Real-Time Parameter Estimation for Mini Aerial Vehicles using Low-Cost Hardware

Von der Fakultät für Maschinenwesen der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Andreas Gäb

Berichter: Univ.-Prof. Dr.-Ing. W. Alles
          Univ.-Prof. Dr.-Ing. D. Moormann


Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.
## Contents

1 Introduction
   1.1 State of technology ........................................ 5
      1.1.1 Mini Aerial Vehicles (MAV) ............................ 5
      1.1.2 Parameter estimation .................................. 5
      1.1.3 Onboard hardware for MAV applications .......... 6
      1.1.4 Low-cost sensors ....................................... 6
      1.1.5 Flow angle sensors .................................... 8
   1.2 Scope of this work ......................................... 8

2 Mathematical background ................................. 11
   2.1 The Kalman filter ....................................... 11
      2.1.1 System model ........................................ 11
      2.1.2 Filter algorithm ................................真 12
      2.1.3 Parameter estimation with the Kalman filter ... 13
   2.2 Simplifications for aerodynamic parameter estimation 14
      2.2.1 Simplifications for constant states ................. 14
      2.2.2 Simplifications for coefficient-based aerodynamics model ... 14
      2.2.3 The simplified filter ................................. 15
      2.2.4 Reformulation for computational efficiency .......... 16
      2.2.5 Disabling parameters ................................ 17
   2.3 Observability ........................................... 17
   2.4 Reconstruction of aerodynamic coefficients from inertial measurements 18
      2.4.1 Calculation of total forces and moments ........... 18
      2.4.2 Non-aerodynamic effects ............................. 19
      2.4.3 Normalization ........................................ 19
   2.5 Comparison of time histories ............................ 19

3 Demonstrator aircraft ................................. 21
   3.1 Concept .................................................. 21
   3.2 Onboard computer ......................................... 22
      3.2.1 Main processor ....................................... 22
      3.2.2 Control laws .......................................... 23
      3.2.3 Control modes ........................................ 25
   3.3 Sensors ................................................ 25
      3.3.1 Air data probe ....................................... 25
      3.3.2 Integrated navigation system (INS) ............... 29
      3.3.3 Infrared attitude sensors ............................ 29
# List of Figures

3.1 Components of the demonstrator aircraft system .................................. 22  
3.2 The demonstrator aircraft on the field between flights ............................ 23  
3.3 Block diagram of the lateral control laws ............................................ 24  
3.4 Front view of the five hole probe ..................................................... 26  
3.5 Five hole probe calibration setup .................................................... 27  
3.6 Data flow ......................................................................................... 33  

4.1 Half model testing .............................................................................. 35  
4.2 Baseline $C_L$ and polar of the clean configuration at $V_A = 17.5 \text{ m/s}$ .... 36  
4.3 $C_L$ increment due to throttle setting ............................................... 37  
4.4 $C_L$ increment due to elevator deflection ........................................... 37  
4.5 $C_L$ increment due to right aileron deflection ..................................... 38  

5.1 Trajectory of the simulation flight plan ............................................... 50  
5.2 Inflow characteristics of the simulated flight ....................................... 50  
5.3 Angular rates of the simulated flight .................................................. 51  
5.4 Control settings of the simulated flight .............................................. 51  
5.5 $C_D$ buildup ...................................................................................... 54  
5.6 $C_L$ derivatives, true inputs ............................................................. 55  
5.7 $C_n$ derivatives, true inputs ............................................................. 56  
5.8 PID results for $C_L$ ........................................................................... 58  
5.9 PID results for $C_m$ ........................................................................... 58  
5.10 PID results for $C_D$ ......................................................................... 59  
5.11 PID results for $C_C$ ......................................................................... 59  
5.12 PID results for $C_l$ .......................................................................... 60  
5.13 PID results for $C_n$ .......................................................................... 60  

6.1 The demonstrator aircraft during final approach .................................... 63  
6.2 Scatter plot of lift coefficient ................................................................ 66  
6.3 Post-flight EKF results: Lift coefficient derivatives ............................. 70  
6.4 Post-flight EKF results: Drag coefficient derivatives ......................... 71  
6.5 Post-flight EKF results: Pitching moment coefficient derivatives ....... 72  
6.6 Post-flight EKF results: Crosswind force coefficient derivatives ....... 73  
6.7 Post-flight EKF results: Rolling moment coefficient derivatives ......... 74  
6.8 Post-flight EKF results: Yawing moment coefficient derivatives ......... 75  
6.9 Lift coefficient derivatives .................................................................. 82  
6.10 Drag coefficient derivatives .............................................................. 82  
6.11 Pitching moment coefficient derivatives ........................................... 83  
6.12 Crosswind force coefficient derivatives ............................................ 83
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.13</td>
<td>Rolling moment coefficient derivatives</td>
<td>84</td>
</tr>
<tr>
<td>6.14</td>
<td>Yawing moment coefficient derivatives</td>
<td>84</td>
</tr>
<tr>
<td>B.1</td>
<td>Views of the demonstrator aircraft in XFLR5 (original fuselage)</td>
<td>95</td>
</tr>
<tr>
<td>C.1</td>
<td>$C_L$ buildup</td>
<td>99</td>
</tr>
<tr>
<td>C.2</td>
<td>$C_m$ buildup</td>
<td>100</td>
</tr>
<tr>
<td>C.3</td>
<td>$C_C$ buildup</td>
<td>100</td>
</tr>
<tr>
<td>C.4</td>
<td>$C_l$ buildup</td>
<td>101</td>
</tr>
<tr>
<td>C.5</td>
<td>$C_n$ buildup</td>
<td>101</td>
</tr>
<tr>
<td>D.1</td>
<td>Inflow data</td>
<td>103</td>
</tr>
<tr>
<td>D.2</td>
<td>Control settings</td>
<td>104</td>
</tr>
<tr>
<td>D.3</td>
<td>Angular rates</td>
<td>105</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Execution times of PID sub-tasks ............................................ 32
3.1 Modeled aspects in the consolidated nonlinear aerodynamics model . . 41
4.2 Reference values for aerodynamic model .................................... 43
4.3 Eigenmotions ........................................................................... 44
4.1 Correlation coefficients of PID inputs ....................................... 52
5.2 Noise standard deviations \( \sigma \) in simulation runs ......................... 53
5.3 Contribution RMS values .......................................................... 54
5.4 RAAEs (in percent) of derivatives in simulation run ..................... 62
6.1 Standard data link configuration .................................................. 64
6.2 Data link message contents for high update rate ......................... 64
6.3 Noise standard deviations of measured values ............................. 65
6.4 Longitudinal derivatives identified from full rate data .................. 67
6.5 Lateral derivatives identified from full rate data ........................... 67
6.6 Flight measured correlation coefficients of PID inputs .................. 68
6.7 Allowable value ranges ............................................................. 77
6.8 Standard data link configuration .................................................. 96
6.9 Control surface deflections ........................................................ 96
6.10 Reference values for inertial data .............................................. 96
6.11 1\( \sigma \) sensor noise levels in static test conditions ....................... 98
E.1 Summary of PID results for all analyses of this thesis – longitudinal motion 108
E.2 Summary of PID results for all analyses of this thesis – lateral motion 109
## Nomenclature

### Latin Letters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>System matrix</td>
<td>var</td>
<td>$n_x \times n_x$</td>
</tr>
<tr>
<td>B</td>
<td>Input matrix</td>
<td>var</td>
<td>$n_x \times n_u$</td>
</tr>
<tr>
<td>b</td>
<td>Wingspan</td>
<td>m</td>
<td>scalar</td>
</tr>
<tr>
<td>$\bar{c}$</td>
<td>Reference chord</td>
<td>m</td>
<td>scalar</td>
</tr>
<tr>
<td>C</td>
<td>Output matrix</td>
<td>var</td>
<td>$n_y \times n_x$</td>
</tr>
<tr>
<td>$\bar{C}_{(i)}$</td>
<td>Aerodynamic coefficient</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>D</td>
<td>Throughput matrix</td>
<td>var</td>
<td>$n_y \times n_u$</td>
</tr>
<tr>
<td>d</td>
<td>Dimensionless damping ratio</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>e</td>
<td>Oswald efficiency factor</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>F</td>
<td>Process noise distribution matrix</td>
<td>var</td>
<td>$n_x \times n_x$</td>
</tr>
<tr>
<td>$\bar{G}$</td>
<td>Measurement noise distribution matrix</td>
<td>var</td>
<td>$n_y \times n_x$</td>
</tr>
<tr>
<td>I$_{(i)}$</td>
<td>Component of the moment of inertia tensor</td>
<td>kg m$^2$</td>
<td>scalar</td>
</tr>
<tr>
<td>K</td>
<td>Kalman gain matrix</td>
<td>var</td>
<td>$n_x \times n_y$</td>
</tr>
<tr>
<td>K</td>
<td>Transfer function gain</td>
<td>var</td>
<td>scalar</td>
</tr>
<tr>
<td>k$_{(i)}$</td>
<td>Calibration factor</td>
<td>var</td>
<td>scalar</td>
</tr>
<tr>
<td>L</td>
<td>Rolling moment</td>
<td>Nm</td>
<td>scalar</td>
</tr>
<tr>
<td>M</td>
<td>Acting external moment</td>
<td>Nm</td>
<td>3</td>
</tr>
<tr>
<td>M</td>
<td>Pitching moment</td>
<td>Nm</td>
<td>scalar</td>
</tr>
<tr>
<td>N</td>
<td>Dimension of matrix</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>N</td>
<td>Yawing moment</td>
<td>Nm</td>
<td>scalar</td>
</tr>
<tr>
<td>n$_{(i)}$</td>
<td>Number of elements of a vector</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>O</td>
<td>Observability matrix</td>
<td>var</td>
<td>$n_y \times n_x n_y$</td>
</tr>
<tr>
<td>p</td>
<td>Pressure</td>
<td>Pa</td>
<td>scalar</td>
</tr>
<tr>
<td>Q</td>
<td>Process noise covariance matrix</td>
<td>var</td>
<td>$n_x \times n_x$</td>
</tr>
<tr>
<td>$\bar{R}$</td>
<td>Measurement noise covariance matrix</td>
<td>var</td>
<td>$n_y \times n_y$</td>
</tr>
<tr>
<td>r</td>
<td>Diagonal element of $R$ matrix</td>
<td>var</td>
<td>scalar</td>
</tr>
<tr>
<td>R$^2$</td>
<td>Coefficient of determination</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>$R_{m}$</td>
<td>Specific gas constant for air</td>
<td>J/kg/K</td>
<td>scalar</td>
</tr>
<tr>
<td>S</td>
<td>Reference area</td>
<td>m$^2$</td>
<td>scalar</td>
</tr>
<tr>
<td>s</td>
<td>Laplace variable</td>
<td>s$^{-1}$</td>
<td>scalar</td>
</tr>
<tr>
<td>s</td>
<td>Semispan</td>
<td>m</td>
<td>scalar</td>
</tr>
<tr>
<td>t</td>
<td>Time</td>
<td>s</td>
<td>scalar</td>
</tr>
</tbody>
</table>

*continued on next page*
### Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_d$</td>
<td>Dead time</td>
<td>s</td>
<td>scalar</td>
</tr>
<tr>
<td>$T_R$</td>
<td>Rolling motion time constant</td>
<td>s</td>
<td>scalar</td>
</tr>
<tr>
<td>$t_{PWM}$</td>
<td>PWM duty cycle</td>
<td>s</td>
<td>scalar</td>
</tr>
<tr>
<td>$T_{ref}$</td>
<td>Reference temperature for air data calculation</td>
<td>K</td>
<td>scalar</td>
</tr>
<tr>
<td>$u$</td>
<td>System inputs</td>
<td>var</td>
<td>$n_u$</td>
</tr>
<tr>
<td>$v$</td>
<td>Measurement noise</td>
<td>var</td>
<td>$n_y$</td>
</tr>
<tr>
<td>$v$</td>
<td>Velocity</td>
<td>m/s</td>
<td>3</td>
</tr>
<tr>
<td>$V_A$</td>
<td>Airspeed</td>
<td>m/s</td>
<td>scalar</td>
</tr>
<tr>
<td>$w$</td>
<td>Process noise</td>
<td>var</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$x$</td>
<td>System states</td>
<td>var</td>
<td>$n_x$</td>
</tr>
<tr>
<td>$X$</td>
<td>Force along x axis</td>
<td>N</td>
<td>scalar</td>
</tr>
<tr>
<td>$y$</td>
<td>System outputs</td>
<td>var</td>
<td>$n_y$</td>
</tr>
<tr>
<td>$z$</td>
<td>Measurements</td>
<td>var</td>
<td>$n_y$</td>
</tr>
<tr>
<td>$Z$</td>
<td>Force along z axis</td>
<td>N</td>
<td>scalar</td>
</tr>
</tbody>
</table>

### Greek Letters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Angle of attack</td>
<td>rad</td>
<td>scalar</td>
</tr>
<tr>
<td>$\beta$</td>
<td>Angle of sideslip</td>
<td>rad</td>
<td>scalar</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>Time step size</td>
<td>s</td>
<td>scalar</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Control element deflection</td>
<td>var</td>
<td>scalar</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Symmetric aileron (flap) command</td>
<td>0...1</td>
<td>scalar</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>Aspect ratio</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Forgetting factor</td>
<td>–</td>
<td>scalar</td>
</tr>
<tr>
<td>$\omega_0$</td>
<td>Eigenfrequency</td>
<td>s$^{-1}$</td>
<td>scalar</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>State transition matrix</td>
<td>var</td>
<td>$n_x \times n_x$</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Air density</td>
<td>kg/m$^3$</td>
<td>scalar</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Standard deviation</td>
<td>var</td>
<td>scalar</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Delay time constant</td>
<td>s</td>
<td>scalar</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Filter innovation or residual</td>
<td>var</td>
<td>$n_y$</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Parameters</td>
<td>var</td>
<td>$n_\theta$</td>
</tr>
<tr>
<td>$\xi$</td>
<td>Aileron deflection angle</td>
<td>rad</td>
<td>scalar</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>Rudder deflection angle</td>
<td>rad</td>
<td>scalar</td>
</tr>
</tbody>
</table>

### Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D</td>
<td>Analog/digital</td>
</tr>
<tr>
<td>AoA</td>
<td>Angle of attack $\alpha$</td>
</tr>
<tr>
<td>CG</td>
<td>Center of gravity</td>
</tr>
<tr>
<td>EEM</td>
<td>Equation error method</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman filter</td>
</tr>
</tbody>
</table>

---

*continued on previous page*
**Nomenclature**

*continued from previous page*

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FHP</td>
<td>Five hole probe</td>
</tr>
<tr>
<td>FTR</td>
<td>Fourier transform regression</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter-integrated circuit, a two-wire serial data bus</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial measurement unit</td>
</tr>
<tr>
<td>INS</td>
<td>Integrated navigation system</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>ISA</td>
<td>International standard atmosphere, see [36]</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman filter</td>
</tr>
<tr>
<td>LLA</td>
<td>Geodetic position (latitude, longitude, altitude)</td>
</tr>
<tr>
<td>MAV</td>
<td>Micro (or mini) aerial vehicle</td>
</tr>
<tr>
<td>MEMS</td>
<td>Microelectromechanical systems</td>
</tr>
<tr>
<td>PID</td>
<td>Parameter identification</td>
</tr>
<tr>
<td>PPM</td>
<td>Pulse position modulation</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse width modulation</td>
</tr>
<tr>
<td>R/C</td>
<td>Radio control</td>
</tr>
<tr>
<td>RAAE</td>
<td>Relative area of absolute error</td>
</tr>
<tr>
<td>RLS</td>
<td>Recursive least squares</td>
</tr>
<tr>
<td>RMS</td>
<td>Root mean square</td>
</tr>
<tr>
<td>RS232</td>
<td>Serial port interface</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned aerial vehicle</td>
</tr>
<tr>
<td>UTC</td>
<td>Coordinated universal time</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible markup language</td>
</tr>
</tbody>
</table>

**Superscripts**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>in aerodynamic coordinate system (“wind axes”)</td>
</tr>
<tr>
<td>$F$</td>
<td>caused by propulsion system</td>
</tr>
<tr>
<td>$f$</td>
<td>in body-fixed coordinate system</td>
</tr>
<tr>
<td>$k$</td>
<td>Time step number</td>
</tr>
<tr>
<td>*</td>
<td>Normalized angular rate</td>
</tr>
</tbody>
</table>

**Subscripts**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>aerodynamic (vehicle relative to air)</td>
</tr>
<tr>
<td>$a$</td>
<td>Aileron</td>
</tr>
<tr>
<td>$a, sym$</td>
<td>Symmetric aileron</td>
</tr>
<tr>
<td>$C$</td>
<td>Commanded value</td>
</tr>
<tr>
<td>$C$</td>
<td>Crosswind force (sideforce in wind axes)</td>
</tr>
<tr>
<td>$D$</td>
<td>Drag</td>
</tr>
<tr>
<td>$K$</td>
<td>kinematic (vehicle relative to inertial system)</td>
</tr>
<tr>
<td>$L$</td>
<td>Lift</td>
</tr>
<tr>
<td>$l$</td>
<td>Rolling moment</td>
</tr>
</tbody>
</table>

*continued on next page*
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>Pitching moment</td>
</tr>
<tr>
<td>$n$</td>
<td>Yawing moment</td>
</tr>
<tr>
<td>$r$</td>
<td>Rudder</td>
</tr>
<tr>
<td>$sen$</td>
<td>at sensor location</td>
</tr>
<tr>
<td>$t$</td>
<td>Throttle setting</td>
</tr>
<tr>
<td>$0$</td>
<td>Constant contribution, bias</td>
</tr>
<tr>
<td>$0$</td>
<td>Initial value</td>
</tr>
</tbody>
</table>

Choice and nomenclature of flight mechanical variables generally follow the German norm [13] with the indices of the coefficients changed to English language versions (Widerstandsbeiwert $C_W$ becomes drag coefficient $C_D$, Querkraftbeiwert $C_Q$ becomes crosswind force coefficient $C_C$ etc.). To be able to use the results of the calculation programs from section 4.2, the lateral reference length is chosen to be the full wingspan instead of the semispan. This also affects the dimensionless angular rates, whose definition is given in eq. 4.10.
Abstract

This work describes the design and implementation of a real-time aerodynamic parameter estimation algorithm on a small remotely piloted aircraft.

The Extended Kalman Filter is adapted for aerodynamic parameter estimation. A formulation is given which is similar to the recursive least squares (RLS) algorithm but uses noise covariances instead of a forgetting factor for tuning. Optimization for low computing power hardware is discussed. A demonstrator aircraft based on an R/C model is equipped with the required hardware (air and inertial data sensors, onboard processor, telemetry). Wind tunnel tests and calculations produce a reference data set for aerodynamics and propulsion of this aircraft, which is then used for simulation. This simulation allows to prove the performance of the parameter identification (PID) algorithm and predict the set of parameters which is identifiable with the given hardware. The main influence on identifiability is the relative contribution of a derivative to a coefficient in relation to the output noise level. Correlation issues are identified which arise because of the very fast rolling motion (and somewhat less the pitching motion) in comparison to the achievable update rates. Two sorts of flight test results are presented: post-flight analyses of logged flight data and identified parameters from the working real-time algorithm. Although some minor derivatives are not identifiable, the results prove the general feasibility of the approach.
1 Introduction

Mini or micro unmanned aerial vehicles (MAV) are in use for an ever increasing number of
tasks, ranging from hobby flying over aerial photography to telecommunication platforms.
The term micro UAV usually refers to vehicles with much less than 1 m of maximum
dimension, while mini UAV may reach 2 m and more of wingspan. Contrary to larger
UAV, they are almost exclusively powered electrically. Although manual control or at
least supervision by a pilot is mandatory for most applications, UAV are characterized
by their capability to operate autonomously at least to a certain level. Therefore they
need an onboard device performing control tasks. This device is usually called autopilot
and in most cases consists of an embedded microprocessor with corresponding peripheral
hardware like sensors, data link, etc.

Miniaturization is still an ongoing trend in electronics. Especially hardware designed for
application in embedded form, i.e. microprocessors on small printed circuit boards, is
still taking big jumps in terms of computing power while at the same time shrinking in
size. The same goes for sensors, e.g. pressure transducers, which are also available with
their own analog/digital converters and connectors to standard bus systems like I2C
or CAN. Microelectromechanical systems (MEMS) constituted a big leap in this area.
Prices for all these hardware parts keep lowering. Another important aspect of modern
hardware is its reduced energy consumption, allowing for smaller energy sources and
alleviating cooling problems.

Nevertheless, this kind of hardware poses its own problems. Although the gap is
shrinking, embedded processors are still not comparable to desktop processors in speed,
memory or storage size. For high computational demands, the lack of a dedicated
floating point unit is often a limiting factor. Smaller sensors have increased noise levels
and systematic errors compared to larger ones. These performance criteria must be kept
in mind when selecting the hardware for an application.

In MAV, small size and especially low weight are mandatory requirements for hardware
to be included onboard. Due to the low energy density of batteries in comparison to
liquid fuels, MAV endurances are very small in comparison to larger aircraft. Higher
weight leads to an increase in drag and thus to a direct penalty in endurance, as does
higher energy consumption of other devices than the propulsion, at least in small vehicles
that only have a single battery as available source of power for all components. If one
wants to increase the endurance of a MAV, small, lightweight and energy-economical
hardware is required. This still holds when techniques for energy extraction from the
environment are applied, like e.g. solar cells, thermal soaring or gust energy extraction.
Besides, the low prices are also often critical to enable such projects for universities, small businesses or private individuals.

Increasing endurance is a common goal. Recent attempts [7] aim for 200 km over open sea. In such circumstances, monitoring of the aircraft’s flying characteristics becomes interesting. Although it is not to be expected that the aircraft may enter icing conditions or suffer battle damage, changes in the aerodynamics may still occur. Surface degradation due to salty air comes to mind for overseas flights, but also failures of control rods because of prolonged vibrations or reduced propeller slipstream as battery voltage decreases can have an effect of the aircraft’s flying qualities.

Detecting such changes in real time and possibly reacting to them – by adapting autopilot controller gains or even changing the mission – requires a constant monitoring of the aerodynamics. Although an operator at the ground station may diagnose certain performance losses from criteria like ground speed and trajectory, an onboard real time parameter estimation algorithm can provide significantly more information on the type and severity of change that has occurred. Such an algorithm can provide direct estimates of aerodynamical derivatives – values that are crucial for the design of the autopilot and may directly be drawn on to adapt controller gains.

The input data that needs to be provided for aerodynamic parameter estimation consists of inertial data, air data and control surface deflections. Inertial data is often available anyway, when integrated navigation systems (INS) or at least inertial measurement units (IMU) are used to provide position and/or angular rates for the damping cascade of the autopilot. A pitot tube and static ports to measure airspeed are not uncommon in MAV applications either; for the task at hand, however, the flow angles need to be known as well. This can be achieved by means of wind vanes or by a multi- (typically five) hole probe, see section 1.1.5 below.

Control surface deflections are usually measured by external potentiometers if required. Internal sensors require installation space in wings or tails, which is difficult to provide in MAV, and low-cost actuators usually do not have an accessible internal position feedback. Given an actuator model, it is also possible to estimate them from the commanded values which are available in the autopilot, thus eliminating the need for an extra set of sensors, albeit for similar quality a very sophisticated actuator model, possibly including air loads, is required.

The flow angle sensor provides additional measured quantities, which may suggest to utilize these to compensate errors of existing sensors and create some kind of analytical redundancy, i.e. perform a flight path reconstruction like in [27] or chapter 10 of [23]. However, these techniques assume that the aircraft moves in still air or at least wind velocities are negligible compared to inertial velocities. This is not at all the case of for MAV, which operate at low airspeeds and altitudes where the atmospheric boundary layer causes complex three-dimensional wind flow patterns the speed of which may reach a significant fraction of the aircraft’s cruise speed or even exceed it. Thus, the aerodynamic and kinematic velocities are in general independent quantities and no
redundancy exists in the system that could be exploited analytically. (The analytical redundancy between integration of inertial measurements and external position fixes, which is used by integrated navigation systems to estimate sensor biases, is of course usable in MAV as well.)

1.1 State of technology

1.1.1 Mini Aerial Vehicles (MAV)

The technology for building both mini and micro aerial vehicles can be considered as more or less established, at least for fixed-wing configurations. A good hint for this are the agendas and proceedings for conferences like Spring IMAV 2011 (see springimav2011.org) or (Fall) IMAV 2011 (imav2011.org). Papers on fixed-wing MAV have become scarce, focusing on unconventional layouts. Topics more at today’s center of attention are e.g. flapping wings and – including projects here at the Institute [39] – hybrid configurations capable of both horizontal flight and vertical take-off and landing (VTOL). Onboard hardware, especially new sensor systems like video image processing [12], is also of interest. So are multi-rotor configurations, despite the abundance of quadrocopters on the market. Sporadically, miniaturization is driven further to produce even nano aerial vehicles with maximum dimensions below 7.5 cm.

1.1.2 Parameter estimation

Parameter estimation, as a special aspect of system identification, has been an important subarea of mathematics for a long time. Its origins can be traced back at least to C.-F. Gauss’s determination of the orbit of the asteroid Ceres in 1801 [14] which he documented in his 1809 book Theoria Motus [18]. Since then, the principle has found wide-spread use in an large number of topics ranging from economics to biology. First applications in flight mechanics date back to 1919, when Glauert deduced characteristics of the phugoid motion from airspeed recordings [19]. With the advent of digital computing, parameter estimation methodology advanced significantly wherever it was applied. Today, parameter estimation is a standard procedure of data analysis in virtually every flight test of manned aircraft, perhaps with the exception of microlights and homebuilts. The method of choice is post-flight analysis using the output error method. Recent monographies on parameter estimation of aircraft are the books by Jategaonkar [23] and Klein and Morelli [26].

These books also cover recursive methods suitable for real-time applications. Several of these methods have already been tested in flight, see e.g. [28]. More recent experiments at NASA Langley Research Center [35] included already flight tests with a subscale model [24], however, the hardware in use for that was rather sophisticated than low-cost. The GARTEUR flight mechanics action group FM/AG-16 on Fault Tolerant Control stimulated some research on real-time parameter estimation methods as a possibility
to avoid loss-of-control accidents of transport aircraft. At TU Delft, experiments were conducted using a manned flight simulator [29]. In general, real-time parameter estimation in flight mechanics has not yet found widespread industrial application and remains a subject of research.

The terms parameter estimation and parameter identification are used somewhat interchangeably, as they refer to the same subject. In contrast to system identification, where the form of the mathematical model to describe a physical system is also still unknown, in parameter estimation, this form is known. The model consists of a set of equations containing a set of variables (the parameters), the values of which are sought. The frequent use of “parameter identification” instead of the technically more correct “parameter estimation” is probably due to the handiness of the abbreviation PID.

1.1.3 Onboard hardware for MAV applications

The PID projects cited above were not targeted for use in very small aircraft and thus did not face the severe limitations listed in the introduction above. In this case, the main processing may be done on laptop PC hardware, provided that suitable interfaces to the sensors are available. To reach the levels of integration required for MAV applications, embedded microprocessors are unavoidable. The bare processor chip however is of little use to the experimenter; it needs an interface to its periphery, usually in form of jacks or the like which are assembled on a printed circuit board.

The design of such a board is a rather ambitious project, even for someone with a lot of experience in electrical engineering, but fortunately there exists a variety of publicly available boards which can be purchased at relatively inexpensive prices. Some are specific to multi- copter operation, but there are also fixed-wing variants. A survey of commercially available autopilot boards is given in [9]. This source does however not include the Arduino-based systems like ArduPlane [3] or FlyDuino [1], which are based on 16 MHz Atmega processors. The selection of Paparazzi [40] boards has also evolved since, with the arrival of the Tiny 2.11, Lisa/L and Lisa/M, the latter based on 72 MHz STM32 processors.

The softwares for FlyDuino, ArduPlane and Paparazzi (onboard and ground station components) are publicly available under open source licenses. All use cascaded proportional-integral-derivative controller loops for attitude control, the latter two also for waypoint navigation and automatic take-off and landing. Paparazzi also allows special maneuvers like circling a waypoint, survey patterns and the like.

1.1.4 Low-cost sensors

Fortunately, the quality of low cost sensors has been increasing steadily. However, there is still a trade-off compared to higher end products. As an example, consider the analog/digital converters. 16-bit converters are mainly used in expensive sensors.
They provide a resolution of $2^{16} = 65536$ different steps. For a pressure transducer with a range from 0 to 1000 Pa, adequate as sensor for dynamic pressure in an MAV, this gives a theoretical resolution of 0.015 Pa per digital step.\(^1\) A low cost sensor, on the other hand, might only have a resolution of 12 bit, corresponding to $2^{12} = 4096$ different steps. The theoretical resolution shrinks to 0.24 Pa per step, or, to put it another way, quantization noise increases. However, not long ago the typical resolution of a low-cost A/D converter was only 10 bit, and built-in converters of many microprocessors still do not feature a higher one.

To make use of a sensor’s internal converter, it must be connected to the main processor with a suitable interface. A variety of protocols exists, of which one must be chosen that is supported by sensor and processor (both digitally and in terms of voltage levels) and allows a sufficient data rate. If multiple sensors are to be used, they must be uniquely addressable.

Another aspect that may need to be considered is temperature stability. Low cost sensors will typically have a bias that strongly varies with temperature. Users will have to add their own temperature sensor, calibration model and correction code if this needs to be compensated for.

Depending on the kind of sensor, low cost hardware may also use a sensing element based on a different physical phenomenon, which in turn may experience more or different systematic errors. Low cost angular rate sensors for example may be built without any spinning part but rather use MEMS vibratory gimbals ([15], chapter 4.4). While reducing price, weight and overall size considerably compared to the spinning gyro version, noise is increased vastly and other sources of vibration (including adjacent sensors for other axes of rotation) may introduce sensor cross-talk. Highly integrated inertial measurement units (IMUs) based on such MEMS angular rate sensors, accelerometers and possibly a magnetometer are commercially available for less than €100.

Small integrated navigation systems are usually based on MEMS components like described above. By combining them with temperature sensors and including a microprocessor capable of running a sophisticated calibration model along with the strapdown algorithm, many of the systematic errors can be eliminated. This does however require a considerable amount of insight into the sensors and their error behavior, which manufactures usually keep their business secret, so that buying an INS as purchased part will usually result in increased performance. But with the price of the INS an order of magnitude or two higher than that of the sensors alone, the appropriateness of the term “low cost” is questioned, although the price range for INS products stretches for further orders of magnitude.

\(^1\)Typically, the usable number of steps is smaller to avoid saturation or wraparound at the ends.
1.1.5 Flow angle sensors

The main sensor that has to be added to allow aerodynamic parameter estimation is a flow angle sensor. Two main concepts are in wide application: wind vanes or multi-hole probes. A special case of wind vanes is a probe head that aligns with the airflow. A force-sensing solution without moving parts [10] has not yet found widespread application and is covered by patents.

For use in an MAV, a five hole probe has several advantages over a set of wind vanes. Wind vanes are moving parts that are more prone to damage, especially when landings take place in grass or rough terrain without a landing gear, which is often the case for MAV. A five hole probe instead is very robust and can be built smaller and with less weight. The probe front hole can act as pitot port providing dynamic pressure to measure airspeed. Its sensing elements (the pressure transducers) do not have to be located at the probe but may be moved safely into the fuselage, further increasing crashworthiness. On the other hand, the main advantage of wind vanes is that their calibration is fairly simple (generally linear) and not dependent on airspeed or dynamic pressure.

An important difference between wind vanes and multi-hole probes is their dynamic behavior. Wind vanes are usually second-order oscillators with a low damping coefficient, which usually results in constant oscillations with the vane eigenfrequency. The pressure tubes and sensors of the multi-hole probes act as first order delays, so no oscillations are possible.

Common to all flow angle sensors is that they need to be placed in free stream to avoid errors caused by aircraft upwash or downwash, which is usually achieved by a nose or wingtip boom carrying them. The length of the boom should be at least one fuselage diameter resp. wing chord, preferably more. As the resulting sensor location will be away from the center of gravity, the measured flow angles will need to be corrected for velocities induced by rotation of the aircraft.

1.2 Scope of this work

This work discusses the realization of the real time parameter estimation task in flight test. It is limited to the identification of flight mechanical derivatives and does not include further steps like decision-making based on these results. A further goal is the assessment of the effects the low-cost hardware has on the quality of the results.

Several steps are taken to achieve these goals. Firstly, a suitable algorithm needs to be selected. The extended Kalman filter (EKF) is a very general algorithm for use in real-time parameter estimation. It includes recursive least squares (RLS) as a limiting case. This algorithm is discussed and adapted to the task at hand in chapter 2.
1.2 Scope of this work

To carry out the flight test, a demonstrator aircraft is required which must be equipped with all necessary hardware like onboard processor, sensors and so on. Chapter 3 describes the realization of this platform in detail. A commercially available R/C model aircraft serves as a basic airframe, into which additional hardware is installed. Low-cost items are bought as off-the-shelf purchased parts unless they were already available at the Institute, like the five hole probe. Special attention is paid to the calibration of this sensor, which thanks to the small size of the aircraft can take place in the final assembled configuration including the complete fuselage. Information on the onboard computer and the software to be run on it can also be found in this chapter.

An important step is simulation to evaluate and tune the PID algorithm. This is only possible if at least some of the aircraft’s characteristics are known beforehand from reliable sources. Chapter 4 comprises the generation of such a reference flight dynamics model. One source for the data are wind tunnel tests. The small aircraft size here again comes in handy, as it allows to test a half exemplar of the final vehicle at typical airs speeds, thus eliminating a varying Reynolds number as possible error source.

This kind of test can however only yield static longitudinal characteristics, so the reference model has to be augmented by other methods. Calculation programs are chosen for this task, especially the Digital Datcom which allows to produce dynamic and lateral characteristics based on the wind tunnel results. Complementing them with inertial data, a full non-linear flight dynamics data set can be achieved.

Tuning of the PID algorithm, but also of the autopilot control laws, can be a time-consuming and possibly risky task. It can however greatly be facilitated by using a simulation, which may also provide preliminary insight into the performance of both the algorithm and the aircraft. In the first parts of chapter 5, the implementation of such a simulation is described. Besides the airframe model – which is reduced to a linear one to simplify PID algorithm evaluation –, models of all involved components are required, focusing on the sensors and their expected significant error levels.

The last part of chapter 5 contains results of selected simulation runs and addresses some problems that surfaced already in simulation, namely identifiability of minor derivatives and correlations between control surface deflections and angular rates.

Flight test results are presented in chapter 6. Different analysis methods are applied, starting with the post-flight batch equation error method producing scalar values to compare to the reference model. The next step is to fine-tune the algorithm using input values produced in flight during offline runs on a desktop computer. The last part of this chapter then shows results which were produced by actual real-time parameter estimation onboard the demonstrator aircraft during flight.

Finally, chapter 7 rounds off this thesis with summary, conclusions and outlook.
2 Mathematical background

2.1 The Kalman filter

The Kalman filter (KF) is a mathematical tool for estimating the state of a dynamical system given a set of measurements and a mathematical model of the dynamical system. It operates in a recursive manner, making it suitable for real-time implementation. Both the system and the measurements may be subject to noise. The algorithm is divided into a prediction step, using the state equations of the system model and their derivatives, and a correction step, using the observation equations and their derivatives. The steps are sometimes also referred to as state update and measurement update, respectively.

Different formulations of the algorithm are available in the literature as it has a very widespread use. A derivation of the KF algorithm can be found at various places, see e.g. [23], appendix F or [21], chapter 4.2. Therefore, only the final equations of the algorithm are restated here. This work will follow the terminology from [23].

2.1.1 System model

The system to be considered for the simple Kalman filter is a multiple input, multiple output linear time-invariant dynamical system in continuous time. It is given by

\[
\dot{x}(t) = Ax(t) + Bu(t) + Fw(t) \quad (2.1)
\]
\[
y(t) = Cx(t) + Du(t) \quad (2.2)
\]
\[
z(t) = y(t) + Gv(t) \quad (2.3)
\]

where eq. 2.1 is called the system equation (or state equation), eq. 2.2 is the output equation and eq. 2.3 is called the measurement equation (or observation equation). The dimensions of the system are given by the number of states \(n_x\), the number of outputs or measurements \(n_y\) and the number of control inputs \(n_u\). The system is non-deterministic because it is subject to the noises \(v\) and \(w\). These noises are assumed to be white Gaussian. System states may possibly be hidden, only outputs are available for measurement.

In digital signal processing, the system equation is often formulated in discrete time. The discretised solution of eq. 2.1 is given by

\[
x^{k+1} = x^k + \int_{t_k}^{t_{k+1}} \dot{x} \, dt \quad (2.4)
\]
which can be expressed in terms of the matrices as

\[ x^{k+1} = \Phi x^{k} + \Psi B u^{k} + \Psi (\Delta t)^{-\frac{1}{2}} F w^{k} \]  

where the superscript \( k \) denotes time steps and \( \Phi \) is the discrete state transition matrix \( \Phi = e^{A \Delta t} \) (2.6)

and \( \Psi \) its integral

\[ \Psi = \int_{0}^{\Delta t} e^{A \tau} d\tau. \]  

(2.7)

This is useful for integrating the state equation during the prediction step of the Kalman Filter. However, it is also possible to approximate the integral by any numerical integration method, e.g. a simple Euler step:

\[ x^{k+1} \approx x^{k} + \Delta t \dot{x}^{k}. \]  

(2.8)

2.1.2 Filter algorithm

The filter algorithm yields estimates for the states \( x \), their accuracy (in form of the state error covariance matrix \( P \)) and the true outputs \( y \) given the measurements \( z \) and the inputs \( u \) (the latter assumed to be measured exactly). As it has to be implemented on a digital computer, a discretisation of the system eq. 2.1 with respect to time is necessary. In this work, the Euler approximation from eq. 2.8 will be used for this purpose. Still, for propagation of the covariance matrix, the transition matrix \( \Phi \) from eq. 2.6 is required, too.

The filter consists of the prediction step, where the estimation of the states and their error covariance are propagated to the next time step, and the correction step, where these variables are updated according to available measurements. Variables calculated during the prediction step are denoted by a tilde (e.g. \( \tilde{P} \)), while those calculated during the correction step wear a hat, e.g. \( \hat{P} \).

The prediction step is given by

\[ \tilde{x}^{k} = \tilde{x}^{k-1} + \Delta t (A \tilde{x}^{k-1} + B u^{k-1}) \]  

(2.9)

\[ \tilde{P}^{k} = \Phi \tilde{P}^{k-1} \Phi^T + Q \]  

(2.10)

and the correction step by

\[ \hat{y}^{k} = C \tilde{x}^{k} + D u^{k} \]  

(2.11)

\[ K^{k} = \tilde{P}^{k} C^T (C \tilde{P}^{k} C^T + R)^{-1} \]  

(2.12)

\[ \hat{x} = \tilde{x}^{k} + K^{k} (z^{k} - \hat{y}^{k}) \]  

(2.13)

\[ \hat{P}^{k} = (I - K^{k} C) \tilde{P}^{k} (I - K^{k} C)^T + K^{k} R (K^{k})^T \]  

(2.14)
where $Q = \Delta t F F^T$ and $R = G G^T$. Again, superscripts denote time steps. The difference between measurements and estimated outputs is usually called the residual or innovation

$$\nu^k = z^k - \hat{y}^k.$$  

(2.15)

The involved matrices $A$ to $G$ are assumed as known and constant in the basic formulation. Therefore, they have to be supplied to the algorithm, as well as an initial state vector $x_0$ and an initial state covariance matrix $P_0$. The matrix $K$ is called the Kalman gain.

In this formulation, eq. 2.14 is the so-called long or Joseph form of the covariance update. There is also a short form, which is computationally less involved, but also numerically less stable because it does not guarantee a symmetric positive definite covariance matrix when calculations can only be performed with finite precision. It is given by

$$\hat{P}^k = (I - K^k C) \tilde{P}^k.$$  

(2.16)

See eq. F.30 of [23] for a proof of the mathematical equivalence of the two formulae. In this work only the long form will be employed.

Prediction and correction step do not have to be executed with the same rate. It is even possible to split the measurement vector into different parts if different sensors are sampled at different rates, so that each measurement may be processed with its own update rate.

### 2.1.3 Parameter estimation with the Kalman filter

Using ordinary parameter estimation methods like the output error method, there is usually a strict separation between states and parameters. However, the Kalman filter can only estimate states. The solution is therefore to consider the parameters to be additional states. The state vector $x$ is augmented by the parameter vector $\theta$ forming the augmented state vector

$$x_a = \begin{pmatrix} x \\ \theta \end{pmatrix}.$$  

(2.17)

Accordingly, the system equation has to be augmented by a model for the propagation of the parameters in time. Usually, constant parameters are assumed, but this approach also allows for flexible models of parameter drift. A constant parameter would be modeled by setting its derivative and thus the corresponding rows of the system matrix $A$ and the input matrix $B$ to be zero regardless of anything. Slowly varying parameters could be modeled by Markov chains of different order. Of course, deterministic models (e.g. a known temperature dependency of a sensor bias or an aerodynamic derivative varying with angle of attack) are possible as well.
2 Mathematical background

If both parameters and states are to be estimated, the system will usually contain multiplications of these with one another and thus become nonlinear. In this case, a variant of the Kalman filter catering for non-linear system models like e.g. the Extended Kalman Filter (EKF) has to be used. However, for the application in this work, the system model remains linear.

2.2 Simplifications for aerodynamic parameter estimation

If the Kalman filter is restricted to only estimate the parameters of a linear (in the parameters) model without any states in the narrower sense, some simplifications of the original algorithm are possible. They allow to reduce computational burden considerably.

2.2.1 Simplifications for constant states

The parameters are assumed to be constant, and no other states are included in the filter. In this case, the state prediction step (eq. 2.9) may be dropped ($\hat{x}^k = \hat{x}^{k-1}$, so the need to maintain separate variables is omitted) and the covariance prediction step (eq. 2.10) simplifies to

$$\hat{P}^k = \hat{P}^{k-1} + Q$$

because $\Phi = I$ as $A = 0$. The correction step remains untouched. Note that the assumption of constant states does not prevent the filter from actually tracking varying states, as they will converge to their current value during the correction step. The states are still allowed to be subject to random noise.

2.2.2 Simplifications for coefficient-based aerodynamics model

We now assume an aerodynamics model based on classical Taylor expansion of the coefficients, e.g.

$$C_L = C_{L0} + C_{La} \cdot \alpha + C_{Lq} \cdot q.$$  \hspace{1cm} (2.19)

The parameters to estimate, and thus the states for the Kalman filter, are the derivatives ($C_{L0}$, $C_{La}$, $C_{Lq}$ in this case). As output for the Kalman filter the aerodynamic coefficient ($C_L$) is chosen. Although these coefficients cannot be measured directly, they may be calculated from available measurements, see section 2.4. Finally, the inputs of the filter are the independent variables 1, $\alpha$ and $q$. If the filter remains constrained to a single

\footnote{In flight, they are of course not really independent}
output, i.e. a single aerodynamic coefficient, the $C$ matrix may be replaced by the transposed input vector, and the system model reads
\begin{align*}
\dot{x}(t) &= Fw(t) \\
y(t) &= u(t)^T x(t) \\
z(t) &= y(t) + Gv(t).
\end{align*}

### 2.2.3 The simplified filter

If we apply all these simplifications to the original filter algorithm of equations 2.9 to 2.14, the following equations remain (superscript $k$ has been omitted):
\begin{align*}
K &= \frac{1}{u^T P u + r P} \\
x^{k+1} &= x + K (z - u x^T) \\
P^{k+1} &= (I - K u^T) (P + Q) (I - K u^T)^T + r K K^T
\end{align*}

where $r$ is the corresponding diagonal element of the measurement covariance matrix $R$. All equations belong to the correction step, so the hat above $x$ and $P$ is omitted from now on.

These equations are quite similar to those of the Recursive Least Squares (RLS) algorithm with forgetting factor, which can e.g. be found in chapter 6.II.B of [23]. Changing variable names and writing $x$ for $\theta$, $z$ for $y$ and $u$ for $x$, they read
\begin{align*}
K &= \frac{1}{u^T P u + \lambda P} \\
x^{k+1} &= x + K (z - u x^T) \\
P &= \frac{1}{\lambda} (I - K u^T) P.
\end{align*}

The main difference (apart from the covariance update using the short form of eq. 2.16) is that in the EKF the process noise $Q$ and measurement noise $r$ are specified independently while in the RLS case the forgetting factor $\lambda$ accounts for both. This has been found to allow significant improvements if the constants are tuned correctly; however, there is now the additional $Q$ matrix to be tuned.

One more important difference to notice is the way the covariance matrix entries grow: In the RLS case, the $P$ matrix is multiplied by a constant factor $\frac{1}{\lambda}$ in each timestep, leading to an exponential growth of each element. Although the factor is usually only very slightly greater than one, in phases where little information is added (like cruise flight), the covariances tend to become orders of magnitude too high, as the second factor $I - K u^T$ does not produce enough reduction. This behavior may be alleviated by
increasing the forgetting factor, however if one does so, the algorithm’s ability to track changes or even to converge to the true parameter values is slowed down as well.

On the other hand, in the EKF case the growth of the covariances is linear because only summands are added ($Q$ and $rKK^T$, respectively).

**Comparison to FTR** Another algorithm that was used for similar applications is the Fourier Transform Regression (FTR) algorithm [33]. This was also applied in a preceding thesis at the Institute of Flight System Dynamics [37], albeit on dedicated stationary real-time processing hardware. However, it is not completely recursive. Only the Fourier transformation is calculated each timestep, while the regression happens batch-wise every few timesteps. The regression could of course be replaced by the RLS algorithm, in which case the Fourier transformation would constitute an upstream bandpass filter. Such a filter can be expected to have a positive influence especially in noisy environments; however, it comes at the cost of a large increase in computational burden, not least because complex numbers have to be processed. Lastly, the required memory for FTR is larger because for every input and output value at each frequency of interest a value has to be stored.

### 2.2.4 Reformulation for computational efficiency

The formulation of the covariance matrix update as given in eq. 2.25 has a few drawbacks for a fast algorithm implementation. It requires 2 matrix-matrix multiplications, which become very expensive with increasing matrix dimensions. The total number of operations required is $2N^3 + 3N^2$ multiplications and $2N^3 + 3N$ additions, where $N$ is the dimension of the matrices. However, it is possible to reformulate the expression to increase computational efficiency.

Substituting $P'$ for $P + Q$, employing the matrix relation $AB = (B^T A^T)^T$ and making use of the fact that $P'$ is symmetric and $u$ and $K$ are vectors, eq. 2.25 was reformulated to read

$$P_{k+1} = P' + (P' u (K^T u - 2) + r K) K^T \quad (2.29)$$

without any matrix-matrix multiplications and a total number of $2N^2 + 3N$ multiplications and $2N^2 + 1$ additions. Similar reformulations can be made for the other equations of the algorithm, although the savings are smaller. As the matrices are symmetric, it is also possible to avoid the calculation of the elements below the main diagonal, thus further reducing the number of operations.

The final C implementation for the algorithm with 4 inputs is given as an example in appendix [A.1.1]
2.2.5 Disabling parameters

In some cases, e.g. when a parameter turns out to have no influence on the output, it may be desirable to prevent a single parameter from being identified and keep it at a fixed value of usually zero. This can be achieved by setting the initial value of the parameter to the fixed value and zero out its corresponding rows in the initial P and Q matrices. The computational burden will however not be reduced by this method, so if the modification is to be permanent, it may be more advisable to use a formulation based on reduced order matrices.

2.3 Observability

The observability of a dynamic system is a measure of how well the underlying (hidden) states can be estimated from the measurable outputs. Its formal definition is as follows:

A system is said to be observable if the initial state $x_0$ can be determined uniquely by examining the system output $y(t)$ for $t > t_0$ in a finite interval over some period of time (23, appendix B).

In the linear time-invariant case it can be shown (l.c.) that this is equivalent to the fact that the observability matrix $O$ has full column rank. This matrix is given by

$$O = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n_x-1} \end{pmatrix}. \quad (2.30)$$

(Superscripts are exponents.) For the discrete time case, $\Phi$ is used instead of $A$.

In the time-variant case the above equation changes to

$$O = \begin{pmatrix} C \\ C^k A \\ C^{k-1} A^k A^{k-1} \\ \vdots \\ C^{k-n} A^k A^{k-1} \cdots A^{k-n} \end{pmatrix}. \quad (2.31)$$

where superscripts are now time step numbers. The number of rows now does not have to equal the number of states any more but corresponds to the number of time steps observed. Again, in discrete time, $A$ has to replaced by $\Phi$.

Applying the simplifications from section 2.2.2, $C$ becomes $u^T$ and $\Phi$ becomes the identity matrix, so the observability matrix looks like (note the transposition)

$$O = \begin{pmatrix} u^k & u^{k-1} & \ldots & u^{k-n} \end{pmatrix}^T. \quad (2.32)$$
2 Mathematical background

It has full rank if its columns are linearly independent. Each column of this matrix corresponds to the time history of an independent variable, i.e. we obtain the intuitively plausible result that the parameters are observable if the independent variables are linearly independent. Another formulation of this fact is than none of the independent variables may have a mutual correlation coefficient of $\pm 1$.

2.4 Reconstruction of aerodynamic coefficients from inertial measurements

The system model described in section 2.2.3 operates on aerodynamic coefficients. Those however cannot be measured directly but have to be calculated from inertial measurements.

2.4.1 Calculation of total forces and moments

An INS yields (in a body-fixed coordinate system) the kinematic accelerations $\mathbf{a}_{K,\text{sen}}^f$ at the sensor location $\mathbf{r}_{\text{sen}}$ and the rotational rates $\omega_K^f$ which are independent of the location on a rigid body. Both are assumed to be bias-free because of the integrated sensor fusion algorithm of the INS. The accelerations have to be shifted to the CG according to

$$\mathbf{a}_{K,\text{CG}}^f = \mathbf{a}_{K,\text{sen}}^f - \omega_K^f \times \mathbf{r}_{\text{sen}}^f - \dot{\omega}_K^f \times \mathbf{r}_{\text{sen}}^f.$$  \hspace{1cm} (2.33)

Then the forces acting at the CG are obtained by multiplying $\mathbf{a}_{K,\text{CG}}^f$ with the airframe mass.

Calculation of the moments is not as straightforward. Usually, rotational accelerations are not measured; only rotational rates are available. Numerical differentiation of the rates yields the accelerations, but the noise level is increased with this operation. Finally, the reconstruction of the moments is not a simple multiplication but relies on inverting the moment of inertia tensor. The governing equation is the principle of angular momentum

$$\mathbf{M} = I_f^f \omega_K^f + I_f^f \omega_K^f \times I_f^f \omega_K^f.$$  \hspace{1cm} (2.34)

If the airframe is assumed to be symmetrical, i.e. the components $I_{xy}$ and $I_{yz}$ of the moment of inertia tensor vanish, eq. 2.34 is expanded to the following scalar equations:

$$L = I_{xx} \dot{\hat{p}} - I_{xx} (\dot{\hat{r}} + p \dot{q}) - (I_{yy} - I_{zz}) q r$$  \hspace{1cm} (2.35)

$$M = I_{yy} \dot{\hat{q}} + I_{xx} (p^2 - r^2) - (I_{zz} - I_{xx}) p r$$  \hspace{1cm} (2.36)

$$N = I_{zz} \dot{\hat{r}} - I_{xx} (\dot{\hat{p}} + q \dot{r}) - (I_{xx} - I_{yy}) p q$$  \hspace{1cm} (2.37)

where all angular rates are kinematic and given in the body-fixed frame. The product of inertia $I_{xz}$ would only vanish if the $f$-frame was aligned with the principal axes of the airframe which is usually not the case.
2.4.2 Non-aerodynamic effects

The calculations above yield the total forces and moments acting at the CG. For parameter estimation purposes however only the fractions contributed by aerodynamic effects are of interest. To separate these, the forces and moments produced by the propulsion system must be subtracted from the total ones. If they can be measured directly during flight, all the better; but in most cases this is not possible, so a propulsion model has to provide their current values dependent on flight state. The propulsion model for this thesis will be described in section 4.1.2.

Gravitational forces are not measured by accelerometers (they only sense the sum of the external forces, divided by mass) and therefore do not need to be compensated for. As they act in the CG, they do not introduce moments at all. In free flight, no other external effects like ground reactions or towing forces need to be considered.

2.4.3 Normalization

To obtain the non-dimensional coefficients, the aerodynamic forces and moments are divided by reference area and acting dynamic pressure; in case of the moments as well by the reference lengths.

2.5 Comparison of time histories

A common task in parameter identification is to compare the time histories of two values, usually a measured quantity $z$ and its modelled estimate $y$. Such a comparison should ideally yield a scalar value as measure for the goodness of fit. Commonly used for this task is the coefficient of determination $R^2$, whose definition is given e.g. in [26], chapter 5.1. It is based on the different possible sums of squares (total sum of squares $SS_T$, regression sum of squares $SS_R$ and error (residual) sum of squares $SS_E$). The most important equations are quoted here:

\[ SS_T = \sum_{i=1}^{N} [z_i - \bar{z}]^2 \]  \hspace{1cm} (2.38)

\[ SS_R = \sum_{i=1}^{N} [y_i - \bar{z}]^2 \]  \hspace{1cm} (2.39)

\[ SS_E = \sum_{i=1}^{N} [z_i - y_i]^2 \]  \hspace{1cm} (2.40)

\[ R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_E}{SS_T} \]  \hspace{1cm} (2.41)
where $\bar{z}$ is the mean value of the vector $z$ and the index $i$ denotes the $i$-th component of a vector.

Switching to real time PID, the time histories of the parameters move into focus. When evaluating algorithm performance by means of simulation, their convergence to a known constant value (“truth”) is of interest. This value takes the role of the measured quantity $z$, while the parameter time history is the model output $y$. In this case, $SS_T$ becomes zero, so the coefficient of determination cannot be used to assess the fit quality. To re-normalize $R^2$, the mean value $\bar{z}$ can be dropped from equations 2.38 and 2.39. However, due to the values being squared, the $R^2$ value tends to be very high even for moderately good fits. This can be overcome by using a quantity which is not based on squaring.

To this end, define the relative area of absolute error (RAAE) $a_e$ to be given by

$$a_e = \frac{\sum_{i=1}^{N} |z_i - y_i|}{\sum_{i=1}^{N} |z_i|} = \frac{1}{N |z|} \sum_{i=1}^{N} |z_i - y_i| \quad \text{for constant } z.$$  

(2.42) (2.43)

This value relates the area between two curves to the area the truth curve encloses with the $x$ axis. For two constants, it becomes the relative error between them, making it an intuitive measure for the goodness of an estimation result. Unlike with the coefficient of determination, a relative area of absolute error of zero indicates a perfect fit while the value tends to infinity as the fit gets worse.

The RAAE definition is somewhat similar to the ITAE criterion (integral of time-multiplied absolute error, see e.g. [47], chapter 8.3) which is often used as a cost function in optimal control applications. The difference is that the time dependency is dropped and the value is normalized by a reference area.

The RAAE criterion – as well as any other criterion based on normalization by a true value – will fail because of division by zero if the true value is identically zero, which may happen e.g. in the evaluation of models where parameters have been temporarily dropped. In this case, a non-normalizing criterion has to be used. The RAAE can be adapted to that end by dropping the term $|z|$ from eq. 2.43 resulting in the mean absolute value of the parameter.
3 Demonstrator aircraft

3.1 Concept

To prove the real-time PID functionality in flight tests, a demonstrator aircraft was designed and built. The wing for it was taken from a commercially available styrofoam R/C model airplane, for which a new fuselage was built that could accommodate the required sensor hardware. More powerful engines were also installed to compensate for the additional weight. A twin engine configuration was chosen because it allows a nose boom with an air data sensor to be installed. Air flow at the nose boom position is less disturbed than it were with a single-engine configuration.

For the task of real-time parameter estimation, the aircraft had to be equipped with additional equipment, notably the following:

Onboard computer:
The onboard computer provides the hardware to run the estimation algorithm on, but it is also assigned some other tasks like gathering sensor data and communicating with the ground station. It can also act as an autopilot performing e.g. waypoint navigation. In manual control mode, control laws like aileron differentiation are calculated here.

Air data sensor:
Airflow around the aircraft needs to be known by magnitude and direction. The five hole probe on the nose boom in combination with static pressure orifices in the rear fuselage gathers the pressures required for this task. They are then measured by pressure transducers mounted on a circuit board inside the fuselage.

Integrated Navigation System:
Forces and moments currently acting onto the aircraft can be deducted from accelerations and angular rates of the rigid body, which are provided by the INS. Additionally, orientation data provides valuable insights during post-flight analysis, although it is not directly needed by the PID algorithm.

Data link:
A bidirectional data link allows to control the flying aircraft by high level commands instead of relying on manual R/C control while at the same time sending current flight data (sensor readings, computer states, algorithm results) to a ground station for supervision. As the onboard computer does not possess a mass storage device, the data link is also the only way to store flight test results.
An overview of the participating components and their communication protocols among each other is given in figure 3.1. A specific list of the hardware in use can be found in appendix B.2. Figure 3.2 shows the realized aircraft ready to fly.

![Diagram of aircraft components](image)

**Figure 3.1: Components of the demonstrator aircraft system**

### 3.2 Onboard computer

As a hardware platform for the demonstrator, the Paparazzi system [5] [10] was chosen. This system comprises an open source, open hardware on-board computer platform with a navigation controller intended for R/C aircraft. A ground station and supervision software system is available as well which includes comfortable means of adopting the on-board software to specific R/C hardware, telemetry, flight plans and aircraft configurations.

The system was chosen because it provides all basic functionality to operate a remotely controlled vehicle by high-level commands while at the same time having interfaces for various sensors. The complete code base is accessible and thus extensions could easily be undertaken. The required hardware could be bought from third-party suppliers. It is more powerful than comparable products like e.g. ArduPlane. A broad user base and numerous MAV competition prizes had already proven the system’s applicability in practice.

#### 3.2.1 Main processor

The Paparazzi project offers various onboard processor boards. In the demonstrator aircraft, a Tiny v2.11 board was used. Its main processor is a Philips LPC2148
3.2 Onboard computer

Figure 3.2: The demonstrator aircraft on the field between flights

A microcontroller with a 32bit ARM7 architecture clocked at 60 MHz, 40 kB RAM and 512 kB Flash ROM. A GPS receiver and a power supply are included in the board and various connectors are available to access the processor’s interfaces like e.g. analog input channels, PWM output or I2C bus.

This particular Paparazzi board was chosen because at the time this dissertation project started it was the most powerful available autopilot board. However, by now others are available offering improved performance, e.g. the Paparazzi project’s Lisa/L board. Adaptation to a new board was beyond the scope of this project, though.

3.2.2 Control laws

The Paparazzi code implements a hierarchic control law structure. The autopilot maintains abstract commands which are translated into PWM outputs for the different actuators. At least roll, pitch and throttle commands are present; for the present thesis, user-defined commands for yaw and flaps were added. The (feedforward only) control laws describe how e.g. a roll command is translated into an aileron servo PWM signal. For example, classical command mixing can be realized this way.

For the demonstrator aircraft at hand, the following control laws were used:
3 Demonstrator aircraft

**Engines:**
Both engines were driven directly by the throttle command.

**Ailerons:**
Each aileron has its own servo, so software-based aileron differentiation was realized with the lowering aileron deflecting 66\% of the raising one with the total deflection given by the roll command $\xi_C$. Additionally, the flap command $\kappa_C$ allowed symmetric lowering of both ailerons. Full flap deflection corresponded to 50\% aileron deflection in order not to jeopardize roll control.

**Elevator:**
Elevator deflection corresponded directly to the pitch command $\eta_C$. A possible extension would be a combination with the flap command to avoid trim changes. This was however not used in order to avoid correlation problems for the PID algorithm.

**Rudder:**
The yaw command $\zeta_C$ directly drove the rudder, but additionally, feedforward turn coordination was realized by mixing the roll command $\xi_C$ into the rudder deflection. To keep correlations of aileron and rudder deflection low, the rudder law included a saturating element so that rudder deflection increased linearly with roll command up to a maximum of 18\%, at which it was clipped unless a manual yaw command input was given.

All control law processing was realized in onboard software, the R/C transmitter was configured to only transmit the positions of the control elements without doing any command mixing itself. A block diagram of the lateral branch of the control laws is shown in figure 3.3.

![Figure 3.3: Block diagram of the lateral control laws](image)

24
3.3 Sensors

3.2.3 Control modes

The R/C receiver in this setup is connected directly to the autopilot, all R/C commands are thus processed according to the autopilot’s control laws before being sent to the servos. It is possible to operate the aircraft in three different modes:

Manual:
The R/C control inputs are taken as the values of the roll, pitch, etc. commands which then are processed according to defined control laws and sent to the servos. No feedback control is done.

Auto 1 (Attitude control):
The R/C control inputs for elevator and rudder become commanded values for pitch and bank angle which are to be reached using feedback automatic control. The required pitch and roll commands are calculated by the autopilot and processed according to the control laws. Actual attitude angles are taken from the horizon sensors. Other R/C commands are processed as in manual mode.

Auto 2 (Waypoint/maneuver navigation):
The aircraft autonomously follows a flight plan consisting of waypoints and other navigation elements like circles or survey patterns up to automatic landings. A high level navigation algorithm provides the commanded values for pitch and bank angle as well as throttle. R/C control is disabled except for the possibility to return to another mode.

3.3 Sensors

In the standard configuration, the Paparazzi system uses only its integrated GPS sensor (yielding position and kinematic velocity) and a set of infrared horizon sensors to estimate aircraft pitch and roll attitude. The information from these sensors is sufficient for waypoint navigation, but for parameter estimation, additional values are required, viz. air data (3D inflow velocity), translational accelerations and angular rates. Control surface positions also need to be known, but it was decided to rely on actuator modeling instead of sensors to reduce hardware complexity.

3.3.1 Air data probe

Air data is collected by a five hole probe mounted on a carbon fiber nose boom in connection with static pressure ports in the fuselage side wall. The probe is depicted in detail in figure [3.4] while in figure [3.2] it is hidden below the protection cover on the nose boom. A set of SMD pressure sensors integrated into a dedicated sensor board measures static pressure $p_{stat}$, probe front pressure $p_5$ and probe differential pressures $\Delta p_{13}$ and $\Delta p_{24}$. To make use of the sensor builtin A/D converters whose resolution is
finer than that of the main processor’s converters, the sensor board is connected to the autopilot board via an I2C bus interface allowing sequential transmission of the pressure data.

**Calibration** The probe was calibrated in the Institute’s wind tunnel in an already mounted state \[15\] for flow angles ranging from $-10^\circ$ to $20^\circ$ for $\alpha$ and $\pm20^\circ$ for $\beta$. Figure 3.5 shows the setup on the panning device. Calibration took place before the new fuselage of the aircraft was planned and built, and unfortunately, no further wind tunnel tests could be done with the new configuration. Due to the length of the noseboom, it is however expected that the influence of the fuselage on the airflow at the pressure orifice location can be neglected.

The complete measuring chain as used in flight was also used in the wind tunnel. Reference data from the sensor board was sent to the calibration software using the autopilot’s telemetry link. Because of the rigid wind tunnel hardware, only static measurements could be conducted. As a first attempt, a calibration model of the
following form was determined:

\[ p_{dyn} = p_5 + \frac{k_{d\alpha}}{p_5} \Delta p_{13} + \frac{k_{d\beta}}{p_5} \Delta p_{24} + p_{dyn,0} \]  \hspace{1cm} (3.1)

\[ \rho = \frac{\frac{p_{stat}}{R_m T_{ref}}}{p_5} \]  \hspace{1cm} (3.2)

\[ V_A = \sqrt{\frac{2 p_{dyn}}{\rho}} \]  \hspace{1cm} (3.3)

\[ \alpha = k_{\alpha_1} \frac{\Delta p_{13}}{p_{dyn}} + k_{\alpha_2} \frac{\Delta p_{24}}{p_{dyn}} \]  \hspace{1cm} (3.4)

\[ \beta = k_{\beta_1} \frac{\Delta p_{13}}{p_{dyn}} + k_{\beta_2} \frac{\Delta p_{24}}{p_{dyn}} \]  \hspace{1cm} (3.5)

with the range of values suitably limited to prevent numerical issues in the calculations based on the sensor data. The additional terms in eq. (3.1) compensate for drops in dynamic pressure due to off-axis inflow. Because the probe geometry does not have a complete rotational symmetry, different factors \( k_{d\alpha} \) and \( k_{d\beta} \) had to be introduced.

The calibration was found to give acceptable accuracy (RMS \( \Delta V < 0.3 \text{ m/s} \), RMS \( \Delta \alpha, \beta < 0.2^\circ \)) in an airspeed range of 10 \ldots 30 \text{ m/s} and inflow angles below 10°. However, as could already be determined from simulation runs, the simple model was found to be insufficient for the full range of flow angles expected during maneuvering flight. Additional non-linear terms had to be included to compensate errors arising from large
deviations from trim state. The method of multivariate orthogonal functions as described e.g. in [26], chapter 5.1.6, yielded a set of terms to be included for a calibration over the full flow angle range. Candidate regressors were the measured pressures, with and without normalization by front hole pressure \( p_5 \). The significance of the remaining terms was confirmed the step-wise regression method from [26], chapter 5.4.2. The final model had the following form:

\[
p_{dyn} = k_{d1}p_5 + k_{d2}\frac{\Delta p_{13}}{p_5} + k_{d3}\frac{\Delta p_{24}}{p_5} + k_{d4}\left(\frac{\Delta p_{24}}{p_5}\right)^4 + p_{dyn,0} \quad (3.6)
\]

\[
\alpha = k_{\alpha1}\frac{\Delta p_{13}}{p_{dyn}} + k_{\alpha2}\frac{\Delta p_{24}}{p_{dyn}} + k_{\alpha3}\left(\frac{\Delta p_{13}}{p_{dyn}}\right) + \ldots + k_{\alpha4}\Delta p_{13} + k_{\alpha5}\Delta p_{24} + k_{\alpha5}p_{dyn} + \alpha_0 \quad (3.7)
\]

\[
\beta = k_{\beta1}\frac{\Delta p_{13}}{p_{dyn}} + k_{\beta2}\frac{\Delta p_{24}}{p_{dyn}} + k_{\beta3}\left(\frac{\Delta p_{24}}{p_{dyn}}\right)^3 + k_{\beta4}\Delta p_{24} + k_{\beta5}p_{dyn} + \beta_0 . \quad (3.8)
\]

The sensor driver code also corrected the flow angles \( \alpha \) and \( \beta \) for the off-CG sensor location by performing the inverse calculation of eq. [5.2].

As the expected flight envelope of the demonstrator does not include large temperature variations and no possibility to measure the outside air temperature was available, a constant temperature had to be programmed before the flight. Subsequent calculations of air density \( \rho \) would use this constant value. For flights with large variations in temperature, the airspeed may thus be in error; but for the flight tests carried out this was not the case.

**Interface considerations**  As the I2C interface can only read the data from one sensor at a time, the pressure values arrive sequentially at the autopilot. One possibility is to read data from all sensors (one after the other) in one call to the sensor module main function. However, due to lacking multi-tasking ability this means the processor has to spend a significant amount of time in a busy wait state for the I2C data to arrive. This totals at a CPU load increase of 7% which may or may not be tolerated depending on the number of other modules being used. Additionally, it has to be ensured that the processor cannot remain in the wait state longer than a small fraction of a second, e.g. when a hardware failure occurs.

The alternative is to process only one sensor per call of the main function. At the end of the function, data from the next sensor is requested. The request itself is processed asynchronously in an interrupt routine, so no busy wait is required. The drawback of this method is that each sensor is read only at a fraction of the main control loop rate corresponding to the number of sensors; in this case at one fourth. In addition, the data of the individual sensors is sampled at significantly different times, making it necessary to interpolate them to a fixed time point. Data quality is thus much worse.
During the first flight tests, the one-sensor-per-call method was used to rule out failures due to processor overload, but as soon as the safety of the single-call method had been established, the method was switched.

Sensor noise remained an issue, thus a small moving average filter was introduced. The final values for speed and flow angles were calculated as the average of the last three measurements.

### 3.3.2 Integrated navigation system (INS)

The demonstrator aircraft is equipped with an integrated navigation system (INS) of type Xsens MTi-G providing inertial acceleration $a_f$, velocity $v_p$, position, roll rates $\omega_f$ and orientation, the latter electively in form of Euler angles, attitude quaternion or direction cosine matrix (see e.g. [45], chapter 3.6 for details on these representations). The INS comprises an integrated inertial measurement unit (IMU) and a GPS receiver. It uses its own GPS antenna and runs its own on-board filter algorithm to fuse its sensors and compensate their errors.

Communication with the autopilot board is realized via an RS232 interface, at a configurable rate of up to 120 data messages per second. With a data message length of about 90 bytes and a transmission rate of 460800 bits per second (the maximum for the onboard computer), a single message transfer took about $1.5\ \text{ms}$, eliminating the data transfer as limiting factor for the frequency. Data was transferred as four-byte floating point values. See appendix B.2 for details on the connection.

To keep noise levels at an acceptable level, the accelerations and rates had to be filtered with a moving average filter over 3 timesteps. As the PID code needs to know the angular accelerations, the sensor driver code performed numerical differentiation of the angular rates. For the differentiation, several methods of noise reduction were tried, e.g. second order finite difference schemes or the differentiation of the filtered rates. However, in simulation runs this did not improve the quality of the PID results, so finally a first order differentiation scheme on the unfiltered rates was used.

### 3.3.3 Infrared attitude sensors

The standard Paparazzi configuration uses infrared thermopile sensors to estimate the aircraft’s pitch and roll angles. Six of these sensors are arranged orthogonal to each other, with the opposing pairs wired together. This setup allows to measure the mean IR radiation gradient in three dimensions. As the sky is much colder than the ground, the gradient can be considered an indicator of the local vertical and converted back into attitude angles. It does not yield the heading angle, though. The IR sensors are connected to the main processor’s internal A/D converters.
In this thesis, the IR sensors were only used during early flight tests for system integration. When the INS was included and proven, attitude reference for the autopilot was taken from the INS system.

### 3.3.4 Control surface deflections

For parameter identification, the current positions of all relevant control surfaces (in this case two ailerons, elevator and rudder plus two throttles) need to be known. However, their direct measurement using e.g. potentiometers adds a lot of complexity. Instead it was decided to rely on actuator modeling to estimate the required data. The actuator model will be described in section 5.2.2. For the flying code, it was integrated directly in the Online PID module, see below.

### 3.4 Onboard software

The Paparazzi on-board system as is comprises attitude stabilization and high-level navigation algorithms. These interface with the R/C receiver and the aircraft’s actuators (servos) and can communicate with the ground station via the data link. Some sensor drivers are included, among these the GPS and horizon sensor interfaces which are used by default by the autopilot code.

The software is basically divided into three main types of functions: initialization routines being run once at system startup, synchronous (“periodic”) tasks being run with a specified frequency (at the main loop’s frequency of 60 Hz or a unit fraction of it) and asynchronous (“event”) tasks being run whenever a certain event occurs, e.g. when a bus interface signals a new message available. By periodic event triggering by an external source it is possible to achieve a synchronous code execution with higher rates than the main loop.

The Paparazzi code is highly configurable. Using a set of XML files it is possible to adapt the software system to the setup at hand consisting of the aircraft itself, radio control, telemetry link and flight plan. Furthermore, the „settings“ functionality allows a subset of autopilot configuration variables (e.g. controller gains or PID covariance matrices) to be made tunable during the actual flight via data link. The other variables will remain constant at their value specified in the corresponding XML file. Having specified all details, the on-board code will be compiled on the ground station computer and then uploaded to the processor board’s flash memory.

One more configuration feature which was heavily used in this thesis is the possibility to use so-called modules. A module is a piece of software which may optionally be included in the compilation process. Such a module comprises source code files and possibly compiler definitions. Each module can provide, among others, an initialization function, a periodic function and an event function to provide analogous functionality as the main loop. Available modules range from drivers for additional sensors to advanced...
navigation routines like formation flight. The inclusion of a module requires the code to be re-compiled and uploaded again; however module parameters may be tuned via the settings mechanism as well.

3.4 Onboard software

3.4.1 Parameter identification software

During this thesis, code for parameter identification was added to the system in the form of modules. These additions comprise the main estimation algorithm, the calculation of the coefficients, the sensor interface and driver for the five hole probe, substantial additions to the INS interface and the adaptation of the airframe configuration files to the given setup of the demonstrator aircraft. The respective modules are described in detail in the following.

INS interface:
While a basic interface to the Xsens MTi-G sensor was already available, it had to be expanded for this thesis to differentiate the angular rates and apply filtering as described in section 3.3.2. Also, the available code could only address the sensor hardware, so for simulation purposes a sensor model had to be added. This module’s main function had to be asynchronous, as it was triggered by the arrival of a data message from the INS sensor.

Five hole probe sensor interface:
This module was split into two parts. An architecture dependent part comprised the direct communication with the sensor hardware or with the simulation kernel yielding raw A/D converter readouts for the measured pressures. The architecture independent part calculated physical values from the raw numbers and applied the sensor model from section 3.3.1 as well as the corrections for an off-CG location to these to reconstruct airspeed and flow angles.

Coefficient calculation:
Using the translational and rotational accelerations from the INS module and the flow angle from the five hole probe, this module implemented the calculations from section 2.4 as well as the thrust model to be described in section 4.1.2.

Online PID:
Besides the main PID algorithm as given in section 2.2.3 and section 2.2.4, this module also comprised a simple second order actuator model to estimate the current control surface deflections. In simulation, all relevant data were written to a log file to allow posterior analysis and tuning. Such a comfortable data store was however unavailable during flight test.

During the first tests, the calculations were done in synchronous mode with the maximum possible frequency of 60 Hz, i.e. the main loop frequency. This however lead to synchronization issues, especially during INS state changes (e.g. loss of GPS fix). Therefore, the asynchronous mode was used instead. The INS was configured to periodically send
3 Demonstrator aircraft

<table>
<thead>
<tr>
<th>Sub-task</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parsing of the INS data message buffer</td>
<td>220 µs</td>
</tr>
<tr>
<td>Air data sensor read out</td>
<td>960 µs</td>
</tr>
<tr>
<td>Calculation of thrust and coefficients</td>
<td>390 µs</td>
</tr>
<tr>
<td>PID algorithm</td>
<td>2700 µs</td>
</tr>
<tr>
<td>Data transfer to control loop</td>
<td>140 µs</td>
</tr>
</tbody>
</table>

Table 3.1: Execution times of PID sub-tasks

data messages, the reception of which triggered the event task of the INS module which in turn sequentially called the main functions of the other modules. Included in the data message was a UTC timestamp whose increments were used as denominator for the numerical differentiation. Thus the INS provided reference for the execution frequency for the sensor and PID calculation, allowing to decouple this frequency from the main loop.

A timing analysis of the running onboard code revealed that the whole event task required about 4.4 ms to complete, with the contributions of the various sub-tasks listed in table 3.1. The maximum possible frequency for the given setup was found to be 90 Hz.

In software-in-the-loop simulation, however, no event triggering is possible without substantial changes to the code base. All asynchronous functions are also run synchronously with the main loop frequency, thus in this case the PID frequency stayed fixed at 60 Hz.

All modules added their own specific messages to the data link so that their information could be send to the ground station. This was also the only possibility to log data during flight test, as no onboard mass storage device was present.

The calculation of the coefficients and the main PID algorithm could be configured using C preprocessor definitions whose values were specified in the XML airframe file. Thus it was possible to use either the original simulated values or the results of the component models for the propulsive forces, the coefficients or the regressors of the PID algorithm.

An overview of the control and data flow of the additional modules is given in figure 3.6. In accordance with the standard practice in the Paparazzi project, all information relevant for more than one module was stored in global variables.
3.4 Onboard software

Figure 3.6: Data and control flow of the additional modules
4 Reference dataset generation

To allow assessment of the onboard software beforehand in simulation, a reference dataset of the aircraft’s aerodynamic and flight mechanical characteristics is required. Detailed data for the static longitudinal motion could be attained from wind tunnel tests with a half model. These data were then completed for the lateral-directional motion and dynamic effects using numerical methods.

4.1 Wind tunnel tests

The Institute’s wind tunnel with a test section of $\varnothing1.5 \times 3$ m allowed to test one half of an original exemplar of the demonstrator aircraft. During a student’s research project [43], such a half aircraft was constructed and integrated into a test setup that originated...
from another research project [8]. Unfortunately, like with the five hole calibration, the half model tests were done with the original fuselage, before the new one was envisioned, so certain differences to the actual flying configuration can be expected.

In the measurement campaign, the model’s lift, drag, pitching moment and incremental rolling moment were measured as function of airspeed, angle of attack, power setting, aileron deflection and elevator deflection. The test setup including the half model is shown in figure 4.1.

4.1.1 Lift coefficient

\[ C_{L0} = 0.28, C_{L\alpha} = 4.7/\text{rad} \]

\[ C_{D0} = 0.06 \]

Figure 4.2: Baseline \( C_L \) and polar of the clean configuration at \( V_A = 17.5 \text{ m/s} \)

Example measurements for an airspeed of 17.5 m/s can be seen in figures 4.2 to 4.5. Figure 4.2 shows the baseline \( C_L \) of the clean airframe with propellers windmilling and control surfaces neutral. The curve exhibits an articulate linear range and docile stalling characteristics. At a maximum lift coefficient of 1.05, a stalling speed of about 10 m/s can be expected for the demonstrator airframe in final configuration.

Increments to \( C_L \) due to throttle setting can be seen in figure 4.3. The additional lift is caused by propeller downwash over a large fraction of the wing, which also alleviates the lift breakdown beyond the stall angle of attack. A slight dip in the increments can be seen around the clean stalling angle of \( \alpha \approx 12.5^\circ \), but the additional lift then increases further throughout the measured range. Total lift however drops after stall even with full throttle, albeit much slower than in the clean configuration. The maximum lift coefficient increases to 1.14 at full throttle.

Lift changes caused by deflection of the control surfaces are shown in figures 4.4 (elevator) and 4.5 (single right aileron). Elevator efficiency is fairly linear in both deflection and angle of attack over the linear \( C_L \) range, justifying a linear derivative approach for
4.1 Wind tunnel tests

Figure 4.3: $C_L$ increment due to throttle setting

Figure 4.4: $C_L$ increment due to elevator deflection
4.1.2 Propulsion characteristics

An important aspect for which the data was taken from the wind tunnel tests is the propulsion characteristics, i.e. the forces and moments introduced by the propellers. These are required by the PID algorithm, see section 2.4.2. From the wind tunnel experiments, propulsion force increments with respect to the clean configuration were available over a range of airspeeds, angles of attack and throttle settings. From these, a parametric model was derived which had the following form:

\[ X^F_f = \delta_t \cdot \left( X_0 + X_\alpha \cdot \alpha + X_t \cdot \delta_t + X_V \cdot \frac{1}{V_A} \right) \]  
\[ Z^F_f = \delta_t \cdot \left( Z_0 + Z_\alpha \cdot \alpha + Z_t \cdot \delta_t + Z_{\alpha t} \cdot \alpha \cdot \delta_t + Z_{\alpha V} \cdot \frac{\alpha}{V_A} \right) \]  
\[ M^F_f = X^F_f \cdot z^F. \]

The additional Z force corresponds to the lift increments depicted in figure 4.3. The pitching moment due to the \( Z^F \) force was neglected because its magnitude is smaller than that of the \( X^F \) force and the additional force is expected to attack very close to
the center of gravity. This decision is supported by the wind tunnel data, where the measured pitching moment increments were hardly above the noise level.

Rolling and yawing moments caused by the $X^F$ and $Z^F$ forces can be expected to cancel each other because of the symmetry of the engine installations. While asymmetric power settings are principally possible with the aircraft at hand, they were not included in the control laws and thus, both engines are expected to deliver equal forces over the whole flight envelope.

Neither side force and moments due to sideslip nor propeller reaction torque could be measured with the half model, thus although the propellers are rotating in the same direction, asymmetric effects of the propulsion (i.e. torque reaction moment and an asymmetric center of thrust due to angle of attack (“P-Factor”)) were neglected altogether.

The wind tunnel measurements were unfortunately carried out with a single supply voltage of 12 V only, although in flight the accumulator voltage may vary from 10 V to 12.5 V. To take this into account, the thrust could be multiplied by a factor $\frac{V}{12V}$ or its square if the actual supply voltage is measured. However, although the motor rotation speed is proportional to the voltage, thrust is only proportional to rotation speed squared when the advance ratio is constant. Therefore, more measurements are desirable. This has to be kept in mind when using this propulsion model.

### 4.2 Numerical methods

#### 4.2.1 Digital Datcom

The United States Air Force Stability and Control Datcom (Data compendium) [16] is a collection of analytical methods and extensive experimental data which can be used to estimate an aircraft’s flight mechanical characteristics from its geometry. The program Digital Datcom [46] [17] incorporates these methods for computer based applications. It is possible to add experimental data (e.g. in form of lift, drag and pitching moment as function of angle of attack) to the geometry input which improves the quality of the generated data. Adding experimental data is recommendable when analyzing small aircraft because the original data base was intended for full-scale manned aircraft. Especially the the airfoil characteristics tend to become inaccurate due to the low Reynolds numbers of model aircraft for which the code was not designed. A former work found the drag coefficient of MAV to be underestimated by a factor of about 2 ([11], chapter 3.1.1).

If experimental data is available (or possibly produced by an analysis program better suited for the given aircraft), the Datcom still provides a fast method to estimate dynamic and control derivatives. Therefore it was used to complete the wind tunnel dataset. Because experimental data can only be input for the various components of
an aircraft (wing, tail, body, ...) but not for the aircraft as a whole, data for the isolated horizontal tail was calculated beforehand by the Datcom program. These results were subtracted from the wind tunnel measurements, and the difference was input as experimental data for the wing-body configuration.

4.2.2 XFLR5

XFLR5 [11] is another program to estimate a flight mechanics data set from aircraft geometry. It is intended for the design of model sailplanes but can also be used for the analysis of existing aircraft. The program allows to analyze 3D aircraft geometries using a variety of numerical methods, among which are lifting line theory, vortex lattice methods and panel methods. Making use of the XFOIL code [15] to calculate 2D airfoil polars, its estimation of the airfoil characteristics in the low Reynolds number range typical for R/C model aircraft is believed to be better than Datcom’s. In this special case, however, the lift curve prediction was equally good with both programs while both predicted the drag much too small compared to the wind tunnel data. Generally, the program aims mainly at the longitudinal stability characteristics. Lateral, dynamic and control derivatives are available for trimmed conditions, but the functionality is still experimental at the time of this writing.

The results from the XFLR5 calculations were mainly used as an independent comparison to the other tools to verify their plausibility. They were not actually used in the final model.

4.3 Consolidated flight dynamics model

Skeleton JSBSim (see section 5.1) input files to be used for simulation are provided by the Datcom+ variant of the Digital Datcom. To combine these with the wind tunnel test data, a Matlab framework was designed to automatically create Datcom input files with experimental data sections activated, run them, and complement the resulting JSBSim files with these wind tunnel data elements that Datcom cannot provide (e.g. lift as function of both angle of attack and power setting). Effects of the rudder deflection were unfortunately unavailable from both Datcom and wind tunnel data, so they had to be calculated by hand from the formulae of chapter 12.4 of [42]. Drag due to sideslip was estimated using the Datcom to estimate the drag due to angle of attack for a rotated body-vertical tail configuration. This matched well with the derivative calculated from the wind tunnel data for the fuselage/tail combination that was gained as extra during the calibration of the five hole probe, see section 3.3.1. The fuselage/tail value was then increased by 30% to account for the additional drag of wing and nacelles.

An overview of the resulting aerodynamics model is given in table 4.1. All derivatives were stored in coefficient form as lookup tables as function of the values listed in the respective column in the table. Propulsion effects were not included in the aerodynamics model but modeled separately as will be described in section 5.2.4.
<table>
<thead>
<tr>
<th>Cause</th>
<th>Function</th>
<th>Data origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift</td>
<td>clean aircraft</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>Pitching moment</td>
<td>pitch rate</td>
<td>$q$</td>
</tr>
<tr>
<td></td>
<td>AoA rate</td>
<td>$\alpha, \dot{\alpha}$</td>
</tr>
<tr>
<td></td>
<td>elevator deflection</td>
<td>$\alpha, \eta$</td>
</tr>
<tr>
<td></td>
<td>aileron deflection</td>
<td>$\alpha, \xi$</td>
</tr>
<tr>
<td>Drag</td>
<td>clean aircraft</td>
<td>$\alpha$</td>
</tr>
<tr>
<td></td>
<td>sideslip</td>
<td>$\beta^2$</td>
</tr>
<tr>
<td></td>
<td>elevator deflection</td>
<td>$\alpha, \eta$</td>
</tr>
<tr>
<td></td>
<td>aileron deflection</td>
<td>$\alpha, \xi$</td>
</tr>
<tr>
<td>Sideforce</td>
<td>sideslip</td>
<td>$\beta, \alpha$</td>
</tr>
<tr>
<td></td>
<td>roll rate</td>
<td>$p, \alpha$</td>
</tr>
<tr>
<td></td>
<td>yaw rate</td>
<td>$r$</td>
</tr>
<tr>
<td></td>
<td>rudder deflection</td>
<td>$\zeta$</td>
</tr>
<tr>
<td>Rolling moment</td>
<td>sideslip</td>
<td>$\beta, \alpha$</td>
</tr>
<tr>
<td>Yawing moment</td>
<td>roll rate</td>
<td>$p, \alpha$</td>
</tr>
<tr>
<td></td>
<td>yaw rate</td>
<td>$r, \alpha$</td>
</tr>
<tr>
<td></td>
<td>aileron deflection</td>
<td>$\alpha, \xi$</td>
</tr>
<tr>
<td></td>
<td>rudder deflection</td>
<td>$\zeta$</td>
</tr>
</tbody>
</table>

$^1$ Modeled separately for left and right wing

Table 4.1: Modeled aspects in the consolidated nonlinear aerodynamics model

### 4.3.1 Reduced model for parameter identification

Because of the restricted computing resources available at the onboard computer, the reference dataset described above had to be simplified to some extent, so that only a relatively small set of parameters had to be estimated. Thus, a simple first order Taylor series expansion was chosen for all coefficients, except for $C_D$ where squared dependencies are added:

\[
C_D = C_{D0} + C_{DC} \cdot \alpha + C_{DP} \cdot \beta + C_{Dq} \cdot \dot{q} + C_{DN} \cdot \eta \]

\[
C_L = C_{L0} + C_{L\alpha} \cdot \alpha + C_{Lq} \cdot \dot{q} + C_{LN} \cdot \eta \]

\[
C_m = C_{m0} + C_{m\alpha} \cdot \alpha + C_{mq} \cdot \dot{q} + C_{m\eta} \cdot \eta \]

\[
C_C = C_{C0} + C_{C\beta} \cdot \beta + C_{C\alpha} \cdot \dot{\alpha} + C_{C\eta} \cdot \dot{\eta} + C_{C\xi} \cdot \xi + C_{C\zeta} \cdot \zeta \]

\[
C_l = C_{l0} + C_{l\beta} \cdot \beta + C_{lp} \cdot \dot{p} + C_{lt} \cdot \dot{r} + C_{lx} \cdot \xi + C_{lz} \cdot \zeta \]

\[
C_n = C_{n0} + C_{n\beta} \cdot \beta + C_{np} \cdot \dot{p} + C_{nr} \cdot \dot{r} + C_{nx} \cdot \xi + C_{nz} \cdot \zeta \]

According to the standard practice in the anglophone countries, the full wingspan $b$ was chosen as lateral reference length. As usual (e.g. [16], chapter 7), the non-dimensional
angular rates feature an additional factor 2 in the denominator and are defined as

\[
p^* = \frac{p}{2V_A} \quad q^* = \frac{q}{2V_A} \quad r^* = \frac{r}{2V_A}
\] (4.10)

Dependencies on \(\dot{\alpha}\) are dropped as they require special maneuvers to be separated from \(q\) dependencies and are usually expected to be small compared to the latter. Thus the \(q\) derivatives in fact represent the sum of the \(q\) and \(\dot{\alpha}\) derivatives.

Induced drag is captured by the \(C_{DC2}\) derivative. This term sometimes appears in simple quadratic polars, but more frequently the Oswald efficiency factor \(e\) is used for this purpose. \(e\) cannot be used directly as parameter of the PID algorithm because it occurs in the denominator, thus the model would not be linear in the parameters any more. The two variables are interconnected by the relation \(C_{DC2} = \frac{1}{\pi \Lambda e}\).

Sideforce is expressed in the wind axes, i.e. the crosswind force coefficient \(C_c\) is used rather than \(C_Y\). This choice is expected to yield a larger linear regime for the sideforce with varying angle of sideslip. The bias terms like \(C_{C0}\) in the equations for the lateral-directional parameters should in theory always be zero (and were so in the reference data set). However, in practice it is impossible to build a perfectly symmetric flight test aircraft with a perfectly aligned air data probe. Thus small biases are always present. The inclusion of these terms in the model for a PID algorithm is expected to significantly improve the accuracy.

The model adopted here is fairly generic but covers the most important flight mechanical characteristics of typical fixed-wing aircraft. It can easily be adopted to special cases by including more quantities in the input vector and assigning corresponding parameters to the coefficients. Just as easily, parameters may be dropped, e.g. if they are insignificant for a certain coefficient. However, computing requirements increase disproportionately with the number of inputs, so a number of 6 independent variables appears to be the maximum feasible for the PID algorithm on the given hardware.

### 4.3.2 Reference values

For simulation purposes, all parameters from equations 4.4 to 4.9 were assigned a reference value. The resulting model was used in simulation for PID evaluation tasks, which means that in theory no modeling errors could appear. The scalar derivatives were calculated from the multi-dimensional parameter arrays from table 4.1 by linearizing around a trim point corresponding to steady horizontal flight with 15 m/s at an altitude of 200 m ISA, corresponding to an angle of attack of \(\alpha = 2^\circ\) (\(C_L = 0.46\), Reynolds number about 200 000). Derivatives with respect to \(\dot{\alpha}\) were subsumed with the \(q\) derivatives. Some derivatives exhibit a strong dependence on the angle of attack or the lift coefficient, e.g. \(C_{lr}\) and \(C_{np}\), but they were nevertheless reduced to a single value. In case of the lateral cross-couplings this is expected to be a neglectable error because of their minor
influence, but notably the basic pitching moment curve as measured in the wind tunnel was also degressive, leading to a $C_{ma}$ derivative becoming more negative with higher AoA. This may serve as an explanation for the low stability margin of $-\frac{\partial C_m}{\partial C_L} = -\frac{C_m}{C_L} = 0.06$, which is rather small for an aircraft like the demonstrator, which is inherently very stable.

The reference values are given in table 4.2. The figure of $-11$ for $C_{mq}$ is in fact the sum of $C_{mq} = -9$ and $C_{m\dot{\alpha}} = -2$.

<table>
<thead>
<tr>
<th>$C_L$</th>
<th>$C_m$</th>
<th>$C_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>0.05</td>
<td>0</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>-0.3</td>
<td>$C_L^2$</td>
</tr>
<tr>
<td>$q$</td>
<td>-11.0</td>
<td>$\beta^2$</td>
</tr>
<tr>
<td>$\eta$</td>
<td>0.55</td>
<td>-1.2</td>
</tr>
</tbody>
</table>

(a) Longitudinal derivatives

<table>
<thead>
<tr>
<th>$C_C$</th>
<th>$C_l$</th>
<th>$C_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>-0.3</td>
<td>-0.08</td>
</tr>
<tr>
<td>$p$</td>
<td>-0.05</td>
<td>-0.5</td>
</tr>
<tr>
<td>$r$</td>
<td>0.04</td>
<td>0.08</td>
</tr>
<tr>
<td>$\xi$</td>
<td>0.0</td>
<td>-0.3</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>0.06</td>
<td>0.03</td>
</tr>
</tbody>
</table>

(b) Lateral derivatives

Table 4.2: Reference values for aerodynamic model

4.4 Inertial data

While the aircraft’s mass could easily be measured, its moments of inertia are not as accessible. Experiments to deduct them from the aircraft’s oscillating frequency when suspended as a pendulum did not produce reliable results, presumably because of considerable aerodynamic effects produced by the rotation. Thus, the final values had to be calculated. The aircraft structure was divided into several components (wings, various body parts, tails, ...) which were approximated as simple cuboids whose moments of inertia are given analytically. Installed equipment like engines, servos, sensors etc. was considered as point masses at their respective offset to the CG. The final values are given in table B.3.

The CG location was the one recommended by the manufacturer of the original airframe kit, which is 80 mm below and 80 mm behind the wing leading edge. As the aircraft is powered electrically, no shifts in CG occur during flight, so the CG location was also chosen as aerodynamic reference point.
4.4.1 Additional mass effect

One point that has to be considered when calculating the moments of inertia is the so-called additional mass effect (also called apparent mass effect), see e.g. [44]. A body rotating in a medium transfers momentum to that medium, so that angular acceleration is slower than it were in a vacuum for the same acting moment. This effect can be accounted for by using virtual moments of inertia consisting of the structure values plus an additional value calculated by the methods from [44] or [20].

For the demonstrator aircraft, calculation yielded an additional moment of inertia around the $x$ axis which was about 30% of the structure value. For the $y$ and $z$ axes, the relative magnitudes were 8% and 2%, respectively.

4.5 Analysis of the eigenmotions

Using the data from the previous sections, the time constants respectively frequencies and dampings of the characteristic eigenmotions of the demonstrator aircraft can be calculated. This can be done either using the classical approximation formulae given e.g. in [6], chapter 8.4 (note different reference lengths), or by direct numerical linearization of the simulation model to be described in the following chapter. As the eigenmotions are dependent on the flight state, the reference state from section 4.3.2 was chosen once again. Except for the phugoid and spiral motions, both methods are in very good agreement. The differences in the two mentioned motions can be explained by the cross-coupling derivatives that are not included in the approximation formulae. Table 4.3 shows the results of the calculations. The values given are those from direct numerical linearization. Note the very low value of 26 ms for the rolling motion time constant.

<table>
<thead>
<tr>
<th>Eigenmotion</th>
<th>Frequency in Hz</th>
<th>Damping ratio</th>
<th>Time constant in s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short period</td>
<td>1.5</td>
<td>0.82</td>
<td>Rolling motion</td>
</tr>
<tr>
<td>Phugoid</td>
<td>0.13</td>
<td>0.13</td>
<td>Spiral motion</td>
</tr>
<tr>
<td>Dutch roll</td>
<td>0.75</td>
<td>0.15</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Eigenmotion characteristics of the demonstrator aircraft at 15 m/s and 200 m ISA
5 Simulation

5.1 Simulation framework

The Paparazzi system comes with a built-in software-in-the-loop tool which is primarily intended to test pre-defined flight plans before actually flying them. However, this simulator is of the five degree of freedom bank-to-turn type and thus considers only trimmed flight states. Aircraft specific characteristics are subsumed into two values named yaw and roll response factors. While this is sufficient for flight plan testing, as test bed for parameter identification a full-fledged six degree of freedom simulation is necessary, which in turn requires a complete flight mechanics dataset. The Paparazzi system provided already basic support for the use of JSBSim \[4\] as such a simulator.

JSBSim itself is an open source flight simulation program which is used as the main simulation kernel e.g. in the FlightGear \[38\] simulator program but may also be run as a standalone module. It can be compiled into a shared library which the Paparazzi simulator program employs as simulation kernel. JSBSim allows to create arbitrary models for flight dynamics, propulsion, autopilot, actuators and other systems using XML files and may be run in real-time or batch mode.

Some extensions to both the Paparazzi system and JSBSim had still to be made to allow the integration of all needed functionality for PID testing. Among these are refined actuator and sensor models which had to be interfaced to the airborne code. The aerodynamic model for the demonstrator aircraft was created according to equations \[4.4\] to \[4.9\] using the values from table \[4.2\].

5.2 Component models

5.2.1 Autopilot

Building an autopilot model for simulation is comparably easy with the Paparazzi system. The main autopilot software – including additions like the PID algorithm – can be compiled for different build targets, among which is simulation as well as the hardware application. Most sufficiently high-level routines can use the same code in both cases. Low-level functions like e.g. PWM generation need to be adopted. The build system defines a special preprocessor macro SITL if the code is to be compiled for simulation which can be used to mask out certain code parts. Another approach is to
create a separate file for each build target and have the build system decide which one to compile.

As an example, for the five hole probe, the code to get the raw pressure sensor readings was separated into different functions for each build target, while the conversion to physical pressures, calculation of the flow angles and correction for induced speeds (see section 5.2.3 below) used the same code for simulation and final application.

### 5.2.2 Actuators

To allow for proper testing of the identification algorithms, the existing actuator model of the Paparazzi/JSBSim combination was not sufficient, as the control laws from the Paparazzi airframe could not directly be used in the JSBSim aircraft file. In this model, only the high-level commands for the pitch, roll, yaw and throttle channels were used as inputs for JSBSim. The resulting control surface deflections were calculated by JSBSim, which resulted in noticeable deviations from the Paparazzi control laws.

As a remedy, the Paparazzi code was altered to directly send the servo PWM pulse widths as input to JSBSim. In JSBSim, the servos were then modeled as lookup tables resolving the pulse widths into actual surface deflections. The data for these tables could be measured at the actual demonstrator aircraft. The servo models were completed by a second order delay and dead time element each, representing the mechanical travel with finite speed and the processing time for the PWM signal.

The resulting transfer function was thus of the form

\[
\delta(s) = K \cdot \frac{\omega_0^2}{s^2 + 2D\omega_0 + \omega_0^2} \cdot e^{-sT_d} \cdot t_{PWM}(s) \tag{5.1}
\]

where \(K\) represents the lookup table and servo tests in the Institute’s servo test bed \[30\] yielded the constants \(\omega_0\), \(D\) and \(T_d\). \(t_{PWM}(s)\) is the PWM pulse width (also called duty cycle). Such a model was used for each actuator both in the simulation kernel to simulate the actual control surface behavior as well as in the PID algorithm to get estimate for the deflection as input value.

Some characteristics of real servos could not addressed with this model:

- The dead time of a PWM-controlled servo may vary from the minimum duty cycle duration to the whole PWM frame width, typically from 1 to 20 ms. In this simulation, only a fixed value could be used which was chosen to be 10 ms.

- The transfer function parameters \(\omega_0\) and \(D\) may vary, especially with the size of the commanded step and with the acting external load. Such a variation is not supported by the simulation.
• There is a maximum deflection speed for real servos, while a transfer function like eq. [5.1] in theory allows infinite speeds. Therefore, the resulting deflection speeds were analyzed after the simulation runs and found not to exceed the real servo’s capabilities. In general, the maximum deflection speed is dependent on the servo’s supply voltage. Here, such a variation did not have to be taken into account because the servos were fed by the motor controllers’ battery elimination circuits which are stabilized at 5 V.

5.2.3 Sensors

The main sensors in use are the air data probe and the INS. For both the degradation of the measured signal with respect to the true state was modeled.

In case of the angular rates, the degradation was considered to consist only of additional Gaussian noise. The variance of the noise was measured with the original INS in static desktop tests. As the INS does its own on-board processing to compensate sensor biases and temperature-induced variations, the measured rates can be expected to be bias-free. Lag of inertial sensors is usually so small that it can be neglected. Angular rates are the same anywhere on a rigid body, so compensation for an off-CG sensor location is not necessary.

The translational accelerations do vary with the sensor offset, however; thus for them the additional influence had to be modeled. The governing equation is eq. [2.33] but in simulation of course \( \mathbf{a}_{K,CG}^f \) is known while \( \mathbf{a}_{K,\text{sen}}^f \) has to be calculated. Lag and biases were neglected for the same reasons as with angular rates, and noise was considered to be Gaussian here as well with variances measured in static tests.

Flow angle sensors like the air data probe are also subject to degradation due to sensor offset. This is caused by additional induced velocities at the sensor location \( \mathbf{r}_{\text{sen}}^f \) given by

\[
\mathbf{v}_{A,\text{sen}}^f = \mathbf{v}_{A}^f + \mathbf{\omega}_{K}^f \times \mathbf{r}_{\text{sen}}^f.
\]  

Flow angles at the sensor can be calculated from the cartesian velocity components in the usual manner, e. g.

\[
\alpha_{\text{sen}} = \arctan \frac{w_{A,\text{sen}}^f}{u_{A,\text{sen}}^f}.
\]  

Because of induced velocities, the dynamic pressure at the probe head may be higher than that of the free stream. However, as this effect is at least partially compensated by the sensitivity of the total pressure orifice decreasing for off-axis inflow (see eq. [3.1]), it is not included in the model.

Additionally, as the pressure sensors are situated in the fuselages at the end of pressure hoses coming from the probe head, the lag arising from pressure changes traveling
through the hoses must be taken into account. According to [28], chapter 5.4, this lag has an acoustic and a viscous component. For the given flow angle probe, the viscous drag is insignificant, while the acoustic component amounts to a delay time constant of $\tau = 1.5 \text{ ms}$, which is still quite low.

Another peculiarity is that body-fixed sensors like the probe at hand, as well as most wind vanes, do not measure the aerodynamic angle of sideslip $\beta$ (pivoting around the $x^a$ axis) but the so-called flank angle $\beta_f$, pivoting around the $x^f$ axis. These two diverge from each other with increasing angle of attack $\alpha$, see e.g. [32] for further discussion.

To model all the effects listed above, a dedicated flow angle sensor class for JSBSim was written and included in the simulation framework.

As mentioned in section 3.3.1, calibration tests of the installed five hole probe had been conducted in the Institute’s wind tunnel. This allowed to simulate the actual measured I2C values as function of airspeed, angle of attack and angle of sideslip being sent over the interface, so that the same sensor driver code, including the corrections for the effects mentioned above, could be used for simulation and flight test.

### 5.2.4 Propulsion

Two different propulsion models were used. In the beginning, the classical JSBSim approach was used, which consists of splitting the propulsion chain into an engine and a thruster, in this case a propeller. The propeller is characterized by lookup tables for its thrust and power coefficients which have to be given as function of the advance ratio. To produce these data, the demonstrator aircraft’s propellers were roughly measured in terms of chord and incidence angle at several radial stations. These dimensions served as input for the JavaProp tool [22] which calculated the required tables. Up to the time of writing this thesis, the electrical engine model of JSBSim is rather rudimentary and consists of multiplying the maximum available power by the normalized throttle setting.

The analytic model however showed significant differences to the model derived in section 4.1.2, causing convergence problems in the PID algorithm. Therefore, a new model was introduced, which used JSBSim’s external force capability. From the wind tunnel measurements, the force increments $X_f^F$ and $Z_f^F$ were extracted and tabulated as functions of throttle setting, angle of attack and airspeed. Pitching moment was considered by having the forces attack at the propeller location. Lateral effects had to be neglected as they were not available from the wind tunnel measurements. This lookup table model was used in all simulation runs mentioned below.
5.3 Flight plan

For the simulation runs, a simple flight plan was defined carrying out a short flight at a nearby R/C model airfield. It comprised takeoff, climb to maneuvering altitude, a single figure eight maneuver during which multistep maneuvers of the modified 1123-type (see e.g. [23], chapter 2.III.B) were carried out (one in elevator, aileron and rudder, respectively), a descending circle to approach altitude and landing. This flight plan produced the high-level commands which the autopilot translated into control surface deflections. A trajectory plot of the flight is given in figure 5.1.

Time histories of the flow speed and angles, angular rates and control settings, i.e. the input values for the PID algorithm, can be found in figures 5.2 to 5.4. In the latter figure, the control surface deflections are normalized with the values from table B.2. From this figure, it can be seen that during transitions from one flight phase to another the elevator and aileron deflections are of the same order of magnitude as the maneuver deflections, while the only noticeable rudder deflection comes from the corresponding maneuver. Note that the final approach is carried out at a relatively low airspeed and high angle of attack, which is on the border of the validity of the five hole probe calibration. During this phase, control activity also cedes, leaving little information left for PID.

Because the identification is not restricted to the maneuvering but continues over the whole flight, correlation of the inputs is hardly avoidable. The correlation coefficients of the input signals are shown in table 5.1. The highest values occur for $p$-$\xi$ and $q$-$\eta$, indicating that the aircraft reacts promptly to pitch and roll commands. A value of more than 90%, as seen here for $p$ and $\xi$, will generally (although not necessarily) affect parameter estimation (see e.g. [23], chapter 11.II). Concerning yaw, the correlation between $r$ and $\zeta$ is much smaller because rudder deflection mainly happens during entering and exiting turns to overcome adverse yaw, while prolonged phases of high yaw rate appear during steady turns, where control surface deflections are low. Noticeable values in the longitudinal motion between $\alpha$, $\eta$ and $\delta_t$ hint at the change of trim angle of attack that accompanies a change in throttle setting. In the lateral motion, the adverse yaw can be found in the $\beta$-$\xi$ correlation.

5.4 PID simulation runs

Simulations have the advantage of knowing the "true" states and parameters, which are unavailable in real experiments. This allows algorithms to be tested and tuned. All data relevant for the PID task was logged by the simulation software both in true and measured (according to the simulated sensors) form. The PID algorithm could thus be tested by running it on different combinations of input data, starting with only true values and subsequently replacing true by measured quantities.
5 Simulation

Figure 5.1: Trajectory of the simulation flight plan

Figure 5.2: Inflow characteristics of the simulated flight
5.4 PID simulation runs

Figure 5.3: Angular rates of the simulated flight

Figure 5.4: Control settings of the simulated flight
5 Simulation

<table>
<thead>
<tr>
<th></th>
<th>$\alpha$</th>
<th>$q^*$</th>
<th>$\eta$</th>
<th>$\delta_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1.00</td>
<td>0.34</td>
<td>-0.27</td>
<td>-0.46</td>
</tr>
<tr>
<td>$q^*$</td>
<td>0.34</td>
<td>1.00</td>
<td>-0.83</td>
<td>0.28</td>
</tr>
<tr>
<td>$\eta$</td>
<td>-0.27</td>
<td>-0.83</td>
<td>1.00</td>
<td>-0.47</td>
</tr>
<tr>
<td>$\delta_t$</td>
<td>-0.46</td>
<td>0.28</td>
<td>-0.47</td>
<td>1.00</td>
</tr>
</tbody>
</table>

(a) Longitudinal inputs

<table>
<thead>
<tr>
<th></th>
<th>$\beta$</th>
<th>$p^*$</th>
<th>$r^*$</th>
<th>$\xi$</th>
<th>$\zeta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>1.00</td>
<td>0.45</td>
<td>0.23</td>
<td>-0.68</td>
<td>-0.32</td>
</tr>
<tr>
<td>$p^*$</td>
<td>0.45</td>
<td>1.00</td>
<td>-0.13</td>
<td>-0.94</td>
<td>-0.46</td>
</tr>
<tr>
<td>$r^*$</td>
<td>0.23</td>
<td>-0.13</td>
<td>1.00</td>
<td>0.11</td>
<td>-0.09</td>
</tr>
<tr>
<td>$\xi$</td>
<td>-0.68</td>
<td>-0.94</td>
<td>0.11</td>
<td>1.00</td>
<td>0.51</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>-0.32</td>
<td>-0.46</td>
<td>-0.09</td>
<td>0.51</td>
<td>1.00</td>
</tr>
</tbody>
</table>

(b) Lateral inputs

Table 5.1: Correlation coefficients of PID inputs

5.4.1 Noise covariances

The comparison of true and measured values from simulations runs allows to assess the expected levels of measurement noise in the various quantities. This helps selecting the tuning parameters of the PID algorithm for the flight test.

For the force coefficients, noise arises from the measurement noise of the accelerometers, but also, as the coefficients are expressed in the aerodynamic rather than in the body-fixed axes, from uncertainties in the measured flow angles. The latter consist of measurement noise of the pressure transducers, but also of modeling errors of the five hole probe calibration. In $C_D$ and $C_L$, errors of the propulsion model add as well. These influences move the spectrum away from whiteness and add power in the lower noise frequencies.

Noise covariances of the moment coefficients are dominated by the measurement noise of the angular rate gyro, amplified by numerical differentiation and multiplied by the corresponding moment of inertia. The latter fact explains the difference between the $C_l$ and $C_n$ noise levels. $C_m$ is again even worse because of the thrust model. Control surface deflection noise consists of a delay between true and measured values because $T_d$ from eq. 5.1 could only be approximated in multiples of the main loop period.

It is worth noting that although the noise of the IMU accelerometers and gyro is isotropic, i.e. it has the same power in all three dimensions, the noise in the coefficients is far from isotropic.

Table 5.2 shows the resulting noise standard deviations from a reference simulation run. They were calculated as the sample standard deviation of the difference between the time histories of simulated and true values. Note these need to be squared for use in the covariance matrices.
Table 5.2: Noise standard deviations $\sigma$ in simulation runs

<table>
<thead>
<tr>
<th>Coefficients</th>
<th>Longitudinal inputs</th>
<th>Lateral inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_D$</td>
<td>$\alpha$</td>
<td>$\beta$</td>
</tr>
<tr>
<td>0.0046</td>
<td>0.22°</td>
<td>0.19°</td>
</tr>
<tr>
<td>$C_C$</td>
<td>$q^*$</td>
<td>$p^*$</td>
</tr>
<tr>
<td>0.00092</td>
<td>$8.5 \times 10^{-5}$</td>
<td>0.00065</td>
</tr>
<tr>
<td>$C_L$</td>
<td>$\eta$</td>
<td>$r^*$</td>
</tr>
<tr>
<td>0.036</td>
<td>0.16°</td>
<td>0.00035</td>
</tr>
<tr>
<td>$C_l$</td>
<td></td>
<td>$\xi$</td>
</tr>
<tr>
<td>0.00044</td>
<td></td>
<td>0.13°</td>
</tr>
<tr>
<td>$C_m$</td>
<td></td>
<td>$\zeta$</td>
</tr>
<tr>
<td>0.0046</td>
<td></td>
<td>0.13°</td>
</tr>
<tr>
<td>$C_n$</td>
<td>0.0012</td>
<td></td>
</tr>
</tbody>
</table>

For the PID algorithm, the measurement covariances $r$ were chosen as the covariance of the difference between true and measured value histories of the respective coefficient as listed in table 5.2. For the process noise covariance matrix $Q$, the diagonal elements were chosen as the covariance of the difference between true and measured value of the respective input, multiplied by $\Delta t^2$. The elements corresponding to bias derivatives remained at zero.

The choice of the $Q$ elements proved not to be critical. If in doubt, they should be chosen rather to small than too large, possibly even by 2 or 3 orders of magnitude. The value mentioned above appears to be a threshold below which the results remain nearly the same. Too large a choice of $Q$ leads to strong oscillations of the parameter time histories. For the $r$ values, there is no such threshold; however, within an order of magnitude of the true value, the algorithm is only affected very little as well. An $r$ value too small will produce oscillations because the algorithm tries to track random noise, while too large a choice will slow down convergence and introduce non-vanishing deviations from the true value, because the validity of incoming data is underestimated.

### 5.4.2 Coefficient buildups

According to the model from section 4.3.1, each coefficient is built up as a sum of different contributions caused by separate influence variables. The relative size of these contributions (e.g. $C_{L_0}$, $C_{L_\alpha}$), in comparison to each other as well as to the coefficient’s output noise standard deviation, is crucial for the PID process. A derivative’s contribution to the coefficient must be distinguishable from other derivatives’ contributions and from random measurement noise to make this derivative identifiable.

As an example, figure 5.5 shows the time histories of the contributions to the drag coefficient $C_D$. The coefficient’s measurement noise standard deviation is also given as black dashed lines. The root mean square values of these time histories for all coefficients are given in table 5.3. True parameters and input values were used for these figures. From the graph, it is obvious that the drag due to elevator deflection (purple curve) never exceeds the level of the expected measurement noise. Therefore it is not to be expected that the $C_{D\eta}$ derivative can be identified with the given setup. Drag due to sideslip (green curve) also remains below the noise level for most of the time, but during maneuvering, some peaks are visible which indicate a significant contribution at least
Figure 5.5: $C_D$ buildup

<table>
<thead>
<tr>
<th>Bias</th>
<th>$\alpha$</th>
<th>$q$</th>
<th>$\eta$</th>
<th>$\sigma^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_L$</td>
<td>0.30</td>
<td>0.34</td>
<td>0.013</td>
<td>0.015</td>
</tr>
<tr>
<td>$C_m$</td>
<td>0.050</td>
<td>0.022</td>
<td>0.023</td>
<td>0.033</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bias</th>
<th>$C_D^2$</th>
<th>$\beta^2$</th>
<th>$\eta$</th>
<th>$\sigma^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_D$</td>
<td>0.060</td>
<td>0.027</td>
<td>0.0072</td>
<td>0.00068</td>
</tr>
</tbody>
</table>

(a) Longitudinal coefficients

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>$p$</th>
<th>$r$</th>
<th>$\xi$</th>
<th>$\zeta$</th>
<th>$\sigma^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_C$</td>
<td>0.013</td>
<td>0.00072</td>
<td>0.00052</td>
<td>—</td>
<td>0.00085</td>
</tr>
<tr>
<td>$C_l$</td>
<td>0.0034</td>
<td>0.0072</td>
<td>0.0010</td>
<td>0.0090</td>
<td>0.00043</td>
</tr>
<tr>
<td>$C_n$</td>
<td>0.0013</td>
<td>0.00043</td>
<td>0.00065</td>
<td>0.00060</td>
<td>0.0010</td>
</tr>
</tbody>
</table>

(b) Lateral coefficients

1 Measurement noise standard deviation, copied from table 5.2

Table 5.3: RMS values of the contributions to the different coefficients
during these short times. Zero-lift and induced drag are well above the noise level and are also of comparable order of magnitude. Thus, these derivatives should not pose significant identifiability problems.

Further evaluation of table 5.3 and the graphs for the other coefficients as given in appendix C.1 shows that lift is dominated by the bias and $\alpha$ contributions, while $q$ and $\eta$ fractions are much smaller and rise only slightly above the noise level during the elevator maneuver. In the pitching moment, all contributions are of similar order of magnitude and well above the noise level. The only significant contribution to the crosswind force on the other hand comes from the angle of sideslip $\beta$, all other contributions to this coefficient are smaller by a factor of about 20. Considering the rolling moment, the cross-coupling caused by $C_{lr}$ appears rather small and the contribution due to rudder deflection $\zeta$ is below the noise level except for some small excursions during the rudder maneuver. The case of the yawing moment is special because the output noise covariance here is so high that the RMS values of most contributions are below. However, during maneuvering, all contributions rise significantly above this level.

In summary, it is to be expected that the following derivatives cannot be identified from the reference flight using noisy measurements: $C_{Lq}$, $C_{Ln}$, $C_{Dq}$, $C_{Cr}$, $C_{C\zeta}$ and $C_{l\zeta}$.

![Figure 5.6: PID results for lift coefficient derivatives for true input values. Solid lines: EKF, dotted lines: RLS](image)
5.4.3 True input values

PID results for true input values (i.e. both the coefficients and the independent variables have exactly the values that were used in the simulation kernel) are shown exemplary for the lift and yawing moment coefficient parameters in figures 5.6 and 5.7. Note that only the first second of the simulation run is shown, because convergence finishes during this timespan.

The solid lines show the time histories of the derivatives identified by the algorithm as stated in equations 2.23 to 2.25 referred to in the following as EKF algorithm. The noise covariances (Q matrix and r) were set to zero, reflecting the fact that there are no uncertainties in the “measurements”. All parameters were set to zero initially. For comparison, the dotted lines show the time histories of the derivatives for the RLS algorithm (equations 2.26 to 2.28) with a forgetting factor of $\lambda = 0.987$, which was experimentally found to give the fastest convergence without becoming unstable. Although the RLS algorithm does converge, it takes much longer than the EKF. For most derivatives, it needs about 15 seconds.

The EKF algorithm instead takes only 0.05 s in the longitudinal motion and 0.15 s in the lateral motion to converge. Within a second, the relative area of absolute error (eq. 2.43) of all derivatives is better than $10^{-5}$, except for $C_{D\beta}$, which requires 8 seconds to reach this precision. Sufficient information is available from the very beginning, because the simulation does not commence with a trimmed state but rather sets full throttle and a pre-defined airspeed. This causes a strong variation in the angle of attack and a small

![Figure 5.7: PID results for yawing moment coefficient derivatives for true input values. Solid lines: EKF, dotted lines: RLS](image-url)
dutch roll from the very beginning which serves already as a useful maneuver for PID input.

The EKF algorithm was also run with the covariance settings for the standard scenario including sensor noise according to section 5.4.1 as an assessment for its robustness. In this case, convergence did also occur, but took much longer. Most derivatives only converged after the corresponding multistep maneuver had been carried out. Some secondary derivatives took even longer (worst convergence: $C_{Lq}$). For the major influences, the finally reached relative area of absolute error was the same as with zero covariances, but for the secondary ones, it was considerably higher, topping at $1.3 \times 10^{-2}$ for $C_{Lq}$.

### 5.4.4 Using sensor models

To assess the algorithm performance, an usual approach would be to subsequently add noise to one measured quantity after the other, possibly even starting with lower noise levels. There are however some drawbacks to this approach. Firstly, this leads to different time alignments of the measured and true signals. Because of the lag introduced by the sensors and the subsequent filtering, the deteriorated signal will stay behind the other ones, affecting the PID algorithm much more severely than noise only would. Another difficulty in this particular project is posed by the need to invert the pressure sensor calibration model for the I2C bus if different sensor noise levels are to evaluated. The only feasible amelioration with respect to the flying code would be to use the true thrust values instead of the model from section 5.2.4. This has been tried but found to improve the results only marginally, so these results will not be shown.

**Analysis of the results** Figures 5.8 to 5.13 show the time histories for all derivatives during a representative simulation run. The dashed lines indicate the true values, i.e. the reference values of the derivatives that were used for simulation. Initial values for all derivatives were again zero. It is obvious that the effects of the noise are not the same to different coefficients. $C_m$ is affected the least, as the buildup of this coefficient is the most favorable. The convergence phase is increased with respect to the true values case, but with the onset of the elevator maneuver, all derivatives approach their true value and remain close to it until the end of the run.

Regarding $C_L$, section 5.4.2 predicted $C_{Lq}$ and $C_{L\eta}$ to be unidentifiable, which is confirmed by the $C_{Lq}$ value diverging indefinitely. $C_{L\eta}$ remains nearly constant during the figure eight and descending circle phases, but at a strongly biased value. As an interesting aside, $C_{L\alpha}$ only converges after the rudder maneuver. Via the $C_{D\beta^2}$ derivative, this maneuver couples into the longitudinal motion, and the natural stability leads to an increase in angle of attack to overcome lift lost because of decreasing airspeed. This causal chain decouples a change in $\alpha$ from $\eta$, improving the identification of $C_{L\alpha}$. It is also visible that after a phase of convergence, the derivatives start to drift away, beginning at about $t = 75$ s. Shortly before, the aircraft has started its final glide for
Figure 5.8: PID results for $C_L$ derivatives, simulation including sensor models

Figure 5.9: PID results for $C_m$ derivatives, simulation including sensor models
5.4 PID simulation runs

Figure 5.10: PID results for $C_D$ derivatives, simulation including sensor models

Figure 5.11: PID results for $C_C$ derivatives, simulation including sensor models
Figure 5.12: PID results for $C_l$ derivatives, simulation including sensor models

Figure 5.13: PID results for $C_n$ derivatives, simulation including sensor models
landing, during which the speed drops down to about 11 m/s. However, the calibration of the five hole probe tends to over-estimate the current angle of attack in this flight regime, regardless of whether the simple or complex calibration model is used. This leads to the noticeable drop of $C_{L\alpha}$, which in turn is compensated for by an increase in $C_{L0}$.

The effect of the rudder causal chain can also be found in the drag derivatives $C_{C^2}$ and $C_{D\beta^2}$. Zero lift drag $C_{D0}$ as the largest contributor is identified already earlier. $C_{D\eta}$, unidentifiable as predicted, does not attain a steady state.

The crosswind force coefficient is dominated by the $\beta$ contribution, resulting in early convergence of $C_{C\beta}$. Its other derivatives show some transient oscillations until the rudder maneuver provides enough information to get at least a rough estimate of $C_{Cp}$ and $C_{C\zeta}$, while $C_{Cr}$ as the smallest contributor cannot be identified.

For $C_l$ every derivative maintains a deviation to the true value. While convergence is generally achieved, the derivatives maintain an offset to the true value. This is caused by the correlation problems described below.

$C_n$ is again relatively good, with convergence starting with the first turn entering the figure eight while a steady state is only reached after the rudder maneuver.

**Correlation problems** Regarding $C_n$ and especially $C_l$, another problem besides the noise comes to the fore. From table 5.1 it is visible that the correlation between $p^*$ and $\xi$ over the whole of the flight is around 94%. Even reducing the timespan for analysis to the aileron maneuver only, the value stays over 92%. This correlation is caused by the very low rolling motion time constant ($T_R = 26$ ms according to table [4.3]) in comparison to the main PID frequency (60 Hz in simulation, cf. $\Delta t = 16.7$ ms). In other words, rolling speed caused by aileron deflection builds up so quickly that the effects of both can hardly be separated. As effect, the derivatives with respect to $\xi$ and $p^*$ are consistently underestimated. For $C_l$, these two are the dominating contributions, which affects the convergence of the remaining $C_l$ derivatives negatively. For $C_n$, at least $C_{n\beta}$ and $C_{n\zeta}$ remain identifiable.

To alleviate the correlation problems, either the PID algorithm period has to be decreased or $T_R$ has to be increased. According to

$$T_R \approx -\frac{I_{xx}}{2V S b \frac{b}{2} C_{lp}}, \quad (5.4)$$

this is possible by increasing $I_{xx}$ or decreasing $C_{lp}$. As changing the PID algorithm period in simulation would require substantial changes to the code (many parts rely on the 60 Hz because of hard-coded formulations), additional simulation runs with increased $I_{xx}$ were carried out. These exhibited improvements especially in the $C_l$ derivatives, which corresponded to the magnitude of $I_{xx}$. 

61
A listing of the relative areas of absolute error (RAAE, see section 2.5) for all derivatives is given in table 5.4. This summarizes the above findings. The results indicate that with the configuration at hand, some minor derivatives drop below the identifiability levels, but the algorithm should be able to converge on the others. Because of the low rolling motion time constants, problems with $p$-$\xi$-correlation are to be expected.

<table>
<thead>
<tr>
<th>$C_L$</th>
<th>5.4</th>
<th>0.17</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_m$</td>
<td>6.1</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 5.4: RAAEs (in percent) of derivatives in simulation run
6 Flight test

Figure 6.1: The demonstrator aircraft during final approach

Flight tests were carried out with the demonstrator aircraft during summer of 2011 on the Orsbach model aircraft airfield near Aachen, Germany. All take-offs and landings were carried out in manual mode. In some flights, the automatic modes were activated, but as the focus was on identification, only the basic autopilot functionality was verified by completing circles and ovals autonomously. During manual control, maneuvering was carried out consisting of varying unsteady inputs in all three control surfaces.

A total of 24 flights was completed with the full equipment and various versions of the working PID algorithm. Typical flight times were about ten minutes. All flights had to take place in moderate to strong winds as the airfield is located in a particularly windy spot. The last flight ended in a hard landing due to loss of R/C connection during low approach. No further flights were conducted because the five hole probe head was damaged and after repair, a new calibration is required.
The main control loop was run at a frequency of 60 Hz, while the PID algorithm had a frequency of 90 Hz. As no mass storage device was available onboard, all relevant data had to be send to the ground station via the telemetry link. This severely restricted the amount of available data. Data was transferred in the form of discrete messages being sent periodically, where the frequency of each message was specified independently. Table 6.1 gives an overview of the PID related messages in the data link configuration used during flight tests.

<table>
<thead>
<tr>
<th>Name</th>
<th>Frequency</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>INS</td>
<td>6.67 Hz</td>
<td>Position, $v^g_K$, $\Phi$, $\omega^g_K$, $\alpha^g_K$, $\omega^f_K$, $t$</td>
</tr>
<tr>
<td>Air data</td>
<td>6.67 Hz</td>
<td>$p_{stat}$, $p_{dyn}$, $\Delta p_{13}$, $\Delta p_{24}$, $V_A$, $\alpha$, $\beta$</td>
</tr>
<tr>
<td>Coefficients</td>
<td>6.67 Hz</td>
<td>$C_D$, $C_C$, $C_L$, $C_I$, $C_m$, $C_n$, $X^F$, $Z^F$</td>
</tr>
<tr>
<td>Derivatives</td>
<td>1 Hz</td>
<td>all parameters of equations 4.4 to 4.9</td>
</tr>
</tbody>
</table>

Table 6.1: Standard data link configuration

The low data rates make it impossible to post-process the data and apply posterior offline methods, especially as the different messages are sent alternatingly and the time stamping introduced upon reception at the ground station is not very accurate. To overcome this, four flights were carried out with a different configuration, where after completing a PID step, the PID input data was transmitted. The messages used for this combination are shown in table 6.2. At the full update rate of 90 Hz, it was only possible to send either the longitudinal or the lateral data message, and all other messages had to be reduced to frequencies below 1 Hz in order not to flood the data link. This means that for two flights, the longitudinal data is available and for the other two the lateral data. The derivatives message unfortunately was too long to be sent between two PID data messages, thus a direct comparison of post-flight and onboard analysis for the same flight was not possible.

| Longitudinal data | $C_D$, $C_L$, $C_m$, $\alpha$, $q^*$, $\eta$, $\delta_t$, $t$ |
| Lateral data      | $C_C$, $C_I$, $C_n$, $\beta$, $p^*$, $r^*$, $\xi$, $\zeta$, $t$ |

Table 6.2: Data link message contents for high update rate

6.1 Post-flight analysis

The data from the flights with the high update rate data could be used for post-flight analysis using batch methods as well as recursive methods. For batch analysis, only the equation error method could be used, as it can operate on the pre-processed data that was recorded (coefficients and independent variables). For the output error method, forces and moments need to be calculated, which was not possible because neither the dynamic pressure (to reconstruct forces from coefficients) nor the attitude (to calculate gravity direction) were available from the high update rate messages.

During post-flight analysis a problem surfaced concerning very strong periodic oscillations in the longitudinal force coefficients during phases flown at a throttle setting of around...
6.1 Post-flight analysis

60%. The likely cause is a resonating structural mode excited by the propellers, which has probably been measured in an aliased way. Similar vibrations are visible throughout the flight range with throttle settings above about 30%, although their magnitude is by far greatest around 60%. This amount was set for initial climb during the first 130 s of the first flight and for the first 50 s of the second one.

Noise standard deviations of the measured values are given in table 6.3 As no true values are known, these are derived from the difference of the original signal to one that was filtered by a 30-fold application of the binomial smoothing filter [31]. Usually these values are measured during ground tests, but as all values (except for the controls) are in some way dependent on air data, the figures in the tables are from the actual flight data. Atmospheric turbulence is thus included to a certain amount, but the PID algorithm will have to take this into account anyway, so the values can be used as a basis for covariance matrices.

These standard deviations compare to the noise covariance matrices $F$ and $G$ of section 2.1.2. Angular rates and moment coefficients have higher noise levels than in simulation, indicating that the gyro noise in flight is higher than simulated. This is probably also caused by the vibrations mentioned above, which cause a variation whose level is well above the noise measured in the static tests to produce simulation values.

<table>
<thead>
<tr>
<th>Coefficients</th>
<th>Longitudinal inputs</th>
<th>Lateral inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_D$</td>
<td>0.0015</td>
<td></td>
</tr>
<tr>
<td>$C_C$</td>
<td>0.012</td>
<td></td>
</tr>
<tr>
<td>$C_L$</td>
<td>0.033</td>
<td></td>
</tr>
<tr>
<td>$C_l$</td>
<td>0.0021</td>
<td></td>
</tr>
<tr>
<td>$C_m$</td>
<td>0.015</td>
<td></td>
</tr>
<tr>
<td>$C_n$</td>
<td>0.0021</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>0.35°</td>
<td>$\beta$ 0.37°</td>
</tr>
<tr>
<td>$q^*$</td>
<td>0.00023</td>
<td>$p^*$ 0.0031</td>
</tr>
<tr>
<td>$\eta$</td>
<td>0.21°1</td>
<td>$r^*$ 0.00071</td>
</tr>
<tr>
<td>$\delta_t$</td>
<td>0.005°1</td>
<td>$\xi$ 0.085°1</td>
</tr>
<tr>
<td>$\delta_t$</td>
<td>0.005°1</td>
<td>$\zeta$ 0.23°1</td>
</tr>
</tbody>
</table>

1 During manual control only (caused by the R/C transmitter), values in automatic modes are noise free (neglecting servo jitter)

Table 6.3: Noise standard deviations of measured values

6.1.1 Batch analysis – longitudinal motion

Figure 6.2 shows the total lift coefficient over angle of attack for a selected time range of 35 seconds during one of the flights. During this time, elevator maneuvering at constant throttle and roughly constant airspeed was carried out.

The red line corresponds to a linear fit for the range of $\alpha < 5°$, which was considered to be the angle of attack where stall sets on and the lift curve becomes nonlinear. This fit yields derivatives of $C_{L0} = 0.59$ and $C_{La} = 5.0$ (neglecting all other regressors). The very high value for $C_{L0}$ (the reference value of table 4.2 is 0.3) and the low stalling $\alpha$ indicate that the longitudinal axis of the five hole probe is inclined by about 7° with respect to the aircraft zero lift axis, which in turn was calculated to be at $\alpha \approx -3°$. The remaining 4° of deviation are attributed to imperfections caused by the manual fabrication of the
aircraft and installation of the nose boom. The maximum lift coefficient occurs at 8°, which combined with the 4° deviation matches the wind tunnel result well.

An overview of the longitudinal derivatives identified with the offline equation error method is given in table 6.4. The correlation coefficient of $q^*$ and $\eta$ for the considered time range as well as for the whole flight is $-0.93$ (cf. table 6.6), making the derivatives $C_{Lq}$ and $C_{L\eta}$ unidentifiable. For the value of $C_{Lq}$ in the table, $\eta$ as regressor was dropped. The value of $C_{Lq}$ is thus biased because it contains $\eta$ influences.

For the drag, it turned out that minimum drag does not occur at zero lift, but at a positive value of the lift coefficient. This is believed to be caused by fuselage drag, which has its minimum at a positive wing lift because of wing incidence angle and camber. To improve the fit of the drag model, a term linear in the lift coefficient was added, hence the $C_{DCL}$ value in the table. The minimum drag with this model is $C_D = 0.12$ at $C_L = 0.45$. As $\beta$ was not transmitted in the longitudinal analysis flights, the $C_{D\beta}$ derivative could not be addressed.

The pitching moment derivatives could all be estimated well from the given time range. A distinction of $C_{mq}$ and $C_{m\eta}$ is possible because of their greater relative contribution as described in section 5.4.2.

Additional derivatives with respect to symmetric aileron (“flap”) deflection (see also section 3.2.2) are listed in the table. During both high rate longitudinal data flights, symmetric aileron was used to vary the base-line derivatives and assess the PID performance with moving targets. The listed flap derivatives were identified by the equation

Figure 6.2: Scatter plot of lift coefficient for 30s of flight
6.1 Post-flight analysis

Table 6.4: Longitudinal derivatives identified from full rate data

Comparison of the identified values with the expected ones from table 4.2 shows a good match for lift and pitching moment. The deviations in the bias terms can be explained by the five hole probe angle as above. $C_L\alpha$ is found to be a bit higher (4.8) than expected (4.6). This results from the correlation issues (cf. table 6.6), as the $\alpha$-$q$ correlation of 0.80 is rather high as well. Thus the $\alpha$ derivative gets biased towards a more positive value. The $C_Lq$ derivative is smaller than the reference because of the negative $q$-$\eta$ correlation. If $C_Lq$ is dropped as well, we arrive at $C_L\alpha = 5.0$ as in the scatter plot fit.

Drag however is considerably higher than the reference value, both in terms of zero-lift drag and induced drag. This can be explained by additional installations on the flight test model, including sensors and structure augmentations. These also lead to a reduction in dynamic pressure at the tail, which may serve as an explanation for the elevator efficiency being lower than expected.

6.1.2 Batch analysis – lateral motion

Analysis of the lateral motion yielded that as well some derivatives were unidentifiable, similar to the longitudinal motion. This matches the conclusions from section 5.4.2 with a few exceptions: $C_{C\zeta}$ turned out to be identifiable, probably because its identified value is much larger than expected. $C_{lr}$ is on the verge of identifiability, and adverse yaw $C_{n\xi}$ was unidentifiable, probably because of the aileron control law. The inclusion of lateral biases significantly reduced the estimated error margin of the $\beta$ derivatives and $C_{\ell r}$, indicating asymmetries in the airframe. The results of the equation error method are shown in table 6.5.

Table 6.5: Lateral derivatives identified from full rate data

Here, deviations are more prominent than in the longitudinal motion. The larger derivatives of the crosswind force and yawing moment may be explained by the demonstrator
6 Flight test

<table>
<thead>
<tr>
<th></th>
<th>α</th>
<th>q*</th>
<th>η</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1.00</td>
<td>0.80</td>
<td>-0.65</td>
</tr>
<tr>
<td>q*</td>
<td>0.80</td>
<td>1.00</td>
<td>-0.93</td>
</tr>
<tr>
<td>η</td>
<td>-0.65</td>
<td>-0.93</td>
<td>1.00</td>
</tr>
</tbody>
</table>

(a) Longitudinal inputs

<table>
<thead>
<tr>
<th></th>
<th>β</th>
<th>p*</th>
<th>r*</th>
<th>ξ</th>
<th>ζ</th>
</tr>
</thead>
<tbody>
<tr>
<td>β</td>
<td>1.00</td>
<td>-0.42</td>
<td>-0.12</td>
<td>0.38</td>
<td>0.48</td>
</tr>
<tr>
<td>p*</td>
<td>-0.42</td>
<td>1.00</td>
<td>-0.08</td>
<td>-0.93</td>
<td>-0.82</td>
</tr>
<tr>
<td>r*</td>
<td>-0.12</td>
<td>-0.08</td>
<td>1.00</td>
<td>0.13</td>
<td>-0.17</td>
</tr>
<tr>
<td>ξ</td>
<td>0.38</td>
<td>-0.93</td>
<td>0.13</td>
<td>1.00</td>
<td>0.84</td>
</tr>
<tr>
<td>ζ</td>
<td>0.48</td>
<td>-0.82</td>
<td>-0.17</td>
<td>0.84</td>
<td>1.00</td>
</tr>
</tbody>
</table>

(b) Lateral inputs

Table 6.6: Flight measured correlation coefficients of PID inputs

aeroplane’s square fuselage cross section. Datcom assumes a more or less circular fuselage, while flat side areas are known ([2], chapter 4.2.1) to be more susceptible to sideslip as well as to increase vertical tail efficiency.

Concerning the rolling moment, the values are consistently below the references. This may on the one hand indicate that the value of the rolling moment of inertia has been estimated too low. Judging from the aileron efficiency $C_{l\xi}$, which can be calculated relatively easy and depends mainly on easily measurable geometric parameters, the calculated $I_{xx}$ is only 2/3 of the true value. A possible reason for this may be an underestimated additional mass effect (section 4.4.1) because of modeling errors and too much simplifications. On the other hand, in section 5.4.4 it was already seen in simulation that the rolling moment derivatives were consistently underestimated due to correlation between $p$ and $\xi$, so it is probable that the same effect arises here. In fact, the correlation of 0.93 is similarly high as in the simulation, so this is more probable as cause for the deviations.

To overcome this, higher execution rates of the whole measurement and calculation chain would be necessary, which is not possible with the given setup. One last point that might contribute to the deviations is the propulsion system. Large parts of the wing are in propeller slipstream, which means that both the additional angle of attack caused by rolling motion and the wing angle of sideslip are reduced compared to the bare wing. This may serve as an explanation for the smaller values of $C_{l\beta}$ and $C_{lp}$. As both propellers are rotating the same way, they also introduce a non-vanishing reaction moment which further complicates things. It can at least partly be absorbed by the $C_{l0}$ derivative.

6.1.3 Post-processing using the flying code

Using a Matlab implementation of the flying real-time PID code (see appendix A.1.2), the high-rate data was post-processed offline.

Output noise covariance settings $r$ for these runs corresponded to measurement noise standard deviations from table 6.3, while for the process noise $Q$ these values were multiplied by $\Delta t^2$ and additionally by the corresponding $r$ value to mitigate the strong
6.1 Post-flight analysis

Jitter in the parameter time histories produced by the original values. The choice of derivatives corresponds to the offline analysis, except that $C_l_\nu$ became unidentifiable and for $C_D$ a throttle derivative was added to capture errors in the propulsion model. A squared dependency on throttle setting is used because thrust increases with RPM squared, while RPM increases linearly with voltage which is effectively controlled by throttle setting, hence $C_{D\delta_\nu}^2$.

The lateral biases did not have the same beneficial effect as with the equation error method and were thus omitted from the online analysis.

Results are shown in figures 6.3 to 6.8. The blue and red curves refer to different flights. Initial values for all derivatives were zero. For one of the flights with lateral data, $C_n$ was not available because of a typing error in the flying code that was corrected for the second flight.

For the longitudinal derivatives, two different constellations were considered, one with an additional derivative for symmetric aileron deflection, one without. In the graphs, these are shown as solid and dashed lines, respectively. To avoid divergence in the initial phases of the flight, the flaps position $\delta_{a,sym}$ was set to an exact zero when below a minimum value of 2.5%. Thus, noise in the retracted position was eliminated, and the derivatives remain zero until the ailerons are lowered for the first time.

**Longitudinal motion** In the longitudinal motion, most derivative histories converge to a value corresponding more or less to table 6.4. In the lift coefficient channel, the $C_{L_0}$ derivative first takes on a higher value during initial climb (which was characterized by structural vibrations as described in section 6.1) and then drops to the final level corresponding to the batch result. A similar behavior is seen for $C_{L\alpha}$ and the pitching moment derivatives. The noise introduced by the vibrations impedes convergence during this phase. Starting the PID algorithm later (after the vibrations have ceded) speeds up the convergence process, especially for $C_L$.

The flap efficiency $C_{L_\delta a,sym}$ starts to converge quickly after the first deployment. Regarding the pitching moment, the situation is similar to the lift, but again improved by the possible distinction of $C_{m\eta}$ and $C_{m\eta}$ because of their greater relative contribution.

For all derivatives, convergence does not occur perfectly, nor do both flights produce an exact match. This cannot be expected, because the noise levels impose a border below which deviations will not be corrected by the algorithm. These deviations arise because between flights and even between different parts of the same flight, the surrounding conditions change, e.g. wind and turbulence, temperature, battery voltage and so on. As the flights were controlled by hand, strict repeatability of the maneuvering is not possible either.

When the flap derivative is not enabled, its effects are attributed to all available other derivatives, as can be seen from the dashed lines. This happens because the flap setting was adjusted only at discrete times and each of the adjustments entailed changes in
Figure 6.3: Post-flight EKF results: Lift coefficient derivatives
6.1 Post-flight analysis

Figure 6.4: Post-flight EKF results: Drag coefficient derivatives
Figure 6.5: Post-flight EKF results: Pitching moment coefficient derivatives
Figure 6.6: Post-flight EKF results: Crosswind force coefficient derivatives
Figure 6.7: Post-flight EKF results: Rolling moment coefficient derivatives
6.1 Post-flight analysis

Figure 6.8: Post-flight EKF results: Yawing moment coefficient derivatives
flight state, e.g. a lower trim angle of attack and stronger negative elevator deflection, resulting in correlated changes of regressors. Thus, $C_{L0}$ increases while $C_{L\alpha}$ decreases. In the drag channel, the flap effect is rather small. It is mostly attributed to the throttle derivative when no $C_{D\delta_{a,sym}}$ is enabled. In the pitching moment, the effects on $C_{m\theta}$ and $C_{mq}$ are comparable to the lift, but with the opposite sign, as flap deflection causes a lift increase, but nose down moment. $C_{mq}$ is strongly affected as well because of the change in trim deflection. The decrease in $C_{mq}$ in the second flight probably occurs because the magnitude of maneuvering decreased after flap deployment.

A throttle derivative was found to improve algorithm performance only in case of the drag. Test runs with $C_{L\delta}$ and $C_{m\delta}$ showed these derivatives to vary strongly and in an unpredictable manner, indicating an unidentifiable parameter as opposed to $C_{D\delta}$, which increases steadily over time. This behavior is expected, because battery voltage decreases with time, i.e. the propulsion model tends to overpredict the thrust, which in turn is compensated for by an increasing “virtual” throttle dependent drag.

**Lateral motion** For the lateral motion, many derivatives take steep jumps at the beginning of certain maneuver parts. For example, during the third flight the first significant rudder deflection occurs at around 240 s, while around 350 s steady-heading sideslip phases in both directions begin. This leaves noticeable traces in the crosswind force derivatives. Even stronger, in the second flight, the first rudder deflection is at 80 s. This leads to jumps in all lateral derivatives. Most of them remain relatively constant afterward. The crosswind force derivatives jump again at 310 s, when steady-heading sideslip is initiated. The crosswind force values before the sideslip match table 6.5 well. In the rolling moment, $C_{l\beta}$ is good, but $C_{lp}$ and $C_{l\xi}$ are both too small, hinting at the high $p$-$\xi$-correlation. Yawing moment values match the table again very well.

**RLS** Some additional post-processing runs employed the standard RLS algorithm. It was however not possible to find a forgetting factor that prevented the (identifiable) parameters from total temporary divergence while at the same time allowing them to converge within the total time of each flight. Therefore, no RLS results are shown.

### 6.2 Restrictions to avoid divergence

Several flights were carried out with the complete onboard PID system running. In the beginning, all parameters were included in the algorithm, which was working for the whole time of the flight. For subsequent flights, some parameters were deactivated so that the remaining set corresponded to tables 6.4 and 6.5. During the first test flights, some issues became visible that deteriorated the performance of the PID algorithm. They were addressed with the restrictions described below.

For example, the dynamic pressure sensed by the air data probe frequently dropped much below the values reached during calibration. This happened during stall tests, but also for short instances during maneuvering flight due to strongly off-axis inflow because
of angle of attack induced by pitching rate. The calibration formulae are not valid for these ranges; extrapolation produces wrong values, possibly even negative dynamic pressure. The dynamic pressure appears in the denominator of various calculations, namely the aerodynamic coefficients and the inflow angles (equations 3.7 and 3.8). These quantities in turn become much too large. In combination, these effects are sufficient to produce divergence of the PID algorithm.

To avoid this, the following restrictions were implemented in the onboard code:

- Dynamic pressure must be positive. If it is calculated to be below zero, set $p_{dyn} = 0$, $\alpha = 0$, $\beta = 0$, $V_A = 0.01$ m/s.
- Only calculate aerodynamic coefficients if $V_A > 7$ m/s.
- Limit aerodynamic coefficients to suitable values (see table 6.7).
- Only run an iteration of the PID algorithm if $7$ m/s < $V_A$ < $25$ m/s and neither NaNs nor infinities have occurred during the preceding calculations.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Range</th>
<th>Quantity</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{dyn}$</td>
<td>&gt; 0</td>
<td>$C_D$</td>
<td>-0.1 ... 0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C_L$</td>
<td>± 0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C_C$</td>
<td>± 0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C_m$</td>
<td>± 0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C_L$</td>
<td>-1 ... 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C_n$</td>
<td>± 0.5</td>
</tr>
</tbody>
</table>

Table 6.7: Allowable value ranges for various quantities of the PID algorithm

For the parameters, another restriction was evaluated to keep unidentifiable parameters within suitable bounds and limit their effect on the other ones. Their value was constrained to a range of -1 ... +3 times their reference value as given in table 4.2. Parameters whose reference value is zero were allowed a range of ±0.01. If during an iteration of the algorithm a parameter exceeded these borders, it was reset to its reference value and initial covariance. However, this restriction did neither prevent these parameters from oscillating, nor influence the other parameters positively; thus, it was dropped again.

### 6.3 In-flight real-time parameter identification results

In the course of the various flights, the PID algorithm was fine-tuned in terms of PID algorithm frequency, restrictions and covariance settings to improve performance. The values for the last flight are shown here as example. As this was the flight with the hard landing, the time histories are clipped at the point where the loss of control occurred. Because the high data rate flights had not yet been analyzed fully before this last flight,

\[1\] Not a number, an indication by the floating point processing routine that an illegal operation like e.g. division by zero has occurred
the covariance settings in use were still somewhat different from the post-flight findings and the PID algorithm performance was thus suboptimal as will be discussed below.

6.3.1 Input data

Graphs showing the recorded input data for this flight can be found in appendix D.1. The whole flight was piloted manually. It was carried out at medium to low airspeeds, as can be seen from the upper graph from figure D.1. During transients, the measured airspeed drops much below the stationary stalling speed, however, these drops are likely caused by inflow diverging strongly from the air data probe axis, which means that the front orifice pressure is not the total pressure any more.

As can be seen from the control surface deflections in figure D.2 and the angular rates in figure D.3 during the whole flight maneuvering was done. The flight starts with the climb to maneuvering altitude of about 50 m above ground, which lasts 25 s. Next, the pilot tries to establish a trimmed state, which is being hampered by wind gusts leading to the variations in airspeed around $t = 30$ s. This is followed by stall approaches around $t = 50$ s and then fast maneuvering in all controls 15 s later. In the time range from 90 to 120 s, steady-heading sideslip was performed. After stabilization, repeated sequences of continuous (steered) control oscillations follow, at $t = 140$ and 190 s in elevator and at $t = 160$ and 210 s in aileron.

Even though the control deflections were moderate and the dynamic pressure rather low, large variations in the flow angles resulted. No flap deflection occurred during this particular flight. The angular rates as shown are representative for most flights, however, in other flights with more aggressive maneuvering the maximum attainable rates were found to be around $400 \, ^\circ/\text{s}$ for $p$ and $200 \, ^\circ/\text{s}$ for $q$ and $r$, with the normal load factor peaking at around 5.

6.3.2 Identified parameters

The results of the PID algorithm as received via the downlink are shown in figures 6.9 to 6.14. Some derivatives have been multiplied with a factor 0.1 or 10 in order to display all derivatives of each coefficient in a single graph. Temporal resolution is lower than for the input data, as the derivatives were only sent with 1 Hz.

$C_L$ Looking at the $C_L$ graph, all three remaining derivatives appear to be far from convergence. On a second look however, at least $C_{L0}$ and $C_{L\alpha}$ oscillate around a mean value in the expected order of magnitude. From figure D.1 we see that $\alpha$ frequently rises above the expected stall onset angle of $5^\circ$, i.e. the lift coefficient leaves the linear range. This leads to the observed fluctuations in the derivatives, as the specified noise covariance was below the level of variation introduced by the nonlinearity. An additional derivative with respect to $\alpha^2$ could improve the situation here, albeit at the cost of deteriorating the fit in the linear range.
Even without the $C_{L\eta}$ derivative being enabled, $C_{Lq}$ does not converge. A comparison of its time history with the input data shown in appendix D.1 reveals that phases of strong changes in $C_{Lq}$ coincide with elevator maneuvering and strong changes in $\alpha$ and $q$. The $\alpha$-$q$ correlation in these phases can be expected to be even higher than the value of 0.80 found in section 6.1.1, while the information content of the phases in between is too small to provide compensation. The fact that the average airspeed on this flight was rather slow also contributes to this problem, because the average lift coefficient is very high, and its increase is achieved by higher angles of attack. This means that the relative contribution of $C_{L\alpha}$ is higher, while those of $C_{Lq}$ and $C_{L\eta}$ are smaller. All in all, this leads to $C_{Lq}$ becoming unidentifiable as well. With higher airspeeds however, the situation should be alleviated because the relative contribution of $C_{L\alpha}$ sinks.

$C_m$ Regarding $C_m$, the covariance in use was specified much larger than required in an attempt to compensate thrust model imperfections. This also absorbs large parts of the variations due to stall approach which were seen with $C_L$. Derivative histories with much less jitter are thus expected and can be seen in figure 6.11.

The $C_{m\eta}$ derivative increases slowly for the time from $t = 70\,\text{s}$ to $110\,\text{s}$, from a value around $-1.3$ to about $-0.6$. This covers the sideslip maneuver, during which also the angle of attack is even higher than in slow cruise. The derivative change is expected because elevator efficiency generally decreases for higher angles of attack, as the local lift curve at the elevator reaches its nonlinear range. In sideslip, additionally the leeward half of the elevator is shadowed from the free stream by the fuselage. This affects not only $C_{m\eta}$, but also $C_{ma}$, decreasing the efficiency of both horizontal stabilizer and elevator.

However, because of the high covariances, the algorithm attributes this partly to noise in the coefficient measurement and only slowly adjusts the parameter. The increased stability in the time histories is paid for by slower tracking of variations. Shortly before the sideslip, the angle of attack drops below $-5^\circ$ for the first time, causing $C_{m\eta}$ to sink before starting the slow climb. Another problem that arises here is that the $C_{ma}$ derivative increases and even becomes positive during the sideslip. According to the wind tunnel measurements, it should become more negative with increasing angle of attack. This is suspected to be an artifact of $\alpha$-$\eta$ correlation, but this cannot be stated for sure, as the available input data was sent in different messages and thus corresponds to different time steps, making the calculation of correlation coefficients impossible.

$C_D$ An example of quick parameter tracking can be seen with the drag. At $t = 75\,\text{s}$, $C_{D0}$ takes a jump from $0.08$ to $0.15$. As mentioned with the pitching moment, at this time (shortly before the sideslip starts), the angle of attack drops below $-5^\circ$ for the first time, providing data over a much wider $\alpha$ range and improving the fit of the drag parabola derivatives.

At the beginning of the sideslip maneuver, the $C_{D\beta^2}$ derivative shows an explicit convergence process, resembling an exponential decay of the difference to 0.85. The end of the
sideslip maneuver has no noticeable influence on the derivative.

Drag derivatives are better than expected, but still indicate a strong dependence on flight state, visible very well from the $C_{DCL}$ curve taking marked steps. When around $t = 140\,\text{s}$, the flight returns to maneuvering at moderate speed, the derivatives return from their sideslip excursion to values closer to those from table 6.4. The marked steps at flight state changes in combination with the relatively constant parts between indicate that the covariance settings for the drag were correct for this flight.

$C_C$ In the side force, the sideslip derivative $C_{C\beta}$ as strongest influence is identified quite stably, but the other derivatives vary strongly. At $t = 80\,\text{s}$, $C_{C\beta}$ jumps from $-0.5$ to $-0.55$. This jump coincides with the first dedicated rudder maneuver which – as well as the subsequent sideslip maneuver – comprises angles of sideslip of about 20°. This is the validity limit of the air data probe and also beyond the linear range of the sideforce curve, so a variation in the derivative can be expected. After the end of the sideslip, $C_{C\beta}$ returns to its old value, although the slow change indicates that the covariance was specified somewhat too high.

It is unclear why $C_{C\zeta}$ does not converge, as $C_{\zeta \zeta}$ and $C_{n\zeta}$ do. In most other flights, $C_{C\zeta}$ was identifiable much better; the cause for the exception in this particular flight could not be found.

$C_l$, $C_n$ Considering the rolling as well as the yawing moment derivatives, the relations of the results are comparable to table 6.5 but the values are consistently smaller. The discussion from section 6.1.2 applies here as well, i.e. that either the moment of inertia around the roll axis was underestimated or the correlation of the relevant inputs is too high, introducing biases in the derivatives. Concrete values for the correlation coefficients could not derived here either, for the same reasons as described in the paragraph on $C_m$.

However, the previous experience indicates that correlation is the main contribution to biases in this case as well. Again here, prominent changes at the beginning and the end of the sideslip maneuver are found, indicating either a change of the derivatives with the flight state or the leaving of the linear range.

The curves for $C_l$ indicate that the covariances were chosen well; for $C_n$, it is visible that $C_{np}$ and $C_{n\xi}$ vary strongly, hinting at their small influence. $C_{n\xi}$ was already found to be unidentifiable in section 6.1.2 $C_{nr}$ convergence also takes until after the sideslip maneuver, similar to the $C_l$ derivatives.

**General observations** The variation of many derivatives with flight state shows that the assumption of constant derivative does not hold over the entire flight envelope; most derivatives are dependent at least on the angle of attack, which again corresponds to elevator trim and throttle setting during stationary flight. In sideslip, the magnitude of the angle of sideslip is another major influence on the derivatives.
Because of the high noise levels, strong maneuvering is required to produce sufficient excitation for the PID algorithm. This in turn leads to large variations in the flow variables, which again may challenge the linear approach because most derivatives are only applicable to small variations.

The convergence process of the parameters is also strongly affected by the choice of noise covariances, as was seen with $C_L$ and $C_m$. Analysis of the previous flights also indicated that the covariances themselves varied between flights, most probably because of different atmospheric conditions. Thus it would be desirable to get noise covariance calibrations during flight, or at least precede a measurement flight with a calibration flight shortly before in the same conditions.

### 6.4 Comparison of the results

A tabular summary of results is given in appendix E. The tables show the values produced for each derivative with the different methods applied over the course of this thesis.

Looking at the Reference and Simulation columns of both tables, it is obvious that the algorithm is capable of re-extracting the reference values from the simulated measurement data even in the presence of noise, as already seen in detail in section 5.4.4. A comparison to the other columns yields that at least in the longitudinal motion, the flight test results are generally consistent among the different tests, although discrepancies to the reference values are visible. These comprise mainly the larger $C_{L0}$ due to the air data boom alignment and the higher drag due to the additional installations. Inconsistencies in the flight test results show up in $C_{Lq}$ (unidentifiable because of correlation), $C_{m0}$ and $C_{m\alpha}$ (unfinished convergence in in-flight real-time results because of too high covariances specified) and $C_{D\delta}$ (unidentifiable with the equation error method because of too little variation in the selected time range).

In the lateral motion, the situation is not quite as good. Deviations between flight test and reference are high in crosswind force and rolling moment derivatives. For the latter, this is attributed to input correlation, while for $C_{C\gamma}$, the quality of the reference data is more questionable. In the yawing moment, the match is better. The correlation also produces inconsistencies between the flight test results in the rolling moment. Yawing moment is again better; in the sideforce, $C_{C\zeta}$ was a problem in the selected flight only.
Figure 6.9: Lift coefficient derivatives

Figure 6.10: Drag coefficient derivatives
6.4 Comparison of the results

Figure 6.11: Pitching moment coefficient derivatives

Figure 6.12: Crosswind force coefficient derivatives
Figure 6.13: Rolling moment coefficient derivatives

Figure 6.14: Yawing moment coefficient derivatives
7 Summary, conclusions and outlook

7.1 Summary

Online PID algorithm  The parameter identification (PID) algorithm used in this work is a specialized version of the Kalman filter. It is restricted to a single output and resembles the formulation of the recursive least squares (RLS) algorithm. The main difference is that input and output noise covariances can be specified independently, replacing the function of the forgetting factor in RLS. This improves convergence speed without increasing the susceptibility to divergence. The drawback is that more parameters need to be tuned; however, these parameters are now based on physical properties of the signals. An optimized formulation in C language for use on microprocessors is given in appendix A.1.1.

A suitable choice for the output noise covariance \( r \) is the measured sample variance of the corresponding coefficient from a reference flight. The process noise covariances matrix \( Q \) should be diagonal, with the elements equaling the sample variance of the corresponding input signal multiplied by the squared time step of the algorithm and the corresponding output variance. Some manual tuning may however still give improvements.

Sensors for a demonstrator aircraft  An R/C model aircraft was equipped with an onboard processor (Paparazzi Tiny) and the required sensors (Xsens MTi-G INS, five hole probe on a noseboom) to allow assessment of the algorithm in flight. Calibration data for the five hole probe could be attained from wind tunnel tests with the original fuselage/sensor setup. During flight tests it was however found that the flight envelope had been underestimated. In maneuvering flight, effective airspeeds and flow angles exceeded the anticipated values by far because of induced flow at the probe location.

Two calibration models for the five hole probe were implemented: a simple one with linear dependency of the flow angles on the differential pressures, and a more complex one including terms of higher polynomial order. The second one was required because the linear approach turned out to be valid only for small flow angles at moderate to high speeds. Slow airspeeds and high flow angles led to considerable deviations of the measured values from the true ones.

The INS was interfaced as is with the onboard processor. Running its own strapdown algorithm, it provided bias-corrected translational accelerations and angular rates as inputs for the PID algorithm.
Control surface positions were not measured but calculated internally from the commanded deflections using an actuator model.

**Demonstrator aircraft reference data** To provide a reference model for simulation purposes, the demonstrator aircraft was tested as a half model in the Institute’s wind tunnel. This yielded static longitudinal characteristics as function of airspeed, angle of attack and various control settings. Using these as experimental input, dynamic derivatives were calculated using the Digital Datcom program. A non-linear dataset containing aerodynamics and propulsion characteristics was generated by consolidation of the above results.

This dataset was then reduced to a derivative-based model which allowed easy performance assessment of the PID algorithm. Linearization of the multi-dimensional data around a trim state provided the required derivatives which then remained constant.

Moments of inertia were obtained from calculations, taking into account the additional mass effect.

**Simulation** The existing Paparazzi simulation code was extended to allow development, assessment and tuning of the PID algorithm implementation. JSBSim served as six degree of freedom simulation kernel. Required extensions comprised sensor models for INS and air data boom as well as a much more detailed control surface interface. The linear aerodynamics model from above was integrated into the airframe model.

A reference flight plan was carried out in simulation with various configurations of the PID algorithm. Using true data without sensor noise, the algorithm proved to converge much quicker than RLS and be less susceptible to divergence. With simulated sensor noise, identifiability problems for some parameters showed up. This concerned the so-called minor influence derivatives. Breaking the total coefficients (e.g. $C_L$) up into their corresponding contributions ($C_{L\alpha} \cdot \alpha$ etc.) produced the relative contributions to all coefficients. $C_m$ was found to have all derivative contributions of similar magnitude, while $C_C$ was dominated by the $C_{C3}$ contributions. The other derivatives were in between these extremes. Combining these with the expected output noise allowed to assess the identifiability of the derivatives.

Another problem surfacing in simulation is the correlation of $p$ and $\xi$. The aircraft’s very small time constant of the rolling motion means it reaches its stationary rolling rate within a few time steps of the PID algorithm at the maximum achievable algorithm frequency on the given hardware. This leads to biased derivatives when both inputs are used on the same coefficient. A similar situation is found in the longitudinal motion concerning $q$ and $\eta$, which also affects $\alpha$, although less severely.

**Flight test** The demonstrator aircraft completed several test flights. During these, the lack of an onboard mass storage device and the limited telemetry data link capacity proved to be a severe restriction. In normal operation, PID input data and results
could only be recorded with a reduced rate and alternating values. To facilitate tuning of the algorithm in post-flight analysis, some flights were carried out where all PID inputs were transmitted with the full update rate. This could however only be done for longitudinal and lateral inputs separately. Results of post-flight analyses of these data are given in the text both for equation error method and the online algorithm as implemented on the flying hardware. The data indicated a structural vibration excited by the propulsion which affected PID performance because of the increased noise level. Some restrictions had to be applied to the allowable ranges of the input data to avoid divergence, especially at low airspeeds.

Parameters identified from flight test data generally matched the reference values fairly well in the identifiable derivatives. Deviations in the longitudinal biases can be attributed to misalignment of the nose boom. Higher drag resulted from additional installations not yet present in the wind tunnel experiment. Rudder efficiency was higher than expected, but the prediction was based on very rough formulae. The set of identifiable parameters corresponded to the simulation results. Correlation issues were also similar as in simulation, affecting mainly the rolling moment derivatives and the secondary lift derivatives. Because of the shortcomings of the data transmission, the exact correlation values could however not be calculated.

In the flights for longitudinal post-flight analysis, symmetric aileron lowering was used as a means of changing aerodynamic characteristics during flight. The resulting changes in lift and pitching moment coefficient led to variations in all derivatives of these coefficients, partly because correlation with elevator deflection and pitch angle was unavoidable.

Results of the online algorithm during actual flight are given for one flight. Covariance settings and parameter selection on this flight were not completely optimal, but as due to a hard landing the air data sensor was destroyed afterwards, no further flights could be undertaken. Because the flight was carried out at low airspeed, partly in the non-linear range of the lift curve, convergence was affected, especially for $C_L$. However, at least during the second half of the flight at higher speeds, convergence could be achieved for most derivatives, especially in the lateral motion. A survey of the results can be found in the tables of appendix E.

7.2 Conclusions

Real-time parameter estimation using low-cost hardware for mini aerial vehicles differs from conventional PID tasks in several aspects. On the one hand, the vehicle itself poses challenges. Its agility will be much higher than that of a larger one, as will be its susceptibility to atmospheric disturbances, which also will be higher themselves because of the lower operating altitudes. Depending on the configuration, classical approaches to aerodynamics modeling may have to be expanded, taking into account e.g. vortex lift for delta wings, cross-couplings between aerodynamics and propulsion because of large parts of the lifting surface being exposed to propeller slipstream or velocity-dependent Reynolds number effects.
The hardware on the other hand brings its own peculiarities. Noise of low-cost sensors is generally high, and possibly contains additional systematic sources of error because of different measurement principles. Suitable onboard processors suffer from limited computational power and especially lack of mass storage devices, making it difficult to perform post-flight analyses. Telemetry links can alleviate this problem only in a very limited way.

Still, the task as posed is generally feasible with today’s hardware, as the completed flight tests of this thesis show. Time histories of the demonstrator aircraft’s flight mechanical derivatives have been produced in real time by an algorithm running onboard a mini aerial vehicle. The poor sensor quality (and partly the greater effect of atmospheric disturbances) however lead to noticeable deterioration of the results compared to literature results from experiments with manned aircraft (e.g. [26], examples 8.1 and 8.2 or [28]). In the following, the conclusions and recommendations drawn from the experience of this thesis are shown.

Sensor noise and other errors may be absorbed if the covariance settings of the algorithm are increased, but the price for this is slower convergence, slower tracking of varying parameters and larger possible deviations of the identified parameters from the true values.

The aerodynamics model for a particular application should be constrained to an identifiable subset of parameters pre-selected from simulation results. Correlation of the input values must be avoided; this is the dominating aspect requiring high update rates. For most configurations, the $p-\xi$ correlation will be the critical one because high aileron efficiency and a low moment of inertia around the roll axis lead to a fast rolling eigenmotion. For flying wing configurations however, $q-\eta$ may be more critical.

Small vehicles with high eigenfrequencies need considerably higher update rates than larger aircraft. While for manned aircraft rates of 25 Hz may be sufficient ([24], chapter 2.V), for the aircraft of this thesis, 90 Hz were not enough to de-correlate the rolling motion. Faster update rates require more computational power, as in a recursive algorithm, every new data point has to be processed immediately. This means that with decreasing vehicle size, the requirements increase twofold, because more performance has to be delivered by hardware of smaller size.

It is necessary to provide sufficient excitation to the algorithm by rather strong maneuvering. Correlation will increase without maneuvering. The allowable amount of maneuvering for a given mission may dictate the set of identifiable parameters. For missions including prolonged steady cruise flight, all derivatives may converge to biased values.

Different flight states (e.g. sideslip, but also different trim airspeeds) will lead to different values for many derivatives. A higher level decision making routine will have to take this into account, as well as the effects arising from lack of information during cruise. In case of the lift (and possibly also drag) coefficient, the flight state may even have an influence.
on the identifiability of secondary derivatives, because the average lift coefficient affects their relative contributions.

If these different flight states are to be adjusted by a human pilot during flight tests, some kind of feedback device like e.g. a display for current angle of attack is very advisable to help to attain certain states quickly and consistently, especially in the presence of noticeable wind.

Structural vibrations of the airframe need to be avoided. If they are caused by the propulsion system, careful balancing of the propellers may help. Analogous low-pass filters between accelerometers/rate gyros and signal processing provide a good means to alleviate the problem. However, they are usually unavailable in low-cost hardware. Thus, the vibration effects need to be anticipated by the software, e.g. in terms of noise covariances.

### 7.3 Outlook

The destroyed five hole probe is being replaced by a new one; after a new calibration which should pay special attention to the flight envelope demonstrated by the completed flights, more flights should be completed with optimal noise covariance settings and parameter selection. Recently, an interesting approach was published that omitted the need for a flow angle sensor [51], but it is questionable if this could be implemented on the current hardware because of limited computing power. The probe’s susceptibility to off-axis inflow could be alleviated by a special shaping of the front orifice (so-called Kiel probe).

By now, the Paparazzi code supports automatic airspeed control. Using this might allow to identify a set of parameters for a certain airspeed and thus a certain angle of attack and assess the derivative variation with AoA. Direct automatic control of AoA could be implemented based on the air data sensor but is probably not recommendable because the signal to noise ratio is much worse than for airspeed and other effects like turbulence or induced flow angles have much greater effect on AoA than on airspeed.

PID algorithm performance during automatically controlled flights is another thing that should be evaluated with the existing demonstrator aircraft. The few results gained so far indicate that correlation issues get much worse, leading to strongly biased values, but also less jitter in the parameter time histories. This underlines the fact that maneuvering is necessary to provide information beyond the sensor noise.

Computational power of microprocessors as well as their integration level has not stopped increasing. The Paparazzi project offers a new autopilot board (Lisa/L[\textsuperscript{1}]) with the option to include a Gumstix computer-on-a-model with flash memory card slot. Porting the PID software to the Gumstix would allow to increase the update rate by an order of

\[ \text{http://paparazzi.enac.fr/wiki/Lisa/L} \]
magnitude or more and provide onboard storage of full-rate data for post-flight analysis. Also, more sophisticated aerodynamics models become feasible that could include e.g. AoA-dependent derivatives. However, the effort of adopting the software to a new architecture should not be underestimated.

With the advent of smart phones featuring comprehensive sensor suites including accelerometers, angular rate sensors and a GPS receiver along with powerful multi-core processors, lots of onboard storage and various data links, the idea is tempting to make use of them for MAV applications. However, besides the software porting, a problem that has to be solved for this goal is interfacing them to other hardware, especially actuators and external sensors, as air data probes are rather unlikely to be introduced even in sophisticated smart phones. A first step could be to use a smart phone as piggyback computer for high level tasks, communicating to a conventional autopilot board via USB.

Once a sufficiently robust PID algorithm is available, its results can be used to adapt autopilot controller gains and/or realize a decision making system which takes them into account to do online mission planning.

Concerning the algorithm itself, a self-tuning part could be possible that estimates the current noise variances e.g. by taking measuring the sample covariances during stationary flight, or possibly by comparison to a reference model.
A Selected code snippets

A.1 PID algorithm

The following shows the implementation of the parameter identification algorithm as described in section 2.2.3 with the covariance matrix update according to eq. 2.29.

A.1.1 C formulation

The C formulation reflects the code that was actually used in flight. The snippets given below have been extracted from various source files, as some of them were also used in other parts of the code. To improve execution speed, linear algebra operations like matrix-vector multiplication has been implemented as preprocessor macros for a single dimension of inputs only. The following snippets are for four-dimensional matrices/vectors. For higher dimensions, it is advisable to name the matrix and vector elements as e.g. \texttt{a}11 or similar instead of using the alphabetical scheme used here.

```c
/* Data type definitions */

/// 4-element vector
typedef struct {
float u,v,w,x;
} FloatVect4;

/// symmetric 4x4 matrix
typedef struct {
float a,b,c,d,
e,f,g,
h,i,
j;
} FloatSymMat44;

/* Basic operations */

/// dot product
#define VECT4_DOTPROD(v1,v2) ( 
   v1.u*v2.u + v1.v*v2.v + v1.w*v2.w + v1.x*v2.x 
)

/// scalar multiplication
#define VECT4_SMUL(v1, s) { 
   v1.u *= s; v1.v *= s; v1.w *= s; v1.x *= s; 
```

91
// copy
#define VECT4_COPY(a, b) { 
    a.u = b.u; a.v = b.v; a.w = b.w; a.x = b.x; 
}

// addition a + b
#define VECT4_ADD(a, b) { 
    a.u += b.u; a.v += b.v; a.w += b.w; a.x += b.x; 
}

// addition M + N
#define SYMMAT44_ADD(M, N) { 
    M.a += N.a; M.b += N.b; M.c += N.c; M.d += N.d; M.e += N.e; 
    M.f += N.f; M.g += N.g; M.h += N.h; M.i += N.i; M.j += N.j; 
}

/// dyadic (outer) product of two vectors
#define VECT4_DYADPROD(mat, v1, v2) { 
    SYMMAT44_ASSIGN(mat, 
    v1.u*v2.u, v1.u*v2.v, v1.u*v2.w, v1.u*v2.x, 
    v1.v*v2.v, v1.v*v2.w, v1.v*v2.x, 
    v1.w*v2.w, v1.w*v2.x, 
    v1.x*v2.x) 
}

/// matrix vector multiplication res = mat*vec
#define SYMMAT44_VECT4_MUL(res, mat, vec) { 
    res.u = mat.a * vec.u + mat.b * vec.v + mat.c * vec.w + mat.d * vec.x; 
    res.v = mat.b * vec.u + mat.e * vec.v + mat.f * vec.w + mat.g * vec.x; 
    res.w = mat.c * vec.u + mat.f * vec.v + mat.h * vec.w + mat.i * vec.x; 
    res.x = mat.d * vec.u + mat.g * vec.v + mat.i * vec.w + mat.j * vec.x; 
}

/* The main algorithm */

/// One iteration step of the Extended Kalman Filter algorithm
/** for a configuration with 4 inputs and 1 output. Floating point implementation. */

void ekf_step41 (FloatVect4 *theta, FloatVect4 *x,
                 FloatSymMat44 *P, FloatSymMat44 *Q,
                 float y, float r) {
    FloatVect4 K;
    /* Px = P*x */
    SYMMAT44_VECT4_MUL(K, (*P), (*x));
    /* xPxpr = x^T*P*x + r */
A.1 PID algorithm

```c
float xPxpr = r + VECT4_DOTPROD(*x), K);

/* K = 1/(x^T*P*x + r)*P*x */
VECT4_SMUL(K, 1.0/xPxpr);

/* dtheta = (y - x^T*theta)*K */
FloatVect4 dtheta;
VECT4_COPY(dtheta, K);
VECT4_SMUL(dtheta, (y - VECT4_DOTPROD(*x), (*theta)));

/* theta <- theta + dtheta */
VECT4_ADD(*theta, dtheta);

/* Joseph form of covariance matrix update may be reformulated as */
P <- (P + Q) + ((P + Q)*x*(K^T*x - 2) + r*K)*K^T */
SYMMAT44_ADD(*P, (*Q));

/* Px = P*x */
FloatVect4 Px;
SYMMAT44_VECTOR4_MUL(Px, (*P), (*x));

/* Kxm2 = K^T*x - 2 */
float Kxm2 = VECT4_DOTPROD(K, (*x)) - 2;

/* Px <- Kxm2*Px */
VECT4_SMUL(Px, Kxm2);

float rK;
VECT4_COPY(rK, K);
VECT4_SMUL(rK, r);

/* Px <- P*x*(K^T*x - 2) + r*K */
VECT4_ADD(Px, rK);

/* dP = Px*K^T */
FloatSymMat44 dP;
VECT4_DYADPROD(dP, Px, K);

/* P <- P + dP */
SYMMAT44_ADD(*P, dP);
```

// example call
FloatVect4 pid_theta_lift, x_lift;
FloatSymMat44 P_lift, Q_lift;
float C_L, R_lift;
ekf_step51(&pid_theta_lift, &x_lift, &P_lift, &Q_lift, C_L, R_lift);
```
A.1.2 Matlab formulation

The Matlab implementation was used for quick testing with known deterministic input data and for posterior analyses of the full rate data from section 6.1.

```matlab
function [Theta, Sigma] = ekf_scalar(X, Z, Q, R, Theta0, P0)

% Arguments:
% X: Matrix of inputs, nData x nq
% Z: Vector of output, nData x 1
% Q: Process noise covariance matrix, nq x nq
% R: Output measurement noise covariance, scalar
% theta0: Parameter initial values, nq x 1
% P0: P matrix initial value, nq x nq

% Returns:
% Theta: Identified parameters for each time step, nData x nq
% Sigma: Parameter variance for each time step, nData x nq

% [snip] argument error checking

Theta = zeros(nData,nq); % Time histories of identified parameters
theta = Theta0; % initial value for parameters
Sigma = zeros(nData,nq); % Time histories of parameter variances
P = P0; % Initial value

Sigma(1,:) = sqrt(diag(P0))';

for i=1:nData
    x = X(i, :)'; % extract current input values
    z = Z(i, :); % extract current output values
    P = single(P); % convert to single precision to mimic execution on the Paparazzi board
    x = single(x); % execution on the Paparazzi board

    % Simplified filter algorithm
    K = P*x;
    xPxpr = single(R) + x' *K;
    K = 1.0/xPxpr*K;
    nu = single(z) - x'*theta;
    dtheta = K *nu;
    theta = theta + dtheta;

    % Covariance matrix update
    P = P + single(Q);
    Px = P*x;
    Kxm2 = K'*x - single(2);
    Px = Px + Kxm2;
    rK = K*R;
    Px = Px + rK;
    dP = Px*K';
    P = P + dP;

    % record time histories
    Theta(i,:) = theta;
    Sigma(i,:) = sqrt(diag(P))';
end
```
B Aircraft and hardware

B.1 Demonstrator aircraft reference data

Figure B.1 shows orthogonal and isometric views of the demonstrator aircraft as it was modeled in XFLR5. The model contains still the original fuselage. Engine nacelles and additional installations like the nose boom can unfortunately not be included in this program.

Reference dimensions, control surface deflections and inertial data can be found in tables B.1, B.2 and B.3 respectively.

Figure B.1: Views of the demonstrator aircraft in XFLR5 (original fuselage)
B Aircraft and hardware

### Table B.1: Dimensions of the demonstrator aircraft

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference wing area $S$</td>
<td>0.308 m$^2$</td>
</tr>
<tr>
<td>Reference wing chord $c$</td>
<td>0.22 m</td>
</tr>
<tr>
<td>Reference wing span $b$</td>
<td>1.4 m</td>
</tr>
<tr>
<td>Aspect ratio $\Lambda$</td>
<td>6.5</td>
</tr>
<tr>
<td>Overall length</td>
<td>1.08 m</td>
</tr>
</tbody>
</table>

### Table B.2: Control surface deflections

<table>
<thead>
<tr>
<th>Surface</th>
<th>Deflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ailerons</td>
<td>$\pm 15^\circ$</td>
</tr>
<tr>
<td>Elevator</td>
<td>$-23 - 13^\circ$</td>
</tr>
<tr>
<td>Rudder</td>
<td>$\pm 25^\circ$</td>
</tr>
</tbody>
</table>

### Table B.3: Reference values for inertial data

<table>
<thead>
<tr>
<th>$m$</th>
<th>2.0 kg</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{xx}$</td>
<td>0.0356 kg m$^2$</td>
</tr>
<tr>
<td>$I_{yy}$</td>
<td>0.0847 kg m$^2$</td>
</tr>
<tr>
<td>$I_{zz}$</td>
<td>0.106 kg m$^2$</td>
</tr>
<tr>
<td>$I_{xz}$</td>
<td>$-0.012$ kg m$^2$</td>
</tr>
</tbody>
</table>
B.2 Hardware

The following list shows a compilation of the exact models and makes of the hardware used in this project. Choices of most components were based on experiences made in preceding projects at the Institute. Some components were re-used, especially the five hole probe and INS. For details on the choice of the autopilot, see section 3.2.

Aircraft:
- based on Multiplex TwinStarII, new fuselage designed and built at the Institute to accommodate additional hardware

Motors:
- Flyware microREX 220-12-1800

Battery:
- Kokam, 4000 mAhm, 3S1P, 30C max discharge current

Propellers:
- GWS EP8040, 8” x 4”

Motor controllers:
- YGE 12 (12 A), cooling element added to prevent overtemperature during low airspeed/high power flight phases

Servos:
- Hitec HS 82-MG, ailerons powered via BEC, tail servos by autopilot

R/C receiver:
- Multiplex RX-7 Synth IPD

R/C transmitter:
- Multiplex mc3030

Autopilot:
- Paparazzi Tiny v2.11

Data link modem:
- Radiotronix WI232EUR

INS:
- Xsens MTi-G, 400 °/s, ±5 g, GPS antenna WiSys WS 3910 (active)

Five hole probe:
- Inherited from preceding project [25]

Pressure transducers:
- Sensor Technics HCA and HDI series, viz.
  - HCA0811ARH8 (static pressure, 0.8...1.1 bar),
  - HDIM010DUE8P5 (dynamic pressure, 0...1000 Pa),
  - HDIM010DBESP5 (five hole probe differential pressures, −1000...1000 Pa)
The INS was connected to the autopilot’s UART0 bus which is hard-wired to its built-in GPS module. It is however possible to disable the GPS by pulling low the GPS_RESET pin on the DOWNLOAD connector on the autopilot, which allows to freely use the UART0. In turn, the position data for the autopilot has to be provided by the INS. To convert the different voltage levels of INS and autopilot, a connector cable was built in-house which included a MAX232A chip.

Sensor noise characteristics are listed in table B.4 as $1\sigma$ standard deviations. The values for the pressure sensors result from the wind tunnel test, while the INS sensors were evaluated in static desktop tests.

<table>
<thead>
<tr>
<th></th>
<th>dynamic</th>
<th>differential</th>
<th>static</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressure</td>
<td>42 Pa</td>
<td>16 Pa</td>
<td>7.3 Pa</td>
</tr>
<tr>
<td>INS</td>
<td>accelerations 0.015 m/s</td>
<td>angular rates 0.35°</td>
<td></td>
</tr>
</tbody>
</table>

Table B.4: $1\sigma$ sensor noise levels in static test conditions
C Simulation results

C.1 Coefficient buildups

Figure C.1: $C_L$ buildup
C Simulation results

Figure C.2: $C_m$ buildup

Figure C.3: $C_C$ buildup
C.1 Coefficient buildups

Figure C.4: $C_l$ buildup

Figure C.5: $C_n$ buildup
D Flight test results

D.1 Onboard PID flight

Figure D.1: Inflow data
Figure D.2: Control settings
Figure D.3: Angular rates
E Summary of results

The following tables summarize the parameters found with the different methods in this thesis. The reference values column repeats table 4.2. EKF results are given as scalar values, reflecting the mean value of the corresponding parameter history for starting at a certain point in time, after which convergence is considered to have occurred. These time points are

Simulation: \( t = 25 \text{s} \) (after the rudder maneuver has finished)
Longitudinal analysis of post-flight EKF: \( t = 360 \text{s} \) (some time after flaps deployment)
Lateral analysis of post-flight EKF: \( t = 150 \text{s} \) (after rudder maneuvering has started)
In-flight EKF: \( t = 145 \text{s} \) (after sideslip and elevator maneuvering)

The percentage reflects the \( 1\sigma \) standard deviation of this time history, indicating convergence rather than accuracy.

In the post-flight EEM column, the values given are the parameters calculated by the equation error method, along with their estimated standard errors.
<table>
<thead>
<tr>
<th></th>
<th>Reference</th>
<th>Simulation</th>
<th>Sensor models</th>
<th>Flight test</th>
<th>Flight test</th>
<th>Flight test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>EKF</td>
<td>EKF</td>
<td>EEM</td>
<td>EKF</td>
<td>EKF</td>
<td>EKF</td>
</tr>
<tr>
<td>Refer section</td>
<td>4.3.2</td>
<td>5.4.3</td>
<td>5.4.4</td>
<td>6.1.1</td>
<td>6.1.3</td>
<td>6.3.2</td>
</tr>
<tr>
<td>$C_L$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{L0}$</td>
<td>0.3</td>
<td>0.3</td>
<td>&lt;0.1%</td>
<td>0.32</td>
<td>1.2%</td>
<td>0.58</td>
</tr>
<tr>
<td>$C_{Lx}$</td>
<td>4.6</td>
<td>4.6</td>
<td>&lt;0.1%</td>
<td>4.3</td>
<td>5.9%</td>
<td>4.8</td>
</tr>
<tr>
<td>$C_{Lq}$</td>
<td>6.5</td>
<td>6.4</td>
<td>0.45%</td>
<td>18</td>
<td>23%</td>
<td>3.9</td>
</tr>
<tr>
<td>$C_{Ln}$</td>
<td>0.55</td>
<td>0.55</td>
<td>0.24%</td>
<td>1.1</td>
<td>9.3%</td>
<td>——</td>
</tr>
<tr>
<td>$C_{L\delta \alpha, sym}$</td>
<td>0</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td>$C_m$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{m0}$</td>
<td>0.05</td>
<td>0.05</td>
<td>&lt;0.1%</td>
<td>0.05</td>
<td>0.21%</td>
<td>0.065</td>
</tr>
<tr>
<td>$C_{mx}$</td>
<td>-0.3</td>
<td>-0.3</td>
<td>&lt;0.1%</td>
<td>-0.28</td>
<td>1.2%</td>
<td>-0.33</td>
</tr>
<tr>
<td>$C_{mq}$</td>
<td>-11</td>
<td>-11</td>
<td>&lt;0.1%</td>
<td>-11</td>
<td>2%</td>
<td>-12</td>
</tr>
<tr>
<td>$C_{mn}$</td>
<td>-1.2</td>
<td>-1.2</td>
<td>&lt;0.1%</td>
<td>-1.2</td>
<td>1.3%</td>
<td>-1</td>
</tr>
<tr>
<td>$C_{m\delta \alpha, sym}$</td>
<td>0</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td>$C_D$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{D0}$</td>
<td>0.06</td>
<td>0.06</td>
<td>&lt;0.1%</td>
<td>0.062</td>
<td>0.65%</td>
<td>0.14</td>
</tr>
<tr>
<td>$C_{DC_L}$</td>
<td>0</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>-0.099</td>
<td>-0.95%</td>
</tr>
<tr>
<td>$C'_{DC_L}$</td>
<td>0.06</td>
<td>0.06</td>
<td>&lt;0.1%</td>
<td>0.06</td>
<td>6.6%</td>
<td>0.11</td>
</tr>
<tr>
<td>$C_{D\beta^2}$</td>
<td>1</td>
<td>1</td>
<td>&lt;0.1%</td>
<td>0.96</td>
<td>1.2%</td>
<td>——</td>
</tr>
<tr>
<td>$C_{Dn}$</td>
<td>0.025</td>
<td>0.025</td>
<td>&lt;0.1%</td>
<td>-0.01</td>
<td>&gt;100%</td>
<td>——</td>
</tr>
<tr>
<td>$C_{D\phi^2}$</td>
<td>0</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
</tr>
</tbody>
</table>

Table E.1: Summary of PID results for all analyses of this thesis – longitudinal motion
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EKF</th>
<th>EKF</th>
<th>EEM</th>
<th>EKF</th>
<th>EKF</th>
<th>EKF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refer section</td>
<td>4.3.2</td>
<td>5.4.3</td>
<td>5.4.4</td>
<td>6.1.2</td>
<td>6.1.3</td>
<td>6.3.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C_C</th>
<th>C_C0</th>
<th>C_Cb</th>
<th>C_CP</th>
<th>C_Cr</th>
<th>C_Cξ</th>
<th>C_Cζ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>-0.3</td>
<td>-0.05</td>
<td>0.04</td>
<td>0</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>&lt;10^-7</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
</tr>
<tr>
<td></td>
<td>65%</td>
<td>37%</td>
<td>2.9%</td>
<td>19%</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>0.00036</td>
<td>-0.29</td>
<td>-0.065</td>
<td>-0.012</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>37%</td>
<td>4.2%</td>
<td>2.9%</td>
<td>19%</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>0.0032</td>
<td>-0.45</td>
<td>-0.5</td>
<td>-0.12</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>7.2%</td>
<td>-0.83%</td>
<td>4.4%</td>
<td>-0.012</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>——</td>
<td>-0.47</td>
<td>2.5%</td>
<td>0.028</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>100%</td>
<td>——</td>
<td>——</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C_l</th>
<th>C_l0</th>
<th>C_lb</th>
<th>C_lp</th>
<th>C_lr</th>
<th>C_lξ</th>
<th>C_lζ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>-0.08</td>
<td>-0.5</td>
<td>0.08</td>
<td>-0.3</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>&lt;10^-7</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
</tr>
<tr>
<td></td>
<td>&gt;100%</td>
<td>5.9%</td>
<td>-0.41</td>
<td>0.068</td>
<td>-0.25</td>
<td>-0.3</td>
</tr>
<tr>
<td></td>
<td>60%</td>
<td>5.9%</td>
<td>0.34%</td>
<td>3.1%</td>
<td>0.67%</td>
<td>3.5%</td>
</tr>
<tr>
<td></td>
<td>37%</td>
<td>-1.3%</td>
<td>-0.18</td>
<td>0.069</td>
<td>-0.2</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>4.8%</td>
<td>-1.3%</td>
<td>-0.13</td>
<td>5.4%</td>
<td>-0.2</td>
<td>3.5%</td>
</tr>
<tr>
<td></td>
<td>99%</td>
<td>4.5%</td>
<td>-0.11</td>
<td>-0.11</td>
<td>5.1%</td>
<td>5.4%</td>
</tr>
<tr>
<td></td>
<td>18%</td>
<td>4.5%</td>
<td>-0.11</td>
<td>5.4%</td>
<td>5.1%</td>
<td>5.4%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C_n</th>
<th>C_n0</th>
<th>C_nb</th>
<th>C_np</th>
<th>C_nr</th>
<th>C_nξ</th>
<th>C_nζ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0.03</td>
<td>-0.03</td>
<td>-0.05</td>
<td>-0.05</td>
<td>-0.07</td>
</tr>
<tr>
<td></td>
<td>&lt;10^-7</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
<td>&lt;0.1%</td>
</tr>
<tr>
<td></td>
<td>34%</td>
<td>2.7%</td>
<td>1.8%</td>
<td>0.88%</td>
<td>0.3%</td>
<td>0.11%</td>
</tr>
<tr>
<td></td>
<td>-0.0013</td>
<td>-0.033</td>
<td>-0.046</td>
<td>-0.017</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>34%</td>
<td>2.7%</td>
<td>1.8%</td>
<td>0.88%</td>
<td>0.3%</td>
<td>0.11%</td>
</tr>
<tr>
<td></td>
<td>-2.1%</td>
<td>-0.11</td>
<td>-0.054</td>
<td>-0.052</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
</tr>
<tr>
<td></td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
</tr>
</tbody>
</table>

Table E.2: Summary of PID results for all analyses of this thesis – lateral motion
Bibliography


[8] BUFFO, Rainer: Windkanalstudie zur Widerstandsreduzierung von Lkw anhand einer speziellen Modellanordnung, Institute of Aeronautics and Astronautics, RWTH Aachen University, Student research project, 2006


111


[40] Paparazzi Project Homepage. http://paparazzi.enac.fr/wiki/Main_Page


[43] SONTHEIM, David M.: Beschaffung des flugmechanischen Datensatzes eines Modellflugzeuges, Institute of Flight System Dynamics, RWTH Aachen University, Student research project, August 2009


[48] ZIMMER, A.: Integration und Kalibrierung einer Fünflochsonde in einem automatisch gesteuerten Flugzeug, Institute of Flight System Dynamics, RWTH Aachen University, Student research project, 2009