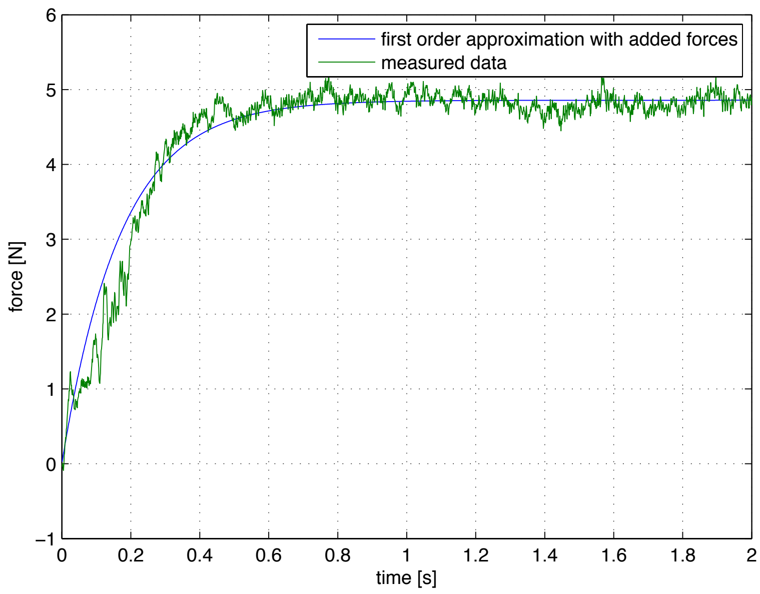


Figure 5.10: step response of both rotors with 0.6 as step size of both inputs

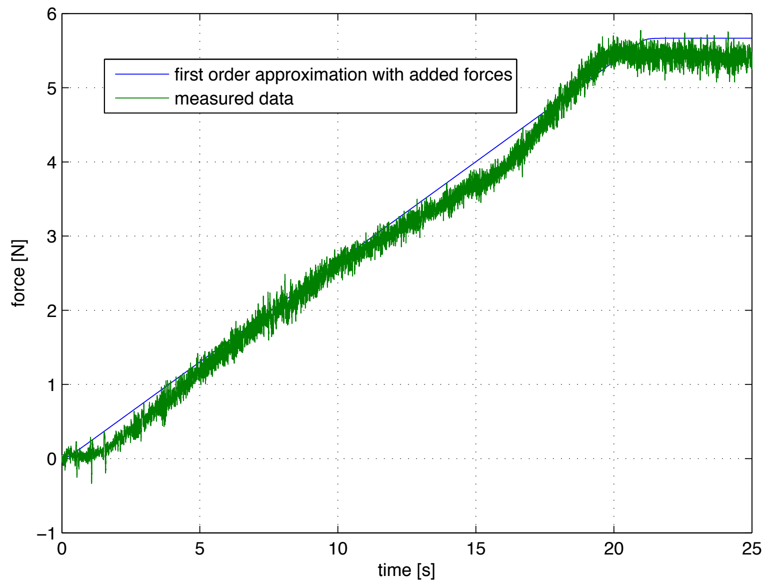


Figure 5.11: ramp response with the same input on both drive systems

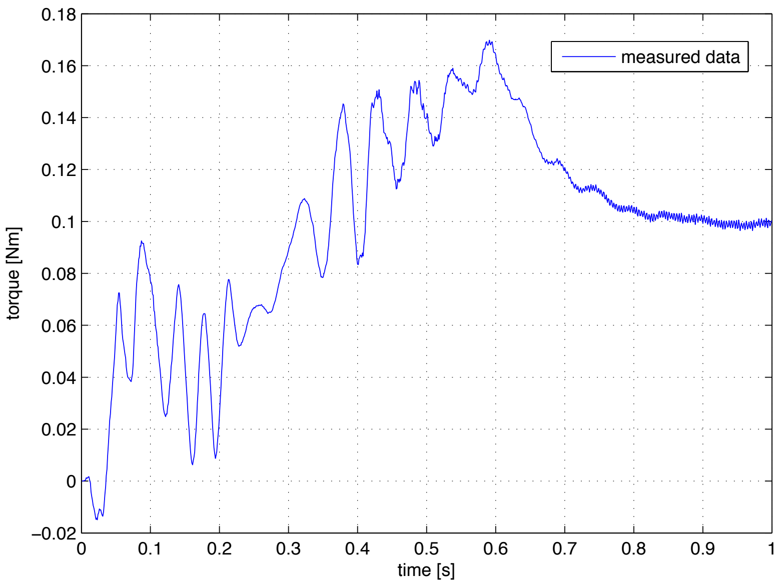


Figure 5.12: step response of the upper rotor with 0.8 as step size of the input

This has to be, because it takes the same time to get in the steady state behaviour. The steady state gain k_U^T (or k_L^T) is derived the same way as it was done in chapter 5.3.1 for the factor k_U^T (or k_L^T).That leads to the following transfer functions:

$$\boxed{\begin{array}{ll} P_L^T(s) = \dfrac{k_L^T}{1 + T_L^T \cdot s} & P_U^T(s) = \dfrac{k_U^T}{1 + T_U^T \cdot s} \\ k_L^T = k_U^T \approx 0.135Nm & T_L^T = T_U^T \approx 0.17s \end{array}} \tag{5.31}$$

Where the input is i_L respectively i_U and the output is T_L respectively T_U. Figure 5.13 shows the compared step responses of the approximation and the measured data of the upper rotor. Figure 5.14 shows the ramp responses of the approximation and real plant. One can see, that the higher order or non-linear effects appear there at the beginning too (high swinging effects), but the behaviour is afterwards very similar for both curves, what certifies the assumptions from above.

Remark 8. *It would of course be possible to derive a higher order approximation for the transfer function P_L^T (and P_U^T), but first controller tests showed, that the assumed model allows suitable results.*

5.4.2 Coupling effects of both rotors running simultaniously

Figure 5.15 shows the ramp responses of force and torque, the same input ramp was given to both systems. One can see in the upper plot, that the cross-coupling effects hardly influence the transfer functions of the torques. The sum of both torques - the signs are exceptionally considered - remains more or less zero. Therefore, no cross-coupling effects are assumed in the case of the torques.

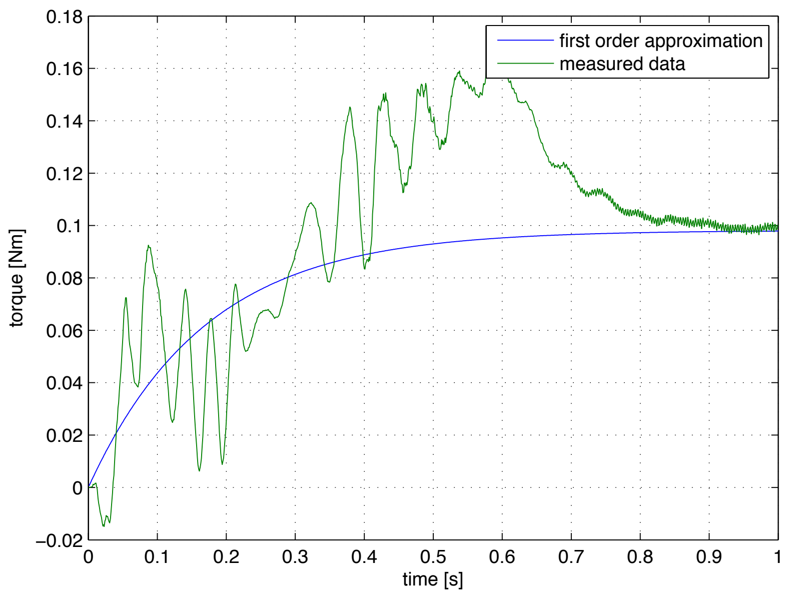


Figure 5.13: step response of the upper rotor with 0.8 as step size of the input

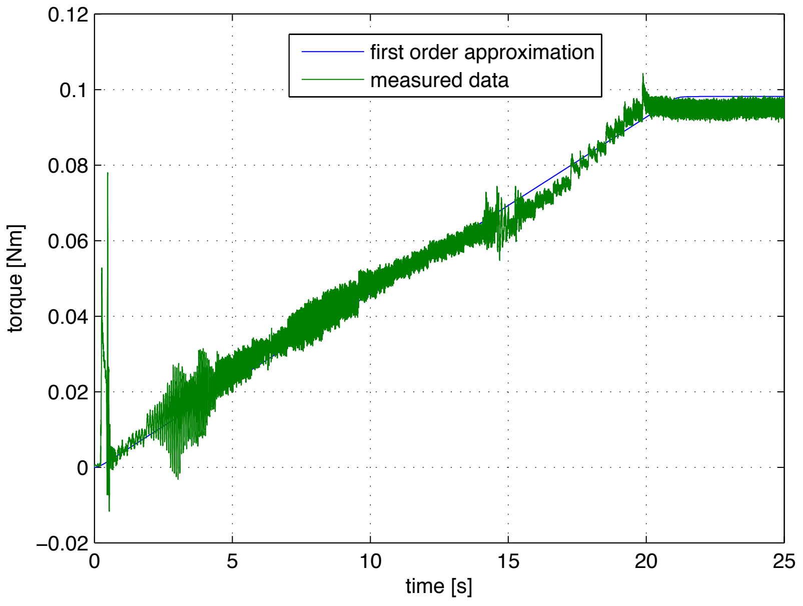


Figure 5.14: ramp response of the upper rotor system

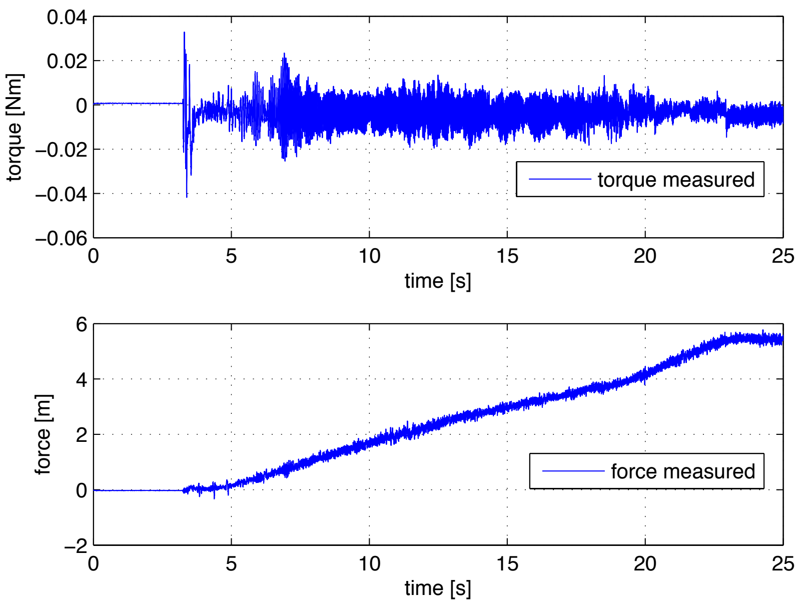


Figure 5.15: ramp response of both drive systems together

Chapter 6

newton-dynamics

Figure 6.1 shows again the different subsystems. The dynamics of the drive/rotor systems are already derived. In this chapter, the focus is set on the remainig two subsystems, the dynamics of the position and of the Euler - angles. This will be done over the Newton dynamics. The helicopter is therefore assumed to be a rigid body.

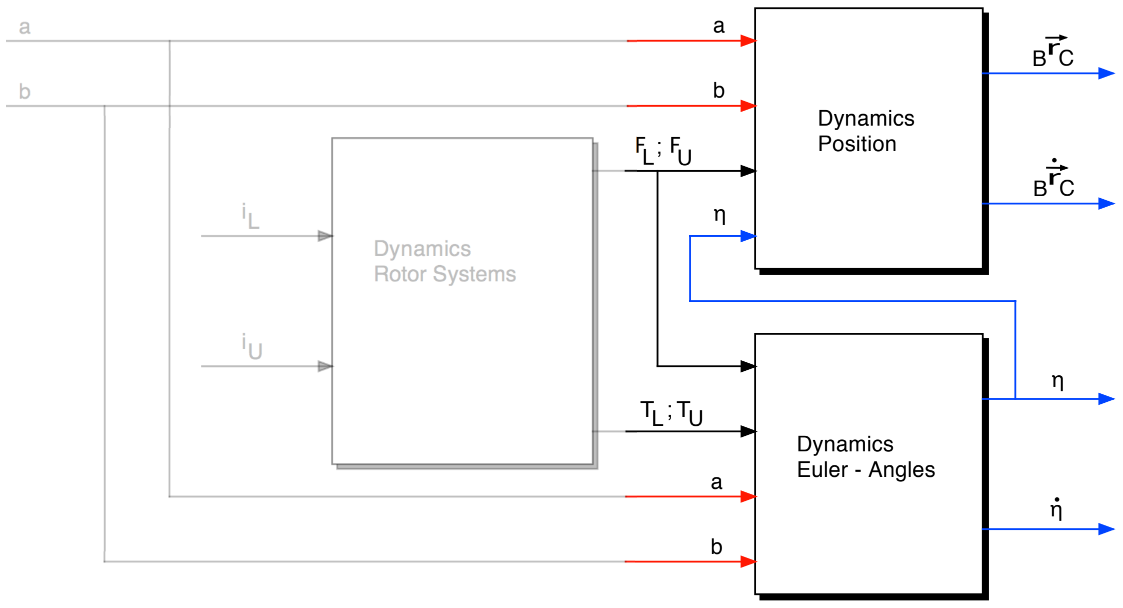


Figure 6.1: remaining subsystems

The equations for the *law of conservation of momentum* and *angular momentum* for a rigid body referred at its center of gravity are:

$$\boxed{\begin{aligned} m \cdot {}_I\vec{a}_C = m \cdot {}_I\ddot{\vec{r}}_C = {}_I\vec{F}^a \\ {}_B\bar{\Theta}_C \cdot {}_B\dot{\vec{\Omega}} = {}_B\vec{T}^a_C - {}_B\vec{\Omega} \times {}_B\bar{\Theta}_C \cdot {}_B\vec{\Omega} \end{aligned}} \qquad \begin{aligned} (6.1) \\ (6.2) \end{aligned}$$

m	the overall mass of the helicopter $[kg]$

${}_I\vec{a}_C$	the acceleration vector expressed in the body fixed coordinates $[m/s^2]$
${}_I\ddot{\vec{r}}_C$	the second time derivative of ${}_I\vec{r}_C$, $({}_I\ddot{\vec{r}}_C = {}_I\vec{a}_C)$
${}_I\vec{r}_C$	the coordinates of the center of gravity of the rigid body $[m]$
${}_I\vec{F}^a$	the sum of all force vectors acting on the rigid body in frame I $[N]$
A_{IB}	transformation from the body frame B to inertial frame I (chapter 3)
${}_B\bar{\Theta}_C$	the inertia tensor at the center of gravity expressed in the body frame B, $[kg \cdot m^2]$
${}_B\vec{T}^a_C$	the overall torque vector acting on rigid body reduced at its center of gracity $[Nm]$
${}_B\vec{\Omega}$	angular velocity vector of the fixed body $[rad/s]$
${}_B\dot{\vec{\Omega}}$	angular acceleration vector of the fixed body, the time derivative of ${}_B\vec{\Omega}$ $[rad/s^2]$

Remark 9. *The angular velocity vector ${}_B\vec{\Omega}$ is not equal to the time derivative of the Euler angles vector η:*

$${}_B\vec{\Omega} \neq \dot{\eta}$$

6.1 law of conservation of momentum

The equation (6.1) can be expressed in the body frame B:

$$m \cdot {}_I\ddot{\vec{r}}_C = m \cdot A_{IB} \cdot {}_B\ddot{\vec{r}}_C = {}_I\vec{F}^a$$
$${}_B\ddot{\vec{r}}_C = \frac{1}{m} A_{IB}^{-1} \cdot {}_I\vec{F}^a = \frac{1}{m} A_{IB}^T \cdot {}_I\vec{F}^a = \frac{1}{m} A_{BI} \cdot {}_I\vec{F}^a = \frac{1}{m} {}_B\vec{F}^a \tag{6.3}$$

The overall force ${}_B\vec{F}^a$ contains the two thrust vectors of the lower and upper rotors ${}_B\vec{F}_L$ and ${}_B\vec{F}_U$, besides the gravity vector ${}_B\vec{G}$ expressed in frame B. ${}_B\vec{F}_L$ can be calculated as shown in equation (4.6):

$${}_B\vec{F}_L = G(a,b) \cdot F_L \tag{6.4}$$

${}_B\vec{F}_U$ is aligned along the z_B-axis with negative sign:

$${}_B\vec{F}_U = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} F_U \tag{6.5}$$

${}_B\vec{G}$ can be derived via ${}_I\vec{G}$:

$${}_B\vec{G} = A_{BI} \cdot {}_I\vec{G} = m \cdot g \cdot A_{BI} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{6.6}$$

Equations (6.4), (6.5) und (6.6) lead to ${}_B\vec{F}^a$:

$${}_B\vec{F}^a = G(a,b) \cdot F_L + \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} F_U + m \cdot g \cdot A_{BI} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{6.7}$$

Equation (6.7) in (6.1) leads to the dynamics of the center of gravity:

$$\boxed{{}_B\ddot{\vec{r}}_C = \frac{1}{m}{}_B\vec{F}^a = \frac{1}{m}\left(G(a,b)\cdot F_L + \begin{pmatrix}0\\0\\-1\end{pmatrix}F_U\right) + g\cdot A_{BI}\begin{pmatrix}0\\0\\1\end{pmatrix}} \tag{6.8}$$

6.2 law of conservation of angular momentum

Equation (6.2) needs a reformulation of the angular velocity vector ${}_B\vec{\Omega}$ depending on the Euler angles η. ${}_B\vec{\Omega}$ describes the rotation of the rigid body around its center of gravity expressed in coordinate frame B. ${}_I\vec{\Omega}$ is the same vector expressed in frame I. It is equivalent to ${}_I\vec{\omega}_{IB}$, the angular velocity vector of frame B concerning frame I:

$${}_I\vec{\Omega} = {}_I\vec{\omega}_{IB}$$

First, the definition of the matrix $\tilde{a}$ of vector $\vec{a}$:

$$\vec{a} = \begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix} \to \tilde{a} := \begin{pmatrix}0 & -a_3 & a_2\\ a_3 & 0 & -a_1\\ -a_2 & a_1 & 0\end{pmatrix} \tag{6.9}$$

Remark 10. *With this definition, one can calculate the matrix cross product:*

$$\vec{a}\times\vec{b} = \tilde{a}\cdot\vec{b} \tag{6.10}$$

${}_K\vec{\omega}_{KD}$ of any to coordinate frames can be calculated via the transformation matrix A_{KD}:

$${}_K\tilde{\omega}_{KD} = \dot{A}_{KD}\cdot A_{KD}^T \quad\Rightarrow\quad {}_K\vec{\omega}_{KD} = \begin{pmatrix}\left|\dot{A}_{KD}\cdot A_{KD}^T\right|_{(3,2)}\\ \left|\dot{A}_{KD}\cdot A_{KD}^T\right|_{(1,3)}\\ \left|\dot{A}_{KD}\cdot A_{KD}^T\right|_{(2,1)}\end{pmatrix} \tag{6.11}$$

${}_I\vec{\Omega}$ can now be calculated with the coordinate transformation (3.10) based on the Euler angle vector $\eta = (\Psi,\Theta,\Phi)^T$

$${}_I\vec{\Omega} = \begin{pmatrix}\left|\dot{A}_{IB}\cdot A_{IB}^T\right|_{(3,2)}\\ \left|\dot{A}_{IB}\cdot A_{IB}^T\right|_{(1,3)}\\ \left|\dot{A}_{IB}\cdot A_{IB}^T\right|_{(2,1)}\end{pmatrix} = \begin{pmatrix}\cos\Psi\cos\Theta\dot{\Phi} - \sin\Psi\dot{\Theta}\\ \cos\Theta\sin\Psi\dot{\Phi} + \cos\Psi\dot{\Theta}\\ \dot{\Psi} - \sin\Theta\dot{\Phi}\end{pmatrix} \tag{6.12}$$

${}_B\vec{\Omega}$ is derived with the transformations matrix $A_{BI} = A_{IB}^T$:

$$\boxed{{}_B\vec{\Omega} = A_{IB}^T\cdot{}_I\vec{\Omega} = \begin{pmatrix}\dot{\Phi} - \sin\Theta\dot{\Psi}\\ \cos\Theta\sin\Phi\dot{\Psi} + \cos\Phi\dot{\Theta}\\ \cos\Phi\cos\Theta\dot{\Psi} - \sin\Phi\dot{\Theta}\end{pmatrix}} \tag{6.13}$$

${}_B\dot{\vec{\Omega}}$ is the time derivative of ${}_B\vec{\Omega}$ (6.13):

$$ {}_B\dot{\vec{\Omega}} = \begin{pmatrix} \ddot{\Phi} - \cos\Theta\dot{\Theta}\dot{\Psi} - \sin\Theta\ddot{\Psi} \\ \cos\Theta\sin\Phi\ddot{\Psi} + (-\sin\Theta\sin\Phi\dot{\Theta} + \cos\Theta\cos\Phi\dot{\Phi})\dot{\Psi} + \cos\Phi\ddot{\Theta} - \sin\Phi\dot{\Theta}\dot{\Phi} \\ \cos\Phi\cos\Theta\ddot{\Psi} + (-\sin\Phi\cos\Theta\dot{\Phi} - \cos\Phi\sin\Theta\dot{\Theta})\dot{\Psi} - \sin\Phi\ddot{\Theta} - \cos\Phi\dot{\Theta}\dot{\Phi} \end{pmatrix} \tag{6.14} $$

This can be rewritten as:

$$ {}_B\dot{\vec{\Omega}} = \bar{M}^A \cdot \ddot{\vec{\eta}} + \Delta\vec{M}^A \tag{6.15} $$

Where:

$$ \boxed{ \begin{aligned} \ddot{\vec{\eta}} &= \begin{pmatrix} \ddot{\Psi} \\ \ddot{\Theta} \\ \ddot{\Phi} \end{pmatrix} && (6.16) \\ \bar{M}^A &= \begin{pmatrix} -\sin\Theta & 0 & 1 \\ \cos\Theta\sin\Phi & \cos\Phi & 0 \\ \cos\Phi\cos\Theta & -\sin\Phi & 0 \end{pmatrix} && (6.17) \\ \Delta\vec{M}^A &= \begin{pmatrix} -\cos\Theta\dot{\Theta}\dot{\Psi} \\ (-\sin\Theta\sin\Phi\dot{\Theta} + \cos\Theta\cos\Phi\dot{\Phi})\dot{\Psi} - \sin\Phi\dot{\Theta}\dot{\Phi} \\ (-\sin\Phi\cos\Theta\dot{\Phi} - \cos\Phi\sin\Theta\dot{\Theta})\dot{\Psi} - \cos\Phi\dot{\Theta}\dot{\Phi} \end{pmatrix} && (6.18) \end{aligned} } $$

The next step is to to find ${}_B\vec{T}_C^a$, the overall torque reduced at the center of gravity of the rigid body. The gravity vector (6.6) is acting at the center of gravity and does therefore not end in a torque. The forces ${}_B\vec{F}_L$ and ${}_B\vec{F}_U$ lead to the following torques based on the geometry given in figure (3.1). ${}_B\vec{F}_L$ and ${}_B\vec{F}_U$ are given in equation (6.4) respectively (6.5). The torque fraction of the force vector ${}_B\vec{F}_L$ is:

$$ \begin{pmatrix} 0 \\ 0 \\ -l_L \end{pmatrix} \times {}_B\vec{F}_L = \begin{pmatrix} 0 \\ 0 \\ -l_L \end{pmatrix} \times G(a,b) \cdot F_L \tag{6.19} $$

The torque fraction of the force vector ${}_B\vec{F}_U$ is:

$$ \begin{pmatrix} 0 \\ 0 \\ -l_U \end{pmatrix} \times {}_B\vec{F}_U = \begin{pmatrix} 0 \\ 0 \\ -l_U \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} F_U = \vec{0} \tag{6.20} $$

That makes sense, because the force vector of the upper rotor is always aligned along the z_B-axis through the center of gravity of the helicopter. The drag torque ${}_B\vec{T}_L$ of the lower rotor is (as given in equation (4.8)):

$$ {}_B\vec{T}_L = G(a,b) \cdot T_L \tag{6.21} $$

The drag torque ${}_B\vec{T}_U$ of the upper rotor is:

$$ {}_B\vec{T}_U = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} T_U \tag{6.22} $$

${}_B\vec{T}_C^a$ is the sum of equations (6.19), (6.21) and (6.22):

$$\boxed{{}_B\vec{T}_C^a = \begin{pmatrix} 0 \\ 0 \\ -l_L \end{pmatrix} \times G(a,b) \cdot F_L + G(a,b) \cdot T_L + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} T_U} \tag{6.23}$$

Equation (6.2) can be rewritten with equation (6.15):

$$\begin{aligned}
{}_B\bar{\Theta}_C \cdot {}_B\dot{\vec{\Omega}} &= {}_B\vec{T}_C^a - {}_B\vec{\Omega} \times {}_B\bar{\Theta}_C \cdot {}_B\vec{\Omega} \\
{}_B\bar{\Theta}_C \cdot \left(\bar{M}^A \cdot \ddot{\eta} + \Delta\vec{M}^A\right) &= {}_B\vec{T}_C^a - {}_B\vec{\Omega} \times {}_B\bar{\Theta}_C \cdot {}_B\vec{\Omega} \\
{}_B\bar{\Theta}_C \cdot \bar{M}^A \cdot \ddot{\eta} + {}_B\bar{\Theta}_C \cdot \Delta\vec{M}^A &= {}_B\vec{T}_C^a - {}_B\vec{\Omega} \times {}_B\bar{\Theta}_C \cdot {}_B\vec{\Omega} \\
{}_B\bar{\Theta}_C \cdot \bar{M}^A \cdot \ddot{\eta} &= {}_B\vec{T}_C^a - {}_B\vec{\Omega} \times {}_B\bar{\Theta}_C \cdot {}_B\vec{\Omega} - {}_B\bar{\Theta}_C \cdot \Delta\vec{M}^A
\end{aligned}$$

$$\boxed{\ddot{\eta} = \left({}_B\bar{\Theta}_C \cdot \bar{M}^A\right)^{-1} \left({}_B\vec{T}_C^a - {}_B\vec{\Omega} \times {}_B\bar{\Theta}_C \cdot {}_B\vec{\Omega} - {}_B\bar{\Theta}_C \cdot \Delta\vec{M}^A\right)} \tag{6.24}$$

Equation (6.24) describes together with equations (6.13), (6.16), (6.17), (6.18) and (6.23) the dynamics of the Euler angles $\eta = (\Psi, \Theta, \Phi)^T$.

Part II

sensor fusion

Chapter 7

least square algorythms

As the conditions of the pressure sensor change, the range where the sensor works changes too. E.g. if the weather conditions or the height above sea level changes. The distance sensor (ultrasonic) has always the same range once it has been calibrated. Therefore it would be usefull to have a recursive function, which does calibrate the pressure sensor based on the distance sensor during the first few moments of the flight online. Of course, this algorythm will work for every similar problem.

7.1 offline LS-filter

The first step is to introduce the main idea of the least square with the offline version. The meaning of this kind of least square filter is to collect first the whole measurement data and to calculate afterwards the paramaters.

First some general things about the nomenclature. The measurement data will be calibrated in a reasonable approximation:

$$\hat{y} = \phi^T \theta$$

Where:

- $\hat{y}$ is the *estimated value*
- ϕ is the *regression vector* $\in \mathbb{R}^{1 \times p}$
- θ is *parameter vector* $\in \mathbb{R}^{p \times 1}$

Examples for the regression vector:

- $\phi = [1 \quad x \quad x^2 \quad ... \quad x^p]^T$
- $\phi = [1 \quad e^{-x} \quad e^{-2x} \quad ... \quad e^{-px}]^T$
- $\phi = [1 \quad sin(x) \quad sin(x)^2 \quad ... \quad sin(x)^p]^T$
- etc.

In our case, we try to approximate the pressure sensor as a linear function $\hat{y} = ax + b$. It follows that:

$$\phi(i) = \begin{pmatrix} x(i) \\ 1 \end{pmatrix} \tag{7.1}$$

$$\theta = \begin{pmatrix} a \\ b \end{pmatrix} \tag{7.2}$$

Expanding this over the whole N measurement points leads to:

$$\Phi^T = \begin{pmatrix} \phi(1)^T \\ \phi(2)^T \\ \vdots \\ \phi(N)^T \end{pmatrix} \qquad \hat{Y}^T = \begin{pmatrix} \hat{y}(1) \\ \hat{y}(2) \\ \vdots \\ \hat{y}(N) \end{pmatrix} \tag{7.3}$$

with:

$$\hat{Y} = \Phi^T \theta \tag{7.4}$$

To find the best estimate for the parameter vector θ, a true (or reference) value $y(i)$ is needed which will be approximated with $\hat{y}(i) = ax(i) + b$. In our case we use the already calibrated distance sensor to get the values $y(i)$. $x(i)$ are the raw values given by the pressure sensor.

The goal is to find θ which minimizes the following *cost function* $V(\theta)$:

$$V(\theta) = \sum_{i=1}^{N} (y(i) - \hat{y}(i)) = \left(Y - \hat{Y}\right)^T \left(Y - \hat{Y}\right) \tag{7.5}$$

Using equation (7.4) in equation (7.5) leads to:

$$V(\theta) = \left(Y - \Phi^T\theta\right)^T \left(Y - \Phi^T\theta\right) \tag{7.6}$$

Now expanding the right-hand side of (7.6):

$$V(\theta) = \hat{Y}Y - \hat{Y}^T\Phi^T\theta - \theta^T\Phi Y + \theta^T\Phi\Phi^T\theta \tag{7.7}$$

This is a quadratic function of the *parameter vector* θ and can easily be minimized with respect to θ. The following two condition (7.8) and (7.9) have to be fulfilled for a certain value of θ to minimize the *cost function* (7.5). This is not only a local minimum, it is a global one too, because the function $V(\theta)$ to be minimized is a quadratic function.

$$\frac{\partial V(\theta)}{\partial \theta} = -(\Phi Y)^T - Y^T\Phi^T + 2\theta^T\Phi\Phi^T \overset{!}{=} 0 \tag{7.8}$$

$$\frac{\partial^2 V(\theta)}{\partial \theta^2} = 2\Phi\Phi^T \quad \text{should be a positive definite matrix} \tag{7.9}$$

The first condition (7.8) will be used to find a solution of the best estimate of θ. The second condition is always fulfilled, because a matrix multiplied with its transposed is always *positive semi definite*, in the non-trivial case even *positive definite*, which holds for all reasonable cases.

The next step is to solve the equation/condition (7.8) for θ. With some simple calculations one gets the following result:

$$\boxed{\theta_{opt} = (\Phi\Phi^T)^{-1}\Phi Y} \tag{7.10}$$

7.2 recursive LS-filter

The main idea is to generate a new estimate $\theta(k)$ at the time $t_k = kT_s$ based on the old estimate $\theta(k-1)$ and new measurement data $y(k)$. For the derivation of the *recursive least square filter*, the formulations (7.11) and (7.12) will be needed.

$$\Phi_N\Phi_N^T = \sum_{s=1}^{N} \phi(s)\phi(s)^T \tag{7.11}$$

$$\Phi_N Y_N = \sum_{s=1}^{N} \phi(s)y(s) \tag{7.12}$$

Furthermore, the *matrix inversion Lemma* (7.13) will be used. Let the matrices A, C and $(C^{-1} + DA^{-1}B)$ be invertible. It holds:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \tag{7.13}$$

Let $P(k)$ be defined using equation (7.11):

$$P(k) = \left(\Phi_k\Phi_k^T\right)^{-1} = \left(\sum_{s=1}^{k} \phi(s)\phi(s)^T\right)^{-1} \tag{7.14}$$

$$\begin{aligned} P(k) = \left(\sum_{s=1}^{k} \phi(s)\phi(s)^T\right)^{-1} &= \\ \left(\sum_{s=1}^{k-1} \phi(s)\phi(s)^T + \phi(k)\phi(k)^T\right)^{-1} &= \\ \left(P^{-1}(k-1) + \phi(k)\phi(k)^T\right)^{-1} \end{aligned}$$

Together with (7.13), this can be rearranged in a recursive form, which could easily be implemented. To use (7.13), let $A = P^{-1}(k-1)$, $B = \phi(k)$, $C = 1$ and $D = \phi^T(k-1)$.

$$P(k) = P(k-1) - \frac{P(k-1)\phi(k)\phi(k)^T P(k-1)}{1 + \phi(k)^T P(k-1)\phi(k)} \tag{7.15}$$

The next step is to find a recursive expression of θ_{opt}. (7.10) can rewritten with the equations (7.11), (7.12) and (7.14) the following way:

$$\theta_{opt}(k) = \left(\sum_{s=1}^{k} \phi(s)\phi(s)^T\right)^{-1} \left(\sum_{s=1}^{k} \phi(s)y(s)\right) = P(k)\left(\sum_{s=1}^{k} \phi(s)y(s)\right) \tag{7.16}$$

Now it is possible to derive a recursive expression:

$$\begin{aligned}\theta_{opt}(k) = P(k)\left(\sum_{s=1}^{k}\phi(s)y(s)\right) &= \\ P(k)\left(\sum_{s=1}^{k-1}\phi(s)y(s)+\phi(k)y(k)\right) &= \\ P(k)\left(P^{-1}(k-1)\theta_{opt}(k-1)+\phi(k)y(k)\right) &= \\ P(k)\left(\left(P^{-1}(k)-\phi(k)\phi^T(k)\right)\theta_{opt}(k-1)+\phi(k)y(k)\right) &= \\ P(k)\left(P^{-1}(k)\theta_{opt}(k-1)-\phi(k)\phi^T(k)\theta_{opt}(k-1)+\phi(k)y(k)\right)&\end{aligned}$$

$$\theta_{opt}(k) = \theta_{opt}(k-1) + P(k)\phi(k)\left[y(k)-\phi^T(k)\theta_{opt}(k-1)\right] \tag{7.17}$$

Wtih the equations (7.15) and (7.17), the *recursive least square filter* is given by:

$$\boxed{\begin{aligned}P(k) &= P(k-1) - \frac{P(k-1)\phi(k)\phi(k)^T P(k-1)}{1+\phi(k)^T P(k-1)\phi(k)} && (7.18)\\ \theta_{opt}(k) &= \theta_{opt}(k-1) + P(k)\phi(k)\left[y(k)-\phi^T(k)\theta_{opt}(k-1)\right] && (7.19)\end{aligned}}$$

7.3 formulation for the implementation of the first-order-LS-filter

The *recursive least square filter* can be written in an easy way to implement just by multiplying the different terms out. To do that, equation (7.18) is put into equation (7.19). After some rearrangement one can get:

$$\theta_{opt}(k) = \theta_{opt}(k-1) + \frac{P(k-1)\phi(k)}{1+\phi^T(k)P(k-1)\phi(k)}\left[y(k)-\phi^T(k)\theta_{opt}(k-1)\right] \tag{7.20}$$

The calculation of the single terms was done by Mathematica using the approach of (7.1) and (7.2) together with the equations (7.18) and (7.20). The Mathematica-calculations can be found in section B.1, where $x(k)$ is defined as *press* meaning the actual value of the pressure sensor and y is defined as *dist* representing the already calibrated value of the distance sensor. The function implemented for Matlab can be seen at the section A.1. This code might easily be adapted to other computer languages as C/C++.

Chapter 8

complementary filter: velocity and position fusion

The main idea of the complementary filter is to reduce the noise of one sensor with a low-pass-filter and to compensate the delay with the dynamics of the other sensor. Two sensors are used, where the derivate of the physical meaning of one sensor is the physical meaning of the other one. In the case of the Pixhawk, this would be the fusion of the angular velocity of the gyros and the angular position calculated from the IMU. The needed output is the attitude of the helicopter, i.e. the angular position. Basically this is the same as the one coming from the IMU, but this is really noisy and if it is just leaded through a low-pass-filter, the delay would be to huge to use it for control objectives. Therefore, the dynamic behavior of the gyros is needed.

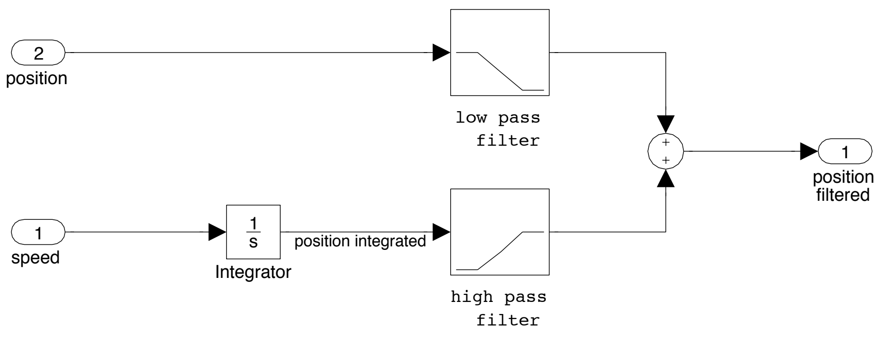


Figure 8.1: schema of the complementary filter

The schema of the complementary filter is shown in figure 8.1. The basic principle is to use the static behavior of the position sensor and the dynamic behavior of the velocity sensor. The signal from the position sensor is lead through a low-pass-filter. The velocity is integrated, so it can be compared with the physical value of the position sensor. Afterwards it is fed through a high-pass-filter to capture the dynamics. The noise of the velocity sensor will be supressed too, because an integrator together with a high-pass-filter has a roll-off behavior for higher frequencies.

8.1 continious derivation

The first step is to derive a continious version of the complementary filter. The integration part can be written as the following transfer function:

$$I(s) = \frac{1}{s} \tag{8.1}$$

After the integration of the velocity, the signal should theoretically be the same as the one of the position sensor. But in practice, this isn't the case at all, e.g. offset drifts caused by the numerical integration. This is the reason why the static behavior is taken from position sensor only and just the dynamics of the integrated signal of the velocity sensor.

To have a physically valid output, the following condition should hold:

$$\boxed{LP(s) + HP(s) \overset{!}{=} 1} \tag{8.2}$$

Where $LP(s)$ is the transfer function of the low-pass-filter and $HP(s)$ is the one of the high-pass-filter. The explanation of the condition (8.2) is very simple, but first a few definitions. The variable for the measured velocity is v, the one of the positon is s. s_I is the integrated velocity and s_o is the output of the complementary filter. The related variables of the different signals in the laplace-domain are given as capital letters, i.e. $V(s)$, $S(s)$, $S_I(s)$ and $S_o(s)$.

$$\begin{aligned}
S_o(s) = LP(s)S(s) + HP(s)I(s)V(s) &= \\
LP(s)S(s) + HP(s)S_I(s) &\approx \\
LP(s)S(s) + HP(s)S(s) &= \\
(LP(s) + HP(s))S(s) &\overset{!}{=} S(s) \\
\Rightarrow LP(s) + HP(s) &\overset{!}{=} 1
\end{aligned}$$

Condition (8.2) guarantees that the output of the complementary filter does have unity gain and zero phase with reference to the position.

As low-pass-filter, the following transfer function is chosen:

$$LP(s) = \frac{\omega_c}{s + \omega_c} \tag{8.3}$$

ω_c is a tuning parameter. It represents the frequency, where magnitude of the filter starts to decrease with $20dB/dec$ in the bode plot. Figure 8.2 shows the magnitude part of the bode plot of the low-pass-filter for some different ω_c's.

As high-pass-filter, the following transfer function is chosen:

$$HP(s) = \frac{s}{s + \omega_c} \tag{8.4}$$

ω_c is the frequency, after which the magnitude of the high-pass-filter doesn't increase anymore and remains constant. Figure 8.3 shows the bode plot of the high-pass-filter for different ω_c's. It is by the way also the frequency, where the magnitudes of the low-pass- and high-pass-filter cross each other. With the tuning parameter ω_c, one can decide on which frequency domains the two different filters work. The two following rules can help with the tuning of the complementary filter:

- $\omega_c \nearrow$: the low-pass-filter also lets "higher" frequencies through (fig. 8.2)

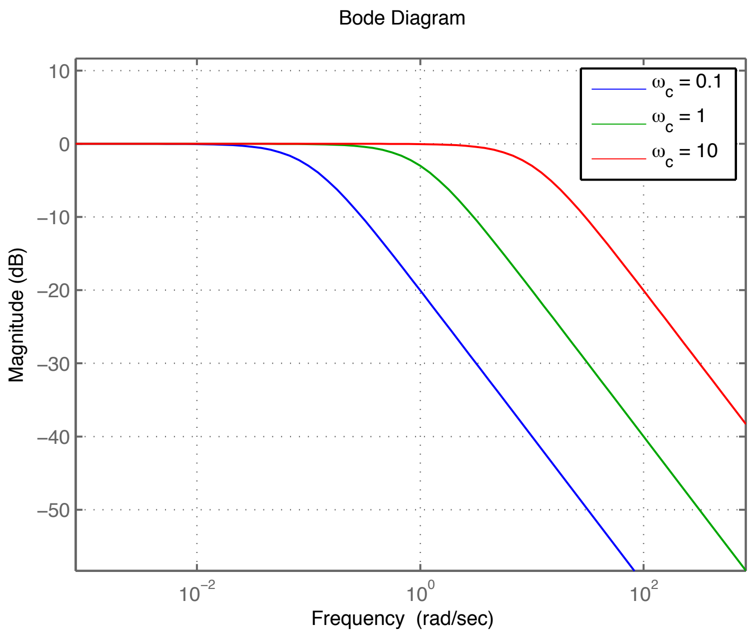


Figure 8.2: bode plot (magnitude) of the low-pass-filter

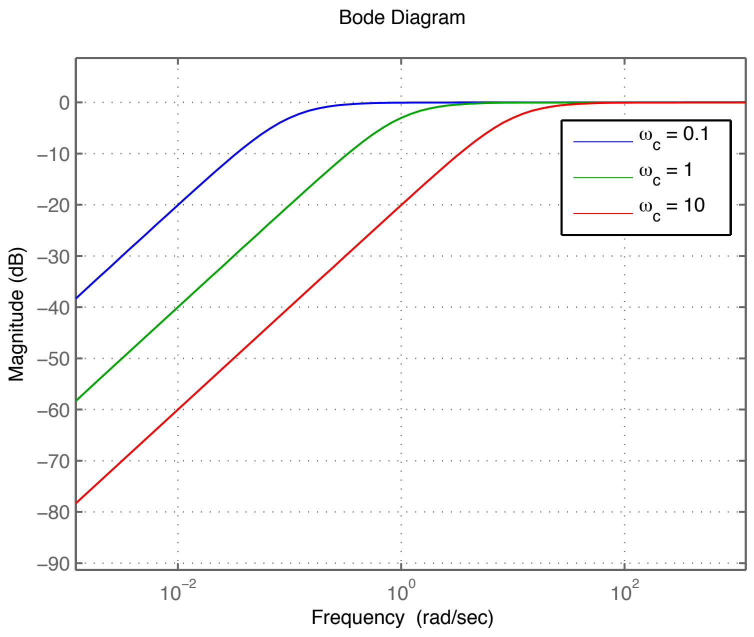


Figure 8.3: bode plot (magnitude) of the high-pass-filter

- $\omega_c \searrow$: the high-pass-filter also lets "lower" frequencies through (fig. 8.3)

To show that the condition (8.2) holds for the chosen filters ((8.3) and (8.4)), the equations (8.3) and (8.4) are inserted into the left-hand side of condition (8.2):

$$LP(s) + HP(s) = \frac{\omega_c}{s+\omega_c} + \frac{s}{s+\omega_c} = \frac{s+\omega_c}{s+\omega_c} = 1 \qquad \text{(OK)}$$

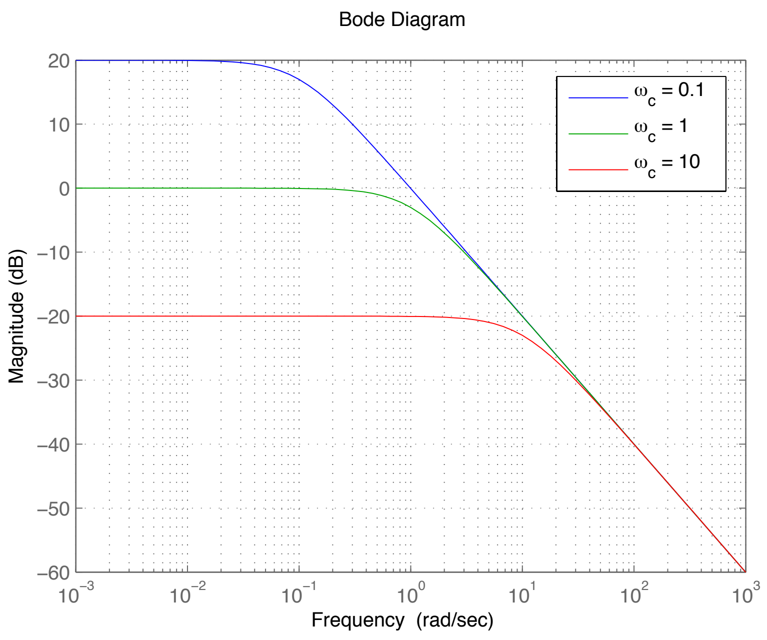


Figure 8.4: bode plot (magnitude) of the high-pass-filter together with integrator

Figure 8.4 shows the bode plot (magnitude) of the high-pass-filter together with the integrator for the same ω_c's as in the figures 8.2 and 8.3. One can see there, that the high-pass-filter together with the integrator is a low-pass-filter too, but with static gain not obligatory equal to 1 ($0dB$). This is welcome, because the noise of the velocity sensor will also be suppressed.

The continious transfer function of the complementary filter is given as:

$$CF(s) = \begin{pmatrix} \frac{\omega_c}{s+\omega_c} & \frac{s}{(s+\omega_c)s} \end{pmatrix} = \begin{pmatrix} \frac{\omega_c}{s+\omega_c} & \frac{1}{s+\omega_c} \end{pmatrix} \tag{8.5}$$

With $U(s)$ as input vector, one can write the dynamics of the complementary filter the following way:

$$S_o(s) = CF(s)U(s) = CF(s)\begin{pmatrix} S(s) \\ V(s) \end{pmatrix} =$$
$$= \frac{\omega_c}{s+\omega_c}S(s) + \frac{1}{s+\omega_c}V(s)$$

Remark 11. *The first order approaches of the high-pass- and low-pass-filters can also be extended to higher order terms, as long as the condition (8.2) holds.*

Chapter 9

Kalman-Filter

The main idea is to fuse the measurement of a physical coordinate and its second derivative, the acceleration along the physical coordinate. The resulting output should have less noise than the initially measured values of the sensors. As an additional achievement, one can get also informations about the velocity (first derivative of the physical coordinate). To reach this goal, a Kalman Filter will be used.

9.1 second-order derivation

The first step is to define the dynamics, which relate the two sensors to each other. The first one measures the coordinate $z(t)$ representing the altitude. In the case of the Pixhawk helicopter, this is done by an ultrasonic distance sensor facing downwards. The second sensor measures the accelerations along the altitude axis, called $\gamma_z(t)$. There is some bias/offset B_γ on the acceleration sensor. Let the relation of the true acceleration $\ddot{z}(t)$ with the bias B_γ and the acceleration sensor $\gamma_z(t)$ be defined as:

$$\gamma_z(t) = \ddot{z}(t) + B_\gamma \tag{9.1}$$

This will be used in chapter 9.2 when the bias is introduced as a new state. The general continious equations of a dynamic model are given as first order differential equations:

$$\dot{x}(t) = f(x(t), u(t)) + w(t) \tag{9.2}$$

$$y(t) = h(x(t), u(t)) + v(t) \tag{9.3}$$

$x(t)$ is the state vector, $u(t)$ the input vector, $w(t)$ the process noise, $y(t)$ the measurement vector and $v(t)$ the measurement noise. This equations can be linearized as shown in chapter **??**.

Figure 9.1 shows the relations between the different sensors used in the Kalman Filter, represented in a continious state space diagram. One can can see there, that the associated system of differential equations is already linear and is given in equation (9.4). For the sake of simplicity, the time dependence $\cdots(t)$ is omitted in the next steps. The following notation is used:

- $x = \begin{pmatrix} z \\ \dot{z} \end{pmatrix}$ is the state vector
- $u = \ddot{z} = \gamma_z - B_\gamma$ is the input to the system ($\in \mathbb{R}^{1\times 1}$)

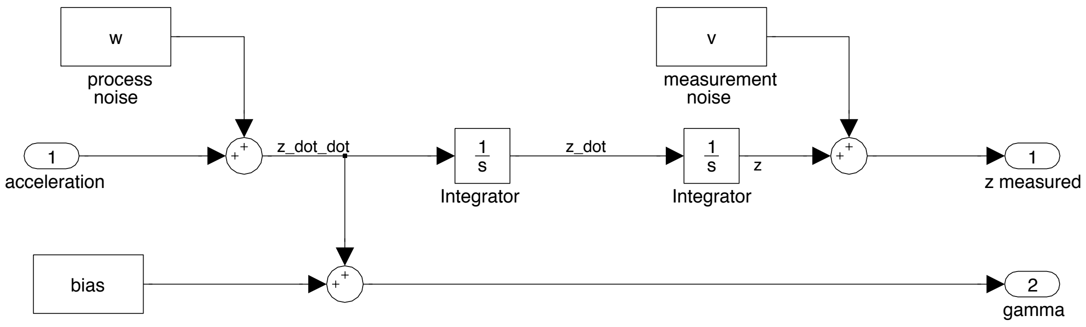


Figure 9.1: state space diagram of the sensor-relations

- $y = z_m$ is the measured altitude, i.e. the output of the system

$$\dot{x} = \begin{pmatrix} \dot{z} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} z \\ \dot{z} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u + w \tag{9.4}$$

The (linear) measurement equation is given as:

$$y = z_m = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} z \\ \dot{z} \end{pmatrix} + v \tag{9.5}$$

9.2 third-order with extended bias-state

B_γ is introduced as a new state variable. This has the reason, that the offset of the acceleration sensor don't has to be known. The Kalman Filter can be tuned so that it can work more efficient and has better robustness. Assuming that B_γ is static, the dynamics can be extended by the following equation:

$$\frac{dB_\gamma}{dt} = \dot{B}_\gamma = 0 \tag{9.6}$$

Using equations (9.1) and (9.6), the system of differential equations (9.4) can be rewritten in a 3rd order representation:

$$\begin{pmatrix} \dot{z} \\ \ddot{z} \\ \dot{B}_\gamma \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} z \\ \dot{z} \\ B_\gamma \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \gamma_z + w \tag{9.7}$$

The measurement equation can be written as:

$$y = z_m = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} z \\ \dot{z} \\ B_\gamma \end{pmatrix} + v \tag{9.8}$$

Therefore, the constinious state-space description is given with the following matrices:

- $A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$

- $B = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

- $C = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$

- $D = 0$

The extended state vector of the system is now given by:

$$x = \begin{pmatrix} z \\ \dot{z} \\ B_\gamma \end{pmatrix} \tag{9.9}$$

The new input u to the system is the value of the acceleration sensor γ_z and the output y of the system is again z_m (the measured altitude).

The final Kalman Filter will be implemented in a discrete form. To reach that, a discrete-time formulation of the continious state-space system (9.7) is needed. The sample-time is called T_s. The discrete notation with time index will be used, x_k for example is the state vector $x(t)$ at the time step $t = kT_s$. The continious transition matrix is defined as:

$$\Phi(t) = e^{At} \tag{9.10}$$

The discrete transition matrix Φ_k is constant for linear time-invariant systems as the one given in equation (9.7) and is equal to the continious transition matrix (9.10) evaluated at the time $t = T_s$:

$$\Phi_k = \Phi(t = T_s) = e^{AT_s} \tag{9.11}$$

A Taylor series is used to approximate the matrix exponential of $\Phi(t)$:

$$\Phi(t) = e^{At} = I + At + \frac{A^2t^2}{2!} + \ldots + \frac{A^nt^n}{n!} + \ldots \tag{9.12}$$

Using the A matrix from above leads to:

$$\begin{aligned} A &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} \\ A^2 &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ A^3 = A^2A &= \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ \Rightarrow A^3 = A^4 = \ldots = A^n = \ldots &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

The higher terms of the Taylor approximation are all equal to zero, which means that the exact version of the continious transition matrix $\Phi(t)$ can be calculated:

$$\Phi(t) = e^{At} = I + At + \frac{A^2t^2}{2!} =$$
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + t \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix} + \frac{t^2}{2} \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \Phi(t) = \begin{pmatrix} 1 & t & -\frac{t^2}{2} \\ 0 & 1 & -t \\ 0 & 0 & 1 \end{pmatrix} \tag{9.13}$$

And the discrete transition matrix Φ_k is given as:

$$\boxed{\Phi_k = \Phi(t = T_s) = \begin{pmatrix} 1 & T_s & -\frac{T_s^2}{2} \\ 0 & 1 & -T_s \\ 0 & 0 & 1 \end{pmatrix}} \tag{9.14}$$

The discrete input matrix G_k can be calculated using the following equation:

$$G_k = \int_0^{T_s} \Phi(\tau) d\tau B \tag{9.15}$$

Together with equation (9.13) and the continious input matrix B from above, one can get:

$$G_k = \int_0^{T_s} \begin{pmatrix} 1 & \tau & -\frac{\tau^2}{2} \\ 0 & 1 & -\tau \\ 0 & 0 & 1 \end{pmatrix} d\tau \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} T_s & \frac{T_s^2}{2} & -\frac{T_s^3}{6} \\ 0 & T_s & -\frac{T_s^2}{2} \\ 0 & 0 & T_s \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\boxed{\Rightarrow G_k = \begin{pmatrix} \frac{T_s^2}{2} \\ T_s \\ 0 \end{pmatrix}} \tag{9.16}$$

The discrete time representation (state update equation) is formulated as:

$$x_{k+1} = \Phi_k x_k + G_k u_k + w_k = \begin{pmatrix} 1 & T_s & -\frac{T_s^2}{2} \\ 0 & 1 & -T_s \\ 0 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} \frac{T_s^2}{2} \\ T_s \\ 0 \end{pmatrix} u_k + w_k \tag{9.17}$$

And the discrete measurement equation is given analog to equation (9.8) as:

$$y_k = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} x_k + v_k \tag{9.18}$$

The discrete measurement matrix is defined as:

$$\boxed{H_k = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}} \tag{9.19}$$

Where:

- $x_k = \begin{pmatrix} z(t=kT_s) \\ \dot{z}(t=kT_s) \\ B_\gamma(t=kT_s) \end{pmatrix}$ is the discrete state vector
- $u_k = \gamma_z(t=kT_s)$ is the input to the discrete system
- $y_k = z_m(t=kT_s)$ is the measured altitude at time step k
- w_k and v_k are the process- and measurement-noise at time step k

The next step is to find the discrete process-noise covariance matrix Q_k. To calculate this, the continious process-noise covariance matrix Q is needed.

$$Q = E\left[w(t)w^T(\tau)\right] \tag{9.20}$$

Let Q_{2nd} of the second order system (9.4) be:

$$Q_{2nd} = \begin{pmatrix} 0 & 0 \\ 0 & \Gamma_{acc} \end{pmatrix} \tag{9.21}$$

The process noise is assumed to enter the system at the highest order differential equation. Γ_{acc} is the process power spectral density and describes the quality of the chosen mathematical model (see fig. 9.1), but later more about this. For the third order system (9.7), Q is defined as:

$$Q = \begin{pmatrix} \multicolumn{2}{c}{Q_{2nd}} & \begin{matrix} 0 \\ 0 \end{matrix} \\ 0 & 0 & \Gamma_{bias} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Gamma_{acc} & 0 \\ 0 & 0 & \Gamma_{bias} \end{pmatrix} \tag{9.22}$$

Γ_{bias} is the process power spectral density of the differential equation (9.6). With this tuning parameter, one can control the dynamic behaviour of the offset B_γ, i.e. how fast the offset can change online.

Q_k can be calculated as:

$$Q_k = E\left[w_k w_k^T\right] = \int_0^{T_s} \Phi(\tau) Q \Phi(\tau)^T d\tau \tag{9.23}$$

Together with equations (9.13) and (9.22), this leads to:

$$\begin{aligned} Q_k = &\int_0^{T_s} \begin{pmatrix} 1 & \tau & -\frac{\tau^2}{2} \\ 0 & 1 & -\tau \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Gamma_{acc} & 0 \\ 0 & 0 & \Gamma_{bias} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ \tau & 1 & 0 \\ -\frac{\tau^2}{2} & -\tau & 1 \end{pmatrix} d\tau = \\ &\int_0^{T_s} \begin{pmatrix} 0 & \tau\Gamma_{acc} & -\frac{\tau^2}{2}\Gamma_{bias} \\ 0 & \Gamma_{acc} & -\tau\Gamma_{bias} \\ 0 & 0 & \Gamma_{bias} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ \tau & 1 & 0 \\ -\frac{\tau^2}{2} & -\tau & 1 \end{pmatrix} d\tau = \\ &\int_0^{T_s} \begin{pmatrix} \tau^2\Gamma_{acc} + \frac{\tau^4}{4}\Gamma_{bias} & \tau\Gamma_{acc} + \frac{\tau^3}{2}\Gamma_{bias} & -\frac{\tau^2}{2}\Gamma_{bias} \\ \tau\Gamma_{acc} + \frac{\tau^3}{2}\Gamma_{bias} & \Gamma_{acc} + \tau^2\Gamma_{bias} & -\tau\Gamma_{bias} \\ -\frac{\tau^2}{2}\Gamma_{bias} & -\tau\Gamma_{bias} & \Gamma_{bias} \end{pmatrix} d\tau = \\ &\left.\begin{pmatrix} \frac{\tau^3}{3}\Gamma_{acc} + \frac{\tau^5}{20}\Gamma_{bias} & \frac{\tau^2}{2}\Gamma_{acc} + \frac{\tau^4}{8}\Gamma_{bias} & -\frac{\tau^3}{6}\Gamma_{bias} \\ \frac{\tau^2}{2}\Gamma_{acc} + \frac{\tau^4}{8}\Gamma_{bias} & \tau\Gamma_{acc} + \frac{\tau^3}{3}\Gamma_{bias} & -\frac{\tau^2}{2}\Gamma_{bias} \\ -\frac{\tau^3}{6}\Gamma_{bias} & -\frac{\tau^2}{2}\Gamma_{bias} & \tau\Gamma_{bias} \end{pmatrix}\right|_0^{T_s} \end{aligned}$$

$$\boxed{\Rightarrow Q_k = \begin{pmatrix} \frac{T_s^3}{3}\Gamma_{acc} + \frac{T_s^5}{20}\Gamma_{bias} & \frac{T_s^2}{2}\Gamma_{acc} + \frac{T_s^4}{8}\Gamma_{bias} & -\frac{T_s^3}{6}\Gamma_{bias} \\ \frac{T_s^2}{2}\Gamma_{acc} + \frac{T_s^4}{8}\Gamma_{bias} & T_s\Gamma_{acc} + \frac{T_s^3}{3}\Gamma_{bias} & -\frac{T_s^2}{2}\Gamma_{bias} \\ -\frac{T_s^3}{6}\Gamma_{bias} & -\frac{T_s^2}{2}\Gamma_{bias} & T_s\Gamma_{bias} \end{pmatrix}} \tag{9.24}$$

Let finally R be the covariance matrix of the measurement noise vector $v(t)$:

$$R = E\left[v(t)v^T(\tau)\right] \tag{9.25}$$

In the case, where $y(t)$ (or y_k) of the measurement equation is one-dimensional, R is one-dimensional and equal to the standard deviation of the sensor:

$$\boxed{R_k = R = \sigma_{alt}^2} \tag{9.26}$$

σ_{alt} is the standard deviation of the altitude sensor.

The results (9.14), (9.16), (9.19), (9.24) and (9.26) are everything which is needed to run the Kalman filter. The algorithm is given as:

The Kalman gain matrix:

$$K_k = P_{k|k-1}H_k^T\left[H_kP_{k|k-1}H_k^T + R_k\right]^{-1} \tag{9.27}$$

The new state estimation:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left[y_k - H_k\hat{x}_{k|k-1}\right] \tag{9.28}$$

State prediction (extrapolation):

$$\hat{x}_{k+1|k} = \Phi_k\hat{x}_{k|k} + G_ku_k \tag{9.29}$$

Update of the state vector error covariance matrix:

$$P_{k|k} = \left[I - K_kH_k\right]P_{k|k-1} \tag{9.30}$$

Prediction of the state vector error covariance matrix:

$$P_{k+1|k} = \Phi_kP_{k|k}\Phi_k^T + Q_k \tag{9.31}$$

time-shift over one sample time T_s:

$$P_{k+1|k} \rightarrow P_{k|k-1} \qquad \hat{x}_{k+1|k} \rightarrow \hat{x}_{k|k-1} \tag{9.32}$$

$\hat{x}_{k|k}$ is the estimation of state x_k based on the informations at step k and $\hat{x}_{k|k-1}$ is the estimation of the state x_k based on the informations at step $k-1$. In other words, $\hat{x}_{k|k-1}$ is the state prediction out of $\hat{x}_{k-1|k-1}$ and u_k, what can be seen in equation (9.29).

$P_{k|k-1}$ and $P_{k|k}$ are the covariance matrices of the state vector errors $e_{k|k-1}$ and $e_{k|k}$ respectively:

$$P_{k|k-1} = E\left[e_{k|k-1}e_{k|k-1}^T\right] \qquad P_{k|k} = E\left[e_{k|k}e_{k|k}^T\right] \tag{9.33}$$

Where:

$$e_{k|k-1} = x_k - \hat{x}_{k|k-1} \qquad e_{k|k} = x_k - \hat{x}_{k|k} \tag{9.34}$$

As it can be seen in equation (9.28), the Kalman estimation $\hat{x}_{k|k}$ of the state x_k relies, depending on the Kalman gain K_k, either more on the state prediction $\hat{x}_{k|k-1}$, on the measurement y_k or any trade-off between the two. If the sensor(s) is/are poor, the elements of the R_k matrix increase, the K_k matrix is getting smaller (see equation (9.27)) and the state estimation is more or less equal to the predicted state. The same happens, if the confidence on the mathematical model (dynamic system) is large, i.e. the process power spectral density Γ_{acc} is small. One can say, if the gain matrix K_k is large, the estimation $\hat{x}_{k|k}$ relies more on the measurement data y_k and if K_k is small, the estimation relies more on the state prediction based on the mathematical model.

In summary:

- measurement noise variance large $\Rightarrow R \nearrow \Rightarrow K \searrow$
- measurement noise variance small $\Rightarrow R \searrow \Rightarrow K \nearrow$
- process noise variance large $\Rightarrow P \nearrow \Rightarrow K \nearrow$
- process noise variance small $\Rightarrow P \searrow \Rightarrow K \searrow$

9.3 formulation for the implementation

To bring the Kalman filter in a simple form to implement, equations (9.27) to (9.32) can be rearranged the following way. The equation (9.27) is put into equation (9.28):

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + P_{k|k-1} H_k^T \left[H_k P_{k|k-1} H_k^T + R_k \right]^{-1} \left[y_k - H_k \hat{x}_{k|k-1} \right] \tag{9.35}$$

Equation (9.29) is left as it was:

$$\hat{x}_{k+1|k} = \Phi_k \hat{x}_{k|k} + G_k u_k \tag{9.36}$$

Equation (9.30) is used in equation (9.31):

$$P_{k+1|k} = \Phi_k \left[I - K_k H_k \right] P_{k|k-1} \Phi_k^T + Q_k \tag{9.37}$$

Equations (9.35), (9.36) and (9.37) are all that are needed to run the Kalman filter. In the case of the Pixhawk, the matrices (9.14), (9.16), (9.19), (9.24) and (9.26) are used to calculate every term with the help of Mathematica. The resulting equations don't consist of any matrix calculation, so it can easily be implemented on any embedded controller. The calculations on Mathematica can be read at chapter B.2. An example of how it can be implemented is shown in chapter A.2 as Matlab function.

The used syntax is more or less the same in both:

- **Ts** stands for the sample time T_s
- **sgmAlt** $= \sigma_{alt}$ is the standard deviation of the measurement noise (altitude sensor)
- **Gmacc** $= \Gamma_{acc}$ is the process power spectral density
- **Gmbias** $= \Gamma_{bias}$ is the process power spectral density of the bias differential equation
- **altMeas** $= y_k = z_m$ is the measured altitude
- **accMeas** $= u_k = \gamma_z$ is the system input, i.e. the measured acceleration
- **Xpred** $= x_{k|k-1}$ is the previous prediction of the actual state based on the previous state estimation and system input
- **XestNew** $= x_{k|k}$ is the Kalman estimation of the actual state

- **XpredNew** $= x_{k+1|k}$ is the prediction of the next state based on the actual state estimation and system input
- **P** $= P_{k|k-1}$ is the previous prediction of the covariance matrix of the state vector error
- **Pnew** $= P_{k+1|k}$ is the next prediction of the covariance matrix of the state vector error

9.4 results based on non-flying pixhawk unit

The shown Kalman filter was tested on the real Pixhawk unit, which was manually moved. The results are shown in figure 9.2. One can see the unfiltered data from the ultrasonic sensor, filtered data from a low-pass filter and filtered data from the Kalman filter (altitude estimated). It's obvious, that the low-pass filter has a delay as expected. The signal from the Kalman filter on the other hand, has a sufficient time behaviour.

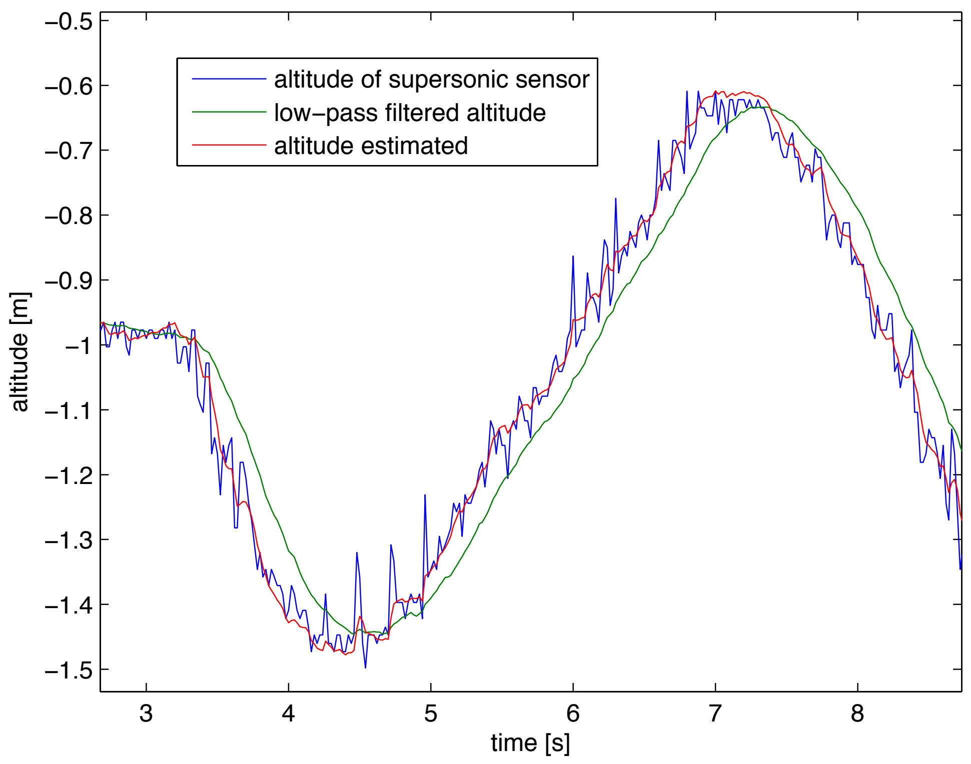


Figure 9.2: Kalman filter tested on Pixhawk unit

As it was said before, the Kalman filter has the velocity as output too. This is very usefull in control application, e.g. a state controller for the altitude can be implemented without additional observer. Figure 9.3 shows the associated velocity of the example in figure 9.2. But as the velocity was not measured, there is no reference signal plotted.

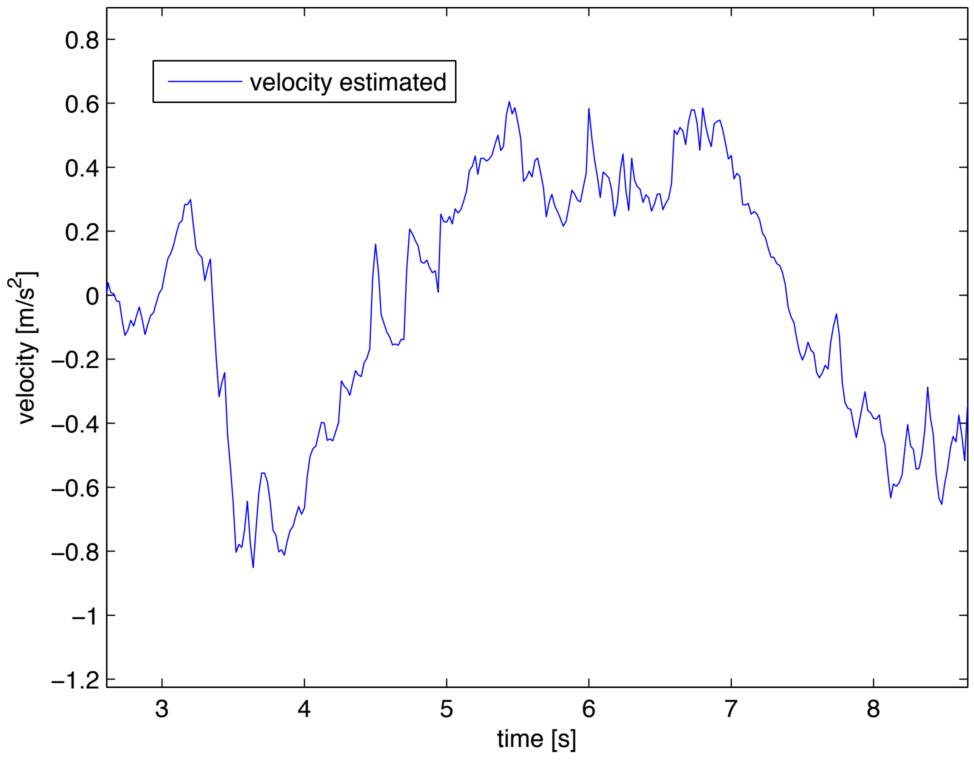


Figure 9.3: Kalman filter velocity estimation on Pixhawk unit

Appendix A

Matlab codes and Simulink files

A.1 Matlab-implementation of the recursive least-square-filter

The following Matlab function is a *recursive least-square-filter* used for a first order approximation:

```
function [Theta_new, P_new] = least_square(press_raw, dist, Theta, P)

P11 = P(1,1);
P12 = P(1,2);
P21 = P(2,1);
P22 = P(2,2);

a = Theta(1);
b = Theta(2);

% New pressure-sensor values: altitude(press_raw) = a*press_raw + b [m]
a_new = a + ((P12 + P11*press_raw)*(-b + dist - a*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1);
b_new = b + ((P22 + P21*press_raw)*(-b + dist - a*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1);

% Update least-square states
P11_new = P11*(1 - (press_raw*(P12 + P11*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1))...
    - (P21*(P12 + P11*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1);
P12_new = P12*(1 - (press_raw*(P12 + P11*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1))...
    - (P22*(P12 + P11*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1);
P21_new = P21*(1 - (P22 + P21*press_raw)/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1))...
    - (P11*press_raw*(P22 + P21*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1);
P22_new = P22*(1 - (P22 + P21*press_raw)/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1))...
    - (P12*press_raw*(P22 + P21*press_raw))/...
    (P22 + P12*press_raw + press_raw*(P21 + P11*press_raw) + 1);

```

```
% Sample-Time shift
Theta_new = [a_new; b_new];
P_new = [P11_new P12_new; P21_new P22_new];
```

A.2 Matlab-implementation of the Kalman filter

The following Matlab function is a *3rd order Kalman filter with extended bias state*:

```
function [x_est_new x_pred_new Pkk_1_new] = altitude_kalman(x_pred,...
    Pkk_1, altMeas, accMeas, Ts, sgmAcc, sgmAlt, Gmacc, Gmbias)

P11 = Pkk_1(1,1);
P12 = Pkk_1(1,2);
P13 = Pkk_1(1,3);
P21 = Pkk_1(2,1);
P22 = Pkk_1(2,2);
P23 = Pkk_1(2,3);
P31 = Pkk_1(3,1);
P32 = Pkk_1(3,2);
P33 = Pkk_1(3,3);

hPred = x_pred(1);
hDotPred = x_pred(2);
BiasPred = x_pred(3);

    % New Estimation (embedded)
    hEstNew = hPred + P11*(altMeas - hPred)/(P11 + sgmAlt^2);
    hDotEstNew = hDotPred + P21*(altMeas - hPred)/(P11 + sgmAlt^2);
    BiasEstNew = BiasPred + P31*(altMeas - hPred)/(P11 + sgmAlt^2);

x_est_new = [hEstNew hDotEstNew BiasEstNew]';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% here control step out of the state estimation x_est %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % New prediction and update (embedded)
    hPredNew = hEstNew + Ts*hDotEstNew - Ts^2/2*BiasEstNew + Ts^2/2*accMeas;
    hDotPredNew = hDotEstNew -Ts*BiasEstNew + Ts*accMeas;
    BiasPredNew = BiasEstNew;




x_pred_new = [hPredNew hDotPredNew BiasPredNew]';

    P11New = (1/(60*(P11 + sgmAlt^2)))*(P11*(60*sgmAlt^2 + Ts^2*(60*P22...
        + Ts*(20*Gmacc - 30*P23 - 30*P32 + 15*P33*Ts + 3*Gmbias*Ts^2)))...
        + Ts*(30*P21*(2*sgmAlt^2 + P13*Ts^2) + 30*P12*(2*sgmAlt^2 +...
        Ts*(-2*P21 + P31*Ts)) + Ts*(-15*P13*(2*sgmAlt^2 + P31*Ts^2) +...
        sgmAlt^2*(60*P22 - 30*P31 + Ts*(20*Gmacc - 30*P23 - 30*P32 +...
        15*P33*Ts + 3*Gmbias*Ts^2)))));
    P12New = (1/(8*(P11 + sgmAlt^2)))*(4*P12*(2*sgmAlt^2 + Ts*(-2*P21 +...
        P31*Ts)) + Ts*(-4*P13*(2*sgmAlt^2 + Ts*(-2*P21 + P31*Ts)) +...
        P11*(8*P22 + Ts*(4*Gmacc - 8*P23 - 4*P32 + 4*P33*Ts +...
        Gmbias*Ts^2)) + sgmAlt^2*(8*P22 + Ts*(4*Gmacc - 8*P23 - 4*P32 +...
        4*P33*Ts + Gmbias*Ts^2))));
    P13New = ((P11 + sgmAlt^2)*Ts*(6*P23 - Ts*(3*P33 + Gmbias*Ts)) +...
        3*P13*(2*sgmAlt^2 + Ts*(-2*P21 + P31*Ts)))/(6*(P11 + sgmAlt^2));
    P21New = P21 - (P11*P21)/(P11 + sgmAlt^2) - (P31*sgmAlt^2*Ts)/(P11 +...
        sgmAlt^2) + (Gmacc*Ts^2)/2 + (Gmbias*Ts^4)/8 + (Ts*(P12*(-P21 +...
        P31*Ts) + (P11 + sgmAlt^2)*(P22 - P32*Ts)))/(P11 + sgmAlt^2) -...
        (Ts^2*(P13*(-P21 + P31*Ts) + (P11 + sgmAlt^2)*(P23 - P33*Ts)))/...
        (2*(P11 + sgmAlt^2));
    P22New = P22 + (P12*(-P21 + P31*Ts))/(P11 + sgmAlt^2) + (1/3)*Ts*...
        (3*Gmacc - 3*P23 - 3*P32 + (3*P13*P21)/(P11 + sgmAlt^2) +...
        3*P33*Ts - (3*P13*P31*Ts)/(P11 + sgmAlt^2) + Gmbias*Ts^2);
    P23New = P23 - (1/2)*Ts*(2*P33 + Gmbias*Ts) + (P13*(-P21 + P31*Ts))/...
        (P11 + sgmAlt^2);
    P31New = ((P11 + sgmAlt^2)*Ts*(6*P32 - Ts*(3*P33 + Gmbias*Ts)) +...
        3*P31*(2*sgmAlt^2 + Ts*(-2*P12 + P13*Ts)))/(6*(P11 + sgmAlt^2));
    P32New = P32 - (P12*P31)/(P11 + sgmAlt^2) - P33*Ts + (P13*P31*Ts)/...
        (P11 + sgmAlt^2) - (Gmbias*Ts^2)/2;
    P33New = P33 - (P13*P31)/(P11 + sgmAlt^2) + Gmbias*Ts;

Pkk_1_new = [P11New P12New P13New;...
    P21New P22New P23New;...
    P31New P32New P33New];
```

Appendix B

Mathematica calculations

B.1 Recursive least-square-filter

This is the derivation of the terms of the *recursive least-square-filter* based on a first order approximation:

```
In[198]:= P = {{P11, P12}, {P21, P22}}

Out[198]= {{P11, P12}, {P21, P22}}

In[199]:= Phi = {{press}, {1}}

Out[199]= {{press}, {1}}

In[200]:= Theta = {{a}, {b}}

Out[200]= {{a}, {b}}

In[201]:= y = dist

Out[201]= dist

In[202]:= Fact = Extract[Extract[1 / (Transpose[Phi] . P . Phi + 1), 1], 1]

                               1
Out[202]= ──────────────────────────────────────────────
          1 + P22 + P12 press + press (P21 + P11 press)

In[203]:= ThetaNew =
           Fact * P . Phi * (y - Extract[Extract[Transpose[Phi] . Theta, 1],
             1]) + Theta

                  (-b + dist - a press) (P12 + P11 press)
Out[203]= {{a + ──────────────────────────────────────────────},
                1 + P22 + P12 press + press (P21 + P11 press)

            {b + (-b + dist - a press) (P22 + P21 press)
                 ──────────────────────────────────────────────}}
                 1 + P22 + P12 press + press (P21 + P11 press)

In[204]:= P11New = Extract[Extract[(IdentityMatrix[2] - Fact * P . Phi . Transpose[Phi]) . P, 1], 1]

                      P21 (P12 + P11 press)
Out[204]= - ────────────────────────────────────────────── +
            1 + P22 + P12 press + press (P21 + P11 press)

                           press (P12 + P11 press)
            P11 (1 - ──────────────────────────────────────────────)
                     1 + P22 + P12 press + press (P21 + P11 press)

In[205]:= P12New = Extract[Extract[(IdentityMatrix[2] - Fact * P . Phi . Transpose[Phi]) . P, 1], 2]

Out[205]= - (P22 (P12 + P11 press)) / (1 + P22 + P12 press + press (P21 + P11 press)) +
            P12 (1 - (press (P12 + P11 press)) / (1 + P22 + P12 press + press (P21 + P11 press)))

In[206]:= P21New = Extract[Extract[(IdentityMatrix[2] - Fact * P . Phi . Transpose[Phi]) . P, 2], 1]

Out[206]= - (P11 press (P22 + P21 press)) / (1 + P22 + P12 press + press (P21 + P11 press)) +
            P21 (1 - (P22 + P21 press) / (1 + P22 + P12 press + press (P21 + P11 press)))

In[207]:= P22New = Extract[Extract[(IdentityMatrix[2] - Fact * P . Phi . Transpose[Phi]) . P, 1], 1]

Out[207]= - (P21 (P12 + P11 press)) / (1 + P22 + P12 press + press (P21 + P11 press)) +
            P11 (1 - (press (P12 + P11 press)) / (1 + P22 + P12 press + press (P21 + P11 press)))
```

B.2 Kalman filter

This is the derivation of the terms of the *3rd order Kalman filter with extended bias state*

```
In[58]:= B = {{0}, {1}, {0}}

Out[58]= {{0}, {1}, {0}}

In[59]:= A = {{0, 1, 0}, {0, 0, -1}, {0, 0, 0}}

Out[59]= {{0, 1, 0}, {0, 0, -1}, {0, 0, 0}}

In[60]:= Phi[t] = MatrixExp[A * t]
```

Out[60]= $\left\{\left\{1, t, -\frac{t^2}{2}\right\}, \{0, 1, -t\}, \{0, 0, 1\}\right\}$

```
In[61]:= Phik = Phi[t] /. t → Ts
```

Out[61]= $\left\{\left\{1, Ts, -\frac{Ts^2}{2}\right\}, \{0, 1, -Ts\}, \{0, 0, 1\}\right\}$

```
In[62]:= Gk = Integrate[Phi[t], {t, 0, Ts}].B
```

Out[62]= $\left\{\left\{\frac{Ts^2}{2}\right\}, \{Ts\}, \{0\}\right\}$

```
In[63]:= Q = {{0, 0, 0}, {0, Gmacc, 0}, {0, 0, Gmbias}}

Out[63]= {{0, 0, 0}, {0, Gmacc, 0}, {0, 0, Gmbias}}

In[64]:= Qk = Integrate[Phi[t].Q.Transpose[Phi[t]], {t, 0, Ts}]
```

Out[64]=
$$\left\{\left\{\frac{\text{Gmacc}\,Ts^3}{3} + \frac{\text{Gmbias}\,Ts^5}{20}, \frac{\text{Gmacc}\,Ts^2}{2} + \frac{\text{Gmbias}\,Ts^4}{8}, -\frac{\text{Gmbias}\,Ts^3}{6}\right\},\right.$$
$$\left\{\frac{\text{Gmacc}\,Ts^2}{2} + \frac{\text{Gmbias}\,Ts^4}{8}, \text{Gmacc}\,Ts + \frac{\text{Gmbias}\,Ts^3}{3}, -\frac{\text{Gmbias}\,Ts^2}{2}\right\},$$
$$\left.\left\{-\frac{\text{Gmbias}\,Ts^3}{6}, -\frac{\text{Gmbias}\,Ts^2}{2}, \text{Gmbias}\,Ts\right\}\right\}$$

```
In[65]:= H = {{1, 0, 0}}

Out[65]= {{1, 0, 0}}

In[66]:= R = sgmAlt^2
```

Out[66]= sgmAlt^2

```
In[67]:= Xpred  = Transpose[{{hPred, hDotPred, BiasPred}}]

Out[67]= {{hPred}, {hDotPred}, {BiasPred}}

In[68]:= P = {{P11, P12, P13}, {P21, P22, P23},
          {P31, P32, P33}}

Out[68]= {{P11, P12, P13}, {P21, P22, P23}, {P31, P32, P33}}

In[69]:= (*XestNew =*)
         Simplify[Xpred + P.Transpose[H].Inverse[H.P.Transpose[H] + R].(altMeas - H.Xpred)]
```

Out[69]=
$$\left\{\left\{\frac{\text{altMeas P11} + \text{hPred sgmAlt}^2}{\text{P11} + \text{sgmAlt}^2}\right\},\right.$$
$$\left.\left\{\text{hDotPred} + \frac{(\text{altMeas} - \text{hPred})\ \text{P21}}{\text{P11} + \text{sgmAlt}^2}\right\}, \left\{\text{BiasPred} + \frac{(\text{altMeas} - \text{hPred})\ \text{P31}}{\text{P11} + \text{sgmAlt}^2}\right\}\right\}$$

```
In[70]:= XestNew  = Transpose[{{hEstNew, hDotEstNew, BiasEstNew}}]

Out[70]= {{hEstNew}, {hDotEstNew}, {BiasEstNew}}
```

```
In[71]:= XpredNew = Simplify[Phik.XestNew + Gk * accMeas]
```

Out[71]=

$$\left\{\left\{\mathrm{hEstNew} + \frac{1}{2}\,\mathrm{Ts}\,(2\,\mathrm{hDotEstNew} + (\mathrm{accMeas} - \mathrm{BiasEstNew})\,\mathrm{Ts})\right\},\ \{\mathrm{hDotEstNew} + (\mathrm{accMeas} - \mathrm{BiasEstNew})\,\mathrm{Ts}\},\ \{\mathrm{BiasEstNew}\}\right\}$$

```
In[72]:= Pnew =
          Simplify[Phik.((IdentityMatrix[3] - P.Transpose[H].Inverse[H.P.Transpose[H] + R].H).P).
            Transpose[Phik] + Qk]
```

Out[72]=

$$\begin{aligned}
\Bigg\{\Bigg\{ & \frac{1}{60\,(\mathrm{P11} + \mathrm{sgmAlt}^2)} \Big(\mathrm{P11}\,\big(60\,\mathrm{sgmAlt}^2 + \mathrm{Ts}^2\,\big(60\,\mathrm{P22} + \mathrm{Ts}\,\big(20\,\mathrm{Gmacc} - 30\,\mathrm{P23} - 30\,\mathrm{P32} + 15\,\mathrm{P33}\,\mathrm{Ts} + 3\,\mathrm{Gmbias}\,\mathrm{Ts}^2\big)\big)\big) + \\
& \quad \mathrm{Ts}\,\big(30\,\mathrm{P21}\,\big(2\,\mathrm{sgmAlt}^2 + \mathrm{P13}\,\mathrm{Ts}^2\big) + \\
& \qquad 30\,\mathrm{P12}\,\big(2\,\mathrm{sgmAlt}^2 + \mathrm{Ts}\,(-2\,\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts})\big) + \mathrm{Ts}\,\big(-15\,\mathrm{P13}\,\big(2\,\mathrm{sgmAlt}^2 + \mathrm{P31}\,\mathrm{Ts}^2\big) + \\
& \qquad\quad \mathrm{sgmAlt}^2\,\big(60\,\mathrm{P22} - 30\,\mathrm{P31} + \mathrm{Ts}\,\big(20\,\mathrm{Gmacc} - 30\,\mathrm{P23} - 30\,\mathrm{P32} + 15\,\mathrm{P33}\,\mathrm{Ts} + 3\,\mathrm{Gmbias}\,\mathrm{Ts}^2\big)\big)\big)\big)\Big), \\
& \frac{1}{8\,(\mathrm{P11} + \mathrm{sgmAlt}^2)} \Big(4\,\mathrm{P12}\,\big(2\,\mathrm{sgmAlt}^2 + \mathrm{Ts}\,(-2\,\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts})\big) + \\
& \quad \mathrm{Ts}\,\big(-4\,\mathrm{P13}\,\big(2\,\mathrm{sgmAlt}^2 + \mathrm{Ts}\,(-2\,\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts})\big) + \\
& \qquad \mathrm{P11}\,\big(8\,\mathrm{P22} + \mathrm{Ts}\,\big(4\,\mathrm{Gmacc} - 8\,\mathrm{P23} - 4\,\mathrm{P32} + 4\,\mathrm{P33}\,\mathrm{Ts} + \mathrm{Gmbias}\,\mathrm{Ts}^2\big)\big) + \\
& \qquad \mathrm{sgmAlt}^2\,\big(8\,\mathrm{P22} + \mathrm{Ts}\,\big(4\,\mathrm{Gmacc} - 8\,\mathrm{P23} - 4\,\mathrm{P32} + 4\,\mathrm{P33}\,\mathrm{Ts} + \mathrm{Gmbias}\,\mathrm{Ts}^2\big)\big)\big)\Big), \\
& \frac{(\mathrm{P11} + \mathrm{sgmAlt}^2)\,\mathrm{Ts}\,(6\,\mathrm{P23} - \mathrm{Ts}\,(3\,\mathrm{P33} + \mathrm{Gmbias}\,\mathrm{Ts})) + 3\,\mathrm{P13}\,\big(2\,\mathrm{sgmAlt}^2 + \mathrm{Ts}\,(-2\,\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts})\big)}{6\,(\mathrm{P11} + \mathrm{sgmAlt}^2)}\Bigg\}, \\
\Bigg\{ & \mathrm{P21} - \frac{\mathrm{P11}\,\mathrm{P21}}{\mathrm{P11} + \mathrm{sgmAlt}^2} - \frac{\mathrm{P31}\,\mathrm{sgmAlt}^2\,\mathrm{Ts}}{\mathrm{P11} + \mathrm{sgmAlt}^2} + \frac{\mathrm{Gmacc}\,\mathrm{Ts}^2}{2} + \frac{\mathrm{Gmbias}\,\mathrm{Ts}^4}{8} + \\
& \quad \frac{\mathrm{Ts}\,\big(\mathrm{P12}\,(-\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts}) + (\mathrm{P11} + \mathrm{sgmAlt}^2)\,(\mathrm{P22} - \mathrm{P32}\,\mathrm{Ts})\big)}{\mathrm{P11} + \mathrm{sgmAlt}^2} - \\
& \quad \frac{\mathrm{Ts}^2\,\big(\mathrm{P13}\,(-\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts}) + (\mathrm{P11} + \mathrm{sgmAlt}^2)\,(\mathrm{P23} - \mathrm{P33}\,\mathrm{Ts})\big)}{2\,(\mathrm{P11} + \mathrm{sgmAlt}^2)}, \\
& \mathrm{P22} + \frac{\mathrm{P12}\,(-\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts})}{\mathrm{P11} + \mathrm{sgmAlt}^2} + \\
& \quad \frac{1}{3}\,\mathrm{Ts}\,\left(3\,\mathrm{Gmacc} - 3\,\mathrm{P23} - 3\,\mathrm{P32} + \frac{3\,\mathrm{P13}\,\mathrm{P21}}{\mathrm{P11} + \mathrm{sgmAlt}^2} + 3\,\mathrm{P33}\,\mathrm{Ts} - \frac{3\,\mathrm{P13}\,\mathrm{P31}\,\mathrm{Ts}}{\mathrm{P11} + \mathrm{sgmAlt}^2} + \mathrm{Gmbias}\,\mathrm{Ts}^2\right), \\
& \mathrm{P23} - \frac{1}{2}\,\mathrm{Ts}\,(2\,\mathrm{P33} + \mathrm{Gmbias}\,\mathrm{Ts}) + \frac{\mathrm{P13}\,(-\mathrm{P21} + \mathrm{P31}\,\mathrm{Ts})}{\mathrm{P11} + \mathrm{sgmAlt}^2}\Bigg\}, \\
\Bigg\{ & \frac{(\mathrm{P11} + \mathrm{sgmAlt}^2)\,\mathrm{Ts}\,(6\,\mathrm{P32} - \mathrm{Ts}\,(3\,\mathrm{P33} + \mathrm{Gmbias}\,\mathrm{Ts})) + 3\,\mathrm{P31}\,\big(2\,\mathrm{sgmAlt}^2 + \mathrm{Ts}\,(-2\,\mathrm{P12} + \mathrm{P13}\,\mathrm{Ts})\big)}{6\,(\mathrm{P11} + \mathrm{sgmAlt}^2)}, \\
& \mathrm{P32} - \frac{\mathrm{P12}\,\mathrm{P31}}{\mathrm{P11} + \mathrm{sgmAlt}^2} - \mathrm{P33}\,\mathrm{Ts} + \frac{\mathrm{P13}\,\mathrm{P31}\,\mathrm{Ts}}{\mathrm{P11} + \mathrm{sgmAlt}^2} - \frac{\mathrm{Gmbias}\,\mathrm{Ts}^2}{2}, \\
& \mathrm{P33} - \frac{\mathrm{P13}\,\mathrm{P31}}{\mathrm{P11} + \mathrm{sgmAlt}^2} + \mathrm{Gmbias}\,\mathrm{Ts}\Bigg\}\Bigg\}
\end{aligned}$$