Proceedings of the EuroGNC 2013, 2nd CEAS Specialist Conference
on Guidance, Navigation & Control, Delft University of Technology,
Delft, The Netherlands, April 10-12, 2013

Experiences with the Barracuda UAV Auto Flight System

J. van Tooren[1], R. Hammon[2]

Abstract Operational surveillance and reconnaissance requirements not only put requirements on the mission systems, but also affect requirements on the reliable and autonomous operation of a UAV platform. To enable the safe and care free operation of UAVs in complex mission scenarios Cassidian has invested in the development of necessary technologies for reliable and autonomous Auto Flight systems for UAVs. Furthermore, due to decreasing budgets the design and development phases of such systems need to be cheaper and faster, even though functional complexity is constantly increasing. This paper details the Cassidian experience with the Auto Flight system on the Barracuda technology demonstrator. The guidance and control functional architecture and control law design are detailed regarding the newly developed Auto Flight system which successfully flew in multiple flight tests in 2012 on the Barracuda UAV demonstrator.

[1] J. van Tooren

Cassidian, EADS Deutschland GmbH, Rechlinerstraße, 85077 Manching, Germany
email: Joost.vanTooren@cassidian.com

[2] R. Hammon

Cassidian, EADS Deutschland GmbH, Rechlinerstraße, 85077 Manching, Germany
email: Reiner.Hammon@cassidian.com

2 Design drivers and philosophy

Typical mission scenarios for UAVs include prolonged operation in a mission area during which new information (either from the aircraft's own payload sensors or external assets) can cause a re-tasking of the system by the operator. To ensure the operator has good situational awareness on the operation of the UAV platform, it is mandatory not to overload the operator with unnecessary tasks and information.

The Cassidian approach is to provide a high level of autonomy of the Auto Flight system. The stored flight plan is flown automatically using on-board trajectory generation and care free flight control with the possibility (not necessity) of only high level interaction with the system, using so-called High Level Commands (i.e. no remote control using a conventional pilot stick). All failure detection, isolation and reconfiguration logic is designed such that the system has a graceful degradation in case of failures.

As the aircraft must always be able to navigate back to a safe landing, even in case of a data link failure, it is necessary to conduct certain safety critical functions such as navigation, guidance and control on-board the platform. Therefore, all necessary sensing, flight plan handling, trajectory generation and flight control functions are integrated into a high integrity computing architecture. A clear segregation of mission relevant functions and safety-critical functions is ensured, such that any possible errors in the mission critical software do not adversely affect the safety critical functions. Furthermore, since the development cycles of the mission and safety critical software are not necessarily synchronous and done by different teams, interdependencies are kept to a minimum to avoid development and test planning problems.

Conventional autopilot systems usually navigate to the active waypoint using a moding of multiple control laws, depending on the aircraft's position and orientation. The flown trajectory is therefore only implicitly determined in the design. However, on Barracuda an explicit trajectory towards the active waypoint is generated on board of the UAV for the following reasons:

- To ensure optimal Operator situational awareness by displaying the planned trajectory towards the active waypoint on the ground control station. It is essential that this trajectory is identical to the actual trajectory flown by the UAV.
- To enable both conventional and complex mission waypoint types which can be assessed by both the operator and the mission system.

These design drivers have resulted in a new design of the Barracuda Auto Flight system. The following sections detail this design and the experiences at Cassidian during concept development, implementation and testing of the new system.

Reference [1] has been used for the definition of all flight mechanic nomenclature and conventions throughout this paper.

Auto Flight Functional Architecture

To enable the fast integration of an increasing number of functional requirements on the system and to keep development time and cost low, a modular functional architecture is used. The architecture is similar to manned aviation Auto Flight systems, with the exception that almost all functionalities are safety critical (i.e. the functionalities must work at all times) and are therefore integrated into the high integrity flight control computers.

The Auto Flight System consists of a Flight Management, Flight Guidance and Autopilot / Control Law system as depicted in Fig. 1. The Flight Management System (FMS) is responsible for the moding of primary Phases of Flight and all flight plan and waypoint handling.

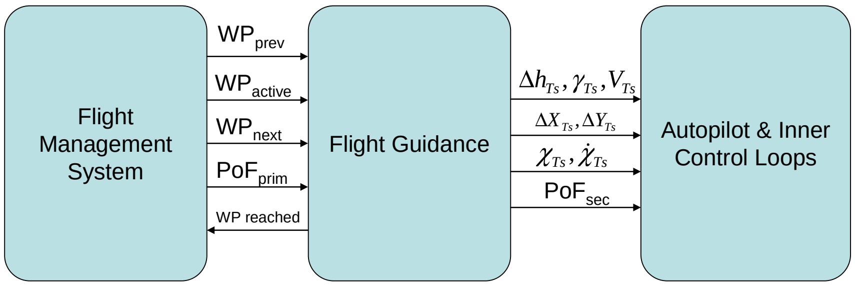
Flight Management System
WP_{prev}
WP_{active}
WP_{next}
PoF_{prim}
WP reached
Flight Guidance
$\Delta h_{Ts}, \gamma_{Ts}, V_{Ts}$
$\Delta X_{Ts}, \Delta Y_{Ts}$
$\chi_{Ts}, \dot{\chi}_{Ts}$
PoF_{sec}
Autopilot & Inner Control Loops

Fig. 1. Auto Flight Functional Architecture

The FMS Phase of Flight moding constitutes of the main phases of flight such as Taxi, Take-off, Climb, En-route and Landing. These 'primary' Phases of Flight are provided by the FMS to the Flight Guidance which respectively determines the internal secondary Phase of Flight detailing the Flight Guidance modes (e.g. flare, de-crab, de-rotation, …).

Furthermore, the FMS interfaces with the Flight Guidance by providing three waypoints: Previous, Active and Next waypoint. The Flight Guidance can then generate an explicit trajectory towards the Active waypoint, taking into account the location and altitude of the Next waypoint. As the Active waypoint is reached, the Flight Guidance reports this to the FMS, which in its turn provides the new triplet of waypoints.

The Flight Guidance interfaces with the Autopilot by stating the aircraft's deviation from the commanded trajectory as shown in Fig. 2. The Autopilot contains all control laws necessary for the ground and flight phases. The lateral and vertical trajectory and speed profile is commanded to the Autopilot with a certain time advance such that the Autopilot can follow this trajectory as closely as possible.

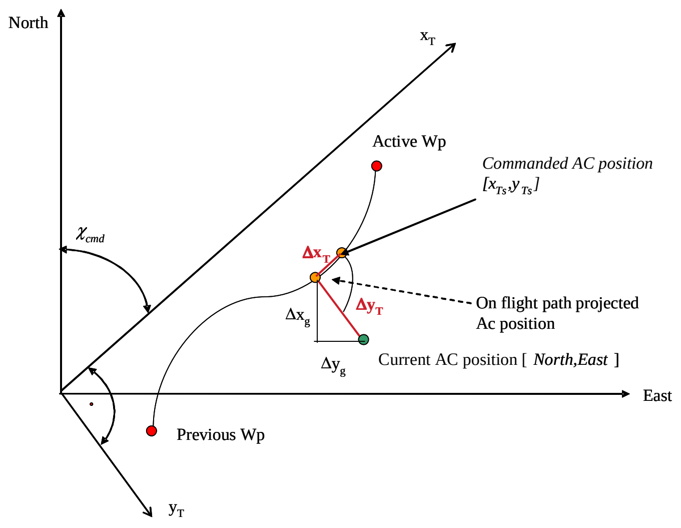

Fig. 2. Definition of trajectory commands from the Flight Guidance to the Autopilot

Flight Guidance algorithms

The explicit trajectory generation uses line and arc trajectory segments to construct the lateral trajectory. The segments (and therefore the trajectory) are defined in the earth fixed reference frame. The vertical trajectory and speed profile is then determined along the lateral trajectory. The trajectory generation (i.e. set of line and circle segments) is determined for a certain set of Previous, Active and Next waypoints. Once the trajectory is fixed, the current deviation towards this trajectory is used to generate the commands for the Autopilot.

The Flight Guidance trajectory and command generation is divided into the following four sub-functions:

- Setup: Sets up all information as required for the trajectory planning algorithms based on the Previous, Active and Next waypoints and current aircraft position and orientation.
- Planner: Generates the lateral, vertical and speed trajectory to the active waypoint depending on the waypoint types. The result is a list of trajectory segments (curves and straight lines).
- Scheduler: Determines the current lateral, vertical and speed segment depending on the current state of the aircraft.
- Commander: Calculates the lateral, vertical and speed commands for the autopilot using the deviation towards the projected position along the current segment as defined by the Scheduler.

Fig. 3 shows the trajectory planning as part of the overall Flight Guidance functionality. The following chapter shows some examples of the lateral trajectory planner.

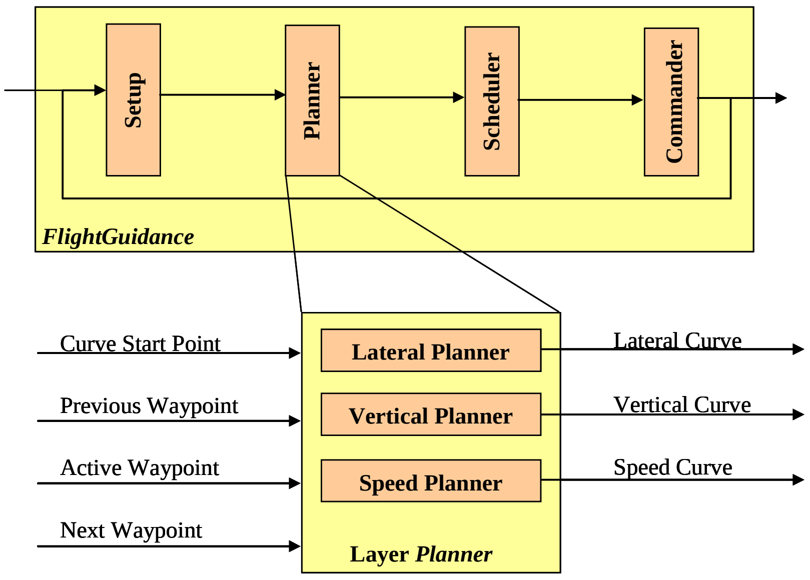

Fig. 3. Flight Guidance trajectory planning

Examples of Lateral Trajectory Planning

The starting point of the lateral trajectory (i.e. Curve Start Point) is normally the last commanded position of the Flight Guidance to the Autopilot to ensure a smooth transition between waypoints, unless the current aircraft position is too far away from this last commanded position (e.g. due to initialisation, wind, etc.).

Depending on the turn type of the active waypoint, the lateral Planner decides which mode of the lateral Planner is to be used to generate the lateral curve. The turn radius depends on the preferred value and type of the active waypoint, although it is limited to the flight performance of the UAV.

Since the Curve Start Point generally does not lie on the leg as defined by the Previous and Active waypoint, it is necessary to construct a so-called 'Transition-to-Leg' segment sequence. This is constructed by defining arc-line-arc segments where the line segment has a certain maximum intercept angle (for instance 45°), depending on the cross track distance and orientation towards the leg.

For a conventional 'Fly By' waypoint the Transition-to-Leg is then followed by a line and an arc segment, such that the trajectory rolls into the Fly-By turn towards the Next waypoint as illustrated in Fig. 4.

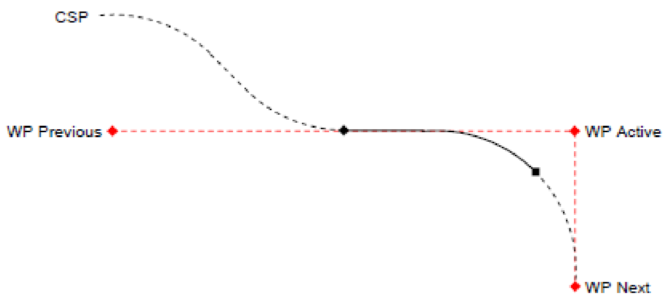

Fig. 4. Example of a 'Fly-By' lateral trajectory

If the Active waypoint is of the type 'Circle Center', the lateral trajectory is constructed using a so-called 'Transition-to-Circle' segment sequence, similar to the 'Transition-to-Leg' sequence, followed by an arc around the Active waypoint with a certain radius and waypoint segment angle as specified in the waypoint attributes. Fig. 5 shows an example of how the lateral trajectory is constructed from the Curve Start Point towards and around the specified Active waypoint.

Trajectories for all waypoint types are constructed in the Planner using similar algorithms and always result in a certain segment sequence list. This list is then transferred to the Scheduler and Commander, which then generate the corresponding commands for the Autopilot control loops.

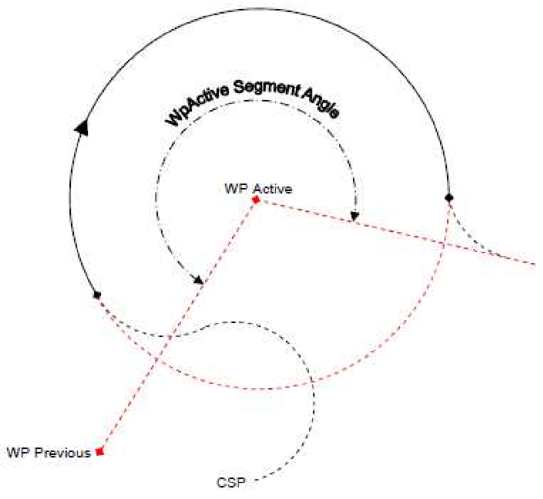

Fig. 5. Example of 'Circle-Center' mission waypoint

Autopilot and Inner Control Loops

The Autopilot has to function throughout the flight envelope for all required failure and environmental conditions. A conventional approach with elaborate gain tuning would have taken too long and therefore a new approach was used for the Barracuda 2012 flight campaign which had been researched in the previous years and partly published in [2] and more elaborately in [3]. Since the flight campaign several improvement potentials have been identified, but the basic principles will remain unchanged and have proven their strengths. In the following sections, the basic principles of this new approach are presented. A detailed elaboration of the control concept however is out of scope of this paper.

The design comprises of inner loops (Primary Control Laws) as well as the design of outer loops (Autopilot) for lateral and longitudinal control of the aircraft. The design method uses a straightforward approach which required no additional tuning of the gains by hand. The flight tests related to this first version of the control laws were conducted in the summer of 2012 and showed a very successful performance of the control laws.

The principal characteristics of the presented Barracuda control law design concept together with the corresponding methods used are depicted in Fig. 6 and are subsequently elaborated in the following sections. This is then followed by an explanation of the usage of the concept for the longitudinal and lateral control of the aircraft. Finally, an example of the implementation of the pitch rate control loop is presented.

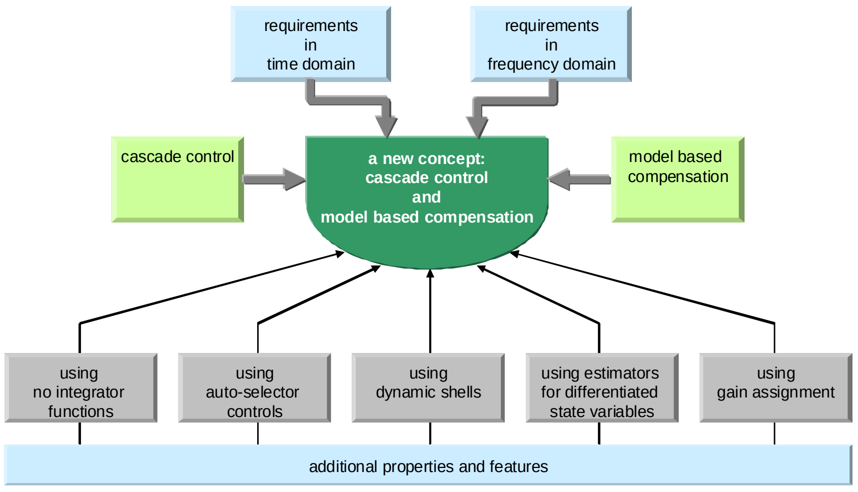
requirements in time domain
requirements in frequency domain
cascade control
a new concept: cascade control and model based compensation
model based compensation
using no integrator functions
using auto-selector controls
using dynamic shells
using estimators for differentiated state variables
using gain assignment
additional properties and features

Fig. 6. Principal characteristics for the Barracuda Control Law design concept

Using cascade control:
Cascade control is a well-proven classical method which is the preferred control method for Autopilot and Auto Throttle (ATHR) design (control laws for aircraft).

The applications for cascade control are mainly restricted to systems with one input and multiple outputs. Furthermore, cascade control seems only to be applicable for plants with relatively long time constants or with integrator-like behaviour such as translatory aircraft modes (e.g. autopilot or ATHR control law design). The presented concept however demonstrates that cascade control can also be used for multi-input, multi-output systems as well as for plants with short time constants. The combination of two methods - 'cascade control' and 'model based compensation' - allows the usage of cascade control in the field of Primary Control Laws as well. Cascade control corresponds in many aspects to the mindset of the control engineer and gives a good understanding of how the control technique works. Mathematical considerations are standing in the background of this paper and are given only when necessary to understand the real application of control.

The basic structure of cascade control is defined in Fig. 7 for the special case that the plant consists of a chain of n integrators. The transfer function for every control step can be described approximately for low frequencies by following transfer function.

$$H_i(s) = \frac{x_i(s)}{x_{ic}(s)} \approx \frac{K_i}{s + K_i}, i = 1,2,...,n$$

Time response and stability margin for control loop *i* depend essentially on the following ratio.

$$c_i = \frac{K_{i-1}}{K_i}, i = 1,2,...,n$$

An increase of c_i is correlated to an increase of the stability margin between loop *i* and loop *i-1* and to an increase of the damping of control loop *i* as well to a deceleration of control. A decrease of c_i is correlated to a decrease of the stability margin between loop *i* and loop *i-1* and to an increase of the damping of control loop *i* as well to an acceleration of control. The question is what will be a good compromise related to an adjustment for c_i. Subsequently, by a recursive dimensioning of all overall gains it can be shown that the step responses of the loops *1* to *n* are converging to a certain type of step response which becomes slower from one loop to the next loop by the factor '*c*'. In many cases the control engineer wants a small overshoot or one that tends to zero in the step response. It can be shown in simulation that the limit case for overshoots for $n \longrightarrow \infty$ converges to about *e = 2.72*. A mathematical proof that this converges to *e* has not been established, but would be an interesting issue of research. The Barracuda cascade was defined exactly by the factor *e*.

The consideration related to the rules of cascade as they are described in this paper are excluding zeros in the transfer functions. Zeros can only be accepted related to considerations which are using certain approximations. The general case for a plant with one input and several outputs and no zeros related to the transfer functions $H_i(s)$ defines additional feedback loops from one integrator output in the chain to the input of one integrator which is placed in one inner loop. The principal idea of model based compensation is to compensate the influence of these additional feedback loops in a good approximation in order to reach comparable results in the time domain. The influence of the compensation effect in frequency domain is reduced essentially if one uses models of state variables with reduced frequency content instead of real state variables. For that purpose model based compensation has to be defined.

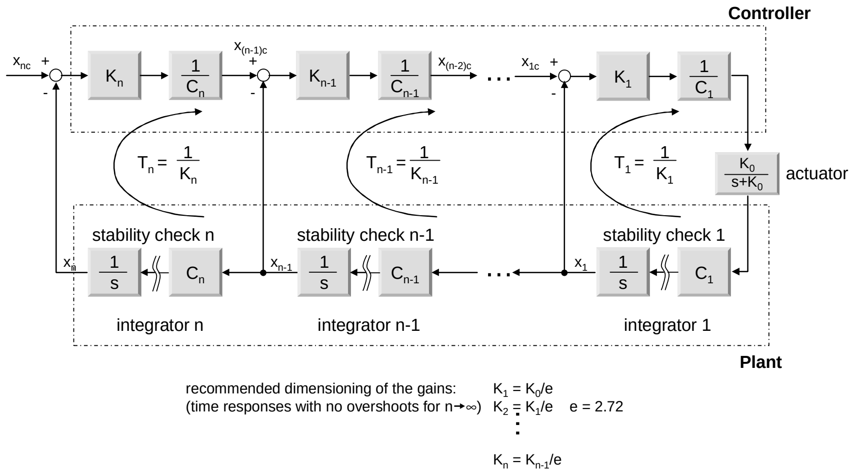

Fig. 7. The ideal case for cascade control

Using model based compensation:

In order to reach comparable control results for a plant with and without additional internal feedbacks, it is necessary to compensate all undesirable additional internal feedback loops of the plant.

The differential equation for $\dot{x}_n$ is defined in general by the following equation, where $a_{n(n-1)} \cdot x_{(n-1)}$ represents the desired dependency on the next inner loop.

$$\dot{x}_n = a_{n1} \cdot x_1 + ... + a_{n(n-1)} \cdot x_{(n-1)} + a_{nn} \cdot x_n + ...$$

Zeros are defined in general by the coefficients a_{n1}, a_{n2},…, $a_{n(n-2)}$. Therefore it is assumed that these coefficients are zero or very small.

As the dynamics of the aircraft are well understood and high fidelity models of the 'plant' (e.g. aerodynamics, engine, intake, mass, inertias, etc.) are available, it is possible to compensate widely for all undesirable coupling terms. Once all undesirable coupling terms are removed, the estimation of $\dot{x}_n$ reduces to a simple integrator equation which is only dependent on the state variable of the next inner loop.

$$\dot{x}_n \approx a_{n(n-1)} \cdot x_{(n-1)}$$

It is therefore the task of 'model based compensation' to compensate for low frequency exactly and for high frequency approximately the undesirable influence of all coupling terms in such a way that the estimation of $\dot{x}_n$ is a simple integrator of the next inner loop.

Instead of using the (measured) state variables directly for the compensation task, the state variables are derived using models from their corresponding command values or alternatively only low pass filtered state variables are used for cascade control. The advantage is that the estimated state variables have reduced frequency content in comparison to their real (measured) values.

Using estimators for differentiated state variables
The compensation task requires differentiated state variables. If using differentiated state variables, it is advisable to use estimates of them with reduced frequency content and not simply differentiated state variables. The construction of suitable estimators for all state variables of control is the central problem and is the heart of the new concept. The dynamics (defined by gains) of these estimators are correlated directly to the overall gains of corresponding control loops.

Using no integrator functions:
The target to be reached for every control step within cascade control is a steady state control error which tends to zero. As far as one prefers designing for an integrator (PI algorithm) at every control step, one has to stabilise one integrator for every control step. This is not desirable as every additional integrator costs additional energy for stabilising which slows down the control dynamics. This disadvantage for PI algorithms disappears if one replaces the PI algorithm by a special PD algorithm. This however causes another problem of the need for differentiated state variables as described in the previous section.

Using gain assignment:
The special case of a plant which can be described as a chain of several integrators (as shown in Fig. 7) gives a good indication of how the overall gains of the cascade control loops can be chosen in general. Nevertheless it is possible to implement other rules for the selection of the overall gains. A recursive definition of the overall gains in one control loop is a method which will work in many cases but not in all. It is assumed that many 'normal' cases can be defined by using this rule. Barracuda is one of these 'normal 'cases (stable aircraft). Pathologic cases, such as unstable aircraft, may require another rule. Nevertheless it can be stated that a recursive definition of all overall gains allows very fast definitions of good solutions for stable aircraft. Only plants with relatively fast time constants or unstable behaviour can be a problem for the selection philosophy.

Using dynamic shells:
The definition of dynamic shells is describing the dynamic behaviour of different control loops. In general one integrator of the plant is located in one single shell as far as only one axis of the aircraft has to be described. In case that more than one

axis of the aircraft shall be considered it is possible to define comparable shells for all axes. In such a case one defines a harmonisation of the dynamics of all axes which is an essential design aspect for many reasons (e.g. in order to coordinate a turn manoeuvre). Examples of how multiple axes of the plant are modelled using equivalent dynamic shells are given in Fig. 8 and Fig. 9, where the principle dependencies of the state variables are represented.

Using auto-selector control:
There are many physical values of the aircraft which have to be kept within a certain safe range and which are not embedded in actual control. As far as one of these values is near its limit the control of a comparable control loop variable has to be interrupted in order to hold the critical value within its limit by controlling this value as a replacement of the normal control variable. Such a feature can be realised by a so-called auto-selector control. Auto-selector control works only for variables within one dynamic shell.

Definition of Longitudinal and Lateral Plant

Before any further control laws are designed, the two parts of the plant (longitudinal part, lateral part) have to be defined from a point of view which allows design of the corresponding control laws for cascade control. The models as shown in Fig. 8 and Fig. 9 are structured in so-called corresponding dynamic shells. The 'feed forward like' influence of the control surface deflections and angle of attack across the dynamic shells is ignored for cascade control design purposes and are therefore displayed using dotted lines. It subsequently needs to be shown that these effects are negligible during linear stability checks of the controller.

Stability checks are defined at the input of every integrator which represents a certain state variable. The stability checks are related to the frequency response of external disturbances to the plant integrator behaviour in each shell as shown in Fig. 7. The cuts are defined at central places (bottlenecks) where signals are flowing from inner dynamic shells to one outer shell. It is also possible to define the cuts within the control laws after every cascade element (one control loop) as this mirrors the plant integrator behaviour at each shell.

The cuts defined at the inputs of the integrators representing the state variables for $p, q, r, \alpha_a, \beta_a$ define the necessary checks for Primary Control Laws. The cuts defined at the inputs of the integrators representing the state variables for $\mu_a, V_a, \gamma_a, \chi_a, h, y$ define the necessary checks for the Autopilot function and auto throttle.

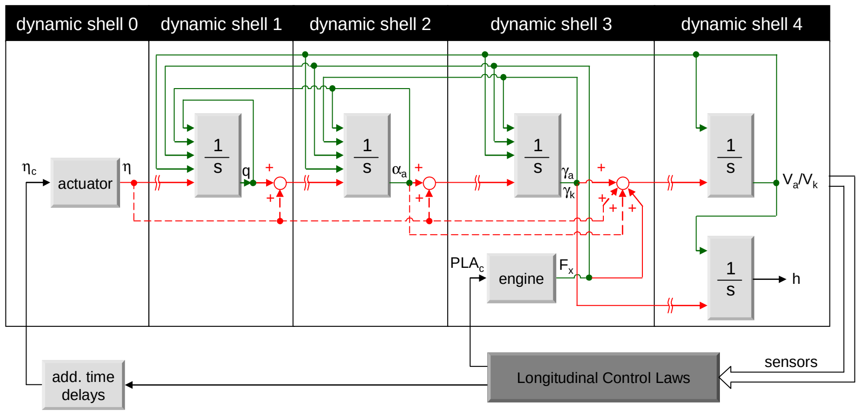

Fig. 8. Model of the dependencies of the state variables in the longitudinal part of the plant

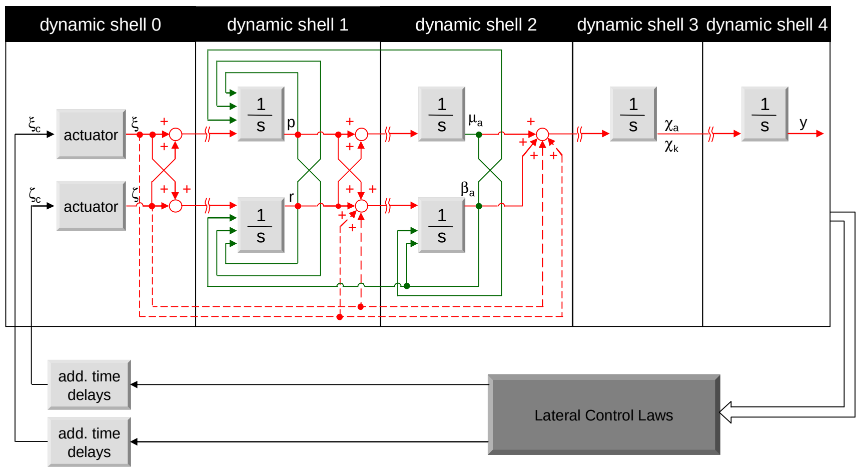

Fig. 9. Model of the dependencies of the state variables in the lateral part of the plant

Autopilot and Inner Loop Control Law Structure

Unlike conventional Autopilots only a single Autopilot mode is necessary in Barracuda for all waypoint types as the Flight Guidance generates an explicit trajectory and commands the deviation to the Autopilot. This reduces the complexity of the required moding logic in the Autopilot and furthermore only one Autopilot control law design is necessary for all flight phases.

The outer loops are shown in Fig. 10 for the Autopilot and ATHR which control the states as commanded by the Flight Guidance. The couplings between longitudinal and lateral dynamics described by the function block 'transformation of load factor increments Δn_{za} and Δn_{ya} to commands of α_a and μ_a' disappear for the symmetric case $\phi_a = \mu_a = 0$. Only during turn manoeuvres ($\mu_a \neq 0$) the modelled couplings are non-zero. The commands of the Flight Guidance for the Autopilot control laws need to be calculated T$_3$ seconds (time constant of the 3rd dynamic shell) in advance as shown in Fig. 10 in order to compensate for the time delays of the inner loop steering designed for PLA, γ_k, χ_k.

As the designed time advancing only needs to be active for the inner loops defined by PLA, γ_k, χ_k, (dynamic shell 3) some additional low pass filters (time constant T$_3$) have to be embedded in the outer loops in order to compensate the general time advancing of the corresponding steering commands (Index s) coming from the Flight Guidance. The time constant of the engine (T$_E$) is in general less than T$_3$. The additional lead/lag filter slows down the dynamics of the engine to a time constant T$_3$ in order to justify a placement of the engine in dynamic shell 3.

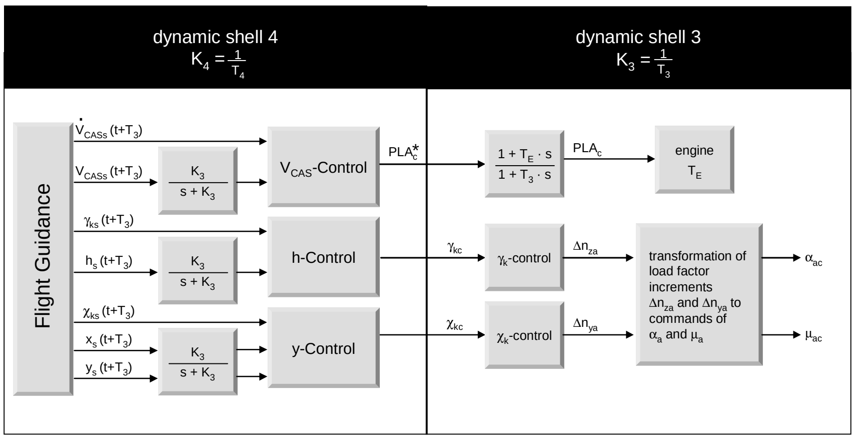

Fig. 10. Principal structure of outer control law loops (Flight Path Steering Mode)

The principle structure of the control laws for the inner loop are depicted in Fig. 11.

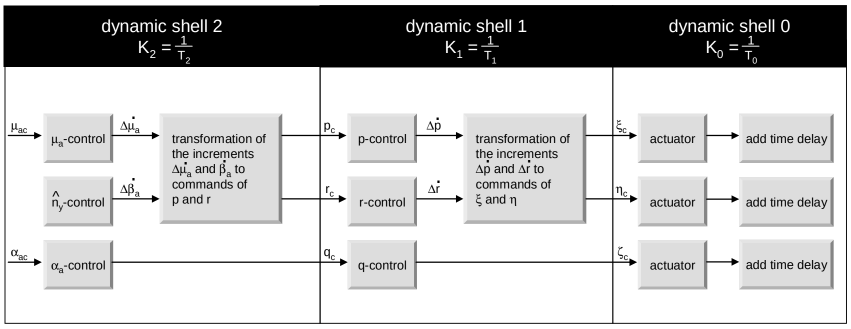

Fig. 11. Principal structure of the inner control law loops (dynamic shells 1 and 2)

Example of a control element within the control cascade

To illustrate the presented concept, an example of a control law element within the control cascade is shown in Fig. 12 for two concrete and one general case. This structure is very simple, but the complexity of control is hidden within the estimators as these have to mirror the dynamics of the state variable in the plant.

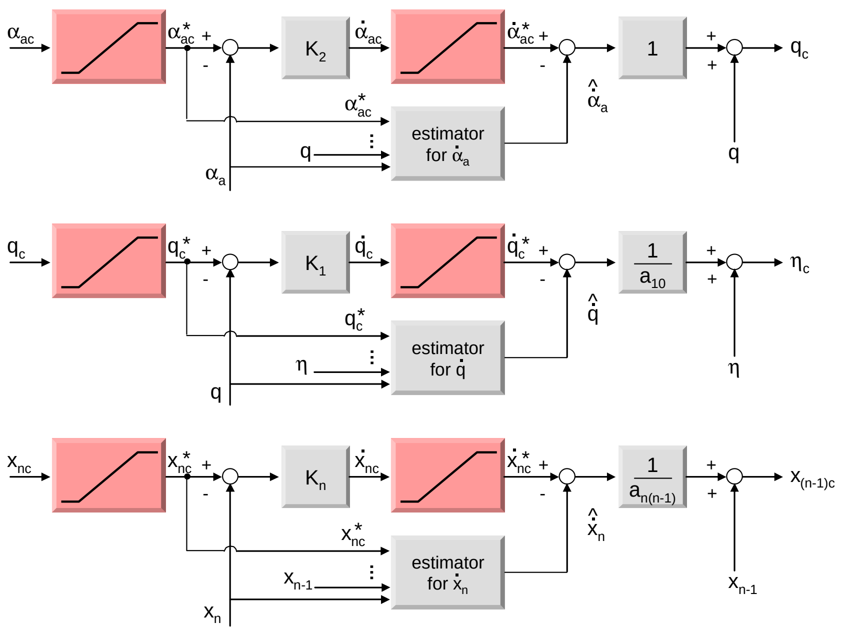

Fig. 12. General structure of one control law element and two examples for q and α_a control

Example of an estimator for differentiated state variables

As already mentioned, the definition of the estimators for differentiated state variables is an essential part of the control concept. The differential equation for a state variable is defined in general by the following equation

$$\dot{x}_n = \ldots + a_{n(n-2)} \cdot x_{(n-2)} + a_{n(n-1)} \cdot x_{(n-1)} + a_{nn} \cdot x_n + a_{n(n+1)} \cdot x_{(n+1)} + \ldots$$

The dependency related to x_n is derived from its command $x_{nc}{}^*$ using a corresponding model (model based compensation) which is represented by a simple, first order, low pass filter. The dependency related to $x_{(n+1)}$ is represented for low frequencies by the real state variable $x_{(n+1)}$ and for high frequencies by the state variable x_n. Further dependencies of state variables in upper shells *(n+2,…)* do not need to be respected since these dependencies are already covered sufficiently by the auxiliary loop for $\dot{x}_n$. Dependencies of state variables related to shell *n-2* and lower have to be represented by corresponding low pass filtered real state variables. The dependency related to state variable x_{n-1} is a dependency without any additional filtering since this state variable is finally cancelled out at the output of the control law element (see Fig. 12). Fig. 13 demonstrates that all dependencies which have to be compensated are using signals which are low pass filtered by a first order filter function.

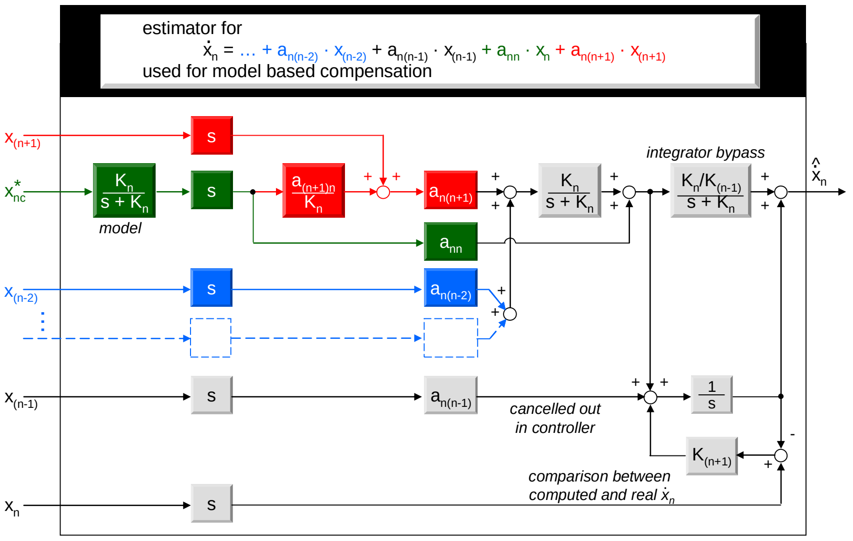
estimator for
$\dot{x}_n = \ldots + a_{n(n-2)} \cdot x_{(n-2)} + a_{n(n-1)} \cdot x_{(n-1)} + a_{nn} \cdot x_n + a_{n(n+1)} \cdot x_{(n+1)}$
used for model based compensation
$x_{(n+1)}$
x^*_{nc}
$x_{(n-2)}$
$x_{(n-1)}$
x_n
$\frac{K_n}{s + K_n}$
model
s
$a_{(n+1)n}$ / K_n
$a_{n(n+1)}$
a_{nn}
$a_{n(n-2)}$
$a_{n(n-1)}$
cancelled out in controller
integrator bypass
$\frac{K_n/K_{(n-1)}}{s + K_n}$
$\frac{1}{s}$
$K_{(n+1)}$
comparison between computed and real $\dot{x}_n$
$\hat{\dot{x}}_n$

Fig. 13. Common definition for an estimator computing the differentiated value of one state variable to be placed within shell *n*.

The modelled dependency related to x_n does not use any direct information of x_n. It is derived completely from its command x_{nc}*. All signals which are used for compensating the described dependencies have to be 'phase advanced' in order to compensate for the time delay of the next inner control loop. The necessary 'phase advance' is realised by a filter function which bypasses the integrator.

The computed result for $\dot{x}_n$ has to be compared with the real differentiated value of x_n. Differences between both values have to be corrected by a corresponding feedback loop with dynamics one order slower than the given dynamic shell (gain $K_{(n+1)}$ as shown in Fig. 13). The time constant $T_{(n+1)}$ of this auxiliary loop is greater than the time constant T_n of the main control (control of x_n) by the factor *e*.

Fig. 14 shows an example of the estimation of $\dot{q}$ according to the above described methodology.

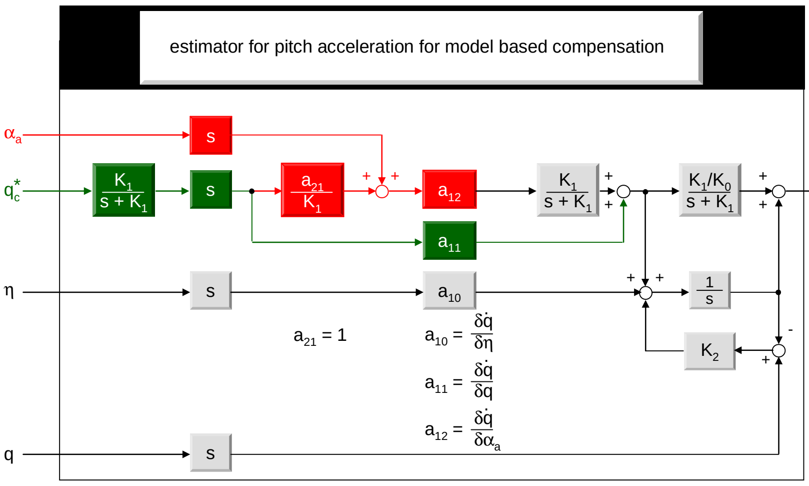
estimator for pitch acceleration for model based compensation
α_a
q_c^*
η
q
$\hat{q}$
$\frac{K_1}{s+K_1}$
s
$\frac{a_{21}}{K_1}$
a_{12}
a_{11}
a_{10}
$\frac{K_1/K_0}{s+K_1}$
$\frac{1}{s}$
K_2
$a_{21} = 1$
$a_{10} = \frac{\delta\dot{q}}{\delta\eta}$
$a_{11} = \frac{\delta\dot{q}}{\delta q}$
$a_{12} = \frac{\delta\dot{q}}{\delta\alpha_a}$

Fig. 14. An example of the implementation of the estimator for $\dot{q}$

Gain adaptation

Every dynamic shell is defined by a certain time constant T_i and a certain overall gain $K_i = 1/T_i$ as described in the sections above. Besides these main gains the control laws as well as the estimators use further gains which have a concrete physical meaning, such as aerodynamic control power gains. All gains which have been determined on an aerodynamic basis are scheduled using altitude, Mach and Angle of Attack (AoA). Both sorts of gains are well defined using a straightforward method for the dimensioning of the gains and cannot be considered a result of an elaborate gain tuning process.

This structured approach has proven to be an important advantage of the presented control law concept as it saves development time and maintains transparency for the control engineer during control law assessments.

Conclusions

Following the Cassidian Auto Flight strategy for UAVs, the workload of the Operator is to be reduced to a supervisory role, such that the Operator can focus and interact with the system in a more optimal way, increasing situational awareness and enhancing safety of flight. This results in alleviation of the Operator from those tasks which can be done automatically, while still enabling sufficient interaction and intervention possibilities. This approach poses new requirements on the Auto Flight system causing the need for new functional and system architectures.

Safety of flight and operator situational awareness is satisfied using a flight guidance which generates an explicit and flyable trajectory, which can be visualized to the Operator. This enhances predictability and therefore situational awareness. Furthermore, a care free auto flight system limits all relevant flight parameters, such that safety of flight can be sufficiently guaranteed.

Low development costs are ensured by using a generic functional architecture which has a high commonality between different UAV platforms. The explicit trajectory generation enables the use of a single autopilot flight path steering mode, without complicated control law moding and subsequent difficulties during validation and verification. Furthermore the presented control law structure has been designed with a focus on low effort regarding gain tuning and uses model based compensation to adapt to air vehicle specific flight dynamics.

The architecture in this paper therefore satisfies the need for safety of flight, combined with optimal situational awareness whilst ensuring low development costs. The presented Auto Flight system has been implemented on the Barracuda UAV demonstrator and successfully flew in a flight test campaign in the summer of 2012.

Acknowledgments The work presented in this paper has been the result of a highly motivated and dedicated team, without which a state-of-the-art flight test demonstration on the Barracuda UAV would not have been possible. The authors wish to express special thanks to Rüdiger Amos, Gerald Klein, Markus Kurze, Maximilian Merz and Jan Waßner for their contributions to the concept, its implementation and this paper.

References

[1] DIN 9300 – Luft- und Raumfahrt; Begriffe, Größen und Formelzeichen der Flugmechanik
[2] R. Brockhaus et al. – Flugregelung, Springer Verlag, 2011
[3] R. Hammon – Auslegung einer robusten Kaskadenregelung, DASA-S-R-1685-A, Universitätsbibliothek Hannover, 1995

Acronyms

AoA	Angle of Attack
ATHR	Auto Throttle
FMS	Flight Management System
PLA	Power Lever Angle
UAV	Uninhabited Air Vehicle