MSc-Thesis - A SIMULINK environment for Flight Dynamics and Control analysis - application to the DHC-2 'Beaver' Part II. Nonlinear analysis of the 'Beaver' autopilot.

Thesis · September 1993

CITATIONS
14

READS
1,571

2 authors, including:

Marc O. Rauw
Delft University of Technology
5 PUBLICATIONS **167** CITATIONS

SEE PROFILE

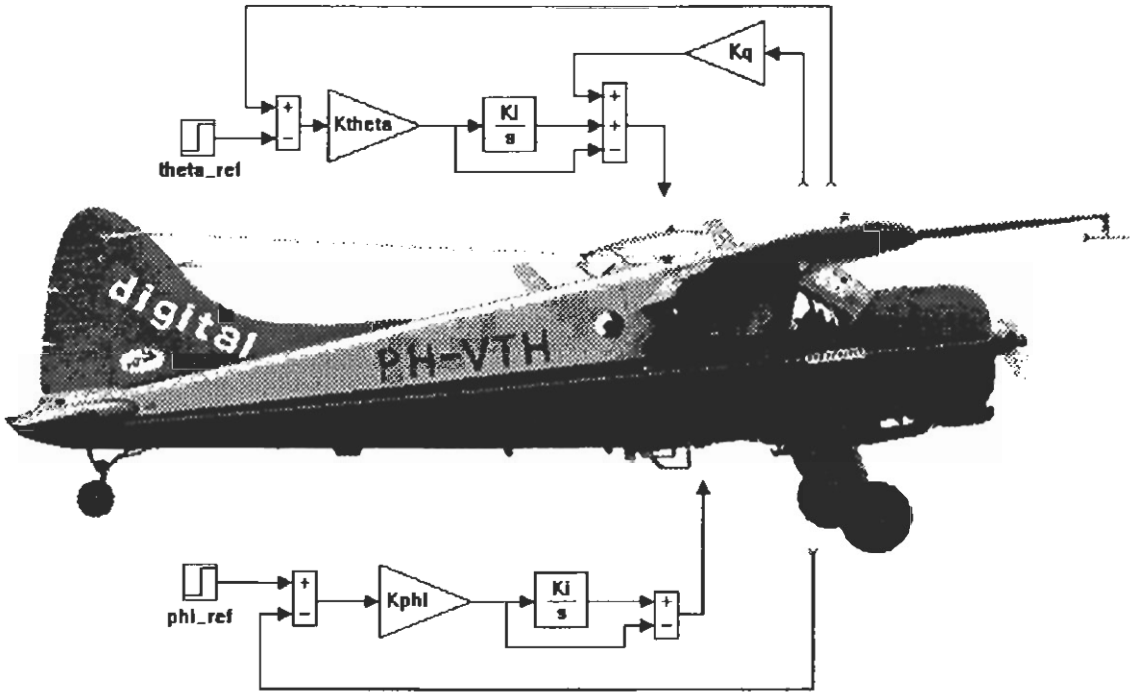
Graduate's Thesis

A SIMULINK environment for Flight Dynamics and Control analysis – application to the DHC-2 'Beaver'

Part II. Nonlinear analysis of the 'Beaver' autopilot

September 1993

M.O. Rauw



TU Delft
Delft University of Technology

Faculty of Aerospace Engineering
Disciplinary Group for Stability and Control

A SIMULINK environment for Flight Dynamics and Control analysis – application to the DHC-2 'Beaver'

Part II. Nonlinear analysis of the 'Beaver' autopilot

Final thesis (MSc-thesis) of M.O. Rauw, Delft University of Technology, Faculty of Aerospace Engineering, Disciplinary Group for Stability and Control, July 1993.

To obtain the MATLAB programs and SIMULINK systems discussed in this report, please contact the author. Up-to-date contact information can be found at http://www.dutchroll.com. The programs and simulation models from report were created with MATLAB 3.5/SIMULINK 1.1 for MS Windows 3.1; this software is for the public domain and may be copied freely. Later versions of the software are available under different conditions, via the above-mentioned website.

Common Documentation License

Version 1.0 — February 16, 2001

Please read this License carefully before downloading or using this material. By downloading or using this material, you are agreeing to be bound by the terms of this License. If you do not or cannot agree to the terms of this License, please do not download or use this material.

0. **Preamble.** The Common Documentation License (CDL) provides a very simple and consistent license that allows relatively unrestricted use and redistribution of documents while still maintaining the author's credit and intent. To preserve simplicity, the License does not specify in detail how (e.g. font size) or where (e.g. title page, etc.) the author should be credited. To preserve consistency, changes to the CDL are not allowed and all derivatives of CDL documents are required to remain under the CDL. Together, these constraints enable third parties to easily and safely reuse CDL documents, making the CDL ideal for authors who desire a wide distribution of their work. However, this means the CDL does not allow authors to restrict precisely how their work is used or represented, making it inappropriate for those desiring more finely-grained control.

1. **General; Definitions.** This License applies to any documentation, manual or other work that contains a notice placed by the Copyright Holder stating that it is subject to the terms of this Common Documentation License version 1.0 (or subsequent version thereof) ("License"). As used in this License:

   1.1 "Copyright Holder" means the original author(s) of the Document or other owner(s) of the copyright in the Document.
   1.2 "Document(s)" means any documentation, manual or other work that has been identified as being subject to the terms of this License.
   1.3 "Derivative Work" means a work which is based upon a pre-existing Document, such as a revision, modification, translation, abridgment, condensation, expansion, or any other form in which such pre-existing Document may be recast, transformed, or adapted.
   1.4 “You" or "Your" means an individual or a legal entity exercising rights under this License.

2. **Basic License.** Subject to all the terms and conditions of this License, You may use, copy, modify, publicly display, distribute and publish the Document and your Derivative Works thereof, in any medium physical or electronic, commercially or non-commercially; provided that: (a) all copyright notices in the Document are preserved; (b) a copy of this License, or an incorporation of it by reference in proper form as indicated in Exhibit A below, is included in a conspicuous location in all copies such that it would be reasonably viewed by the recipient of the Document; and (c) You add no other terms or conditions to those of this License.

3. **Derivative Works.** All Derivative Works are subject to the terms of this License. You may copy and distribute a Derivative Work of the Document under the conditions of Section 2 above, provided that You release the Derivative Work under the exact, verbatim terms of this License (i.e., the Derivative Work is licensed as a "Document" under the terms of this License). In addition, Derivative Works of Documents must meet the following requirements:

   (a) All copyright and license-notices in the original Document must be preserved.
   (b) An appropriate copyright notice for your Derivative Work must be added adjacent to the other copyright notices.
   (c) A statement briefly summarizing how your Derivative Work is different from the original Document must be included in the same place as your copyright notice.
   (d) If it is not reasonably evident to a recipient of your Derivative Work that the Derivative Work is subject to the terms of this License, a statement indicating such fact must be included in the same place as your copyright notice.

4. **Compilation with Independent Works.** You may compile or combine a Document or its Derivative Works with other separate and independent documents or works to create a compilation work ("Compilation"). If included in a Compilation, the Document or Derivative Work thereof must still be provided under the terms of this License, and the Compilation shall contain (a) a notice specifying the inclusion of the Document and/or Derivative Work and the fact that it is subject to the terms of this License, and (b) either a copy of the License or an incorporation by reference in proper form (as indicated in Exhibit A). Mere aggregation of a Document or Derivative Work with other documents or works on the same storage or distribution medium (e.g. a CD-ROM) will not cause this License to apply to those other works.

5. **NO WARRANTY**. THE DOCUMENT IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, AND THE COPYRIGHT HOLDER EXPRESSLY DISCLAIMS ALL WARRANTIES AND/OR CONDITIONS WITH RESPECT TO THE DOCUMENT, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY, OF SATISFACTORY QUALITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY, OF QUIET ENJOYMENT, AND OF NONINFRINGEMENT OF THIRD PARTY RIGHTS.

6. **LIMITATION OF LIABILITY.** UNDER NO CIRCUMSTANCES SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO THIS LICENSE OR YOUR USE, REPRODUCTION, MODIFICATION, DISTRIBUT ION AND/OR PUBLICATION OF THE DOCUMENT, OR ANY PORTION THEREOF, WHETHER UNDER A THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EVEN IF THE COPYRIGHT HOLDER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY.

7. **Trademarks.** This License does not grant any rights to use any names, trademarks, service marks or logos of the Copyright Holder (collectively "Marks") and no such Marks may be used to endorse or promote works or products derived from the Document without the prior written permission of the Copyright Holder.

8. **Versions of the License.** Apple Computer, Inc. ("Apple") may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Once a Document has been published under a particular version of this License, You may continue to use it under the terms of that version. You may also choose to use such Document under the terms of any subsequent version of this License published by Apple. No one other than Apple has the right to modify the terms applicable to Documents created under this License.

9. **Termination.** This License and the rights granted hereunder will terminate automatically if You fail to comply with any of its terms. Upon termination, You must immediately stop any further reproduction, modification, public display, distribution and publication of the Document and Derivative Works. However, all sublicenses to the Document and Derivative Works which have been properly granted prior to termination shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive, including but not limited to Sections 5, 6, 7, 9 and 10.

10. **Waiver; Severability; Governing Law.** Failure by the Copyright Holder to enforce any provision of this License will not be deemed a waiver of future enforcement of that or any other provision. If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License will be enforced to the maximum extent permissible so as to effect the economic benefits and intent of the parties, and the remainder of this License will continue in full force and effect. This License shall be governed by the laws of the United States and the State of California, except that body of California law concerning conflicts of law.

EXHIBIT A

The proper form for an incorporation of this License by reference is as follows:

"Copyright (c) [year] by [Copyright Holder's name].
This material has been released under and is subject to the terms of the Common Documentation License, v.1.0, the terms of which are hereby incorporated by reference. Please obtain a copy of the License at http://www.opensource.apple.com/cdl/ and read it before using this material. Your use of this material signifies your agreement to the terms of the License."

TECHNISCHE UNIVERSITEIT DELFT
FACULTEIT DER LUCHTVAART- EN RUIMTEVAARTTECHNIEK
Vakgroep Stabiliteit en Besturing

Delft, 5 november 1992

Afstudeeropdracht voor de heer M.O. Rauw, studienr. 680828.

Voor het effektief ontwerpen en evalueren van regelsystemen voor vliegtuigbesturing wordt integratie van lineaire en niet-lineaire systeemanalyse op een PC of werkstation nagestreefd. Als eerste aanzet hiertoe is voor het voorbereidende afstudeerwerk (Lit. 1), het niet-lineaire simulatiemodel van de De Havilland DHC-2 'Beaver' uit (Lit. 2) geïmplementeerd in SIMULINK. Voortbouwend op het voorbereidende afstudeerwerk wordt u het volgende gevraagd:

1) De niet-lineaire SIMULINK modellen van de 'Beaver' moeten worden aangevuld met regelwetten van de autopilot. De juiste werking van alle autopilot modes dient vervolgens te worden geëvalueerd, waarbij met name de koppeling van symmetrische en asymmetrische bewegingen aandacht vereist. Eventuele modificaties en/of aanvullingen in de regelwetten moeten uitvoerig worden beargumenteerd en gedocumenteerd.

2) Het motormodel van de 'Beaver' moet worden toegevoegd, zodat ook autothrottle regelsystemen geëvalueerd kunnen worden. Daarnaast is toevoeging van een model voor atmosferische turbulentie gewenst.

3) Er moet een gebruikersvriendelijke 'toolbox' voor het in SIMULINK implementeren van simulatie modellen van willekeurige vliegtuigen worden ontwikkeld. Bij het ontwikkelen van deze toolbox blijven de voor de 'Beaver' geïmplementeerde modellen als voorbeeld dienen. Ideeën m.b.t. de opbouw van de toolbox kunnen o.a. worden ontleend aan Lit. 3, 4 en 5.
De simulatiemodellen moeten worden geoptimaliseerd voor implementatie van regelsystemen, die met de grafische blokdiagram-representatie van SIMULINK kunnen worden toegevoegd. Met de linearisatie routines van SIMULINK moet het de gebruiker ten slotte mogelijk worden gemaakt om een lineair vliegtuigmodel in een gewenste vlieg- en vliegtuigtoestand te ontwikkelen.

De resultaten van het onderzoek dienen te worden vastgelegd in een volledig en overzichtelijk rapport. Het deel over de 'Beaver' autopilot dient te worden voorzien van een beknopte, doch volledige, beschrijving van alle autopilot modes.

Literatuur:

1. Voorbereidend afstudeerverslag (M.O. Rauw);
2. R.T.H. Tjee, J.A. Mulder: Stability and Control Derivatices of the De Havilland DHC-2 'Beaver' Aircraft. Report LR-556, TU-Delft, 1988;

3. R.D. Colgren: A Workstation for the Integrated Design and Simulation of Flight Control Systems. Lockheed Aeronautical Systems Company, Burbank California;
4. R.F. Stengel, S. Sircar: Computer-Aided Design of Flight Control Systems. AIAA-91-2677-CP, 1991;
5. E.L. Duke, B.P. Patterson, R.F. Antoniewicz: User's manual for LINEAR, a Fortran Program to derive linear aircraft models. NASA TP-2768, 1987.

Mentoren	: Ir. S. Bennani, Ir. J.C. van der Vaart, Prof.dr.ir. J.A. Mulder
Thema	: Vliegproeven en Besturing
Projectgroep	: I-1 Active Flight Controls
Aanvang opdracht	: 15 augustus 1992

Preface.

This report is the result of my graduation research, which has been done in the period from may 1992 to the date of publishing, starting with some preliminary work which was published in ref.[13]. This report describes the construction of a model library in SIMULINK for evaluating automatic control systems for the 'Beaver' aircraft and it presents a detailed analysis of the 'Beaver' autopilot as a case-study. The report has been divided in two separate parts:

- Part I gives a detailed description of the SIMULINK models which were developed during this research. These models form a further improvement of the ones described in my preliminary thesis, ref.[13]. Part I of this report can be read independently of ref.[13], although there is of course some overlap (this version is better, however!).

- Part II describes a case study of the 'Beaver' autopilot, which has been implemented in SIMULINK, using the models from part I. The autopilot structure is discussed and many simulation results are shown. Part II has been structured as a separate report, so the results of the autopilot analysis can be read independently, without having detailed knowledge of the SIMULINK implementation.

Both parts of the report can (and should!) be used as references for future research on Automatic Aircraft Control Systems (AACSs), aimed at the new DUT/NLR laboratory aircraft, the Cessna 'Citation II'. The process which leads from preliminary AACS design to in-flight testing of the system is quite complex and should be mastered in detail to ensure safe and satisfying AACS operation in flight. This can only be achieved if appropriate hardware and software tools for system analysis are developed.

The reader should be familiar with 'classical' control theory. Some basic knowledge about aircraft dynamics and MATLAB is also required. A short introduction to SIMULINK has been included in part I.

Acknowledgements.

I wish to express my thanks to all members of the 'Beaver Taskforce' for their support and for the pleasant working climate. The taskforce was founded by prof.dr.ir. Bob Mulder and lead by dr.ir. Hans van der Vaart of the disciplinary group for Stability and Control, for redesign, nonlinear evaluation, implementation, and in-flight testing of the 'Beaver' autopilot. The other taskforce members were Patrick Wever, Bardia Ghahramani, Marcel van Witzenburg, Samir Bennani, and Zainal Abidin. Later also Eric Kruijsen joined the group. Thank you all for the constant feedback of ideas!

Special thanks go to Samir Bennani for letting me (ab-?) use his personal computer. I really wouldn't have known what to do without it. I also would like to thank my father for his support during the preparation of this report, which really helped me out when time became a real issue.

Marc Rauw, Delft, september 1993.

Summary of part II.

This report describes the nonlinear analysis of a digital autopilot for the De Havilland DHC-2 'Beaver' laboratory aircraft of the Faculty of Aerospace Engineering, using the flexible SIMULINK environment from part I. The implementation of the control laws has been described in detail, because it gives a clear illustration of the practical use of the models and tools from part I, and because the autopilot simulation structure will be used to analyze the results of flight tests of the 'Beaver' autopilot.

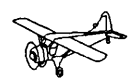
The 'Beaver' autopilot, which is based upon 'classical' linear control theory, serves as an example for a similar *baseline autopilot*, to be developed for the new Cessna Citation II laboratory aircraft. It is very important to spend time in developing such 'classical' aircraft controllers, because it is the only way to encounter all practical problems which may arise in the autopilot development process in practice. The *baseline autopilot* for the Cessna Citation II will serve as a reference system, against which all new, more advanced automatic controllers will be measured.

The nonlinear simulations of the 'Beaver' autopilot, presented in this report, basically confirmed the linear results from ref.[16], although the inclusion of longitudinal-lateral cross-coupling effects in the nonlinear models yielded some important adjustments of the control laws. In particular, an effective turncompensation law could be developed, using results of nonlinear simulations. Furthermore, nonlinear simulations were also used for fine-tuning the limiters. In general, the performance of the control laws was satisfactory throughout the whole flight-envelope, albeit with a small degradation for the lowest airspeeds. For the 'Beaver', the influence of the propeller-slipstream causes large sideslip angles if the velocity or engine power is changed. In turns, the sideslip angle was suppressed effectively by means of a turncoordinator which worked well, even for the fast turns that were allowed for the 'Beaver' autopilot.

The experiences from this analysis of the 'Beaver' autopilot have been taken into account when redesigning the simulation models from this report (part I *and* part II). In order to be able to effectively assess all phenomena which may or may not affect the autopilot performance, it absolutely necessary to use such a fast, flexible environment for linear and nonlinear analysis. In this report, this has been illustrated effectively for the analysis of the influence of the Least Significant Bit of 4 feet, used for the altitude measurements.

The control laws can be implemented in SIMULINK in a way which closely resembles the structure of the software for the Flight Control Computers of the aircraft. For future research aimed at the Cessna Citation II, it is strongly recommended to automate the conversion of the control laws from the SIMULINK structure to real-time flightsimulators and the FCCs of the aircraft as much as possible, to reduce both the development time and the risks of making conversion errors.

Contents (part II).



List of tables.

List of figures.

Symbols and definitions (part II).

Due to the very large number of variables which will be used in this report, it is sometimes necessary to use the same symbols for different variables. Often, the meaning of the symbol follows directly from the context of its use. Some symbols will be overlined to prevent confusion with other quantities, using the same symbol. Vectors will be typeset in **boldface** and matrices and reference frames (reference axes) are denoted with CAPITALS. Software names are typeset in SMALL CAPITALS and commands from software packages in *CAPITALS + ITALICS*. *Italics* will also be used for examples, equations, and symbols.

0.1 Symbols.

b	wingspan, [m]
$\bar{c}$	mean aerodynamic chord, [m]
CD	course datum, [rad]
d_{gs}	distance from aircraft $\perp$ glideslope reference plane (see section 2.2.4), [m].
d_{loc}	distance from aircraft $\perp$ localizer reference plane (see section 2.3.3), [m]
dpt	$\frac{\Delta p_t}{\frac{1}{2}\rho V^2}$, [-]
F_B	body-fixed reference frame
F_E	earth-fixed reference frame
F_F	runway ('field') reference frame
F_S	stability reference frame
F_V	vehicle-carried vertical reference frame
g	acceleration of gravity, [m s^{-2}]
H	altitude, [m]
H_f	altitude of aircraft above the runway, [m]
H_{RW}	altitude of runway above sea level, [m]
L_g	general scale length for turbulence, [m]
L_{gs}	scale length for glideslope noise, [m]
L_{loc}	scale length for localizer noise, [m]
L_u	scale length for turbulence velocity along X_B-axis, [m]
L_v	scale length for turbulence velocity along Y_B-axis, [m]
L_w	scale length for turbulence velocity along Z_B-axis, [m]
n	engine speed, [RPM]
p	angular roll-rate, [rad s^{-1}]
P	engine power, [N m s^{-1}]
p_z	manifold pressure, ["Hg]
q	angular pitch-rate, [rad s^{-1}]
r	angular yaw-rate, [rad s^{-1}]
R_{gs}	ground distance from aircraft to glideslope transmitter, [m]
R_{loc}	ground distance from aircraft to localizer transmitter, [m]
R_{VOR}	ground distance from aircraft to VOR transmitter, [m]
S	wing area, [m^2]
t	time, [s]
t_s	sampling time (stepwith for discrete systems), [s]

u	velocity along X_B-axis, [m s^{-1}]
u_w	wind velocity component along X_B-axis, [m s^{-1}]
v	velocity along Y_B-axis, [m s^{-1}]
v_w	wind velocity component along Y_B-axis, [m s^{-1}]
V	true airspeed, [m s^{-1}]
$V\Delta\delta_a$	command signal to FCS (needed for change in deflection of ailerons); [V]
$V\Delta\delta_e$	command signal to FCS (needed for change in deflection of elevator); [V]
$V\Delta\delta_r$	command signal to FCS (needed for change in deflection of rudder); [V]
w	velocity along Z_B-axis, [m s^{-1}]
W	aircraft weight, [N]
w_w	wind velocity component along Z_B-axis, [m s^{-1}]
x_{gs}	X-coordinate of glideslope antenna relatively to F_F (see appendix C, part I), [m]
x_{loc}	X-coordinate of localizer antenna relatively to F_F (see appendix C, part I), [m]
x_{VOR}	X-position of VOR antenna relatively to F_E, [m]
y_{gs}	Y-coordinate of glideslope antenna relatively to F_F (see appendix C, part I), [m]
y_{VOR}	Y-position of VOR antenna relatively to F_E, [m]
α	angle of attack, [rad]
β	angle of sideslip, [rad]
γ	flight path angle, [rad]
γ_{gs}	reference flight path angle at glideslope, [rad]
Γ_{loc}	angle between *localizer* reference plane and line from ground position of aircraft to *localizer* antenna (see section 2.3.3), [rad]
Γ_{VOR}	angle between CD, and VOR bearing where aircraft flies, [rad]
δ_a	angle of deflection of ailerons ($\delta_a = \delta_{a_{right}} - \delta_{a_{left}}$), [rad]
δ_e	angle of deflection of elevator, [rad]
δ_f	angle of deflection of flaps, [rad]
δ_r	angle of deflection of rudder, [rad]
Δ	increment, [-]
$\Delta\delta_a$	change in aileron deflection, [rad]
$\Delta\delta_e$	change in elevator deflection, [rad]
$\Delta\delta_r$	change in rudder deflection, [rad]
Δp_t	increase of total pressure over the propeller, [N m^{-2}]
Δn	change in engine speed, [RPM]
Δp_z	change in manifold pressure, ["Hg]
ε	angle between *glideslope* reference plane, and line through aircraft and *glideslope* antenna (see appendix C), [rad]
θ	pitch angle, [rad]
σ	standard deviation
σ_{gs}	standard deviation of glideslope noise, [A]
σ_{loc}	standard deviation of localizer noise, [A]
σ_u	standard deviation of turbulence velocity in X_B-direction, [m/s]
σ_v	standard deviation of turbulence velocity in Y_B-direction, [m/s]
σ_w	standard deviation of turbulence velocity in Z_B-direction, [m/s]
τ	time interval, time constant of filters

φ	roll angle, [rad]
Φ	bank angle, [rad]
χ	azimuth angle, [rad]
ψ	yaw angle, [rad]
ψ_{RW}	runway heading, [rad]

0.2 Vectors. *(definitions between brackets valid for simulation model BEAVER and APILOT3, see text)*

$\mathbf{u}(t)$	input vector ($\mathbf{u} = [\ \mathbf{u}_a^T\ \ \mathbf{u}_t^T\ \ \mathbf{u}_{wind}^T\ ]^T$)
$\mathbf{u}_a$	vector with aerodynamic inputs ($\mathbf{u}_a = [\ \delta_e\ \ \delta_a\ \ \delta_r\ \ \delta_f\ ]^T$)
$\mathbf{u}_t$	vector with engine inputs ($\mathbf{u}_t = [\ n\ \ p_z\ ]^T$)
$\mathbf{u}_w$	vector with wind velocities and accelerations ($\mathbf{u}_{wind} = [\ u_w\ \ v_w\ \ w_w\ \ \dot{u}_w\ \ \dot{w}_w\ \ \dot{v}_w\ ]^T$)
$\mathbf{V}$	true airspeed vector
$\mathbf{V}_w$	wind velocity vector
$\mathbf{x}(t)$	state vector ($\mathbf{x} = [\ V\ \alpha\ \beta\ p\ q\ r\ \psi\ \theta\ \varphi\ x_e\ y_e\ H\ ]^T$)
$\mathbf{y}(t)$	output vector ($\mathbf{y} = [\ \mathbf{x}^T\ \ \dot{H}\ \ \mathbf{y}_{dl}^T$)
$\mathbf{y}_{dl}$	vector with dimensionless rotational velocities ($\mathbf{y}_{dl} = \left[\frac{pb}{2V}\ \frac{q\bar{c}}{V}\ \frac{rb}{2V}\right]^T$)

0.3 Matrices.

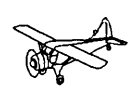
A	system matrix of linear state-space system
B	input matrix of linear state-space system
T_{ab}	transformation matrix (from reference frame F_a to F_b)
Θ	transformation matrix for first Euler rotation (section 0.7.2)
Φ	transformation matrix for second Euler rotation (section 0.7.2)
Ψ	transformation matrix for third Euler rotation (section 0.7.2)

0.4 Indices.

0	nominal value
a	aileron
e	elevator
f	flaps
gs	glideslope
loc	localizer
MA	Moving Average filter
r	rudder
ref	reference value of a signal (used for inputs to control loops)
RW	runway
VOR	VOR
w	wind velocity (-components along body axes)

0.5 Superscripts.

T	transpose of matrix or vector
.	derivative with respect to time



0.6 Abbreviations.

AACS	Automatic Aircraft Control System
ALH	Altitude Hold mode of autopilot
ALS	Altitude Select mode of autopilot
CD	Course Datum
c.g.	centre of gravity
DHC	De Havilland of Canada Ltd.
DLS.	Data Logging system.
DME	Distance Measuring Equipment
DUT	Delft University of Technology
FCC	Flight Control Computer
FCS	Flight Control System
GA	Go Around mode of autopilot
GS	Glideslope-coupled mode of autopilot
HH	Heading Hold mode of autopilot
HS	Heading Select mode of autopilot
ILS	Instrument Landing System
LOC	Localizer-coupled mode of autopilot
LSB	Least Significant Bit
MA	Moving Average
NAV	Navigation mode of autopilot
NLR	Nationaal Lucht- en Ruimtevaartlaboratorium (Dutch Aerospace Laboratory)
PAH	Pitch Attitude Hold mode of autopilot
RAH	Roll Attitude Hold mode of autopilot
STOL	Short Take Off and Landing
TAS	True Airspeed, V
VOR	Very high frequency Omnidirectional Range

0.7 Reference frames and sign conventions.

0.7.1 Definitions.

The definitions of the reference frames used within this report is given below. The equations for translational and rotational velocities are referenced to the body axes F_B. The aircraft *attitude* is defined by the Euler angles ψ, θ, and φ, for which purpose the vehicle-carried vertical reference frame F_V is introduced. The aircraft *position* is defined with respect to the earth-fixed reference frame F_E.

The body-fixed reference frame F_B.

The body-fixed reference frame of the aircraft is a right-handed orthogonal system $O_BX_BY_BZ_B$. The origin O_B lies in the centre of gravity of the aircraft. The $X_BO_BZ_B$ plane coincides with the aircraft's plane of symmetry if it is symmetric, or is located in a plane, approximating what would be the plane of symmetry. The X_B-axis is directed towards the nose of the aircraft, the Y_B-axis points to the right wing (starboard), and the Z_B-axis towards the bottom of the aircraft. The positive directions for p, q, r, u, v, w, F_x, F_y, F_z, L, M, and N are shown in figure 0-1.

The stability reference frame F_S.

The stability reference frame $O_SX_SY_SZ_S$ is a special body-fixed frame, used in the study of small deviations from a nominal flight condition. The reference frames F_B and F_S differ in the orientation of the X-axis. The X_S-axis is chosen parallel to the projection of V on the $O_BX_BZ_B$-plane (if the aircraft is symmetric, this is the plane of symmetry), or parallel to V itself in case of a symmetrical nominal flight condition. The Y_S-axis coincides with the Y_B-axis.

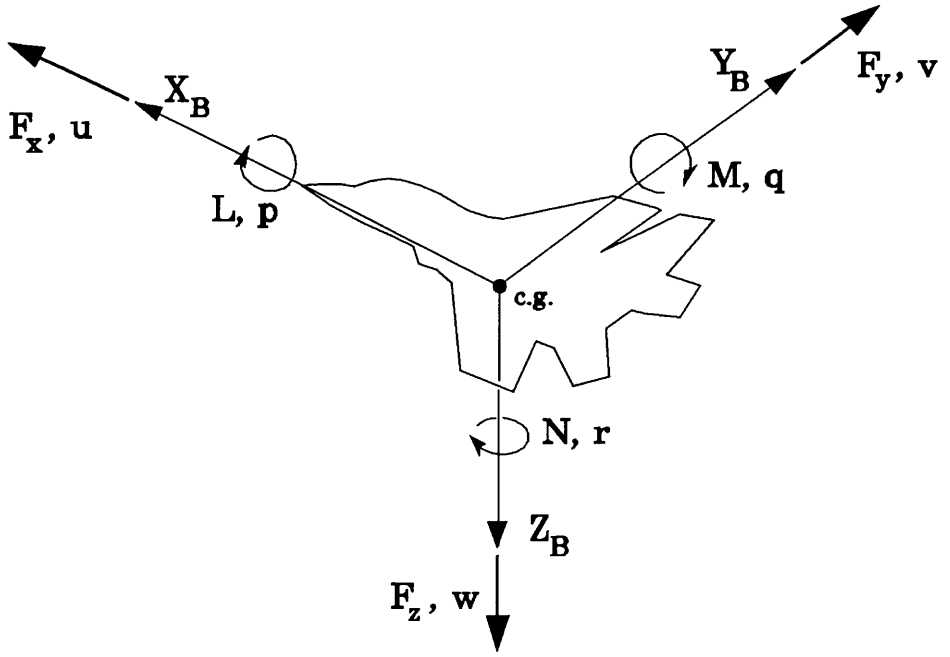


Figure 0-1. The body-axes reference frame F_B and the positive directions of the body-axes forces F_x, F_y, F_z, moments L, M, N, velocities u, v, w, and rotational velocities p, q, r.

The flight path reference frame F_W.

The flight path reference frame $O_WX_WY_WZ_W$, also called the wind reference frame, has its origin at the c.g. of the aircraft. The X_W axis is aligned with the velocity vector of the aircraft. The Y_W-axis coincides with the Y_B-axis. The Z_W and Z_S-axes are parallel.

The earth-fixed reference frame F_E.

The earth-fixed reference frame, also called the topodetic reference frame, is a right-handed orthogonal system $O_EX_EY_EZ_E$, which is considered to be fixed in space. Its origin can be placed at an arbitrary position, but will be chosen to coincide with the aircraft's centre of gravity at the start of a flight test manoeuvre. The Z_E-axis points downwards, parallel to the local direction of the gravitation. The X_E-axis is directed north, the Y_E-axis east. For the

simulation of ILS approaches a runway-fixed reference frame will be introduced and a beacon-fixed reference frame will be used for simulating VOR navigation.

The vehicle-carried vertical axis reference system F_V.

The vehicle-carried vertical axis system $O_VX_VY_VZ_V$ has its origin at the c.g. of the aircraft. The X_V-axis is directed to the north, the Y_V-axis to the east, and the Z_V-axis downwards (parallel to the local direction of gravity).

The runway-fixed reference frame F_F.

A runway-fixed reference frame $O_FX_FY_FZ_F$ is used for simulation of ILS approaches. The origin O_F is located at the point of intersection of the runway threshold and runway centerline. The X_F-axis is directed along the runway centerline, in take-off and landing direction, Y_F points to the left (seen from an approaching aircraft), Z_F points downwards. See appendix C of part I.

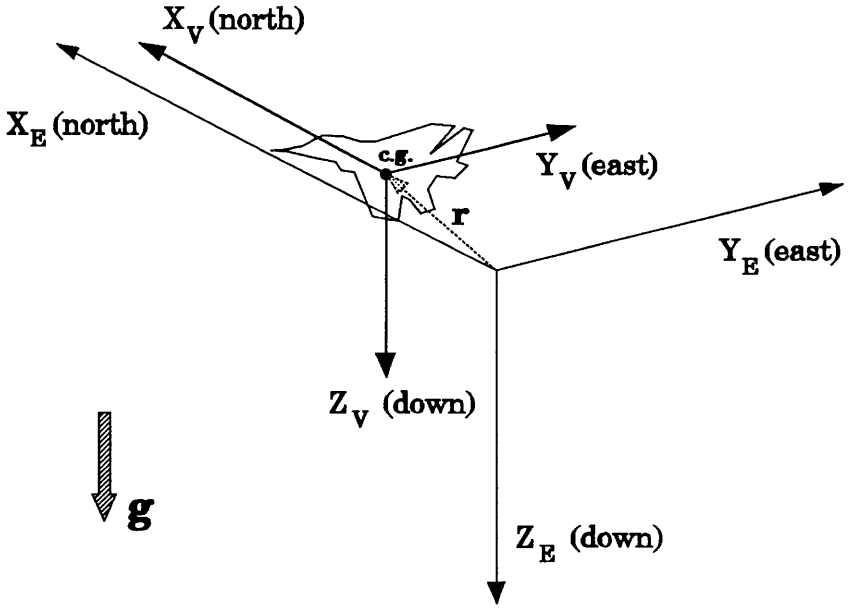


Figure 0-2. Relationship between the vehicle-carried vertical reference frame F_V and the earth-fixed reference frame F_E.

0.7.2 Relationships between the reference frames.

In figure 0-2 the relationship between the earth-fixed and vehicle-carried vertical axis sytems is shown. F_E and F_V differ only in the position of their origins. The relationship between the vehicle-carried vertical and body axes is shown in figure 0-3. The Euler angles ψ, θ, and φ define the orientation of F_B with respect to F_V, hence they define the attitude of the aircraft with

respect to the earth. The transformation matrices expressing each of the Euler rotations separately are:

$$\Psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{0-1}$$

$$\Phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \tag{0-2}$$

The total transformation matrix from F_V to F_B then becomes:

$$T_{VB} = \Phi \cdot \Theta \cdot \Psi =$$

$$= \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\varphi - \sin\psi\cos\varphi & \sin\psi\sin\theta\sin\varphi + \cos\psi\cos\varphi & \cos\theta\sin\varphi \\ \cos\psi\sin\theta\cos\varphi + \sin\psi\sin\varphi & \sin\psi\sin\theta\cos\varphi - \cos\psi\sin\varphi & \cos\theta\cos\varphi \end{bmatrix} \tag{0-4}$$

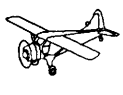
so the relation between a vector y_B in F_B and y_V in F_V is:

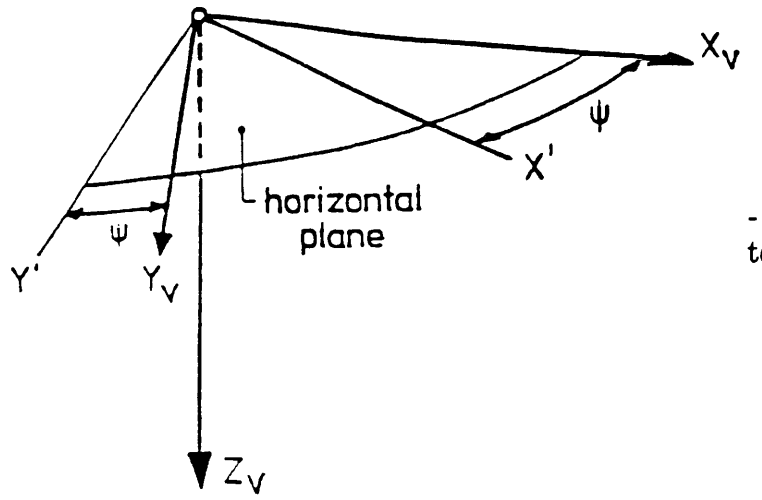
$$y_B = T_{VB} \cdot y_V \tag{0-5}$$

The orientation of the flight path axes with respect to the vehicle-carried vertical axes can also be expressed in terms of Euler angles (χ, γ and μ). This is shown in figure 0-4. The relationships between the body, flight path, and stability axes are shown in figure 0-5. All three axis sytems have their origin at the aircraft's centre of gravity. The X_W-axis is aligned with the velocity vector of the aircraft. The orientation of the flight path axis relatively to the body axes is defined by the angle of attack α and the sideslip angle β. The stability axis reference system is displaced from the flight path axis system by a rotation β and from the body axis sytem by a rotation $-\alpha$.

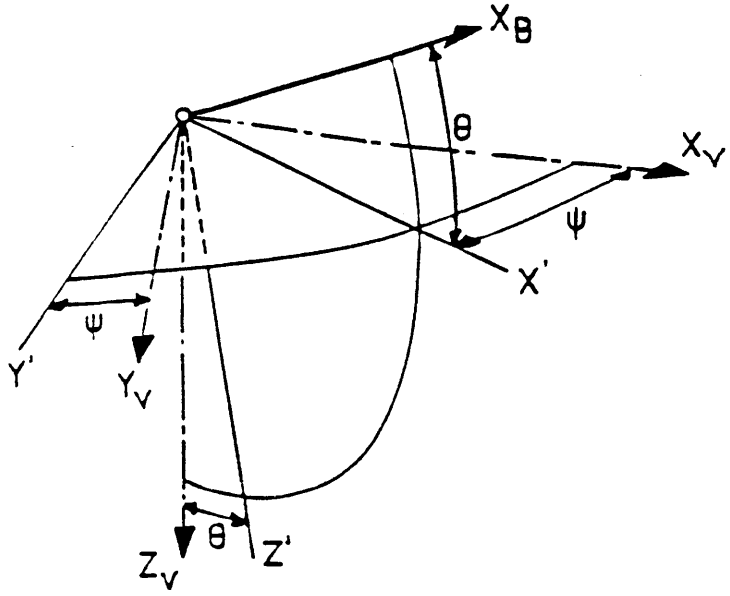
0.7.3 Sign conventions for deflections of control surfaces.

Figure 0-6 shows how the positive deflections of the control surfaces are measured. The positive elevator deflection is measured downwards: a positive δ_e gives a pitch-down movement of the aircraft. The rudder and ailerons deflections are positive if they force the aircraft to move leftwards. If one aileron deflection has a positive sign, the other is negative. The 'total' aileron deflection is therefore defined as: $\delta_a = \delta_{a_{right}} - \delta_{a_{left}}$. The positive flap deflection is measured downwards, similar to the elevator deflection. A positive δ_f corresponds to an increase in lift.

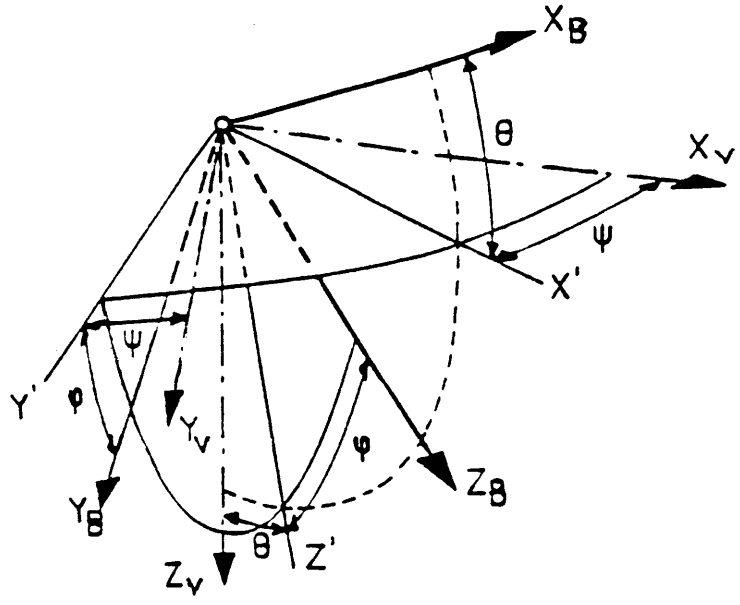




- A rotation by ψ about the Z_V-axis to the intermediate position $X'Y'Z_V$.

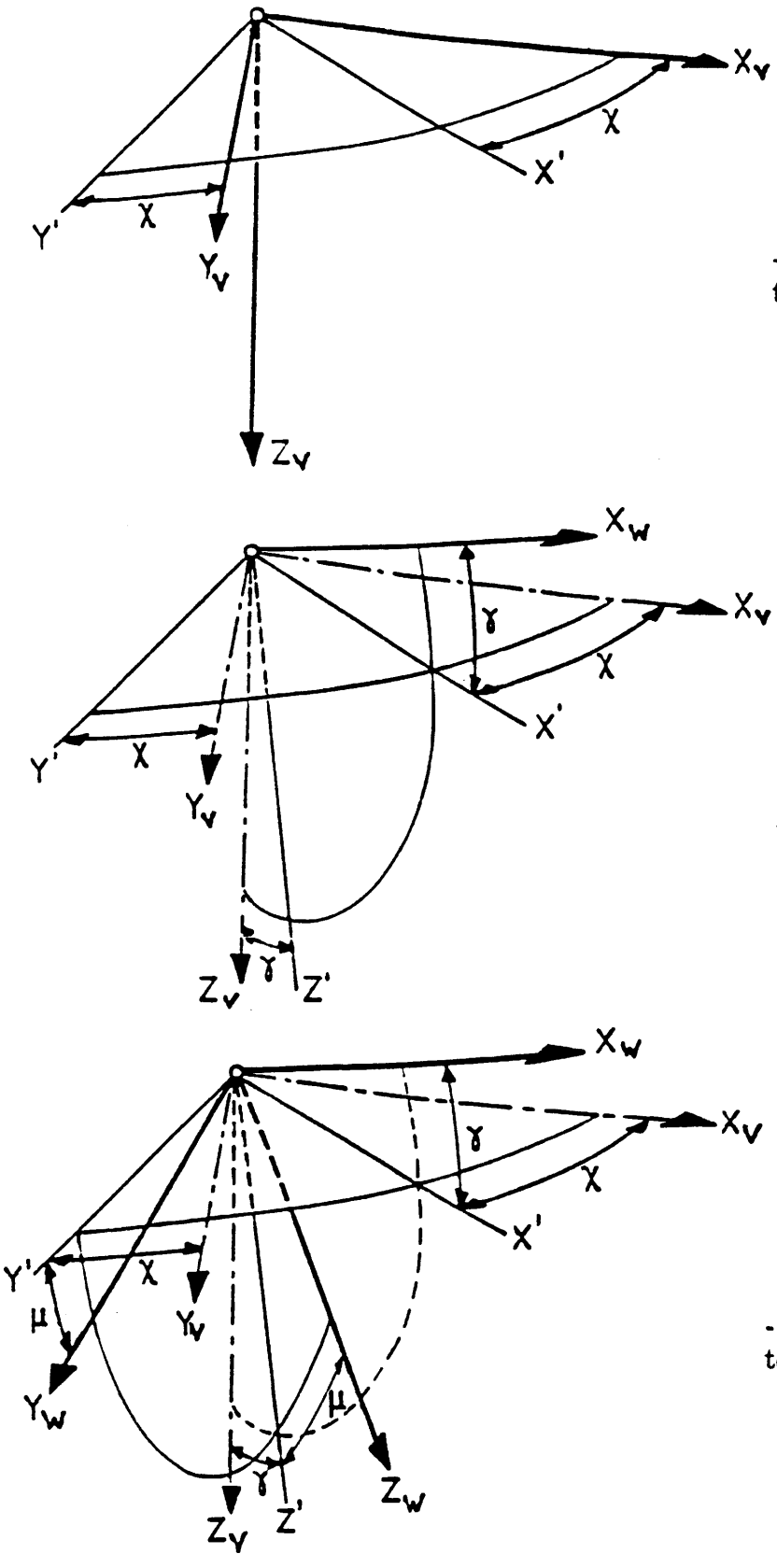


- A rotation by θ about the Y'-axis to the intermediate position $X_BY'Z'$.



- A rotation by φ about the X_B-axis to the final position $X_BY_BZ_B$.

Figure 0-3. Relationship between the vehicle-carried vertical reference frame F_V and the body-fixed reference frame F_B.



- A rotation by χ about the Z_V-axis to the intermediate position $X'Y'Z_V$.

- A rotation by γ about the Y'-axis to the intermediate position $X_WY'Z'$.

- A rotation by μ about the X_W-axis to the final position $X_WY_WZ_W$.

Figure 0-4. Relationship between the flight path reference frame F_W and the body-fixed reference frame F_B.

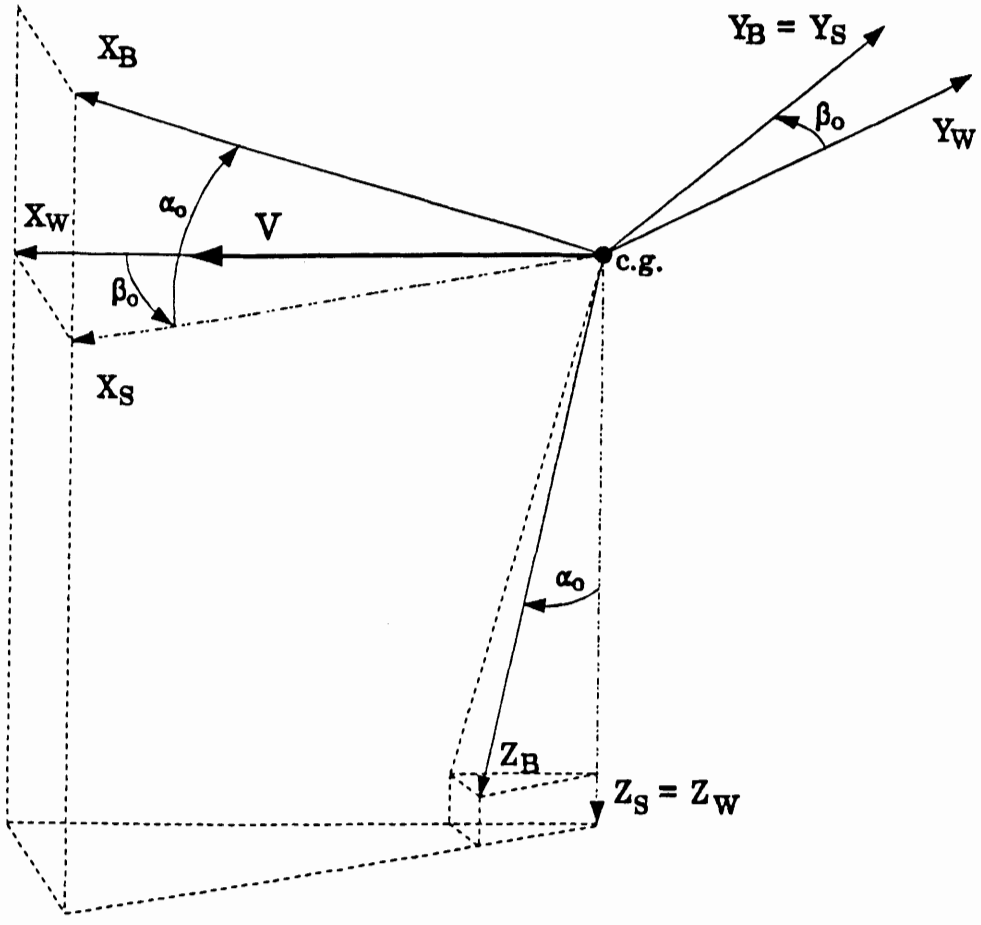


Figure 0-5. Relationships between body-fixed reference frame F_B, flight path reference frame F_W, and stability reference frame F_S.

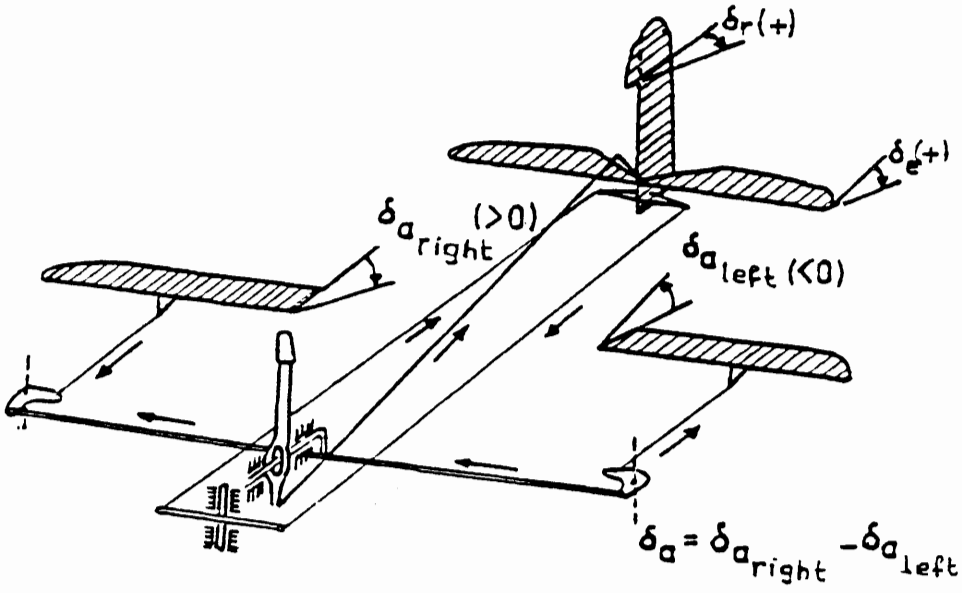


Figure 0-6. Sign conventions for control surface deflections.

Introduction to Part II.

During the late Eighties, a digital autopilot for the DHC-2 'Beaver' laboratory aircraft of the faculty of Aerospace Engineering has been developed by several former graduate students of the Disciplinary Group for Stability and Control. When this autopilot was evaluated in flight in 1990, its performance did not completely meet the expectations [9]. Further evaluation of the autopilot, started at the end of 1991, showed that some control laws were too slow, inadequately damped or even instable. Also, errors in the implementation within the Flight Control Computer software were discovered, and it was shown that the hardware-related minimum sampling time of 0.1 seconds was not fast enough to damp the fastest characteristic motions of the 'Beaver' sufficiently [15, 16].

Based upon these results, a 'Taskforce' was founded to redesign the 'Beaver' autopilot. New, more powerful computer hardware was installed in the aircraft and all control laws were optimized, analyzed with nonlinear simulations, and implemented in C-programs for the new, faster, Flight Control Computers [16]. The flight-test hardware and software was installed by technicians of the faculty and calibrated by some taskforce members [17], and finally, new flight tests were carried out. All this was done in a very short time, because of the upcoming replacement of the 'Beaver' by a Cessna Citation II jet.

This report reviews the nonlinear analysis of the 'Beaver' autopilot, using the SIMULINK environment from part I. The inclusion of the control laws in the simulation package and their use for nonlinear simulations will be described in detail, giving a clear illustration of the practical use of SIMULINK for aircraft dynamics and control research. The nonlinear results from this report complete the results from refs.[15] and [16], which covered the linear autopilot design and analysis.

The illustrative character of the 'Beaver Taskforce' research should be acknowledged. The autopilot design, model development in SIMULINK, flight test preparation, and analysis of the flight test results all serve as examples for future flight controls research with the new Citation II laboratory aircraft, which will be configured as *'National Fly-by-wire Testbed'* (NFT). The 'Beaver Taskforce' project has given an unique possibility to scrutinize all aspects of autopilot design and development in practice as a last preparation for future NFT-projects.

The structure of the autopilot and autothrottle will be outlined in chapter 1 of part II. Chapter 2 reviews all control laws and chapter 3 shows how they have been implemented in SIMULINK. Chapter 4 contains the main results of the nonlinear simulations. Finally, chapter 5 will summarize some important lessons from the 'Beaver Taskforce' project which need to be kept in mind for future NFT research. The appendices contain additional information about the Flight Control System model, SIMULINK blocks which are not contained in the standard blocklibraries, and a detailed description of the implementation of the control laws. Appendix D shows how the SIMULINK models can be installed on your own computer.

Chapter 1. Structure of the 'Beaver' autopilot.

1.1 Introduction.

In this chapter, the basic structure of the 'Beaver' autopilot and autothrottle will be outlined. A detailed description of the autopilot structure can be found in refs.[15], and [16]. Ref.[8] describes the linear design of the autothrottle, which was added to the autopilot from ref.[16] later. The autopilot has been based upon 'classical' feedback control theory, as applied in refs.[4], [10], and [11]. The 'Beaver' autopilot serves as an example for research on automatic control concepts for the new Cessna Citation II laboratory aircraft, which will be configured as *National Fly-by-wire Testbed*. A similar 'classical' *baseline autopilot* will be developed for the Citation, against which other control concepts will be measured.

1.2 Basic autopilot functions.

The functions of an autopilot can be divided in two areas: *guidance* and *control*. These functions are defined as follows [11]:

- Guidance: The action of determining the course and speed, relative to some reference system, to be followed by the vehicle.

- Control: The development and application of appropriate forces and moments to the vehicle, which: *(i)* establish some equilibrium state of vehicle motion, and: *(ii)* restore a disturbed vehicle to its equilibrium state (operating point) and/or regulate, within desired limits, its departure from operation point conditions.

The boundary between these two areas is seldom inherently sharp, because of basic functional, operational, and equipment interactions that they may share.

The control loops ensure a fast and stable response of the aircraft to the commands, created by the guidance loops. They must also eliminate the influence of disturbances such as atmospheric turbulence. Basically, the autopilot structure can now be divided in *inner* and *outer loops*. The *control* function is fulfilled by the inner loops (figure 1-1), which control the pitch and roll attitude of the aircraft. These pitch and roll attitude commands are created by the outer loops, which *guide* the aircraft, equipped with the inner loops, along the desired flight path (figure 1-2). The autothrottle will be treated separately in section 1.5, because this system was simply added to the other control laws as an additional outer loop.

A combination of control loops, needed to fulfil a certain control function is called an *autopilot mode*. It is possible to make a distinction between lateral and longitudinal modes. The longitudinal and lateral movements are in practice not totally independent. To prevent the lateral movements from affecting the performance of the longitudinal control loops, it is necessary to include some lateral/longitudinal interconnection. See section 2.4 on turncompensation.

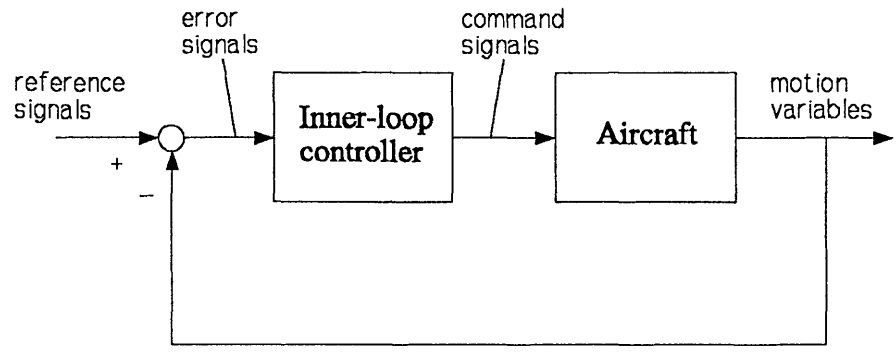


Figure 1-1. Inner loops of the autopilot.

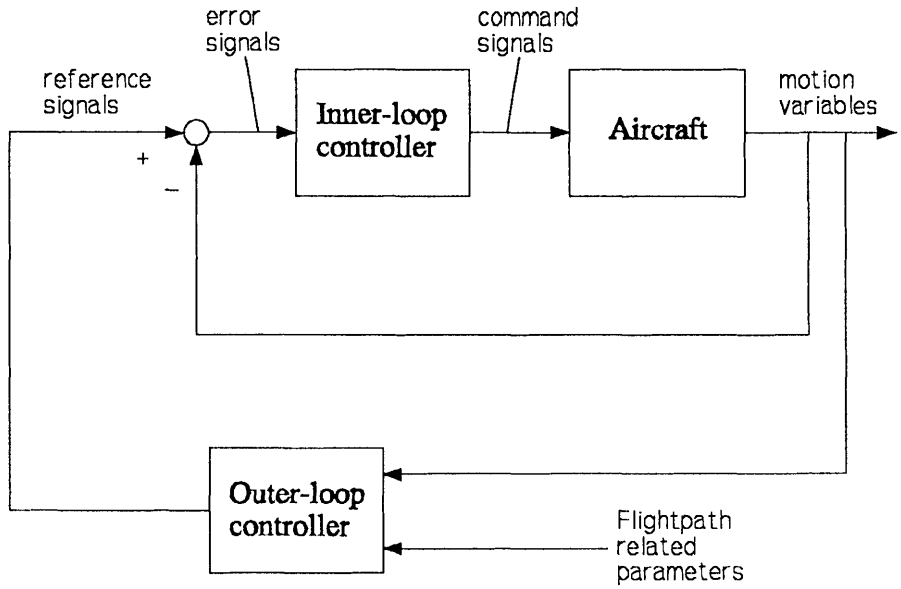


Figure 1-2. Outer loops of the autopilot.

1.3 The DHC-2 'Beaver' aircraft.

The De Havilland DHC-2 'Beaver' laboratory aircraft is shown in figure 1-3. The 'Beaver' is a seven seat, single engine, high-wing, all-metal aircraft. It has a Pratt and Whitney Wasp Jr. R-985 radial engine, which can deliver a maximum power of 450 Hp. Table 1-1 gives some general information about this aircraft.

Since the 'Beaver' is a Short Take Off and Landing (STOL) aircraft, it can fly at very low airspeeds. The flaps are very large and both ailerons can be deflected downwards to give a contribution to the effectiveness of the flaps (this type of ailerons is called 'flaperons'). Due to the large flaps, the aircraft tends to increase its pitch angle considerably if the flaps are deflected.

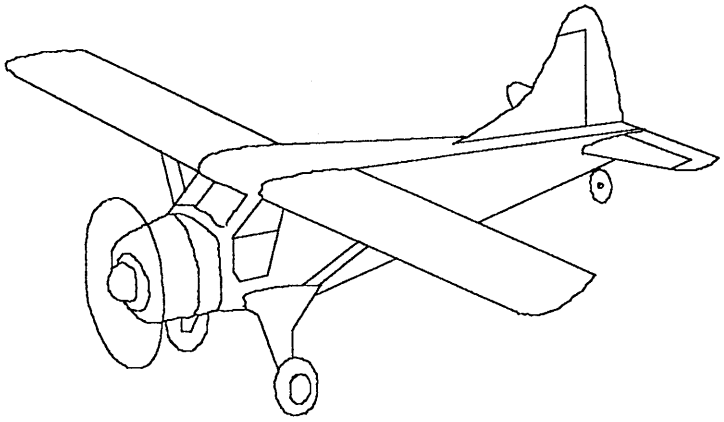


Figure 1-3. The De Havilland DHC-2 'Beaver'.

Manufacturer	The De Havilland Aircraft of Canada Ltd.
Serial no.	1244
Type	Single engine, high-wing, seven seat, all-metal aircraft.
Wing span (b)	14.63 m
Wing area	23.23 m^2
Mean aerodynamic chord ($\bar{c}$)	1.5875 m
Wing sweep	0°
Wing dihedral	1°
Wing profile	NACA 64 A 416
Fuselage length	9.22 m
Max. take-off weight	2315 kgf = 22800 N
Empty weight	1520 kgf = 14970 N
Engine	Pratt and Whitney Wasp Jr. R-985
Max. power	450 Hp at n = 2300 RPM, p_z = 26"Hg
Airscrew	Hamilton Standard, two-bladed metal regulator propeller
Diameter of propeller	2.59 m
Total contents of fuel tanks	521 l
Fuselage front tank	131 l
Fuselage center tank	131 l
Fuselage rear tank	95 l
Wing tiptanks	2 x 82 l
Most forward admissible c.g. position	17.36% $\bar{c}$ at 1725 kgf = 16989 N 29.92% $\bar{c}$ at 2315 kgf = 22800 N
Most backward admissible c.g.position	40.24% $\bar{c}$

Table 1-1. General aircraft data of the DHC-2 'Beaver', PH-VTH.

The large engine causes the aircraft to yaw and slip considerably if the pilot changes the power settings. This effect is particulary strong for the low airspeeds which arise during landings, because of the large angles of attack and the low values of *dpt* (see the expressions for forces and moments from the engines in section 2.2.2 of part I). It is necessary to compensate for yawing and sideslipping with appropriate rudder deflections.

The aircraft is equipped with an autotrim system, which is an extension to the *manual* electric trim system. If the pressure, applied by the elevator control servo to maintain a certain elevator angle, exceeds a certain value during a certain time, the trim angle is automatically adjusted until the servo pressure, and hence, the elevator control force, equals zero again. The autotrim system is necessary for reasons of safety, because the pilot must be able to take over the control of the aircraft if something goes wrong, without having to cope with very large control forces.

According to ref.[9], the automatic trim system degrades the autopilot performance, because it is not certain that the position of the servo remains the same if the autotrim relieves the load, acting on the elevator. Moreover, the control cables will flex when their loads are reduced. The elevator trim and elevator deflection itself will therefore almost certainly deflect slowly upon every change of semi-steady-state flight condition. In this report, the influence of the automatic trim system to the 'Beaver' dynamics will not taken into account, because this effect has never been modelled accurately.

1.4 Elements of the 'Beaver' autopilot.

The autopilot system is built up from the following elements:

- two flight control computers (FCCs) of which one handles the input/output functions, and another contains the control laws;
- a software package which contains the control law algorithms and other control routines;
- an interface between the flight control computers and the pilot and other members of the flight-test crew;
- a system, which transforms command signals from the flight control computers to actual deflections of the control surfaces of the aircraft (in this report, this will be called Flight Control System, or in abbreviated form: FCS);
- sensors to measure the physical quantities, needed within the control laws.

These elements will briefly be described below.

1.4.1 The flight control computers.

The 'Beaver' autopilot uses two different computers:

1 - The ROLM 1603 computer is a 16-bit digital computer, conceived to be used in the harsh environmental conditions, which can arise in the avionics-bay of an aircraft. It is equipped with some I/O cards to handle the data input and output to and from the primary flight control computer, the

Compaq PC. The input signals are digitized and send to the Compaq PC, the output voltage controls the actuators, i.e., the FCS system.

2 - A portable Compaq personal computer, which contains the control laws and the mode controller (see section 1.3.2). The Compaq uses the input signals which it receives from the ROLM to determine the control surface commands, which are sent to the FCS via the ROLM.

The calculations which are necessary to determine the control surface deflections take a certain time. This time-interval is called *computational delay*. The input variables are measured at regular spaced time-intervals. The time between two measurements is called the *sampling time*. It is necessary that the sampling time is at least as big as the computational delay, to make sure that all calculations are done properly.

The previous 'Beaver' autopilot (ref.[9]) used the ROLM computer as primary FCC, in which all calculations to evaluate the control laws were performed. Due to the limitations of the ROLM, the minimum sampling time was 0.1 seconds, which, according to ref.[15] was too low to obtain satisfying results, especially regarding the damping of the fastest characteristic motions, that is: the short-period oscillation, and the Dutch-roll mode. For this reason, the new autopilot uses the ROLM for I/O functions only; the actual calculations are now done within the Compaq portable PC. In this way, the computational delay (and hence, the maximum allowable sampling time) has been reduced to 0.03 seconds, which is so fast that the differences between continuous or discrete versions of the control laws can be neglected (ref.[16]).

Using a PC as primary FCC also has the advantage that a high-level programming language (C) could be used to implement the control law and mode controller software. The relations between the inputsignals, outputsignals, the ROLM, and the Compaq are shown in figure 1-4.

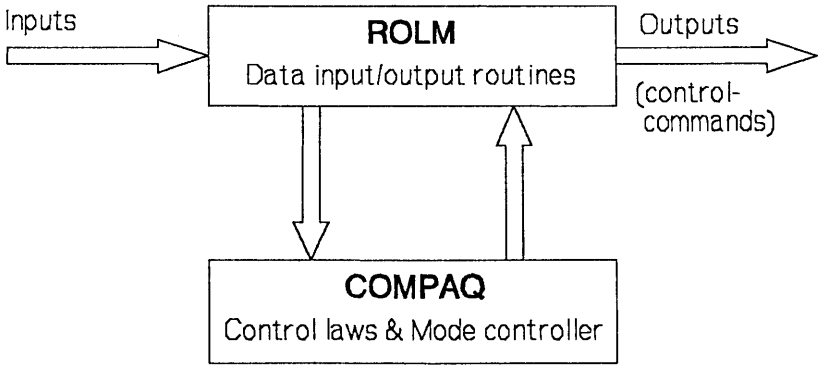


Figure 1-4. Relations between the inputs, outputs, and the two Flight Control Computers.

1.4.2 The software.

The ROLM software contains I/O routines for sending data from the sensors to the Compaq and from the Compaq to the FCS. The ROLM must be pro-

grammed in assembler, because high-level programming languages would require too much memory, and would be too slow for this computer.

The Compaq programs contain the control laws, mode controller routines and I/O routines for communication with the ROLM. These programs have been written in C, because this proved to be the fastest high-level programming language available [16]. The *software*, containing the control laws and mode controller have been described in more detail in ref.[16].

The autopilot uses five longitudinal and five lateral control modes. The longitudinal control laws control the elevator; the lateral control laws control the ailerons and the rudder:

Longitudinal autopilot modes.

- Pitch Attitude Hold (PAH). This mode controls the pitch angle θ. It is used as inner-loop for other longitudinal autopilot modes, after addition of a wash-out filter in the θ-loop. See section 2.2.1.

- Altitude Hold (ALH). This mode maintains a certain reference altitude. See section 2.2.2.

- Altitude Select (ALS). This control law is used if the difference between the reference altitude which is selected by the pilot and the altitude where the aircraft flies at that moment is too large. It actually controls the rate of climb. See section 2.2.3.

- Approach Glideslope (GS). The longitudinal part of the approach mode makes sure that the aircraft starts following the reference glide path to the runway during approach. See section 2.2.4 and appendix C of part I.

- Go Around (GA). This mode has a longitudinal part, as well as a lateral part. The longitudinal part of the GA mode controls the aircraft to maximum pitch angle if an approach is cancelled. See section 2.2.5.

Lateral autopilot modes.

- Roll Attitude Hold (RAH). This mode controls the roll angle φ. It is used as an inner loop for other lateral modes. See section 2.3.1.

- Heading Hold / Heading Select (HH). This mode is used to select or maintain a certain reference heading. See section 2.3.2.

- Approach Localizer (LOC). The lateral part of the approach mode makes sure that the aircraft lines up with the runway. See section 2.3.3.

- Navigation (NAV). This mode is lets the aircraft fly along a VOR bearing. See section 2.3.4.

- Go Around (GA). The lateral part of this mode functions as wing-leveller during abandoned approaches. See section 2.3.5.

The control laws have been programmed as separate subroutines in C. These subroutines are called by the mode controller program. If the autopilot is switched on, a large number of checks are executed, to make sure that everything functions properly. Then the reference values for the pitch and roll angles are measured; the reference values of other variables are read too. The mode controller also switches on appropriate lights on the annunciator panel. In the Altitude Select and Approach modes, switch criteria are used to decide whether another control law must be selected. For instance, in Approach mode, there are two different phases. See sections 2.2.4 and 2.3.3.

1.4.3 The interface between the FCCs and the pilot.

The pilot can select an autopilot mode with the mode selection panel, displayed in figure 1-5. This panel has been taken over from the King KFC-300 autopilot. The Go Around button is located on the throttle lever and the reference heading can be selected with the Omni Bearing Selection bug on the heading indicator.

The autopilot software is started by the observer, who also operates the Data Logging System [1]. As soon as the autopilot has been started, the two basic modes (Pitch Attitude Hold and Roll Attitude Hold) are activated. The attitude of the aircraft at that moment is used as reference attitude. The other autopilot modes can then be selected by pushing other buttons on the mode selection panel. Which mode is running can be seen on the annunciator panel (figure 1-6).

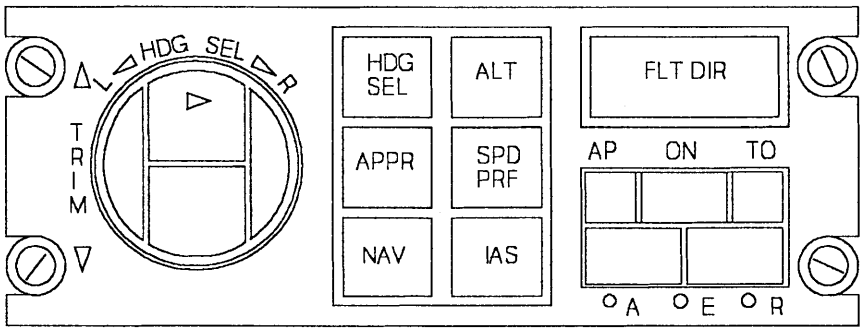


Figure 1-5. Mode selection panel.

Figure 1-6. Annunciator panel.

1) The Data Logging System (DLS) records important flight data during flight tests, which can be used to analyze the performance of the autopilot.

1.4.4 The sensors.

A number of different sensors are used to measure the aircraft's attitude, angular velocities, navigation signals, and airdata quantities. The measured quantities are transformed to digital signals and sent to the ROLM via its I/O cards. The following sensors are used:

- vertical gyro, which measures the pitch attitude θ and the bank angle Φ;
- directional gyro, which measures the yaw angle ψ;
- airdata computer, which calculates the airspeed V, the aircraft's altitude H and the rate of climb C;
- angular velocity gyro, which measures the roll rate p, pitch rate q, and the yaw rate r;
- radio equipment, which receives navigation signals from ILS and VOR transmitters.

The angular velocities are measured with 'strapdown' gyro's, which are fixed to the aircraft, contrary to gyro's which are fixed to a stable platform. See ref.[3].

1.4.5 The Flight Control System (FCS).

Using the inputsignals from the sensors, the control surface deflections are calculated by the Compaq FCC. The outputsignals from this PC are sent to the ROLM, where they are transferred to analog electrical signals. The electronic, hydraulic, and mechanic systems which are used to transfer the currents from the ROLM to actual deflections of the control surfaces will be called *'Flight Control System'* (FCS) in this report.

A full description of the FCS can be found in ref.[5]. A short summary is included in appendix A. Although the FCS model is a dynamical system in itself, it has not been described in part I, because it actually forms a part of the total Automatic Aircraft Control System; in theory it is possible to change the dynamics of this part of the control system by using other servo's, cables, etc. Appendix A also summarizes the possible effects of the FCS system upon the autopilot performance.

1.5 The autothrottle.

The autopilot only functions properly if the pilot takes care of maintaining a more or less constant speed during manoeuvres. As a part of the 'Beaver Task-force' project an autothrottle was added to the existing control structure of the other autopilot modes [8], with different control laws for Pitch Attitude Hold, Altitude Hold, and Altitude Select. This system will not be treated here, but the SIMULINK simulation model has been equipped with input ports which make it possible to include an autothrottle. Both the engine speed n and the manifold pressure p_z may be controlled by the autothrottle, although the auto-throttle from ref.[8] only controlled the manifold pressure. It is planned to design a TECS (Total Energy Control System) - autothrottle/Altitude Hold system too, because such a system will probably have less problems with speed-altitude interference. Such a system can be included to the SIMULINK simula-tion structure in the same way as the other control laws.

1.6 Conclusions.

The autopilot functions can be separated in guidance and control, which, quite naturally leads to a division in inner and outer loops if the control structure is designed with 'classical' control theory. The autopilot system is built up from a number of different hardware and software components. In the remainder of this report, it will be assumed that the hardware can't be altered by the user, which limits the discussion to the development and analysis of the *control laws*. The implementation of the control laws in the SIMULINK simulation structure can be interpreted as an extensive demonstration of the capabilities of the models from part I. In that sense, this analysis may serve as an example for the implementation of other control laws, such as a TECS autothrottle/altitude hold controller, in the future.

Chapter 2. The control laws.

2.1 Introduction.

In this chapter, the structure of all control laws, excluding the autothrottle, will be described in detail. In ref.[16], the linear design of the control laws is described, including the determination of the gains and filter coefficients. Here, the resulting control laws will be analyzed with nonlinear simulations. Section 2.2 gives a review of all longitudinal autopilot modes; section 2.3 treats the lateral modes. The turncompensation loops for suppressing lateral-longitudinal cross-coupling effects will be discussed in section 2.4. Finally, section 2.5 shows which signals need to be limited to make sure that the autopilot fulfils the requirements and always operates safely.

2.2 Longitudinal autopilot modes.

2.2.1 Pitch Attitude Hold mode (PAH).

The Pitch Attitude Hold mode is the basic longitudinal autopilot mode; it controls the pitch angle θ by applying appropriate deflections of the elevator, δ_e, if the value of θ differs from the desired reference value. Normally, the PAH mode serves as inner loop for the Altitude Hold, Altitude Select, and longitudinal Glideslope modes (after adding a filter in the θ-loop). It is also possible to select the PAH mode separately, for instance to control the aircraft with a side-stick (fly-by-wire control). Also, the longitudinal part of the Go-Around mode is a special case of the PAH mode, see section 2.2.5.

The pitch angle θ is fed back to damp the phugoid mode of the 'Beaver', and to ensure that the desired pitch angle is maintained. A proportional and integrating controller is used, to make sure that no steady-state errors $\Delta\theta$ will remain. As long as the error $\theta - \theta_{ref}$ is not equal to zero, the signal from the integrator will increase, which leads to an increasing elevator deflection. The θ-feedback somewhat decreases the damping of the short period mode, which is compensated by adding a feedback loop of the pitch rate q to the elevator, see refs.[4], [10], or [11].

A block-diagram of the PAH mode is shown in figure 2-1. An additional loop for turncompensation will be added later, see section 2.4. Also, some signals will be limited. All gains are functions of the airspeed V, see table 2-1 at the end of this chapter.

2.2.2 Altitude Hold mode (ALH).

The Altitude Hold mode is used to maintain a reference altitude, which is entered by the pilot. This mode uses the PAH mode with an additional washout filter in the θ-loop as inner-loop, hence, it fulfils a basic guidance function. If the reference altitude differs too much from the actual altitude, the *mode controller* automatically switches to Altitude Select mode, see section 2.2.3.

The difference between the reference altitude and the actual altitude, $\Delta H = H_{ref} - H$, is fed back via an amplifier to the inner loops, hence, the outer

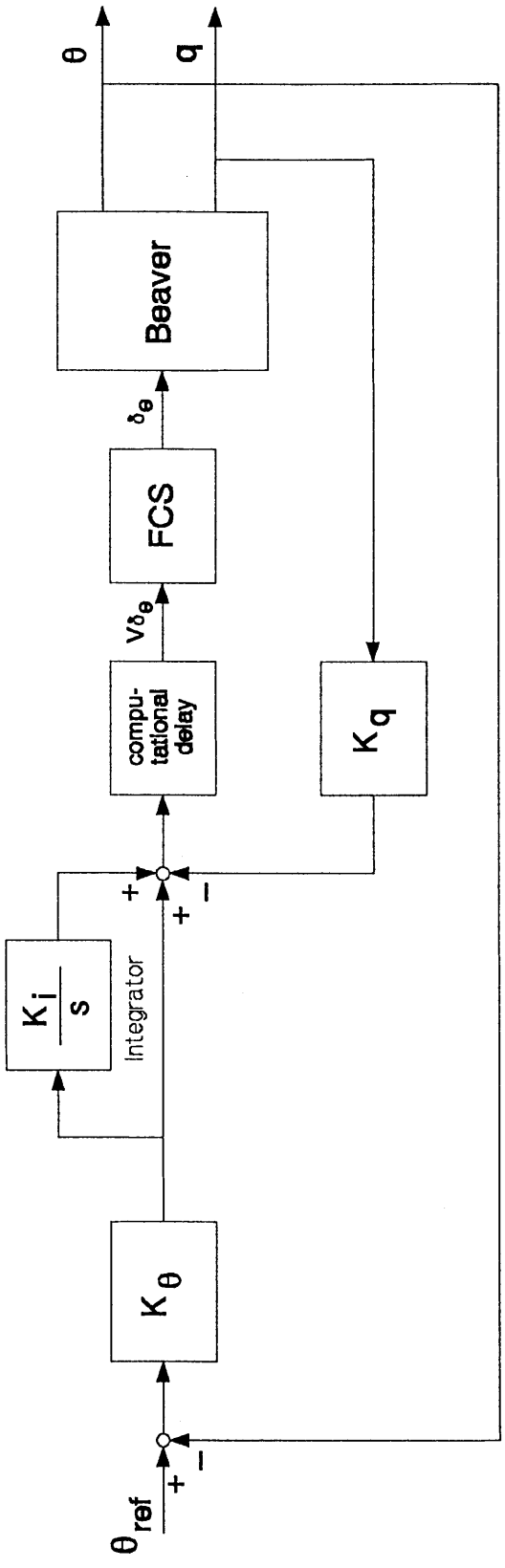


Figure 2-1. Block-diagram of the Pitch Attitude Hold mode (without turncompensation).

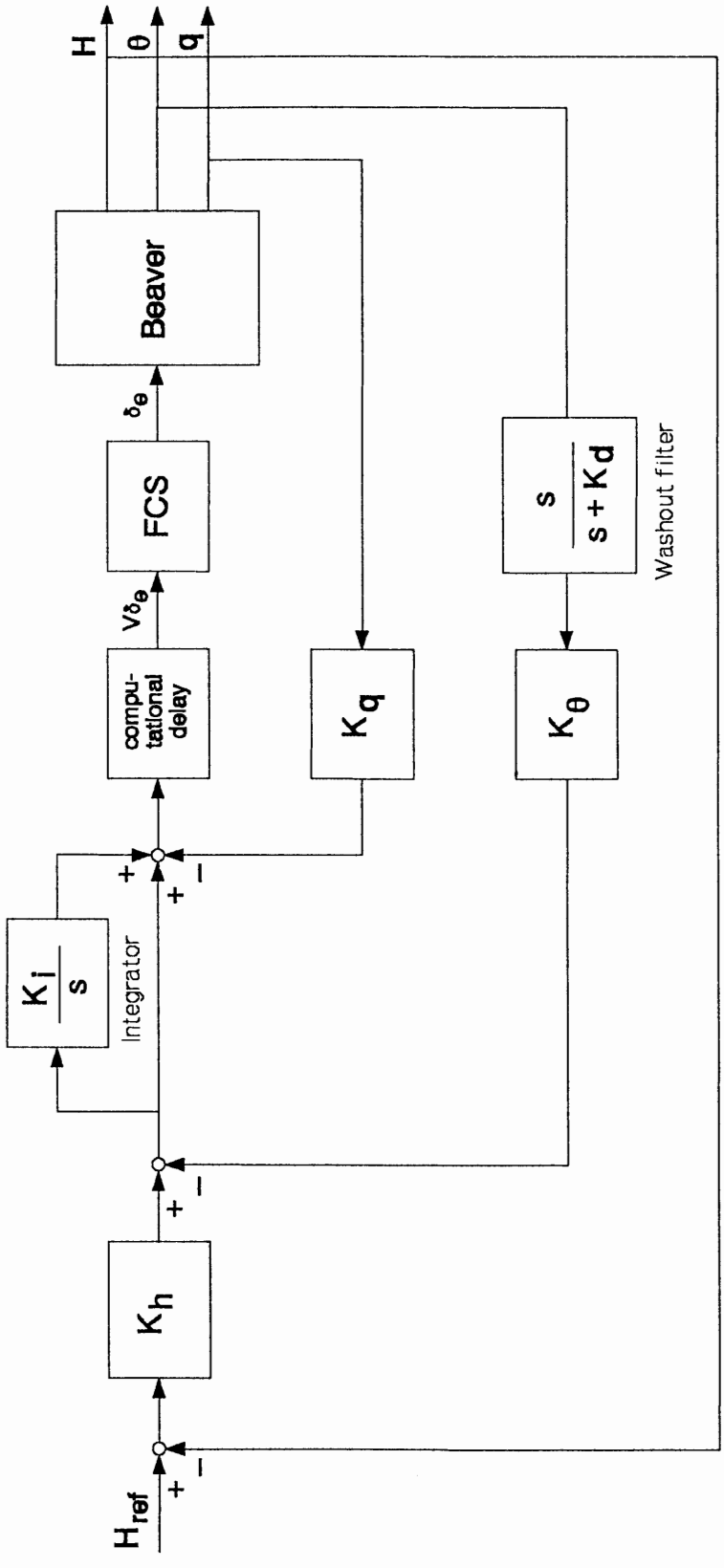


Figure 2-2. Block-diagram of the Altitude Hold mode (without turncompensation).

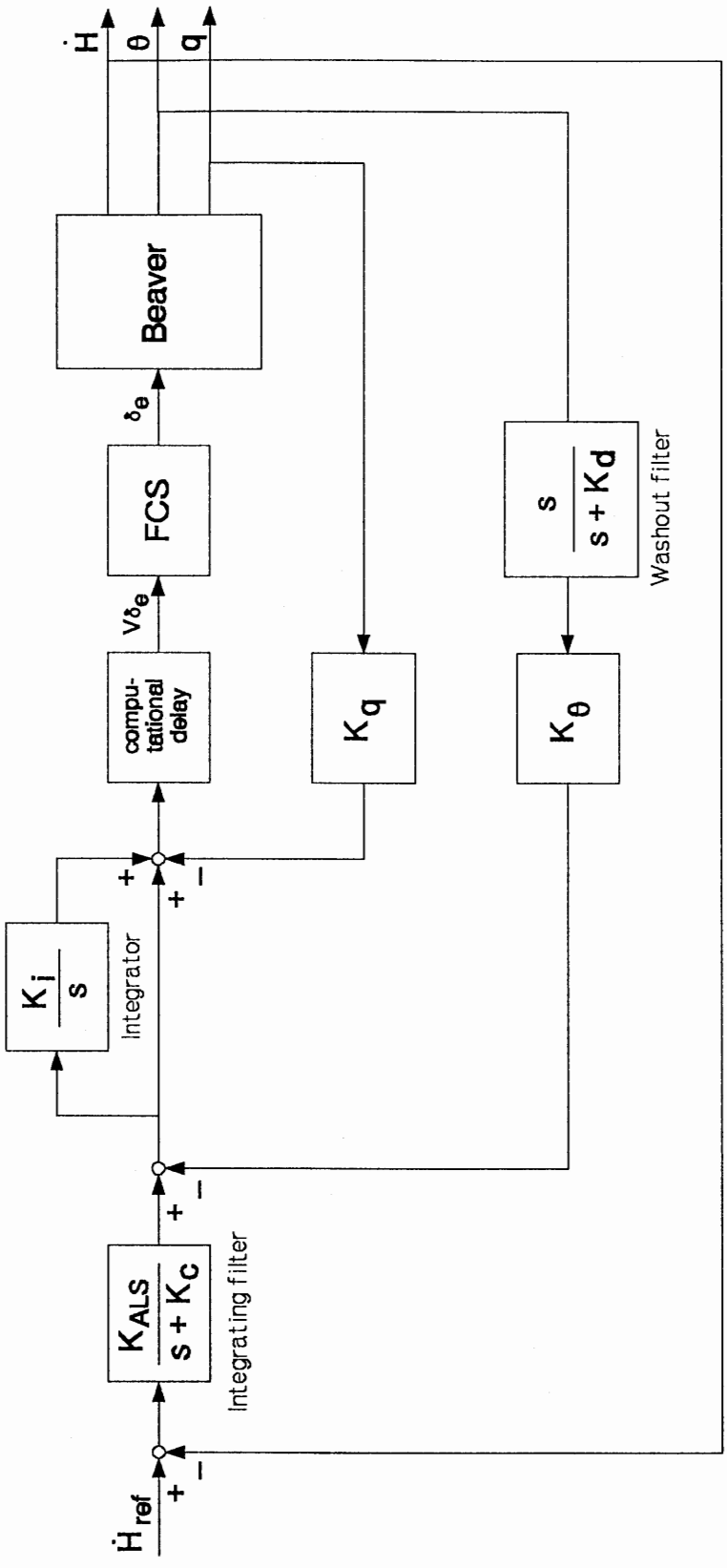


Figure 2-3. Block-diagram of the Altitude Select mode.

loop generates a pitch-command signal θ_{ref} for the inner loops. The washout filter in the θ-loop is necessary, because as soon as H_{ref} is reached the command signal θ_{ref} will become zero, whereas the actual pitch angle usually differs from zero. The signal which leaves the washout filter will be very small if the pitch angle changes very slowly, or doesn't change at all, which ensures that the inner-loop will not tend to maintain a pitch angle $\theta = 0$ if $\Delta H = 0$.

A block-diagram of the ALH mode is shown in figure 2-2. An additional loop for turncompensation will be added later. The gains depend upon V, see table 2-1 at the end of this chapter.

2.2.3 Altitude Select mode (ALS).

The Altitude Select mode actually controls the rate of climb of the aircraft. The climbrate $\dot{H}$ is fed back via a filter to the pitch channel. The PAH mode, equipped with an additional washout filter in the θ-loop serves as inner-loop. The mode controller of the 'Beaver autopilot' automatically decides which rate of climb is to be maintained if the pilot enters a certain desired reference altitude. If the difference between the reference altitude and the actual altitude exceeds a certain value, the ALS mode will be engaged. If the aircraft enters an altitude 'window' around the desired value, the mode controller automatically switches from ALS to ALH mode.

It is essential that the pilot keeps a constant velocity by increasing the engine power during climbs, because otherwise the reference value of the climbrate cannot be reached. If the autopilot is used together with the autothrottle, the system should change the engine settings automatically.

A blockdiagram of the ALS mode is given in figure 2-3. The gains depend upon V, see table 2-1 at the end of this chapter.

2.2.4 Longitudinal part of the Approach mode: Glideslope (GS).

In approach mode, the 'Beaver' must be guided along the reference planes of glideslope and localizer. These reference planes are provided by ILS (Instrument Landing System) radio signals, which can be detected in the aircraft [3]. The Glideslope mode is the longitudinal part of the approach mode, which brings the aircraft from level flight into a descent, following the glideslope reference plane. The glideslope signal is emitted by an antenna, located at some 300 metres beyond the runway threshold. The angle between the glideslope reference plane and the horizontal equals some value between 2 and 4 degrees (refs.[3], [6], and [1]). See appendix C of part I for a description of the ILS system.

The glideslope mode uses a feedback of the error angle between the line passing through the aircraft and the glideslope antenna, ε_{gs}, and the glideslope reference plane (see figure 2-4). The rate of change of this error angle is approximated with a differentiating filter $\frac{s}{s+1}$, which is engaged as soon as the autopilot is turned on, to make sure that the transients during the first couple of seconds after engaging the filter do not affect the glideslope performance. It has a very small phase-lag, so it gives a good approximation of the

actual derivative. Both ε_{gs} and the distance from aircraft to glideslope reference plane d_{gs} are positive if the aircraft flies above the glideslope.

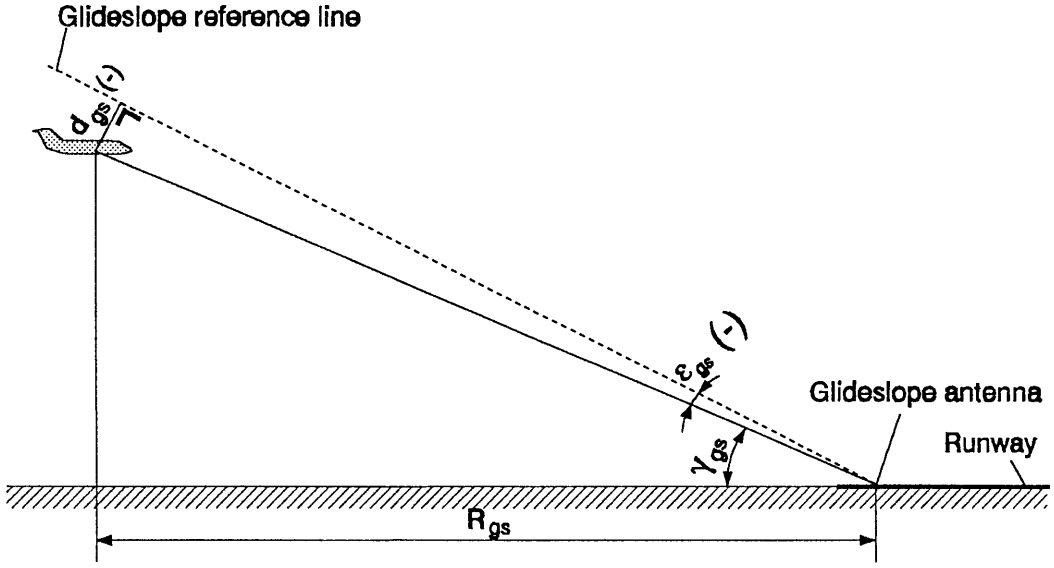


Figure 2-4. Definition of the error angle ε_{gs}, and the distances d_{gs} and R_{gs}.

The longitudinal part of the approach can be separated in two different phases:

- Glideslope Armed. This phase is engaged as soon as the approach mode is selected by the pilot. The longitudinal autopilot mode in which the aircraft was flying before selecting the approach mode, usually ALH, will be maintained until the aircraft has reached the glideslope reference plane.

- Glideslope Coupled. This phase is initiated if the aircraft passes the glideslope reference plane for the first time. In this phase, the error angle ε_{gs} is fed back, together with its approximated rate of change $\dot{\varepsilon}_{gs}$.

In the previous version of the 'Beaver' autopilot, the Glideslope Coupled phase was subdivided in a *Captured* phase, used to intercept the glideslope smoothly, and a *Tracked* phase, used if the aircraft had intercepted the glideslope [9]. However, in practice the autopilot often switched from Armed to Captured rightaway, which resulted in the undesirable response sketched in figure 2-5. Furthermore, undesirable transient effects could occur if the autopilot switched from Armed to Captured or Tracked [15].

In the current autopilot, the Glideslope Captured and Glideslope Tracked modes have been merged into one Glideslope Coupled mode. The autopilot now switches from Armed to Coupled if the aircraft is actually passing through the glideslope plane, to make sure that the intercept will not look like figure 2-5. The resulting overshoot is made as small as possible by using a feedback of both the error angle and its derivative [16].

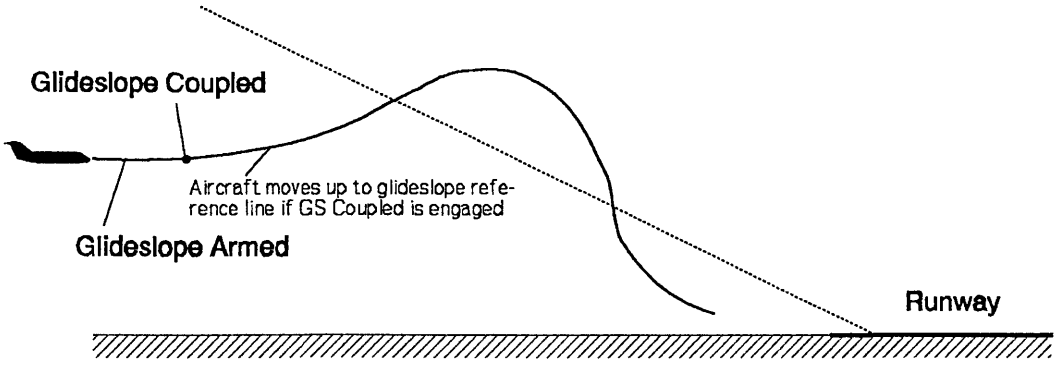


Figure 2-5. Response of the aircraft if the Glideslope Coupled mode is engaged too soon with a feedback of ε_{gs} only.

If the gain $K_{\varepsilon_{gs}}$ is kept constant, the sensitivity of the Glideslope Coupled mode will increase if the aircraft approaches the runway, i.e., if R_{gs} decreases. This is caused by the fact that if the distance from the glideslope reference plane to the aircraft d_{gs} (see figure 2-4) is constant, the error ε_{gs} is larger if the aircraft flies closer to the runway. In practice, increasing the gain $K_{\varepsilon_{gs}}$ would yield a similar effect. If this gain is kept constant, the system would become instable if R_{gs} comes below a certain limit value.

This problem can easily be solved if DME information is available, but this was not possible for the 'Beaver' aircraft, due to hardware limitations. For this reason, $K_{\varepsilon_{gs}}$ has been scheduled with the factor $\frac{H_f}{\sin|\gamma_{gs}|}$ $\left(=\sqrt{H_f^2+R_{gs}^2}\right)$, where H_f is the altitude above the runway, and γ_{gs} is the flight path angle if the aircraft flies along the nominal flight path. This factor equals the three dimensional distance from the aircraft to the glideslope antenna (R_{gs} is the ground distance from aircraft to glideslope antenna).

This gain scheduling can only be applied if the altitude of the runway above sea level and the nominal flight path angle γ_{gs} are known. For our research purposes, the software of the 'Beaver' autopilot has been equipped with information of Rotterdam Airport 'Zestienhoven', but if this autopilot would have been developed further into a commercial system, it would have been necessary to include a database with relevant information about a large number of airports. In other words: a simple flight-management system would be needed.

The mode-controller constantly checks the magnitude of the error signal and decides which control law is to be used. A block-diagram of the GS Coupled mode is presented in figure 2-6. The gains depend upon V, see table 2-1 at the end of this section. A description of the lateral approach mode is given in section 2.3.3. See appendix C of part I for more details about the glideslope signals. The Glideslope Coupled control law only works correctly if the pilot reduces the engine power manually after the intercept has been made. If the autothrottle is used, this should be done automatically.

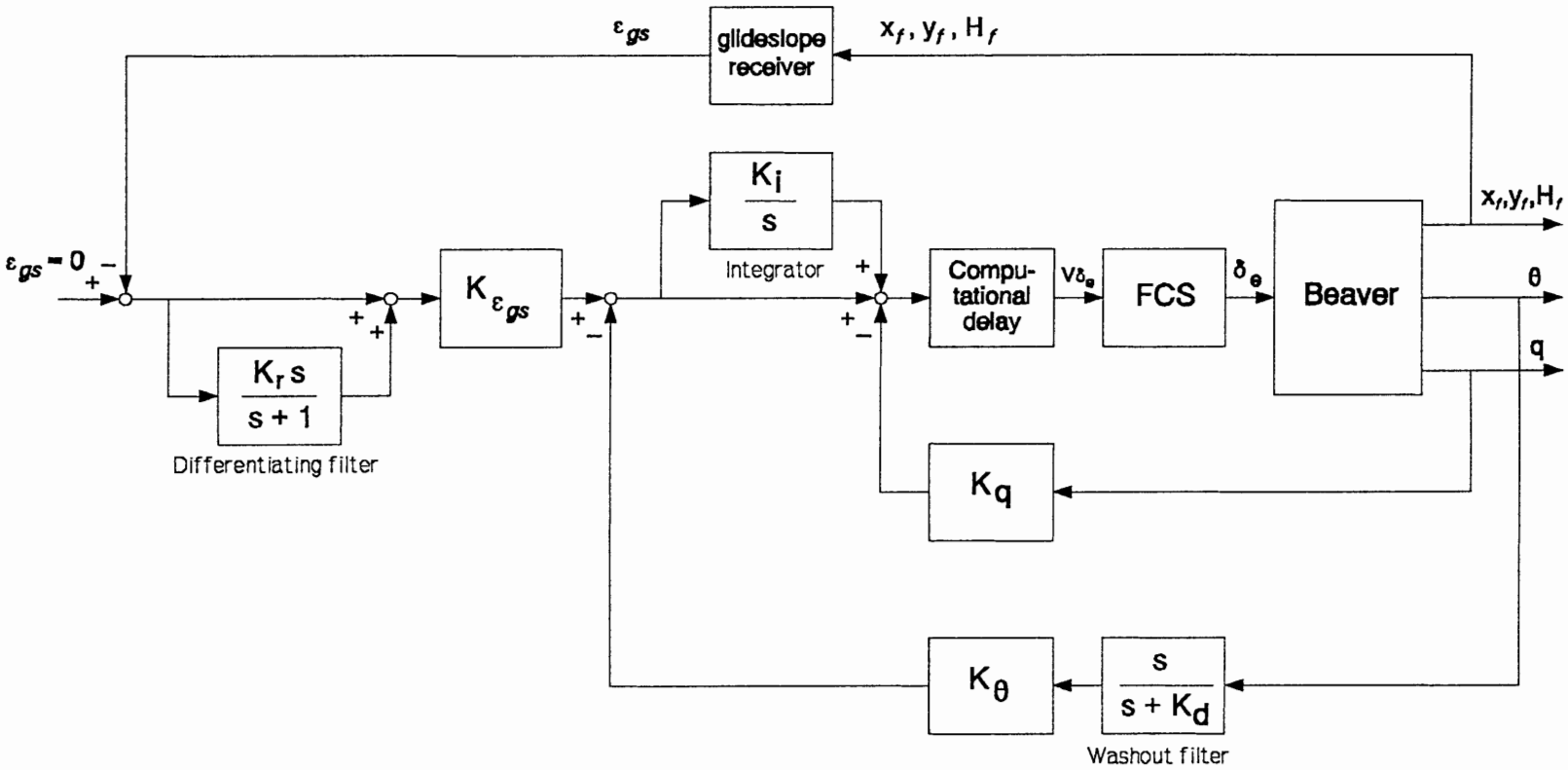


Figure 2-6. Block-diagram of Approach Glideslope mode.

2.2.5 Longitudinal part of the Go Around mode (GA).

The Go Around mode is used to cancel an approach. It has both longitudinal and lateral loops. The longitudinal part of the GA mode is a special case of the Pitch Attitude Hold mode. It uses the control logic of the PAH mode, with a stepwise reference pitch-command of 10°. See section 2.3.5 for a description of the lateral part of the GA mode.The airspeed V should remain constant, which means that the pilot must apply full engine power. If an autothrottle is used, this should be done automatically. Notice that the Go Around mode is potentially dangerous if the pilot does *not* apply full throttle!

2.3 Lateral autopilot modes.

2.3.1 Roll Attitude Hold mode (RAH) with turn coordinator.

The Roll Attitude Hold mode is the basic lateral autopilot mode. It's main purpose it to serve as inner-loop for the other lateral autopilot modes, but it is also possible to use the RAH mode separately, for instance for fly-by-wire control via a side-stick. The deviation of the actual roll attitude from the desired roll attitude is fed back to the ailerons via a proportional, integrating controller, to ensure that the desired roll angle is reached.

The previous autopilot used a separate yaw damper to eliminate the sideslip angle during turns and to damp the Dutch-roll [9]. Unfortunately, this mode didn't work properly, according to refs. [15] and [16]. In the current autopilot design, the yaw damper has been replaced by a turncoordinator. Since the Dutch-roll of the 'Beaver' is damped quite well, the 'Beaver' does not need a yaw-damper anyway (see section 5.3 of part I). In figure 2-7, the block-diagram of the RAH mode, including the turncoordination loops, is shown. Nonlinear simulations have shown that the integrator in the control law for the RAH mode was necessary in order to prevent steady-state errors in φ. Linear analysis did not reveil this problem, because the longitudinal-lateral cross-coupling of the 'Beaver' dynamics was not contained in the linear models. See ref.[13].

The turncoordinator has two functions: it must *(i)* suppress the sideslip angle in turns with appropriate aileron and rudder deflections, and *(ii)* suppress the adverse yaw, which occurs when a turn is initiated by deflecting the ailerons [1)].

1 - In a coordinated turn, it is necessary to apply both aileron and rudder deflections, which must have opposite signs [7]. The deflections δ_a and δ_r depend on the true airspeed V and the yaw-rate r or the roll angle φ [16]. Using the linear aircraft models from ref.[14], it was possible to express

---

1) 'Adverse yaw' arises from the difference in drag due to the down-aileron compared to the drag of the up-aileron. The sign of $C_{n_{\delta_a}}$ depends mainly on the rigging of the ailerons, their profile drag characteristics, and the angle of attack of the airframe. Aileron deflections can also produce side forces on the vertical tail which can become important contributors to $C_{n_{\delta_a}}$ (ref.[11]). If negative, $C_{n_{\delta_a}}$ is called 'adverse yaw' because it causes the aircraft to yaw initially in a direction opposite to that desired by the pilot; if positive, $C_{n_{\delta_a}}$ produces 'proverse yaw'. For the 'Beaver', $C_{n_{\delta_a}}$ is negative.

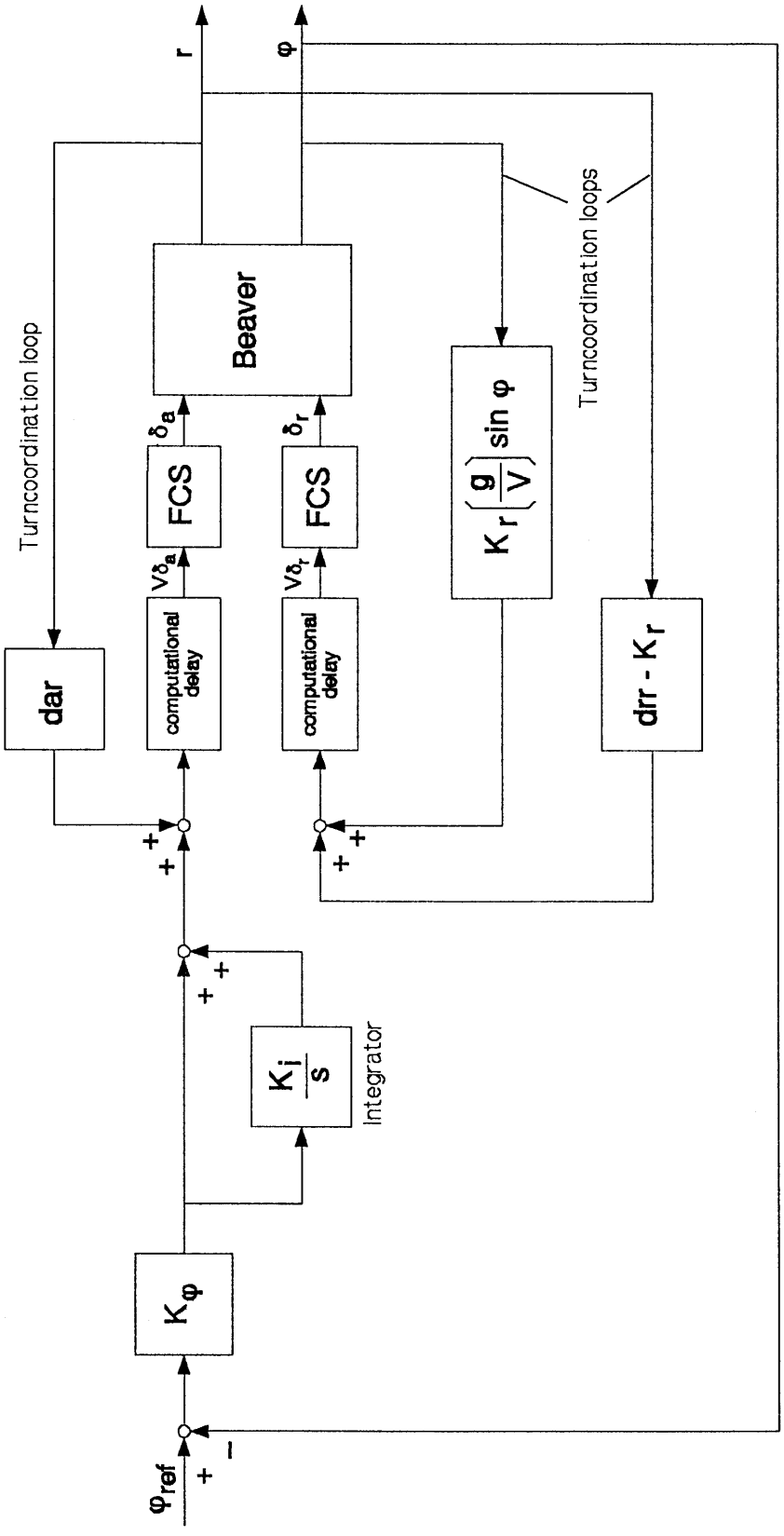


Figure 2-7. Block-diagram of the Roll Attitude Hold mode (with turncoordinator).

these deflections as functions of r for a number of different velocities. The sideslip suppression function of the turncoordinator is included in the blockdiagram by means of the factors *dar* and *drr*, which are functions of V (see table 2-1 at the end of this chapter for the definitions of the gain-scheduling functions for *dar* and *drr*).

2 - The suppression of the adverse yawing due to aileron deflection is based upon the following relation, which is valid only in coordinated turns:

$$r = \frac{g}{V} \sin\varphi \tag{2-1}$$

If the yaw rate is too small, that is if: $\frac{g}{V} \sin\varphi > r$, it is necessary to apply a larger deflection of the rudder. If $\frac{g}{V} \sin\varphi < r$, the aircraft is yawing too fast, hence, a smaller rudder deflection is required. For this reason, the factor $\frac{g}{V} \sin\varphi$ is fed back to the rudder channel via the gain K_r.

The vertical gyro actually measures the bank angle Φ in stead of the roll angle φ. The relation between these angles is defined as follows:

$$\varphi = \frac{\Phi}{\cos(\theta - \alpha_0)} \tag{2-2}$$

If the angle α_0 is known, it is possible to make the appropriate correction, but in the 'Beaver' autopilot, the angle α_0 is *not* measured. Although it is possible to estimate α_0 by using the values given in ref.[14], or computed with the aircraft trim routine *ACTRIM* (see section F.2.7 of part I), and then applying equation (2-2), this has not been done for the following reasons [16]:

- The roll angle is a function of the bank angle Φ, *and* the pitch angle θ, hence, if the correction (2-2) would be applied, the autopilot would change the bank angle if the pitch angle is changed. Thus, an unwanted coupling between longitudinal and lateral control actions would be introduced if the correction (2-2) would be used.

- The cockpit instruments also show the value that is actually measured, that is: the bank angle Φ. If the roll angle φ would be used in RAH mode, there would be a difference between the readings of the cockpit instruments and the signal used in the control law, which is not desirable from a pilot's point of view.

- The resulting error is not very large. The largest error occurs at very low airspeeds (large α_0's), but the error will never be larger than a few percents. Also, using the approximation of α_0 from ref.[14] also would give a small error, because in real flight the aircraft's weight, centre of gravity, etc. will probably differ from the values used in ref.[14].

The gains in figure 2-7 depend upon the true airspeed V. See table 2-1 at the end of this chapter for a list of all gains.

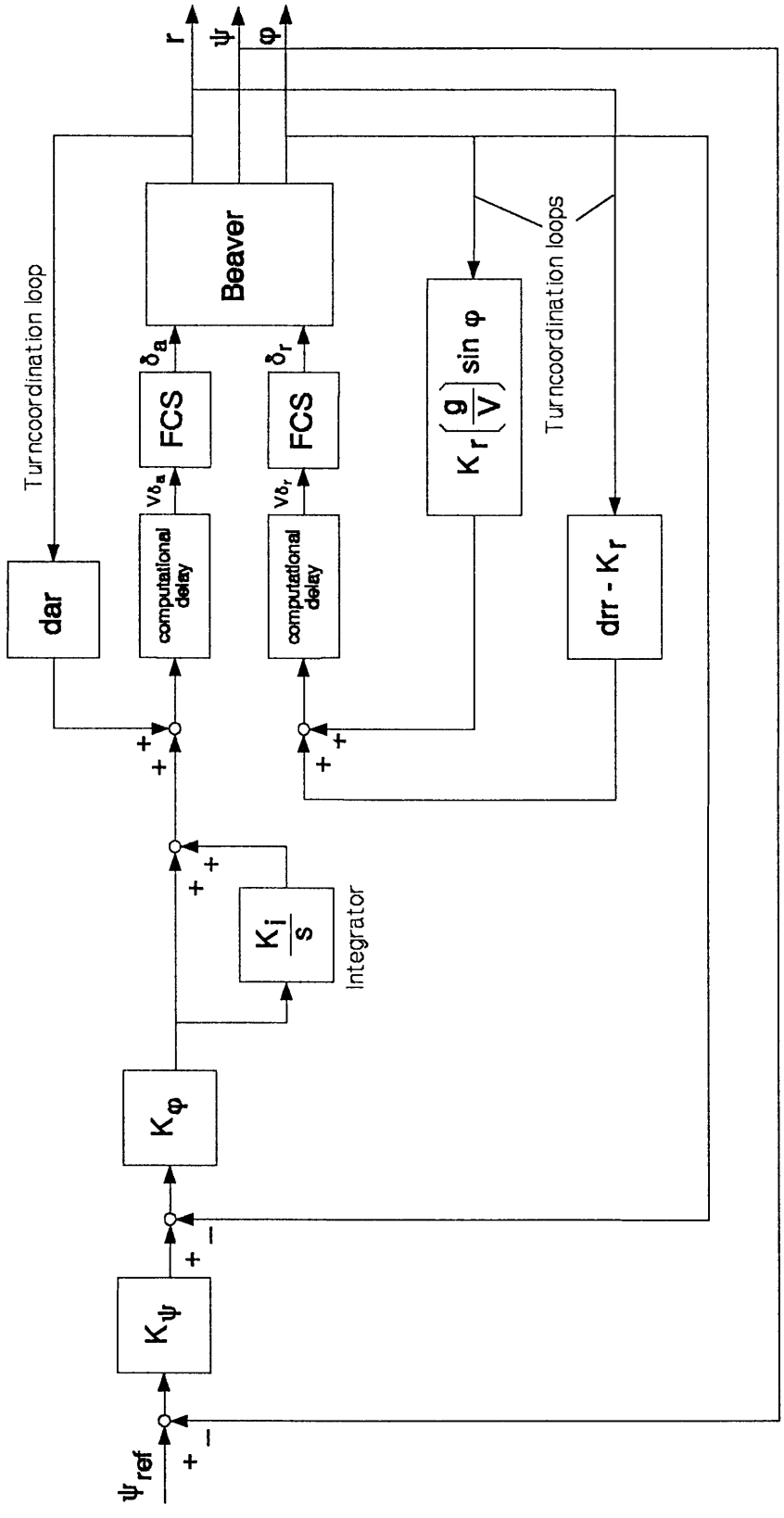


Figure 2-8. Block-diagram of the Heading Hold / Heading Select mode (with turncoordinator).

2.3.2 Heading Hold / Heading Select mode (HH).

The Heading Hold / Heading Select mode is used to maintain or select a certain heading of the vehicle [1)]. A reference value of the roll angle φ is computed by using a feedback of the yaw-angle ψ (in practice, the pilot will want to select an azimuth angle χ, but the sideslip angle β should be 0 degrees, so: $\chi \approx \psi$). This reference value of φ is equal to the difference between the reference heading and the actual heading $\Delta\psi = \psi_{ref} - \psi$, amplified with a certain gain factor. The control laws of the Roll Attitude Hold mode are used as inner loops.

Figure 2-8 shows the resulting block-diagram for the HH mode. See table 2-1 at the end of this chapter for the definitions of the gains, which are functions of the airspeed V.

2.3.3 Lateral part of the Approach mode: Localizer (LOC).

In approach mode, the 'Beaver' autopilot uses ILS signals to line up along the runway centerline, and to follow the glideslope reference line. The latter function has been described in section 2.2.4. To line up in front of the runway, the localizer signal is used. See appendix C of part I, and refs.[3] and [1] for more details about the ILS system. Figure 2-9 shows the most important definitions. Γ_{loc} and d_{loc} are positive if the aircraft is flying at the right-hand side of the localizer reference plane, heading towards the runway.

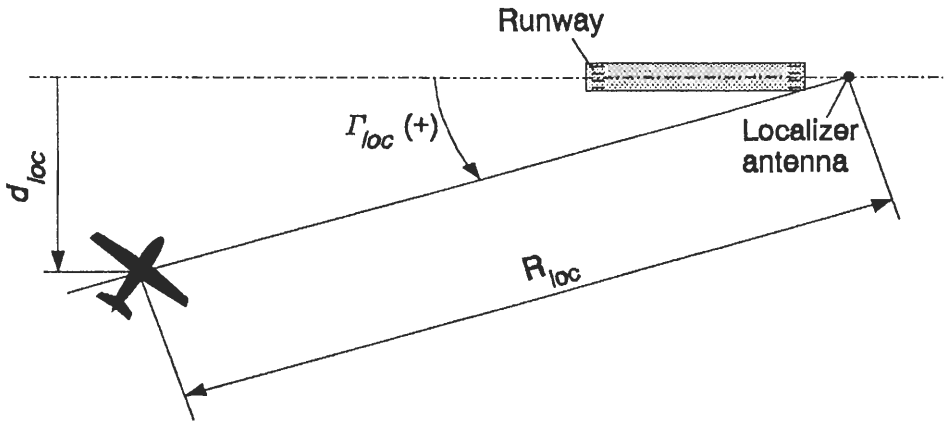


Figure 2-9. Definition of error angle Γ_{loc} and the distances d_{loc} and R_{loc}.

---

1) If the aircraft flies in NAV mode and enters the 'cone of silence' of the VOR-transmitter (see appendix C of part I or ref.[3]), it will maintain its heading by switching to *Heading Hold* mode. If the pilot enters a certain reference heading, the *Heading Select* mode will be switched on. Despite the difference in names, Heading Hold and Heading Select use exactly the same control laws. See section 2.3.4 for a description of the NAV mode.

The pilot must select the approach mode with the mode selection panel, and enter the heading of the runway with the Omni Bearing Selector bug on the heading indicator. Again, there are two phases:

- Localizer Armed. The control laws for this phase are engaged as soon as the approach mode is selected by the pilot. The lateral autopilot mode in which the aircraft was flying before the approach mode was selected, often HH, will be maintained until the aircraft has come near enough to the localizer reference plane. It depends upon $\dot{\Gamma}_{loc}$ how soon the Localizer Coupled phase will be entered.

- Localizer Coupled. This phase is entered if Γ_{loc} and $\dot{\Gamma}_{loc}$ satisfy the following criteria:

$$K_V \cdot \dot{\Gamma}_{loc} + \Gamma_{loc} > 0 \quad \wedge \quad \Gamma_{loc} < 0$$

or: (2-3)

$$K_V \cdot \dot{\Gamma}_{loc} + \Gamma_{loc} < 0 \quad \wedge \quad \Gamma_{loc} > 0$$

  There are *two* criteria, because it must be possible to approach the localizer reference plane from two sides.

The Localizer Coupled control law uses a feedback of the error angle Γ_{loc} (see figure 2-9) and its time derivative $\dot{\Gamma}_{loc}$, which is approximated with a differentiating filter $\frac{s}{s+1}$. This filter is enabled as soon as the autopilot is turned on, to make sure that the transient effects of this filter have died out if the localizer mode is turned on. The filter has a very small phase lag; it approximates the actual derivative quite well. Without the feedback of $\dot{\Gamma}_{loc}$, the aircraft will tend to fly to the localizer reference plane as fast as possible, which means that it will enter the reference plane (almost) perpendicularly, in stead of making a smooth intercept. This is illustrated in figure 2-10.

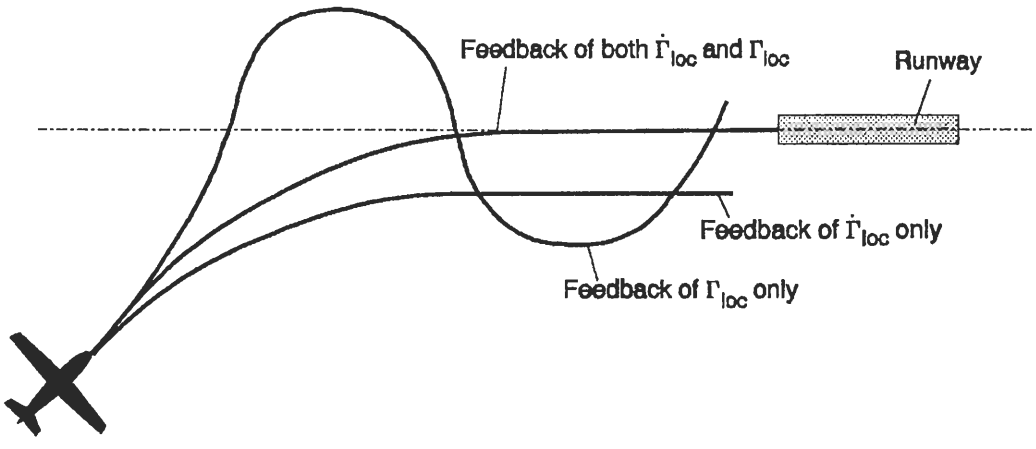


Figure 2-10. The effect of using Γ_{loc}, $\dot{\Gamma}_{loc}$, or a combination of both signals as feedbacks.

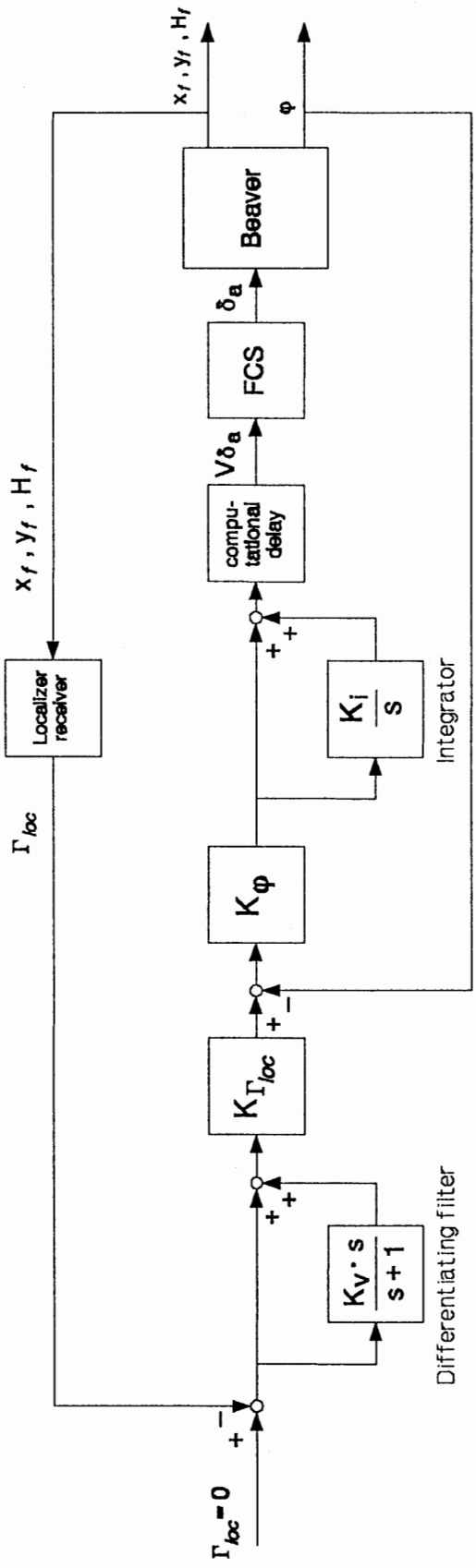


Figure 2-11. Block-diagram of the Approach Localizer mode. See the block-diagram of the Roll Attitude Hold mode (figure 2-7) for turncoordination loops.

The Localizer Coupled mode has replaced the Localizer Captured and Localizer Tracked modes, which were used in the previous autopilot, because the use of two different control laws for Captured and Tracked lead to undesirable transient effects [15]. Also, the switch-criteria did not function properly in the previous autopilot: often, the modecontroller switched from Armed to Captured rightaway, so $\dot{\Gamma}_{loc}$ was often not used at all.

If the aircraft nears the runway, i.e., if the distance R_{loc} to the beacon decreases, the Localizer Coupled control law gets more sensitive. This is due to the fact that Γ_{loc} will increase if the aircraft approaches the runway at a constant lateral distance d_{loc} from the runway centerline. Thus, this effect is similar to an increasing gain K_{loc}. To solve this problem, the gain $K_{\Gamma_{loc}}$ has been scheduled with a factor:

$$\frac{H_f}{\sin(|\gamma_{gs}| + \varepsilon_{gs})} + x_{loc} \quad \left(\approx \sqrt{R_{loc}^2 + H_f^2}\right) \tag{2-4}$$

which is approximately equal to the 3D-distance from aircraft to localizer transmitter. H_f is the altitude of the aircraft above the runway, x_{loc} is the X-distance from runway threshold to localizer transmitter, γ_{gs} is the nominal glidepath angle, and ε_{gs} is the error angle between the line through the aircraft and the glideslope transmitter and the nominal glideslope reference line (see section 2.2.4).

Again, H_f and γ_f need to be known. For the evaluation of the 'Beaver' autopilot, the data from Rotterdam Airport 'Zestienhoven' were used, but for commercial systems it is necessary to include a database with information about a large number of different airports. The gain scheduling only works correctly if the aircraft receives glideslope information. Without this information, it is not possible to include the angle ε_{gs} in the control law. Furthermore, it is important that x_{loc} is adjusted in case of a 'back-beam approach'.

A block-diagram of the Localizer Coupled mode is shown in figure 2-11. The gains depend upon V, see table 2-1 at the end of this chapter.

2.3.4 Navigation mode (NAV).

The Navigation mode guides the aircraft along a VOR-bearing, which is selected by the pilot. For this purpose, the angle between the desired VOR-bearing and the bearing where the aircraft actually flies, Γ_{VOR}, is used as feedback signal. Although in principle the NAV mode and LOC mode function in a similar way, the control structure of the LOC mode is not suitable for the NAV mode, because of the small magnitude of Γ_{VOR} at large distances from the VOR beacon.

In stead of the rate of change of Γ_{VOR}, the heading ψ is used. The combination of ψ and γ_{VOR} for the VOR mode has a similar effect as the combination of Γ_{loc} and $\dot{\Gamma}_{loc}$ for the LOC mode, see figure 2-10. At large distances, where the angle Γ_{VOR} is very small, the feedback of ψ ensures that the heading of the aircraft will correspond with the VOR bearing, so that, at least, the aircraft flies into the right direction.

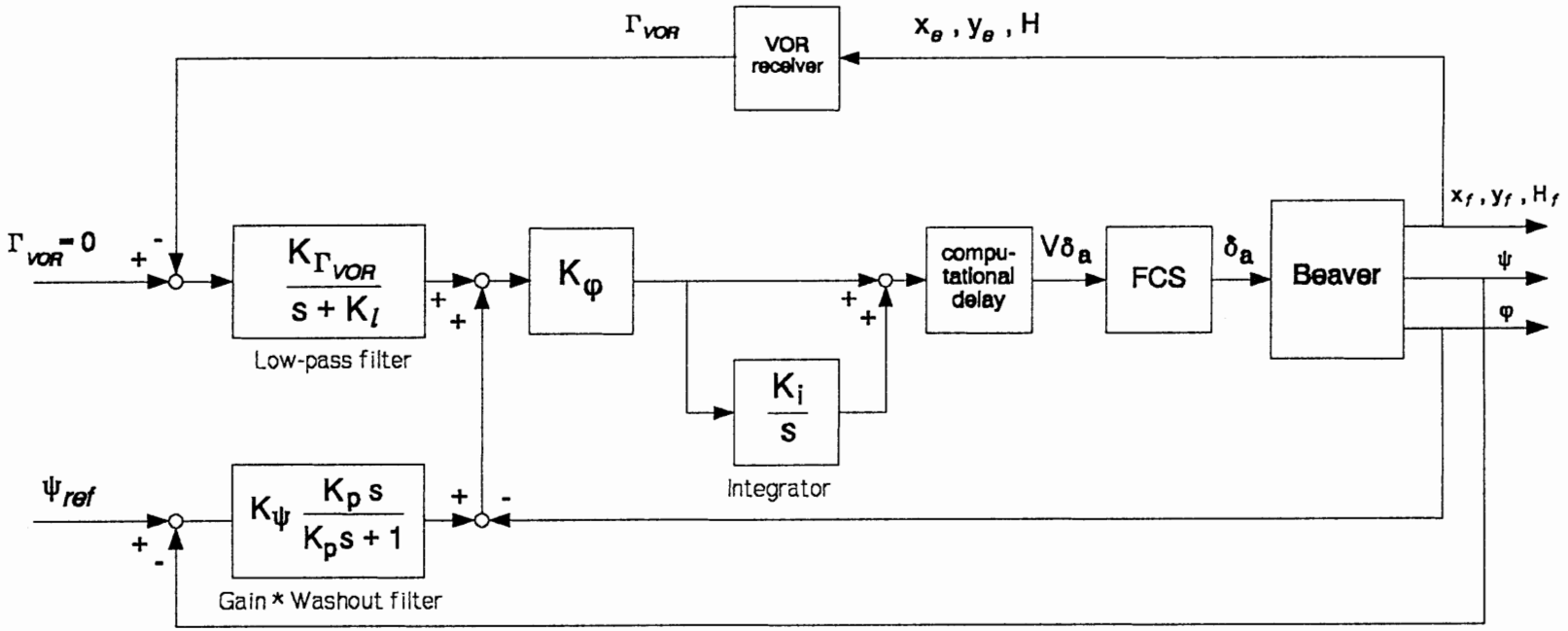


Figure 2-12. Block diagram of the Navigation mode. See the block-diagram of the Roll Attitude Hold mode (figure 2-7) for turncoordination loops.

The NAV mode too has two phases:

- Navigation Armed. This phase is engaged as soon as the NAV mode is turned on. The lateral autopilot mode in which the aircraft was flying before the NAV mode was selected by the pilot, usually HH, will be maintained until the aircraft flies through the VOR bearing for the first time. The mode controller then automatically switches to Coupled.

- Navigation Coupled. This phase is engaged if the aircraft passes the reference bearing for the first time ($\Gamma_{VOR} = 0$). The aircraft will turn towards the VOR bearing and follow the reference line. The NAV Coupled mode uses the RAH mode with turncoordination as inner loops. The signal Γ_{VOR} passes through a 1st-order filter which eliminates the high-frequency components from the VOR signal. A feedback-loop of the yaw angle ψ is used in stead of a $\dot{\Gamma}_{VOR}$-loop, because at large distances, the signal Γ_{VOR} is too small to obtain an accurate value of its derivative. The ψ-loop equals the outer-loop of the Heading Hold / Heading Select mode, equipped with a washout filter to make it possible for the aircraft to 'crab' along a VOR bearing if sidewind compensation is required. The gain $K_{\Gamma_{VOR}}$ transforms the resulting signal into a desired roll angle.

The block-diagram of the Navigation Coupled mode is shown in figure 2-12. Table 2-1 defines the gains at the end of this chapter. Since no DME information could be used due to hardware limitations, it was not possible to schedule the gains as function of the distance to the VOR beacon. This means that the sensitivity of the system grows if the aircraft nears the beacon. If the distance to the VOR transmitter is too small, the system will become instable. The gains have been chosen such that, if the aircraft cruises at 1500 ft, the system will become instable if the aircraft enters the 'cone of silence', an area where the VOR signals cannot accurately be received. See appendix C of part I and refs.[16] and [3].

2.3.5 Lateral part of the Go Around mode (GA).

The lateral part of the Go Around mode, which is used if an approach has to be cancelled, is a special case of the Roll Attitude Hold mode. It uses the control structure of the RAH mode to maintain a roll angle of zero degrees. Hence, it serves as wing-leveller for cancelled approaches. See section 2.2.5 for a description of the longitudinal part of the GA mode.

2.4 Turncompensation.

In turns, the assumption that the longitudinal and lateral motions of the aircraft are independent is no longer valid, because the aircraft tends to loose altitude if the pilot does not apply an appropriate elevator deflection. Although the autopilot has been equipped with an Altitude Hold control law, it is still necessary to compensate for this effect. First, it will be shown that a correction to the pitch-rate q is necessary. Then the correction for the loss of lift in turns will be added to yield the resulting nonlinear control loop for turncompensation.

2.4.1 Correction of the pitch-rate q in turns.

If the aircraft is turning with roll angle φ and yaw-rate r, the pitch-rate gyro will measure a pitch-rate $q_{tot} = q + r \cdot \cos \varphi$ in stead of q, see figure 2-13 (Ω is the resultant angular velocity about the Z_V - axis). This component should not be fed back to the elevator via the q-feedback loop in the longitudinal autopilot modes, because it has nothing to do with the *longitudinal* motions of the aircraft. If the measured pitch-rate q_{tot} is not corrected by subtracting $r \cdot \cos \varphi$, this term will yield a positive contribution to the elevator deflection, i.e: the q-feedback loop gives an unwanted nose-down command if the aircraft has a roll angle $\varphi \neq 0$.

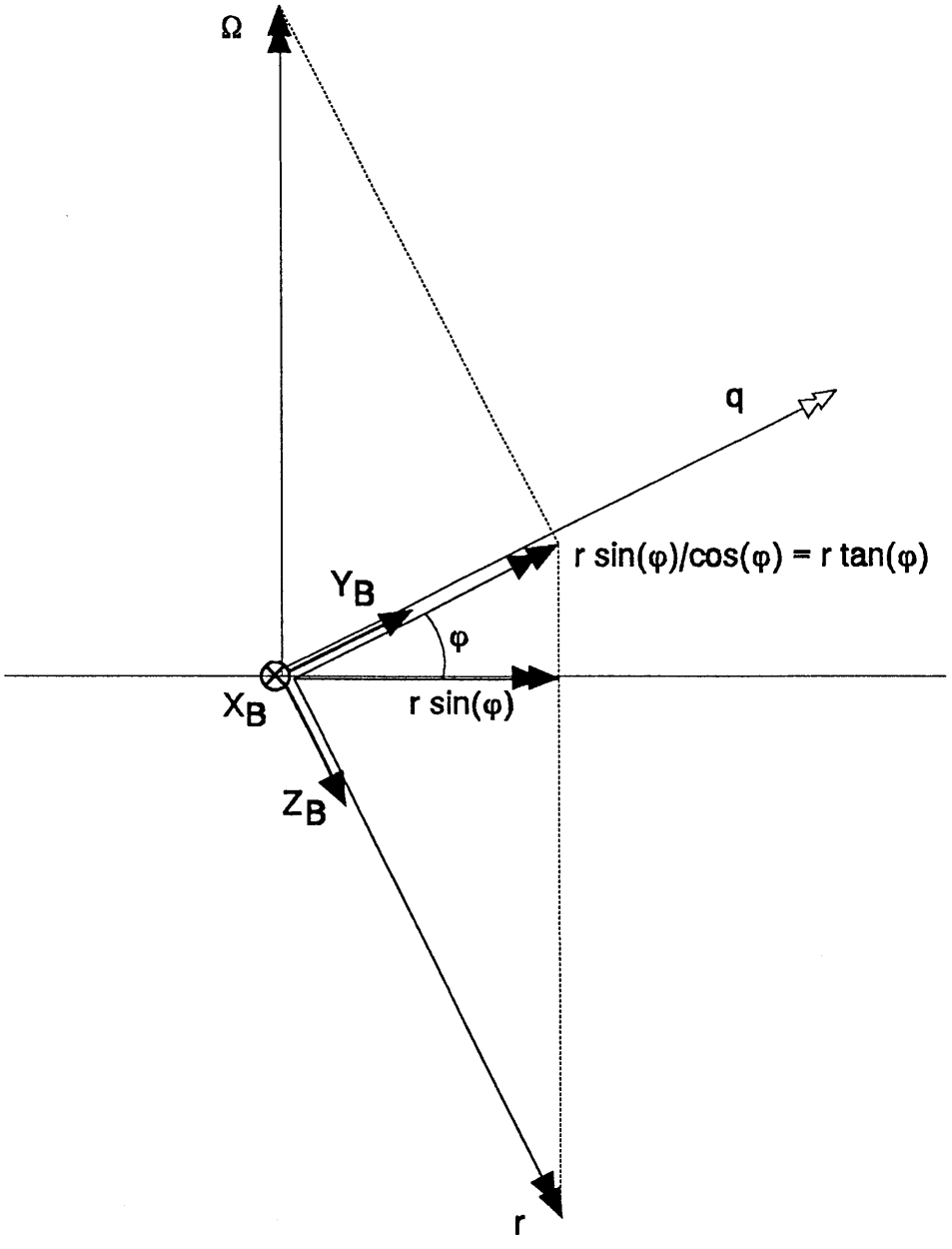


Figure 2-13. Additional contribution $r \cdot \tan \varphi$ to the pitch-rate if the aircraft has a roll angle $\varphi \neq 0$.

To compensate for the loss of lift, a negative (=upward) contribution to the elevator deflection, $\Delta\delta_e$ is needed. Although this will temporarily lead to a further loss of lift, due to the negative value of $C_{Z_{\delta_e}}$, the resulting lift will be greater, because the aircraft will rotate to a larger angle of attack. The additional elevator deflection to compensate for this loss of lift can now be written as:

$$\Delta\delta_e = K_{tc}^* \cdot (\sec\varphi - 1) \tag{2-6}$$

It is convenient to express the compensation for the loss of lift in terms of a correction of q too, so that this compensation can be combined with the correction for $r \cdot \tan\varphi$. This leads to the following total correction of q:

$$q := q - (r\tan\varphi + K_{tc} \cdot (\sec\varphi - 1)) \tag{2-7}$$

where K_{tc} equals $- K_{tc}^* / K_q$. The values of K_{tc} were obtained with simulations of the aircraft flying in Pitch Attitude Hold or Altitude Hold mode in combination with large roll attitude commands for the Roll Attitude Hold loops, see section 4.4.

The turncompensation for ALH needs to be somewhat stronger than the compensation for PAH, i.e., K_{tc} should be larger in ALH than in PAH mode (!), because of the different functions of the two modes.

- The ALH mode is designed to maintain a certain reference altitude. If the actual altitude differs from the desired altitude, a pitch-up or pitch-down movement is initiated by the ALH control law. The turncompensation law for the ALH mode has therefore been optimized to smooth out the variations in altitude, caused by the sudden loss of lift if the aircraft starts to bank. The ALH control law itself is too slow to smooth out this loss of lift entirely. The turncompensation in ALH mode does, however, imply that appropriate pitch-up or pitch-down commands have to be given, so it can be expected that θ will vary.

- In PAH mode, the altitude may vary, as long as the pitch angle θ remains constant. Due to the loss of lift after initiating a turn, θ varies strongly without turncompensation; the PAH loops themselves do not succeed to smooth out these initial variations of θ. However, K_{tc} shouldn't be as large as in ALH mode, because this would yield larger pitch-up commands immediately after initiating a turn. In practice, the PAH mode will only be used separately if a side-stick is used to generate the pitch attitude commands. Corrections for loss of *altitude* can then be generated by the pilot himself.

The longitudinal responses of the 'Beaver' to positive and negative roll angle commands differed quite much. To suppress this asymmetrical behaviour somewhat, the introduction of a small offset in the value of the roll angle which is used for the correction of q has been considered. Such offset yields the following

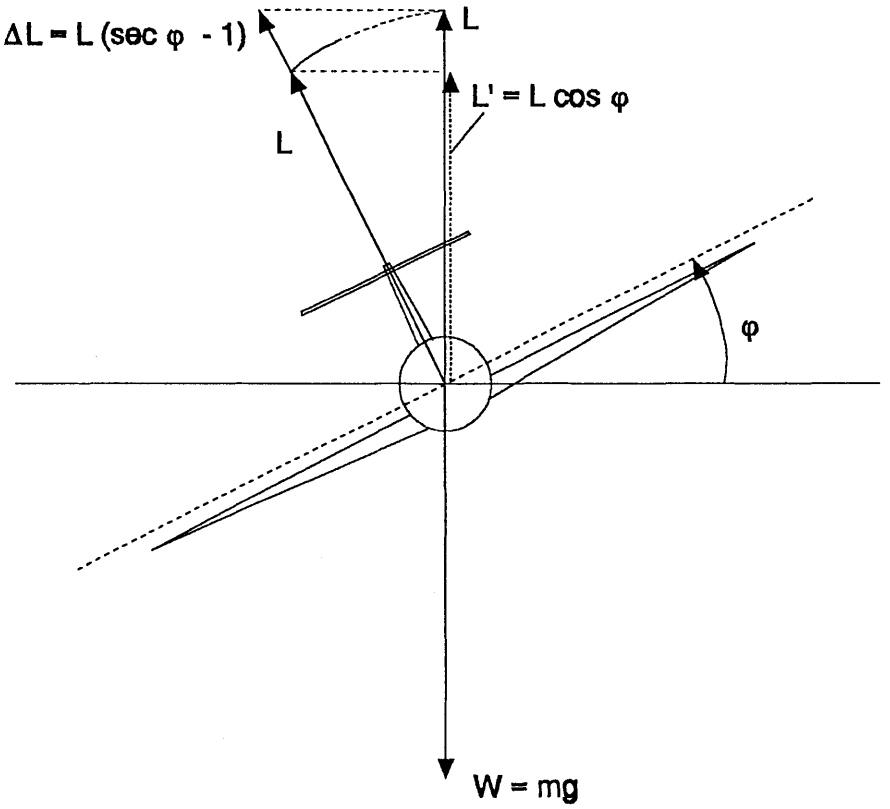


Figure 2-14. Loss of lift if the aircraft has a roll angle $\varphi \neq 0$.

2.4.2 Compensation for the loss of lift in turns.

In turns, it is also necessary to compensate for the loss of lift in Z_V - direction, which is illustrated in figure 2-14. In horizontal, symmetrical, stationary flight, the lift L equals the total weight $W = m \cdot g$. If the aircraft has a roll angle φ, the total lift should be $L + \Delta L$, to make sure that the lift component along the Z_V - axis still equals W. From figure 2-14 it can be seen that:

$$\begin{aligned} L' &= L \cdot \cos\varphi \\ L - L' &= L(1 - \cos\varphi) \\ \Delta L \cos\varphi &= L(1 - \cos\varphi) \end{aligned} \qquad (2\text{-}4)$$

So the extra lift necessary in turns becomes:

$$\Delta L = L\left(\frac{1}{\cos\varphi} - 1\right) = L(\sec\varphi - 1) \qquad (2\text{-}5)$$

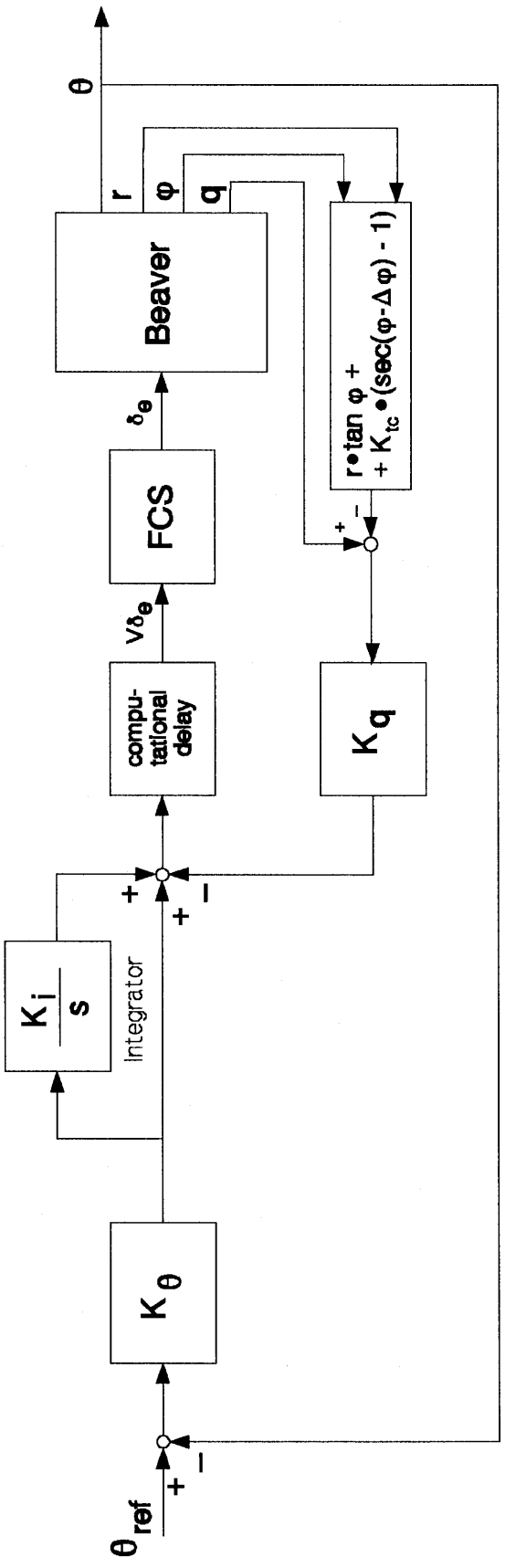


Figure 2-15. Block-diagram of the Pitch Attitude Hold mode (with turncompensation).

turncompensation law (compare with equation (2-7)):

$$q := q - (r\tan\varphi + K_{tc}\cdot(\sec(\varphi + \Delta\varphi) - 1)) \qquad (2\text{-}8)$$

Figure 2-15 shows the block-diagram of the PAH-mode with this turncompensator.

It turned out that the use of some offset in φ yielded better matching responses of left and a right turns, but when the *altitude* responses following a left and right turn started to look more alike, the differences in *pitch attitude* responses became larger, and vice versa. For this reason, a small offset in φ was used in practice only for ALH mode. In this mode some differences in pitch movements for left and right turns were considered to be tolerable, because this mode is not directly controlled via a side-stick. In PAH mode, the best compromise between matching H and θ - responses for left and right turns was to use no offset in φ. See section 4.4 for the results of the simulations.

2.5 The signal limiters.

For reasons of safety, the modecontroller of the 'Beaver' autopilot constantly checks if the magnitude of the inputsignals falls within the region, considered to be 'safe'. If the magnitude of the input signals exceeds certain upper or lower boundary values, the autopilot automatically switches off. These upper and lower boundaries have been listed in table 2-2. For commercial use, it is of course highly undesirable if autopilots autonomously decide to switch themselves off, but this safety measure seems appropriate for the experimental configuration considered here.

Signal	Lower boundary	Upper boundary
θ	-10 [deg]	20 [deg]
φ	-45 [deg]	45 [deg]
p	-12 [deg/s]	12 [deg/s]
q [1)]	-12 [deg/s]	12 [deg/s]
r	-12 [deg/s]	12 [deg/s]

Table 2-2. Boundaries used by modecontroller to decide if flight-condition is 'safe' or not.

1) During the flight-tests, it was found that the pitch-rate could exceed these values under heavy atmospheric turbulence. These pitch-rates would have occured in open-loop flight too; they were *not* caused by the elevator commands from the autopilot. For this reason the upper and lower boundaries for q were later increased to +/- 20 [deg/s].

To make sure that the aircraft will always fly in the 'safe' region defined in table 2-2, the control loops have been equipped with signal limiters. Since all autopilot modes make use of the inner-loops, it is not necessary to use limiters in the outer-loops. If the inner-loops work safely, the outer-loops will work safely too, because the inner-loops are controlled by the outer-loops.

In Pitch Attitude Hold mode and the inner-loops of the other longitudinal autopilot modes, the reference pitch-angle θ_{ref}, the difference θ_{ref} - θ, and the output signal of the integrator have to be limited:

- The reference pitch-angle must lie between the upper and lower boundaries defined in table 2-2, so the values from table 2-2 *minus* a small margin to make sure that the autopilot won't switch off will be used as upper and lower limits for θ_{ref}.
- The difference $\theta_{ref} - \theta$ needs to be limited to make sure that the pitch-rate q remains between the upper and lower boundaries from table 2-2. This limiter has been tuned by examining the q-responses to large pitch commands in linear and nonlinear simulations, see section 4.2.1 c. and ref.[16].
- Finally, the signal from the integrator needs to be limited to prevent 'integrator windup'. If the signal $\theta_{ref} - \theta$ has a value which is not equal to zero and which is positive or negative during a long time interval, the signal which leaves the integrator can become larger than the feasible elevator commands. If the error angle has finally become zero, it can take a long time before the integrator has been 'unloaded' again, which degrades the autopilot performance. The signal which leaves the integrator has been limited to values which correspond with the maximum and minimum feasible elevator deflections.

In Roll Attitude Hold mode and in the inner-loops of the other lateral autopilot modes, the reference roll angle φ_{ref}, the difference $\varphi_{ref} - \varphi$, and the signal which leaves the integrator have to be limited.

- The reference roll-angle has been selected such that the aircraft will turn 'normally', that is: without excessive loss of lift and airspeed, and without extreme values for the load factor n. The pilot will not have to increase the engine power, except for turns at very low velocities. Of course, the roll-angle needs to be kept within the region, defined by the boundaries from table 2-2.
- The difference $\varphi_{ref} - \varphi$ needs to be limited, to make sure that the roll-rate p will not exceed the limits defined in table 2-2. This limiter has been tuned by examining the p-responses to large roll-commands in linear and non-linear simulations, see section 4.3.1 c and ref.[16].
- Finally, the output from the integrator needs to be limited, to prevent that the integrator 'winds up'. The value which leaves the integrator must be large enough to suppress the influence of the engine upon the lateral movements sufficiently, but it needs to be limited, to make sure that the aircraft does not have a very large overshoot when the reference heading is reached. If the integrator is allowed to 'wind up', its output will grow excessively in large turns, which leads to unacceptable overshoots due to the time it takes to 'unload' the integrator.

The values of the limiters, used in the 'Beaver' autopilot have been listed in table 2-3. Note: the maximum values of the command signals to the elevator, ailerons, and rudder (-actuators) can be plus or minus 10 Volts, due to physical limits of the FCS. The resulting maximum and minimum deflections of the control surfaces are smaller than the values which can be applied by a human pilot.

Signal	Lower limit	Upper limit
θ_{ref}	-8°	18°
$\theta_{ref} - \theta$	-2 [V]	2 [V]
Output PAH-integrator	-10 [V]	10 [V]
φ_{ref}	-30°	30°
$\varphi_{ref} - \varphi$ [1)]	-3.75 [V]	3.75 [V]
Output RAH-integrator	-1 [V]	1 [V]

Table 2-3. Definition of signal limiters used in the control laws.

2.6 Conclusions.

All longitudinal and lateral autopilot modes, including turncoordination and turncompensation, have been discussed in this chapter. Block-diagrams of all systems have been given. The guidance modes use the Pitch Attitude Hold and Roll Attitude Hold control structures as inner-loops. The determination of all gains and filter coefficients has been treated in ref.[16]. The models presented in this chapter will be further analyzed with nonlinear simulations, and if necessary, changes will be made. The next chapter describes how the control laws have been implemented in SIMULINK. From chapter 4 onwards, the main results of the simulations will be discussed.

1) In the FCS models from ref.[5], the aileron angle δ_a was defined as: $\delta_a = (\delta_{a,left} - \delta_{a,right})$, whereas the definition, used in the aircraft models was: $\delta_a = \delta_{a,left} - \delta_{a,right}$. This has been taken into account in the expressions for the gains and limiters of the RAH mode.

Longitudinal Autopilot modes:

PAH	K_q	=	$-4.75e\text{-}04*V^2 + 0.0540*V - 1.5931$	[V·s/deg]
	K_i	=	0.5	[s^{-1}]
	K_θ	=	$-1.375e\text{-}03*V^2 + 0.1575*V - 4.8031$	[V/deg]
	K_{tc}	=	$0.02865*V^2 - 1.7189*V + 53.7148$	[V]
	$\Delta\varphi_{tc}$	=	0	[deg]
ALH	K_d	=	$-2.5e\text{-}03*V + 0.2875$	[s^{-1}]
	K_h	=	$-1.0e\text{-}04*V^2 + 0.015*V - 0.5975$	[V/m]
	K_{tc}	=	$1.7189*V + 14.3239$	[V]
	$dphi$	=	2	[deg]
	Other gains & coefficients: see PAH			
ALS	K_c	=	0.15	[s^{-1}]
	K_{als}	=	$-3.875e\text{-}04*V^2 + 0.04025*V - 1.1041$	[V·s/m]
	Other gains & coefficients: see PAH			
GS	K_r	=	1	
	$K_{\varepsilon_{gs}}$	=	$(0.00575*V^2 - 0.63*V + 18.00625)*\left(\frac{H - H_{RW}}{\sin\lvert\gamma_{gs}\rvert} + x_{gs}\right) * \frac{1}{1000}$	[V/deg]
	Other gains & coefficients: see PAH			

Lateral Autopilot modes:

RAH	dar	=	0.165	[V·s/m]
	K_i	=	0.25	[s^{-1}]
	K_φ	=	$9.75e\text{-}04*V^2 - 0.108*V + 2.335625$	[V/deg]
	drr	=	$-7.5e\text{-}05*V^2 + 0.0095*V - 0.4606$	[V·s/deg]
	K_r	=	−4	[V·s/deg]
HH	K_ψ	=	$0.05*V - 1.1$	[deg/deg]
	Other gains & coefficients: see RAH			
LOC	K_v	=	10	
	$K_{\Gamma_{loc}}$	=	$(7.75e\text{-}03*V^2 - 0.76*V + 15.75625)*\left(\frac{H - H_{RW}}{\sin\lvert\gamma_{gs}\rvert} + x_{loc}\right) * \frac{1}{1000}$	[deg/deg]
	Other gains & coefficients: see RAH			
NAV	K_p	=	15	[s]
	K_ψ	=	$0.05*V - 1.1$	[deg/deg]
	K_l	=	1	[s^{-1}]
	$K_{\Gamma_{VOR}}$	=	$5.375e\text{-}02*V^2 - 6.825*V + 153.03125$	[deg/deg]
	Other gains & coefficients: see RAH			

Table 2-1. Gain factors and other coefficients of the control laws.

Chapter 3. Implementation of the control laws of the 'Beaver' autopilot in SIMULINK.

3.1 Introduction.

The nonlinear SIMULINK model of the 'Beaver' dynamics, described in part I of this report, can be applied for detailed evaluation of the 'Beaver' autopilot. This chapter will show how the control laws can be implemented in a modular simulation structure in SIMULINK. Separate subsystems for the inner and outer loops, which contain many switches to set up the autopilot properly will be shown. In this chapter, the main structure of the SIMULINK models will be discussed; see appendix C for a further analysis of the autopilot simulation package.

3.2 General autopilot simulation structure.

Figure 3-1 shows the first level of the system *APILOT3*, which contains the complete simulation model of the 'Beaver' autopilot. The systems *APILOT2* and *APILOT1* contain the same simulation structure, without some external disturbances and sensor models. See appendix C for more details. Here, the complete model from *APILOT3* will be analyzed.

The following subsystems can be distinguished in figure 3-1:

1 - *Beaver dynamics.* This block calls the S-function *BEAVER*, which contains the nonlinear simulation model of the 'Beaver' dynamics. See part I of this report for a detailed description of the internal structure of the system *BEAVER*. The three inputvectors to this block contain wind velocities and accelerations, 'aerodynamic' inputs (flap and control-surface deflections), and engine inputs (engine speed and manifold pressure), respectively. These inputs are combined into one inputvector to the *'S-function'* (= SIMULINK system) *BEAVER* In the block *Beaver dynamics.*

The two output vectors from the block *Beaver dynamics* contain the state vector and rate of climb ($\mathbf{x}$ and $\dot{H}$), and a vector with dimensionless angular velocities along the three body-axes ($\mathbf{y}_{dl}$), respectively. Of these signals, the states and rate of climb are used by the control laws and the dimensionless rotational velocities are used by the Flight Control System models in the subsystem *FCS.*

2 - *FCS.* This block contains the linear third-order state space models of the Flight Control System for elevator, ailerons, and rudder (see appendix A). The input signals to this subsystem are the commanded *changes* in the deflections of these control surfaces ($V\Delta\delta_e$, $V\Delta\delta_a$, and $V\Delta\delta_r$ in Volts), and a vector with dimensionless angular velocities along the aircraft's body-axes ($\mathbf{y}_{dl}$); the outputsignals contain the real values of the changes in control surface deflections ($\Delta\delta_e$, $\Delta\delta_a$, and $\Delta\delta_r$ in rad).

3 - *Add initial inputs.* In this subsystem, the initial values of the inputsignals to the system *BEAVER* are added to the control commands, coming from

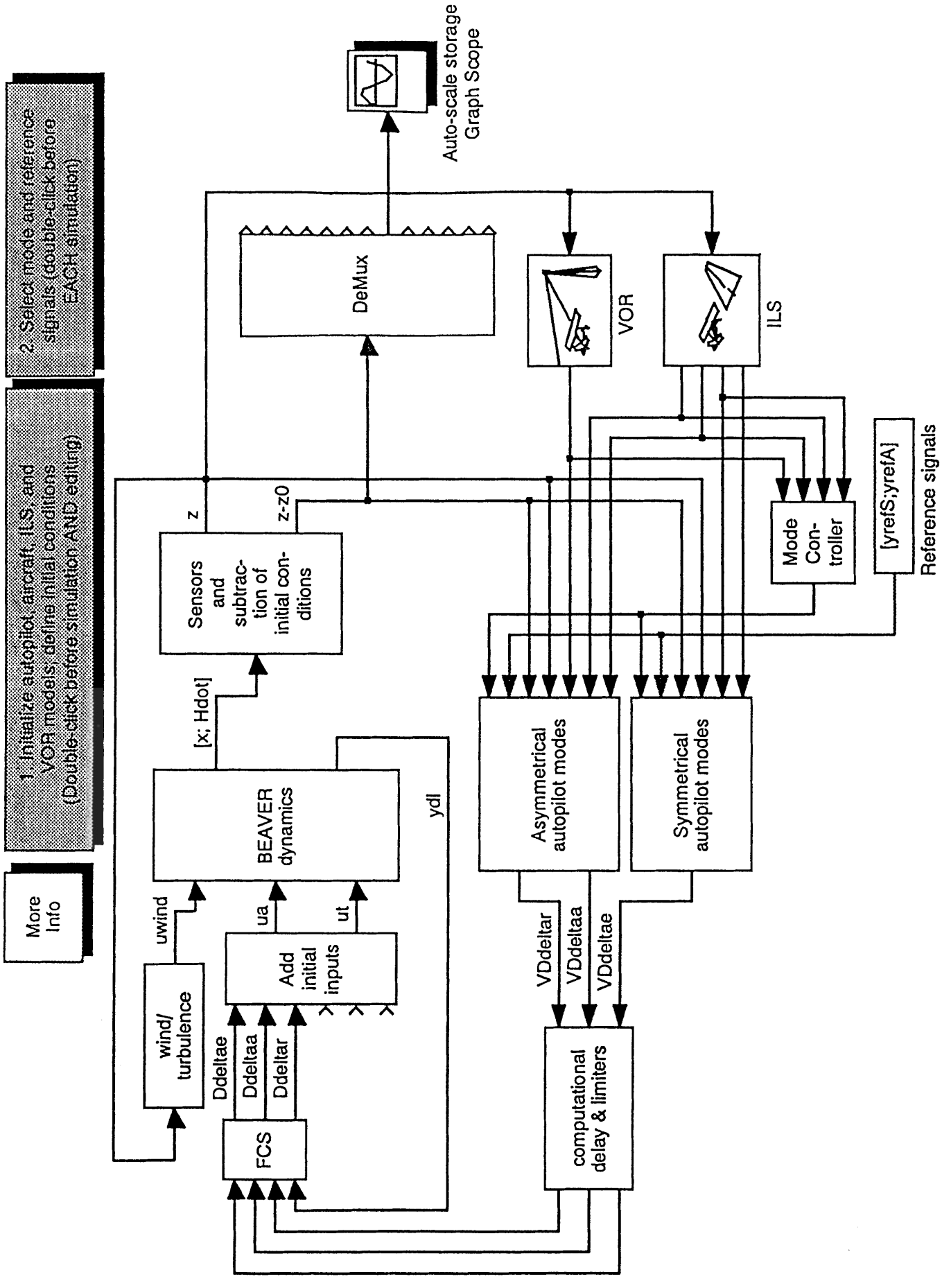


Figure 3-1. Main level of the system *APILOT3*.

the control laws. This is necessary, because the control laws compute *changes* in these control signals relatively to the initial values, *not* the actual values of the elevator, ailerons, and rudder deflections. Only the three input ports for control commands to the elevator, ailerons, and rudder have actually been connected. The remaining ports are intended to be used for control commands to the flaps, engine speed, and manifold pressure. The 'Beaver' autopilot does not control these signals (yet), but the *autothrottle* from ref.[8] does use the manifold pressure as control input. That system has not been implemented in SIMULINK yet (at least, not in the current modular simulation structure).

4 - *Sensors and subtraction of initial conditions.* In this block, sensormodels for all signals from the system *Beaver dynamics* can be included, although the system *APILOT3* only contains an altitude quantizer and Moving Average filter for the ALH mode, as well as some additional transport delays for the airspeed, altitude and rate of climb signals. The first outputvector from this subsystem is the vector **z**, which contains the values of the states and rate of climb *after* they have left the sensorblocks. The second outputvector from the subsystem *Sensors and subtraction of initial conditions* contains the deviations from the initial values of the elements of **z**, which will be called $\mathbf{z} - \mathbf{z}_0$. The 'real' values of the motion variables are needed for gain-scheduling, turncoordination, and turncompensation, whereas the deviations from the initial values are actually used in the feedback loops themselves. The subsystem *Sensors and subtraction of initial conditions* does *not* include blocks for the computation of the *ILS* and *VOR* signals.

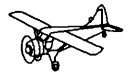
5 - *Symmetrical autopilot modes.* This block contains the symmetrical control laws, including turncompensation. See section 3.3 for more details.

6 - *Asymmetrical autopilot modes.* This block contains the asymmetrical control laws. See section 3.4 for more details.

7 - *Computational delay and limiters.* This block implements a computational delay of 0.03 seconds, which is equal to one sample period T_s (in reality, the computational delay may be shorter than 0.03 seconds, so this is the worst-case situation). Also, limiters for the output signal from the ROLM Flight Control Computer have been included in this subsystem, because the magnitude of those signals may not be larger than 10 Volt. If the signal limiters in the control laws themselves function properly, the limits of 10 Volt will never be reached, so for the current system, these limiters have been implemented for sake of completeness only.

8 - *Mode Controller.* This block takes care of the settings for all switches from the control laws. It also contains criteria for mode-switching, which are checked continuously during simulations. See section 3.5 and appendix C for more details.

9 - *Reference signals.* This is actually a *Constant* block, which contains the reference inputs for the control laws. It is also possible to create arbitrary, time-dependent reference signals for the different control laws, in which case this block needs to be replaced. See also section 3.5 and appendix C.



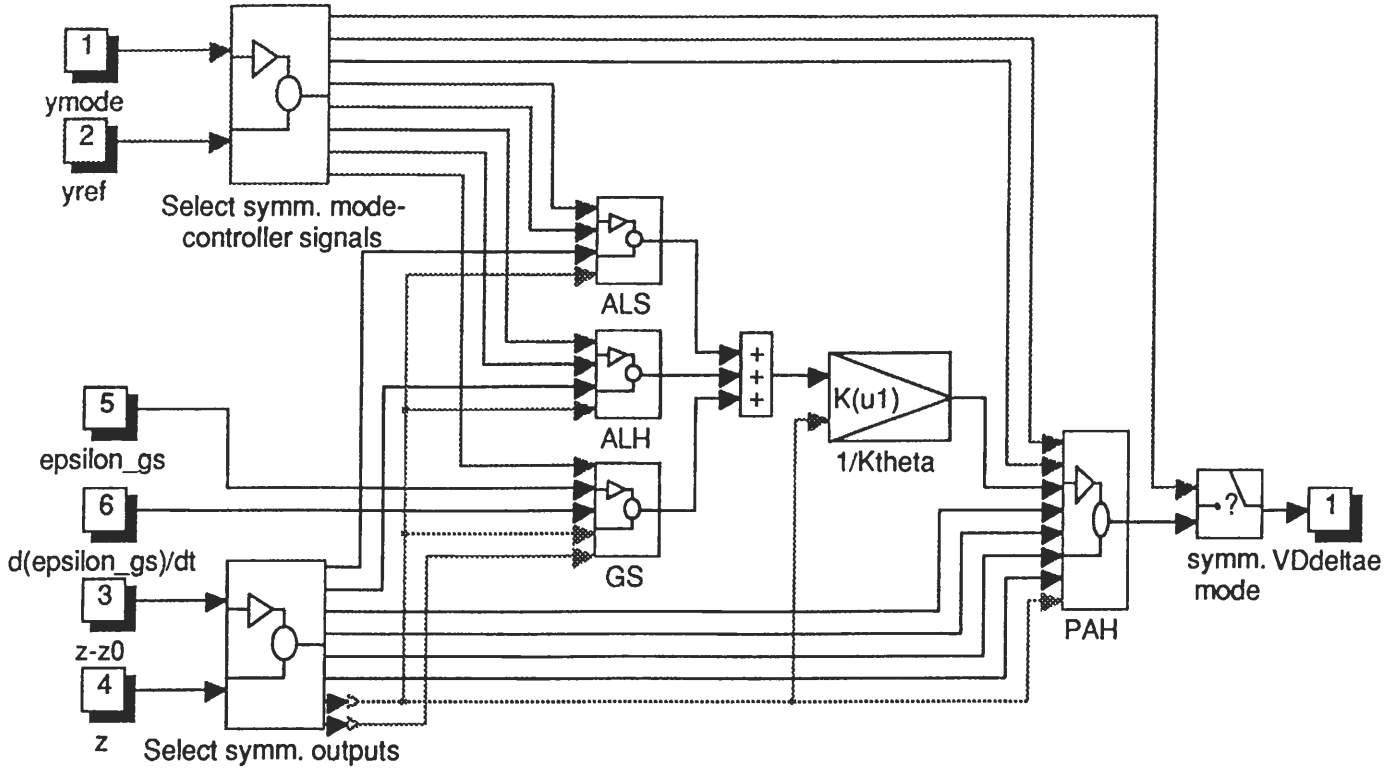


Figure 3-2. Internal structure of the subsystem
Symmetrical autopilot modes.

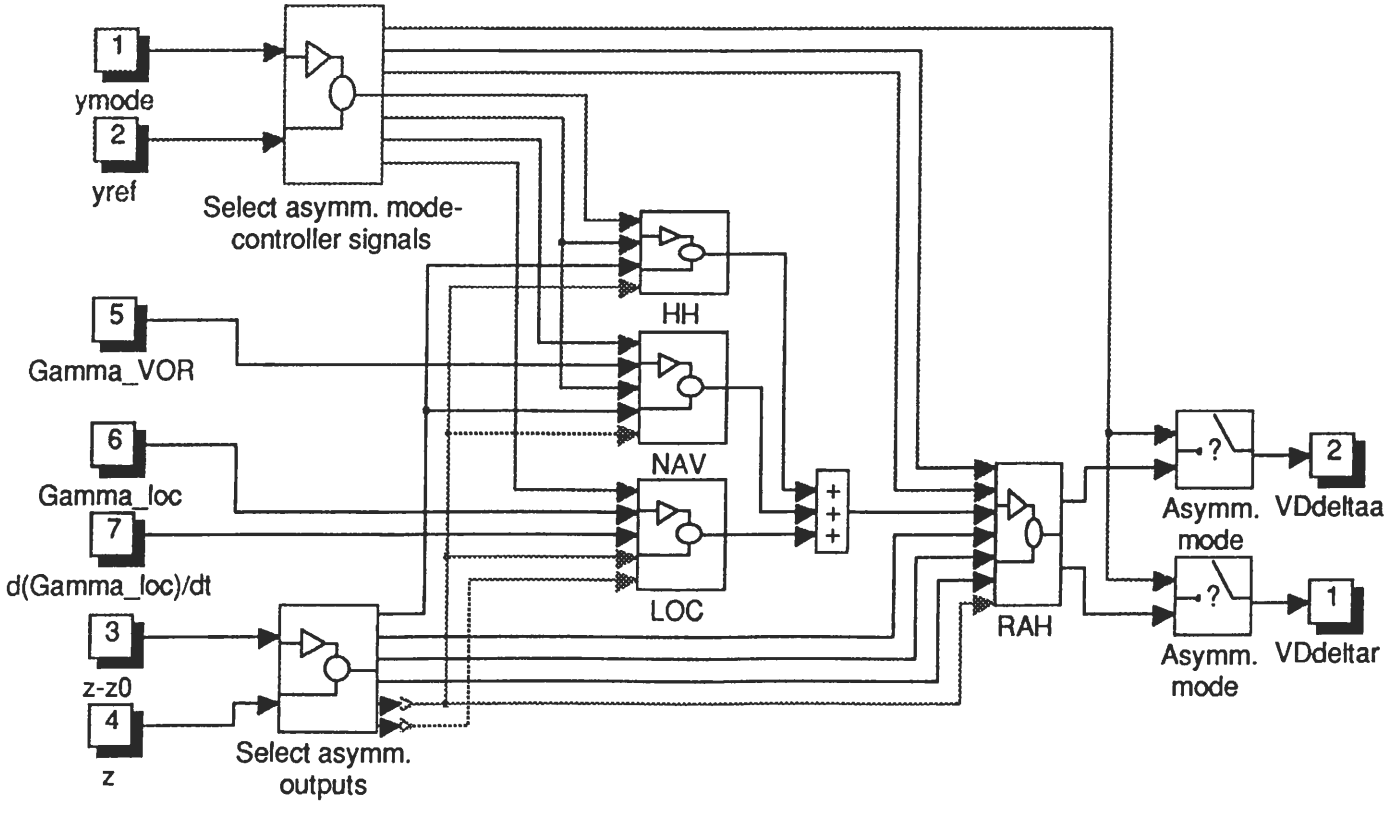


Figure 3-3. Internal structure of the subsystem
Asymmetrical autopilot modes.

9 - *ILS.* In this block, the ILS signals for approach guidance along the glide-slope and localizer are generated, and their time-derivatives are approximated with a differentiating filter, used by the approach modes of the 'Beaver' autopilot (see sections 2.2.4 and 2.3.3). In the system *APILOT3*, ILS noise signals have been included, *APILOT2* contains the nominal signals only.

10- *VOR.* This block is used to compute the VOR signals, used by the navigation mode of the autopilot.

11- *Wind and turbulence.* This block generates disturbances from wind and atmospheric turbulence, which will be used as inputsignals by the subsystem *Beaver dynamics.* See appendix C for more details.

Three buttons which can be double-clicked with a mouse to initialize the system, set the appropriate control law, or get on-line information, have been included on top of the blockdiagrams. These blocks will start MATLAB programs which set parameters and/or load datafiles into the MATLAB workspace. See appendix C for more details.

3.3 Implementation of the symmetrical autopilot modes in SIMULINK.

The internal structure of the subsystem *Symmetrical autopilot modes* is shown in figure 3-2. The subsystems *ALH, ALS,* and *GS* contain the outer loops of the Altitude Hold, Altitude Select, and Glideslope coupled modes; the subsystem *PAH* contains the Pitch Attitude Hold loops, the turncompensator, and the inner-loops for the other symmetrical autopilot modes (which are almost equal to the PAH mode itself, except for an additional washout filter in the θ-loop). These blocks are the SIMULINK equivalents of the systems, shown in sections 2.2, 2.3, and 2.4. Their internal structure will be treated in more detail in appendix C. Due to the modular structure, the division between *guidance* and *control* tasks is clearly visible in figure 3-2.

The inputs coming from the upper left corner of the system *Symmetrical autopilot modes* are the vectors $\mathbf{y}_{mode}$ and $\mathbf{y}_{ref}$, which contain mode-switches and reference signals for the control laws, see section 3.5 and appendix C. Only those elements from these vectors which are actually used by the symmetrical control laws are extracted from these vectors in the subsystem *Select symm. modecontroller signals.* The subsystem *Select symmetrical outputs* extracts the motion variables which are needed by the symmetrical control loops from the outputvectors coming from the sensormodels. Furthermore, glideslope signals ε_{gs} and $\dot{\varepsilon}_{gs}$, which come from the subsystem *ILS,* are used as inputs to the system *Symmetrical autopilot modes.* See appendix C for more details.

3.4 Implementation of the asymmetrical autopilot modes in SIMULINK.

The block *Asymmetrical autopilot modes* contains the control laws for the Roll Attitude Hold, Heading Hold/Heading Select, Localizer, and Navigation modes. The system has the same modular structure as the block *Symmetrical autopilot*

modes, with a clear division between inner and outer loops, representing *guidance* and *control* tasks, see figure 3-3. The block *RAH* contains both the Roll Attitude Hold control law, which forms the inner-loop of the other asymmetrical modes, and the turncoordination loops.

The subsystem *Select asymm. modecontroller signals* is used to extract the reference signals and mode-switches, needed by the asymmetrical autopilot modes. In the lower left corner, 'measured' motion variables, coming from the *Sensor* models, are sorted. Only the signals which are actually needed by the asymmetrical autopilot modes have been connected to the blocks *HH*, *NAV*, *LOC*, and *RAH*. Finally, lateral radio-navigation signals from the ILS and VOR sensors (Γ_{loc}, $\dot{\Gamma}_{loc}$, and Γ_{VOR}) enter the subsystem *Asymmetrical autopilot modes*, in order to be used by the NAV and LOC control laws. See appendix C for more details.

3.5 The mode controller and reference signals.

The simulation model contains a large number of *On/off-switches* which make it possible to select the exact configuration of the control laws by switching control loops on or off. The switches themselves are controlled by inserting a *'switch-signal'* with a value of either 1 or 0, which, as usual, correspond with *'on'* or *'off'*[1]. All switch-signals have been put together in the vector $\mathbf{y}_{mode}$ which is one of the inputvectors to the subsystems *Symmetrical autopilot modes* and *Asymmetrical autopilot modes*. Another inputvector to these subsystems is $\mathbf{y}_{ref}$, which contains the reference signals for the different autopilot modes (not including the reference values for the Navigation and Approach modes, because those values are obtained from the subsystems *ILS* and *VOR*). The exact definitions of $\mathbf{y}_{mode}$ and $\mathbf{y}_{ref}$ can be found in tables C-2 and C-3 of appendix C.

In practice, many simulations are done with one particular switch-setting, corresponding to one particular autopilot mode. However, since some modes use both an *Armed* and a *Coupled* phase, on-line mode-switching facilities must be included in the system. The system *APILOT3* contains a block *Mode Controller* which inserts the proper switch-settings into the system, and which makes it possible to switch between two different symmetrical modes and two different asymmetrical modes independently. Longitudinal and lateral switch-criteria for the Approach and Navigation modes are used, see equations (3-1) to (3-3). The initialization of the switch-settings can be done with a helpfile *APSTART*, which will define the switch-setting in the MATLAB workspace, see appendix C.

The following switch-criteria for the Approach modes (i.e., *Glideslope Armed* to *Glideslope Coupled* and *Localizer Armed* to *Localizer Coupled*) and switch-criterion for the Navigation mode (i.e., *Navigation Armed* to *Navigation Coupled*) have been implemented in the SIMULINK simulation structure:

Glideslope Armed to Coupled:

$$\varepsilon_{gs} = 0 \tag{3-1}$$

---

[1] The block *On/off switch* is not contained in the standard SIMULINK blocklibraries. See section B.5 of appendix B for more details.

Localizer Armed to Coupled:

$$K_v \cdot \dot{\Gamma}_{loc} + \Gamma_{loc} > 0 \quad \wedge \quad \Gamma_{loc} < 0$$

or: (3-2)

$$K_v \cdot \dot{\Gamma}_{loc} + \Gamma_{loc} < 0 \quad \wedge \quad \Gamma_{loc} > 0$$

Navigation Armed to Coupled:

$$\Gamma_{VOR} = 0 \tag{3-3}$$

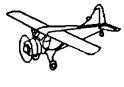
See sections 2.2.4, 2.3.3, and 2.3.4 for more details. Note: the mode controller can switch from *Armed* to *Coupled* only once, so the mode can never be switched back to *Armed*!

In the real aircraft, a far more complicated mode controller is used, which not only selects the appropriate subroutines for all autopilot modes, but which also sets the lights on the annunciator panel, checks if the flight condition can be considered 'safe', etc. This program also contains additional switch-criteria which are used to decide when the aircraft must switch from *Altitude Select* to *Altitude Hold*, depending upon the distance to the reference altitude. Since the ALH and ALS modes have been analyzed separately for this resport, these mode-switches have not been implemented here.

3.6 (Future) relationships between the simulation model and the software of the Flight Control Computer.

The C-programs for the Flight Control Computer which contain the control laws of the autopilot for evaluation of the system in real flight have been developed independently from the SIMULINK simulation structure from this report. These C-programs were based upon older assembler programs for the ROLM computer, which contained the control logic for the autopilot from ref.[9]. Both the assembler and the C-programs are very hard to understand, because all kinds of I/O functions, conversion tables, validity checks, bit-operations, and functions to switch the mode-indicator lights from the annunciator panel on or off have been mixed-up with the actual control laws.

In theory, a closer link between the SIMULINK implementation of the control laws and the FCC software, used for the flight tests, may be achievable if a C-code generator is applied to the SIMULINK system(s) which contain(s) the control laws. Although the development of the interface between SIMULINK and the FCC is not easy (and certainly not trivial!), it is really worthwhile trying to realize it, because such an interface will shorten the control law design and evaluation process considerably, thus eliminating possible conversion errors. If a close link between SIMULINK and the FCCs can be realized, it may also be possible to evaluate the control laws in a real-time flightsimulator, using exactly the same software subroutines as in the Flight Control Computers. This effectively creates the possibility of performing *on-line, closed-loop* laboratory tests, which give a better validation of the control laws in the FCCs than the open-loop tests which were done for the 'Beaver' autopilot software (ref.[16]).



3.7 Conclusions.

In this chapter, the simulation model of the autopilot has been presented. It has a modular structure, so it is easy to change or update certain elements, including the complete nonlinear aircraft model, if necessary. The control laws themselves also have been implemented in a modular structure, which exhibits a clear distinction between inner and outer loops. It is recommended to develop software and/or hardware tools with which automatic conversion of control laws from the SIMULINK simulation structure to the Flight Control Computers of the real aircraft, or from SIMULINK to a real-time simulation model of a flight-simulator becomes possible. In that case, the time, needed to develop automatic controllers will be shortened considerably, and the risks of making conversion errors will be reduced.

Chapter 4. Nonlinear analysis of the basic autopilot modes.

4.1 Introduction.

In this chapter, the performance of the basic autopilot modes will be evaluated by means of nonlinear simulations. These results complement the linear results from refs.[15] and [16]. Here, only the basic autopilot modes, being Pitch Attitude Hold, Roll Attitude Hold, Altitude Hold, Altitude Select, and Heading Hold, will be analyzed. However, it is not difficult, and strongly recommended, to reproduce these results and to create other simulations with the simulation package, which has been included with this report. See chapter 3, appendix C, and part I for more details about the SIMULINK simulation structure. The results for the inner loops (PAH and RAH) are very important, because these control laws serve as inner-loops for the other autopilot modes.

In the following sections, each 'basic' autopilot mode will be analyzed separately. The plots with the results of the simulations have been included at the end of this chapter, because they require a lot of space. The initial conditions for the simulations were computed with the aircraft trim routine *ACTRIM*, which has been discussed in part I of this report (appendix E and section F.2.7). They have been listed in table 4-1. Only one initial altitude has been used, because in ref.[16], it was shown that the influence of the altitude upon the performance of the 'Beaver' autopilot is negligible.

Variable	Airspeed [m/s]		
	35	45	55
α [rad]	0.2189	0.1260	0.0777
β [rad]	-0.0226	-0.0137	-0.0117
p, q, r [rad/s]	0	0	0
ψ [rad]	0	0	0
θ [rad]	0.2189	0.1260	0.0777
φ [rad]	0	0	0
x_e, y_e [m]	0	0	0
H [ft!]	2000	2000	2000
$\dot{H}$ [m/s]	0	0	0
δ_e [rad]	-0.1087	-0.0290	0.0014
δ_a [rad]	0.0081	0.0093	0.0094
δ_r [rad]	-0.0646	-0.0442	-0.0416
δ_f [rad]	0	0	0
n [RPM]	1800	1800	1800
p_z ["Hg]	21.40	22.66	26.47

Table 4-1. Steady-state flight-conditions, used as initial conditions for the simulations from this chapter.

4.2 Symmetrical autopilot loops.

4.2.1 Pitch Attitude Hold.

a. Step responses.

Figures 4-1 and 4-2 show symmetrical aircraft responses to a commanded input $\Delta\theta_{ref} = 1°$ to the Pitch Attitude Hold control law, computed for initial velocities of 35 and 55 m/s respectively. The responses are quite fast and well-damped, and they look similar to the linear results from ref.[16]. The overshoot, which increases somewhat for a larger velocity, is approximately 20%. This has been tolerated, because the response is well-damped, and because this controller yields accurate and fast symmetrical inner-loops. Moreover, in ref.[16], it was shown that the phase and gain margins for this controller were large enough according to common guidelines.

The motion quantities all remain well within the limits, defined in table 2-2, although the initial elevator deflection is really fast. In practice, step inputs to the PAH control law will be generated only in Go Around mode. For reasons of safety, the connection between the actuators and the steering cables of the 'Beaver' has been equipped with fuse pins, which will break immediately if the pilot forcefully pulls the control yoke, or if the rate of change of the control deflections exceeds the structural limits. Also, *fader* routines, which act as some kind of rate-limiters for the control deflections, have been included in the flight control software.

Figure 4-3 shows step-responses to $\Delta\theta_{ref}$-inputs of different magnitude, which have been scaled back to $\Delta\theta_{ref} = 1°$, to analyze the influence of nonlinearity (note: the results for $\Delta\theta_{ref} = -1°$ have been multiplied with a factor −1!). Again, the symmetrical autopilot mode is Pitch Attitude Hold, and the asymmetrical motions are governed by the Roll Attitude Hold mode. Here, only responses for a low initial airspeed (V = 35 m/s) have been shown, because for higher airspeeds the responses only get better.

The shape of the responses to step inputs up to 4 degrees (for higher airspeed even up to 8 degrees) is not affected strongly by the nonlinearity of the system. For larger pitch-commands, the system is destabilized because of the influence of the δ_e-limiters, which can be seen clearly in the δ_e-response itself. It is, however, important to realize that for large θ-steps, the airspeed drops below the speed-range in which the aerodynamic model for the 'Beaver' is valid. In reality, the aircraft will probably stall for the largest θ-steps at low airspeeds, unless the pilot applies sufficient engine power. For this reason, the Go Around mode actually needs to be used only together with an autothrottle (which may need to have a Go Around setting itself), or else, the control behaviour in Go Around mode should be coupled to the engine power applied by the pilot. It is not hard to imagine how dramatic the response in Go Around mode will be if the pilot does not apply sufficient engine power.

Also notice the increase in sideslip angle, which is caused by the drop in airspeed that affects the slipstream of the propeller (see the responses of *dpt*). Here it is visible that no real sideslip suppression has been included in the control system; the turncoordinator from the RAH mode only helps reducing β in turns! If the pilot increases the engine power to maintain the nominal

airspeed, *dpt* will also increase, so β will still be affected. The pilot can, however, correct this by trimming the aircraft for the new equilibrium condition. The Roll Attitude Hold mode does not succeed in keeping the roll angle φ zero for the largest θ-steps, which result in large changes in sideslip angle.

During the autopilot design process, it has been considered to include a feedback-loop of *dpt* to the rudder-channel, to compensate for sideslipping due to the decrease in airspeed or increase in engine power. Some preliminary analysis of this solution showed that this might work, although the resulting control law was very sensitive for changes in the feedback gains. The idea was abandoned completely when it became clear that this control law could not be implemented in the real aircraft in time for the flight tests, due to the limited number of signals which were available for control purposes.

Even though it is not possible to evaluate this compensation law in real flight anymore, since the 'Beaver' has been replaced by a Cessna Citation II, it is still interesting to analyze this idea further, using the nonlinear aircraft models from this report.

b. Responses to atmospheric turbulence.

Figure 4-4 shows responses of the 'Beaver' to atmospheric turbulence for the open-loop situation and for Pitch Attitude Hold and Roll Attitude Hold enabled. These plots have been determined for an initial velocity V = 45 m/s, and an initial altitude H = 2000 ft (see table 4-1 for the initial values of all motion variables). Here the longitudinal responses will be analyzed first; see section 4.2.2.b for details about the lateral responses. The dashed lines in figure 4-4 show the open-loop responses of the 'Beaver', the solid lines represent the responses of the 'Beaver' under PAH and RAH control. The turbulence velocities along the aircraft's body-axes were generated with the Dryden filters, implemented in the blocks *UDRYD1*, *VDRYD1*, and *WDRYD1* from the SIMULINK blocklibrary *WINDLIB* (see section F.3 of part I), which were set up for turbulence with a scale length L_g = 150 m, and a standard deviation σ_g = 1 m/s.

As can be seen from this figure, the Pitch Attitude Hold mode does a good job in maintaining a constant value of θ under atmospheric turbulence. This results in smaller changes in aircraft altitude, although that is not the first requirement for Pitch Attitude Hold. The differences between the open and closed-loop responses of the other symmetrical motion variables are small, apart from the q-response, which is linked directly to θ. It is important to notice that the airspeed, angle of attack, and sideslip angle *include* the contributions from atmospheric turbulence, contrary to ref.[12], where the turbulence influence was modelled in terms of *deviations* of V, α, and β, which were added to the nominal values of these variables.

The elevator makes fast control movements, although the total deviations from the nominal elevator deflection are small compared to the elevator deflections which are applied for the control of $\Delta\theta$-*steps*. Control deflections like these seem to be acceptable at first sight, even though no objective criteria are available at this moment.

The results from figure 4-4 have been determined *after* the flight tests in januari, because unlike the current SIMULINK environment which has been

described in part I of this report, the tools which were available at that time (see ref.[13]) did not yet have the power and flexibility, needed for such analysis. Those models were approximately 6 to 10 times as slow, so even if they had contained the expressions, needed to analyze aircraft responses to atmospheric turbulence, the simulation times would have been excessively large, due to the very small time-steps needed to simulate systems with noisy inputs. However, figure 4-4 confirms the first, qualitative results of the flight tests, described in ref.[15], which showed good control behaviour both with and without atmospheric turbulence.

c. Tuning limiters.

In section 2.5, it was said that in order to keep the pitch-rate below 12 degrees per second, the signal $\theta - \theta_0$ had to be limited. For reasons of safety, the mode controller would have disengaged the autopilot automatically if this limit had been exceeded. This $\theta - \theta_0$ limiter is visible in the block-diagram of figure C-14 in appendix C. The limits have been determined by comparing q-responses to a step input $\Delta\theta_{ref} = 8^\circ$ in PAH mode, which, according to figure 4-2, is already a very large pitch-step. In Figure 4-5, this is illustrated for $V_0 = 55$ m/s, because the largest values of q were obtained for this airspeed. The different lines in these plots represent different settings of the $\theta - \theta_0$ limiter; the solid line gives the final result. In order to provide a sufficient margin to the boundaries of 12 deg/s, the limiter has been tuned such as to limit the pitch-rate to approximately 10 deg/s. With this setting of the limiter, the step-response of θ itself is hardly affected. For lower airspeeds, the maximum pitch-rate was even smaller, and again, the influence upon the θ-responses was hardly noticeable.

4.2.2 Altitude Hold.

a. Step responses.

Figures 4-6 and 4-7 show responses for Altitude Hold mode, following reference inputs $\Delta H_{ref} = 1$ m, for initial speeds of 35 and 55 m/s, respectively. The responses look smooth and fast, although there is some overshoot for higher airspeeds. For $V_0 = 55$ m/s, the overshoot equals 20%. The final value of the altitude is reached almost completely after 30 seconds, which is considerably faster than the results from ref.[9], but fully in accordance with the linear results from ref.[16]. For low airspeeds, the overshoot reduces considerably, and the increase in pitch angle is maintained much longer. This is probably caused by the drop in airspeed, which apparently has a larger influence at low airspeeds. The remaining difference between the required altitude and the actual altitude, which is not really a steady-state error because this difference does reduce slowly, is equal to a few centimetres only.

During the laboratory tests of the FCC software, it was found that the Airdata Computer used a Least Significant Bit (LSB) of 4 ft for the altitude signal, which was necessary in order to be able to cover the complete altitude range of the 'Beaver' with a limited number of bytes. The influence of this LSB has been analyzed by determining step responses for ALH mode with an altitude signal that was quantized at a resolution of 4 ft. In figure 4-8, step responses

for an initial airspeed of 35 m/s are shown; similar results were obtained for higher airspeeds. Here, the altitude step is four metres, so it covers four times the LSB. The reference value of 4 m lies somewhere between 3*4 = 12 and 4*4 = 16 feet. In the responses of H, both the real altitude and the measured altitude are shown.

The altitude response for the LSB of 4 ft differs completely from the response for a continuous signal. In stead of smoothly intercepting the reference altitude of H_0 + 4 metres, the aircraft continues its climb after crossing the reference altitude and it starts to make bouncing movements around an altitude of 16 feet (≈ 4.88 m). The reason why the aircraft does not level off at 4 metres is obvious: even though the actual altitude crosses the reference value, the *measured* value of $H - H_{ref}$ constantly equals 12 ft minus 4 metres (≈ 3.66 – 4 = –0.34 m) until the aircraft crosses an altitude of 16 ft, which yields $H - H_{ref}$ ≈ 4.88 – 4 = +0.88 m. At that moment, the controller finally responds by applying an elevator deflection that presses the aircraft's nose down, however, this correction is stopped immediately after the aircraft has crossed the 16 ft line again. So effectively, a *bang-bang* input signal to the ALH controller has been created. Although the resulting error in altitude of approximately 0.88 metres is acceptable, the response itself is not, which is clearly visible in the plots of the elevator deflection δ_e and the resulting responses of the pitch-rate q and pitch-angle θ.

It was not possible to increase the resolution of the altitude signal, due to limitations of the aircraft's instrumentation. Also, it was not possible to reconstruct the altitude by integrating the rate-of-climb signal, because $\dot{H}$ itself was determined from that same altitude signal in the airdata computer. In order to reduce the bang-bang effect, it was decided to apply a Moving Average filter (*MA-filter*) to the altitude signal, which computes the altitude by taking the mean value out of the last ten samples of H. See section B.3 of appendix B for some general information about this discrete filter.

In figure 4-8, the dashed lines represent the altitude responses for the situation that a MA-filter has been applied after the quantizer block. The ten-sample MA-filter effectively reduces the quantization interval by a factor ten, although it induces an additional time delay of ten sample times = 0.3 seconds. The result is not yet perfect, but it is clearly visible that the severe changes of pitch-angle due to very fast elevator 'pulses' do not occur anymore when a MA-filter is used, hence, the performance of the ALH mode has been restored to an acceptable level. Also, the additional time-lag of 0.3 seconds does not seem to affect the responses in any way.

b. Responses to atmospheric turbulence.

Figure 4-9 shows responses of the 'Beaver' to atmospheric turbulence if the aircraft flies in Altitude Hold/Heading Hold mode, compared with open-loop responses. In section 4.3.2, the asymmetrical results will be analyzed, here the symmetrical responses will be treated first. It is obvious that the variations in altitude are smaller when the ALH mode is engaged, but the differences are not as spectacular as one should expect. However, in open-loop configuration, the turbulence excites the phugoid of the aircraft, which gets an amplitude that increases in time. Under ALH control, the motions remain limited until the end of the simulation. So even though the performance of the autopilot to atmos-

pheric disturbances was never analyzed *before* the flight tests were done, the result still seems to be quite good.

4.2.3 Altitude Select.

In figures 4-10 and 4-11, step-responses of the Altitude Select mode to rate-of-climb commands of 1 m/s have been shown for V_0 = 35 m/s and V_0 = 55 m/s, respectively. In the $\dot{H}$-plots, two different lines are shown: a solid line for the situation where the airspeed is kept constant by artificially specifying $\dot{V} = 0$, and a dashed line, which shows what will happen if the airspeed is allowed to change. The responses for $\dot{V} = 0$ are smooth and quite fast, with an overshoot that is negligible for both airspeeds. However, if the airspeed is allowed to change, that is if $\dot{V} \neq 0$, the required climb rates can not be reached, because in that case, speed is being exchanged for altitude, which is possible for a limited time only. Naturally, the response for low airspeed is worse than the response for higher airspeeds, but for V_0 = 55 m/s, the required rate-of-climb is not reached within 60 seconds either.

Note: the sudden *increase* in velocity and rate-of-climb which can be seen in the $\dot{H}$-plot from figure 4-10 is probably caused by numerical instability, rather than by an actual physical phenomenon. Also, the aerodynamic model of the 'Beaver', used for the simulations, is not accurate for the very low airspeeds which lay in the stall-region of the aircraft. It can be concluded that the Altitude Select mode works well if the airspeed is kept as constant as possible. In that case, the maximum climb-performance of the aircraft is limited by the engine power only.

In the real 'Beaver' autopilot, the ALS mode was engaged if the pilot entered a reference value for the altitude which differed more than a couple of metres from the current altitude. A switch-criterion was used to decide when to switch back from ALS to ALH, depending upon the rate of climb and the altitude. Unfortunately, the rate-of-climb was computed in the airdata computer by using the measured values of H, yielding a signal which was quite noisy and which had an additional time-delay of 0.8 seconds. It is therefore not unlikely that the controller has switched from ALS to ALH too soon in practice. In the future, more information about the character of the signals and data manipulations in the Airdata Computer need to be known in advance, before the control laws are implemented in the real aircraft.

4.3 Asymmetrical autopilot loops.

4.3.1 Roll Attitude Hold.

a. Step responses.

Figures 4-12a and 4-12b show step responses of the Roll Attitude Hold mode to commanded roll angles of 10 degrees for V_0 = 35 m/s and V_0 = 55 m/s, respectively (the results for negative roll angles have been mirrored along the time-axis, for comparison). For low airspeed, the maximum overshoot is approximately 20% for a right turn, and somewhat smaller for a turn to the left. This asymmetrical behaviour is caused by the influence of the slipstream behind the propeller, which is strongest for low airspeed (the responses to

positive and negative reference roll angles for 55 m/s are almost equal). Compared with the linear results from ref.[16], the sideslip angle β becomes quite large during the first few seconds after the turn is initiated. This difference is caused mainly by a different use of reference frames: in ref.[16], the turncoordination loops have been designed using variables expressed in the *stability-reference frame*, whereas the nonlinear aircraft model uses *body-axes*. This induces errors in the yaw rate and roll rate, the relationships between the values in stability axes and body axes being:

$$\begin{aligned} p_S &= p_B \cdot \cos\alpha_0 + r_B \cdot \sin\alpha_0 \\ r_S &= -p_B \cdot \sin\alpha_0 + r_B \cdot \cos\alpha_0 \end{aligned} \qquad (4\text{-}1)$$

where the index B denotes body-axes variables, and S denotes variables, defined with respect to the stability reference frame. In figures 4-13a and 4-13b, it is shown how the responses are changed if the yaw-rate r_B, which is used by the turncoordination loops, is replaced by r_S, to reproduce the situation from ref.[16]. It can be seen that the initial increase in β is considerably smaller, although the overshoot in φ for V = 35 m/s has increased from 20% to 22%. Because of the smaller sideslip angle, the responses from figures 4-13a and 4-13b are preferred over the responses from figures 4-12a and 4-12b.

In the real aircraft, the body-axes roll-rate and yaw-rate are measured too, but α_0 could not be measured accurately. Also the data paths leaved no room for the measurement of α_0 due to hardware limitations. For this reason, an approximation of α_0, based upon trimmed-flight conditions, given in ref.[14] has been implemented in the FCC software. Of course, this solution is not 100% perfect, but at least, it will considerably reduce the difference between the measured body-axes yaw-rate and the stability-axes yaw-rate.

The turncoordination does not succeed in eliminating the sideslip angle completely. For low airspeed, β remains negative, and for high airspeed, β remains positive. If the correction for α_0 is made, the remaining sideslip angle for V_0 = 35 m/s becomes almost zero in the turn; for V_0 = 55 m/s the remaining sideslip angle still stays positive. This was not noticed in the linear simulations from ref.[16], however, it is mainly a matter of fine-tuning the turncompensation gains. The controller does initiate the turns by applying deflections of rudder and ailerons into the same direction and it uses aileron and rudder deflections of opposite signs in the turns themselves to maintain the reference roll angle. This is in accordance the theory for coordinated turns, see refs.[7] and [11]. The largest crossing of the controls in turns is needed for low airspeed.

b. Responses to atmospheric turbulence.

In figure 4-4, responses of the 'Beaver', equipped with Pitch and Roll Attitude control laws has been shown. In section 4.2.1.b, the symmetrical responses were considered; here the performance of the Roll Attitude Hold mode in atmospheric turbulence will be evaluated. Figure 4-4 shows that the roll angle φ varies less than in the open-loop situation, although the differences may seem somewhat disappointing at first sight. However, in the plot of the resul-

ting responses of the yaw-angle ψ, it is clearly visible that the aircraft maintains its *heading* better than without RAH. Apparently, the RAH succeeds well in removing the slower components of the φ-response which directly affect the aircraft's heading.

c. Tuning limiters.

The $\varphi - \varphi_0$ limiter, which is visible in the blockdiagram from figure C-25, has been tuned by means of nonlinear simulations, as shown in figure 4-14. According to table 2-2, the roll rate p may not exceed 12 deg/s, because otherwise the autopilot will be turned off automatically by the mode controller for reasons of safety. Different simulations for a step $\Delta\varphi = 30°$ and a number of limiter-settings ($\leq$10 Volt) were compared to eachother, yielding the plots from figure 4-14. In this figure, the solid line represents the step response for the resulting limiter setting, which yields a maximum roll rate of 10 deg/s. Sufficient room to the mode controller margin of 12 deg/s is left to ensure that the autopilot will not be switched off automatically.

4.3.2 Heading Hold/Heading Select.

a. Step responses.

Step responses for the Heading Hold/ Heading Select mode to reference inputs $\Delta\psi_{ref} = 10°$ are shown in figures 4-15 and 4-16 for $V_0 = 35$ m/s and $V_0 = 55$ m/s, respectively. In both cases, the response is quite fast (the reference value is almost reached within 30 seconds) and the overshoot is kept within reasonable limits. The roll-angle, used to realize this change in yaw-angle is larger for high airspeeds. Also, the initial changes in sideslip angle, caused by the Dutch-roll characteristics of the 'Beaver', are quite large. As shown in section 2.3.1a, this large initial value of β was caused mainly because of the difference of the design reference frame and the reference frame, used for the nonlinear aircraft models and the real aircraft. This yielded wrong values for the yaw rate r, used for turncoordination. Correcting this yaw-rate for α_0 reduced the maximum sideslip angle considerably. Since it was not possible to measure α_0 accurately in the real aircraft, an approximation of α_0 has been included in the software for the Flight Control Computer.

For larger steps in yaw-angle, the signal $\Delta\varphi_{ref}$, which is sent from the outer-loops of the HH mode to the RAH inner-loops, is limited to 30 degrees. In figures 4-17 and 4-18, responses to steps $\Delta\psi_{ref} = 90$ degrees for $V_0 = 35$ m/s and $V_0 = 55$ m/s have been shown (the results for $\Delta\psi_{ref} = -90°$ have been multiplied with -1 for comparison). In these figures, the influence of the $\Delta\varphi_{ref}$-limiter is clearly visible in the φ-responses; the influence of the $\varphi - \varphi_0$ limiter can be seen in the responses of p. The turn is initiated by first applying deflections of ailerons and rudder into the same direction, then, during a couple of seconds, the ailerons have to be deflected in opposite direction, and finally, the aircraft levels off by applying a large aileron and rudder deflection into the same direction. This is exactly what is expected for coordinated turns [7], [11].

The sideslip angle is quite large, but again, this is mainly due to the absence of an α_0-correction in r. The large values of β for low airspeed are caused by a drop in airspeed, which strongly affects the slipstream of the aircraft.

For low airspeeds, the overshoot in ψ is quite large, especially for turns to the left. The slow oscillation with a period of about 25 seconds is obviously caused by the influence of the phugoid, which is visible in the lateral responses due to the longitudinal-lateral cross-coupling. The main cause for these effects is again the influence of the propeller slipstream, which changes during the motion due to a large drop in airspeed. The responses which were obtained when the stability derivatives to *dpt* were set to zero were a whole lot better. Better results are also obtained when the maximum roll angle is limited to 20 °, as can be seen in figure 4-19. The drop in airspeed in this case was almost 1 m/s smaller, which yields smaller slipstream effects. Also, the aircraft levels off more quickly because of the smaller roll angle.

Unfortunately, this change in maximum roll-angle has never actually been implemented in the software for the Flight Control Computers of the aircraft, but flight tests have indeed shown similar large values of the ψ-overshoot for low airspeeds (see ref.[17]). In figures 4-17 to 4-19, it can be seen that the response is very fast. For V_0 = 55 m/s, the change in heading of 90° is reached within 20 seconds! In real flight, this turning behaviour turned out to be not only fast, but also quite uncomfortable. In the future, piloted simulations in a moving-base real-time flightsimulator are obviously needed too, in order to fine-tune the control laws. Also, handling qualities criteria need to be used to evaluate control behaviour of the aircraft.

b. Responses to atmospheric turbulence and wind.

As can be seen in figure 4-9, the Heading Hold controller clearly succeeds in maintaining a constant yaw angle under atmospheric disturbances. The results do not differ very much from the results for RAH control. In both cases, the control deflections are quite fast (especially δ_r), but their magnitude remains within reasonable limits.

4.4 Turncompensation.

The structure of the turncompensation laws has been derived in section 2.4, which resulted in a control law to determine a *correction* of the pitch-rate q according to:

$$q := q - (r\tan\varphi + K_{tc} \cdot (\sec\varphi - 1)) \qquad (4\text{-}2)$$

The gain K_{tc} needed to be determined with nonlinear simulations because of the nonlinear character of the turncompensation law and because the linear models, used in ref.[16] for the determination of the control laws, did not include longitudinal-lateral cross-coupling effects. Separate laws were determined for the Pitch Attitude Hold and Altitude Hold control laws. In PAH mode, the pitch angle must be kept constant, which means that θ may not decrease due to the loss of lift in turns, but θ may not increase for a compensation of the loss of altitude either. In Altitude Hold mode, the pitch angle may increase as long as the altitude is kept constant.

Figure 4-20 shows the θ-responses to roll-angle commands of plus and minus 30 degrees, for V_0 = 35 m/s. The aircraft stays turning at this roll-angle. A clear drop in θ after initiating the turn is visible. The variations of the pitch-

angle are somewhat larger for turns to the right ($\Delta\varphi_{ref} = 30°$) than for left turns, which is again caused by the influence of the engine. Here, the best results for left *and* right turns are obtained for $K_{tc} = 0.5 * 180/\pi$, where the term $180/\pi$ is needed because, unlike the nonlinear aircraft model, the FCC software measures all angles (including the control surface deflections) in degrees in stead of radians. In the same way, turncompensation gains for higher airspeeds were obtained, which yielded the gain-scheduling from table 2-1.

For ALH, *H*-responses to large roll-angle commands for RAH were computed in the same way for different values of the turncompensation gains. Here, it was found that for low airspeed, it was not possible to keep the altitude constant without increasing the engine power. Therefore, the turncompensation gain for $V = 35$ m/s was determined by minimizing the *fast* changes in altitude which occurred just after the turn was initiated. This again gives an indication that the maximum roll angle of 30 degrees is actually too large for low initial airspeed.

In order to get a better agreement between the altitude responses to left and right turns, a small offset $\Delta\varphi_{tc} = 2°$ has been added to the roll-angle, used in the secans expression for ALH turncompensation. This little trick indeed helped to decrease the differences in altitude responses for left and right turns, although it increased the differences in pitch-angle responses to left and right turns. This characteristics were also noticed the other way around: if the v-responses in PAH mode with turncompensation were matched better for left and right turns by using an offset in φ, the altitude responses to left and right turns in PAH mode started to differ more. The Altitude Hold mode works fully automatic, so the difference in pitch-angle responses were tolerated in favour for the better altitude responses, but since the PAH mode is usually controlled manually by means of a side-stick, the offset $\Delta\varphi_{tc}$ has not been applied for the latter mode. The resulting turncompensation for ALH becomes:

$$q := q - (r\tan\varphi + K_{tc} \cdot (\sec(\varphi + \Delta\varphi_{tc}) - 1)) \qquad (4\text{-}3)$$

Compared to ref.[9], the turncoordination has been optimized a lot, because here, the gain K_{tc} has actually been determined by analyzing a large number of simulations, whereas ref.[9] simply used $K_{tc} = 1$. Of course, this only gives good results if the longitudinal-lateral cross-coupling has been modelled correctly. According to ref.[17], the results in real flight were as expected (apart from an initially undiscovered typing error which yielded a value of K_{tc} which differed by a factor 10 from the desired value).

4.5 Conclusions.

The Pitch Attitude Hold, Altitude Hold, Altitude Select, Roll Attitude Hold, and Heading Hold modes all have acceptable performance. The main problem is the control of the sideslip angle, which tends to increase a lot as airspeed drops, especially for low initial airspeeds. The turncoordination works good, yielding control deflections of opposite signs within turns, which is in accordance with the theory (see for instance refs.[7] and [11]). However, since the control laws were designed in the stability axes (using stability-axes angular velocities), the results obtained here are not as good as in ref.[16], unless the angular

velocities p and r are corrected by rotating the axes over an angle α_0. In the real aircraft, an approximation of α_0 has been made by using trimmed-flight data from ref.[14]. Finally, it has been found that a smaller maximum roll-angle for low velocities in Heading Select gives better results, and that the turncoordination laws give satisfying results. The gains for turncompensation had to be tuned with nonlinear simulations.

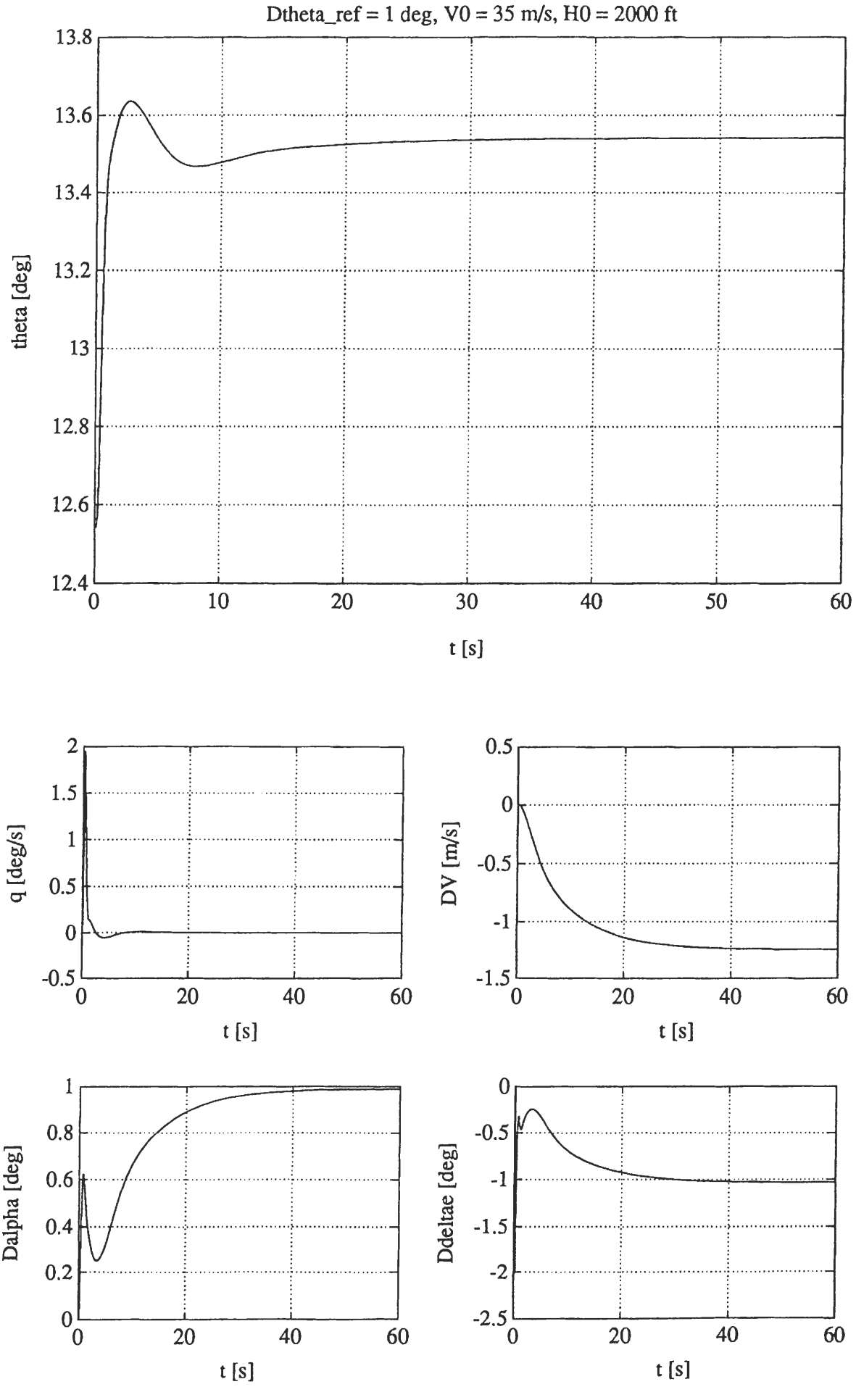


Figure 4-1. Step response of Pitch Attitude Hold mode for V_0 = 35 m/s.

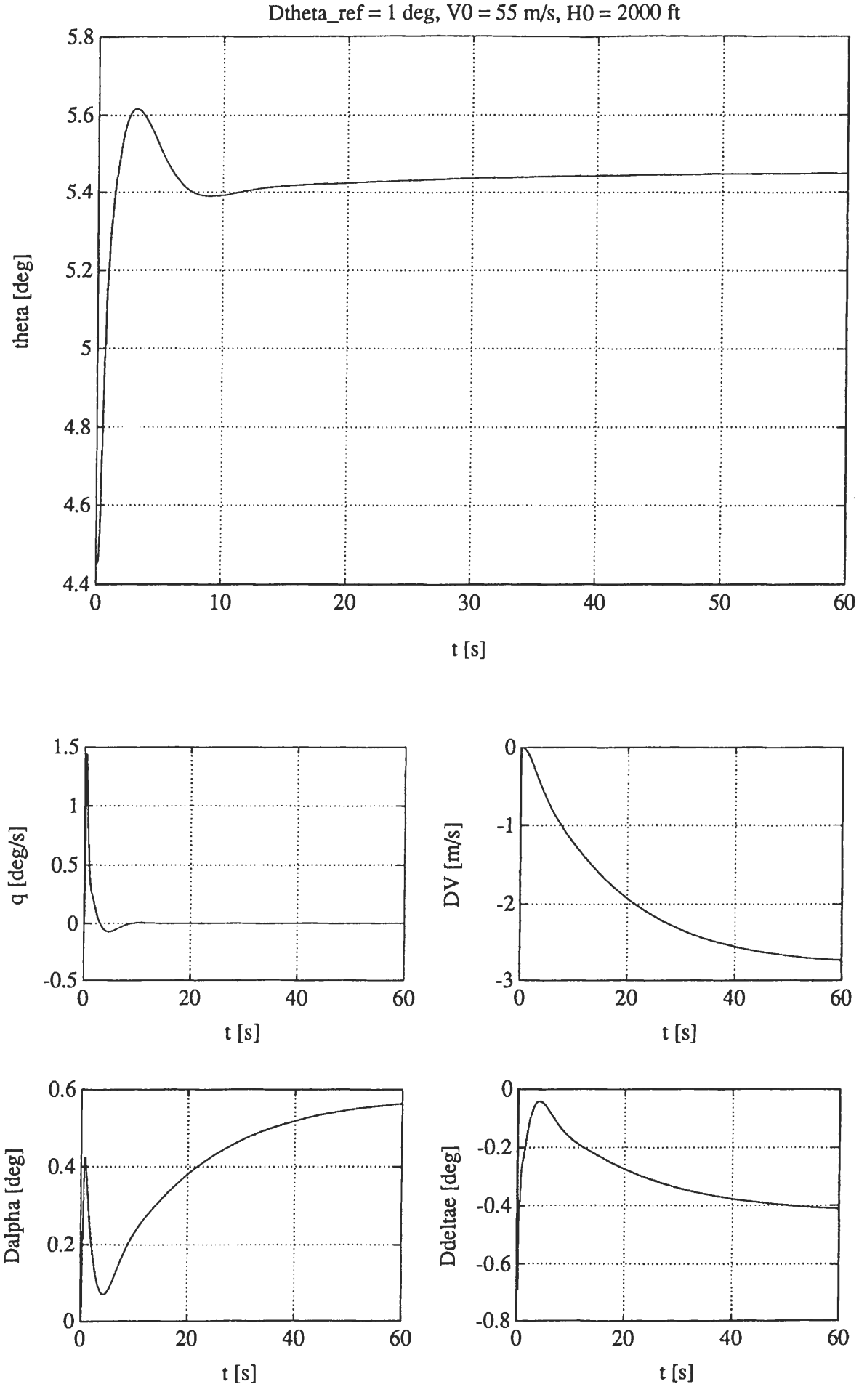


Figure 4-2. Step response of Pitch Attitude Hold mode for V_0 = 55 m/s.

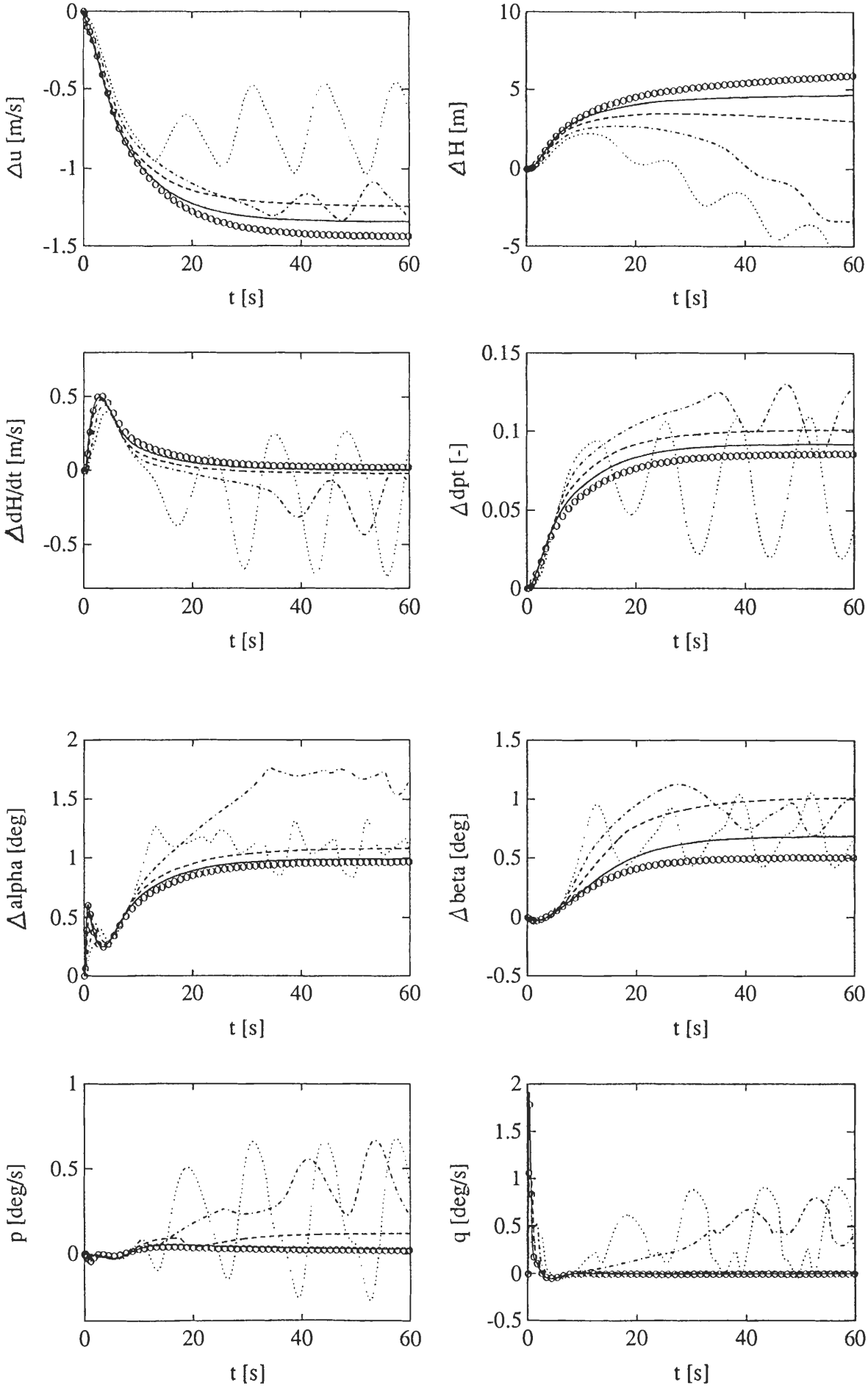


Figure 4-3. Step responses of PAH mode, scaled back to unity-step (asymmetrical mode: RAH). V_0 = 35 m/s, H_0 = 2000 ft (contd. on next page).

——— : $\Delta\theta_{ref}$ = 1°, - - - : $\Delta\theta_{ref}$ = 4°, – ·– ·– : $\Delta\theta_{ref}$ = 8°, ·····: $\Delta\theta_{ref}$ = 12°, ∘ ∘ ∘ ∘ : $\Delta\theta_{ref}$ = -1°

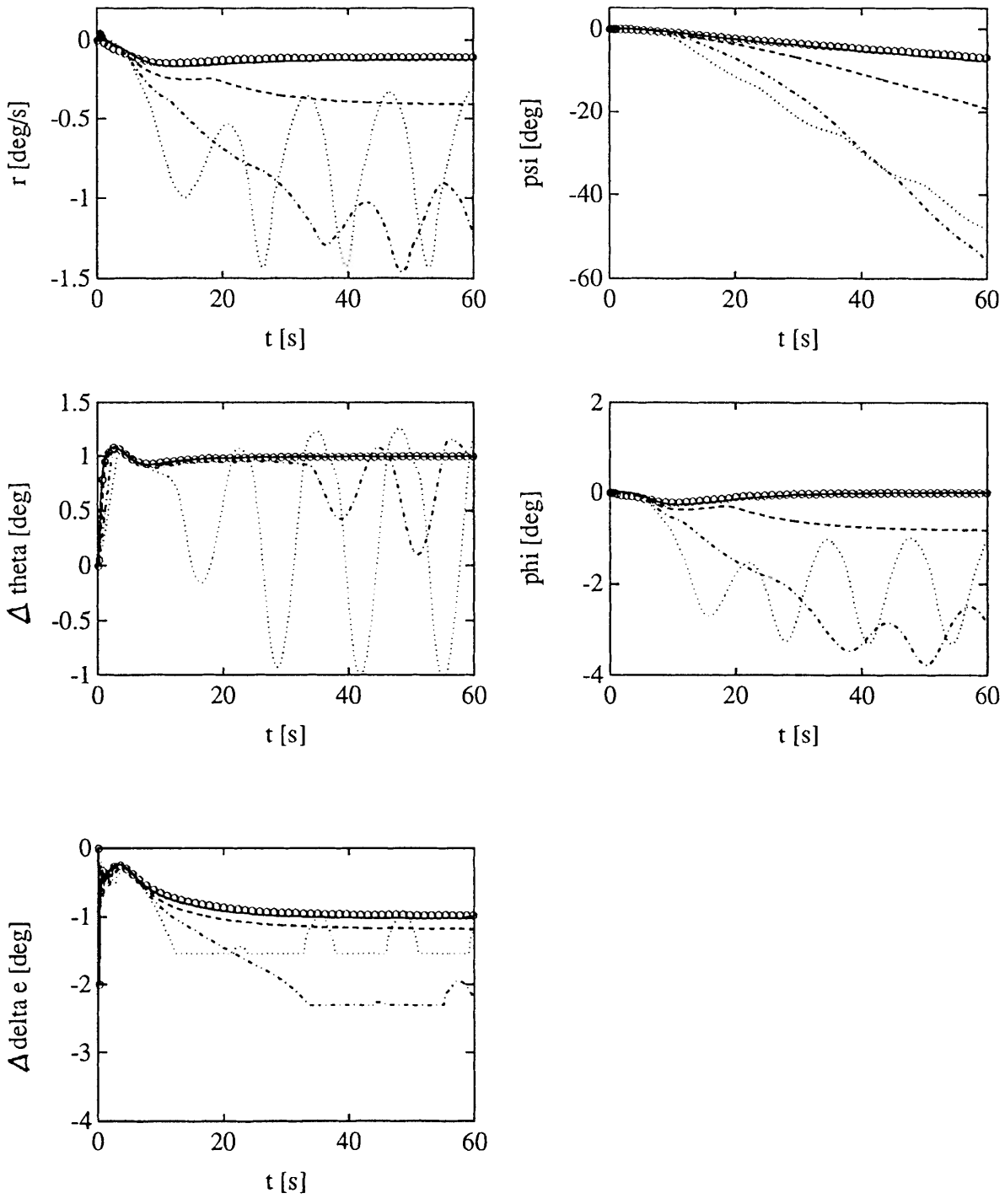


Figure 4-3 (continued).

——— : $\Delta\theta_{ref}$ = 1°, - - - : $\Delta\theta_{ref}$ = 4°, – . – .–: $\Delta\theta_{ref}$ = 8°, .·.·.·.·: $\Delta\theta_{ref}$ = 12°, ∘ ∘ ∘ ∘ : $\Delta\theta_{ref}$ = -1°

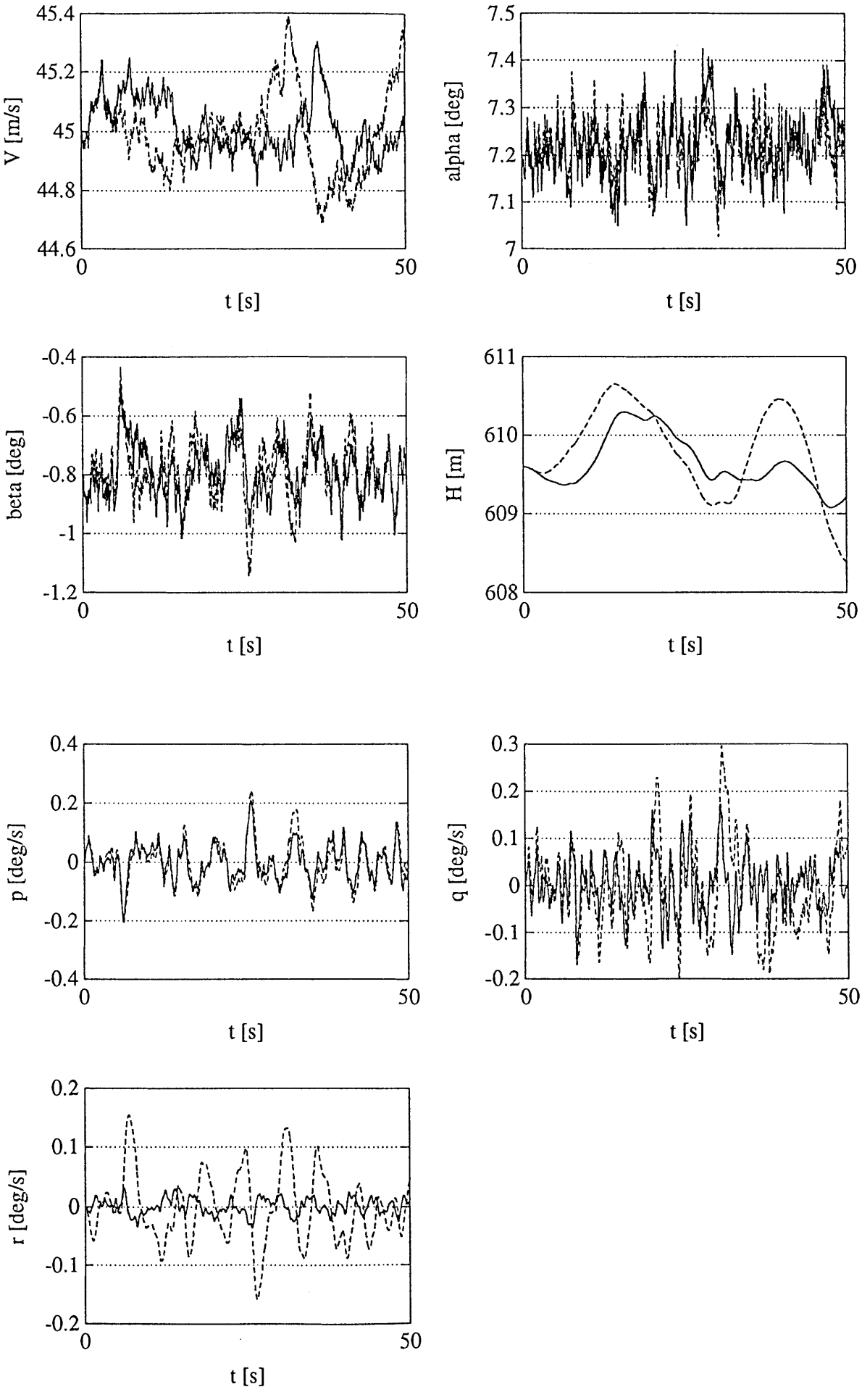


Figure 4-4. Responses to atmospheric turbulence for open-loop situation (dashed lines) and with PAH/RAH control enabled (solid lines)

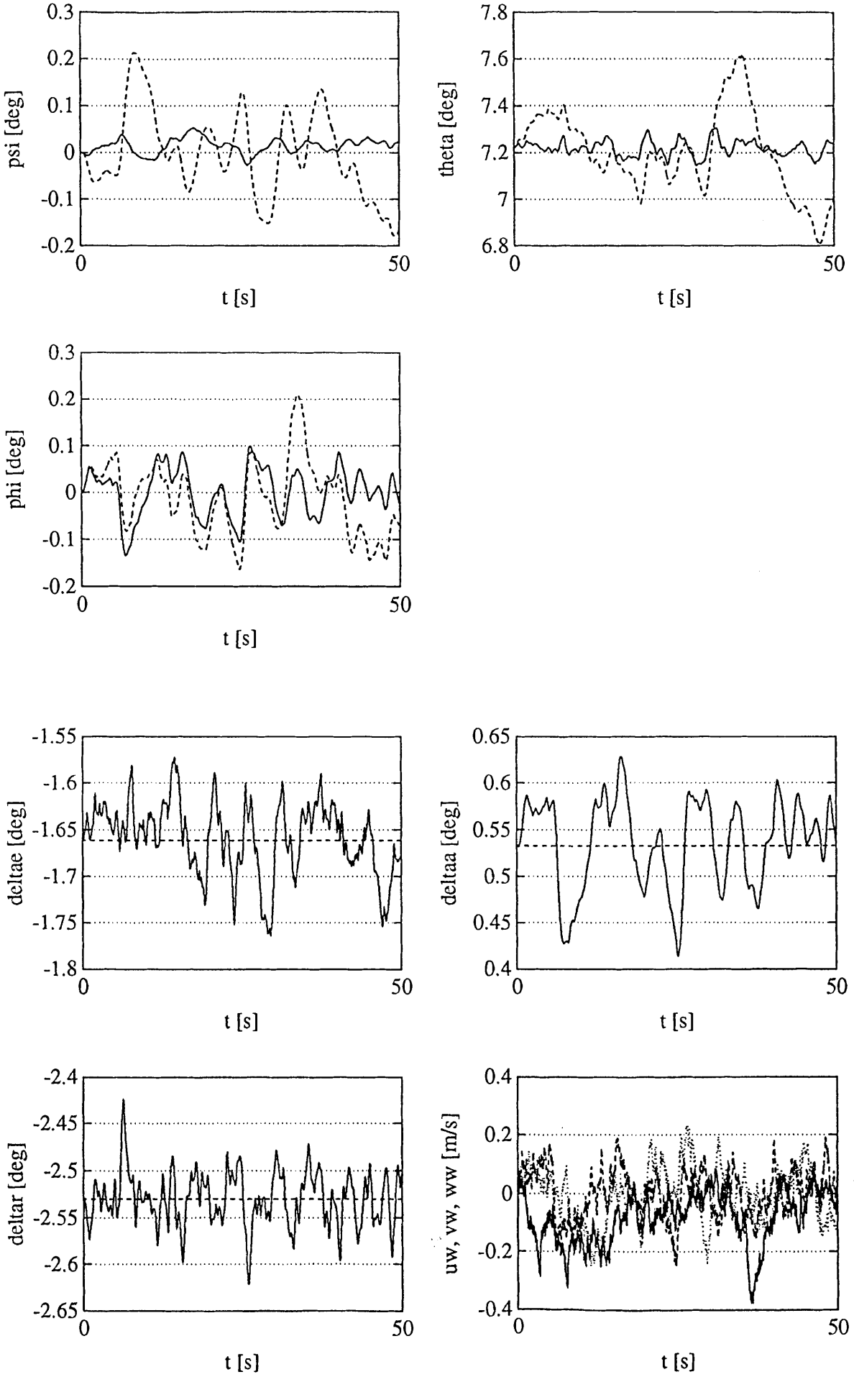


Figure 4-4 (continued). Open-loop responses: dashed lines. Closed-loop responses (PAH/RAH): solid lines.

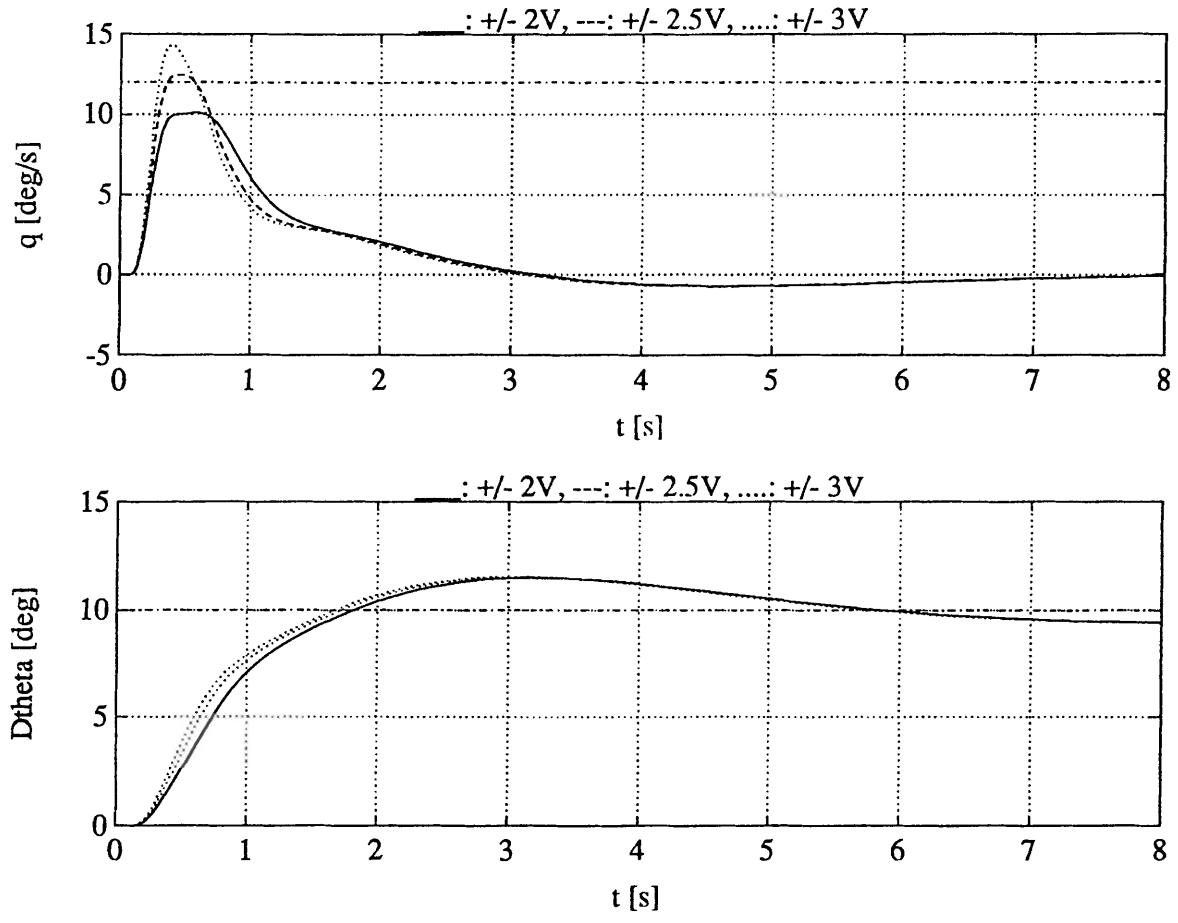


Figure 4-5. Step responses of PAH mode, $\Delta\theta_{ref}$ = 10°, V_0 = 55 m/s, for different settings of the $\theta - \theta_0$ limiters.

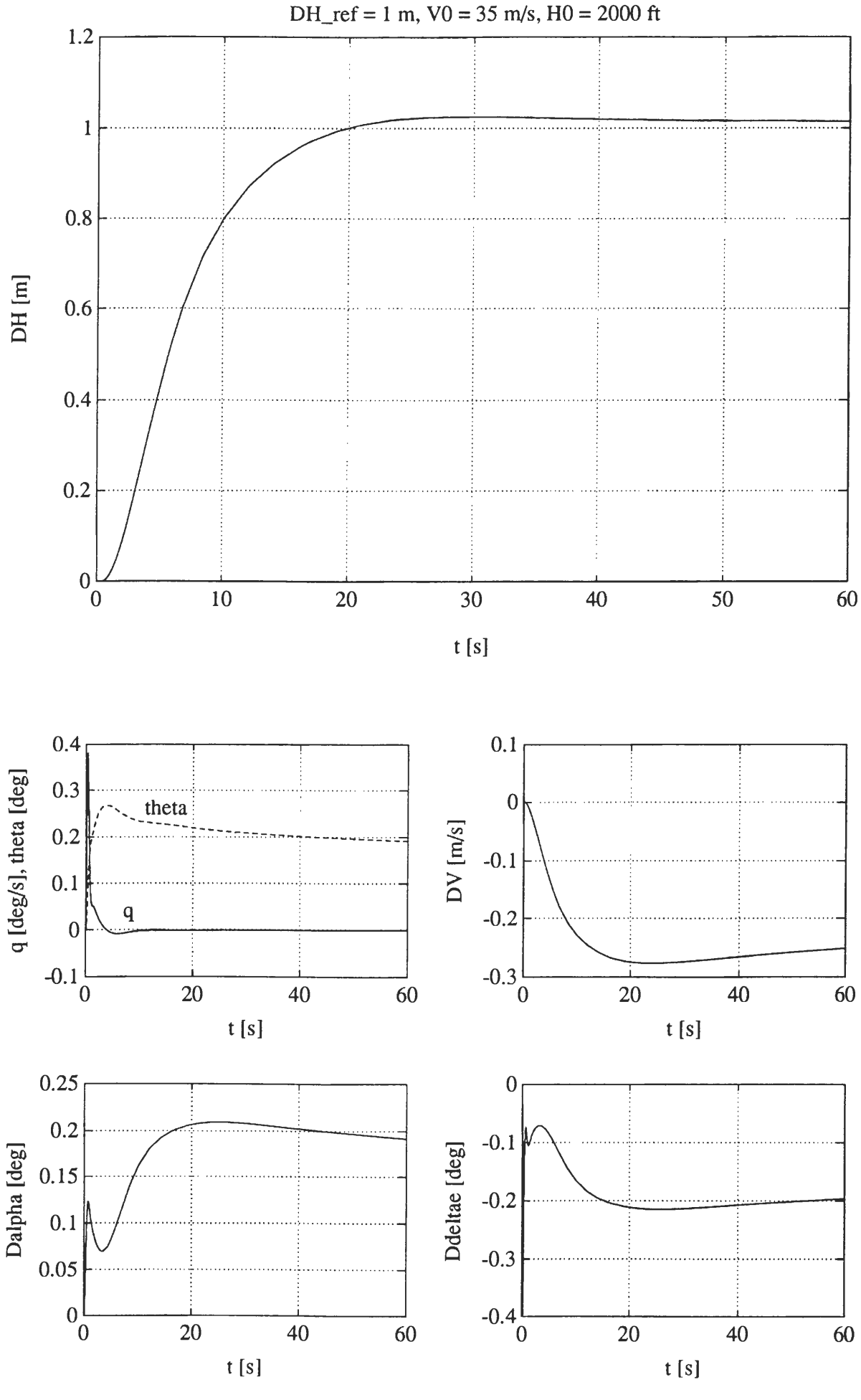


Figure 4-6. Step responses of ALH mode, ΔH_{ref} = 1 m, V_0 = 35 m/s.

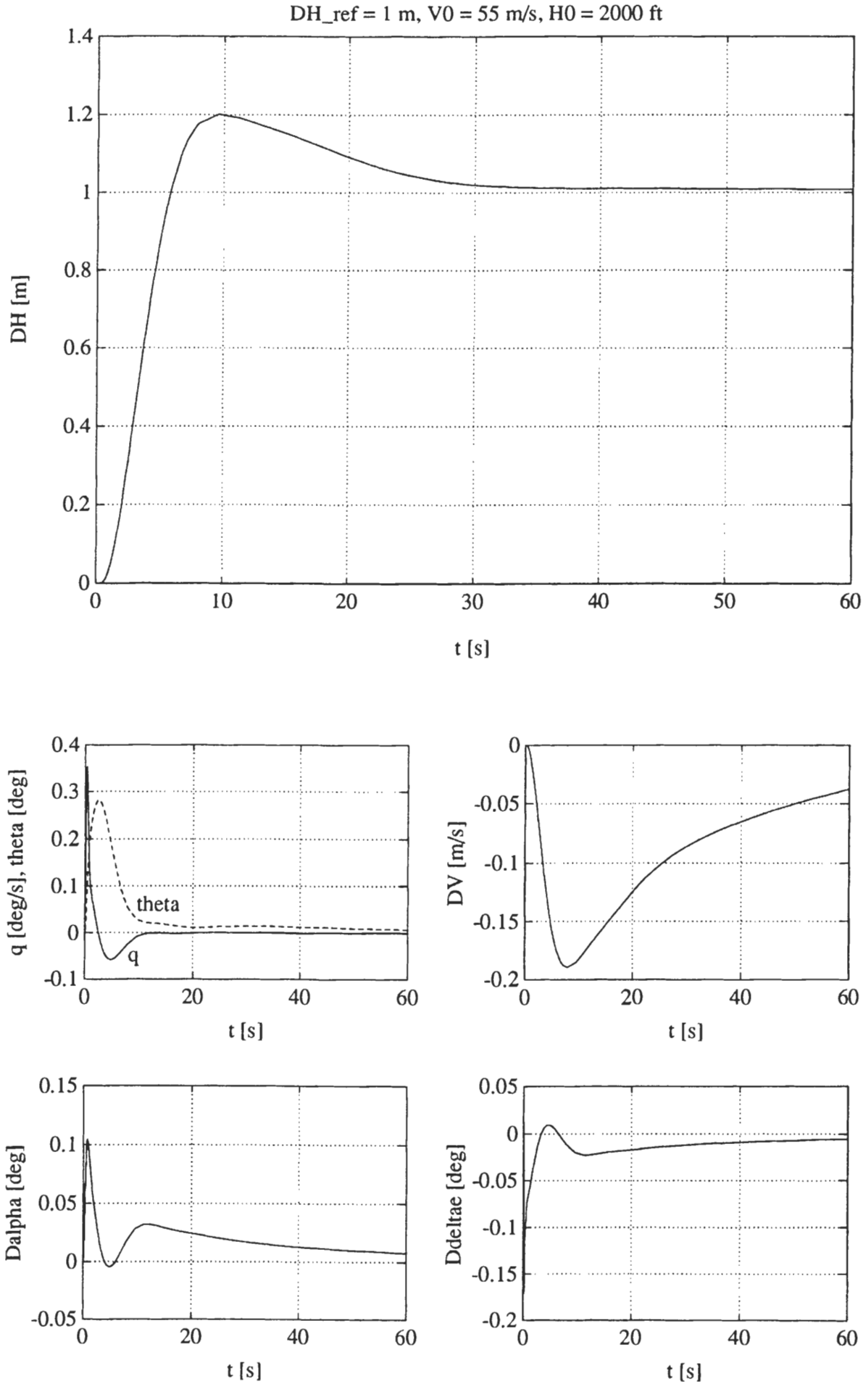


Figure 4-7. Step responses of ALH mode, ΔH_{ref} = 1 m, V_0 = 55 m/s.

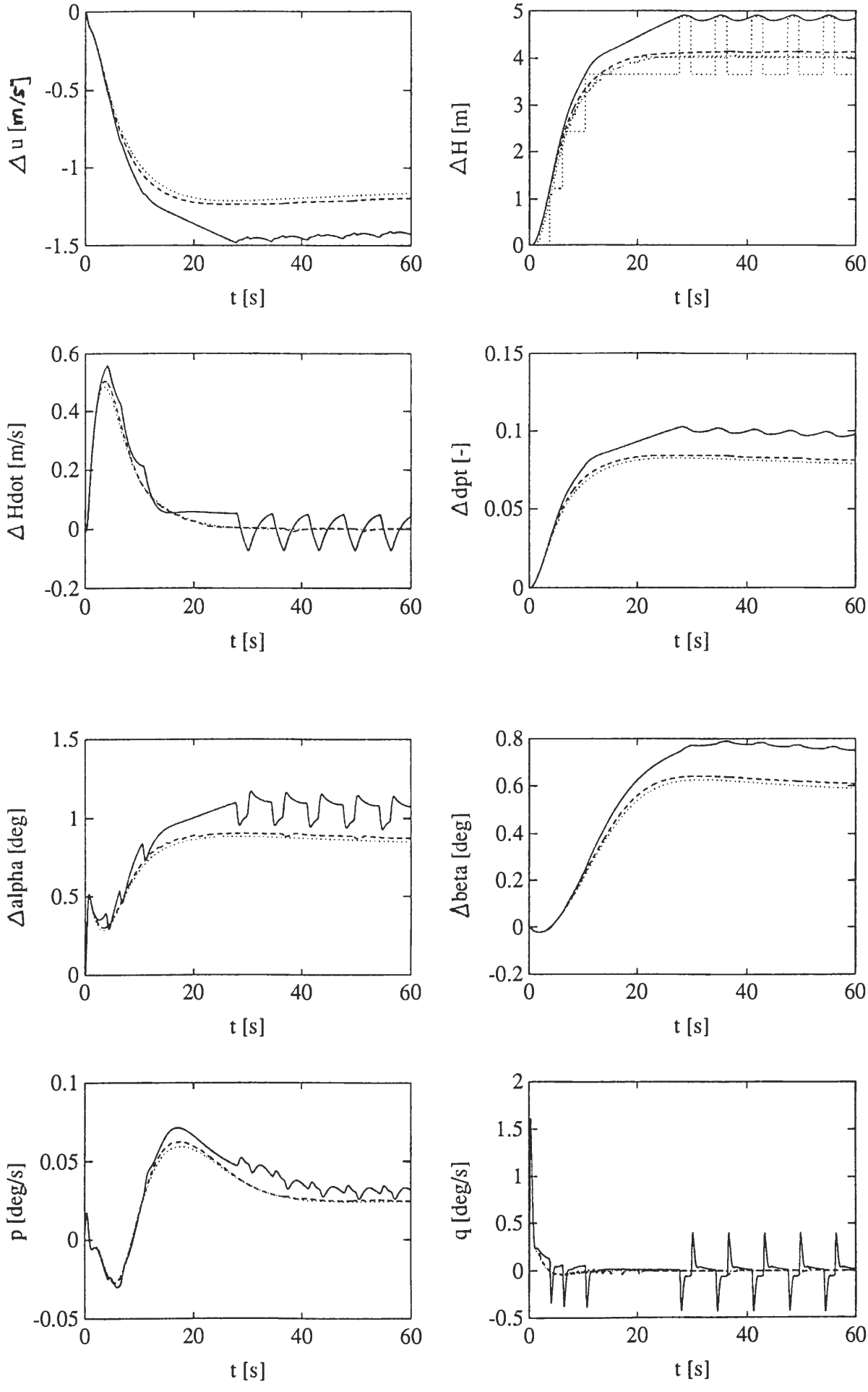


Figure 4-8. Step responses of ALH mode for continuous signal (·····), quantized signal (————), and signal, passing a Moving Average filter (– – – – – –). V_0 = 35 m/s, H_0 = 2000 ft.

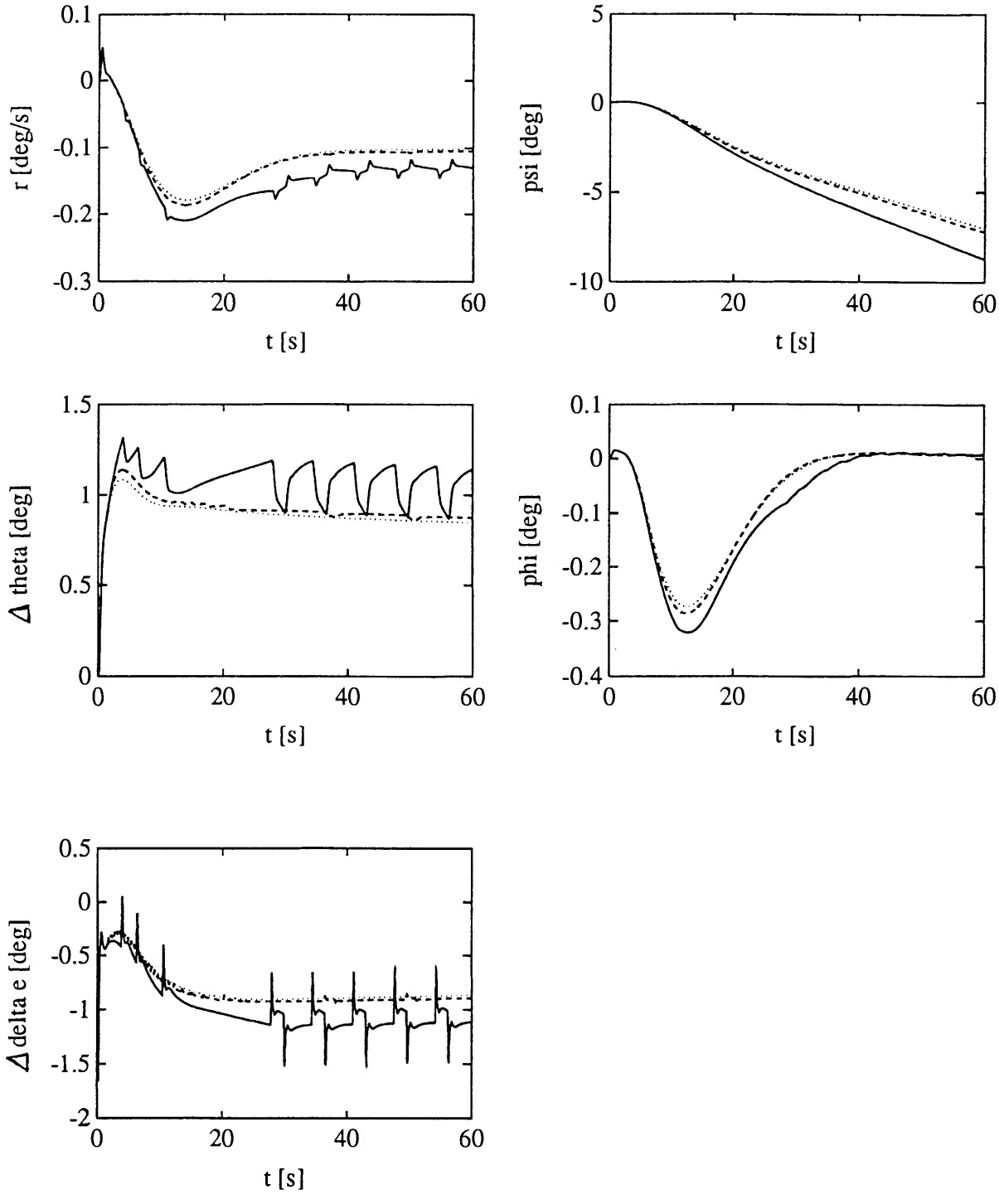


Figure 4-8 (continued).

· · · · · · : continuous signal, ——: quantized signal, – – –: signal through MA-filter

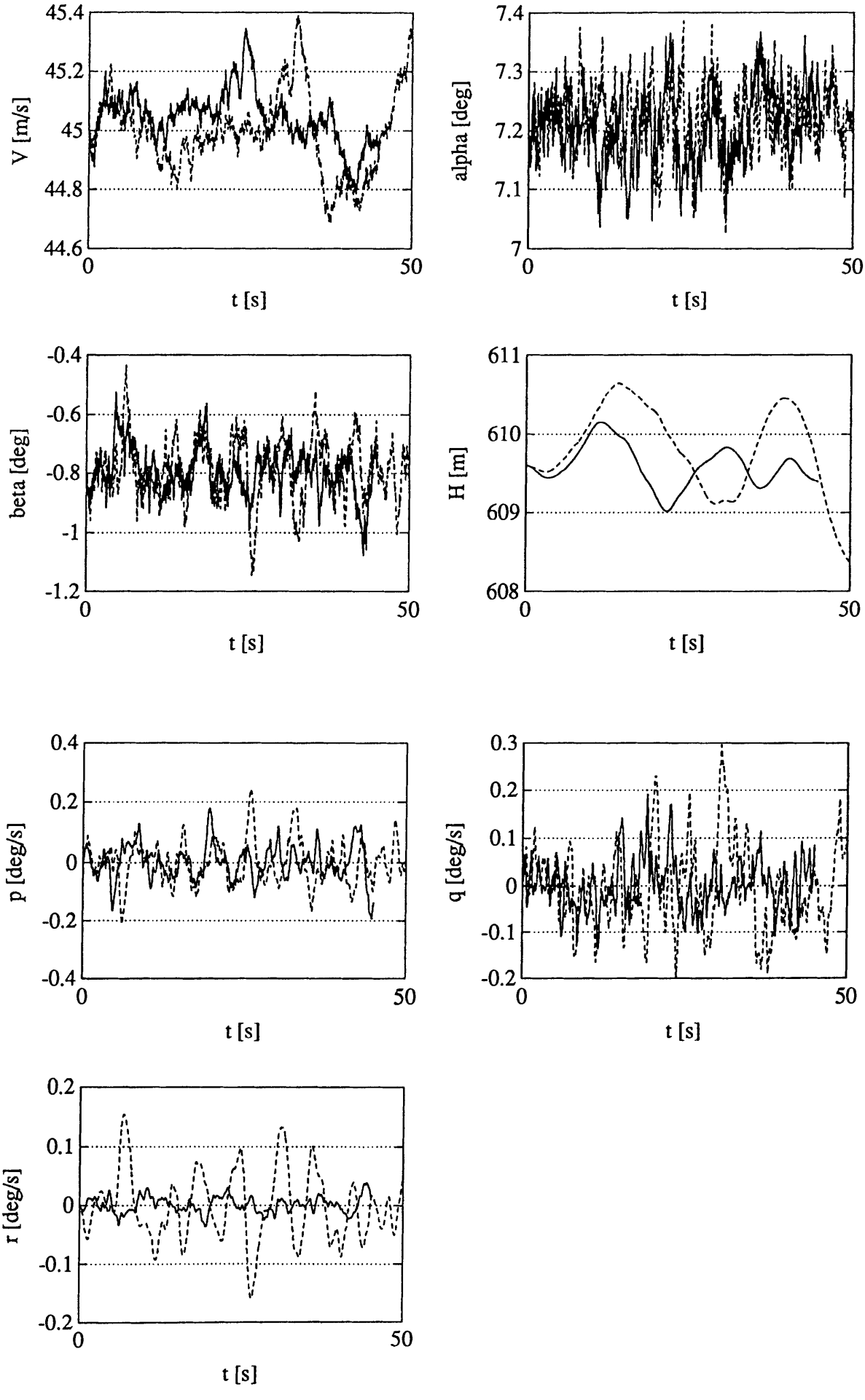


Figure 4-9. Responses to atmospheric turbulence for open-loop situation (dashed lines) and with ALH/HH controller enabled (solid lines).

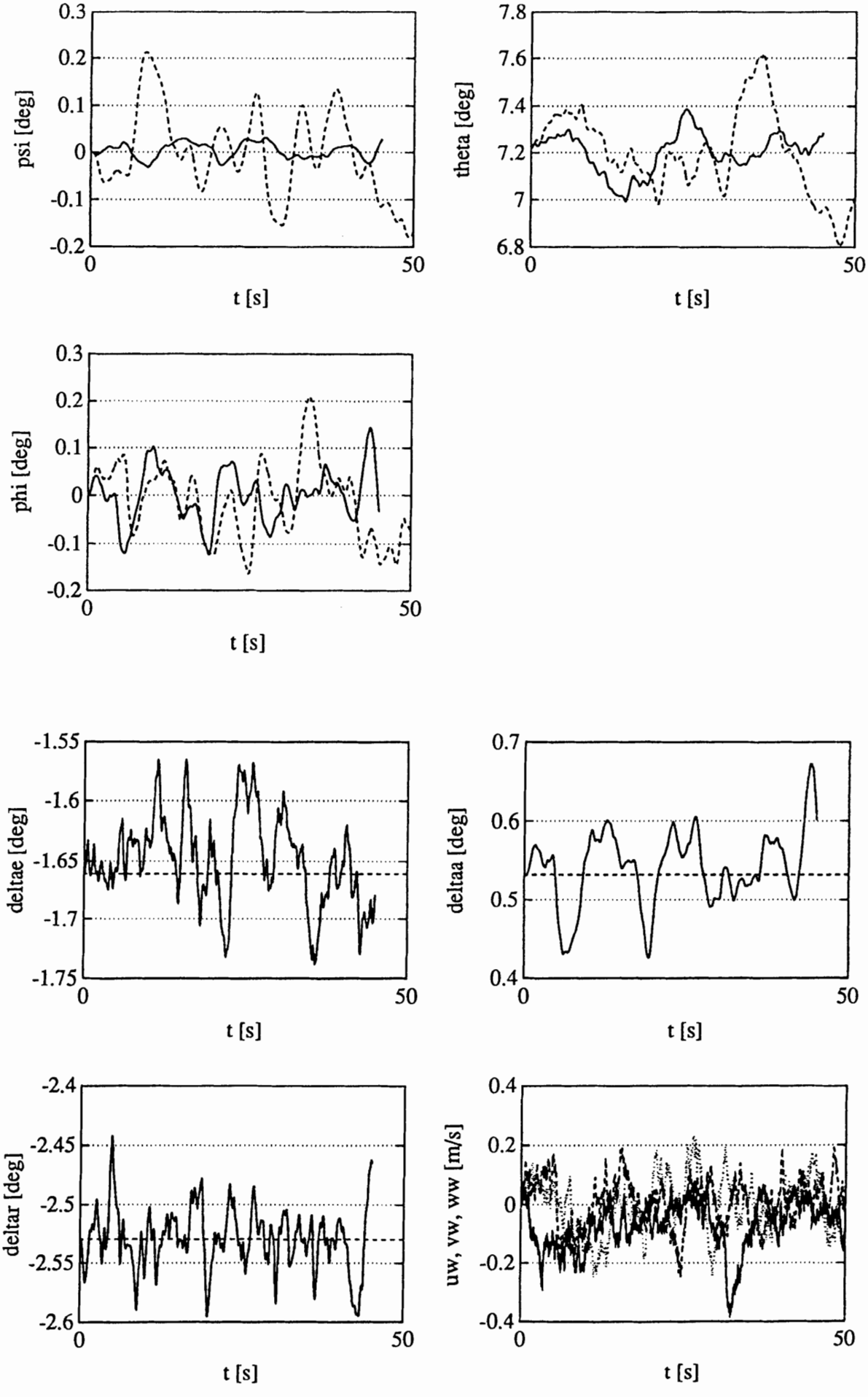


Figure 4-9 (continued). Open-loop responses: dashed lines. Closed-loop responses (ALH/HH): solid lines.

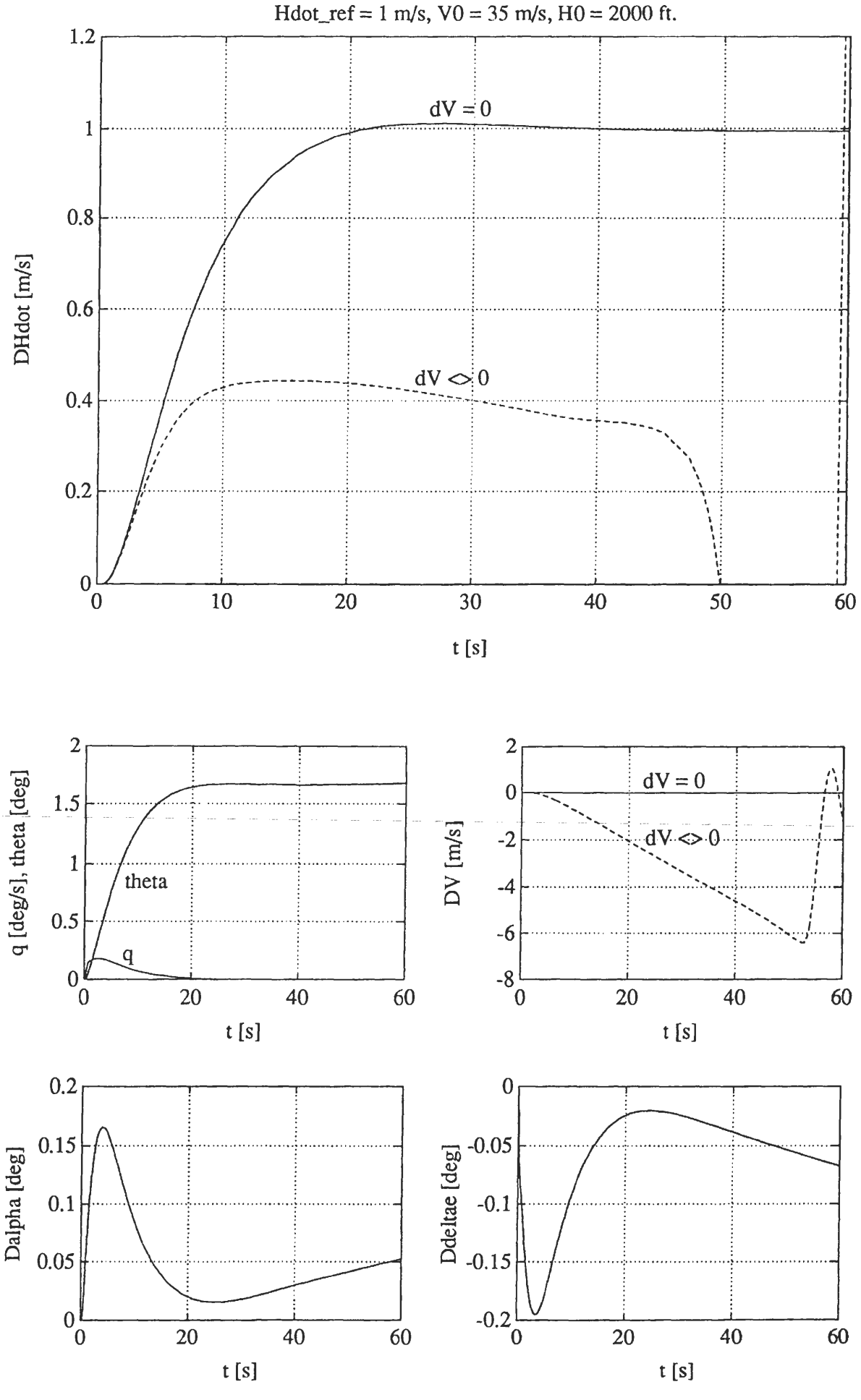


Figure 4-10. Step responses of ALS mode, $\dot{H}_{ref}$ = 1 m/s. Solid lines: dV = 0, dashed-lines: $dV \neq 0$. V_0 = 35 m/s.

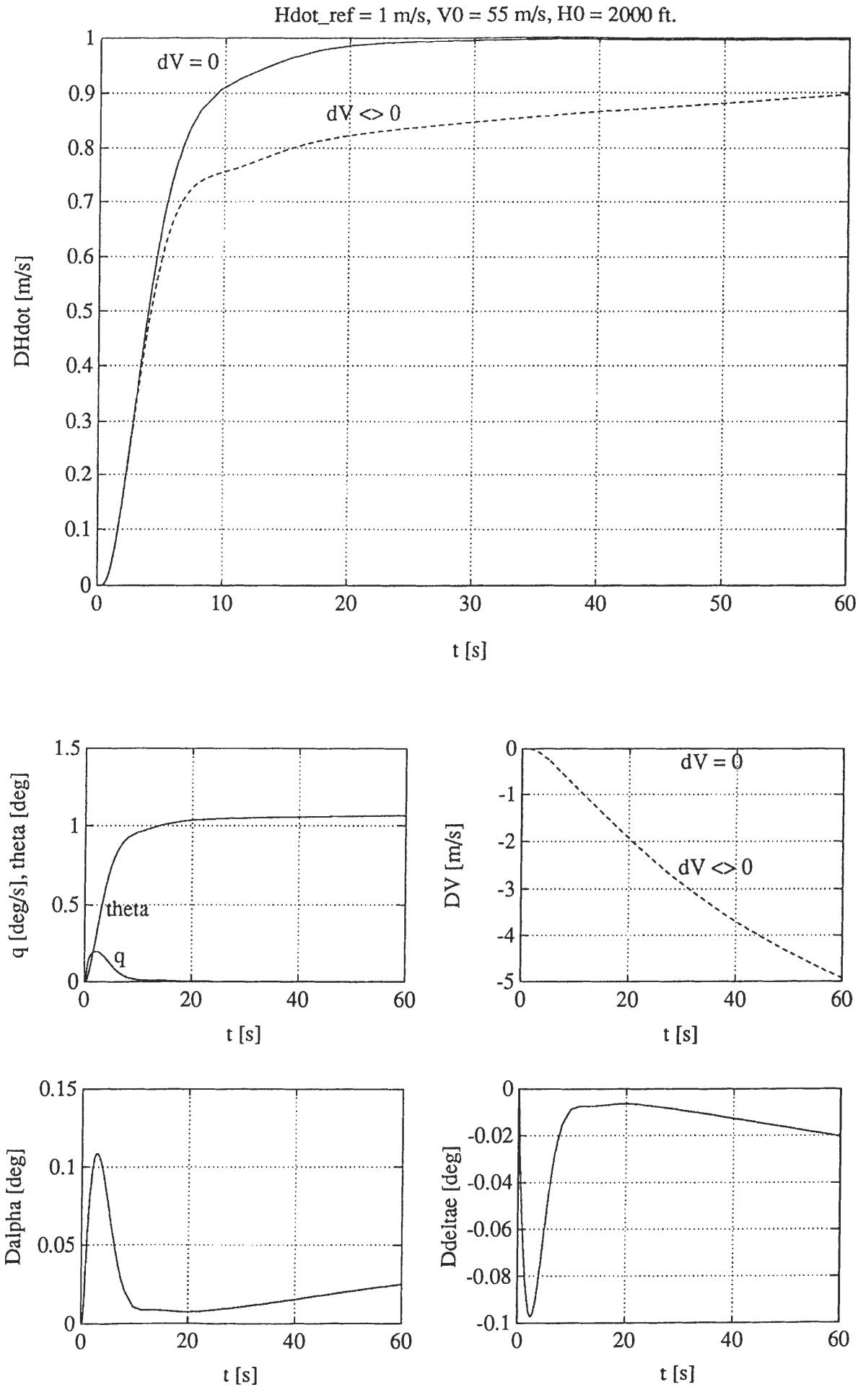
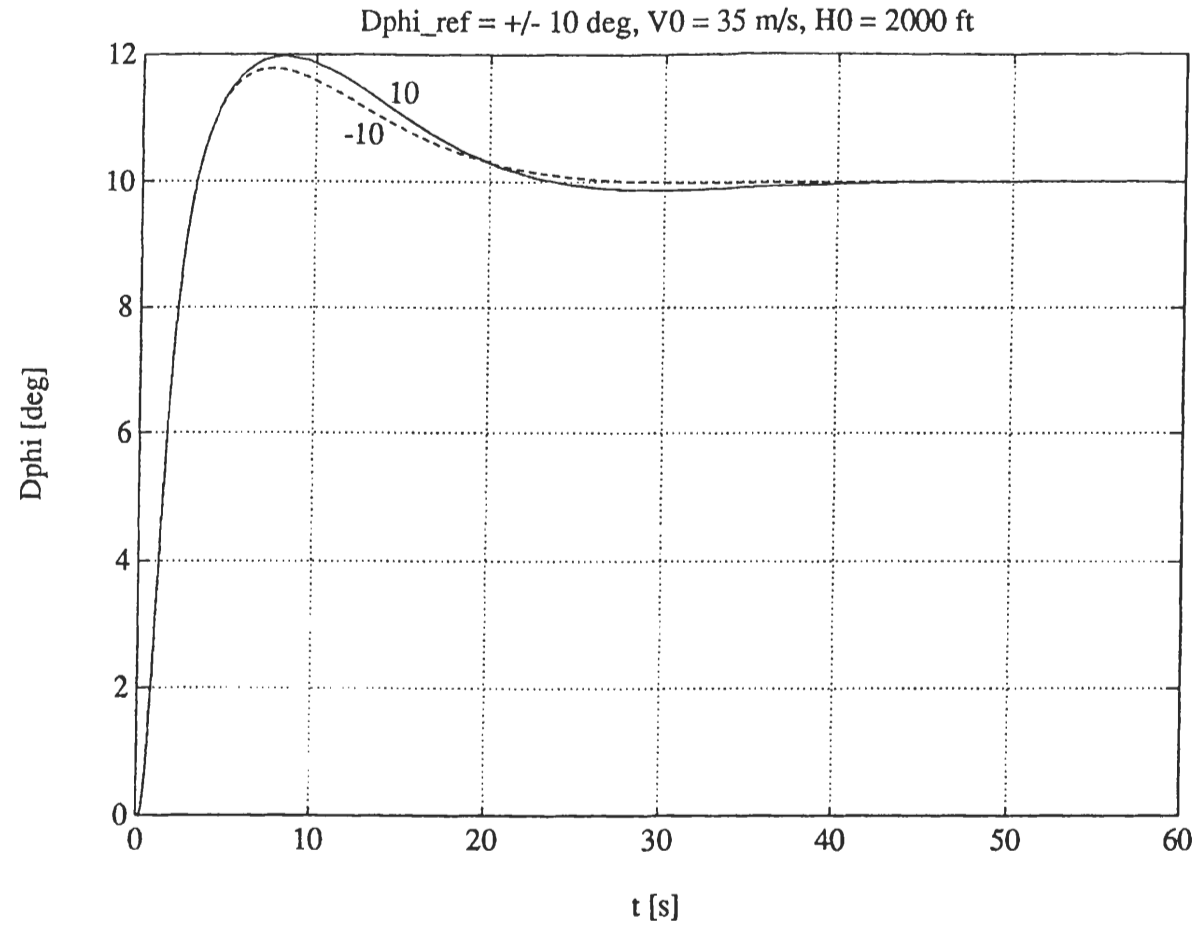
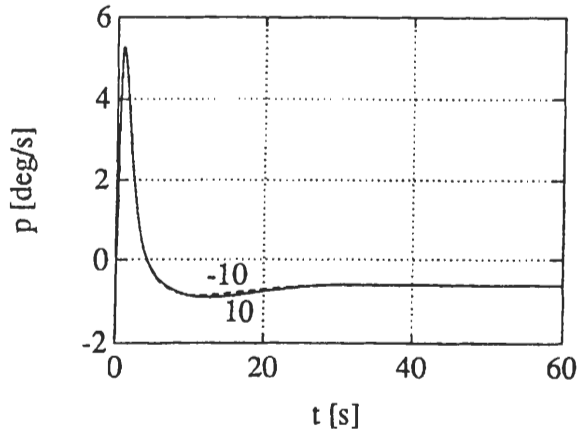
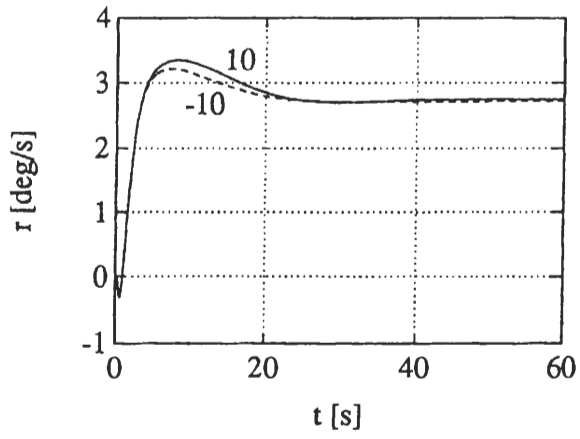
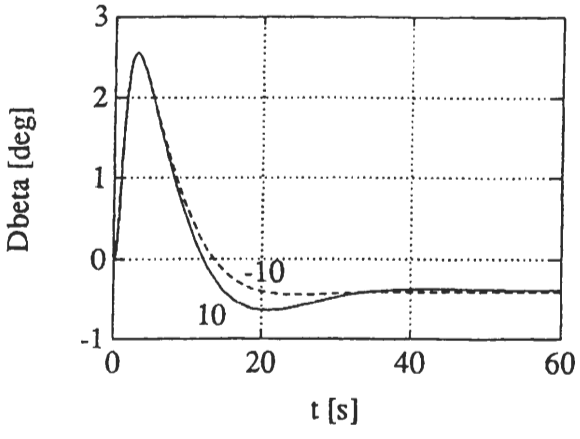


Figure 4-11. Step responses of ALS mode, $\dot{H}_{ref}$ = 1 m/s. Solid lines: dV = 0, dashed-lines: $dV \neq 0$. V_0 = 55 m/s.









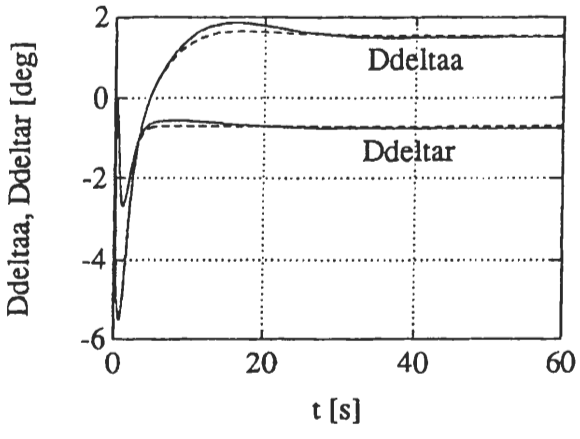


Figure 4-12a. Step responses of RAH mode, $\Delta\varphi_{ref}$ = +/- 10°. V_0 = 35 m/s.

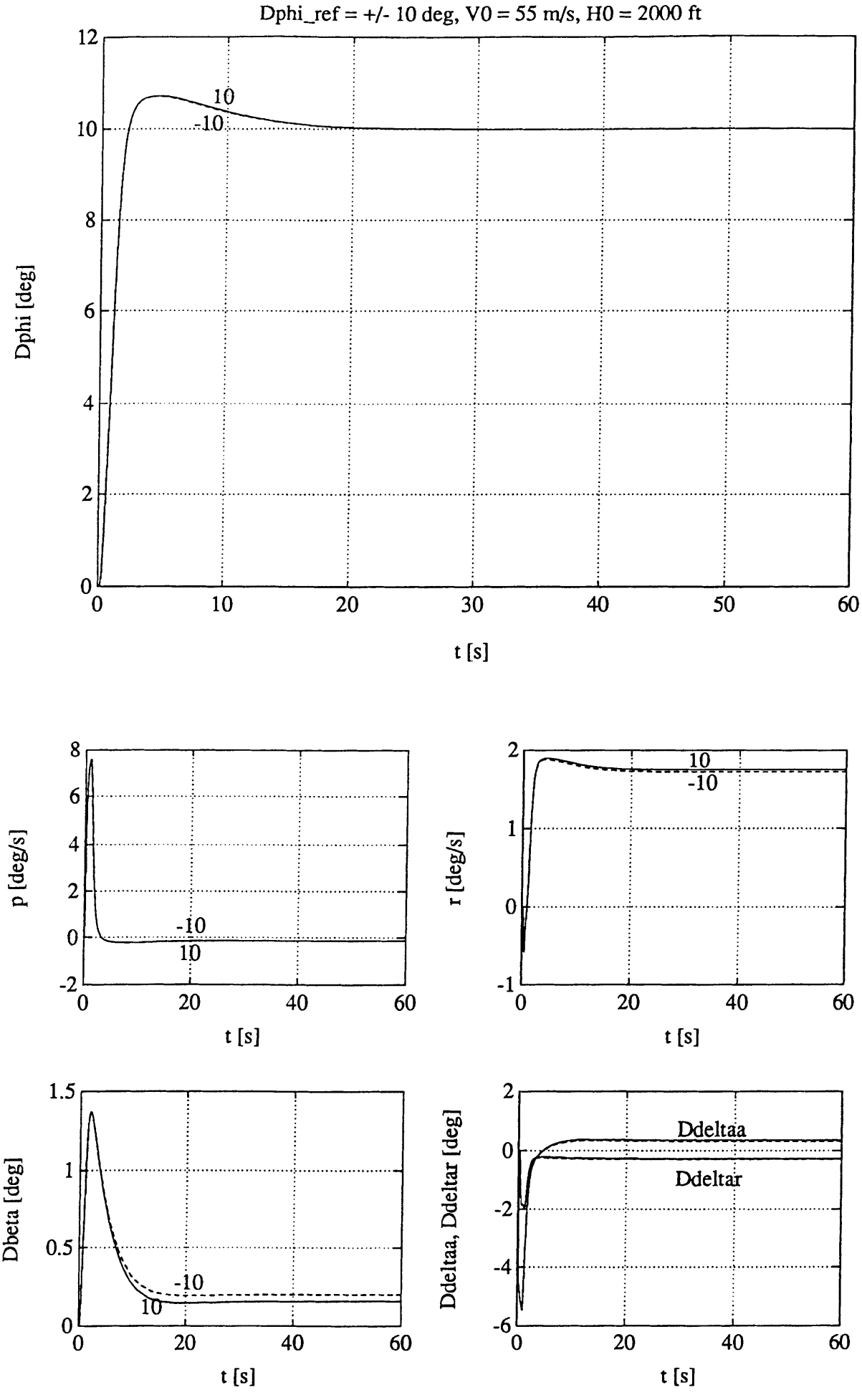


Figure 4-12b. Step responses of RAH mode, $\Delta\varphi_{ref}$ = +/- 10°. V_0 = 55 m/s.

This page is unintentionally left blank

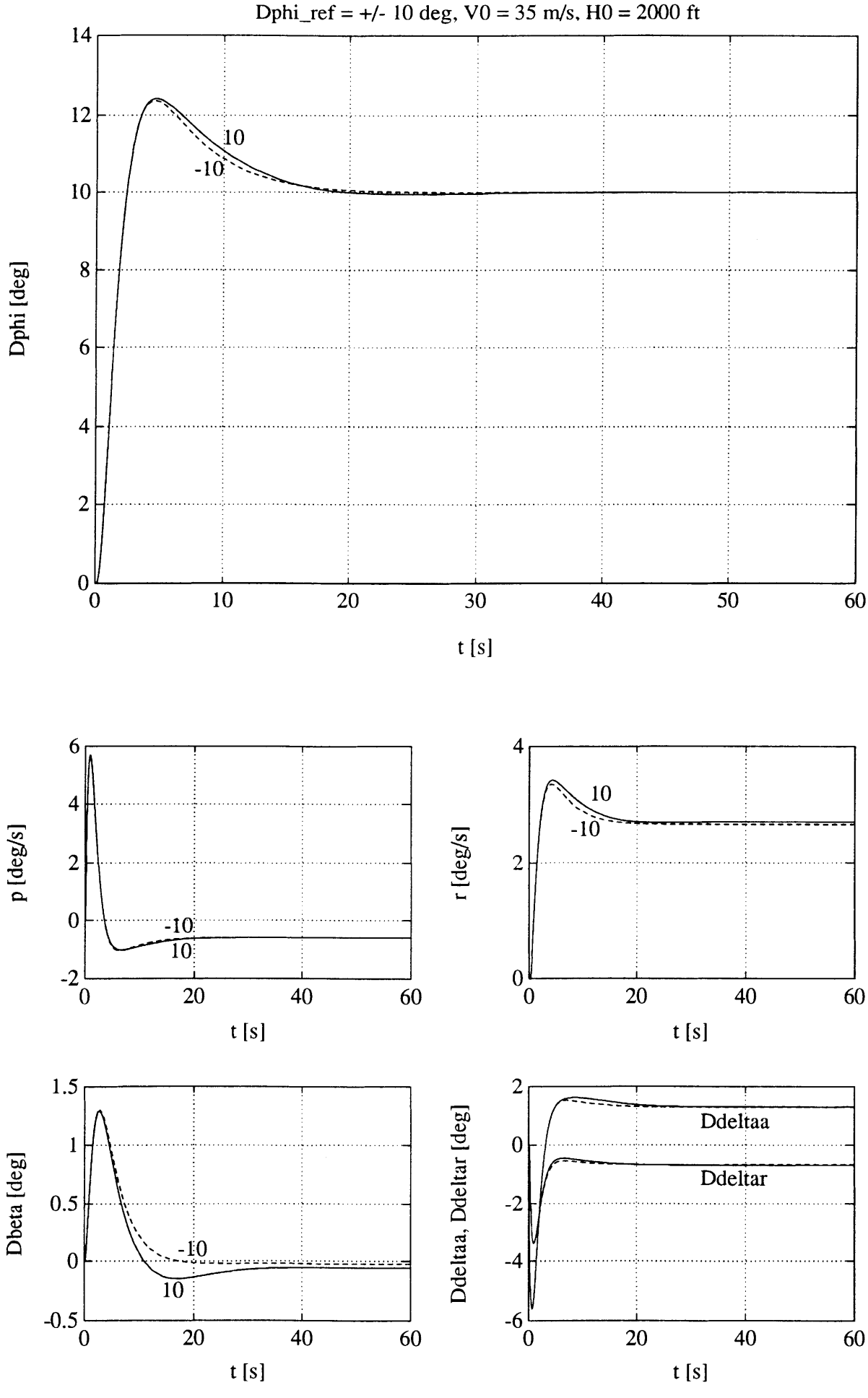


Figure 4-13a. Step response of RAH mode, $\Delta\varphi_{ref}$ = +/- 10°. V_0 = 35 m/s, with α_0-correction for r.

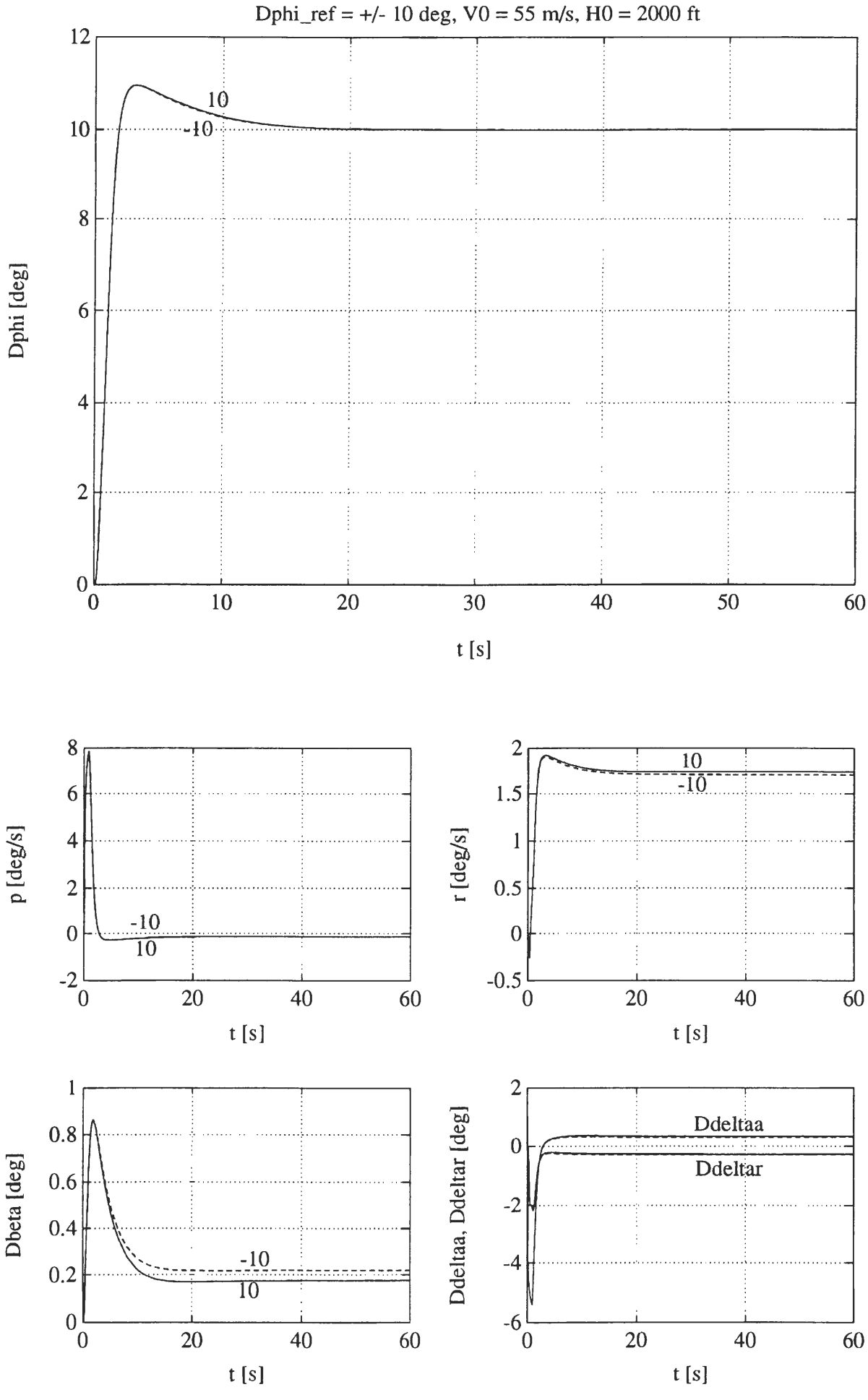


Figure 4-13b. Step responses of RAH mode, $\Delta\varphi_{ref}$ = +/- 10°. V_0 = 55 m/s, with α_0-correction for *r*.

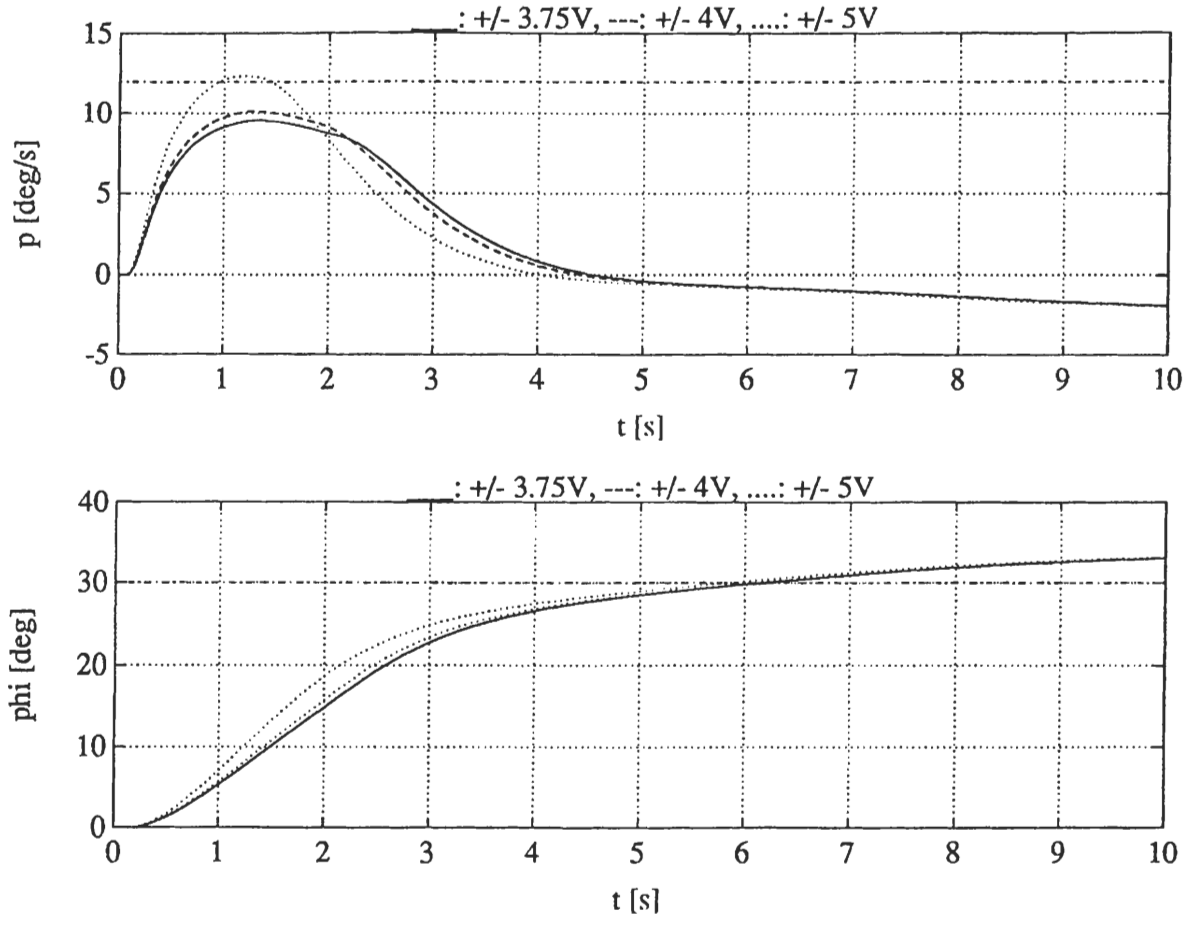


Figure 4-14. Step responses of RAH mode, $\Delta\varphi_{ref}$ = 30°, V_0 = 55 m/s, for different settings of the $\varphi - \varphi_0$ limiters.

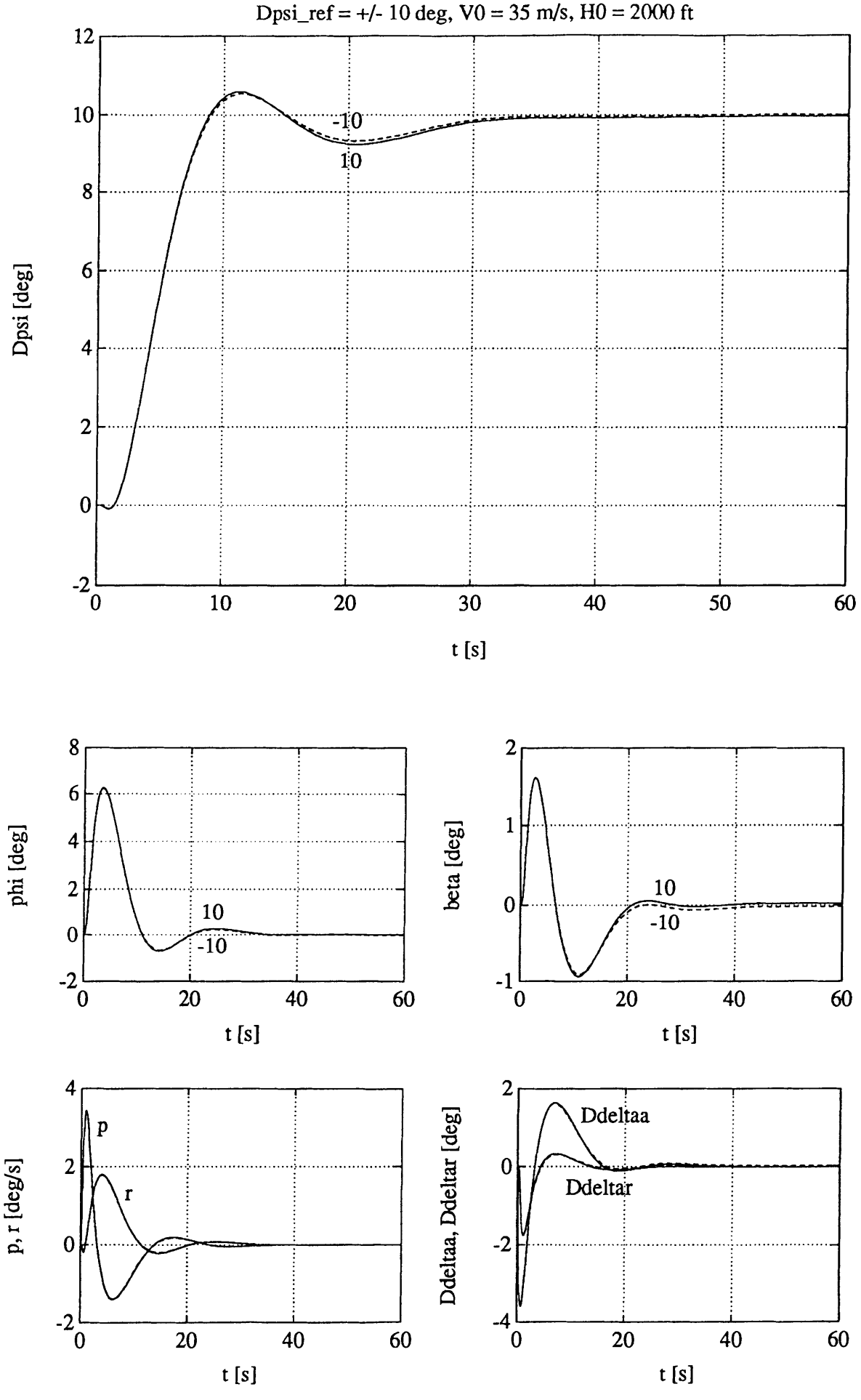


Figure 4-15. Step responses of HH mode, $\Delta\psi_{ref}$ = +/- 10°. V_0 = 35 m/s.

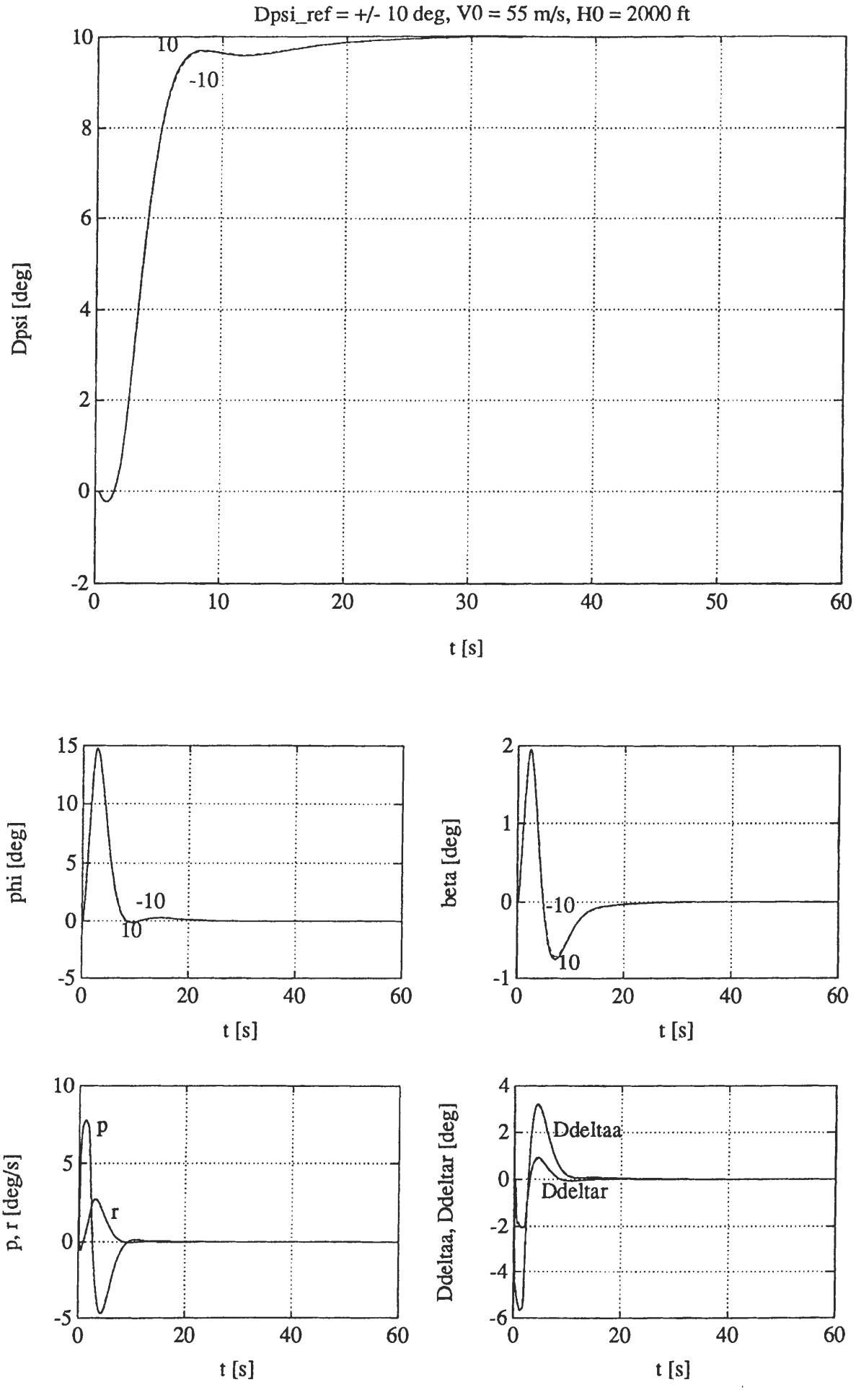


Figure 4-16. Step responses of HH mode, $\Delta\psi_{ref}$ = +/- 10°. V_0 = 55 m/s.

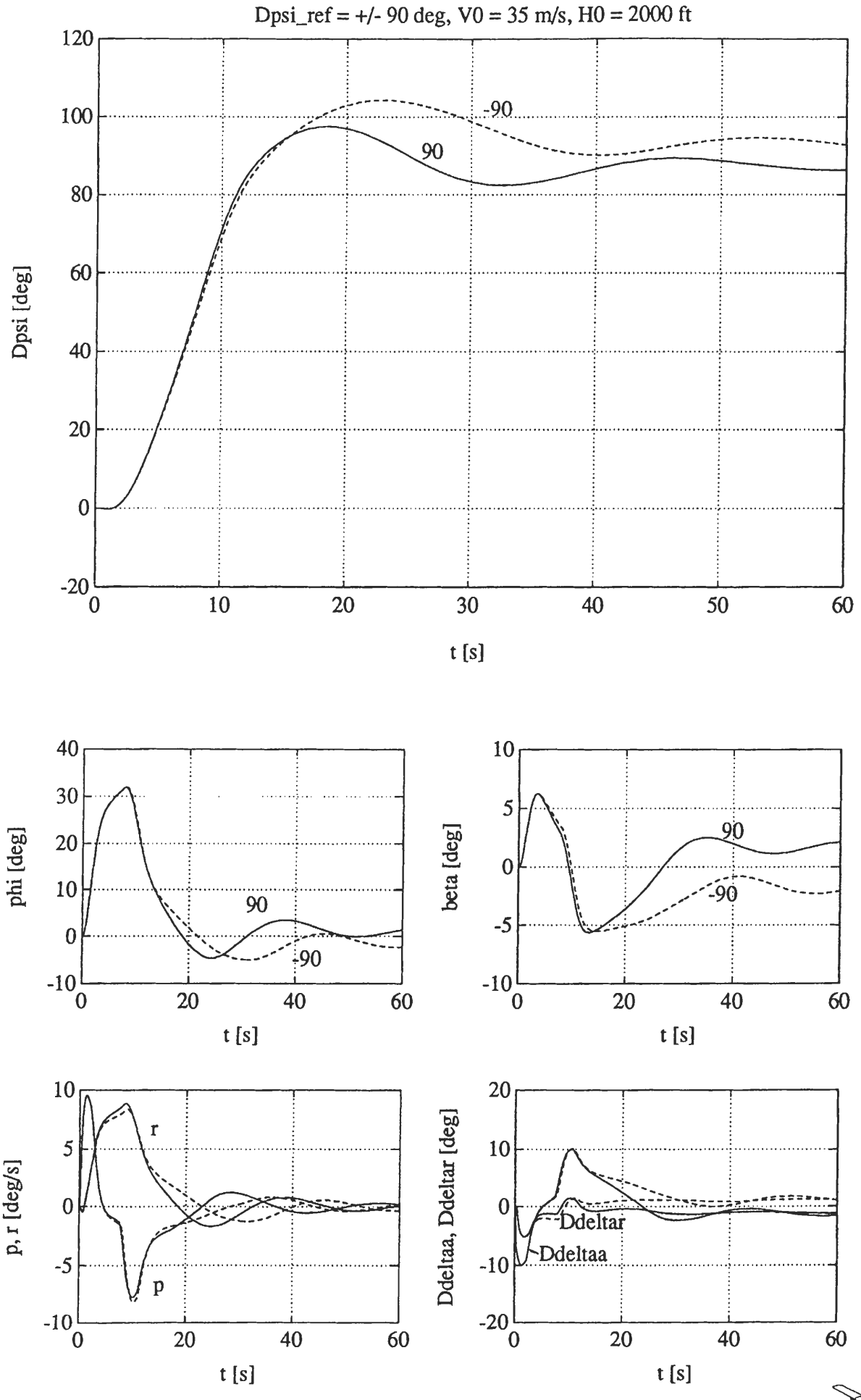


Figure 4-17. Step responses of HH mode, $\Delta\psi_{ref}$ = +/- 90°. V_0 = 35 m/s.

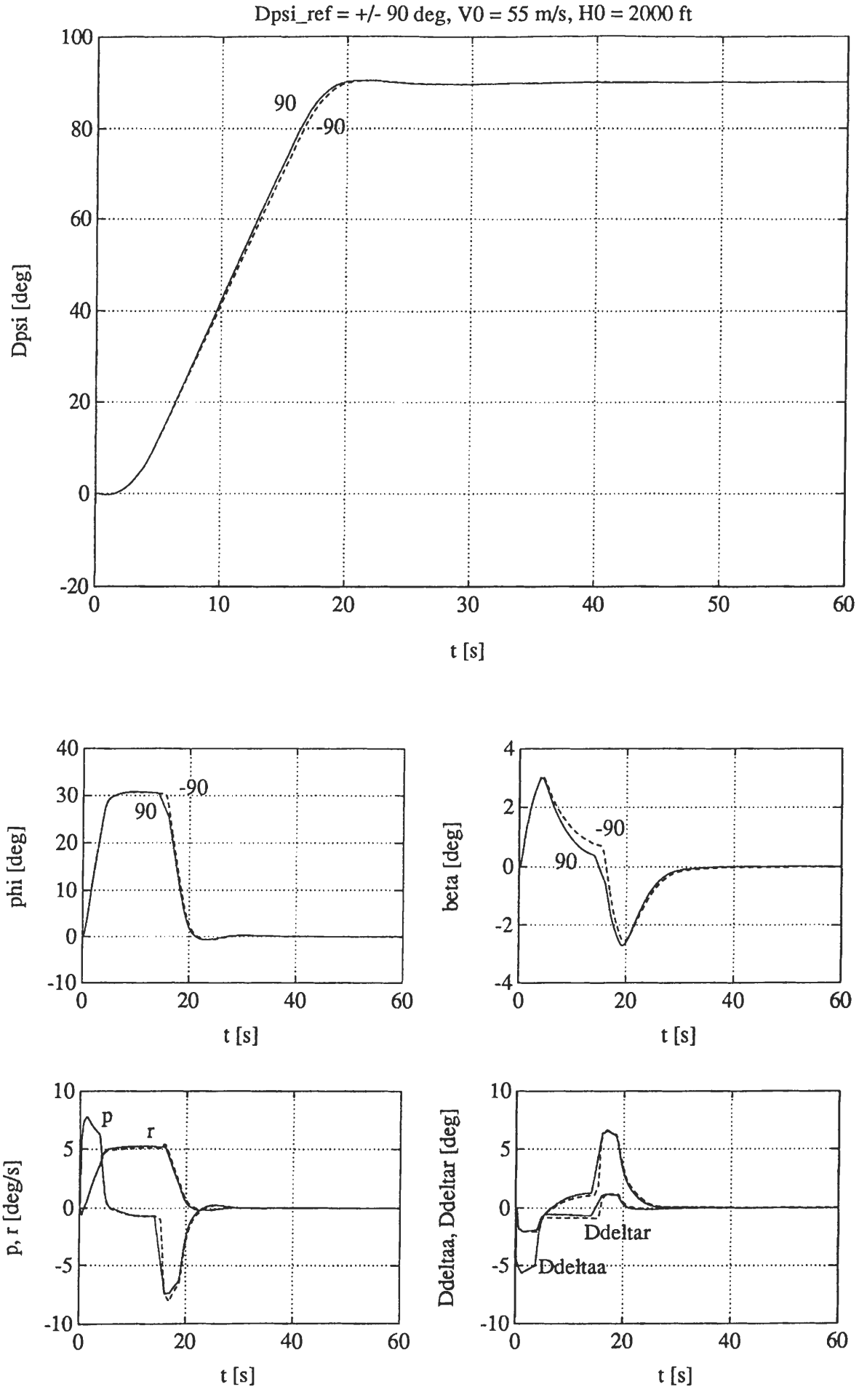
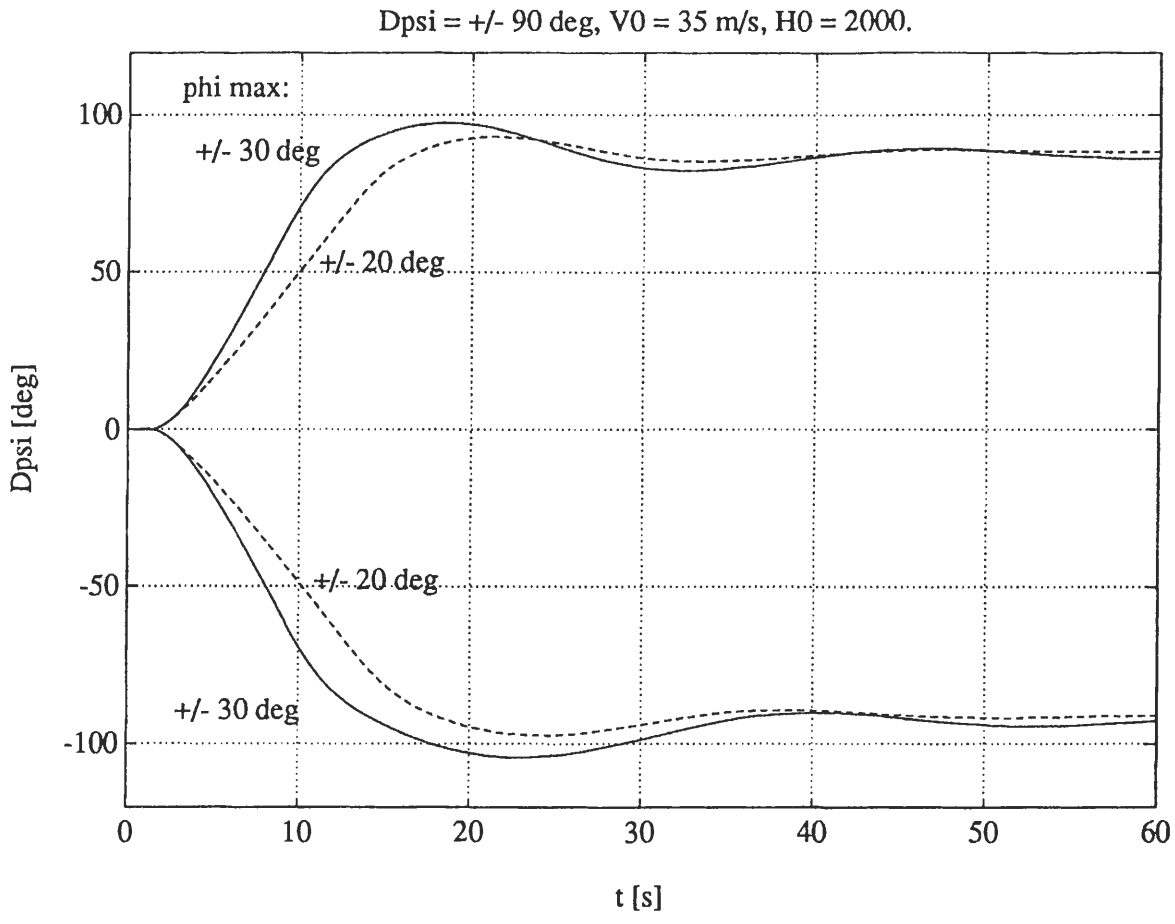


Figure 4-18. Step responses of HH mode, $\Delta\psi_{ref}$ = +/- 90°. V_0 = 55 m/s.



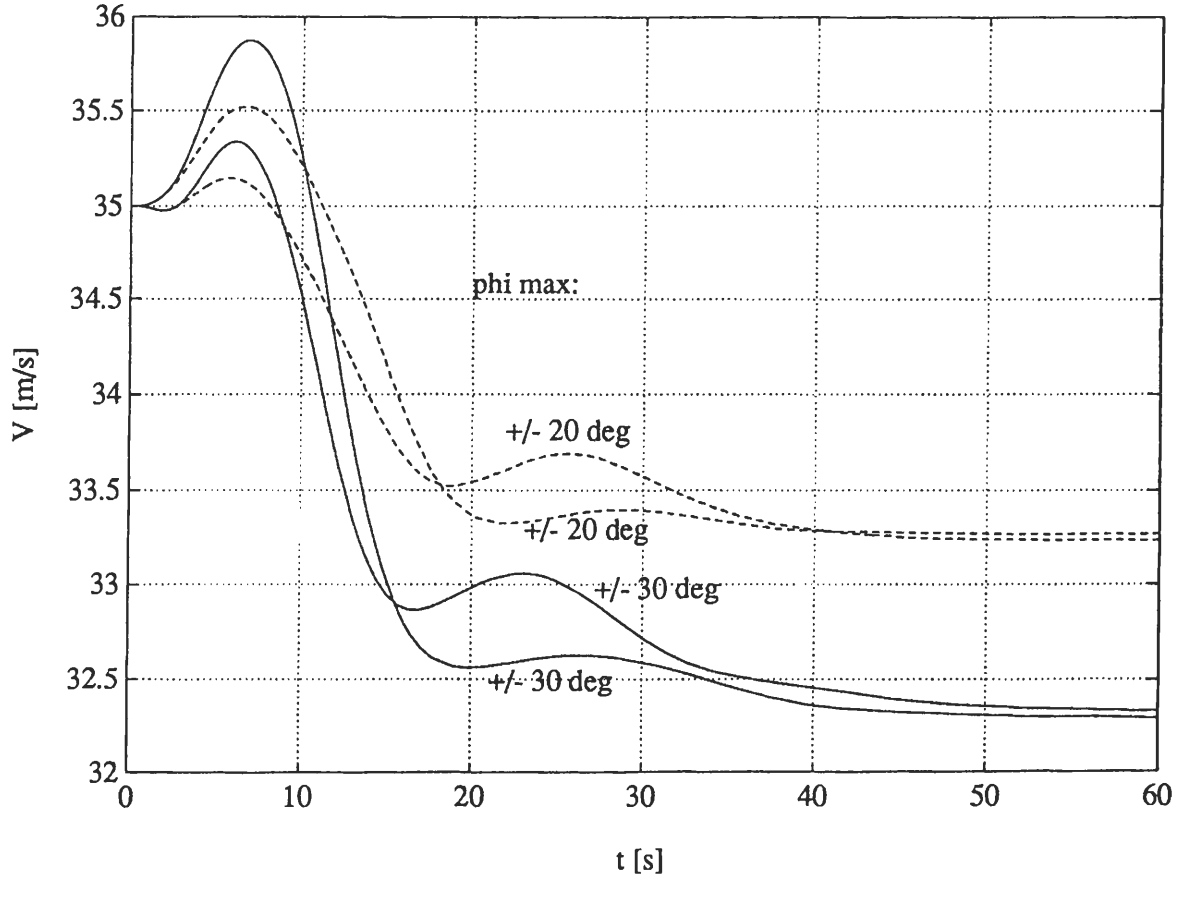


Figure 4-19. Step responses of HH mode, ψ_{ref} = +/- 90$_o$, V_0 = 35 m/s for different settings of the φ-limiters.

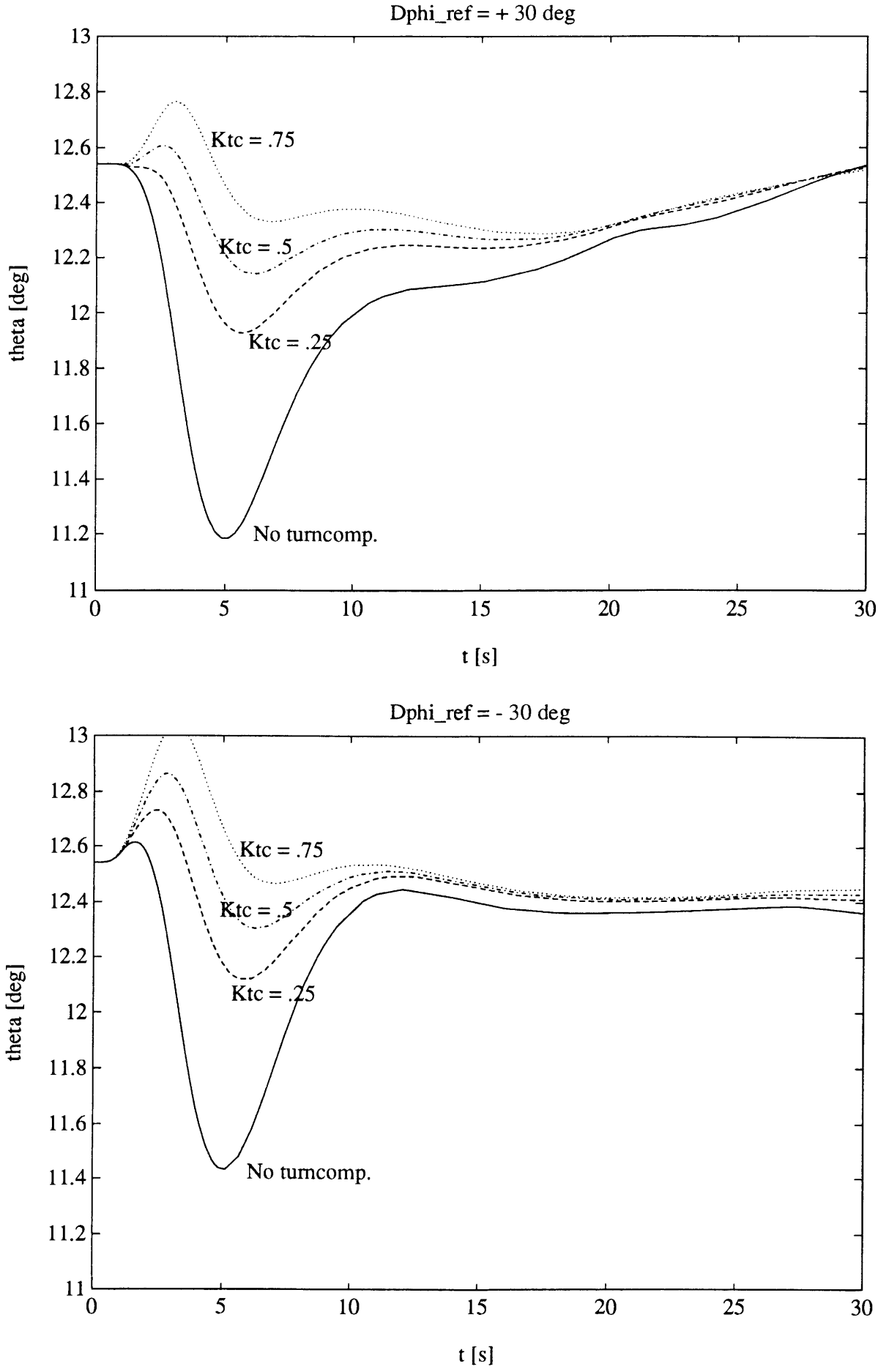
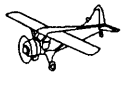


Figure 4-20. Pitch-angle responses to large roll-angle commands for RAH; tuning the turncompensation for PAH, V = 35 m/s.



Chapter 5. Some important lessons from the 'Beaver taskforce' project.

5.1 Introduction.

This graduation work took place within the framework of the 'Beaver Taskforce' project. The primary goal of this 'Taskforce' was to evaluate the theoretical and practical aspects of the autopilot development process once again as a preparation for future research aimed at the new Cessna Citation II laboratory aircraft, which will be configured as *National Fly-by-wire Testbed.* In this chapter, a short summary of some important lessons from this project will be given, in order to take full advantage of this 'Beaver Taskforce' research. It is important to notice the practical character of these conclusions; thorough understanding of the theory of controller design and aircraft dynamics alone is not enough for being able to conduct successful fly-by-wire research!

5.2 Some remarks about the transition from nonlinear simulations to in-flight evaluation.

After the control laws of the 'Beaver' autopilot had been tested in off-line simulations, using MATLAB and SIMULINK, they were implemented directly in the Compaq FCC. Because of the very tight time-schedule, no real-time, piloted flightsimulator sessions could be done. Although this has not resulted in really severe problems, such sessions probably would have been useful for fine-tuning the control laws. In particular, the Heading Select mode turned out to yield very violent turns, which were quite uncomfortable, even though the control law seemed to fulfil the requirements very well [17]. The fast turning speed was visible in simulations, but it was not acknowledged that the passengers comfort would suffer so much. For the Citation FBW-research, it is really necessary to assess the system in piloted flightsimulator sessions before pursuing the flight tests themselves.

The resulting control laws were translated to C after both the MATLAB and the SIMULINK evaluations had given good (and similar) results. The C-programs used for the 'Beaver' autopilot represented translated versions of the ROLM assembler routines, used for the evaluation of the previous autopilot design [9], which were badly structured and very hard to understand. Moreover, the autopilot software could be analyzed with open-loop input/output tests only, which did reveal possible implementation errors such as wrong signs of control deflections, but which were not sufficient to find all software errors before the actual flight tests were done [7].

In the future, tools need to be developed to make it possible to conduct closed-loop analysis of the FCC software too. It is possible to implement the control laws in SIMULINK in a block-diagram structure which resembles the program structure of the FCC software quite well, see chapter 3 and appendix C, and it is therefore recommended to automate the conversion from SIMULINK to the FCCs as much as possible. A quite large number of software errors, made for the evaluations of the autopilot from ref.[9] and the new autopilot

from ref.[16] probably could have been prevented if better tools to validate the software had been available. In refs.[15], [16], and [17], some plain software-errors which were discovered *after* flight tests had been done, have been mentioned. Reading this literature surely increases ones believe in the validity of Murphy's Law...

5.3 Knowing what is measured and which models are used.

During the laboratory-tests of the Flight Control Computer software, a severe quantization effect for the altitude signal due to a Least Significant Bit of 4 feet was discovered. Due to the flexibility of SIMULINK, the influence upon the Altitude Hold control law could be analyzed immediately, see section 4.2.2. However, such information actually needs to be available in advance, to be able to take proper measures if necessary. Therefore, it is very important that the characteristics of the sensors are determined and documented properly, even if this information doesn't seem very important at fist sight. In the example of the altitude quantization, a *steady-state error* of 4 ft would have been perfectly acceptable for normal Altitude Hold operation, but the quantization effect resulted in a bang-bang input to the control law!

In particular, the processes within the *airdata computer* should be made more transparent to the control systems designer. Signal manipulations within the airdata computer may include (or induce) time-delays, Moving Average filters, Least Significant Bits, etc. These manipulations must be known to the designer so that he or she can judge if these complications require changes to the control system. Ideally, the airdata computer itself should be programmable, which would, for instance, create the possibility to increase the resolution of the altitude signal within a certain window around the reference altitude and decrease it if the distance from the aircraft to the reference altitude is large.

It is also very important to know in which reference frames the quantities are measured. This may seem trivial, but unfortunately, practical experiences have shown that it isn't. Somewhere in the design process, is was discovered that the control laws had been designed in the stability reference frame, whereas the motion variables for the aircraft itself and the nonlinear aircraft models were measured in body-axes. The angular velocities in body-axes clearly differ from the angular velocities in stability-axes do to a rotation of axes over an angle α_0, which is quite large at low airspeeds. This negatively affected the performance of the sideslip suppression of the Roll Attitude Hold mode, which could not be solved instantly because α_0 could not be measured accurately and in the case of the 'Beaver' aircraft, there was no space left in the data busses of the FCC to transmit this signal too. The effect was reduced by roughly estimating α_0, using data from ref.[12], but some small errors will probably have remained.

Another error, which could have degraded the autopilot performance severely (as shown in appendix A), was caused by a misunderstanding about the definition of the ailerons angle δ_a. The linear and nonlinear aircraft models used: $\delta_a = \delta_{a_{right}} - \delta_{a_{left}}$, whereas the FCS model from ref.[5] used the definition: $\delta_a = \frac{1}{2} (\delta_{a_{right}} - \delta_{a_{left}})$. This was discovered only a few days before the first flight tests were done!

It is therefore necessary to agree upon standard definitions for reference frames, symbols, etc., to make sure that errors of this kind will not occur anymore.

Usually, an aircraft model will be subdivided in a large number of sub-models. The different people involved in the design process must be supplied with the latest versions of these submodels, and it is very important that everyone uses the same set of models, in order to get 'compatible' results. In practice, this can be a real problem, although the weekly 'Taskforce' meetings helped a lot. Such close cooperation in a 'Taskforce' structure is therefore strongly recommended for similar projects in the future.

5.4 Conclusions.

The 'Beaver Taskforce' project has given us an unique possibility to scrutinize the Aircraft Control System design process from first concept to evaluation in real flight closely. It is important to take lessons from the errors which were made during this design process, in order to prevent such errors in the future. It was discovered that our knowledge of sensor characteristics, and signal manipulations in the airdata computer was not sufficient, misunderstandings about reference frames and notations were made, and pre-flight validation of the software for the Flight Control Computer was too limited. Moreover, it was found that fine-tuning of the control laws would have required piloted flight-simulator sessions too.

These errors and misunderstandings are often of practical nature; the control theory itself never imposed any real problems! Despite these problems, and despite the tight time-schedule, the final flight tests were quite successful, especially with regard to the performance of the basic autopilot modes. This was mainly due to the close cooperation between the different members of the 'Beaver Taskforce'.

Conclusions (part II).

The new control laws for the 'Beaver' autopilot have been implemented in the SIMULINK environment, described in part I of this report. The control laws have been assessed by means of nonlinear simulations, which basically confirmed the linear results from ref.[16]. The following conclusions can be drawn:

1 - The Altitude Hold mode works well, but its performance is seriously affected by the magnitude of the Least Significant Bit for H, used in the Air-data Computer. In order to reduce the 'bouncing effect' which was caused by this LSB, a Moving Average filter has been applied, giving better results.

2 - The performance of the autopilot under atmospheric turbulence has not been evaluated before the flight tests were done, due to the very tight time-schedule. The powerful SIMULINK environment, described in part I, was still under development when the flight tests were done. In this report, some turbulence responses, which were computed afterwards, have been shown, yielding satisfying results.

3 - Due to the fast lateral control laws, the 'Beaver' turns quite violently in Heading Hold / Heading Select mode. The angle of sideslip remains small throughout the whole turn, apart from the initial sideslip angle which occurs when the turn is initiated (see point 4). The first qualitative results of the flight tests confirm these findings. The maximum roll-angle of 30 degrees is too large for low airspeeds, considering the large overshoot in ψ for low initial airspeed.

4 - The large sideslip angle which arises when turns in Roll Attitude Hold or Heading Hold / Heading Select mode are initiated is caused mainly because the linear design of the control laws in ref.[16] has been done in the stability reference frame, whereas the nonlinear analysis uses variables in the body-fixed reference frame, which yields errors in the roll-rate p and the yaw-rate r. Since these quantities are measured in body-axes in the real aircraft too, these variables have been corrected by applying a rotation over α_0, using an approximation of α_0, based upon the based upon the trimmed-flight conditions of ref.[14].

The assessment of the 'Beaver' autopilot has been done by analyzing Bode-plots, Evans root-loci, and time responses with a pilot's eye. However, for future NFT research, more objective (handling qualities-) criteria should be used. It is quite difficult to find out how well the autopilot performs by just evaluating a large number of off-line nonlinear simulations, which was clearly illustrated by the uncomfortably fast turns in Heading Select mode. Therefore, on-line piloted flightsimulator sessions are really needed for successful aircraft controller design, especially in order to fine-tune the control laws.

Recommendations (part II.)

The Aircraft Control System design and analysis package, described in part I of this report, has been developed in parallel with the 'Beaver' autopilot itself. Thus, the current package contains a number of innovations over the simulation structure which was described in ref.[13]. Actually, it now finally represents the design environment which we needed at the time when the 'Beaver' autopilot was developed. It is therefore strongly recommended to use this environment in practice for other control design exercises, and to develop it further for the NFT project. It is also important to create a flexible interface from the MATLAB/SIMULINK analysis to the flightsimulator and the aircraft itself, in order to eliminate conversion time and errors.

During the 'Taskforce' research, it became clear that many problems still arise due to misunderstandings concerning reference frames, units of measurements, and exact meaning of symbols. For this reason, it is necessary to agree upon standard notations, reference frames, and design procedures, to be used for the development of aircraft controllers. It certainly would help a lot if the tools and models for control system analysis are collected in standard SIMULINK/MATLAB libraries, which should be accessible by all members of the Disciplinary Group for Stability and Control.

Such libraries should include a list of sensor dynamics, including possible errors and signal noise models. Information about such quantities as the transport delays of the altitude and velocity sensors, the computational delay of the Airdata Computer, properties like Least Significant Bits, etc. need to be known in advance, *before* starting the control system design process, to eliminate uncertainties and, possibly, unpleasant surprises during flight.

A major uncertainty in the current autopilot system concerns the static gain of the Flight Control System, which converts the control inputs coming from the Flight Control Computer into actual deflections of rudder, elevator, and ailerons. In the current system, the displacements of the actuators are known exactly, but the resulting deflection of the control surfaces may be different from what is expected, due to flexing of the cables, resulting from aerodynamic loads upon the control surfaces.

In order to resolve this uncertainty, it is recommended to either include a feedback-loop of the control surface deflections themselves into the control logic of the FCS, or to connect the actuators directly to the control surfaces. The *dynamics* of the FCS are of great interest, but it is at least as important to be sure that there is no steady-state error in the resulting control deflections.

Finally, the 'Taskforce'-approach has proved to be a very effective way of organizing the autopilot design and evaluation process. Due to the weekly meetings, the information used by the different 'Taskforce' members could be kept up-to-date, and problems could be discussed thoroughly. Therefore, it is recommended to found similar 'Taskforces', consisting of a limited number of students, staff members, and other researchers for other, clearly defined projects in the Disciplinary Group for Stability and Control.

References.

1. Anon. *Approach and Landing Simulation.* AGARD report 632, Ames, 1975.

2. Anon. *SIMULINK, a program for simulating dynamic systems.* User's Guide, The MathWorks Inc., Natick, Massachusetts, USA, 1992.

3. Abbink, F.J.: *Vliegtuiginstrumentatie I/II* (in Dutch). Lecture Notes D-27/D-31, Delft University of Technology (Aerospace Engineering), Delft, 1984/1983.

4. Brandt, A.P., Van den Broek, P.Ph.: *Vliegeigenschappen 2* (in Dutch). Lecture Notes D-34, Delft University of Technology (aerospace engineering), Delft, 1984.

5. Engel, M.C.: *Parameter identificatie electro-hydraulisch besturingssysteem van de DHC-2 'Beaver'* (in Dutch). Graduate's thesis, Delft University of Technology (Aerospace Engineering), Delft, 1992.

6. Gerlach, O.H., *Mathematical model of external disturbances acting on an aircraft during an ILS approach and landing.* Report VTH-159, Delft University of Technology (Aerospace Engineering), Delft, 1970.

7. Gerlach, O.H., *Lecture Notes on Aircraft Stability and Control.* Lecture notes D-26, Delft University of Technology (Aerospace Engineering), Delft, 1981.

8. Ghahramani, B.: *Ontwerp en evaluatie autothrottle voor de DHC-2 Beaver.* Interim report, Delft University of Technology (Aerospace Engineering), Delft, 1993 (to be published, title under reservations).

9. Kruijsen, E.A.C.: *Design and evaluation of automatic approach control laws for the Beaver's digital autopilot.* Graduate's thesis, Delft University of Technology (Aerospace Engineering), Delft, 1990.

10. McLean, D.: *Automatic Flight Control Systems.* Prentice Hall international, Hertfordshire, UK, 1990.

11. McRuer, D., Ashkenas, I., Graham, D.: *Aircraft Dynamics and Automatic Control.* Princeton University Press, Princeton, New Jersey, USA, 1973.

12. Mulder, J.A., Van der Vaart, J.C.: *Aircraft Responses to Atmospheric Turbulence.* Lecture notes D-47, Delft University of Technology (Aerospace Engineering), Delft, edition 1992.

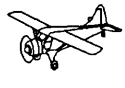
13. Rauw, M.O.: *Using SIMULINK for Nonlinear Analysis of Aircraft Control Systems.* Interim report, Delft University of Technology (Aerospace Engineering), Delft, 1992.

14. Tjee, R.T.H., Mulder, J.A.: *Stability and Control Derivatives of the De Havilland DHC-2 'Beaver' aircraft.* Report LR-556, Delft University of Technology (Aerospace Engineering), Delft, 1988.

15. Wever, P.N.H.: *Een analyse van de regelwetten van het automatische besturingssysteem van het laboratoriumvliegtuig de De Havilland DHC-2 'Beaver'* (in Dutch). Interim report, Delft University of Technology (Aerospace Engineering), Delft, 1992.

16. Wever, P.N.H.: *Ontwerp en implementatie van de regelwetten van het automatisch besturingssysteem van de De Havilland DHC-2 'Beaver'* (in Dutch). Graduate's thesis, Delft University of Technology (Aerospace Engineering), Delft, 1993.

17. Witzenburg, M. van: *Voorbereiding en uitvoering van vliegproeven met een stuurautomaat voor de DHC-2 'Beaver'* (in Dutch). Interim report, Delft University of Technology (Aerospace Engineering), Delft, 1993.



Appendices

Appendix A. Flight control system of the 'Beaver'.

A.1 Introduction.

The control laws of the autopilot generate the command signals for the Flight Control System (FCS), which converts these commands into actual deflections of the control surfaces by means of hydraulic servo's which have been connected to the cable system of the 'Beaver' [1]. The control yoke and rudder move along with the control surfaces; it is not possible to decouple them. In this appendix, the dynamics of the FCS will be described briefly, and it will be shown how the FCS models have been included in the SIMULINK simulation structure. Section A.4 summarizes in which way the FCS dynamics affect the autopilot performance. See ref.[5] for a detailed description of the FCS model and the determination of its parameters.

A.2 The in-flight models of the FCS.

In ref.[5], a third-order dynamic model of the FCS is derived, in which the actuator dynamics have been simplified to a first-order model and the dynamics of the control yoke and cable system are written as a second-order model. The influence of the rotational velocities along the aircraft's body-axes on the aerodynamic hinge-moments of the control surfaces has been taken into account.

The structure of the elevator, ailerons, and rudder models is equal; these models differ in their coefficients only. In the following equations, the subscript c is used to denote the parameters which depend upon the control surface(s) considered, i.e., $c \in \{e, a, r\}$ (elevator, ailerons, rudder). In ref.[5], the following linear state-space model has been derived:

$$\begin{aligned} \dot{\mathbf{x}}_{FCS} &= \mathrm{A}_{FCS}\mathbf{x}_{FCS} + \mathrm{B}_{FCS}\mathbf{u}_{FCS} \\ y_{FCS} &= \mathbf{c}_{FCS}\mathbf{x}_{FCS} \end{aligned} \tag{A-1}$$

where:

$$\mathrm{A}_{FCS} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} ; \quad \mathrm{B}_{FCS} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \tag{A-2}$$

---

1) In this report, the term 'Aircraft Control System' will be used to denote all elements of the control system, including the control laws, whereas the term 'Flight Control System' is used to denote the part of the control system which transfers the command signals into actual deflections of the control surfaces. This definition of the term FCS has been used in several existing reports of the Disciplinary Group for Stability and Control, but remember that other literature often uses different definitions!

The state vector of the FCS model is defined as:

$$\mathbf{x}_{FCS} \equiv \left[y_p \quad \delta_c \quad \dot{\delta}_c \right]^T \tag{A-3}$$

and the input vector is defined as:

$$\text{elevator: } \mathbf{u}_{FCS} = \left[r_p \quad \frac{q\bar{c}}{V} \right]^T; \quad \text{ailerons: } \mathbf{u}_{FCS} = \left[r_p \quad \frac{pb}{2V} \right]^T; \quad \text{rudder: } \mathbf{u}_{FCS} = \left[r_p \quad \frac{rb}{2V} \right]^T \tag{A-4}$$

The elements of the state matrices are given by the following equations:

$$a_{11} = \frac{C_o}{C_o + C_{k_e} h_c} \left(- K_{mp} K_{sp} K_{yp} - \frac{K_{mp}}{C_h} C_{k_e} h_c \right) \tag{A-5}$$

$$a_{12} = \frac{C_o}{C_o + C_{k_e} h_c} \frac{K_{mp}}{C_h} \left(C_{k_e} + C_{a_e} \right) a_c \tag{A-6}$$

$$a_{13} = \frac{C_o}{C_o + C_{k_e} h_c} \left(\frac{K_{mp}}{C_h} \left(K_{k_e} + K_{a_e} \right) a_c + \frac{1}{C_o} \left(C_{k_e} + C_{a_e} \right) a_c \right) \tag{A-7}$$

$$a_{21} = 0 \tag{A-8}$$

$$a_{22} = 0 \tag{A-9}$$

$$a_{23} = 1 \tag{A-10}$$

$$a_{31} = \frac{C_{k_e} h_c}{M_c a_c} + \frac{K_{k_e} h_c}{M_c a_c} \frac{C_o}{C_o + C_{k_e} h_c} \left(- K_{mp} K_{sp} K_{yp} - \frac{K_{mp}}{C_h} C_{k_e} h_c \right) \tag{A-11}$$

$$a_{32} = - \frac{C_{k_e} + C_{a_e}}{M_c} + \frac{K_{k_e} h_c}{M_c} \frac{C_o}{C_o + C_{k_e} h_c} \left(\frac{K_{mp}}{C_h} \left(C_{h_e} + C_{a_e} \right) \right) \tag{A-12}$$

$$a_{33} = - \frac{K_{k_e} + K_{a_e}}{M_c} + \frac{K_{k_e} h_c}{M_c} \frac{C_o}{C_o + C_{k_e} h_c} \left(\frac{K_{mp}}{C_h} \left(K_{k_e} + K_{a_e} \right) + \frac{1}{C_o} \left(C_{k_e} + C_{a_e} \right) \right) \tag{A-13}$$

$$b_{11} = \frac{C_o}{C_o + C_{k_e} h_c} K_{mp} K_{sp} \tag{A-14}$$

$$b_{12} = \frac{C_o}{C_o + C_{k_e} h_c} \frac{K_{mp}}{C_h} C_{p,q,r} \tag{A-15}$$

$$b_{21} = 0 \tag{A-16}$$

$$b_{22} = 0 \tag{A-17}$$

$$b_{31} = \frac{K_{h_e} h_c}{M_c a_c} \frac{C_o}{C_o + C_{k_e} h_c} K_{mp} K_{sp} \tag{A-18}$$

$$b_{32} = -\frac{C_{p,q,r}}{M_c a_c} + \frac{K_{k_e} h_c}{M_c a_c} \frac{C_o}{C_o + C_{k_e} h_c} \frac{K_{mp}}{C_h} C_{p,q,r} \tag{A-19}$$

The output signal y_{FCS} equals:

$$y_{FCS} \equiv \delta_c \tag{A-20}$$

and the vector $\mathbf{c}_{FCS}$ from the output equation (A-1) equals:

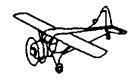
$$C_{FCS} = \begin{bmatrix} 0 & \frac{180}{\pi} & 0 \end{bmatrix}^T \tag{A-21}$$

Note: the coefficient $C_{p,q,r}$ is a multiplication constant for the input signal $\frac{pb}{2V}$, $\frac{q\bar{c}}{V}$, or $\frac{rb}{2V}$ ($C_{p,q,r} = C_p$, C_q, or C_r, respectively).

The C-coefficients bring into account the spring-constants of the system, the K-coefficients bring into account the damping characteristics (viscous friction), and the M-coefficients bring into account the mass-forces of the system. The symbols from the FCS models are defined as follows:

r_p	input voltage to servo (command signal to FCS)
y_p	displacement of piston
δ_c	control surface deflection (index $c \in \{e, a, r\}$)

a_c, C_o, C_h, C_{k_e}, h_c, K_{k_e}, K_{mp}, K_{sp}, K_{yp}, and M_c are coefficients of the FCS model (index $c \in \{e, a, r\}$), see ref.[5]. All constants have been defined or estimated in ref.[5] for three different velocities, spanning the part of the flight-envelope from stall to maximum airspeed (V = 35, 45, and 55 m/s). For this reason, the gains and filter coefficients for the 'Beaver' autopilot were optimized for these three velocities, which resulted in a quadratic gain-scheduling scheme as listed in table 2-1 of chapter 2.



A.3 Application of the FCS models in SIMULINK.

Ready-to-use state-space matrices were created and stored in the MAT-files FCS35.MAT, FCS45.MAT, and FCS55.MAT. Compared to ref.[5], some small corrections had to be made to make it possible to connect these state-space models to the nonlinear aircraft model in SIMULINK, described in part I of this report. In ref.[5], all control surface deflection angles were measured in *degrees* and the values of the angular velocities p, q, and r, used for the determination of $\frac{pb}{2V}$, $\frac{q\bar{c}}{V}$, and $\frac{rb}{2V}$ were measured in *radians per second*, whereas the SIMULINK aircraft model uses angles in radians and angular velocities in radians per second. Hence, the ratio between angular velocities and angles differed by a factor $\pi/180$. This has been corrected by multiplying the column of the B matrices of the FCS models which corresponded with the control surface deflection with this same factor.

Another correction involved the definition of the aileron deflection, which was defined in ref.[5] as: $\delta_a = (\delta_{a_{right}} - \delta_{a_{left}})$, which is a factor 2 smaller than the definition used here. Finally, the static gain of the FCS models turned out to be a function of the engine speed n, although this influence could not accurately be modelled because the measurements used for the identification of the FCS did not cover a sufficient range of n-settings for such modelling (see the list of flight conditions in ref.[5]).

For this reason, it was decided to use the FCS models for V = 35 m/s and V = 55 m/s, and the rudder model for V = 45 m/s, which were all determined for n = 1800 RPM, and to correct the elevator and ailerons FCS models for V = 45 m/s, which were determined for n = 2000 RPM, in such way that the static gain would lie exactly between the static gain of the aileron and elevator models for V = 35 and V = 55 m/s. In table A-1, the C_{ail} and C_{elv} models for V = 45 m/s clearly differ from the other C-matrices, and the correction factors turn out to be quite large. This means that the autopilot performance will differ from the predictions form the simulations if the system is used for other engine speeds!

The resulting state-space models have been stored the files FCS35.MAT, FCS45.MAT, and FCS55.MAT, which have been included to the subdirectory *BEAVER*. These datafiles contain the matrices A_{ail}, B_{ail}, C_{ail}, D_{ail}, A_{elv}, B_{elv}, C_{elv}, D_{elv}, A_{rud}, B_{rud}, C_{rud}, and D_{rud}, with state-space matrices for the FCS models of ailerons, elevator, and rudder. The definition of these matrices is given in table A-1.

Before starting a simulation of the simulation model of the 'Beaver' autopilot, the parameters for the aircraft model and the state matrices for the FCS models must be defined in the MATLAB workspace. This can be done by applying the initialization program *APINIT*, which can be started by double-clicking the initialization button, or by typing *apinit* at the MATLAB command line. Choose option 2 (*load FCS models*). See also section C.13.

Note: the coefficients of the FCS models depend upon the airspeed, but unfortunately the state matrices are available for three different airspeeds only. For simulation purposes, the FCS model which is valid for the airspeed which lies nearest to the speed-range which will be covered in the simulation

which lies nearest to the speed-range which will be covered in the simulation must be loaded. When large changes of the airspeed, covering the complete flight-envelope from stall to V_{max}, will be made, it is probably best to load the state matrices from FCS45.MAT and eventually check the results for FCS35 and FCS55 in order to do some sensitivity-analysis to ensure robustness of the system.

A.4 Influence of the FCS upon the autopilot performance.

The FCS affects the autopilot performance in two ways:

1 - by its static gain, which actually determines the magnitude of the deflections of elevator, ailerons, and rudder,

2 - by its dynamical behaviour.

The actuators of the 'Beaver' autopilot have not been connected directly to the control surfaces. For easy access, they have been mounted somewhere in the aircraft's fuselage, where they have been connected to the cable system which is also used for manual control. Unfortunately, only the actuator deflections are actually controlled in closed-loop; the part of the system between the actuators and the control surfaces themselves behaves as an open-loop system. Therefore, it is not guaranteed that the commanded signals are equal to the resulting deflections of the elevator, ailerons, and rudder. If the static gains of the FCS system differ from the values of the FCS models, this effectively increases or decreases the magnitude of the gains from the control laws, which immediately alters the performance of the autopilot if the system is not very robust.

According to ref.[16], the control laws of the 'Beaver' autopilot all have sufficiently large phase and gain margins. Figures A-1 to A-3 show responses of the Pitch Attitude Hold and Roll Attitude Hold modes for variations in the static gain of the elevator, ailerons, and rudder model, respectively. The step responses of the RAH mode clearly suffer when the static gain of the ailerons model is multiplied or divided by a factor 2, but the responses for variation in the static gain of the elevator and rudder models remain acceptable. Figures A-1 to A-3 show responses for V = 35 m/s; the results for larger airspeeds were comparable to these figures. The RAH responses got better if the static gains of the ailerons and rudder models were increased or increased simultaneously. If the static gains of the ailerons was increased and the static gain of the rudder was decreased, or vice versa, the RAH performance got worse.

The dynamic behaviour of the FCS, which was described in section A.2, mainly affects the fastest characteristic motions of the aircraft, i.e., the short-period oscillation and the Dutch-roll. The dynamics of the FCS are largely caused by the influence of the cable-system and the movements of the control yoke and rudder. For the 'Beaver', it is not possible to decouple the movements of the control surfaces from the control yoke and rudder.

In ref.[16], the influence of the FCS dynamics upon the performance of the autopilot was analyzed more thoroughly. When the FCS dynamics were taken into account, the damping of the short-period and Dutch-roll modes was reduced a little, compared to the situation in which the FCS was modelled as

a static gain only. However, the step responses for both situations were almost identical.

The results from this report were all computed for the complete nonlinear aircraft model, equipped with the third-order state-space blocks for the FCS dynamics, described in section A.2, because that situation resembles the situation in real flight most closely.
Besides the FCS models, a time-delay of 0.03 seconds has been used, to bring into account the computational delay (which in reality is smaller or equal than the sample time T_s = 0.03 seconds). The continuous versions of the control laws have been applied, for reasons of computing speed, and because ref.[16] has shown that the differences between responses of the original continuous-time controller and the discrete version of these controller are very small if a sampling-time of 0.03 seconds is used.

A.5 Conclusions.

The FCS models for elevator, ailerons, and rudder all have the same third-order state-space format, which has been shown in this appendix. For simulation purposes, ready-to-use state-space models have been stored in the files FCS35.MAT, FCS45.MAT, and FCS55.MAT, which can conveniently be loaded to the MATLAB workspace with the initialization routine *APINIT*. The exact values of the static gains of the FCS models are somewhat uncertain due to the influence of the engine speed, which could not accurately be modelled due to a lack of suitable measurements. Errors in the static gains effectively result in larger or smaller gains, so in the future, such uncertainties must be avoided as much as possible! The dynamical behaviour of the FCS mainly affects the short period and Dutch-roll modes. See ref.[5] for a detailed description of the FCS models, the estimation of the model parameters, and a list of available measurements in flight.

V = 35 m/s:

$$A_{ail} = \begin{bmatrix} -10.597 & -1.8986 & -5.4897 \\ 0 & 0 & 1.0000 \\ 1.0877 & -402.22 & -23.387 \end{bmatrix}; \quad B_{ail} = \begin{bmatrix} 27.463 & -0.0196 \\ 0 & 0 \\ 3.0187 & -4.0324 \end{bmatrix}$$

$$A_{elv} = \begin{bmatrix} -10.951 & 7.7246 & 22.099 \\ 0 & 0 & 1.0000 \\ 7.3446 & -710.28 & -18.531 \end{bmatrix}; \quad B_{elv} = \begin{bmatrix} 25.157 & -0.0025 \\ 0 & 0 \\ 5.8694 & -0.2830 \end{bmatrix}$$

$$A_{rud} = \begin{bmatrix} -9.2131 & 4.0945 & 11.881 \\ 0 & 0 & 1.0000 \\ 0.6720 & -682.52 & -42.323 \end{bmatrix}; \quad B_{rud} = \begin{bmatrix} 24.571 & 0.0029 \\ 0 & 0 \\ 7.5436 & -0.5140 \end{bmatrix}$$

$$C_{ail} = [\ 0 \quad 57.296 \quad 0\ ]; \quad D_{ail} = [\ 0 \quad 0\ ] \qquad C_{elv} = [\ 0 \quad 57.296 \quad 0\ ]; \quad D_{elv} = [\ 0 \quad 0\ ] \qquad C_{rud} = [\ 0 \quad 57.296 \quad 0\ ]; \quad D_{rud} = [\ 0 \quad 0\ ]$$

V = 45 m/s:

$$A_{ail} = \begin{bmatrix} -10.597 & -3.2326 & -9.2515 \\ 0 & 0 & 1.0000 \\ 1.0877 & -684.83 & -19.623 \end{bmatrix}; \quad B_{ail} = \begin{bmatrix} 27.463 & -0.0054 \\ 0 & 0 \\ 3.0187 & -1.1144 \end{bmatrix}$$

$$A_{elv} = \begin{bmatrix} -10.951 & 8.8068 & 25.186 \\ 0 & 0 & 1.0000 \\ 7.3446 & -809.80 & -20.197 \end{bmatrix}; \quad B_{elv} = \begin{bmatrix} 25.157 & -0.0005 \\ 0 & 0 \\ 5.8694 & -0.0516 \end{bmatrix}$$

$$A_{rud} = \begin{bmatrix} -9.2131 & 4.8550 & 14.089 \\ 0 & 0 & 1.0000 \\ 0.6720 & -809.30 & -50.348 \end{bmatrix}; \quad B_{rud} = \begin{bmatrix} 24.571 & 0.0042 \\ 0 & 0 \\ 7.5436 & -7.4159 \end{bmatrix}$$

$$C_{ail} = [\ 0 \quad 83.875 \quad 0\ ]; \quad D_{ail} = [\ 0 \quad 0\ ] \qquad C_{elv} = [\ 0 \quad 67.162 \quad 0\ ]; \quad D_{elv} = [\ 0 \quad 0\ ] \qquad C_{rud} = [\ 0 \quad 57.296 \quad 0\ ]; \quad D_{rud} = [\ 0 \quad 0\ ]$$

V = 55 m/s:

$$A_{ail} = \begin{bmatrix} -10.597 & -2.6380 & -7.5845 \\ 0 & 0 & 1.0000 \\ 1.0877 & -558.86 & -23.353 \end{bmatrix}; \quad B_{ail} = \begin{bmatrix} 27.463 & -0.0008 \\ 0 & 0 \\ 3.0187 & -0.1639 \end{bmatrix}$$

$$A_{elv} = \begin{bmatrix} -10.951 & 7.2721 & 20.799 \\ 0 & 0 & 1.0000 \\ 7.3446 & -668.67 & -16.870 \end{bmatrix}; \quad B_{elv} = \begin{bmatrix} 25.157 & -0.0020 \\ 0 & 0 \\ 5.8694 & -0.2267 \end{bmatrix}$$

$$A_{rud} = \begin{bmatrix} -9.2131 & 5.5178 & 16.008 \\ 0 & 0 & 1.0000 \\ 0.6720 & -919.78 & -56.453 \end{bmatrix}; \quad B_{rud} = \begin{bmatrix} 24.571 & 0.0036 \\ 0 & 0 \\ 7.5436 & -0.6364 \end{bmatrix}$$

$$C_{ail} = [\ 0 \quad 57.296 \quad 0\ ]; \quad D_{ail} = [\ 0 \quad 0\ ] \qquad C_{elv} = [\ 0 \quad 57.296 \quad 0\ ]; \quad D_{elv} = [\ 0 \quad 0\ ] \qquad C_{rud} = [\ 0 \quad 57.296 \quad 0\ ]; \quad D_{rud} = [\ 0 \quad 0\ ]$$

Table A-1. State matrices of FCS models, contained in the files FCS35.MAT, FCS45.MAT and FCS55.MAT.

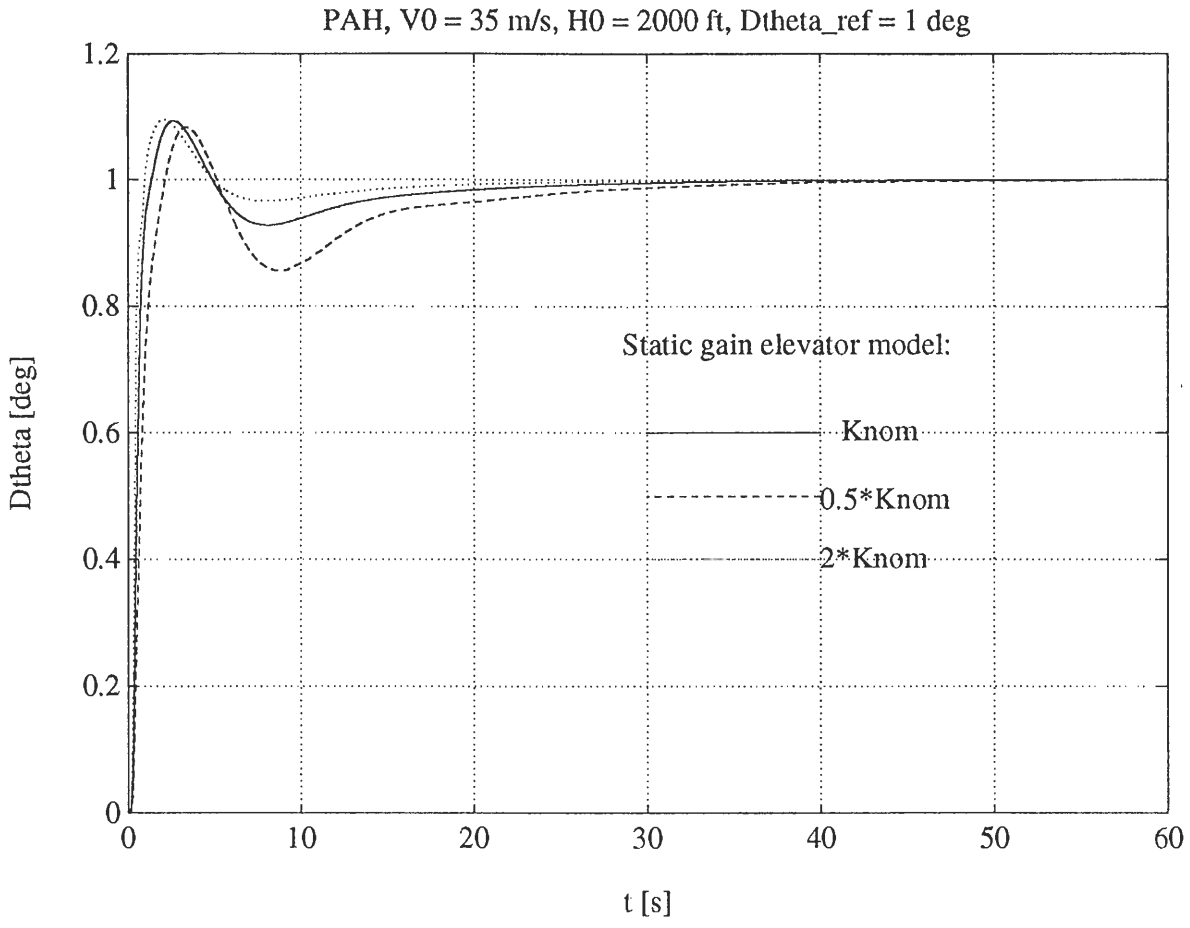


Figure A-1. Step responses of the Pitch Attitude Hold mode for different errors in the static gains of the elevator model (K_{nom} is the nominal value)

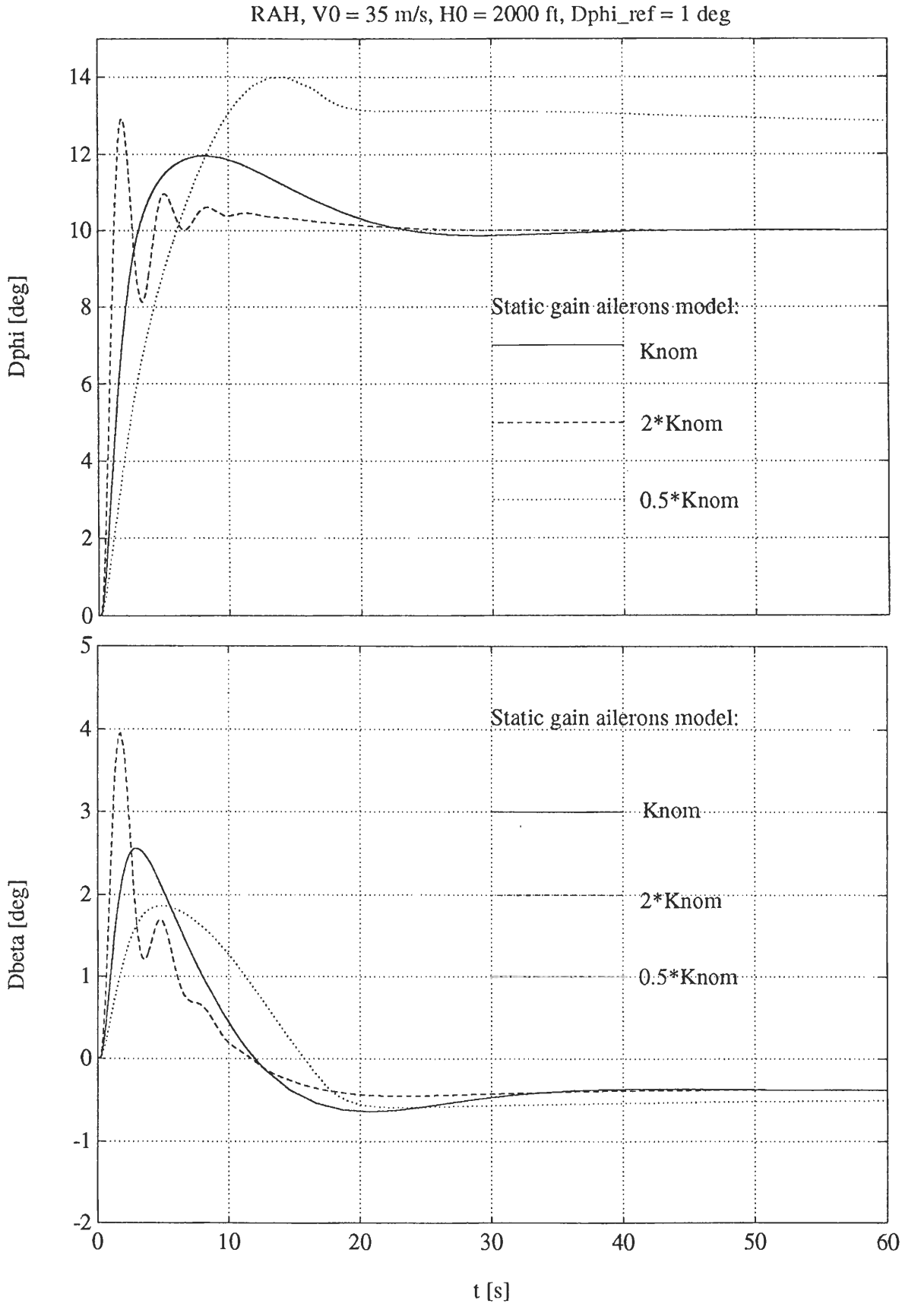


Figure A-2. Step responses of the Roll Attitude Hold mode for different errors in the static gains of the ailerons model (K_{nom} is the nominal value)

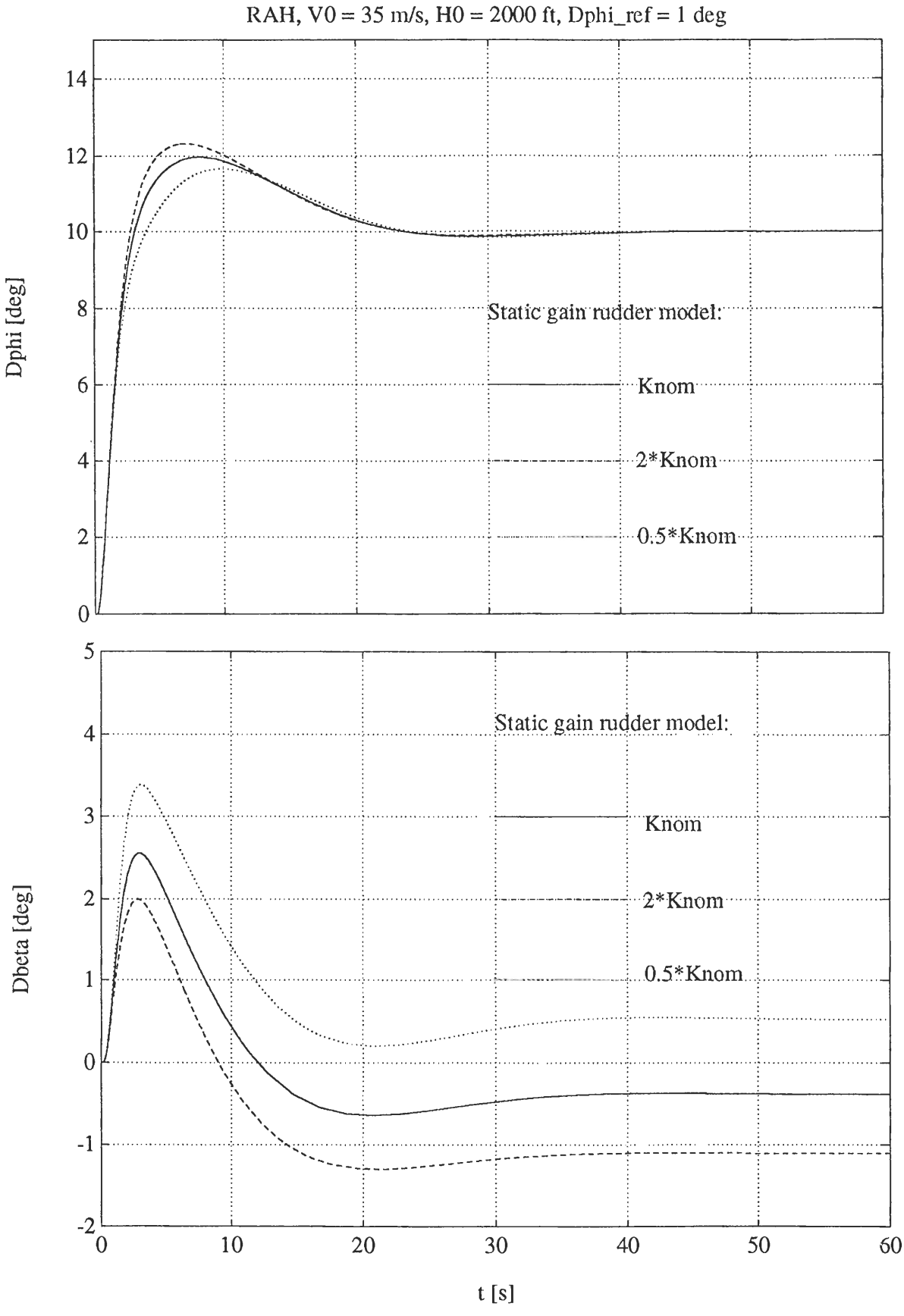


Figure A-3. Step responses of the Roll Attitude Hold mode for different errors in the static gains of the rudder model (K_{nom} is the nominal value)

Appendix B. Masked blocks, not contained in the standard SIMULINK libraries.

B.1 Introduction.

For the implementation of the autopilot control laws it was necessary to implement some blocks which were not contained in the standard SIMULINK blocklibraries, such as a scheduled gain, a Moving Average filter, etc. Since these blocks may be useful for other purposes too, they will be described briefly in this appendix. See appendix G of part I and ref.[Manual] for more information about the standard SIMULINK blocklibraries and the *Mask* function. All blocks from this appendix are contained in the library *TOOLS* which has been included in the subdirectory TOOLS on the floppy disk of this report.

B.2 New input blocks.

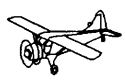
Some inputsignals which are not contained in the standard *Sources* library have been created, using the *Mask* function of SIMULINK:

1 - *Block-Fcn*. This block creates a block-shaped signal by adding two step signals to a constant. The first step has a magnitude of y_{block} and is applied at time $t = t_{start}$. A second step with a magnitude of $-y_{block}$ and step-time $t = t_{stop}$ is added to this signal. If $t_{stop} > t_{start}$, the first step signal initializes the block, and the second step brings the output value back to the initial value. A constant, which contains the initial and final value of the signal from the *Block-Fcn*, is added to these two steps. Note: no errors will appear if the user specifies a value t_{stop} which is smaller than t_{start}, even though this is incorrect. The resulting block signal will have the correct magnitude, but an opposite sign!

Figure B-1 shows the internal structure of *Block Fcn*. The step called *Start Block* initiates the block-signal at $t = t_{start}$, the step *End block* ends it at $t = t_{stop}$. The constant value y_{init} represents the initial (= final) value of the signal. The variables *yblock*, *yinit*, *tstart*, and *tstop* represent y_{block}, y_{init}, t_{start}, and t_{stop}, respectively.

2 - *Doublet*. This block creates a doublet signal. In this case, the user must specify the time at which the doublet must be initiated (*ts1*), two time-intervals (*dt1* and *dt2*), the magnitude of the doublet in the first time-interval (*y1*), the magnitude of the doublet in the second time-interval (*y2*), and the initial (= final) value of the signal which leaves the *Doublet* block (*C*). The symbols between brackets denote the names of these variables.

The *Doublet* is generated in a similar way as the *Block Fcn* with three step-functions and one constant of which the signals are added together. The internal structure of this block is shown in figure B-2. Note: no error message will be displayed if the user enters negative values for the two time-intervals *dt1* and *dt2*, even though the resulting doublet will probably not make any sense.



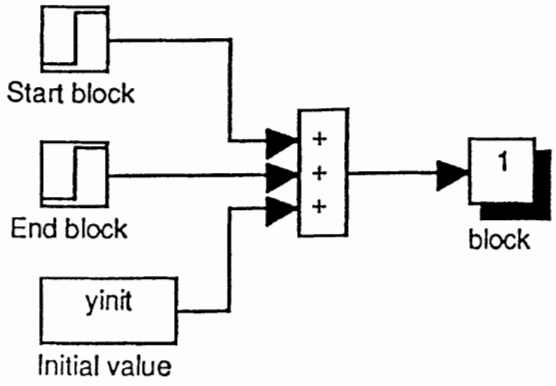


Figure B-1. Internal structure of the block *BlockFcn*.

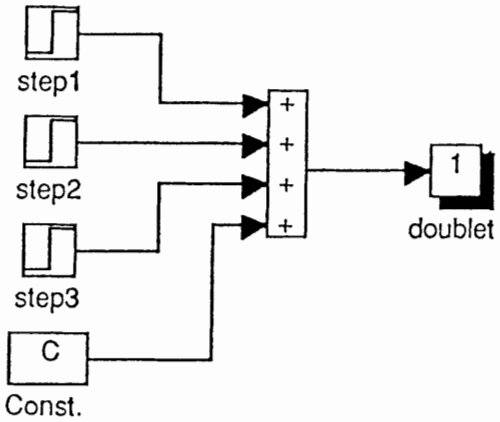


Figure B-2. Internal structure of the block *Doublet*.

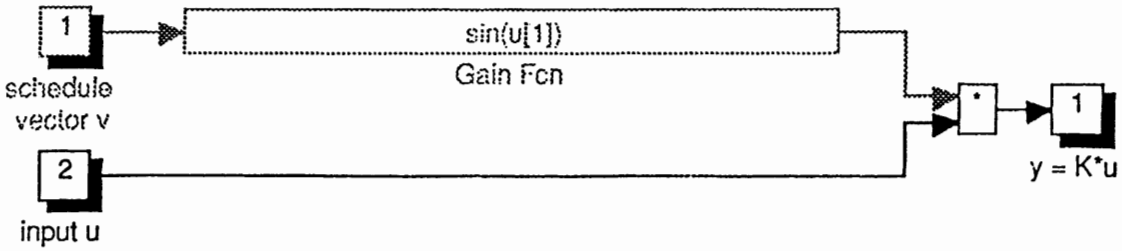


Figure B-3. Internal structure of the block *Scheduled Gain*.

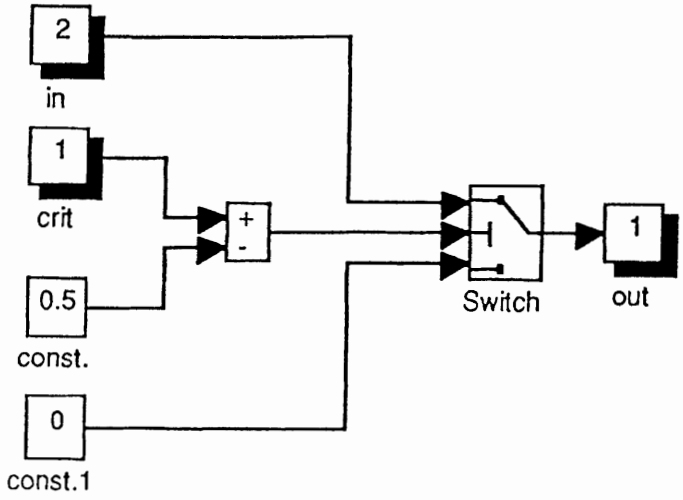


Figure B-4. Internal structure of the block *On/off switch*.

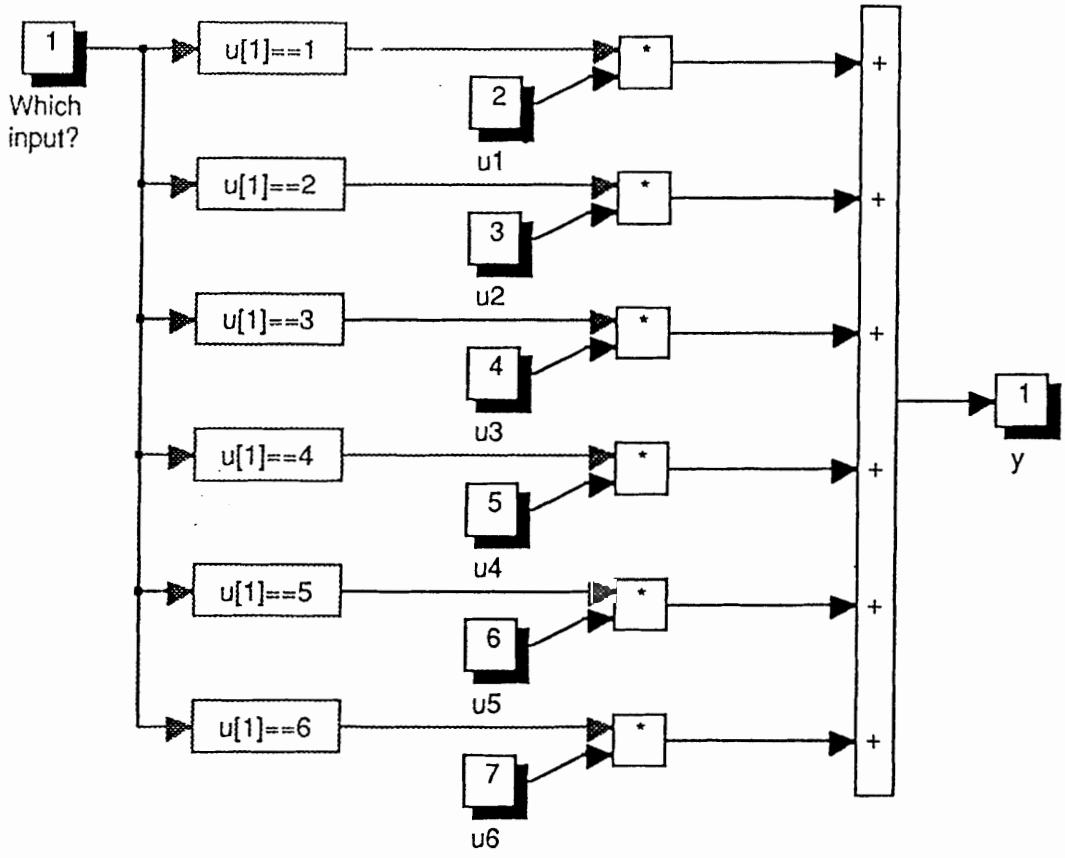


Figure B-5. Internal structure of the block *n-Switch*.

The signal which leaves the *Doublet* block will equal:

$$\begin{aligned} y &= C, && t < tsl \\ y &= C + y1, && tsl < t < (tsl + dt1) \\ y &= C - y2, && (tsl + dt1) < t < (tsl + dt1 + dt2) \\ y &= C, && t > (tsl + dt1 + dt2) \end{aligned} \tag{B-1}$$

B.3 Moving Average filter.

The Moving Average filter, which has been used for the ALH mode, but which can be applied more generally, is used to compute the average of the input signal, using n_s samples which are timely spaced in sample intervals of T_s seconds. The Moving Average filter has been implemented as a discrete *State-space* block, which is defined as follows:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathrm{A}_{MA}\mathbf{x}(k) + \mathbf{b}_{MA}u(k) \\ y(k) &= \mathbf{c}_{MA}\mathbf{x}(k) \end{aligned} \tag{B-2}$$

where:

$$\mathrm{A}_{MA} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & \dots & 0 & 0 \\ 0 & 1 & & & 0 & 0 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & 1 & 0 \end{bmatrix}; \quad \mathbf{b}_{MA} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \tag{B-3}$$

$$\mathbf{c}_{MA} = \begin{bmatrix} 1 & 1 & \dots & \dots & 1 & 1 \end{bmatrix} \cdot \frac{1}{n_s}$$

A_{MA} is an $(n_s \times n_s)$-matrix, $\mathbf{b}_{MA}$ is an $(n_s \times 1)$-vector, and $\mathbf{c}_{MA}$ is an $(1 \times n_s)$-vector. It is easy to see that the output y at time k will equal:

$$y(k) = \frac{u(k-1) + u(k-2) + \dots + u(k-n_s+1) + u(k-n_s)}{n_s} \tag{B-4}$$

i.e., y is the average of ten previously measured samples of u.

The filter is implemented in SIMULINK as a *'Masked' Discrete Transfer Fcn* block, in which the state matrices and the sample time are defined by the follo-

wing initialization lines:

```
ns   = @1;              % number of samples for averaging (user-input #1)
Ama  = [zeros(1,ns); [eye(ns-1) zeros(ns-1,1)]];
Bma  = [1; zeros(ns-1,1)];
Cma  = ones(1,ns)/ns;
Dma  = 0;
Ts   = @2;               % sample-time (user input #2)
uinit = @3;              % initial value of input signal (user input #3)
xinit = uinit * ones(1,ns);   % initial value of the state-vector of the
                               % discrete state-space model
```

When the block is *'Unmasked'*, other initialization lines can be found, but those additional lines are used only for creating the block-icon.

B.4 New Gain blocks.

Two new gain blocks have been included in the library *TOOLS*:

1 - *Matrix Gain.* Although SIMULINK does not contain a standard matrix-multiplication block which makes it possible to multiply the vector **u** with a constant matrix M:

$$\mathbf{y} = \mathrm{M} \cdot \mathbf{u} \tag{B-5}$$

it is possible to create such a matrix gain by applying a linear *State-space* block with an empty state matrix, thus reducing the state-space system to:

$$\mathbf{y} = \mathrm{D} \cdot \mathbf{u}$$

The block *Matrix Gain* is a therefore a *'Masked' State-space* block, which sets the state matrices to: A = [], B = [], C = [], D = M, where M must be specified by the user. An error message will appear if the dimensions of the input vector **u** and the matrix M do not match.

2 - *Scheduled Gain.* This block multiplies the *second* input signal with a gain which is a function of the *first* input. The internal structure of this block is visible in figure B-3. It can be seen that the *Gain Function* must be specified according to the required scheduling. The second input must be a scalar, the first input may be a vector. Unmask the block to find out how the icon was created.

B.5 New switch blocks.

The blocklibrary *TOOLS* also contains two new switch blocks:

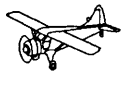
1 - *On/off switch.* This block switches between the second input or zero, depending upon the value of the first input signal, i.e., the second input is either turned on or off. If $u[1] = 0$, the output from the *On/off switch* will be zero; if $u[1] = 1$, the output will equal $u[2]$ (actually, the signal is

switched off if the input $u[1]$ is smaller than 0.5 and it turns on if it is larger or equal than 0.5, but it is recommended to use the values 0 or 1 for switching purposes). The internal structure of the *On/off*-switch is shown in figure B-4.

2 - *n-switch.* This block makes it possible to select one out of n inputs. In the library *TOOLS*, a *6-switch* is implemented, but it is easy to enhance the number of inputs if that is required. The first input to this block specifies the number of the inputsignal which should leave the *n-switch*: if $u[1] = i$, the $(i+1)^{th}$ input will be selected as the *n-switch* output. The internal structure of the *n-switch* is shown in figure B-5.

B.6 Conclusions.

This appendix has shown some blocks which are not contained in the standard SIMULINK blocklibraries, but which may be useful for some purposes. For unexperienced users, it is advised to try to *Unmask* these blocks, to find out how the *Mask*-function works. Using the *Mask* function, it is easy to enhance the *TOOLS* library with other blocks.



Appendix C. Detailed treatment of the SIMULINK models of the autopilot.

C.1 Introduction.

In this appendix, the details of the autopilot simulation structure, implemented in SIMULINK will be shown. Read chapter 3 if you just want to know the general structure of these SIMULINK models. Read part I of this report if you want to know more about the SIMULINK implementation of the aircraft dynamics, ILS and VOR signals, and wind and turbulence models.

The explanation of the blocks from this simulation structure is somewhat less strict than the discussion of the models from part I in appendix F of part I, because the SIMULINK implementation of the autopilot models merely serves as a demonstration of the practical use of the SIMULINK tools and models from part I, and it is not intended to be used for further standardization (at least, not in its current form). For the same reason, there are not as many helpfiles for the blocks from this system as there are for the Aircraft dynamics, VOR, ILS, wind, and turbulence blocks from part I. However, it shouldn't be very difficult to understand the structure of the simulation program if the block-diagrams from chapter 2 are used for comparison.

Note: in the next sections, lists of inputs and outputs for all blocks from the autopilot simulation structure have been included. The variable names, used in the systems are always included between brackets on the right side of the page. Tables C-1 to C-3 contain complete lists with all inputs and outputs of each subsystem block in the main system *APILOT3*. These tables have been included at the end of this appendix.

It is important to notice that the input and output ports have not always been numbered sequentially from top to bottom, in order to prevent that many input and output lines had to cross eachother. The meaning of the color schemes, used in the systems (shown in grey-scales in this report), is explained in the helpfile APHLP.M, which can be displayed by typing *help aphlp* at the MATLAB command-line.

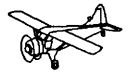
C.2 First level of the autopilot simulation structure.

Figure C-1 shows the first level of the autopilot simulation structure. The heart of this structure is formed by the block *Beaver dynamics*, in which the non-linear aircraft model is called. See part I of this report for a detailed discussion about the internal structure of the *S-function 'BEAVER'* [1]. In this appendix, the other subsystems from this main level of *APILOT3* will be examined.

Note: the systems *APILOT1* and *APILOT2* have almost the same internal structure, but some blocks have been omitted for reasons of computing speed. Here, we will zoom in to the subsystems of *APILOT3*, since that system contains the most complete simulation structure. Type *help aphlp* at the MATLAB command line for more info about the systems *APILOT1*, *APILOT2*, and *APILOT3*.

---

1) An *S-function* is a dynamic system which has been implemented in SIMULINK. This may either be a graphical block-diagram, or a M- or MEX-file with a special I/O structure. See chapter 3 of part I or ref.[2] for more details.



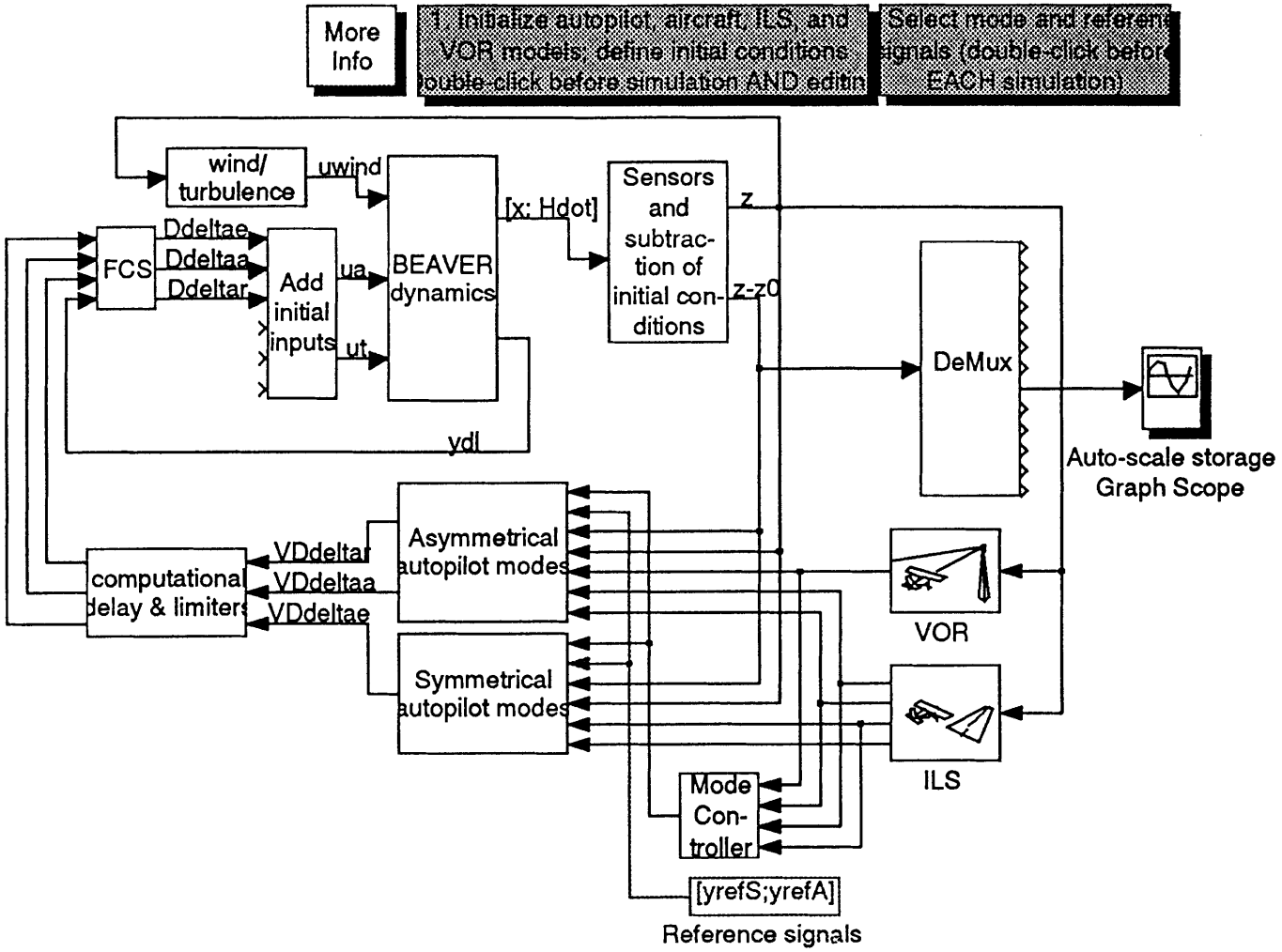


Figure C-1. Main level of the system *APILOT3*.
(APILOT3)

C.3 The subsystem *Beaver dynamics.*

Figure C-2 shows the internal structure of the subsystem *Beaver dynamics*. It can be seen that here the nonlinear model of the 'Beaver' dynamics, which has been described in part I of the report, is called by means of a SIMULINK *S-function* block. The *Mux* block combine the inputs into one vector, the *DeMux* block separates the output vector in two parts.

Inputs:
$\mathbf{u}_a = [\ \delta_e\ \ \delta_a\ \ \delta_r\ \ \delta_f\ ]^T$ (ua)
$\mathbf{u}_t = [\ n\ \ p_z\ ]^T$ (ut)
$\mathbf{u}_{wind} = [\ u_w\ \ v_w\ \ w_w\ \ \dot{u}_w\ \ \dot{v}_w\ \ \dot{w}_w\ ]^T$ (uwind)

Outputs:
$[\ \mathbf{x}\ ;\ \dot{H}\ ] = [\ V\ \ \alpha\ \ \beta\ \ p\ \ q\ \ r\ \ \psi\ \ \theta\ \ \varphi\ \ x_e\ \ y_e\ \ H,\ \ \dot{H}\ ]^T$ ([x;Hdot])
$\mathbf{y}_{dl} = [\ \frac{pb}{2V}\ \ \frac{q\bar{c}}{V}\ \ \frac{rb}{2V}\ ]^T$ (ydl)

Note: the vector $[\ \mathbf{x}\ ;\ \dot{H}\ ]$ will be called **z** after it has left the sensormodels! The datamatrices for the system *BEAVER* need to be loaded before starting a new simulation, by double-clicking the initialization button (*Load datamatrices for aircraft model*) or by calling the routine *APINIT* from the MATLAB command line (choose option 1). A description of the program APINIT.M can be found in section C.13.

C.4 The subsystem *FCS.*

This subsystem contains *State-space* blocks for the implementation of the linear, third order Flight Control System models, described in appendix A. The internal structure of this subsystem is shown in figure C-3.

Inputs:
$V\Delta\delta_e$ (Volt) (VDdeltae)
$V\Delta\delta_a$ (Volt) (VDdeltaa)
$V\Delta\delta_r$ (Volt) (VDdeltar)
$\mathbf{y}_{dl} = [\ \frac{pb}{2V}\ \ \frac{q\bar{c}}{V}\ \ \frac{rb}{2V}\ ]^T$ (ydl)

Outputs:
$\Delta\delta_e$ (rad) (Ddeltae)
$\Delta\delta_a$ (rad) (Ddeltaa)
$\Delta\delta_r$ (rad) (Ddeltar)

The state-space matrices A_{ail}, B_{ail}, C_{ail}, D_{ail}, A_{elv}, B_{elv}, C_{elv}, D_{elv}, A_{rud}, B_{rud}, C_{rud}, and D_{rud} must be loaded from the MAT-file FCS35.MAT, FCS45.MAT, or FCS55.MAT before starting a simulation. This can be done by applying the routine *APINIT* and choosing option 2 (*Load FCS models*). *APINIT* can be called from the MATLAB command line or started by double-clicking the initialization button in the block-diagram. See section C.13 for details.

C.5 The subsystem *Add initial inputs.*

This block is used to add the initial values of the aerodynamic and engine inputs to the commanded *deviations from the nominal control surface deflections*, which are generated by the control laws. Figure C-4 shows the internal

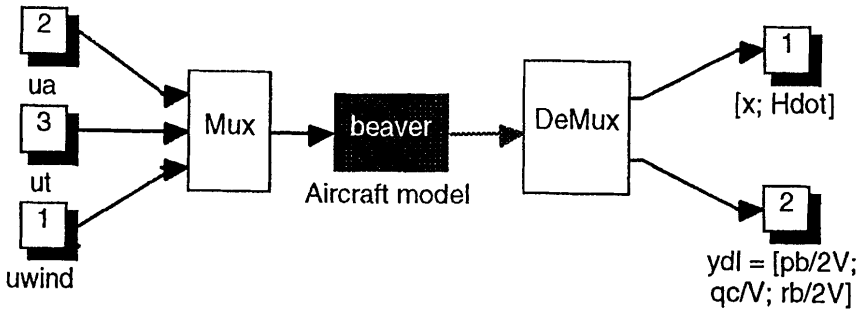


Figure C-2. Internal structure of the block *Beaver dynamics.*
(APILOT3 \ Beaver dynamics)

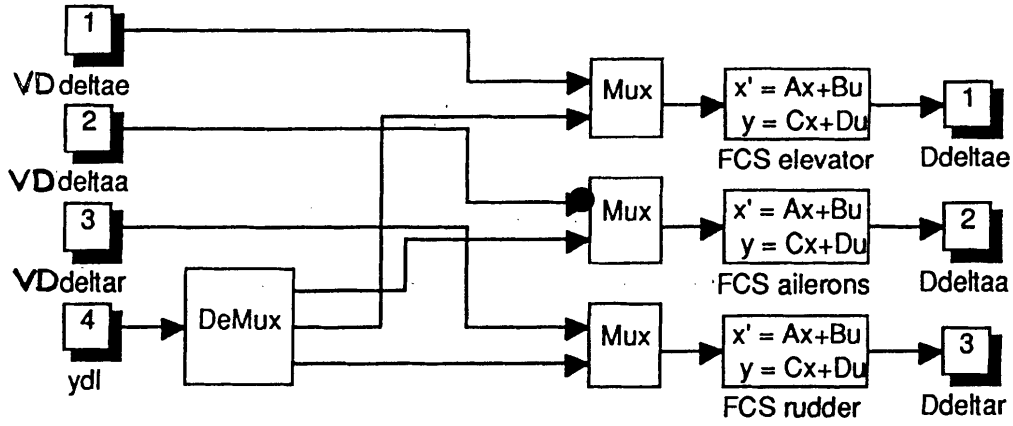


Figure C-3. Internal structure of the block *FCS.*
(APILOT3 \ FCS)

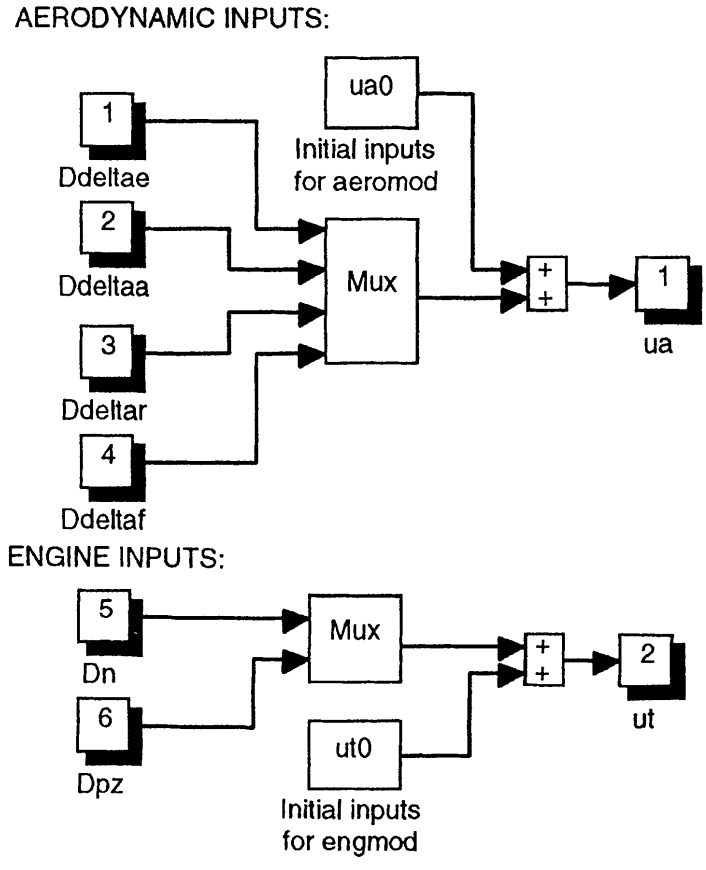


Figure C-4. Internal structure of the block *Add initial inputs.*
(APILOT3 \ Add initial inputs)

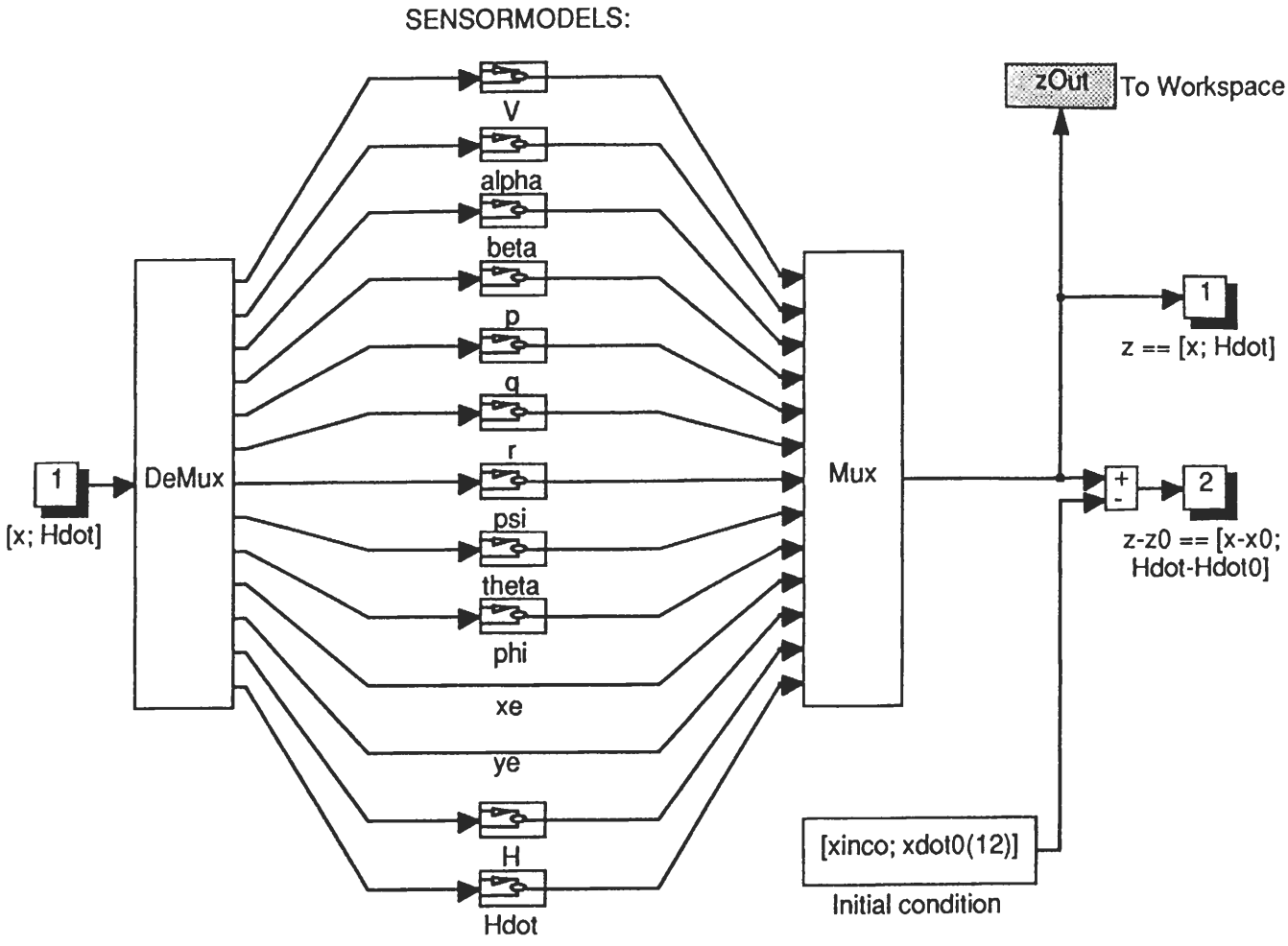


Figure C-5. Internal structure of the block *Sensors and subtraction of initial conditions.*
(APILOT3 \ Sensors and subtraction of initial conditions)

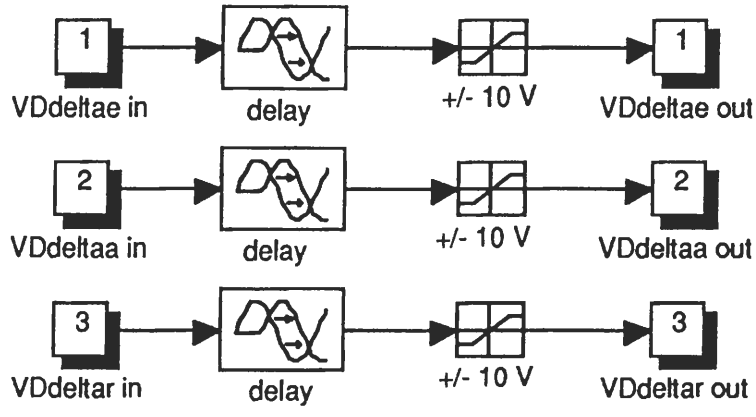


Figure C-6. Internal structure of the block *Computational delay & limiters.*
(APILOT3 \ Computational delay & limiters)

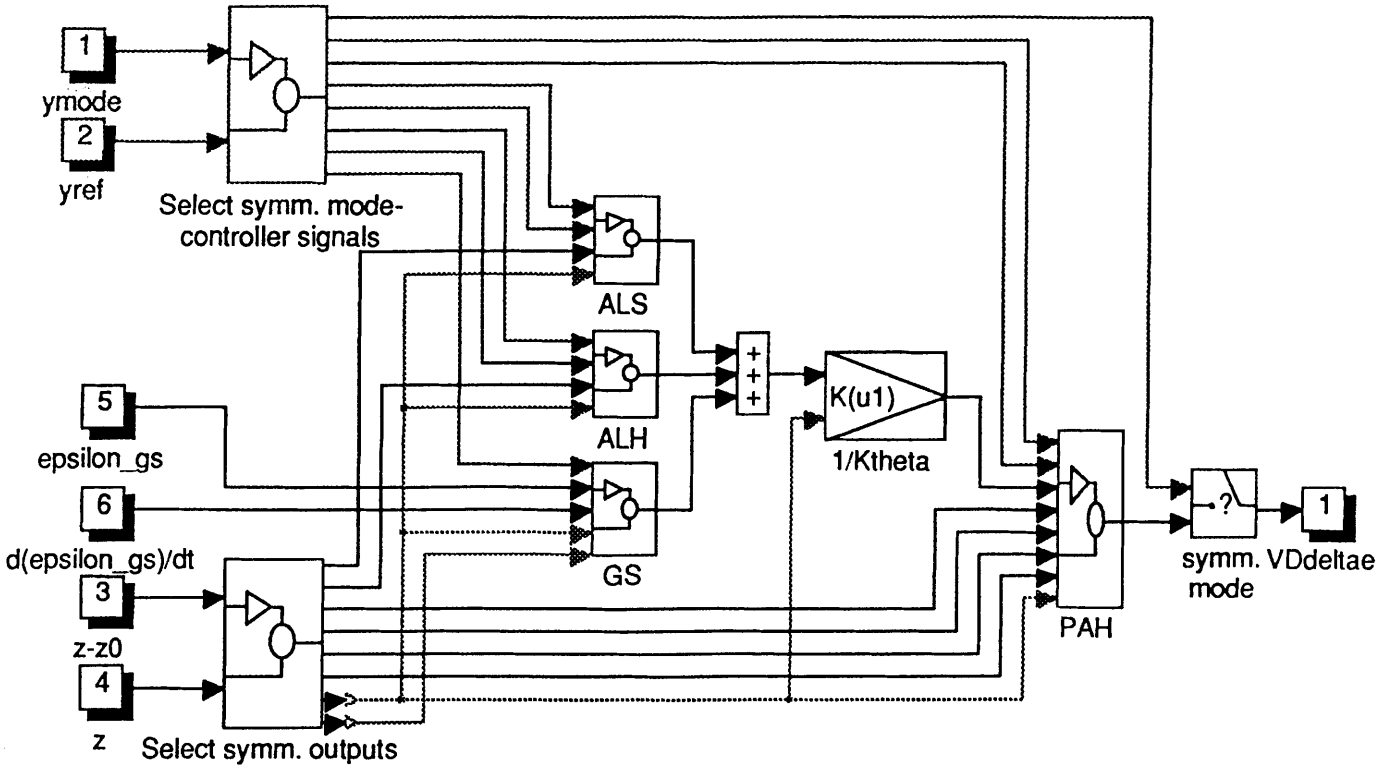


Figure C-7. Internal structure of the block *Symmetrical autopilot modes.*

(APILOT3 \ Symmetrical autopilot modes)

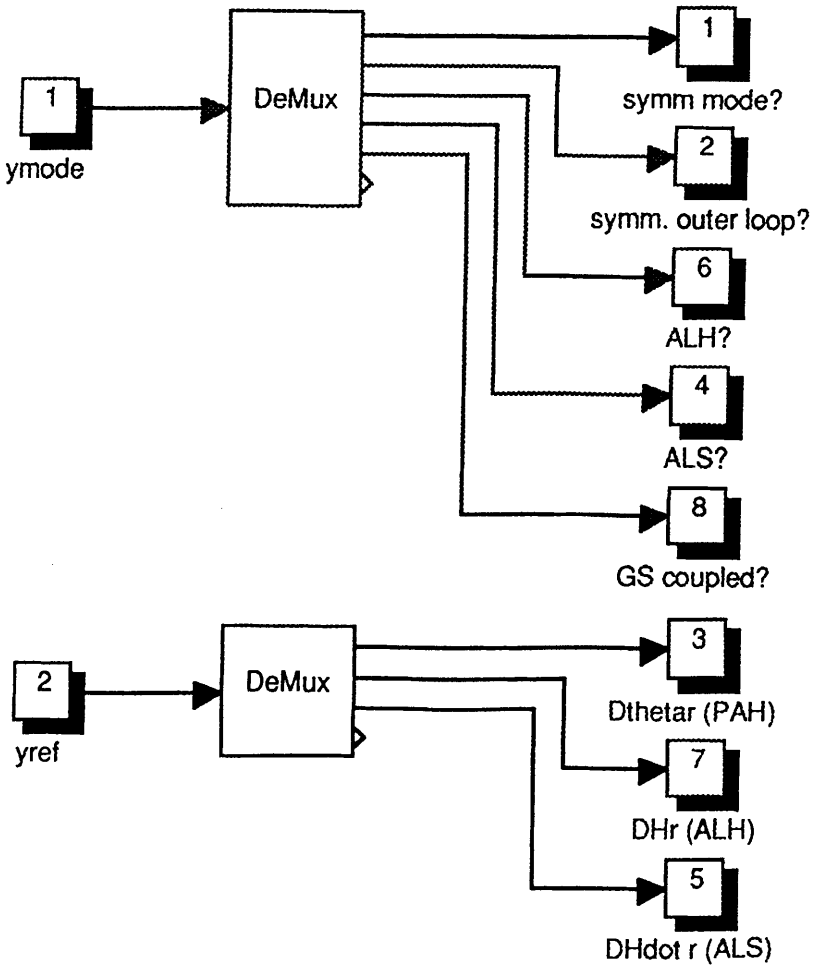


Figure C-8. Internal structure of the block *Select symm. mode-controller signals.*

(APILOT3 \ Symmetrical autopilot modes \ Select symm. modecontroller signals)

structure of this block. Here, only the control inputs to the elevator, ailerons, and rudder have actually been connected to this subsystem, but already input ports for changes in flap settings and engine inputs have been included for future enhancements.

Inputs:	$\Delta\delta_e$ (rad)	(Ddeltae)
	$\Delta\delta_a$ (rad)	(Ddeltaa)
	$\Delta\delta_r$ (rad)	(Ddeltar)
	$\Delta\delta_f$ (rad)	(Ddeltaf)
	Δn (RPM)	(Dn)
	Δp_z ("Hg)	(Dpz)
Outputs:	$\mathbf{u}_a = [\ \delta_e\ \ \delta_a\ \ \delta_r\ \ \delta_f\ ]^T$	(ua)
	$\mathbf{u}_t = [\ n\ \ p_z\ ]^T$	(ut)

The initial values of the inputvectors *ua0* = $\mathbf{u}_a(0)$, and *ut0* = $\mathbf{u}_t(0)$ must be loaded into the MATLAB workspace before starting new simulations. This can be done by double-clicking the initialization button in the SIMULINK system *APILOT3*, by calling the routine *APINIT* (choose option 3: *Define initial flight condition*) or by calling the aircraft trim program *ACTRIM* directly from the MATLAB command-line. See appendix E and section F.2.7 of part I for more information about the steady-state aircraft trim algorithm. See section C.13 for details about the MATLAB program APINIT.M.

C.6 The subsystem *Sensors and subtraction of initial conditions.*

This subsystem contains the dynamics of the sensors for the motion variables, coming from the block *Beaver dynamics*. Here, also initial values are subtracted from the signals to obtain *deviations from nominal values*. See figure C-5 for the internal structure of this block. The sensor models are contained in the subsystem blocks between *DeMux* and *Mux*. Currently, most of these subsystems are dummy blocks, in which the input is connected directly to the output, except for the altitude-sensor block, which contains a quantizer, Moving Average filter, and an additional (pressure-) transport-delay, and the airspeed and rate-of-climb blocks, which also have additional time-delays. It is easy to include 'real' sensor models for the other signals in the future.

Inputs:	$[\ \mathbf{x}\ ;\ \dot{H}\ ] = [\ V\ \ \alpha\ \ \beta\ \ p\ \ q\ \ r\ \ \psi\ \ \theta\ \ \varphi\ \ x_e\ \ y_e\ \ H,\ \ \dot{H}\ ]^T$	([x;Hdot])
Outputs:	$\mathbf{z}$ = [$\mathbf{x}$; $\dot{H}$] after sensors	(z)
	$\mathbf{z} - \mathbf{z}_0$ = $\mathbf{z} - \mathbf{z}(0)$	(z-z0)

The initial conditions *xinco* = $\mathbf{x}_{inco}$ = $\mathbf{x}(0)$ and *xdot0* = $\dot{\mathbf{x}}(0)$ must be defined in the MATLAB workspace before starting a new simulation. Trimmed-flight conditions (i.e., all accelerations and rotational accelerations are zero) can be computed by calling the function *ACTRIM* from the command line, by calling the function *APINIT* from the MATLAB command line, or by double-clicking the initialization button in the blockdiagram (use option 3, *Define initial flight condition*). More info about the trim algorithm and the program ACTRIM.M can be found in appendix E and section F.2.7 of part I. See section C.13 for more info on ACINIT.M.

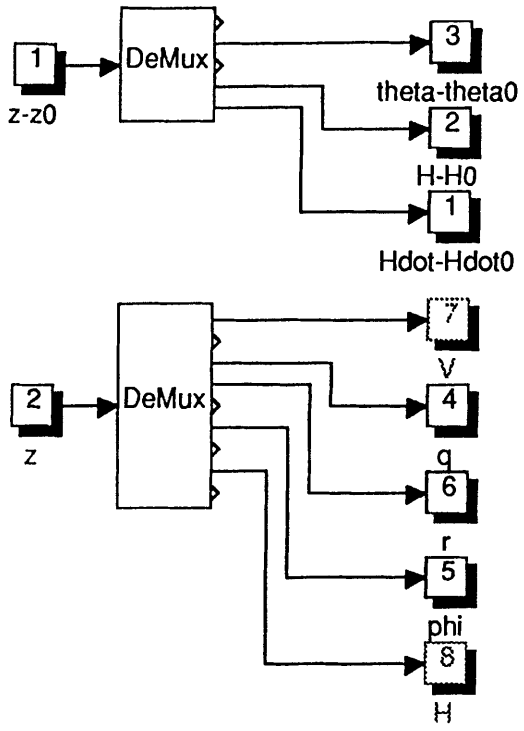


Figure C-9. Internal structure of the block *Select symm. outputs.*
(APILOT3 \ Symmetrical autopilot modes \ Select symm. outputs)

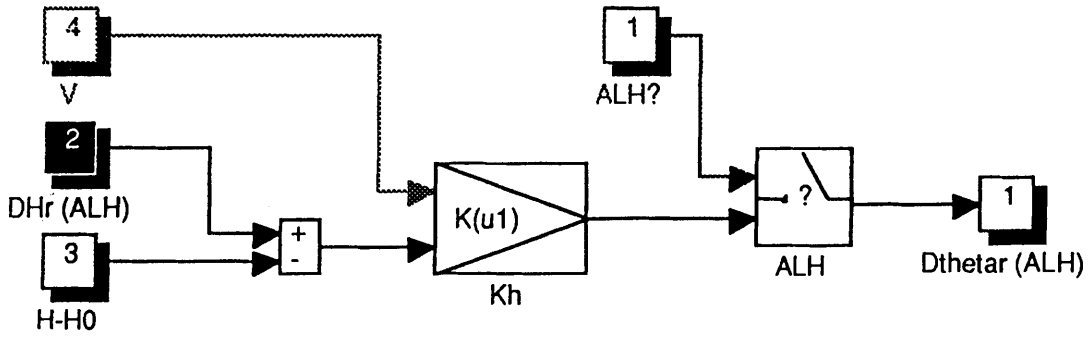


Figure C-10. Internal structure of the block *ALH.*
(APILOT3 \ Symmetrical autopilot modes \ ALH)

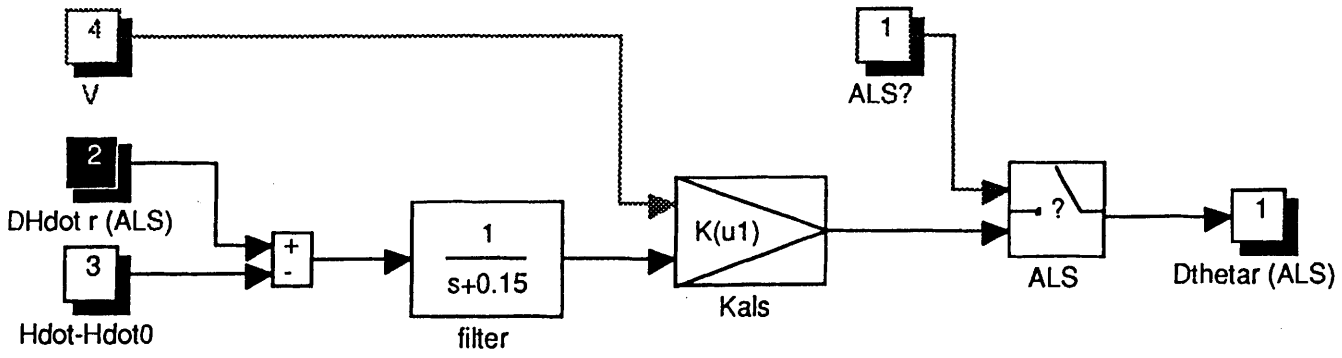


Figure C-11. Internal structure of the block *ALS.*
(APILOT3 \ Symmetrical autopilot modes \ ALS)

C.7 The subsystem *Computational delay & limiters.*

Figure C-6 shows the internal structure of the block *Computational delay & limiters.* This block implements a time-delay of 0.03 seconds, which is equal to one sampling time T_s (in reality, the computational delay may be shorter, so this is the 'worst-case' situation). The signal limiters represent the maximum and minimum possible values of the signals leaving the ROLM FCC, which are equal to plus and minus 10 Volt. Under normal circumstances, these limits will never be exceeded, because the limiters in the Pitch Attitude Hold and Roll Attitude Hold laws already make sure that the magnitude of the commanded signals never exceeds these limits of the ROLM computer, hence, the ± 10 V limiters have been included for sake of completeness only.

Inputs:	$V\Delta\delta_e$ (Volt)	(VDdeltae in)
	$V\Delta\delta_a$ (Volt)	(VDdeltaa in)
	$V\Delta\delta_r$ (Volt)	(VDdeltar in)
Outputs:	$V\Delta\delta_e$ (limited and shifted in time)	(VDdeltae out)
	$V\Delta\delta_a$ (limited and shifted in time)	(VDdeltaa out)
	$V\Delta\delta_r$ (limited and shifted in time)	(VDdeltar out)

C.8 The subsystem *Symmetrical autopilot modes.*

This subsystem has been shown in figure C-7; a list of inputs and outputs is contained in table C-2. The different control laws have been implemented separately, yielding a modular structure, in which the division between inner and outer loops (or guidance and control functions) is clearly visible. Apart from the subsystems *ALS*, *ALH*, *GS*, and *PAH*, there are two blocks which sort signals coming from the *Mode Controller* and *Reference Values* blocks and signals from the block *Sensors and subtraction of initial condition*, respectively.

An additional gain *1/Ktheta* (= $1/K_\theta$) has been included, because the position of the gain K_θ in the θ-loop of the PAH mode differs from the position of K_θ in the symmetrical outer-loop modes ALH, ALS, and GS (see figures 2-1, 2-2, 2-3, and 2-6 in chapter 2). Due to the correction gain $1/K_\theta$, it has become easier to implement the PAH control law as inner-loop of the ALH, ALS, and GS coupled laws. The *On/off switch* in the output line can be used to create simulations of the aircraft, equipped with lateral autopilot modes only. In the real autopilot it is not possible to switch off the lateral or longitudinal modes separately, but for analytical purposes, this option may be useful.

Inputs:	$\mathbf{y}_{mode}$	= [symm. mode on/off, symm. outer-loop on/off, ALH on/off, ALS on/off, GS-coupled on/off, asymm. mode on/off, asymm. outer-loop on/off, HH on/off, NAV on/off, LOC-coupled on/off]^T	(ymode)
	$\mathbf{y}_{ref}$	= [$\Delta\theta_r$(PAH), ΔH_r(ALH), $\Delta\dot{H}_r$(ALS), $\Delta\varphi_r$(RAH) $\Delta\psi_r$(HH)]^T	(yref)
	$\mathbf{z}$	= [$\mathbf{x}$; $\dot{H}$]^T after sensors	(z)
	$\mathbf{z} - \mathbf{z}_0$	= $\mathbf{z} - \mathbf{z}(0)$	(z-z0)
	$\boldsymbol{\varepsilon}_{gs}$		(epsilon_gs)
	$\dot{\boldsymbol{\varepsilon}}_{gs}$		(d(epsilon_gs)/dt)
Output:	$V\Delta\delta_e$ (Volt)		(VDdeltae)

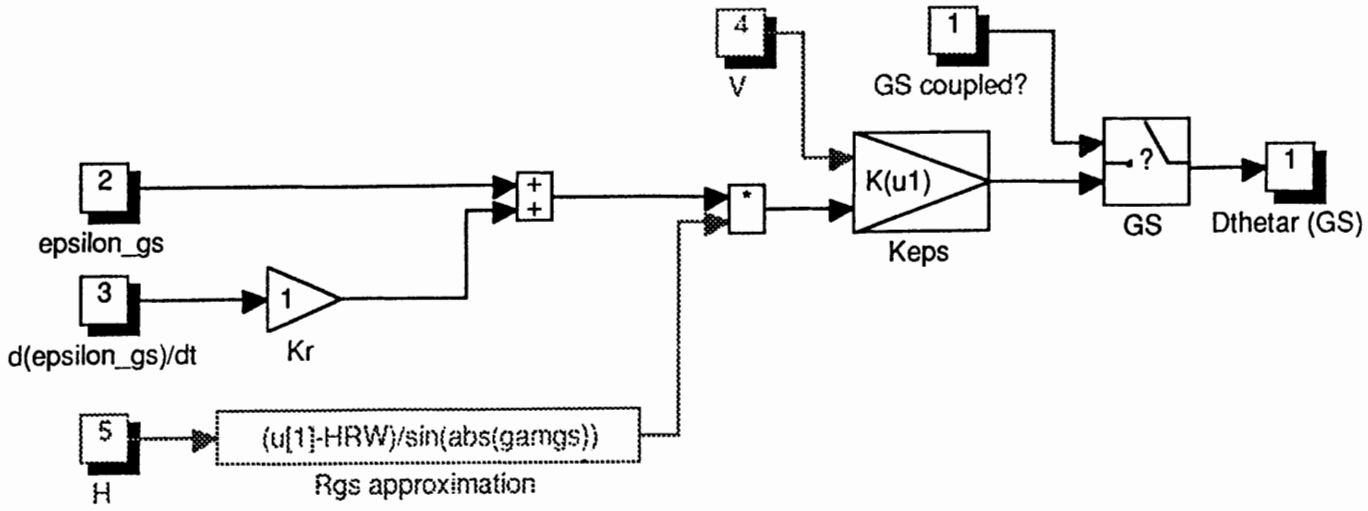


Figure C-12. Internal structure of the block *GS*.
(APILOT3 \ Symmetrical autopilot modes \ GS)

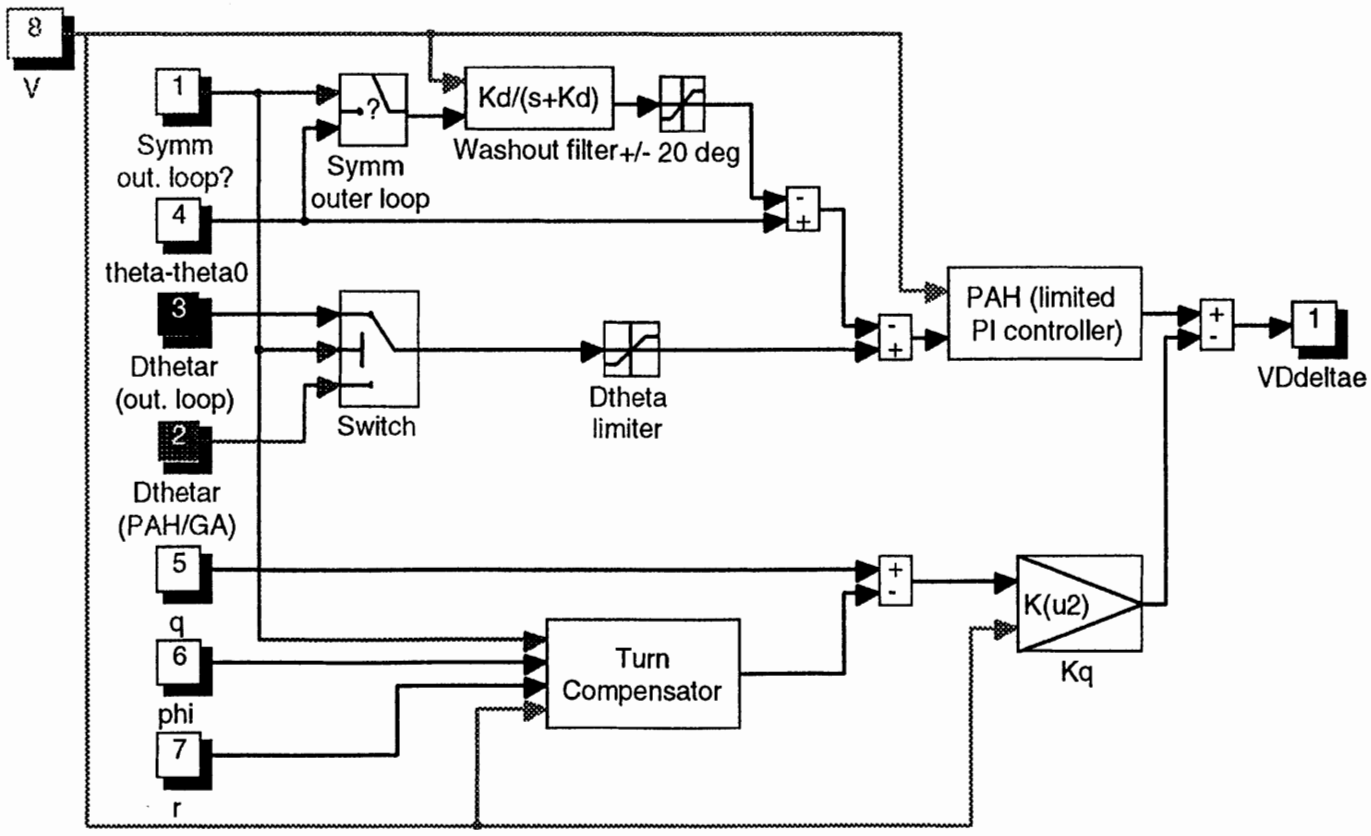


Figure C-13. Internal structure of the block *PAH*.
(APILOT3 \ Symmetrical autopilot modes \ PAH)

Note: *ymode* and *yref* contain the signals from the Mode-controller, which determine the magnitude of the reference signals and the settings of the switches. These switches (that is: the elements, denoted with *'on/off'* in the text and *'?'* in the block-diagrams) have values which are either one (= 'on'), or zero (= 'off'). The internal structure of the *Scheduled Gains* from the control laws will not be shown here; see table 2-1 of chapter 2 for a list of gain-scheduling functions. A general description of the *Scheduled Gain* block can be found in section B.4.

The block *Symmetrical autopilot modes* contains the following subsystems:

1 - *Select symm. modecontroller signals.* This block selects the elements from the modecontroller vectors *ymode* = $\mathbf{y}_{mode}$ and *yref* = $\mathbf{y}_{ref}$, used by the symmetrical autopilot modes. The internal structure of this subsystem is shown in figure C-8.

Inputs:	$\mathbf{y}_{mode}$ (see definition above)	(ymode)
	$\mathbf{y}_{ref}$ (see definition above)	(yref)
Outputs:	symm. mode on/off	(symm. mode?)
	symm. outer-loop on/off	(symm. outer-loop?)
	ALH on/off	(ALH?)
	ALS on/off	(ALS?)
	GS-coupled on/off	(GS?)
	$\Delta\theta_r$ (PAH)	(Dthetar(PAH))
	$\Delta\dot{H}_r$ (ALS)	(DHdot r(ALS))
	ΔH_r (ALH)	(DHr(ALH))

2 - *Select symm. outputs.* This block selects the measured outputs (i.e., the signals, coming from the sensor models) which are needed by the symmetrical autopilot modes. The term 'symmetrical' in the title may be a little confusing, because the roll-angle φ and the yaw-rate r are also selected, because those variables are needed for the turncompensation control law. See figure C-9 for the internal structure of this block.

The control laws of the 'Beaver' autopilot use *deviations from the nominal values* of the state variables and the rate of climb as control inputs. However, the actual values of some state variables are still needed, for instance by the gain-scheduling functions which use the actual value of the true airspeed V in stead of the deviation ΔV.

Inputs:	$\mathbf{z} - \mathbf{z}_0$ (see definitions above)	(z-z0)
	$\mathbf{z}$ (see definitions above)	(z)
Outputs:	$\theta - \theta_0$	(theta-theta0)
	$H - H_0$	(H-H0)
	$\dot{H} - \dot{H}_0$	(Hdot-Hdot0)
	V	(V)
	q	(q)
	r	(r)
	φ	(phi)
	H	(H)

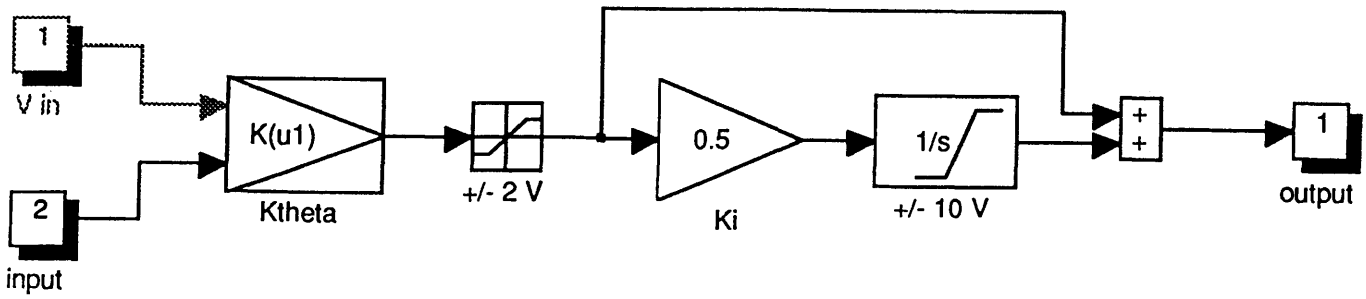


Figure C-14. Internal structure of the block *PI-controller PAH.*
(APILOT3 \ Symmetrical autopilot modes \ PAH \ PI-controller PAH)

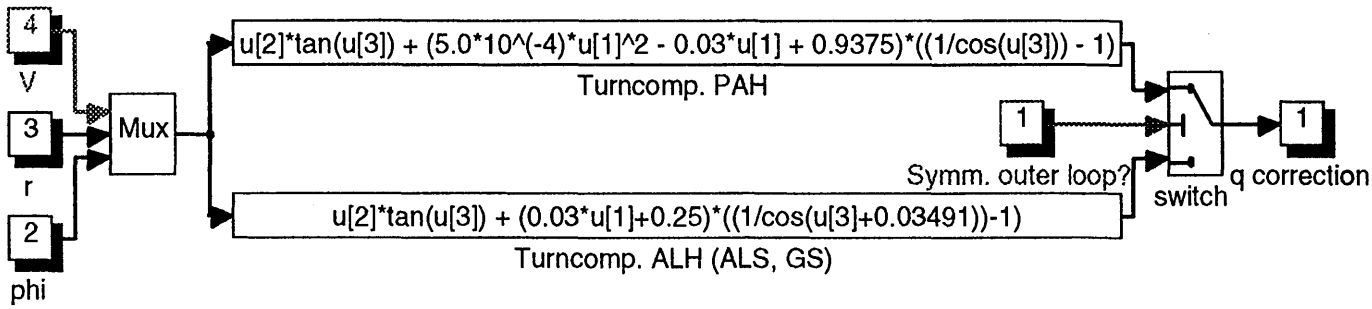


Figure C-15. Internal structure of the block *Turncompensator.*
(APILOT3 \ Symmetrical autopilot modes \ PAH \ Turncompensator)

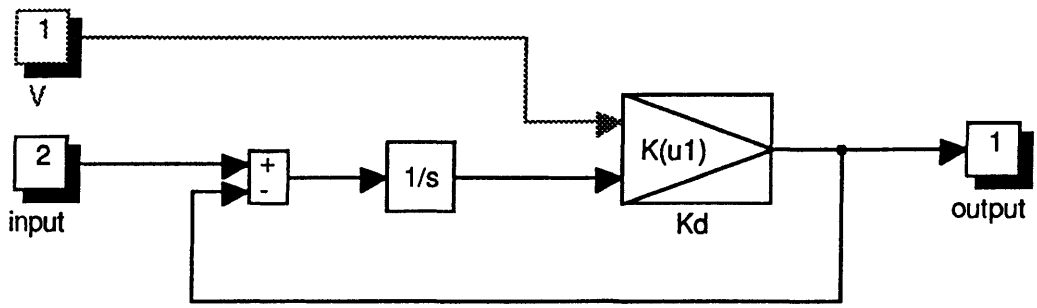


Figure C-16. Internal structure of the block *Washout filter.*
(APILOT3 \ Symmetrical autopilot modes \ PAH \ Washout filter)

3 - *ALH.* This block contains the control logic for the outer-loop of the Altitude Hold mode. The reference change in pitch-angle will be send further to the subsystem *PAH* in which the complete inner-loop logic is contained (including the washout-filter in the θ-loop and a turncompensation law which is valid only for the outer-loop modes *ALH*, *ALS*, and *GS*). See figure C-10.

Inputs:	ALH on/off	(ALH?)
	ΔH_r(ALH)	(DHr(ALH))
	$H - H_0 = H - H(0)$	(H-H0)
	V	(V)
Output:	$\Delta\theta_r$(ALH)	(Dthetar(ALH))

4 - *ALS.* This block contains the control logic for the outer-loop of the Altitude Select mode. The reference change in pitch-angle $\Delta\theta_{ref}$ is sent further to the block *PAH*, which also contains the washout-filter in the θ-loop and the control laws for turncompensation. The internal structure of *ALS* is depicted in figure C-11.

Inputs:	ALS on/off	(ALS?)
	$\Delta\dot{H}_r$(ALS)	(DHdot r (ALS))
	$\dot{H} - \dot{H}_0 = \dot{H} - \dot{H}(0)$	(Hdot-Hdot0)
	V	(V)
Output:	$\Delta\theta_r$(ALS)	(Dthetar(ALS))

5 - *GS.* This block contains the control logic for the outer-loop of the Glideslope-coupled mode, see figure C-12. The reference change of pitch-angle, $\Delta\theta_{ref}$, is computed using the angular deviation from the glideslope transmitter (ε_{gs}) and its time-derivative. Notice the gain-scheduling with *altitude*, which is used as an estimation of the distance from the aircraft to the glideslope antenna. *HRW* is the height of the runway in metres above sea-level, *gamgs* = γ_{gs} is the reference value of the glidepath angle if the aircraft follows the reference line (usually, $\gamma_{gs} = -3°$, although it doesn't matter if you specify *gamgs* = +3°). See refs.[1] and [6] and appendix C of part I for more details about the ILS system.

Before starting a new simulation, run *APINIT* by calling it from the MATLAB command-line or by double-clicking the initialization button in the main level of *APILOT3*, to make sure that the ILS parameters are set properly (use option 4: *Initialize VOR and/or ILS systems*). It is recommended to use *APINIT* also before editing the system, because SIMULINK may beep in some subsystems if some parameters are not defined, even when this is not of real concern for editing. See also sections C.11 and C.13.

Inputs:	GS coupled on/off	(GS?)
	ε_{gs}	(epsilon_gs)
	$\dot{\varepsilon}_{gs}$	(d(epsilon_gs)/dt)
	V	(V)
	H	(H)
Output:	$\Delta\theta_r$(GS)	(Dthetar(GS))

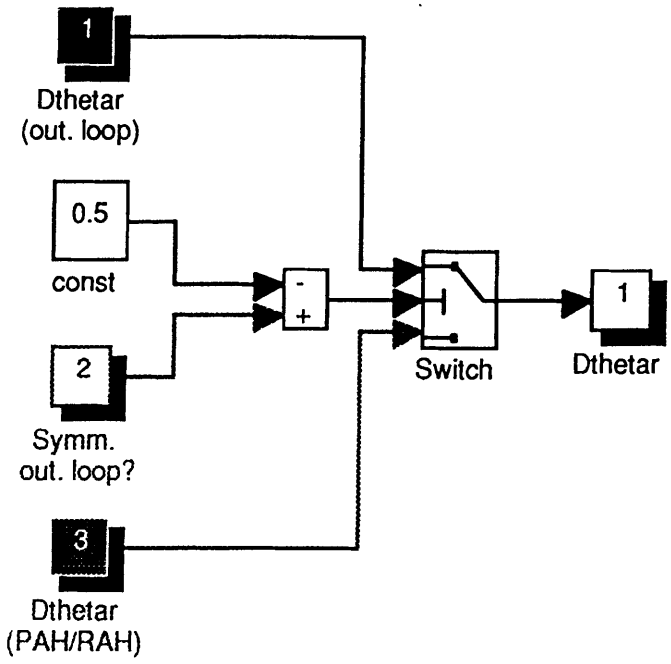


Figure C-17. Internal structure of the block *Switch*.
(APILOT3 \ Symmetrical autopilot modes \ PAH \Switch)

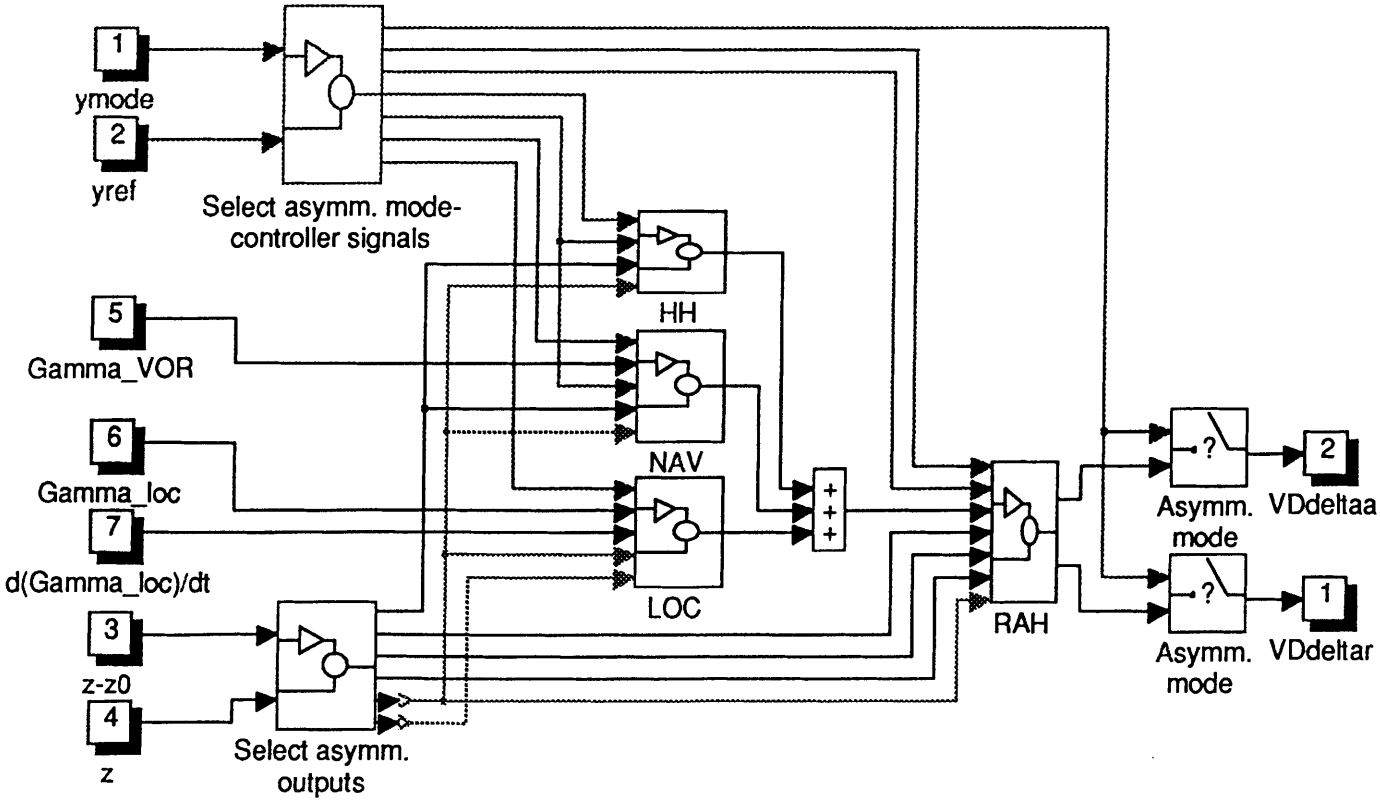


Figure C-18. Internal structure of the block ASYMMETRICAL AUTOPILOT MODES.
(APILOT3 \ Asymmetrical autopilot modes)

6 - *PAH*. This block contains the Pitch Attitude Hold control law, the turncompensation loops, and a washout filter with varying coefficients for the symmetrical outer-loops. The block-diagram is shown in figure C-13. The PI-controller for the PAH mode, the turncompensation logic, the washout filter, and the switch have been implemented in separate subsystems, which are shown in figures C-14, C-15, C-16 and C-17.

To implement the washout filter with coefficients that depend upon the true airspeed, the theory from section D.4 of part I has been applied. In that section, it is shown how a transfer function can be converted into a state-space model, and how the state-space model in turn can be converted into a blockdiagram with first-order integrators and gains only. Figure C-16 shows the resulting block-diagram for the varying washout filter.

Inputs to *PAH*:	Symmetrical outer-loop on/off	(Symm. out.loop?)
	$\Delta\theta_r$ (PAH)	(Dthetar (PAH))
	$\Delta\theta_r$ (ALH / ALS / GS)	(Dthetar (out.loop))
	$\theta - \theta_0 = \theta - \theta(0)$	(theta-theta0)
	q	(q)
	φ	(phi)
	r	(r)
	V	(V)
Output:	$V\Delta\delta_e$ (Volt)	(VDdeltae)

C.9 The subsystem *Asymmetrical autopilot modes.*

The block-diagram of this subsystem is shown in figure C-18. Table C-3, which has been included at the end of this appendix, gives the complete list of inputs and outputs to and from this subsystem. The different lateral control laws have been implemented separately, yielding a clear division between inner and outer-loops. Apart from the subsystems *HH*, *NAV*, *LOC*, and *RAH*, there are two blocks in which signals from the *Mode Controller* and signals, coming from the block *Sensors and subtraction of initial condition*, are sorted. An on/off switch for the total lateral controller has been included, even though this option is not contained in the actual autopilot. Turning off the lateral control loops independently from the longitudinal autopilot modes may be useful for some analytical purposes.

Inputs:	$\mathbf{y}_{mode}$	= [symm. mode on/off, symm. outer-loop on/off, ALH on/off, ALS on/off, GS-coupled on/off, asymm. mode on/off, asymm. outer-loop on/off, HH on/off, NAV on/off, LOC-coupled on/off]^T	(ymode)
	$\mathbf{y}_{ref}$	= [$\Delta\theta_r$ (PAH), ΔH_r (ALH), $\Delta\dot{H}_r$ (ALS), $\Delta\varphi_r$ (RAH) $\Delta\psi_r$ (HH)]^T	(yref)
	$\mathbf{z}$	= [$\mathbf{x}$; $\dot{H}$]^T after sensors	(z)
	$\mathbf{z} - \mathbf{z}_0$	= $\mathbf{z} - \mathbf{z}(0)$	(z-z0)
	Γ_{VOR}		(Gamma_VOR)
	Γ_{loc}		(Gamma_loc)
	$\dot{\Gamma}_{loc}$		(d(Gamma_loc)/dt)

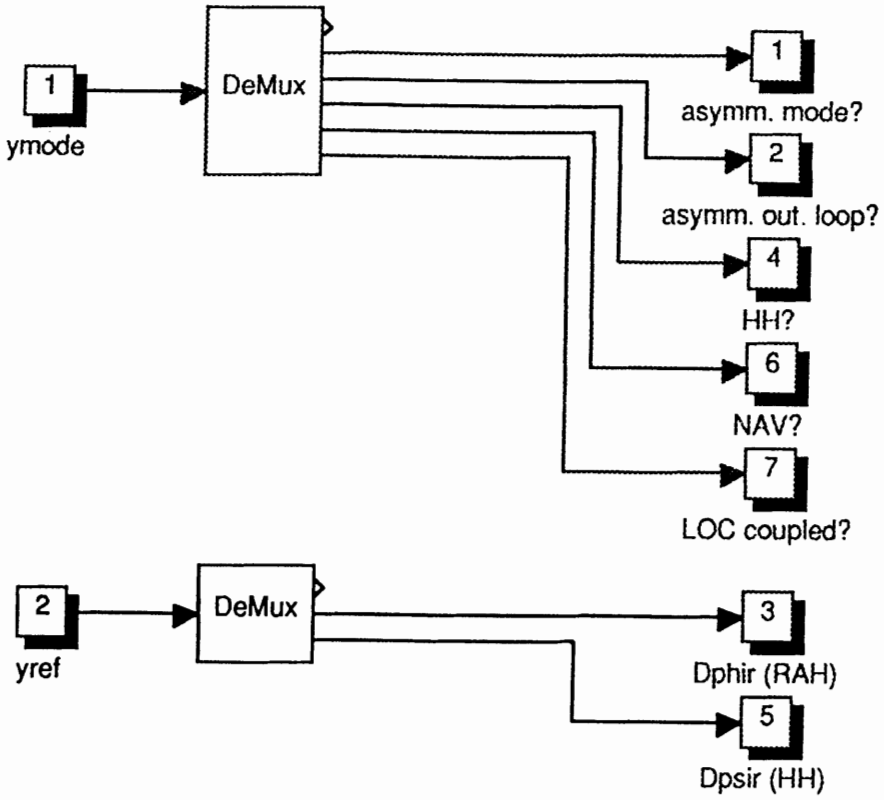


Figure C-19. Internal structure of the block *Select asymm. mode-controller signals.*

(APILOT3 \ Asymmetrical autopilot modes \ Select asymm. modecontroller signals)

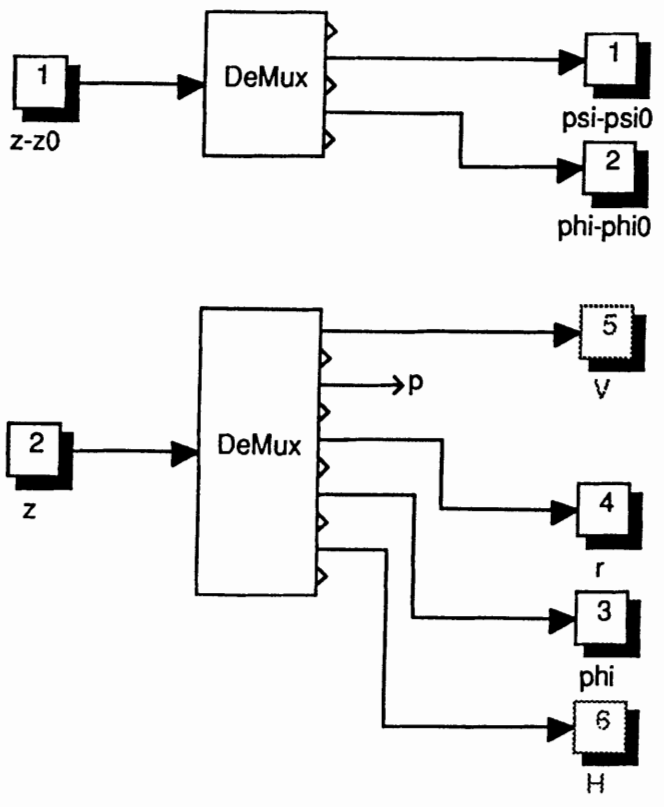


Figure C-20. Internal structure of the block *Select asymm. outputs.*

(APILOT3 \ Asymmetrical autopilot modes \ Select asymm. outputs)

Outputs:	$V\Delta\delta_a$ (Volt)	(VDdeltaa)
	$V\Delta\delta_r$ (Volt)	(VDdeltar)

Note: *ymode* and *yref* contain the signals from the *Mode Controller*, which determines the magnitude of the reference signals and the settings of the switches. The internal structure of the *Scheduled Gains* from the control laws has not been shown here; a list of gain-scheduling functions has been included in table 2-1 of chapter 1. A general description of the *Scheduled Gain* block can be found in section B.4.

The block *Asymmetrical autopilot modes* contains the following subsystems:

1 - *Select asymm. modecontroller signals.* This block selects the elements from the mode controller vectors *ymode* = $\mathbf{y}_{mode}$ and *yref* = $\mathbf{y}_{ref}$, used by the lateral autopilot modes. The internal structure of this subsystem is shown in figure C-19.

Inputs:	$\mathbf{y}_{mode}$ (see definition above)	(ymode)
	$\mathbf{y}_{ref}$ (see definition above)	(yref)
Outputs:	asymm. mode on/off	(asymm. mode?)
	asymm. outer-loop on/off	(asymm. outer-loop?)
	HH on/off	(HH?)
	NAV on/off	(NAV?)
	LOC-coupled on/off	(LOC?)
	$\Delta\varphi_r$ (RAH)	(Dphir(RAH))
	$\Delta\psi_r$ (HH)	(Dpsir(HH))

2 - *Select asymm. outputs.* This block selects the measured outputs (i.e., the signals, coming from the sensor models) which are needed by the lateral autopilot modes, see figure C-20. The altitude H and true airspeed V are also selected, because these quantities are needed for gain-scheduling. Note: the RAH-law uses the deviation $\varphi - \varphi_0$ for controlling the roll-angle, but the real value of φ is needed for turncoordination. For this reason, both φ and $\varphi - \varphi_0$ are selected in the block *Select asymm. outputs*. Under normal circumstances, φ will be equal to $\varphi - \varphi_0$, because the initial condition in which the autopilot is switched on will usually be level-flight.

Inputs:	$\mathbf{z} - \mathbf{z}_0$ (see definition above)	(z-z0)
	$\mathbf{z}$ (see definition above)	(z)
Outputs:	$\psi - \psi_0$	(psi-psi0)
	$\varphi - \varphi_0$	(phi-phi0)
	V	(V)
	r	(r)
	φ	(phi)
	H	(H)

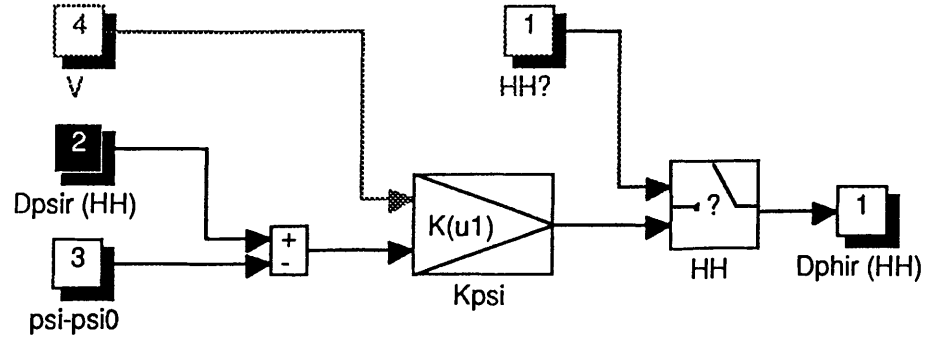


Figure C-21. Internal structure of the block *HH*.
(APILOT3 \ Asymmetrical autopilot modes \ HH)

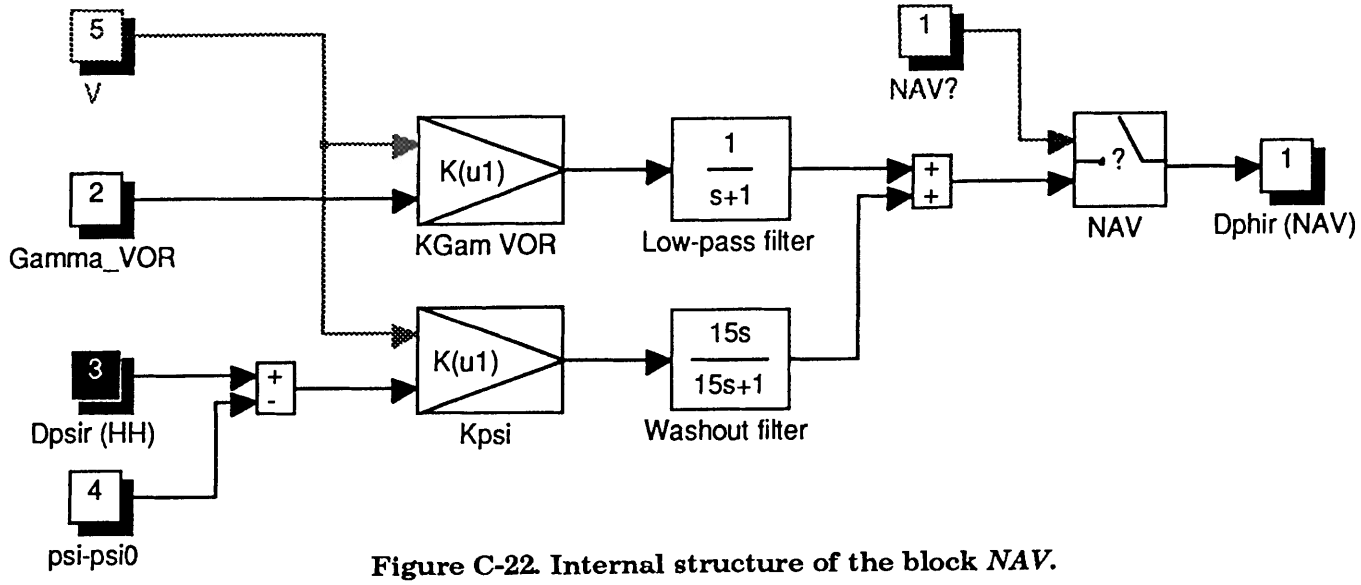


Figure C-22. Internal structure of the block *NAV*.
(APILOT3 \ Asymmetrical autopilot modes \ NAV)

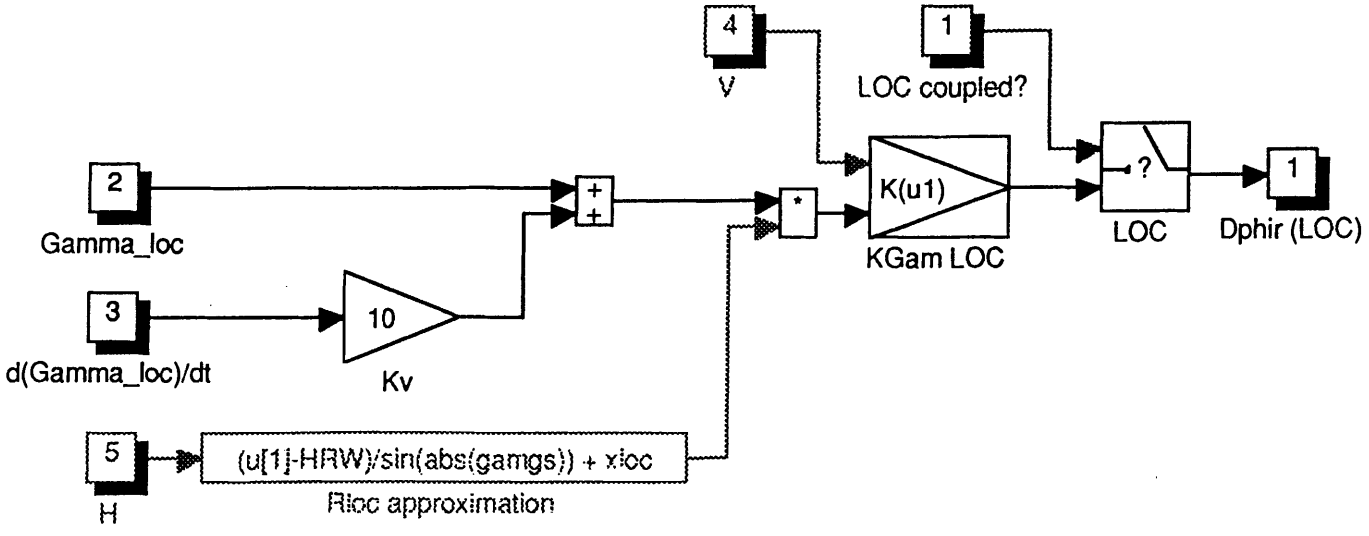


Figure C-23. Internal structure of the block *LOC*.
(APILOT3 \ Asymmetrical autopilot modes \ LOC)

3 - *HH.* This block contains the outer-loop of the Heading Hold / Heading Select mode. It generates a reference roll-angle, which will be sent to the block *RAH*, where the control surface deflections will be computed. Figure C-21 shows the internal structure of *HH*.

Inputs:	HH on/off	(HH?)
	$\Delta\psi_r$(HH)	(Dpsir(HH))
	$\psi - \psi_0 = \psi - \psi(0)$	(psi-psi0)
	V	(V)
Output:	$\Delta\varphi_r$(HH)	(Dphir(HH))

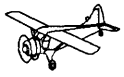
4 - *NAV.* This block contains the control logic of the outer-loop of the Navigation mode. The resulting reference value of the roll-angle is sent to the block *RAH*. *NAV* uses the signal Γ_{VOR} from the *VOR* block, which is equal to the angle between the VOR-bearing where the aircraft flies and the reference VOR-bearing (Course Datum). See figure C-22 for the internal structure or the subsystem *NAV*.

Inputs:	NAV on/off	(NAV?)
	Γ_{VOR}	(Gamma_VOR)
	$\Delta\psi_r$(HH)	(Dpsir (HH))
	$\psi - \psi_0 = \psi - \psi(0)$	(psi-psi0)
	V	(V)

The NAV-mode itself doesn't need to be initialized, but the VOR-block which generates the inputsignals to this block does. So use *APINIT* before starting new simulations by double-clicking the initialization block in the blockdiagram, or by calling it directly from the MATLAB command line. See sections C.11 and C.13 for more details.

5 - *LOC.* In this block, the control logic for the Localizer-coupled mode is contained. The *LOC*-mode uses the angular deviation from the localizer reference plane (Γ_{loc}) and its time-derivative to compute the reference value of the roll-angle which will be sent to the *RAH* block. The subsystem contains a gain-scheduling block, which uses the altitude H to estimate the distance to the runway, which gives accurate results if the aircraft flies along the nominal glidepath. *HRW* is the runway altitude above sea-level (H_{RW}), *gamgs* is the nominal glidepath angle (γ_{gs}), and *xloc* is the distance from the runway threshold to the Localizer antenna (x_{loc}). See figure 2-9 of chapter 2 for the definition of Γ_{loc}. More details about the ILS guidance system can be found in refs.[1] and [6] and in appendix C of part I. The internal structure of the block *LOC* is shown in figure C-23.

Before starting a new simulation, specify the parameters for the ILS block and LOC gain-scheduling by double-clicking the initialization block in the main level of *APILOT3*, or by calling the routine *APINIT* directly from the MATLAB command line (use option 4: *Initialize VOR and/or ILS systems*). See sections C.11 and C.13 for details. It is recommended to apply *APINIT* also before editing, because SIMULINK tends to produce nasty beeps when parameters are not defined in the MATLAB workspace, even if it doesn't actually matter during system building or editing.



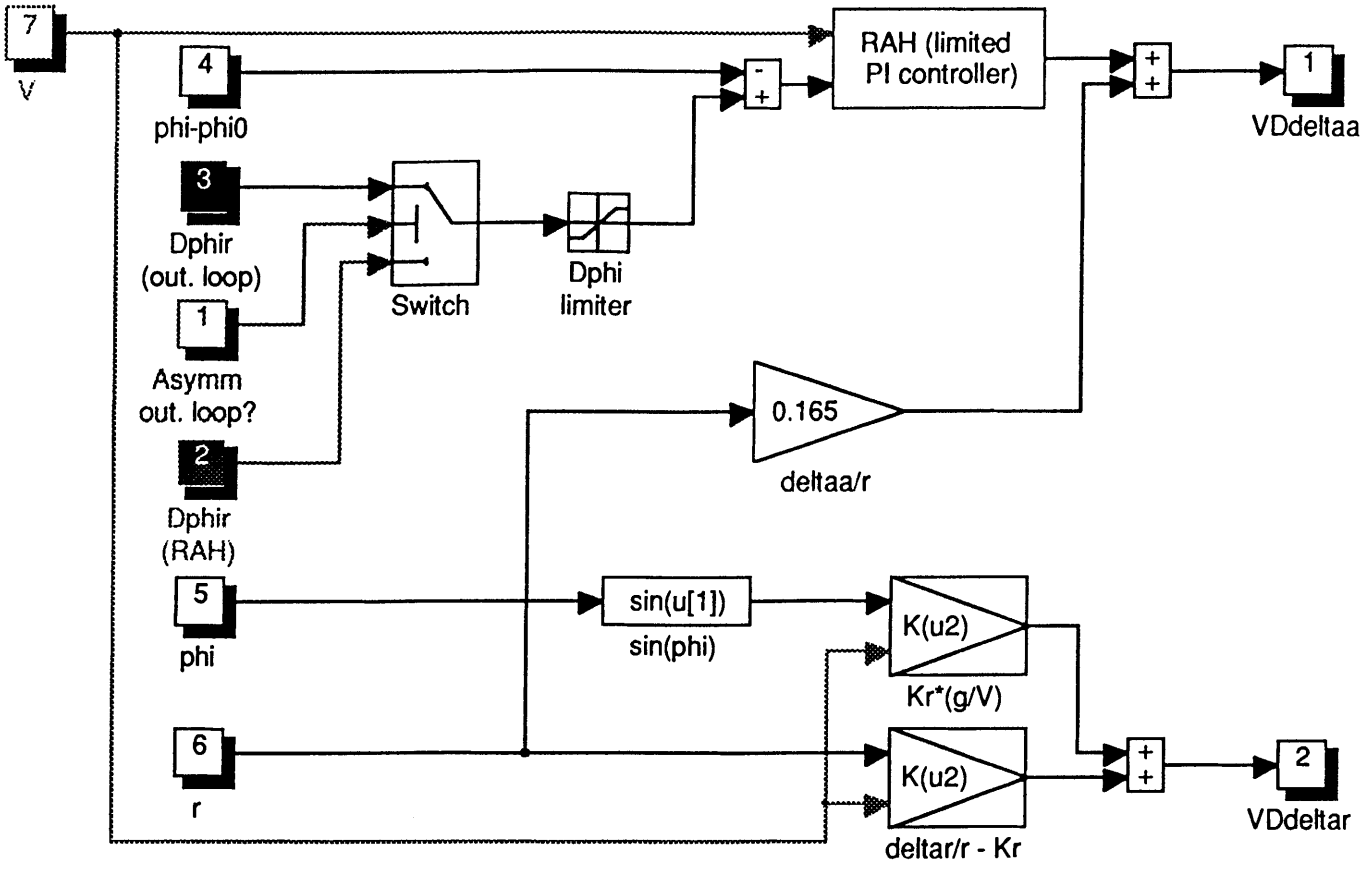


Figure C-24. Internal structure of the block *RAH*.
(APILOT3 \ Asymmetrical autopilot modes \ RAH)

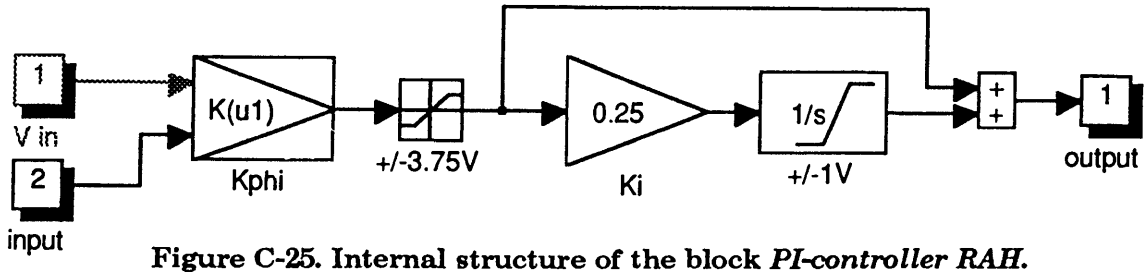


Figure C-25. Internal structure of the block *PI-controller RAH*.
(APILOT3 \ Asymmetrical autopilot modes \ RAH \ PI-controller RAH)

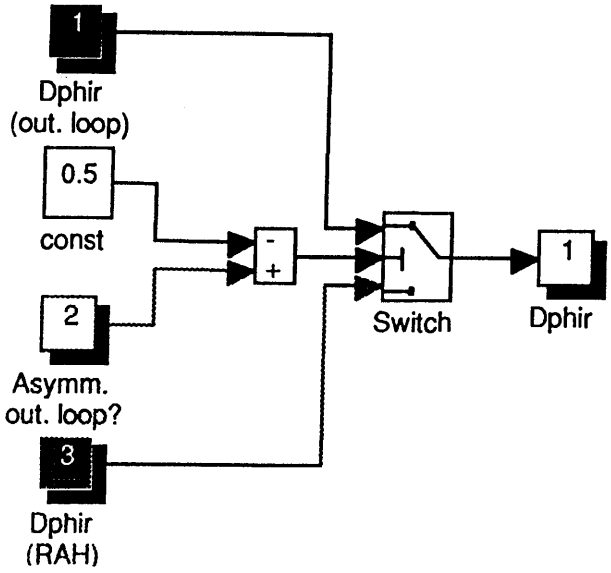


Figure C-26. Internal structure of the block *Switch*.
(APILOT3 \ Asymmetrical autopilot modes \ RAH \ Switch)

Inputs:	LOC-coupled on/off	(LOC coupled?)
	Γ_{loc}	(Gamma_LOC)
	$\dot{\Gamma}_{loc}$	(d(Gamma_LOC)/dt)
	V	(V)
	H	(H)
Output:	$\Delta\varphi_r$(LOC)	(Dphir(LOC))

6 - *RAH*. This block contains the control logic for Roll Attitude Hold, which is also used as inner-loop for the Heading Hold, Navigation, and Localizer modes. The structure of this block is shown in figure C-24, figure C-25 shows the PI-controller of the RAH mode, and figure C-26 shows the internal structure of the block *Switch*. The turncoordination loops are visible in the lower right corner of figure C-24.

Inputs to *RAH*:	Asymmetrical outer-loop on/off	(Asymm. out.loop on/off)
	$\Delta\varphi_r$(RAH)	(Dphir(ALH))
	$\Delta\varphi_r$(HH / NAV / LOC)	(Dphir(out.loop))
	$\varphi - \varphi_0 = \varphi - \varphi(0)$	(phi-phi0)
	φ	(phi)
	r	(r)
	V	(V)
Outputs:	$V\Delta\delta_a$ (Volt)	(VDdeltaa)
	$V\Delta\delta_r$ (Volt)	(VDdeltar)

C.10 The subsystem *Mode Controller* and the block *Reference Signals.*

The settings of the switches from the control laws and the reference inputs for these controllers need to be entered in the subsystems *Symmetrical autopilot modes* and *Asymmetrical autopilot modes* by means of the inputvectors $\mathbf{y}_{mode}$ and $\mathbf{y}_{ref}$, which are defined as:

$\mathbf{y}_{mode}$ = [symm. mode on/off, symm. outer-loop on/off, ALH on/off, ALS on/off, GS-coupled on/off, asymm. mode on/off, asymm. outer-loop on/off, HH on/off, NAV on/off, LOC-coupled on/off]^T (ymode)

$\mathbf{y}_{ref}$ = [$\Delta\theta_r$(PAH), ΔH_r(ALH), $\Delta\dot{H}_r$(ALS), $\Delta\varphi_r$(RAH), $\Delta\psi_r$(HH)]^T (yref)

For most simulations, these vectors will have constant elements. For instance: if a simulation of the Altitude Hold control law for a step input ΔH_{ref} = 1 m is required for the situation that the lateral autopilot mode is Roll Attitude Hold with a roll-angle of zero degrees, these vectors will equal:

$\mathbf{y}_{mode}$ = [1 1 1 0 0 | 1 0 0 0 0]^T,

$\mathbf{y}_{ref}$ = [0 1 0 | 0 0]^T.

However, if an autopilot mode with an *Armed* and a *Coupled* phase needs to be evaluated, it is necessary to switch between two settings, depending upon a certain switch-criterion. This task is fulfilled by the block *Mode Controller*, which generates the switch-settings vector $\mathbf{y}_{mode}$. A block-diagram of this subsystem is shown in figure C-27.

It can be seen that the vector $\mathbf{y}_{mode}$ consists of a symmetrical and an asymmetrical part. The switches from this diagram are used to switch between Glideslope Armed and Coupled, Localizer Armed and Coupled, Navigation Armed and Coupled, and Localizer or Navigation modes. The switches are controlled by the outputs from the blocks *Matlab Fcn*, which call small MATLAB subroutines, which check the switch-criteria, listed in section 3.5.

Inputs to the subsystem *Mode Controller* (used for mode-switching criteria):

Γ_{VOR}	Gamma_VOR
Γ_{loc}	Gamma_loc
$\dot{\Gamma}_{loc}$	d(Gamma_loc)/dt
ε_{gs}	epsilon_gs

Outputvector:

$\mathbf{y}_{mode}$	ymode

Moreover, the block *Mode Controller* expects the following variables to be defined in the MATLAB workspace:

ymod1S	symmetrical part of $\mathbf{y}_{mode}$, valid for the main symmetrical autopilot mode (*PAH, ALH, ALS, GS-Coupled*)
ymod2S	symmetrical part of $\mathbf{y}_{mode}$, valid for the symmetrical submodes (*GS Armed*)
ymod1A	asymmetrical part of $\mathbf{y}_{mode}$, valid for the main asymmetrical autopilot mode (*RAH, HH, LOC-coupled, NAV-coupled*)
ymod2A	asymmetrical part of $\mathbf{y}_{mode}$, valid for asymmetrical submodes (*LOC-coupled, NAV-coupled*)

These variables will be set automatically if the routine *APSTART* is called from the MATLAB command line or by double-clicking the block *Select mode and reference signals* on top of the main level of *APILOT3*. The listing of *APSTART* is given at the end of this section.

The MATLAB programs GSSWITCH.M, LOCSWITCH.M and NAVSWTCH.M, which are called in the *Matlab Fcn* blocks in figure C-27, contain the switch-criteria for the *GS*, *LOC*, and *NAV* modes. Global variables *GSswitched*, *NAVswitched*, and *LOCswitched* are used to check if the particular mode has switched from Armed to Coupled already. If it has, the corresponding variable will be set to one, otherwise it will be zero. In this way, it has been made sure that the autopilot will not switch back from Coupled to Armed. Before *each* simulation, these three variables must be reset, which will be done automatically if the routine *APSTART* is used (call it from the MATLAB command line, or by double-clicking the block *Select mode and reference signals* in the main-level of *APILOT3*. This routine has been listed at the end of this section.

The program listings of these MATLAB subroutines are given below.

Listing of *GSswitch.m:*

```
function gscrit = gsswitch(u);
%-------------------------------------------------------------
% GSSWITCH.m contains the switch-criterion for switching from
% Glideslope-Armed to Glideslope-Coupled.
% NOTE: BEFORE EACH SIMULATION, RUN APSTART TO RESET THE
% GSswitched-VARIABLE!
%-------------------------------------------------------------

% u = epsilon_gs

if GSswitched == 0  % Check criterion only if not already switched from
                    % Armed to Coupled.
   if abs(u) < 0.0001 % Aircraft crosses the glideslope reference line
      GSswitched = 1;
   end
end

gscrit = GSswitched - 0.5; % Negative if Armed, positive if Coupled.
gscrit = ymod1S(5) * gscrit * [1 1 1 1 1];

%-------------------------------------------------------------
% Programmed by: Marc Rauw, September 10, 1993.
```

Listing of *NAVswtch.m:*

```
function navcrit = navswtch(u);
%-------------------------------------------------------------
% NAVSWTCH.m contains the switch-criterion for switching from
% Navigation-Armed to Navigation-Coupled.
% NOTE: BEFORE EACH SIMULATION, RUN APSTART TO RESET THE
% NAVswitched-VARIABLE!
%-------------------------------------------------------------

% u = Gamma_VOR

if NAVswitched == 0  % Check criterion only if not already switched from
                     % Armed to Coupled.
   if abs(u) < 0.0001 % Aircraft crosses the reference VOR bearing
      NAVswitched = 1;
   end
end

navcrit = NAVswitched - 0.5; % Negative if Armed, positive if Coupled.
navcrit = ymod1A(4) * navcrit * [1 1 1 1 1]; % Zero if NAV-mode not selected.

%-------------------------------------------------------------
% Programmed by: Marc Rauw, September 10, 1993.
```

Listing of *LOCswtch.m:*

```
function loccrit = locswtch(u);
%-------------------------------------------------------------
% LOCSWTCH.m contains the switch-criterion for switching from
% Localizer-Armed to Localizer-Coupled.
% NOTE: BEFORE EACH SIMULATION, RUN APSTART TO RESET THE
% LOCswitched-VARIABLE!
%-------------------------------------------------------------

% u = [Gamma_loc  d(Gamma_loc)/dt]

if LOCswitched == 0  % Check criterion only if not already switched from
                     % Armed to Coupled.
   % Localizer switch criterion
   % --------------------------
   if ((10*u(2)+u(1))>0.0001 & u(1)<0) | ((10*u(2)+u(1))<-0.0001 & u(1)>0)
      LOCswitched = 1;
   end
end

loccrit = LOCswitched - 0.5; % Negative if Armed, positive if Coupled.
loccrit = ymod1A(5) * loccrit * [1 1 1 1 1]; % Zero if LOC-mode not selected.

%-------------------------------------------------------------
% Programmed by: Marc Rauw, September 10, 1993.
```

Listing of *APSTART.m:*

```
%=========================================================================
% Mode-switch signals:
%
% ymodS = [Symm. mode on/off; Symm. outer-loop on/off; ALH on/off;
%          ALS on/off; GS on/off]
% ymodA = [Asymm. mode on/off; Asymm. outer-loop on/off; HH on/off;
%          NAV on/off; LOC on/off]
%
% ymod1S contains the setting for the main symmetrical mode, ymod2S con-
% tains the setting for the Glideslope-armed phase of symm. approach.
%
% ymod1A contains the setting for the main asymmetrical mode, ymod2A
% contains the setting for the NAV-armed and Localizer-armed phase of the
% navigation mode and asymm. approach, respectively.
%=========================================================================
% Reference signals:
%
% yrefS = [Dtheta_ref; DH_ref; DHdot_ref] contains reference values for
%            PAH, ALH, and ALS modes, respectively. Only constant reference
%            values are used here (corresponding to step at t=0 if the
%            values are not equal to zero), so change the block "Reference
%            Signals" in the system APILOT1, APILOT2, or APILOT3 if you
%            want other inputs.
%
% yrefA = [Dphi_ref; Dpsi_ref] contains reference values for RAH and HH modes.
%=========================================================================

global GSswitched NAVswitched LOCswitched ymod1S ymod2S ymod1A ymod2A
disp('');
disp('Choose Autopilot Mode');
disp('====================');
disp('');
```

```
% SYMMETRICAL AUTOPILOT MODE AND REFERENCE SIGNALS.
% ------------------------------------------------
ymod1S = [0 0 0 0 0]';  % Symmetrical mode-switch vector for main mode
ymod2S = [0 0 0 0 0]';  % Symmetrical mode-switch vector for submode
                        %                (used only for Glideslope Armed = ALH)
yrefS  = [0 0 0]';      % Symmetrical reference values

smode  = menu('Specify symmetrical autopilot mode','Pitch Attitude Hold',...
         'Altitude Hold','Altitude Select','Glideslope Capture & Hold',...
         'No symmetrical autopilot mode');

disp('');
if smode == 1                           % Pitch Attitude Hold
   disp('PAH selected');
   ymod1S = [1 0 0 0 0]'; % PAH
   PAHopt = menu('What reference signal do you want?','Step input Dtheta_ref',...
                 'Dtheta_ref = 0');
   disp('');
   if PAHopt == 1
      Dthetar = input('Give desired theta-step [deg]: ');
      if isempty(Dthetar)
         Dthetar = 0;
      end
      Dthetar = Dthetar*pi/180;
   else
      disp('Dtheta_ref = 0 set.');
      Dthetar = 0;
   end
   yrefS= [Dthetar 0 0]'; % reference pitch angle: theta_ref = theta0+Dthetar
   clear Dthetar PAHopt

elseif smode == 2                       % Altitude Hold
   disp('ALH selected');
   ymod1S = [1 1 1 0 0]'; % ALH
   ALHopt = menu('What reference signal do you want?','Step input DH_ref',...
                 'DH_ref = 0');
   disp('');
   if ALHopt == 1
      DHr = input('Give desired altitude-step [m]: ');
      if isempty(DHr)
         DHr = 0;
      end
   else
      disp('DH_ref = 0 set.');
      DHr = 0;
   end
   yrefS = [0 DHr 0]'; % reference altitude: H_ref = H0 + DHr
   clear DHr ALHopt

elseif smode == 3                       % Altitude Select
   disp('ALS selected');
   ymod1S = [1 0 1 0 0]'; % ALS
   ALSopt = menu('What reference signal do you want?',...
                 'Step input DHdot_ref','DHdot_ref = 0');
   disp('');
   if ALSopt == 1
      DHdotr = input('Give desired Hdot-step [m/s]: ');
      if isempty(DHdotr)
         DHdotr = 0;
      end
   else
      disp('DHdot_ref = 0 set.');
      DHdotr = 0;
   end
   yrefS = [0 0 DHdotr]'; % reference rate-of-climb: Hdot_ref = Hdot0 + DHdotr
   clear DHdotr ALSopt

elseif smode == 4                       % Glideslope Capture & Hold
   disp('GS selected');
   ymod1S = [1 1 0 0 1]'; % GS-coupled
   ymod2S = [1 1 1 0 0]'; % ALH for GS-armed
   % Note: DHr = 0 (Armed phase), glideslope-coupled reference signal
   % comes from ILS block.
else
```

```
   disp('No symmetrical mode selected');
   % Keep ymod1S = [0 0 0 0 0]', and ymod2S = [0 0 0 0 0]'.
end

% ASYMMETRICAL AUTOPILOT MODE AND REFERENCE SIGNALS.
% -------------------------------------------------
ymod1A = [0 0 0 0 0]';  % Asymmetrical mode-switch vector for main mode
ymod2A = [0 0 0 0 0]';  % Asymmetrical mode-switch vector for submode
                        %         (used for LOC-armed = HH and NAV-armed = HH)
yrefA  = [0 0]';        % Asymmetrical reference values

amode = menu('Specify asymmetrical autopilot mode','Roll Attitude Hold',...
             'Heading Hold/Heading Select','Navigation mode',...
             'Localizer Capture & Hold','No asymmetrical autopilot mode');

disp('');
if amode == 1                           % Roll Attitude Hold
   disp('RAH selected');
   ymod1A = [1 0 0 0 0]'; % RAH
   RAHopt = menu('What reference signal do you want?','Step input Dphi_ref',...
         'Dphi_ref = 0');
   disp('');
   if RAHopt == 1
      Dphir = input('Give desired phi-step [deg]: ');
      if isempty(Dphir)
         Dphir = 0;
      end
      Dphir = Dphir*pi/180;
   else
      disp('Dphi_ref = 0 set.');
      Dphir = 0;
   end
   yrefA = [Dphir 0]'; % Reference roll-angle: phi_ref = phi0 + Dphir
   clear Dphir RAHopt

elseif amode == 2                       % Heading Hold / Heading Select
   disp('HH selected');
   ymod1A = [1 1 1 0 0]'; % HH
   HHopt  = menu('What reference signal do you want?',...
                 'Step input Dpsi_ref','Dpsi_ref = 0');
   disp('');
   if HHopt == 1
      Dpsir = input('Give desired psi-step [deg]: ');
      if isempty(Dpsir)
         Dpsir = 0;
      end
      Dpsir = Dpsir*pi/180;
   else
      disp('Dpsi_ref = 0 set.');
      Dpsir = 0;
   end
   yrefA = [0 Dpsir]'; % Reference yaw-angle = reference heading:
                       %                                 psi_ref = psi0 + Dpsir
   clear Dpsir HHopt

elseif amode == 3                       % Navigation mode
   disp('NAV selected');
   ymod1A = [1 1 0 1 0]'; % NAV-coupled
   ymod2A = [1 1 1 0 0]'; % NAV-armed = HH
   % Note: Dpsir = 0 (Armed phase), reference signal for NAV-coupled comes
   % from VOR block .
elseif amode == 4                       % Localizer mode
   disp('LOC selected');
   ymod1A = [1 1 0 0 1]'; % LOC-coupled
   ymod2A = [1 1 1 0 0]'; % LOC-armed = HH
   % Note: Dpsir = 0 (Armed phase), reference signal for LOC-coupled comes
   % from ILS block.
else
   disp('No asymmetrical mode selected');
   % Keep ymod1A = [0 0 0 0 0]' and ymod2A = [0 0 0 0 0]'
end

clear smode amode
```

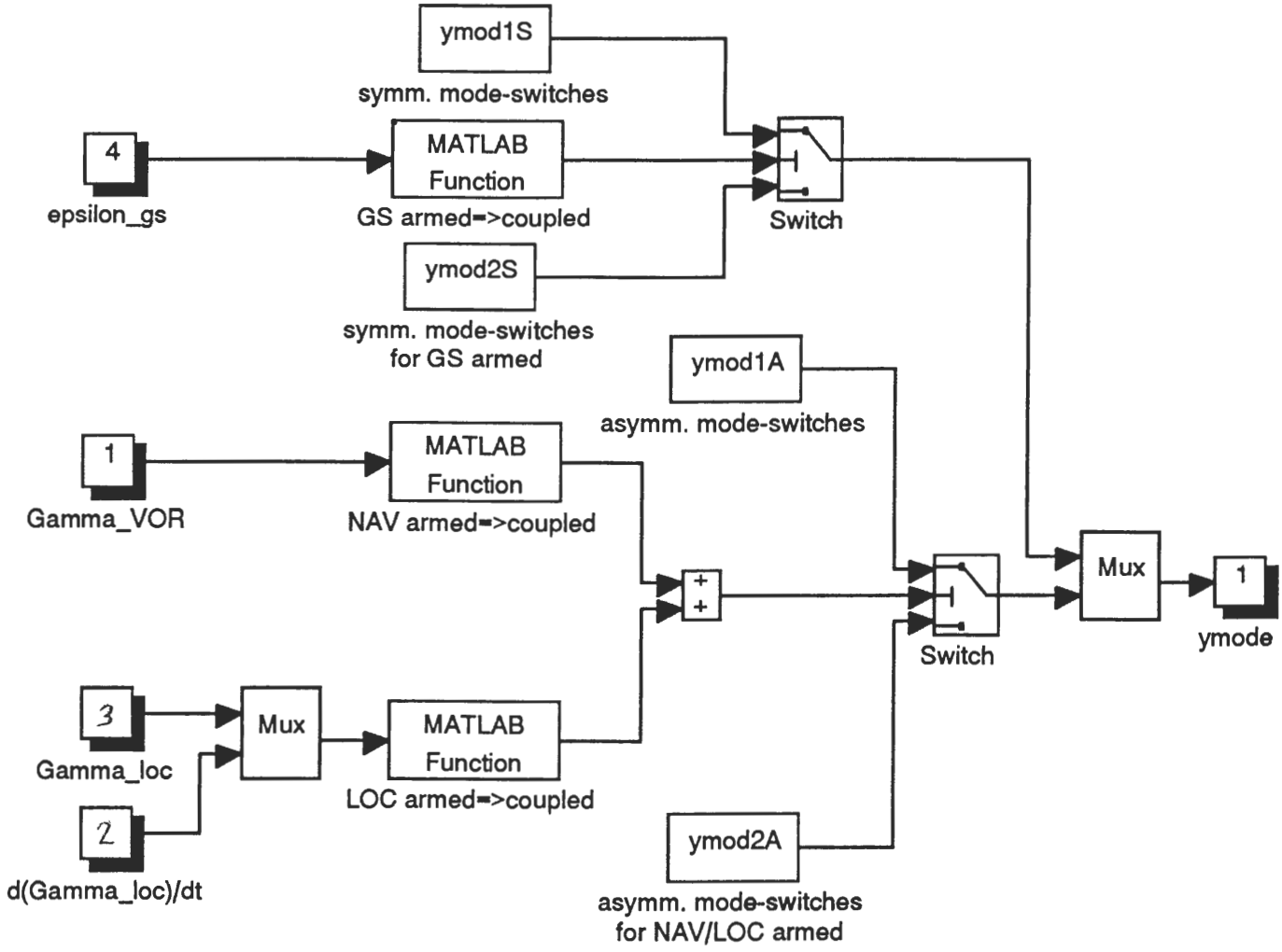


Figure C-27. Internal structure of the block *Mode Controller*.
(APILOT3 \ Mode Controller)

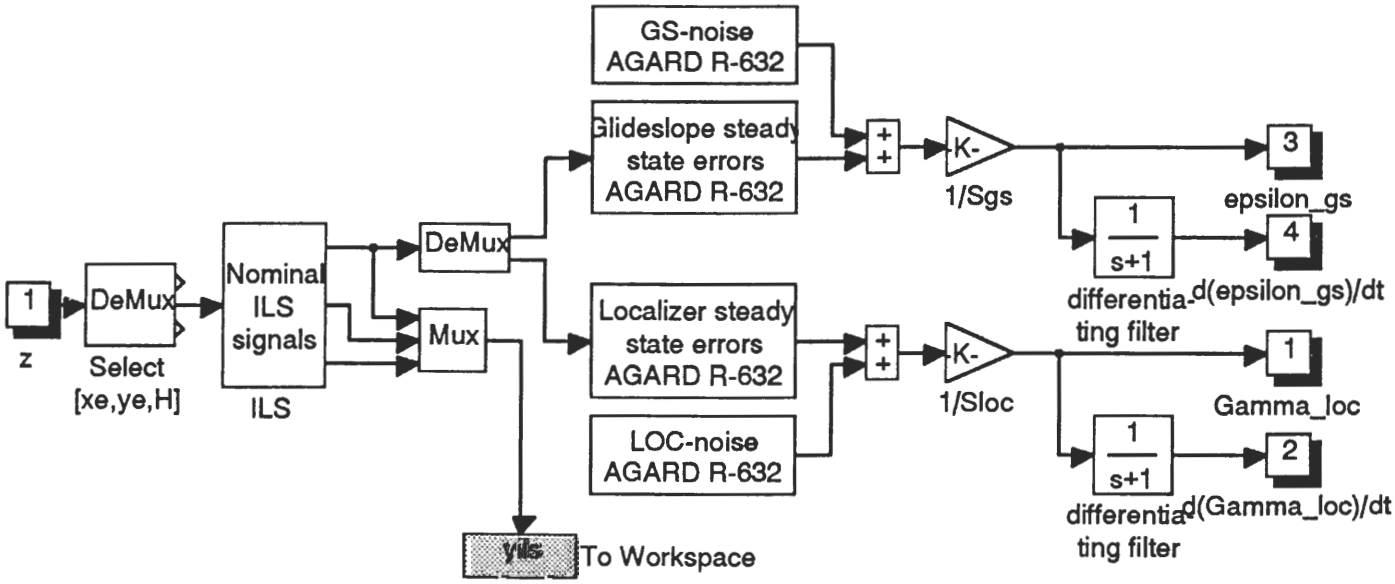


Figure C-28. Internal structure of the block *ILS*.
(APILOT3 \ ILS)

```
% Variables for mode-switching of Glideslope, Localizer, and Navigation modes.
% See GSSWITCH, LOCSWTCH, and NAVSWTCH for more details.
% ------------------------------------------------------------------------
GSswitched  = 0; % No glideslope mode-switches performed yet.
LOCswitched = 0; % No localizer mode-switches performed yet.
NAVswitched = 0; % No navigation mode-switched performed yet.

disp('Ready.');
%------------------------------------------------------------------------
% Programmed by: Marc Rauw. Version: September 10, 1993.
```

The block *Reference Signals* is used to generate the vector $\mathbf{y}_{ref}$. Since the reference values for the 'Coupled' phases of the *Navigation* and *Approach* modes are generated by the subsystems *VOR* and *ILS*, only one setting of $\mathbf{y}_{ref}$ suffices. For this reason, the block *Reference Signals* is nothing more than a *Constant* in which the symmetrical and asymmetrical parts of the vector $\mathbf{y}_{ref}$ are set separately. The following variables must be defined in the MATLAB workspace:

yrefS vector with reference values of symmetrical autopilot modes
yrefA vector with reference values of asymmetrical autopilot modes

so:

$$\mathbf{y}_{ref} \equiv [\ yrefS\ ;\ yrefA\ ]$$

These variables are generated automatically if the routine *APSTART* is used to initialize the autopilot.

C.11 The subsystems *ILS* and *VOR*.

Figures C-28, C-29 and C-30, show the subsystems *ILS* and *VOR*. These subsystems are equal to the corresponding systems from part I (see section F.4 of appendix F), but they do not contain the blocks which check if the aircraft flies within the area where the signals can be received accurately. Also, the dialog-boxes for the user have been omitted, which means that in order to function properly, a number of variables need to be defined in the MATLAB workspace:

Subsystem *ILS*:

HRW altitude of runway in [m] above sea level
xRW X-distance from aircraft to runway at $t = 0$ (in [m], F_E reference frame)
yRW Y-distance from aircraft to runway at $t = 0$ (in [m], F_E reference frame)
gamgs γ_{gs} = nominal flightpath angle at glideslope in [rad]
psiRW heading of runway in [rad]
xloc distance from runway threshold to localizer transmitter, measured along the runway centerline, [m]
xgs distance from runway threshold to glideslope transmitter, measured parallel to runway centerline, [m]
ygs distance from runway threshold to glideslope transmitter, measured perpendicular to runway centerline, [m]
Sgs glideslope sensitivity
Sloc localizer sensitivity

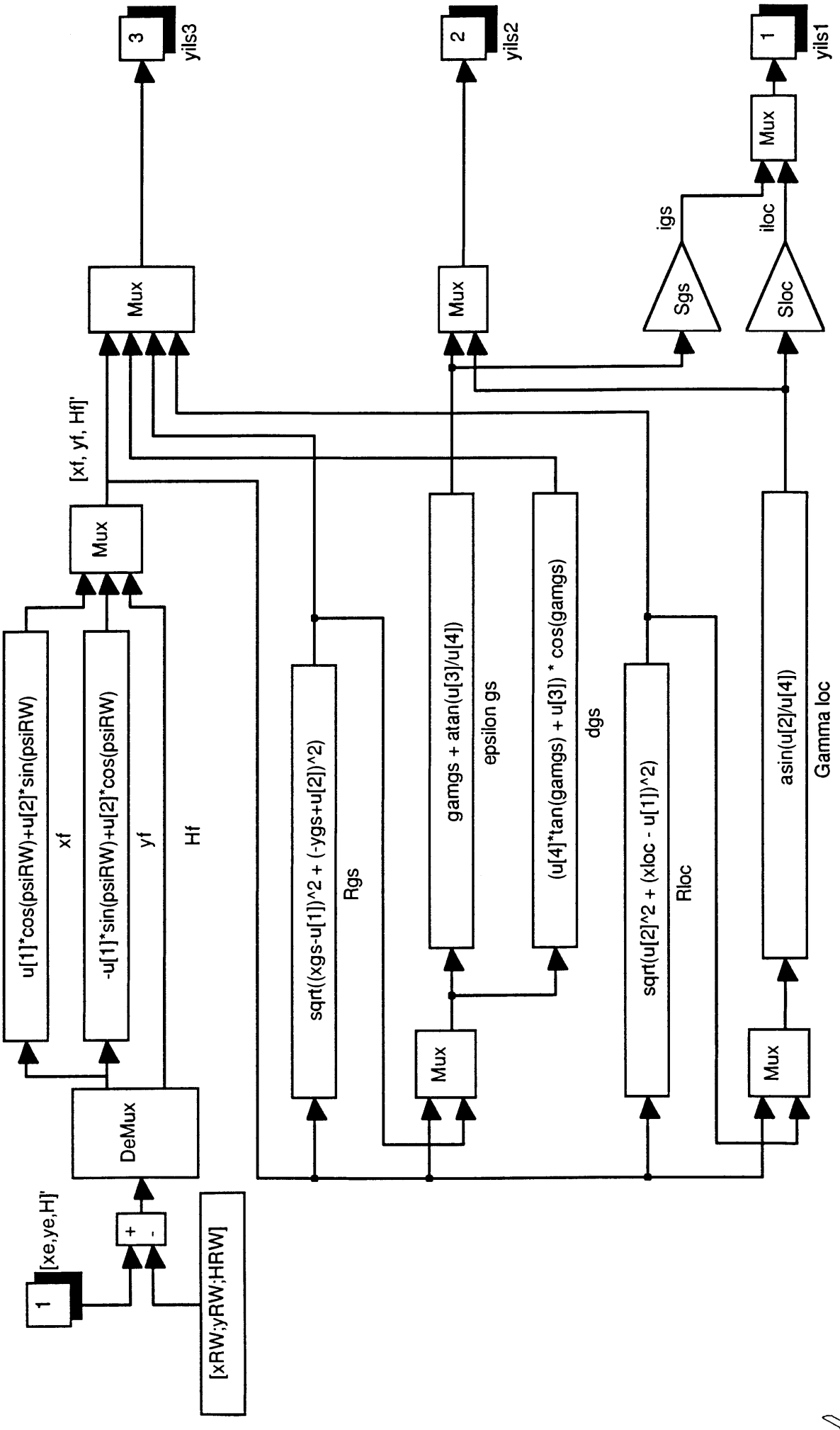


Figure C-29. Internal structure of the block *Nominal ILS signals.*

(APILOT3 \ ILS \ Nominal ILS signals)

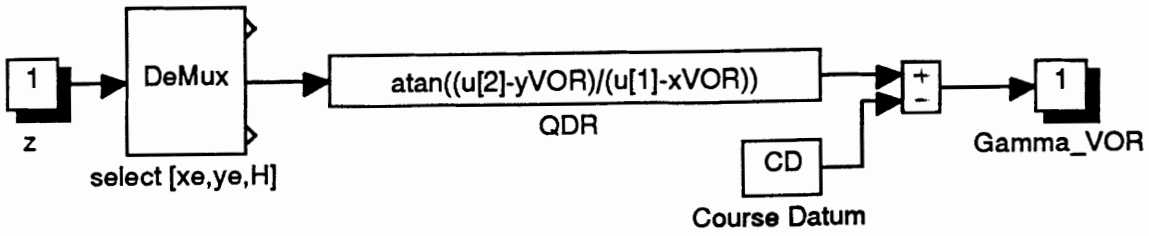


Figure C-30. Internal structure of the block *VOR*.
(APILOT3 \ VOR)

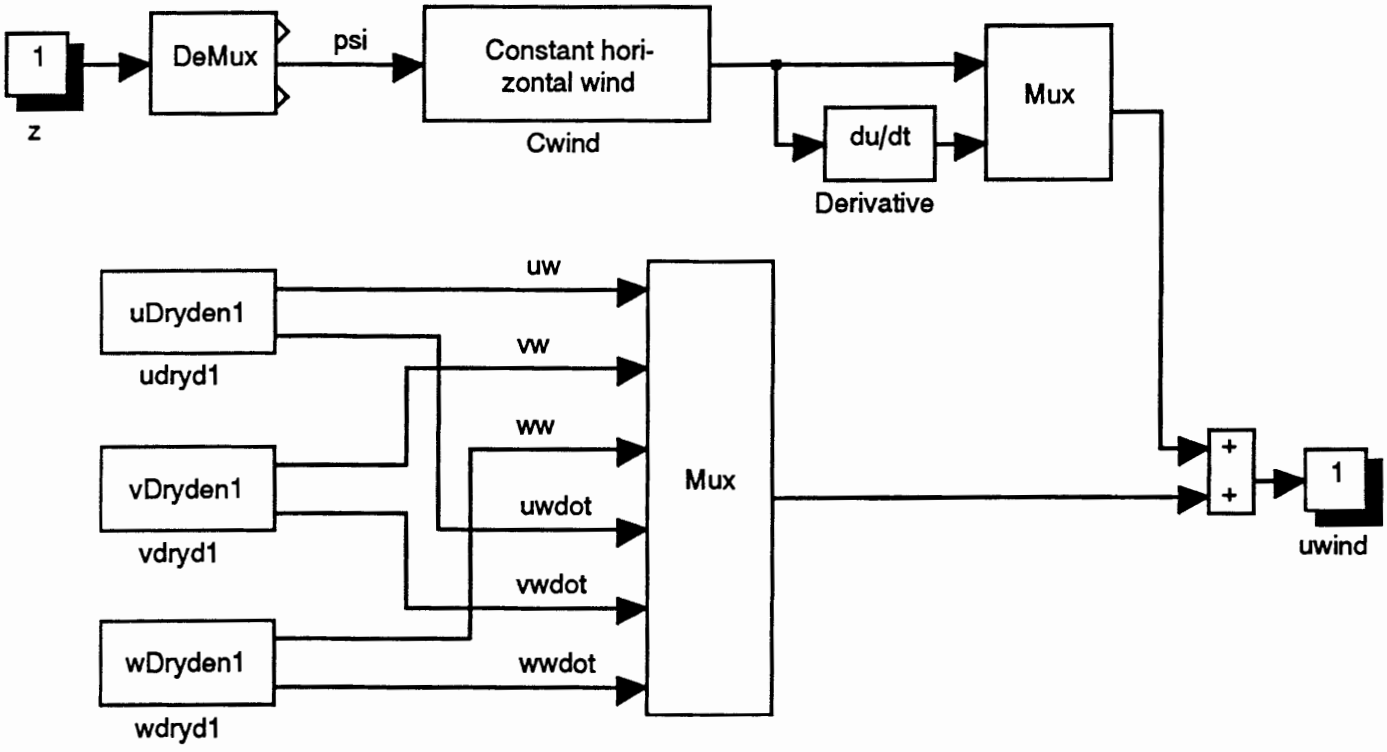


Figure C-31. Internal structure of the block *Wind and turbulence*.
(APILOT3 \ Wind and turbulence)

Subsystem *VOR*:

HVOR altitude of VOR transmitter
xVOR distance from aircraft to VOR transmitter at $t = 0$ (measured in [m] along the X_E-axis)
yVOR distance from aircraft to VOR transmitter at $t = 0$ (measured in [m] along the Y_B-axis)
CD Course Datum (reference VOR-bearing) in [rad]

These variables can be set by applying the routine *APINIT*, which can be called directly from the MATLAB command line or started by double-clicking the initialization block on top of the main level of *APILOT3*. Choose option 4 (*initialize VOR and/or ILS system*). The program APINIT.M will be described in section C.13.

C.12 The block *wind and turbulence.*

In figure C-31, the internal structure of the block *Wind and turbulence* can be seen. The turbulence velocities along the three body-axes of the aircraft are computed with the Dryden-filters *UDRYD1*, *VDRYD1*, and *WDRYD1*, which have been described in part I of this report. The wind is computed with the block *CWIND* that has also been described in part I. The horizontal wind has a constant magnitude and direction, but since the heading of the aircraft itself in general varies during simulations, the time-derivative of the velocity components along the aircrafts body-axes are not equal to zero. Since the variations of the wind velocities are not very fast, the approximation of the time-derivative which is obtained with a standard SIMULINK du/dt block is sufficient.

Inputvector: $\mathbf{z}$ (z)

Outputvector: $\mathbf{u}_{wind} = [\ u_w \quad v_w \quad w_w \quad \dot{u}_w \quad \dot{v}_w \quad \dot{w}_w\ ]^T$

The yaw-angle ψ (= heading of the aircraft if $\beta = 0$) is extracted from the vector $\mathbf{z}$ in order to determine the components of the airspeed along the body-axes of the aircraft.

C.13 The initialization routine *APINIT.*

In the main level of *APILOT3*, a button *Initialize autopilot, aircraft, ILS, and VOR models, define initial conditions* has been included. When this button is double-clicked, the MATLAB program APINIT.M will be started. This program will display a menu with the following options:

1 - *Load datamatrices for aircraft model.* If this option is chosen, the routine *LOADER* will be started, which will load the matrices *GM1*, *GM2*, *AM*, and *EM* into the MATLAB workspace from the files AIRCRAFT.MAT, ENGMOD.MAT, and AEROMOD.MAT. If these datafiles can not be found somewhere in the directories specified in the MATLAB path nor in the workdirectory, the program will ask if these files may be created in the user's workdirectory. If the user answers with *'y'*, the routine *MODBUILD* will

be applied (see appendix F of part I for more details). The datamatrices contain data about the aircraft's geometry and mass-distribution, parameters for the aerodynamic model, and parameters for the engine model, respectively, see appendix F of part I.

2 - *Load FCS models.* If this option is chosen, the FCS models for 35, 45, or 55 m/s will be loaded into the MATLAB workspace from the file FCS35.MAT, FCS45.MAT, or FCS55.MAT, respectively.

3 - *Define initial flight condition.* If this option is chosen, the initial flight condition, defined in the vectors *xinco*, *ua0*, *ut0*, and *xdot0* can be loaded from a MAT-file, or computed with the trim-routine *ACTRIM*. See appendices E and F of part I for more information about ACTRIM.M and aircraft trim in general.

4 - *Initialize VOR and / or ILS systems.* If this option is chosen, the programs VORINIT.M and / or ILSINIT.M will be started, which will ask the user to specify a number of parameters for the blocks *ILS* and *VOR* and the corresponding control laws. See section C.11 and the program listings of these programs.

5 - *Fix states.* This option will start the routine *FIXSTATE*, which can be used to set time-derivatives of some state variables to zero. See appendix F of part I for more info.

6 - *Load data from file.* If this option is chosen, the load routine *INCOLOAD* will be called. See appendix F of part I for more info.

The listings of APINIT.M, ILSINIT.M, and VORINIT.M have been given below.

Program listing of APINIT.M:

```
%------------------------------------------------------------------
% APINIT.m initializes the autopilot, aircraft model, and ILS and
% VOR models. Run this routine before starting new simulations.
%------------------------------------------------------------------
s = 0;
while s ~= 7                                    % While not choosing QUIT...

   disp('Initialize autopilot and aircraft models');
   disp('=======================================');
   disp('');
   disp('Use the sequence shown here for making selections!');
   disp('');
   s = menu('Options:','Load datamatrices for aircraft model',...
            'Load FCS models','Define initial flight condition',...
            'Initialize VOR and/or ILS systems','Fix states',...
            'Load data from file','Quit');

   if s == 1
      disp('');
      e1 = exist('aircraft.mat');
      e2 = exist('aeromod.mat');
      e3 = exist('engmod.mat');
      if e1~=2 | e2~=2 | e3~=2
         answ=input('Files not found. Do you want to create them (y/n)? ','s');
         if answ == 'y'
            modbuild;
            loader;
```

```
      end
   else
      loader;
   end
   clear e1 e2 e3 answ
   disp('');

elseif s == 2
   disp('');
   FCS = menu('Which Flight Control System model?','FCS for V=35 m/s',...
              'FCS for V=45 m/s','FCS for V=55 m/s');
   if FCS == 1
      load fcs35
   elseif FCS == 2
      load fcs45
   elseif FCS == 3
      load fcs55
   end
   clear FCS
   disp('');

elseif s == 3
   disp('');
   answ = menu('Define initial flight condition. Options:',...
               'Run aircraft trim program',...
               'Load initial condition from file');
   if answ == 1
      actrim;
   else
      incoload;
   end
   clear answ
   disp('');

elseif s == 4
   disp('Initializing VOR and/or ILS models');
   disp('');
   answ = 0;
   while answ ~= 3                    % While not going back to the main menu
      answ = menu('Options:','Initialize VOR','Initialize ILS',...
                  'Back to main menu');
      disp('');
      if answ == 1
         vorinit;
      elseif answ == 2
         ilsinit;
      end
   end
   disp('');

elseif s == 5
   disp('')'
   fixstate;
   disp('');

elseif s == 6
   disp('');
   incoload;
   disp('');
end
end
clear s
disp('Ready.');
% ------------------------------------------------------------
% Program written by: Marc Rauw. September 10, 1993.
```

Listing of ILSINIT.M:

```
% ------------------------------------------------------------
% ILSINIT.m sets the proper parameters for the ILS block and
% the GS and LOC control laws in the systems APILOT1, APILOT2,
% and APILOT3. Run this routine before starting a new simulation!
```

```
disp('Initialize the ILS block and the approach control laws');
disp('=================================================');
disp('');
HRW   = input('Runway height above sea-level [m] (default 0 m): ');
if isempty(HRW)
   HRW = 0;
end
xRW   = input('Initial X-distance from aircraft to runway [m] (default 0 m): ');
if isempty(xRW)
   xRW = 0;
end
yRW   = input('Initial Y-distance from aircraft to runway [m] (default 0 m): ');
if isempty(yRW)
   yRW = 0;
end
gamgs = input('Reference flightpath angle on glideslope [deg] (default 3 deg): ');
if isempty(gamgs)
   gamgs = -3;
end
psiRW = input('Reference heading of runway [deg] (default 0 deg): ');
if isempty(psiRW)
   psiRW = 0;
end

gamgs = gamgs * pi/180;
psiRW = psiRW * pi/180;

xloc = input('X-distance from runway threshold to LOC antenna [m] (dflt 2000 m): ');
if isempty(xloc)
   xloc = 2000;
end
xgs = input('X-distance from runway threshold to GS antenna [m] (dflt 300 m): ');
if isempty(xgs)
   xgs = 300;
end
ygs = input('Y-distance from runway centerline to GS antenna [m] (dflt 100 m): ');
if isempty(ygs)
   ygs = 100;
end

if gamgs == 0
   Sgs = 1;
else
   Sgs = 625/abs(gamgs);
end
if xloc == 0
   Sloc = 1;
else
   Sloc = 1.4*xloc;
end

disp('');
disp('Ready.');

% ------------------------------------------------------------
% Program was written by: Marc Rauw. September 10, 1993.
```

Program listing of VORINIT.M:

```
% ------------------------------------------------------------
% VORINIT.m sets the proper parameters for the VOR block in the
% systems APILOT1, APILOT2, and APILOT3. Run this routine be-
% fore starting a new simulation!
% ------------------------------------------------------------

disp('Initialize the VOR block');
disp('=======================');
disp('');
HVOR  = input('Height of VOR antenna above sea level [m] (default 0); ');
   HVOR = 0;
end
```

```
xVOR  = input('Initial X-distance to VOR antenna [m] (default 0): ');
if isempty(xVOR)
   xVOR = 0;
end
yVOR  = input('Initial Y-distance to VOR antenna [m] (default 0): ');
if isempty(yVOR)
   yVOR = 0;
end
CD    = input('Course Datum [deg] (default 0): ');
if isempty(CD)
   CD = 0;
end

CD    = CD * pi/180;

disp('');
disp('Ready.');
% ------------------------------------------------------------------
% Program written by: Marc Rauw. September 10, 1993.
```

C.14 Conclusions.

In this appendix, the internal structure of the autopilot blocks has been discussed. The control laws themselves have been implemented in a modular structure, with a clear division between inner- and outer-loops. Theoretically, it must be possible to decouple these blocks from the SIMULINK system, for direct use in a real-time flightsimulator or even in the Flight Control Computers of the aircraft itself. Although this requires the development of proper interfaces between the SIMULINK Aircraft Dynamics and Control environment, the real-time flightsimulator, and the FCCs of the aircraft, it is really worthwhile trying to make this possible, because it will reduce the time, needed to develop an automatic controller all the way up to flight-test considerably.

With respect to the SIMULINK design environment itself, it is strongly recommended to examine the implementation of the control laws closely, because it clearly demonstrates the power of the tools from part I, and it may help to hit upon some good ideas for your own systems. Moreover, it is not unlikely that readers might have better ideas of how to solve certain problems in SIMULINK, which may lead to a further improvement of this simulation structure.

(sub)system	inputs	outputs
BEAVER dynamics	$\mathbf{u}_{wind} = [\, u_w;\ v_w;\ w_w;\ \dot{u}_w;\ \dot{v}_w;\ \dot{w}_w \,]$ $\mathbf{u}_a = [\, \delta_e;\ \delta_a;\ \delta_r;\ \delta_f \,]$ $\mathbf{u}_t = [\, n;\ p_z \,]$	$[\, \mathbf{x};\ \dot{H} \,]$ $[\, \mathbf{x} - \mathbf{x}_0;\ \dot{H} - \dot{H}_0 \,]$ $\mathbf{y}_{dl} = [\, \frac{pb}{2V};\ \frac{q\bar{c}}{V};\ \frac{rb}{2V} \,]$
Add initial inputs	$\Delta\delta_e$ $\Delta\delta_a$ $\Delta\delta_r$ $\Delta\delta_f$ Δn Δp_z	$\mathbf{u}_a$ $\mathbf{u}_t$
Sensors and sub-traction of initial conditions	$[\, \mathbf{x};\ \dot{H} \,]$	$\mathbf{z} = [\, \mathbf{x};\ \dot{H} \,]$ after sensors $\mathbf{z}-\mathbf{z}_0 = [\, \mathbf{x}-\mathbf{x}_0;\ \dot{H}-\dot{H}_0 \,]$ after sensors
FCS	$V\Delta\delta_e$ $V\Delta\delta_a$ $V\Delta\delta_r$ $\mathbf{y}_{dl}$	$\Delta\delta_e$ $\Delta\delta_a$ $\Delta\delta_r$
computational delay and limiters	$V\Delta\delta_e$ (undelayed, unlimited) $V\Delta\delta_a$ (undelayed, unlimited) $V\Delta\delta_r$ (undelayed, unlimited)	$V\Delta\delta_e$ (delayed, limited) $V\Delta\delta_a$ (delayed, limited) $V\Delta\delta_r$ (delayed, limited)
ILS	$\mathbf{z}$	ϑ_{gs} Γ_{loc} $\dot{\Gamma}_{loc}$
VOR	$\mathbf{z}$	Γ_{VOR}
Mode Controller	Γ_{VOR} $\boldsymbol{\varepsilon}_{gs}$ Γ_{loc} $\dot{\Gamma}_{loc}$	$\mathbf{y}_{mode}$
Wind and turbulence	$\mathbf{z}$	$\mathbf{u}_{wind}$

Table C-1. Definitions of inputs and outputs of the subsystems *BEAVER dynamics*, *Add initial inputs*, *FCS*, *Computational delay and limiters*, *VOR*, *ILS*, *Mode Controller*, and *Wind and turbulence*.

$\mathbf{z}$ = [V; α; β; p; q; r; ψ; θ; φ; x_e; y_e; H; $\dot{H}$] after sensors;
$\mathbf{z} - \mathbf{z}_0$ = $\mathbf{z}$ - $\mathbf{z}(t=0)$
$\mathbf{y}_{mode}$ = [symm. mode on/off; symm. outer-loop on/off; ALH on/off; ALS on/off, GS coupled on/off; · · · · · · asymm. mode on/off; asymm. outer-loop on/off; HH on/off, NAV on/off; LOC coupled on/off]
$\mathbf{y}_{ref}$ = [$\Delta\theta_r$(PAH); ΔH_r(ALH); $\Delta \dot{H}_r$(ALS); $\Delta\varphi_r$(RAH); $\Delta\psi_r$(HH)]

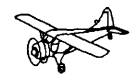
(sub)system	inputs	outputs
Symm. autopilot modes (main)	$\mathbf{y}_{mode}$ $\mathbf{y}_{ref}$ $\mathbf{z}$ $\mathbf{z} - \mathbf{z}_0$ $\boldsymbol{\varepsilon}_{gs}$ $\dot{\boldsymbol{\varepsilon}}_{gs}$	$V\Delta\delta_e$
Select symm. mode-controller signals	$\mathbf{y}_{mode}$ $\mathbf{y}_{ref}$	symm. mode on/off symm. outer-loop on/off ALS on/off ALH on/off GS coupled on/off $\Delta\theta_r$(PAH) $\Delta \dot{H}_r$(ALS) ΔH_r(ALH)
Select symm. outputs	$\mathbf{z}$ - $\mathbf{z}_0$ $\mathbf{z}$	$\dot{H} - \dot{H}_0$ H - H_0 θ - θ_0 q φ r V H
ALS	ALS on/off $\Delta \dot{H}_r$(ALS) $\dot{H} - \dot{H}_0$ V	$\Delta\theta_r$(ALS)
ALH	ALH on/off ΔH_r(ALH) H - H_0 V	$\Delta\theta_r$(ALH)
GS	GS coupled on/off $\boldsymbol{\varepsilon}_{gs}$ $\dot{\boldsymbol{\varepsilon}}_{gs}$ V H	$\Delta\theta_r$(GS)
PAH	Symm. outer-loop on/off $\Delta\theta_r$ (PAH) $\Delta\theta_r$ (ALH, ALS, or GS) θ - θ_0 q φ r V	$V\Delta\delta_e$

Table C-2. Definitions of inputs and outputs of symmetrical autopilot blocks (modenames between brackets denote source or destination of the signals).

$\mathbf{z}$ = [V; α; β; p; q; r; ψ; θ; φ; x_e; y_e; H; $\dot{H}$] after sensors;
$\mathbf{z} - \mathbf{z}_0$ = $\mathbf{z}$ - $\mathbf{z}(t{=}0)$
$\mathbf{y}_{mode}$ = [symm. mode on/off; symm. outer-loop on/off; ALH on/off; ALS on/off, GS coupled on/off; · · · · · · asymm. mode on/off; asymm. outer-loop on/off; HH on/off, NAV on/off; LOC coupled on/off]
$\mathbf{y}_{ref}$ = [$\Delta\theta_r$(PAH); ΔH_r(ALH); $\Delta \dot{H}_r$(ALS); $\Delta\varphi_r$(RAH); $\Delta\psi_r$(HH)]

(sub)system	inputs	outputs
Asymm. autopilot modes (main)	$\mathbf{y}_{mode}$ $\mathbf{y}_{ref}$ $\mathbf{z}$ $\mathbf{z} - \mathbf{z}_0$ Γ_{VOR} Γ_{loc} $\dot{\Gamma}_{loc}$	$V\Delta\delta_r$ $V\Delta\delta_a$
Select asymm. mode-controller signals	$\mathbf{y}_{mode}$ $\mathbf{y}_{ref}$	asymm. mode on/off asymm. outer-loop on/off HH on/off NAV on/off LOC coupled on/off $\Delta\varphi_r$(RAH) $\Delta\psi_r$(HH)
Select asymm. outputs	$\mathbf{z}$ - $\mathbf{z}_0$ $\mathbf{z}$	ψ - ψ_0 φ - φ_0 φ r V H
HH	HH on/off $\Delta\psi_r$(HH) ψ - ψ_0 V	$\Delta\varphi_r$(HH)
NAV	NAV on/off Γ_{VOR} $\Delta\psi_r$(HH) $\psi - \psi_0$ V	$\Delta\varphi_r$(NAV)
LOC	LOC coupled on/off Γ_{loc} $\dot{\Gamma}_{loc}$ V H	$\Delta\varphi_r$(LOC)
RAH	Asymm. outer-loop on/off $\Delta\varphi_r$(RAH) $\Delta\varphi_r$(HH, NAV, or LOC) φ - φ_0 φ r V	$V\Delta\delta_a$ $V\Delta\delta_r$

Table C-3. Definitions of inputs and outputs of asymmetrical autopilot blocks (modenames between brackets denote source or destination of the signals).



Appendix D. Installation procedures.

This report should be accompagnied by a 3½ inch floppy disk, which contains all SIMULINK systems, programs, and help texts, described in appendix F. If this floppy is not present, contact the author at the address on the cover page of this report.

The installation procedure is straightforward: put the floppy in drive A, and type *install* at the DOS-prompt. The INSTALL.BAT file will automatically copy the SIMULINK systems, programs, and help texts to your hard disk, assuming that you want to use drive C to store the programs. *If you want to use drive B as source disk, or another harddisk as destination disk, change INSTALL.BAT BEFORE proceeding with the installation.* The following directories will be created:

- BEAVER will be filled with the SIMULINK model of the 'Beaver' aircraft (*BEAVER*), the library *ACLIB*, the initialization files for the aircraft model, and the *trim* routine ACTRIM.M;
- APILOT will be filled with the systems *APILOT1*, *APILOT2*, and *APILOT3*, which contain the control laws of the 'Beaver' autopilot, and the initialization files for the autopilot and modecontroller;
- WIND will be filled with the wind & turbulence library *WINDLIB*;
- NAVIGATE will be filled with the VOR navigation and ILS approach library *NAVLIB*;
- EXAMPLES will be filled with the open-loop examples from chapter 6 (*OLOOP1*, *OLOOP1a* and *OLOOP2*);
- HELP will be filled with the help texts for the blocks from the other directories (if you add new blocks, you can copy your own help texts to this directory too, to prevent them from becoming mixed-up with the program files themselves!);
- TOOLS will be filled with the library *TOOLS*, which contains some useful general-purpose blocks, created with the SIMULINK *Mask* function.

After running INSTALL.BAT, the file MODINIT.M must be copied to your main SIMULINK directory (which should be contained in the variable *Matlabpath*!).

If the installation has been completed, SIMULINK can be started as usual. Type *modinit* at the MATLAB command line to add the model libraries to the path setting, which is stored in the string variable *Matlabpath*. You may want to include the *modinit*-call in your STARTUP.M file if you use these models very often, to automatically adjust *Matlabpath*.