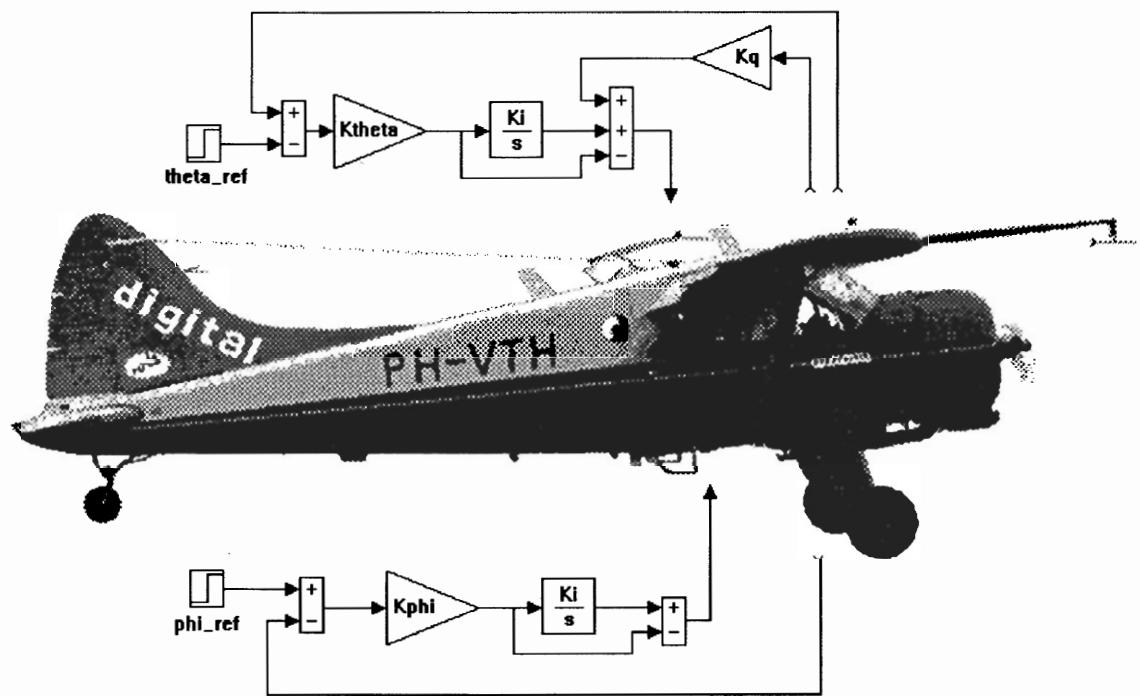


A SIMULINK environment for Flight Dynamics and Control analysis – application to the DHC-2 'Beaver'

Part I. Implementation of a model library in SIMULINK

September 1993

M.O. Rauw



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

Part I. Implementation of a model library in SIMULINK

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.

The on-line version of this document is © 2001 by M.O. Rauw. This version has been released under and is subject to the terms of the Common Documentation License (CDL), v.1.0, the terms of which are incorporated below, with one exception: the last sentence in section 10 of the CDL is hereby withdrawn and should be replaced by "This License shall be governed by the laws of the Kingdom of The Netherlands".

Instead of a list: all trademarks used in this document and the software, described in this document, are registered to whomever it is that owns them.

Common Documentation License

Version 1.0 — February 16, 2001

© 2001 Apple Computer, Inc.

Permission is granted to copy and distribute verbatim copies of this License, but changing or adding to it in any way is not permitted.

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, DISTRIBUTION 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 Derivatives 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.[25]. 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.[25]. Part I of this report can be read independently of ref.[25], 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 (AACSSs), aimed at the new DUT/NLR laboratory aircraft, the Cessna 'Citation II'. The process which leads from preliminary AACCS design to in-flight testing of the system is quite complex and should be mastered in detail to ensure safe and satisfying AACCS 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 I.

Since a couple of years, the Disciplinary Group for Stability and Control of Delft University of Technology, Faculty of Aerospace Engineering, has done research into automatic aircraft control technology with its laboratory aircraft, the De Havilland DHC-2 'Beaver'. Recently, this aircraft has been replaced by a new Cessna Citation II, which will be configured as *National Fly-by-wire Testbed* (NFT), to increase the research efforts in this field.

A prerequisite for successful fly-by-wire research is the availability of powerful tools for control system analysis and design. In this report, a flexible environment for the analysis of aircraft dynamics and control will be developed for the DHC-2 'Beaver' aircraft. This environment uses the power and flexibility of the simulation and system analysis programs SIMULINK and MATLAB.

The heart of the software package is formed by the SIMULINK implementation of the nonlinear 'Beaver' simulation model. In part II of this report, this model will be used for the analysis of the 'Beaver' autopilot. Due to its modular structure, the model can easily be adapted for other aircraft. Aircraft trim and linearization tools have been included, to be able to do the whole linear and nonlinear control system design and analysis from within the same MATLAB/SIMULINK environment. The package also contains blocks to simulate the influence of atmospheric disturbances upon the motions of the aircraft, and to generate radio-navigation signals for the assessment of navigation and approach control laws.

The tools from this report can be used for a broad range of applications in the field of aircraft Stability and Control analysis. Although this report will emphasize their applicability to Automatic Aircraft Control Systems design, they can also be used for system identification and sophisticated open-loop analysis of the aircraft dynamics, due to the availability of the full range of MATLAB toolboxes. The current models can readily be used for educational purposes; application for the NFT project is possible after altering some subsystems for implementation of the Citation II model.

In the future, the tools need to be developed further into a standardized analytical tool which must be applicable to virtually any aircraft. If possible, easy links from this environment to the flightsimulator and flight control computers of the aircraft need to be made, in order to shorten the development cycle of automatic control systems. The tools do require sufficient computer capacity, but the resulting increase in productivity is absolutely necessary for the Disciplinary Group for Stability and Control to be able to set new standards for fly-by-wire research.





Contents (part I).

Preface	v
Acknowledgements	vii
Summary of part I	ix
Symbols and definitions (part I)	1
0.1 Symbols	1
0.2 Vectors and vector functions	5
0.3 Matrices	5
0.4 Functions	5
0.5 Indices	6
0.6 Superscripts	7
0.7 Abbreviations	7
0.8 Reference frames and sign conventions	8
0.8.1 Definitions	8
0.8.2 Relationships between the reference frames	10
0.8.3 Sign conventions for deflections of control surfaces	11
Introduction to part I	17
Chapter 1. Design and evaluation of Automatic Aircraft Control Systems	19
1.1 Introduction	19
1.2 The AACCS design process	19
1.3 The place of this research within the AACCS design process	22
1.4 Conclusions	24
Chapter 2. Dynamic models of the 'Beaver', atmospheric turbulence, and navigation signals	27
2.1 Introduction	27
2.2 Aircraft equations of motion	27
2.2.1 General nonlinear equations of motion	27
2.2.2 Force and moment models	30
2.2.3 Writing the $\dot{\beta}$ -equation explicitly	36
2.2.4 Atmosphere model	37
2.2.5 Additional output equations	40
2.3 Wind and atmospheric turbulence	42
2.4 VOR navigation and ILS approach system signals	43
2.5 Conclusions	46
Chapter 3. Analysis of nonlinear dynamical systems with SIMULINK	49
3.1 Introduction	49
3.2 The programs SIMULINK and MATLAB	49



3.3 Advantages and disadvantages of SIMULINK, compared with other simulation environments	50
3.4 System representation in SIMULINK	51
3.5 The analytical functions of SIMULINK	53
3.5.1 The integration routines	54
3.5.2 The linearization routines	55
3.5.3 The trim routines	55
3.6 SIMULINK block-diagrams	56
3.7 Conclusions	56
Chapter 4. Implementation of the nonlinear 'Beaver' model in SIMULINK .	59
4.1 Introduction	59
4.2 Modular structure of the aircraft models	59
4.3 Adding external disturbances, navigation and approach models, and control loops	66
4.4 Conclusions	68
Chapter 5. Assessment of the new SIMULINK implementation of the 'Beaver' model	71
5.1 Introduction	71
5.2 Results from the aerodynamic and engine models	71
5.3 Assessment of open-loop responses computed with the new SIMULINK models ..	72
5.4 Open-loop responses to atmospheric turbulence	75
5.5 Open-loop analysis of the linearized 'Beaver' model	76
5.6 Conclusions	77
Chapter 6. Using the nonlinear SIMULINK 'Beaver' model in practice . . .	101
6.1 Introduction	101
6.2 Creating nonlinear open-loop responses to deflections of control surfaces and engine inputs	101
6.3 Creating linear open-loop responses to deflections of control surfaces and engine inputs	103
6.4 Creating nonlinear open-loop responses to atmospheric turbulence	104
6.5 Linearization of the aircraft model and applications for the linear model	105
6.6 Conclusions	106
Conclusions (part I)	111
Recommendations (part I)	113
References	115
Appendix A. Nonlinear equations of motion for a rigid aircraft	121
A.1 Introduction	121
A.2 Derivation of the equations for velocities and rotational velocities in the body-fixed reference frame	121
A.2.1 General force equation for a rigid body	121
A.2.2 General moment equation for a rigid body	122

A.2.3 Angular momentum about the centre of gravity	123
A.2.4 General equations of motion for a rigid body	124
A.3 Using the angle of attack, sideslip angle, and total airspeed in stead of the velocity components in F_B	127
A.3.1 Why use flight-path axes for the state equations?	127
A.3.2 Transformations of forces and velocities from body-axes to flight-path axes	129
A.3.3 Derivation of the \dot{V} -equation	130
A.3.4 Derivation of the $\dot{\alpha}$ -equation	131
A.3.5 Derivation of the $\dot{\beta}$ -equation	132
A.4 Equations of motion in a non-steady atmosphere	133
A.5 Kinematic relations and transformation of gravity forces from earth-axes to body-axes	136
A.5.1 The Euler angles	136
A.5.2 Gravity forces in the body-fixed reference frame	136
A.5.3 The position of the aircraft	136
A.6 Summary of the state equations	137
A.7 Conclusions	138
Appendix B. Mathematical models of wind and atmospheric turbulence ..	141
B.1 Introduction	141
B.2 Wind profiles and wind shear	141
B.3 Modelling atmospheric turbulence	142
B.4 Power spectra of atmospheric turbulence	145
B.4.1 The von Kármán spectra	145
B.4.2 The Dryden spectra	145
B.5 Filter design for atmospheric turbulence	146
B.5.1 Modelling atmospheric turbulence as filtered white noise	146
B.5.2 Filter design for the Dryden spectra	147
B.5.3 Filter design for the von Kármán spectra	148
B.6 Conclusions	148
Appendix C. The Instrument Landing System and the VOR navigation system	151
C.1 Introduction	151
C.2 The Instrument Landing System	151
C.2.1 Nominal ILS signals	151
C.2.2 Steady-state ILS offset errors	158
C.2.3 ILS noise characteristics	159
C.3 The VOR navigation system	162
C.3.1 Nominal VOR signals	162



C.3.2 VOR coverage and cone of silence	163
C.3.3 VOR steady-state errors	165
C.4 Conclusions	165
Appendix D. Simulation of dynamic systems in SIMULINK - theoretical backgrounds	167
D.1 Introduction	167
D.2 The integrators	167
D.2.1 Some fundamental concepts	167
D.2.2 A review of several numerical integration methods	171
a. Taylor series methods	171
b. Runge-Kutta methods	171
c. Multistep methods	176
d. Extrapolation methods	180
D.2.3 Stiff differential equations	180
D.2.4 Differential algebraic equations	182
D.3 Simulation of digital controllers	183
D.4 Obtaining state models from transfer functions	184
D.5 Algebraic loops	186
D.6 Conclusions	189
Appendix E. Steady-state trimmed flight	191
E.1 Introduction	191
E.2 Definition of steady-state flight	191
E.3 Specification of the flight condition	192
E.4 The rate-of-climb and turncoordination constraints	193
E.5 The steady-state aircraft trim algorithm	194
E.6 Implementation in MATLAB/SIMULINK: basic principles	196
E.7 Conclusions	198
Appendix F. The SIMULINK simulation models.	201
F.1 Introduction	201
F.2 Nonlinear model of the aircraft dynamics: state and output equations	202
F.2.1 Introduction	202
F.2.2 List of variables, used for the aircraft state and output models	202
F.2.3 The aircraft model library and the complete aircraft model	206
F.2.4 First level of the simulation model <i>BEAVER</i>	206
F.2.5 Beaver dynamics and output equations (second level of <i>BEAVER</i>)	209
a. Airdata Group	210
b. Aerodynamics Group	213
c. Engine Group	217
d. Gravity forces	218
e. Wind forces	218
f. Total forces and moments	218
g. Equations of motion	220
h. Other outputs	226

F.2.6 Helpprograms to create and load datafiles	228
F.2.7 Helpfunctions to compute initial conditions and to linearize the aircraft model	236
F.3 Models of wind and atmospheric turbulence	249
F.3.1 Introduction	249
F.3.2 List of variables, used for the wind and turbulence models	249
F.3.3 Description of the wind and turbulence models and their helpfiles	251
F.4 ILS approach and VOR navigation models	254
F.4.1 Introduction	254
F.4.2 List of variables, used for the ILS and VOR models	254
F.4.3 Description of the ILS and VOR blocks and helpfiles	257
F.5 Conclusions	263
Appendix G. Standard SIMULINK blocks	271
G.1 Introduction	271
G.2 Blocks from the Sources library	271
G.3 Blocks from the Sinks library	272
G.4 Blocks from the Discrete library	272
G.5 Blocks from the Linear library	273
G.6 Blocks from the Nonlinear library	273
G.7 Blocks from the Connections library	274
G.8 The <i>Group</i> and <i>Mask</i> functions	275
G.9 Conclusions	275
Appendix H. Installation procedures	277

List of tables.

1-1	Different steps in the aircraft model identification, performance analysis, and AACs design and evaluation, which can be done with SIMULINK	23
2-1	General aircraft data of the DHC-2 'Beaver', PH-VTH	29
2-2	Aircraft data on which the aerodynamic model is based	31
2-3	Coefficients in the nonlinear aerodynamic model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-range	32/33
2-4	Coefficients in the nonlinear engine model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-range	35
5-1	Initial conditions for $V = 45$ m/s, $H = 6000$ ft, obtained with the trim routine <i>ACTRIM</i>	74
5-2	State-space matrices of linear 'Beaver' model, derived from the nonlinear SIMULINK model with <i>ACLIN</i>	76
5-3	Basic open-loop properties of the linear 'Beaver' model, derived from the nonlinear SIMULINK model with <i>ACLIN</i>	77
6-1	Steps to be taken to simulate the system from figure 6-1	102
6-2	Steps to be taken to simulate the system from figure 6-2	104
A-1	Moments and products of inertia	124
A-2	Definition of inertia parameters	127/128
C-1	Maximum permissible localizer steady-state errors	158
C-2	Maximum permissible glideslope steady-state errors	158



C-3	VOR coverage as a function of altitude above the ground, based on two different flight tests	163
D-1	Fehlberg coefficients	175
D-2	Coefficients for Adams-Basforth methods	178
D-3	Coefficients for Adams-Moulton method	179
D-4	Coefficients for stiffly stable method (<i>GEAR</i>)	182
D-5	Magnitude of the signals for two calculating sequences (example)	187
F-1	Parameters of the aerodynamic model of the 'Beaver' in the format used by the SIMULINK simulation model	264
F-2	Parameters of the engine model of the 'Beaver' in the format used by the SIMULINK simulation model	265
F-3	Aircraft geometry and mass distribution data in the format used by the SIMULINK simulation model	265
F-4	List of basic blocks within the 'Beaver' model, including their inputs and outputs	266
F-5	List of all outputs from the system <i>BEAVER</i> which are sent to the MATLAB workspace	267/268

List of figures.

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 , and r	9
0-2	Relationship between the vehicle-carried vertical reference frame F_V and the earth-fixed reference frame F_E	10
0-3	Relationship between the vehicle-carried vertical reference frame F_V and the body-fixed reference frame F_B	12
0-4	Relationship between the flight path reference frame F_W and the body-fixed reference frame F_B	13
0-5	Relationships between body-fixed reference frame F_B , flight path reference frame F_W , and stability reference frame F_S	14
0-6	Sign conventions for control surface deflections	14
2-1	The De Havilland DHC-2 'Beaver' laboratory aircraft	29
2-2	Basic dimensions of the DHC-2 'Beaver'	30
3-1	Functional relations between MATLAB and SIMULINK	50
4-1	Modular structure of the DUT-flightsimulator software (sequence of computations)	60
4-2	Structure of the main level of the 'Beaver' model in SIMULINK	61
4-3	Structure of the block 'Airdata Group'	63
4-4	Structure of the block 'Aerodynamics Group'	64
4-5	Structure of the block 'Engine Group'	64
4-6	Structure of the block 'Aircraft equations of motion'	65
4-7	Structure of the block 'State derivatives'	65
4-8	Model structure for simulation of an aircraft, equipped with an automatic controller, and affected by external disturbances	67
5-1	Force and moment coefficients of the nonlinear Beaver model as functions of α and β	71
5-2	Open-loop responses of the 'Beaver' to a block-shaped elevator input $\Delta\delta_e = 3^\circ$ during 2 seconds	78/79
5-3	Open-loop responses of the 'Beaver' to a block-shaped aileron input $\Delta\delta_a = 3^\circ$ during 2 seconds	80/81
5-4	Open-loop responses of the 'Beaver' to a block-shaped rudder input $\Delta\delta_r = 3^\circ$ during 2 seconds	82/83
5-5	Open-loop responses of the 'Beaver' to a flap deflection $\Delta\delta_f = 3^\circ$ in 3 seconds	84/85
5-6	Open-loop responses of the 'Beaver' to a change in engine speed $\Delta_n = 200$ RPM in 4 seconds	86/87
5-7	Open-loop responses of the 'Beaver' to a change in manifold pressure $\Delta p_z = 2$ inch Hg in 2 seconds	88/89
5-8	Open-loop responses to longitudinal turbulence	90/91

5-9	Open-loop responses to lateral turbulence	92/93
5-10	Open-loop responses to vertical turbulence	94/95
5-11	Open-loop responses to longitudinal + lateral + vertical turbulence	96/97
5-12	Control-input Bode plots for longitudinal linearized 'Beaver' model	98
6-1	Block-diagram of the system <i>OLOOP1</i> (nonlinear open-loop simulation model for external inputs)	107
6-2	Block-diagram of the system <i>OLOOP1_L</i> (linear open-loop simulation model for external inputs)	107
6-3	Block-diagram of the system <i>OLOOP2</i> (nonlinear open-loop simulation model for atmospheric disturbances)	108
A-1	Relationship between aerodynamic forces in flight-path and body axes reference frames	130
B-1	Wind profile for $\lambda = -0.0065 \text{ K/m}$ and $V_{w_{0,15}} = 1 \text{ m/s}$	142
B-2	Wind velocity components along the aircraft's body axes	143
B-3	Von Kármán and Dryden spectra	145
B-4	Modelling atmospheric turbulence as filtered white noise	146
C-1	Positions of the ILS system components	151
C-2	Layout of the approach path	152
C-3	Required coverage of localizer signals	152
C-4	Hyperbolic intersection of localizer and glideslope reference planes	153
C-5	Required coverage of glideslope signals compared with required coverage of localizer signals	153
C-6	ILS performance categories	154
C-7	Definition of earth-fixed and runway axes, used for approach simulation	157
C-8	Localizer geometry and definition of \tilde{X}_E , \tilde{Y}_E , X_F , Y_F	157
C-9	Glideslope geometry	159
C-10	Maximum allowable ILS localizer and glideslope noise	161
C-11	Geometry of VOR navigation	162
C-12	The 'cone of silence'	164
D-1	Family of solutions of instable ODE	168
D-2	Family of solutions of stable ODE	169
D-3	Discretization error, roundoff error, and total error as a function of h_n	169
D-4	Euler's method for stiff ODE	168
D-5	Digital control simulation scheme	183
D-6	Block-diagram equivalent of transfer function	185
D-7	Model, consisting of three gains	186
D-8	Gain with unity feedback: an algebraic loop	187
D-9	Dynamics, introduced by an algebraic loop	188
D-10	Iterative solution of an algebraic loop	189
E-1	Trim algorithm	195
E-2	Program-structure diagram of ACTRIM.M (main program of aircraft trim routines)	196
E-3	Program-structure diagram of ACCONSTR.M (subroutine with flightpath constraints and kinematic relations)	197
E-4	Program-structure diagram of ACCOST.M (subroutine with cost function for aircraft trim)	197
F-1	Main level of the system <i>BEAVER</i>	207
F-2	Internal structure of the subsystem block <i>Beaver dynamics and output equations</i>	208
F-3	Internal structure of the subsystem block <i>Airdata Group</i>	211
F-4	Internal structure of the 'basic' block <i>Atmosph</i>	211
F-5	Internal structure of the 'basic' block <i>Airdata1</i>	212
F-6	Internal structure of the 'basic' block <i>Airdata2</i>	212
F-7	Internal structure of the 'basic' block <i>Airdata3</i>	212
F-8	Internal structure of the subsystem block <i>Aerodynamics Group</i>	214
F-9	Internal structure of the 'basic' block <i>Dimless</i>	214
F-10	Internal structure of the 'basic' block <i>Aeromod (Beaver)</i>	214
F-11	Internal structure of the 'basic' block <i>FMdims</i>	215



F-12	Internal structure of the subsystem block <i>Engine Group</i>	215
F-13	Internal structure of the 'basic' block <i>Power (Beaver)</i>	216
F-14	Internal structure of the 'basic' block <i>Engmod (Beaver)</i>	215
F-15	Internal structure of the 'basic' block <i>Gravity</i>	219
F-16	Internal structure of the 'basic' block <i>Fwind</i>	219
F-17	Internal structure of the 'basic' block <i>FMsort</i>	219
F-18	Internal structure of the subsystem block <i>Equations of motion</i>	221
F-19	Internal structure of the subsystem block <i>State derivatives</i>	221
F-20	Internal structure of the 'basic' block <i>Hlpfcn</i>	222
F-21	Internal structure of the 'basic' block <i>Vabdot</i>	223
F-22	Internal structure of the 'basic' block <i>pqrdot</i>	223
F-23	Internal structure of the 'basic' block <i>Eulerdot</i>	223
F-24	Internal structure of the 'basic' block <i>uvw</i>	224
F-25	Internal structure of the 'basic' block <i>xyHdot</i>	224
F-26	Internal structure of the 'basic' block <i>xdotcorr</i>	224
F-27	Internal structure of the 'basic' block <i>fpath</i>	227
F-28	Internal structure of the 'basic' block <i>accel</i>	227
F-29	Internal structure of the 'basic' block <i>uvwdot</i>	229
F-30	Internal structure of the block <i>BLWIND</i>	250
F-31	Internal structure of the block <i>CWIND</i>	250
F-32	Internal structure of the block <i>UDRYD1</i>	250
F-33	Internal structure of the block <i>VDRYD1</i>	250
F-34	Internal structure of the block <i>UDRYD2</i>	252
F-35	Internal structure of the block <i>VDRYD2</i>	252
F-36	Internal structure of the block <i>ILS</i>	257
F-37	Internal structure of the block <i>ILSTEST</i>	257
F-38	Internal structure of the block <i>LOCERR</i>	259
F-39	Internal structure of the block <i>GSERR</i>	259
F-40	Internal structure of the block <i>LOCNOISE1</i>	259
F-41	Internal structure of the block <i>GSNOISE1</i>	259
F-42	Internal structure of the block <i>LOCNOISE2</i>	260
F-43	Internal structure of the block <i>GSNOISE2</i>	260
F-44	Internal structure of the block <i>ILSXMPL</i>	260
F-45	Internal structure of the block <i>VOR</i>	262
F-45	Internal structure of the block <i>VORERR</i>	262

Symbols and definitions (part I).

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. A list of variables is included in appendix F.

0.1 Symbols.

a	engine model parameter, [-]
a	speed of sound, [m s^{-1}]
$a_{x,k}$	kinematic acceleration along X_B -axis, [g]
$a_{y,k}$	kinematic acceleration along Y_B -axis, [g]
$a_{z,k}$	kinematic acceleration along Z_B -axis, [g]
A_x	output of accelerometer (specific force) in c.g. along X_B -axis, [g]
A_y	output of accelerometer (specific force) in c.g. along Y_B -axis, [g]
A_z	output of accelerometer (specific force) in c.g. along Z_B -axis, [g]
b	wingspan, [m]
b	engine model parameter, [-]
\bar{c}	mean aerodynamic chord, [m]
CD	course datum, [rad]
C_l	dimensionless moment about X_B -axis (rolling moment), [-]
C_m	dimensionless moment about Y_B -axis (pitching moment), [-]
C_n	dimensionless moment about Z_B -axis (yawing moment), [-]
C_X	dimensionless force along X_B -axis, [-]
C_Y	dimensionless force along Y_B -axis, [-]
C_Z	dimensionless force along Z_B -axis, [-]
\underline{D}	total aerodynamic drag, [N]
d_{gs}	distance from aircraft \perp glideslope reference plane (see appendix C), [m].
d_{loc}	distance from aircraft \perp localizer reference plane (see appendix C), [m]
dpt	$\frac{\Delta p_t}{\frac{1}{\tau} \rho V^2}$, [-]
f_i	function evaluation at time step i
F_B	body-fixed reference frame
F_E	earth-fixed reference frame
F_F	runway ('field') reference frame
F_M	measurement reference frame
F_R	special body-fixed reference frame for the 'Beaver'
F_S	stability reference frame
F_V	vehicle-carried vertical reference frame
F_W	flight path reference frame
fpa	flight path acceleration, [g]
g	acceleration of gravity, [m s^{-2}]



h	pressure altitude, [m]
h	step size for numerical integration, [s]
H	geopotential altitude, [m]
H_f	altitude of aircraft above the runway, [m]
H_{RW}	altitude of runway above sea level, [m]
i	counter or iteration number, [-]
i_{gs}	current through glideslope indicator, [μ A]
i_{loc}	current through localizer indicator, [μ A]
I_i	inertia parameter ($i = 1, 2, \dots, 6$, see table A-2), [kg m^2]
I_x	moment of inertia along X_B -axis, [kg m^2]
I_y	moment of inertia along Y_B -axis, [kg m^2]
I_z	moment of inertia along Z_B -axis, [kg m^2]
J_{xy}	product of inertia in X_B - Y_B plane, [kg m^2]
J_{xz}	product of inertia in X_B - Z_B plane, [kg m^2]
J_{yz}	product of inertia in Y_B - Z_B plane, [kg m^2]
k	timestep (for discrete systems, $t = k \cdot t_s$), [-]
k_i	function evaluations between two output points for Runge-Kutta integrators
L	total aerodynamic rolling moment, [N m]
\bar{L}	total aerodynamic lift, [N]
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]
m	mass of aircraft, [kg]
M	Mach number, [-]
M	molecular weight of air, [kg kmol^{-1}]
M	total aerodynamic pitching moment, [N m]
n	engine speed, [RPM]
n	stepnumber in numerical integration, [-]
N	total aerodynamic yawing moment, [N m]
p	angular rate of roll, [rad s^{-1}]
p	order of numerical integration method, [-]
P	engine power, [N m s^{-1}]
$P_l, P_m,$ $P_n, P_{pp},$ $P_{pq}, P_{pr},$ $P_{qq}, P_{qr},$ P_{rr}	inertia parameters for p -equation (see appendix A, table A-2)
p_s	ambient or free-stream pressure, [N m^{-2}]
p_z	manifold pressure, ["Hg]
q	angular rate of pitch, [rad s^{-1}]
q_c	impact pressure, [N m^{-2}]
q_{dyn}	dynamic pressure, [N m^{-2}]
$Q_l, Q_m,$ $Q_n, Q_{pp},$ $Q_{pq}, Q_{pr},$ $Q_{qq}, Q_{qr},$ Q_{rr}	inertia parameters for q -equation (see appendix A, table A-2)

r	angular rate of yaw, [rad s^{-1}]
R	R_a/M_0 , specific gas constant of air, [$\text{J K}^{-1} \text{ kg}^{-1}$]
R_a	universal gas constant, [$\text{J K}^{-1} \text{ kmol}^{-1}$]
R_c	Reynolds number with respect to c , [-]
R_{DME}	distance from aircraft to DME transmitter, [m]
R_e	Reynolds number per unit length, [m^{-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]
$R_l, R_m,$ $R_n, R_{pp},$ $R_{pq}, R_{pr},$ $R_{qq}, R_{qr},$ R_{rr}	inertia parameters for r -equation (see appendix A, table A-2)
S	
S_{gs}	wing area, [m^2]
S_{loc}	sensitivity of glideslope indicator, [$\mu\text{A/rad}$]
t	sensitivity of localizer indicator, [$\mu\text{A/rad}$]
t_s	time, [s]
T	sampling time (stepwidth for discrete systems), [s]
T_t	ambient or free-stream temperature, [K]
$T_{1/2}$	total temperature, [K]
u	time until a signal has damped out to an amplitude of 0.5 times the initial value
u_w	velocity along X_B -axis, [m s^{-1}]
u_{we}	wind velocity component along X_B -axis, [m s^{-1}]
v	wind velocity component along X_E -axis, [m s^{-1}]
v_w	velocity along Y_B -axis, [m s^{-1}]
v_{we}	wind velocity component along Y_B -axis, [m s^{-1}]
V	wind velocity component along Y_E -axis, [m s^{-1}]
V_c	true airspeed, [m s^{-1}]
V_e	calibrated airspeed, [m s^{-1}]
V_w	equivalent airspeed, [m s^{-1}]
$V_{w_{9.15}}$	wind velocity, [m s^{-1}]
w	wind velocity at an altitude of 9.15 m (reference velocity), [m s^{-1}]
W	velocity along Z_B -axis, [m s^{-1}]
w	aircraft weight, [N]
w_1, w_2, w_3	aircraft weight, [N]
w_w	three independent white noise signals
w_{we}	wind velocity component along Z_B -axis, [m s^{-1}]
X_a	wind velocity component along Z_E -axis, [m s^{-1}]
x_f	aerodynamic force along X_B -axis, [N]
X_{gr}	X-coordinate of aircraft relatively to F_F (see appendix C), [m]
x_{gs}	gravity force along X_B -axis, [N]
x_{loc}	X-coordinate of glideslope antenna relatively to F_F (see appendix C), [m]
X_t	X-coordinate of localizer antenna relatively to F_F (see appendix C), [m]
x_{VOR}	force along X_B -axis due to operation of powerplant, [N]
Y	X-position of VOR antenna relatively to F_E , [m]
Y_a	total aerodynamic sideforce, [N]
Y_{gr}	aerodynamic force along Y_B -axis, [N]
	gravity force along Y_B -axis, [N]



y_{gs}	Y-coordinate of glideslope antenna relatively to F_F (see appendix C), [m]
Y_t	force along Y_B -axis due to operation of powerplant, [N]
y_{VOR}	Y-position of VOR antenna relatively to F_E , [m]
Z_a	aerodynamic force along Z_B -axis, [N]
Z_{gr}	gravity force along Z_B -axis, [N]
Z_t	force along Z_B -axis due to operation of powerplant, [N]
α	angle of attack, [rad]
α_i	coefficient in Runge Kutta and Adams equations
β	angle of sideslip, [rad]
β_i	coefficient in Adams equations
β_{ij}	coefficient in Runge Kutta equations
γ	flight path angle, [rad]
γ	ratio of specific heats of air, [-]
γ_{gs}	reference flight path angle at glideslope, [rad]
Γ_{gs}	angle between <i>localizer</i> reference plane and line from ground position of aircraft to <i>glideslope</i> antenna (see appendix C), [rad]
Γ_{loc}	angle between <i>localizer</i> reference plane and line from ground position of aircraft to <i>localizer</i> antenna (see appendix C), [rad]
Γ_{VOR}	angle between CD, and VOR bearing where aircraft flies, [rad]
γ_i	coefficient in Runge Kutta equation
γ_i^*	coefficient in Runge Kutta equation
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_n	local discretization error at integration step n
δ_r	angle of deflection of rudder, [rad]
Δ	increment, [-]
Δp_t	increase of total pressure over the propeller, [$N\ m^{-2}$]
ϵ	angle between <i>glideslope</i> reference plane, and line through aircraft and <i>glideslope</i> antenna (see appendix C), [rad]
θ	pitch angle, [rad]
λ	$\frac{\partial T}{\partial h}$ (temperature gradient), [$K\ m^{-1}$]
λ	eigenvalue
μ	aerodynamic angle of roll, [rad]
μ	dynamic viscosity, [$kg\ m^{-1}\ s^{-1}$]
ρ	air density, [$kg\ m^{-3}$]
σ	standard deviation
σ_{gs}	standard deviation of glideslope noise, [μA]
σ_{loc}	standard deviation of localizer noise, [μA]
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
φ	roll angle, [rad]
Φ	bank angle, [rad]
χ	azimuth angle, [rad]
ψ	yaw angle, [rad]
ψ_{RW}	runway heading, [rad]
ψ_w	wind direction (north = π rad), [rad]

ω	angular frequency, [rad/s]
ω_n	natural frequency of damped system, [rad/s]
ω_0	natural frequency of undamped system, [rad/s]
Ω	spatial frequency, [rad/m]

0.2 Vectors and vector functions.

\mathbf{a}	body axis acceleration vector
\mathbf{C}_a	vector with dimensionless force and moment coefficients from aerodynamic model
\mathbf{C}_t	vector with dimensionless force and moment coefficients from engine model
$\mathbf{f}(t)$	state equation
\mathbf{F}	resulting force vector acting on rigid body ($\mathbf{F} = [F_x \ F_y \ F_z]^T$)
$\mathbf{g}(t)$	output equation
\mathbf{G}	resulting moment vector about c.g. of rigid body ($\mathbf{G} = [L \ M \ N]^T$)
\mathbf{h}	resulting angular momentum of a rigid body about the centre of gravity ($\mathbf{h} = [h_x \ h_y \ h_z]^T$)
\mathbf{r}	position vector
$\mathbf{u}(t)$	input vector (continuous signal)
$\mathbf{u}(k)$	input vector (discrete signal)
\mathbf{V}	true airspeed vector
\mathbf{V}_w	wind velocity vector
$\mathbf{x}(t)$	state vector (continuous states)
$\mathbf{x}(k)$	state vector (discrete states)
$\mathbf{y}(t)$	output vector (continuous signal)
$\mathbf{y}(k)$	output vector (discrete signal)
$\boldsymbol{\omega}$	rotational velocity vector

0.3 Matrices.

\mathbf{A}	system matrix of linear state-space system
\mathbf{B}	input matrix of linear state-space system
\mathbf{T}_{ab}	transformation matrix (from reference frame \mathbf{F}_a to \mathbf{F}_b)
Θ	transformation matrix for first Euler rotation (section 0.8.2)
Φ	transformation matrix for second Euler rotation (section 0.8.2)
Ψ	transformation matrix for third Euler rotation (section 0.8.2)

0.4 Functions.

$\mathbf{f}(t)$	state equation
f_i	function evaluation at time step i
$\mathbf{g}(t)$	output equation
$H(\omega)$	frequency response of forming filter
k_i	function evaluations between two output points for Runge-Kutta integrators
$S(\omega), S(\Omega)$	power spectral density functions



0.5 Indices.

0	nominal value
0	value at sea level
a	aerodynamic forces and moments (-coefficients)
a	aileron
a	velocity components relative to surrounding atmosphere
$asym$	asymmetrical dynamics (lateral)
$c.g.$	in the centre of gravity
DME	DME
e	elevator
e	velocity components relative to earth axes
f	flaps
gr	gravity forces
gs	glideslope
k	'kinematic' (used for accelerations)
loc	localizer
n	number of step in the numerical integration, [-]
p	derivative with respect to dimensionless rolling speed, $\frac{\partial(\cdot)}{\partial\left(\frac{pb}{2V}\right)}$
q	derivative with respect to dimensionless pitching speed, $\frac{\partial(\cdot)}{\partial\left(\frac{qc}{V}\right)}$
r	derivative with respect to dimensionless yawing speed, $\frac{\partial(\cdot)}{\partial\left(\frac{rb}{2V}\right)}$
r	rudder
ref	reference value of a signal (used for inputs to control loops)
RW	runway
$symm$	symmetrical dynamics (longitudinal/vertical)
t	forces and moments (-coefficients) due to operation of powerplant
VOR	VOR
w	wind velocity (-components along body axes)
w	forces due to non-steady atmosphere
we	wind velocity components along earth axes
α	derivative with respect to angle of attack, $\frac{\partial(\cdot)}{\partial\alpha}$
α^2	derivative with respect to α^2 , $\frac{\partial(\cdot)}{\partial\alpha^2}$
α^3	derivative with respect to α^3 , $\frac{\partial(\cdot)}{\partial\alpha^3}$
$\alpha \delta_f$	derivative with respect to $\alpha \cdot \delta_f$, $\frac{\partial(\cdot)}{\partial(\alpha \cdot \delta_f)}$
$\alpha \Delta p_t^2$	derivative with respect to $\alpha \cdot dpt^2$, $\frac{\partial(\cdot)}{\partial(\alpha \cdot dpt^2)}$
$\alpha^2 \Delta p_t$	derivative with respect to $\alpha^2 \cdot dpt$, $\frac{\partial(\cdot)}{\partial(\alpha^2 \cdot dpt)}$
β	derivative with respect to angle of sideslip, $\frac{\partial(\cdot)}{\partial\beta}$
β^2	derivative with respect to β^2 , $\frac{\partial(\cdot)}{\partial\beta^2}$
β^3	derivative with respect to β^3 , $\frac{\partial(\cdot)}{\partial\beta^3}$

$\dot{\beta}$	derivative with respect to dimensionless sideslip rate, $\frac{\partial(\cdot)}{\partial \left(\frac{\beta b}{2V} \right)}$
$\dot{\delta}_a$	derivative with respect to angular deflection of ailerons, $\frac{\partial(\cdot)}{\partial \delta_a}$
$\dot{\delta}_a \alpha$	derivative with respect to $\delta_a \cdot \alpha$, $\frac{\partial(\cdot)}{\partial (\delta_a \cdot \alpha)}$
$\dot{\delta}_e$	derivative with respect to angular deflection of elevator, $\frac{\partial(\cdot)}{\partial \delta_e}$
$\dot{\delta}_e \beta^2$	derivative with respect to $\delta_e \cdot \beta^2$, $\frac{\partial(\cdot)}{\partial (\delta_e \cdot \beta^2)}$
$\dot{\delta}_f$	derivative with respect to angular deflection of flaps, $\frac{\partial(\cdot)}{\partial \delta_f}$
$\dot{\delta}_r$	derivative with respect to angular deflection of rudder, $\frac{\partial(\cdot)}{\partial \delta_r}$
$\dot{\delta}_r \alpha$	derivative with respect to $\delta_r \cdot \alpha$, $\frac{\partial(\cdot)}{\partial (\delta_r \cdot \alpha)}$
Δp_t	derivative with respect to dimensionless pressure increase across the propeller, $\frac{\partial(\cdot)}{\partial \left(\frac{\Delta p_t}{\frac{1}{2} \rho V^2} \right)} = \frac{\partial(\cdot)}{\partial dpt}$

0.6 Superscripts.

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

0.7 Abbreviations.

AACS	Automatic Aircraft Control System
CD	Course Datum
c.g.	centre of gravity
DHC	De Havilland of Canada Ltd.
DME	Distance Measuring Equipment
DUT	Delft University of Technology
fpa	flight path acceleration
FCC	Flight Control Computer
FCS	Flight Control System
ILS	Instrument Landing System
NLR	Nationaal Lucht- en Ruimtevaartlaboratorium (Dutch Aerospace Laboratory)
ODE	Ordinary Differential Equation
SA	Standard Atmosphere
STOL	Short Take Off and Landing
TAS	True Airspeed, V
VOR	Very high frequency Omnidirectional Range



0.8 Reference frames and sign conventions.

0.8.1 Definitions.

The definitions of the reference frames used within this report are given below. F_M and F_R have been added to this list, although they will only be used in table 2-2 in section 2.2.2. 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 .

Measurement reference frame F_M .

The measurement reference frame $O_M X_M Y_M Z_M$ is a *left-handed* orthogonal reference system, used for correcting the stability and control derivatives of the aircraft if the c.g. position differs from the one used during the flight tests. For the 'Beaver' aircraft, the origin O_M lies in a point, resulting from the perpendicular projection of the foremost point of the wingchord parallel to the $O_B X_B Z_B$ -plane, on this $O_B X_B Z_B$ -plane (ref.[30]). The $X_M O_M Z_M$ -plane coincides with the $O_B X_B Z_B$ -plane. The positive X_M -axis points backwards, the positive Y_M -axis points to the left, and the positive Z_M -axis points upwards.

The body-fixed reference frame F_B .

The body-fixed reference frame of the aircraft is a right-handed orthogonal system $O_B X_B Y_B Z_B$. The origin O_B lies in the centre of gravity of the aircraft. The $X_B O_B Z_B$ 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 (ref.[11]). 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 special body-fixed reference frame F_R for the 'Beaver'.

The body-fixed reference frame F_R , which is defined specially for the 'Beaver' aircraft, is identical to F_B , with one exception: the origin O_R is placed in a body-fixed reference point, which has been selected to coincide with a c.g. position which was actually used during one flight. It has the following coordinates in F_M : $x = 0.5996$ m, $y = 0$ m, $z = -0.8815$ m (ref.[30]). F_R is used only to define the moments and products of inertia for the aircraft condition on which the aerodynamic model is based (see table 2-2 in chapter 2).

The stability reference frame F_S .

The stability reference frame $O_S X_S Y_S Z_S$ 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 $O_B X_B Z_B$ -plane (if the aircraft is

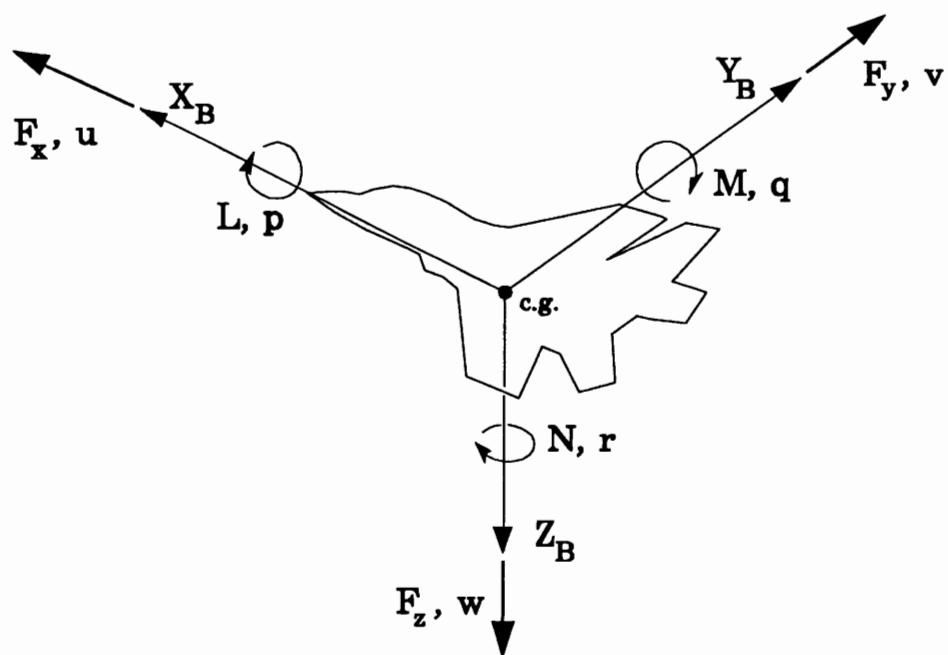


Figure 0-1. The body-axes reference frame F_B and the positive directions of the body-axes forces F_x, F_y, F_z , moments L, M, N , velocities u, v, w , and rotational velocities p, q, r .

symmetric, this is the plane of symmetry), or parallel to \mathbf{V} itself in case of a symmetrical nominal flight condition. The Y_S -axis coincides with the Y_B -axis.

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 [11], is a right-handed orthogonal system $O_EX_EY_EQ_Z_E$, 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, see section C.2 of appendix C. A beacon-fixed reference frame is used for simulating VOR navigation, see section C.5.



The vehicle-carried vertical axis reference system F_V .

The vehicle-carried vertical axis system $O_V X_V Y_V Z_V$ 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 .

In appendix C, a runway-fixed reference frame $O_F X_F Y_F Z_F$ will be introduced. 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 (as seen from an approaching aircraft), Z_F points downwards.

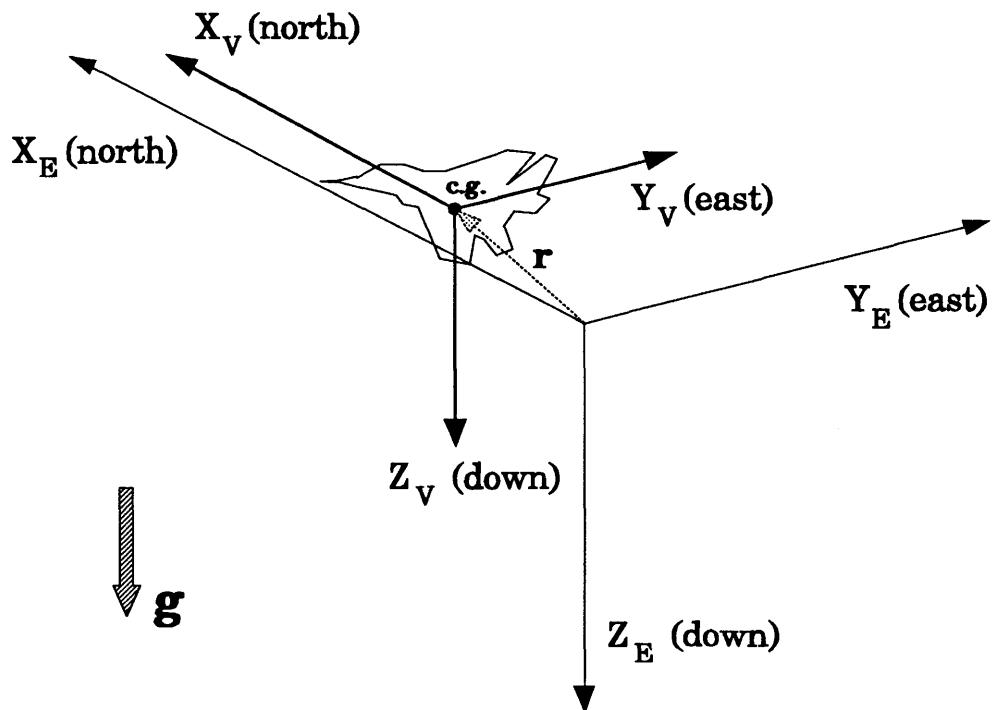


Figure 0-2. Relationship between the vehicle-carried vertical reference frame F_V and the earth-fixed reference frame F_E .

0.8.2 Relationships between the reference frames.

In figure 0-2 the relationship between the earth-fixed and vehicle-carried vertical axis systems 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:

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

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

The total transformation matrix from F_V to F_B then becomes:

$$\begin{aligned} T_{VB} &= \Phi \cdot \Theta \cdot \Psi = \\ &= \begin{bmatrix} \cos\psi \cos\theta & \sin\psi \cos\theta & -\sin\theta \\ \cos\psi \sin\theta \sin\varphi - \sin\psi \cos\varphi & \sin\psi \sin\theta \sin\varphi + \cos\psi \cos\varphi & \cos\theta \sin\varphi \\ \cos\psi \sin\theta \cos\varphi + \sin\psi \sin\varphi & \sin\psi \sin\theta \cos\varphi - \cos\psi \sin\varphi & \cos\theta \cos\varphi \end{bmatrix} \end{aligned} \quad (0-4)$$

so the relation between a vector \mathbf{y}_B in F_B and \mathbf{y}_V in F_V is:

$$\mathbf{y}_B = T_{VB} \cdot \mathbf{y}_V \quad (0-5)$$

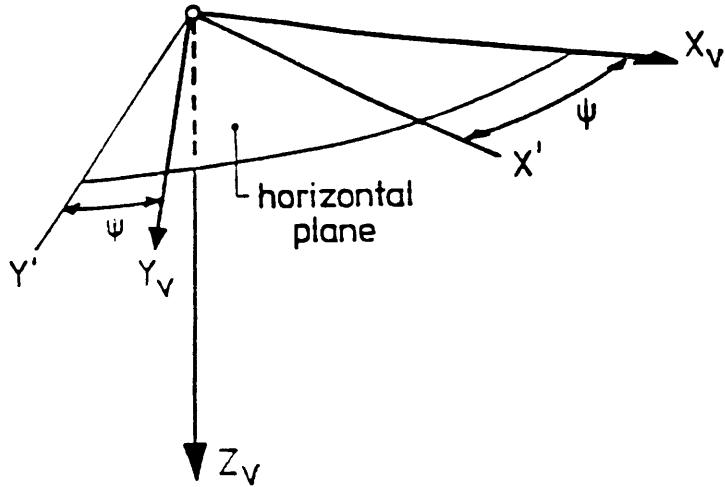
The orientation of the flight path axes with respect to the vehicle-carried vertical axes can also be expressed in terms of Euler angles (χ , γ and μ). This is shown in figure 0-4.

The relationships between the body, flight path, and stability axes are shown in figure 0-5. All three axis systems 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 with respect to the body axes is defined by the angle of attack α and the angle of sideslip β . The stability axis reference system is displaced from the flight path axis system by a rotation β and from the body axis system by a rotation $-\alpha$.

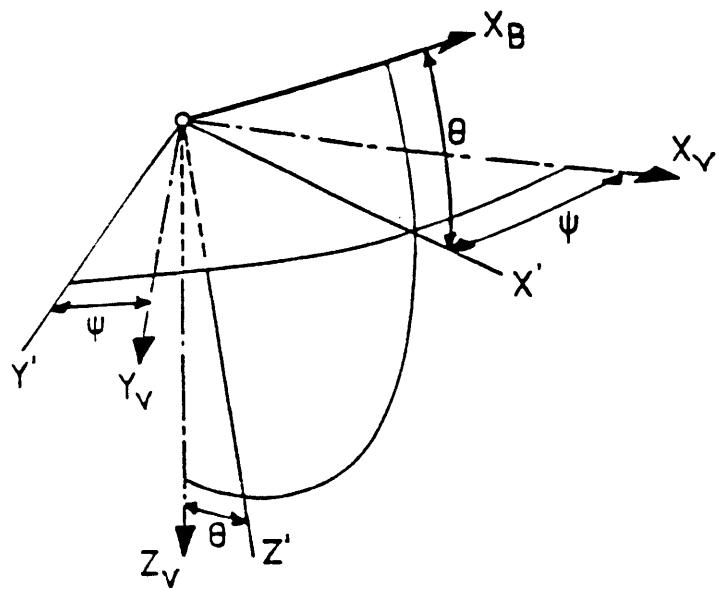
0.8.3 Sign conventions for deflections of control surfaces.

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.

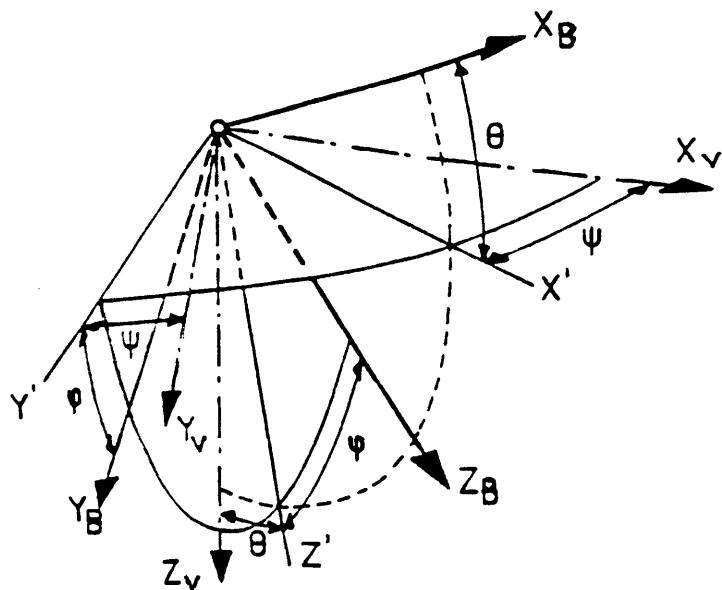




- A rotation by ψ about the Z_v -axis
to the intermediate position $X'YZ_v$.



- A rotation by θ about the Y -axis
to the intermediate position X_BYZ' .



- 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 .

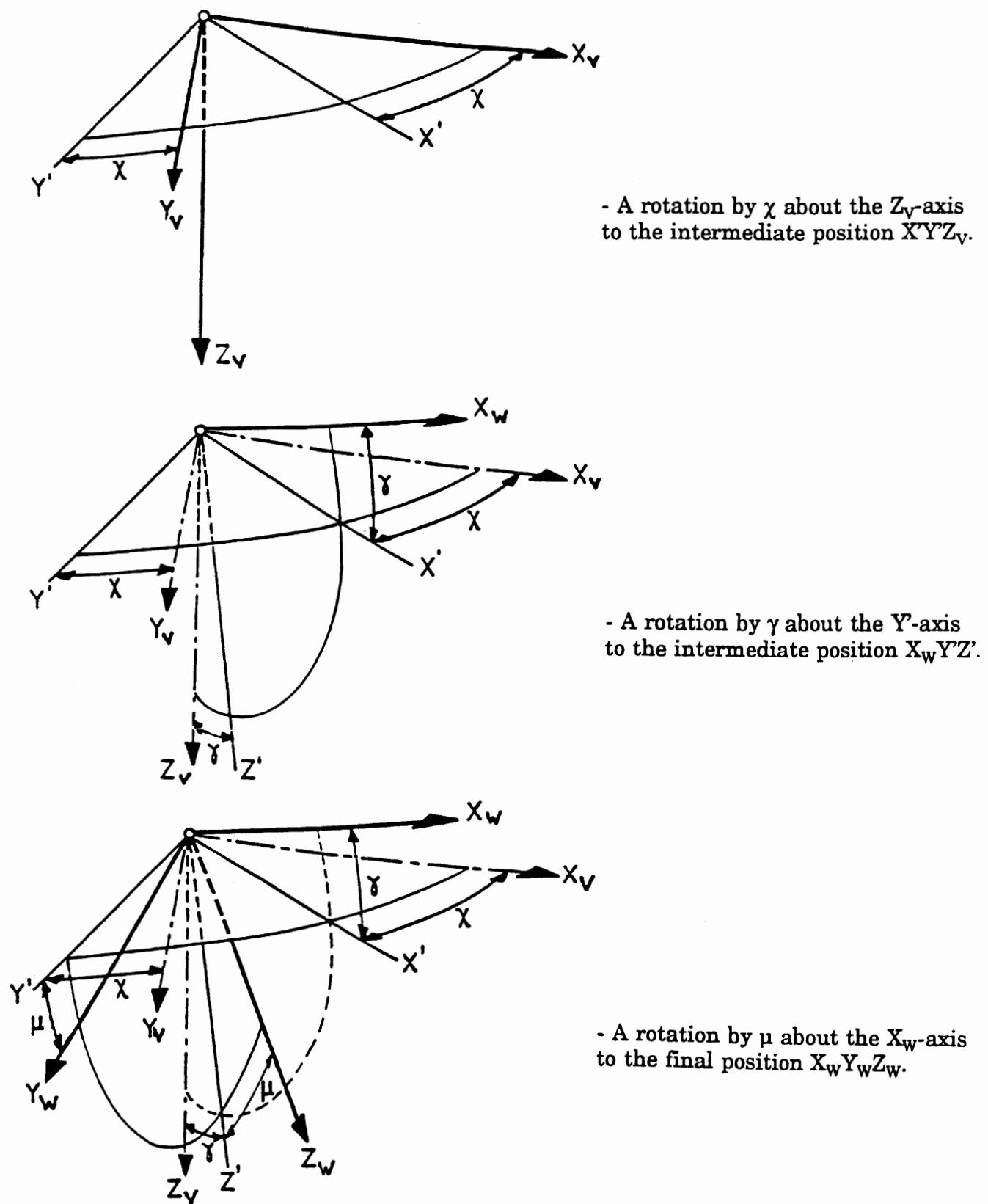


Figure 0-4. Relationship between the flight path reference frame F_w and the body-fixed reference frame F_B .



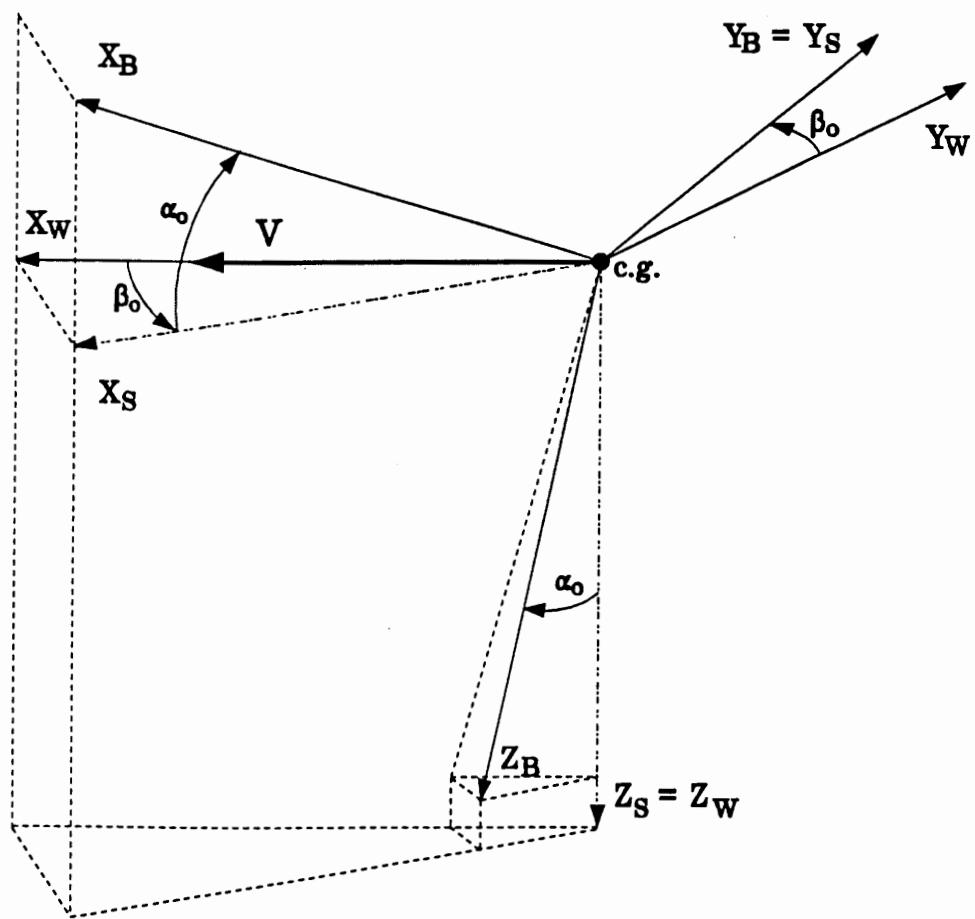


Figure 0-5. Relationships between body-fixed reference frame F_B , flight path reference frame F_W , and stability reference frame F_S .

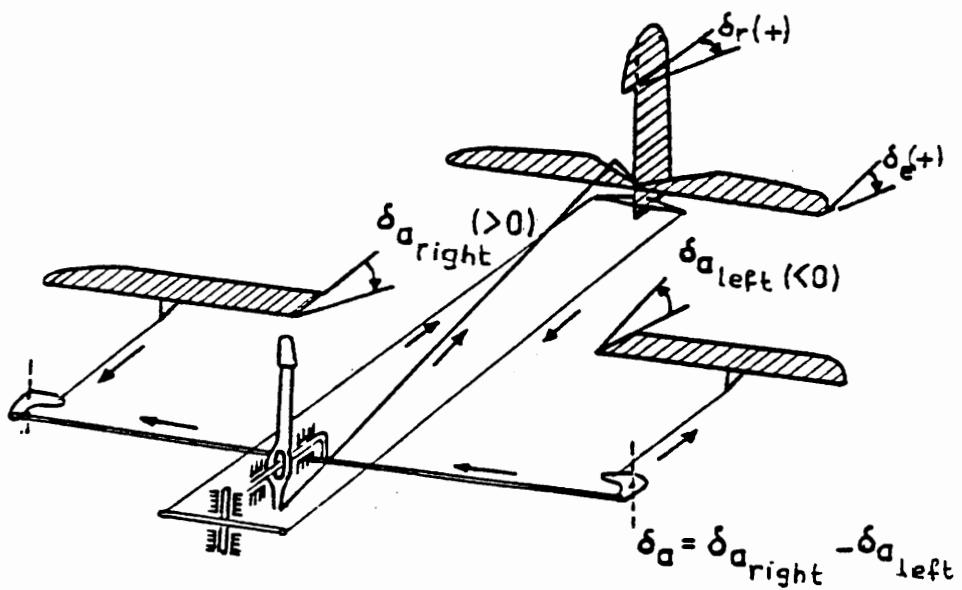


Figure 0-6. Sign conventions for control surface deflections.



Introduction to part I.

Active flight control technology plays an increasingly important role in aerospace engineering. For this reason, the Disciplinary Group for Stability and Control at the Faculty of Aerospace Engineering of Delft University of Technology plans to strongly increase its research efforts in this field. To do so, the faculty has recently acquired a new Cessna Citation II aircraft, which will be configured as 'National Fly-by-wire Testbed' (NFT).

The last autopilot project for the old laboratory aircraft, the De Havilland DHC-2 'Beaver', served as a demonstration to explore the technologies, needed to develop control laws from the first conceptual design all the way up to evaluation in real flight. All phases of the 'Beaver' autopilot development process will be thoroughly analyzed, to find out which parts of this process need to be improved further for the NFT project.

A major prerequisite for successful autopilot development is the availability of flexible tools for linear and nonlinear system analysis. For this reason, the design and testing of the new 'Beaver' autopilot has coincided with the development of a new software environment for aircraft dynamics and control analysis, which makes use of the power and flexibility of the system design and analysis programs SIMULINK and MATLAB. This environment will be treated in detail in this report.

The tools from this report have been worked out for the 'Beaver' aircraft, but they can be adapted easily for other aircraft, including the Citation II. Besides exploring new technologies for the NFT project, these models can be applied for many Stability and Control-related research and educational tasks. Part I of this report describes the tools themselves; in part II they will be applied to the analysis of the 'Beaver' autopilot. Submodels for the simulation of aircraft responses to atmospheric turbulence and the generation of VOR and ILS signals have been included.

Chapter 1 of part I gives a review of the AACs design process. The mathematical models of the 'Beaver' dynamics, wind, turbulence, VOR navigation, and ILS approach are summarized in chapter 2. Chapter 3 gives a short description of SIMULINK, the implementation of the models in SIMULINK is treated in chapter 4, and the models are validated in chapter 5. Chapter 6 shows how the results from chapter 5 were generated, to illustrate the practical use of the SIMULINK models. Appendices A to E describe the theory behind the models, appendix F shows all details of the block-diagrams and program listings, appendix G shows a list of standard SIMULINK blocks for readers who don't have any experience with SIMULINK yet, and appendix H shows how to install the programs on your own computer.

It is strongly recommended to use the tools in practice, to find out their real power, and also to reveal possible deficiencies of the environment. Your experiences can be used for further development of the tools.





Chapter 1. Design and evaluation of Automatic Aircraft Control Systems¹⁾.

1.1 Introduction.

Active control technology has dramatically changed the way aircraft are designed and flown; the flying qualities of modern aircraft are largely determined by a set of control laws in the heart of a computer system. This chapter will discuss the different phases in the control system design process, in order to make clear why a flexible environment for AACs analysis is needed. A detailed treatment of AACs themselves can be found in refs. [22], [23], and [29].

1.2 The AACs design process.

A practical division of the AACs design process into a number of phases is given in ref.[23]. Although ref.[23] is somewhat dated with respect to the available tools (computer software and hardware), the following division is still quite usable:

- 1 - Establish the system purpose and overall system requirements. System purpose can be equated with mission or task definitions. System requirements can be separated in (i) 'operational requirements', derived from the functions needed to accomplish the mission phases and (ii) 'implied requirements', derived from the characteristics of the interconnected components of the control system and the environment in which they operate.
- 2 - Determine the unalterable element, command, and disturbance environment characteristics. The characteristics of some parts of the system cannot easily be changed by the designer. Often the vehicle itself, its control surface actuators and some of its sensors are 'unalterable'. Moreover, the 'structure' of the commands and disturbances is a direct consequence of the mission requirements and the environment.
- 3 - Evolve competing feasible systems (determine the basic block diagrams). There are often more ways to achieve the requirements, e.g. with different sensed motion quantities and/or different control theories. Then it is possible to evolve competing candidate systems for selection on the basis of certain desirable properties.
- 4 - Select the 'best' system. The competing designs can be compared on the basis of (i) design qualities, which include dynamic performance (speed of response, bandwidth, etc.) and physical characteristics (weight, volume, power consumption, etc.), and (ii) design quantities, which

¹⁾ The term 'Automatic Aircraft Control Systems' (AACs) is used here in stead of the more common 'Automatic Flight Control Systems' (AFCSs), because it has become customary to denote the part of the 'Beaver' control system which converts electric signals from the Flight Control Computer (FCC) to actual deflections of the control surfaces as 'Flight Control System'. So here FCS is the part of the control system between the FCC and the control surfaces; AACs is the whole automatic control system, which includes the FCC.



include safety, reliability, maintainability, cost, etc. An optimum system is one that has some 'best' combination of these features.

- 5 - Study the selected system in detail. The selected system must be evaluated for all normal and abnormal operating conditions. At each state of the AACSt *validation* the assumptions, made earlier during the AACSt *design*, must be checked for validity. If necessary, a new iteration of the design should be started from the point where the wrong assumption was made.

This scheme reflects the AACSt design process within a manufacturing environment. In a research context, some modifications to this scheme have to be made. The task of research is to determine what is required and produce a clear and comprehensive definition of the requirements; the manufacturing task is to make and deliver in quantity a reliable and effective product [31]. The first phase in particular will look different in a research environment, because the system requirements are often poorly understood or may even become the objective of the research itself. In addition, the design tools may be immature, and their development may again be an objective of the research.

In a research context, the design, simulation, and implementation of control laws will be similar to the production application, although more flexibility of the tools will sometimes be required. It should, for instance, be possible to change the control laws within the Flight Control Computers (FCCs) of the aircraft rapidly, so that different solutions to typical control law design problems can be developed, evaluated, and compared simultaneously. For research purposes, different control system designs may be developed all the way up to flight test simultaneously, which means that step 4 does not necessarily need to include the selection of the 'best' system. The requirements with respect to fail-safety of the AACSt may be less restrictive in a research context than for manufacturing. The DHC-2 'Beaver' laboratory aircraft is, for instance, equipped with only one Flight Control Computer (a portable PC!), whereas AACSts for production aircraft often use multiple FCCs, which check each other's command signals.

The classical approach to stability and control analysis is to start with the complete equations of motion and make assumptions that enable these equations to be linearized about some local equilibrium point. During the initial AACSt design phase, linear system theory can be applied to these linear mathematical models of the aircraft dynamics, sensors, and actuators. Although these techniques will provide insight in the essential behaviour of the AACSt, only relatively small deviations from the equilibrium state are permitted before the models become invalid. Preliminary design studies can be done with simplified linear models, using programs like MATLAB, see ref.[3].

If linear analysis has yielded satisfying control laws, detailed simulations of the system must be made, to make sure that the AACSt behaves well over the part of the flight envelope for which it is designed. This often demands analysis over a wide range of flight speeds, altitudes, or aircraft configurations, and hence, *nonlinear* simulations. The off-line simulations can be done on a PC or workstation, using appropriate simulation software, such as SIMULINK and MATLAB, see refs.[3] and [4].

In a later stage of the AACCS development, on-line piloted simulations, using a real-time moving-base flightsimulator, are needed. This is particularly important if a pilot is involved in control tasks or system monitoring, i.e., for the assessment of fly-by-wire systems. The pilot must be supplied with information in a timely fashion so that he is able to use his normal control techniques and strategies [26].

Based upon the results of these linear and nonlinear, off-line and on-line simulations, the best solution can be chosen if more feasible AACSSs are available. Notice that if the actuators, sensors, computers, etc. are *not* 'unalterable', the effects of using different equipment should be included in the analysis. For the remainder of this report, all hardware is considered to be unalterable, so that the 'only' problem to be solved is the development of appropriate control laws for a given aircraft.

The control laws can be implemented in the Flight Control Computer(s) of the aircraft if the results of the simulations satisfy the requirements. Suitable actuators and sensors have to be installed and tested. Sometimes the complete control system may need to be tested in an 'Iron Bird' test stand arrangement, for instance for certification purposes.

After finishing simulations and ground tests, the control system can be evaluated in flight. Basically, this phase should be a verification of the previous results that were obtained by linear and nonlinear simulations. If the AACCS does not perform as expected, it may be necessary to make some changes, in which case it is necessary to return to a previous design stage. Then once again linear and/or nonlinear system analysis must be performed.

The *iterative* nature of designing and evaluating an AACCS should be acknowledged. At any stage in the process, the discovery of a fault, design error, or uncertainty requires the return to a previous stage. It is obvious that the control systems designer needs appropriate tools and mathematical models of the system dynamics at any stage in the design process. For this reason, it is very important that the transitions from one phase to another are as smooth as possible, in order to reduce the number of errors, which inevitably will arise if the tools for the different design stages are not 'compatible' (Murphy's Law).

It is therefore necessary that many different tools for linear and nonlinear design and analysis and a set of mathematical models of different complexity are accessible to a designer from a single PC or workstation. Furthermore, the software tools should be able to communicate effectively with each other. Examples of integrated control design environments are described in refs.[10] and [28]. These papers emphasize the need for *multidisciplinary design*, in which aerodynamic, structural, propulsive, and control functions are considered together.

Multidisciplinary design is important, because AACSSs form a more and more central part within the aircraft. AACSSs with mechanical linkages and limited authority have been replaced by full authority, fly-by-wire, digital control systems. The desire for the best flying qualities throughout the whole flight envelope has lead to controllers that excite structural modes and interact with the control-actuator dynamics. There is increasing need to integrate flight controls with engine controls and load-alleviation functions [28]. Interactions between the aerodynamic, propulsion, and structural



models must then be taken into account. Future aircraft will employ such features as extensive use of composite materials (resulting in greater aero-servoelastic coupling) and relaxed static stability, which means that these interacting phenomena will become even stronger [10].

This leads to the conclusion that the control system design tasks are becoming more and more challenging. Powerful tools for AACCS design and analysis should be employed to explore new design approaches. The increasing power of computer systems, PCs and workstations in particular, makes it possible to construct flexible, user-friendly Computer Aided Control System Design (CACSD) packages, which greatly simplify the control system design and analysis.

1.3 The place of this research within the AACCS design process.

In part I of this report, a first step towards the integration of the different AACCS design phases will be made. A nonlinear dynamical model of the 'Beaver' aircraft will be implemented as a graphical SIMULINK system, which can be adapted for other aircraft easily, due to its modular structure. The practical use of this model for open-loop analysis of the 'Beaver' dynamics will be illustrated in chapter 6. In part II of this report, the model will be applied to the nonlinear analysis of the 'Beaver' autopilot.

The SIMULINK model library from this report makes it possible to compare different controllers with a minimum of programming efforts. The nonlinear aircraft model in SIMULINK can be trimmed and linearized with the aircraft trim tool which will be described in appendix F and the SIMULINK linearization tools, to obtain linearized state-space models for the control system design in MATLAB. The models from this report have come a long way in integrating linear and nonlinear system analysis.

Further optimizations concerning flexibility, user-friendliness, and speed may be achievable, and for some applications it may be necessary to further enhance the models. In the future, the transitions from analysis on a PC or workstation to real-time piloted flightsimulation and implementation of control laws in the Flight Control Computer(s) of the actual aircraft should be made as easy as possible by developing appropriate tools which can be applied to the SIMULINK systems.

Besides the implementation of aircraft models, this report also describes models of atmospheric turbulence, wind, and ILS navigation signals. These blocks can easily be implemented in other SIMULINK systems. The structure of the autopilot control loops, presented in part II, can also serve as an example of how to use programs like SIMULINK for AACCS design & analysis.

All models described in this report have been collected in libraries, of which the library with the 'Beaver' model is the most important. Although the library is by no means complete, even this small collection of SIMULINK models already can be used for a wide range of applications, reaching beyond the straightforward use for nonlinear simulations of an aircraft, equipped with an AACCS, which will be shown in part II of this report. It is for instance possible to use the SIMULINK model of the aircraft for open-loop

model validation and system identification, because all contributions to the forces and moments acting upon the aircraft can be visualized easily. Remember that the full set of MATLAB toolboxes can be called from within SIMULINK!

If the models from this report are standardized further, and if models of other aircraft, actuators, sensors, navigational tools, etc. are added to the current model libraries, a very powerful centralized SIMULINK model library can be constructed, which gives the control system designer all the tools he/she needs. This is, of course, highly recommendable for the future research for the 'National Fly-by-wire Testbed' (NFT) project of the Disciplinary Group for Stability and Control.

In the nearby future, all actions from table 1.1 could be done within the MATLAB/SIMULINK environment. Compared to the Fortran tools which are currently used in the Disciplinary Group for Stability and Control (using the ENCORE computer system of the Disciplinary Group), the SIMULINK environment is much more flexible and user-friendly, which results in a greater productivity and less errors.

1. Estimate parameters of the aircraft model, using experimental data from flight tests and/or windtunnel experiments, or analytical data from tools like DATCOM. Apply the tools from the MATLAB SIGNAL IDENTIFICATION TOOLBOX to a generalized nonlinear model in SIMULINK to fit the aerodynamic model, engine model, etc.
2. Linearize the resulting nonlinear aircraft model with the *trim* tool from this report and the SIMULINK *linearization* tools. Analyze the system (stability, open-loop responses) in MATLAB. Analyze the influence of nonlinearities with SIMULINK.
3. Design and analyze control systems, using the CONTROL SYSTEMS TOOLBOX, the ROBUST CONTROL TOOLBOX, or the M-TOOLBOX from MATLAB (or any other appropriate toolbox).
4. Include nonlinear elements (e.g. limiters, time-delays, or general nonlinear functions) in the control system, and analyze their influence upon the overall AACS performance. Assess the control system with nonlinear simulations in all operating points. If necessary, optimize the system further.
5. Translate the control laws into Fortran or C for use within a piloted real-time flightsimulator, or for implementation within the Flight Control Computers of the Aircraft (this step should be automated in the future).

Table 1-1. Different steps in the aircraft model identification, performance analysis, and AACS design and evaluation, which can be done with SIMULINK.



1.4 Conclusions.

This chapter has outlined the different phases which are passed to develop a suitable control system (especially the control laws itself) from concept via linear and nonlinear simulation to implementation in the aircraft. As the analysis proceeds, the models become increasingly complex. The designer should be able to access a number of linear and nonlinear design and simulation tools, and mathematical models of the aircraft from one PC or workstation. Also, the transitions between different design steps should be made as smooth as possible, which imposes compatibility requirements on hardware and software. The objective of this report is to outline and demonstrate some possibilities of the programs SIMULINK and MATLAB for AACs design.



Chapter 2. Dynamic models of the 'Beaver', atmospheric turbulence, and navigation signals.

2.1 Introduction.

This chapter gives a survey of the mathematical models which will be used within the SIMULINK simulation structure. These models include the aircraft dynamics, force and moment models, wind, atmospheric turbulence, and navigation signals (ILS approach and VOR navigation). The aircraft equations of motion are treated in detail in appendix A. Appendix B contains more information about atmospheric turbulence, and appendix C gives a more detailed description about the ILS approach system. The equations of motion are very general, but the forces and moments which act upon the aircraft depend on the characteristics of the aircraft itself. In this report, the models of the De Havilland DHC-2 'Beaver' will be used. In table 2-1 some basic information about this aircraft is summarized. See also figures 2-1 and 2-2.

2.2 Aircraft equations of motion.

It is possible to express the aircraft dynamics as a set of nonlinear ordinary differential equations (ODEs). The state equations presented here are valid for rigid bodies. They express the motions of the aircraft in terms of external forces and moments, which can be subdivided in a number of categories. Only contributions to the forces and moments from aerodynamics, engine, gravity, and non-steady atmosphere will be considered here. In this section, the equations of motion will be presented along with all relevant force and moment equations and a large number of output equations of which some are needed to calculate these forces and moments.

2.2.1 General nonlinear equations of motion.

The derivation of the state equations is included in appendix A. There are six force and moment equations and six equations which determine the aircraft's attitude and position with respect to the earth. The translational equations are expressed in terms of true airspeed V , angle of attack α , and sideslip angle β in stead of the body axes velocity components u , v , and w (see appendix A). The state equations for V , α , β , p , q , and r are valid only when the following restrictive assumptions are made:

- 1 - the airframe is assumed to be a rigid body in the motion under consideration,
- 2 - the airplane's mass is assumed to be constant during the time interval in which its motions are studied,
- 3 - the earth is assumed to be fixed in space, i.e. its rotation is neglected,
- 4 - the curvature of the earth is neglected.

Notice that the aerodynamic forces in the V , α , and β -equations are expressed in terms of aerodynamic lift, drag, and sideforce in stead of body-axes aerodynamic forces. All other forces and all moments are given in their body-axes



components. See the section 'Symbols and definitions' for an explanation of all symbols.

Differential equations for airspeed, angle of attack, and sideslip angle:

$$\dot{V} = \frac{1}{m} [-\bar{D} \cos \beta + \bar{Y} \sin \beta + (X_{gr} + X_t + X_w) \cos \alpha \cos \beta + (Y_{gr} + Y_t + Y_w) \sin \beta + (Z_{gr} + Z_t + Z_w) \sin \alpha \cos \beta] \quad (2-1a)$$

$$\dot{\alpha} = \frac{1}{V_a \cos \beta} \left\{ \frac{1}{m} \left[-\bar{L} - (X_{gr} + X_t + X_w) \cos \alpha + (Z_{gr} + Z_t + Z_w) \sin \alpha \right] \right\} + q - (p \cos \alpha + r \sin \alpha) \tan \beta \quad (2-1b)$$

$$\dot{\beta} = \frac{1}{V_a} \left\{ \frac{1}{m} \left[\bar{D} \sin \beta + \bar{Y} \cos \beta - (X_{gr} + X_t + X_w) \cos \alpha \sin \beta + (Y_{gr} + Y_t + Y_w) \cos \beta - (Z_{gr} + Z_t + Z_w) \sin \alpha \sin \beta \right] \right\} + p \sin \alpha - r \cos \alpha \quad (2-1c)$$

Differential equations for the body-axes rotational velocities:

$$\begin{aligned} \dot{p} &= P_{pp} p^2 + P_{pq} p q + P_{pr} p r + P_{qq} q^2 + P_{qr} q r + P_{rr} r^2 + P_l L + P_m M + P_n N \\ \dot{q} &= Q_{pp} p^2 + Q_{pq} p q + Q_{pr} p r + Q_{qq} q^2 + Q_{qr} q r + Q_{rr} r^2 + Q_l L + Q_m M + Q_n N \\ \dot{r} &= R_{pp} p^2 + R_{pq} p q + R_{pr} p r + R_{qq} q^2 + R_{qr} q r + R_{rr} r^2 + R_l L + R_m M + R_n N \end{aligned} \quad (2-2)$$

Differential equations for the Euler angles:

$$\begin{aligned} \dot{\phi} &= p + q \sin \varphi \cdot \tan \theta + r \cos \varphi \cdot \tan \theta \\ \dot{\theta} &= q \cos \varphi - r \sin \varphi \\ \dot{\psi} &= q \frac{\sin \varphi}{\cos \theta} + r \frac{\cos \varphi}{\cos \theta} \end{aligned} \quad (2-3)$$

Differential equations for the aircraft coordinates:

$$\begin{aligned} \dot{x}_e &= \{u_e \cos \theta + (v_e \sin \varphi + w_e \cos \varphi) \sin \theta\} \cos \psi - (v_e \cos \varphi - w_e \sin \varphi) \sin \psi \\ \dot{y}_e &= \{u_e \cos \theta + (v_e \sin \varphi + w_e \cos \varphi) \sin \theta\} \sin \psi + (v_e \cos \varphi - w_e \sin \varphi) \cos \psi \\ \dot{z}_e &= -u_e \sin \theta + (v_e \sin \varphi + w_e \cos \varphi) \cos \theta \end{aligned} \quad (2-4)$$

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 (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, $p_z = 26^{\prime\prime}$ 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% c at 1725 kgf = 16989 N
Most backward admissible c.g. position	29.92% c at 2315 kgf = 22800 N
	40.24% c

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

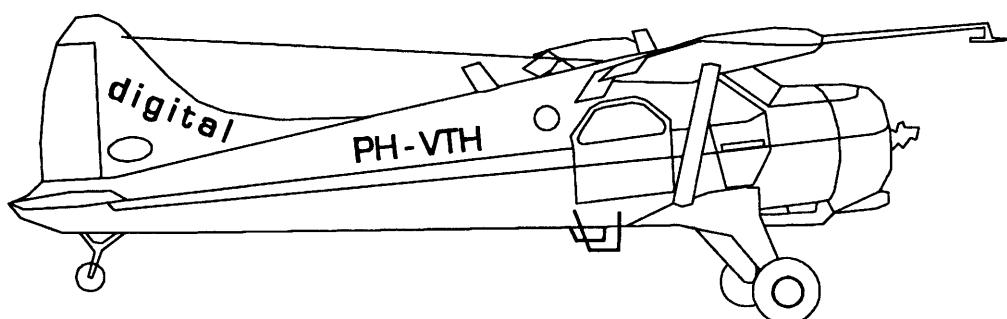


Figure 2-1. The De Havilland DHC-2 'Beaver' laboratory aircraft.



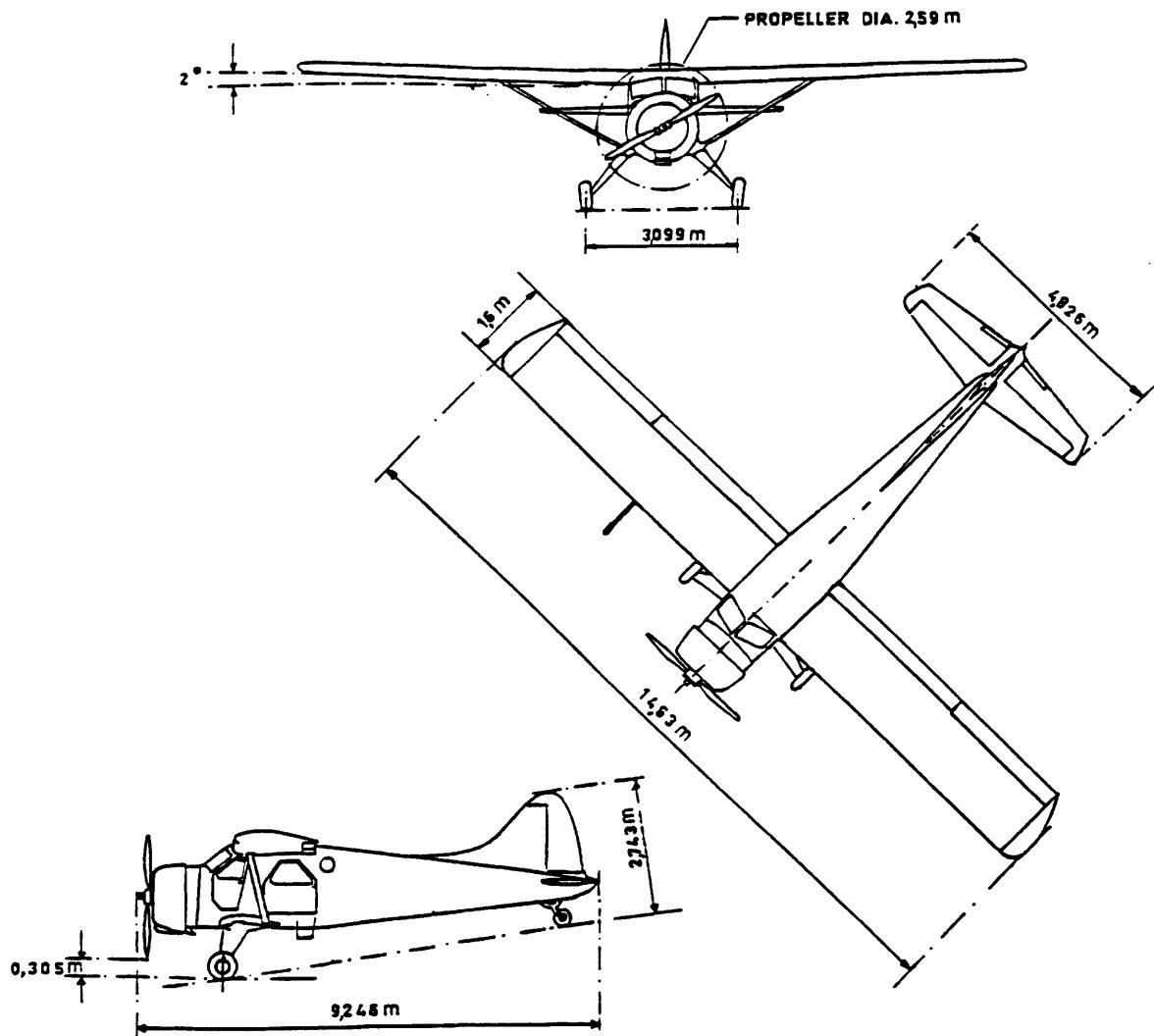


Figure 2-2. Basic dimensions of the DHC-2 'Beaver'.

2.2.2 Force and moment models.

Only four contributions to the external forces and moments will be considered in this report:

- 1 - aerodynamic forces and moments,
- 2 - contributions to the forces and moments due to operation of the power-plant,
- 3 - force contributions from gravity,
- 4 - force contributions from wind and/or atmospheric turbulence.

Aerodynamic forces & moments model for the DHC-2 'Beaver'.

The aerodynamic force and moment coefficients for the 'Beaver' can be written as a set of nonlinear polynomials, see ref.[24]. The aerodynamic model includes effects like longitudinal-lateral cross-coupling and unsteady aerodynamics. The influence of compressibility is neglected, as airspeed is assumed to be low (a

reasonable assumption for the 'Beaver'). Also scale effects are not taken into account, since the effect of Reynolds number variations is considered to be negligible at the relatively high Reynolds numbers that occur in flight. The flight condition for which the model has been determined is listed in table 2-2. Corrections are necessary if a different position of the centre of gravity is used, see ref.[24].

$X_{c.g.}$	=	0.5996	[m]	in F_M
$Y_{c.g.}$	=	0.0	[m]	in F_M
$Z_{c.g.}$	=	-0.8851	[m]	in F_M
I_x	=	5368.39	[kgm^2]	in F_R
I_y	=	6928.93	[kgm^2]	in F_R
I_z	=	11158.75	[kgm^2]	in F_R
J_{xy}	=	0.0	[kgm^2]	in F_R
J_{xz}	=	117.64	[kgm^2]	in F_R
J_{yz}	=	0.0	[kgm^2]	in F_R
m	=	2288.231	[kg]	
h	=	1828.8	[m]	(= 6000 [ft])
ρ	=	1.024	[kg/m^3]	

Table 2-2. Aircraft data on which the aerodynamic model is based.

The aerodynamic model of the 'Beaver' can be written in terms of dimensionless body-axes force and moment coefficients [24]:

$$\begin{aligned}
 C_{X_a} &= C_{X_0} + C_{X_\alpha} \alpha + C_{X_{\alpha^2}} \alpha^2 + C_{X_{\alpha^3}} \alpha^3 + C_{X_q} \frac{q\bar{c}}{V} + C_{X_{\delta_r}} \delta_r + C_{X_{\delta_f}} \delta_f + C_{X_{\alpha\delta_f}} \alpha \delta_f \\
 C_{Y_a} &= C_{Y_0} + C_{Y_\beta} \beta + C_{Y_p} \frac{pb}{2V} + C_{Y_r} \frac{rb}{2V} + C_{Y_{\delta_a}} \delta_a + C_{Y_{\delta_r}} \delta_r + C_{Y_{\delta_r\alpha}} \delta_r \alpha + C_{Y_\beta} \frac{\dot{\beta}b}{2V} \\
 C_{Z_a} &= C_{Z_0} + C_{Z_\alpha} \alpha + C_{Z_{\alpha^3}} \alpha^3 + C_{Z_q} \frac{q\bar{c}}{V} + C_{Z_{\delta_e}} \delta_e + C_{Z_{\delta_e\beta^2}} \delta_e \beta^2 + C_{Z_{\delta_f}} \delta_f + C_{Z_{\alpha\delta_f}} \alpha \delta_f \\
 C_{l_a} &= C_{l_0} + C_{l_\beta} \beta + C_{l_p} \frac{pb}{2V} + C_{l_r} \frac{rb}{2V} + C_{l_{\delta_a}} \delta_a + C_{l_{\delta_r}} \delta_r + C_{l_{\delta_a\alpha}} \delta_a \alpha \\
 C_{m_a} &= C_{m_0} + C_{m_\alpha} \alpha + C_{m_{\alpha^2}} \alpha^2 + C_{m_q} \frac{q\bar{c}}{V} + C_{m_{\delta_e}} \delta_e + C_{m_{\beta^2}} \beta^2 + C_{m_r} \frac{rb}{2V} + C_{m_{\delta_f}} \delta_f \\
 C_{n_a} &= C_{n_0} + C_{n_\beta} \beta + C_{n_p} \frac{pb}{2V} + C_{n_r} \frac{rb}{2V} + C_{n_{\delta_a}} \delta_a + C_{n_{\delta_r}} \delta_r + C_{n_q} \frac{q\bar{c}}{V} + C_{n_{\beta^3}} \beta^3
 \end{aligned} \tag{2-5}$$



The values of the coefficients are listed in table 2-3. The dimensionless force and moment coefficients can be made non-dimensionless using the following equations:

$$\begin{aligned} X_a &= C_{X_a} \cdot \frac{1}{2} \rho V^2 S \\ Y_a &= C_{Y_a} \cdot \frac{1}{2} \rho V^2 S \\ Z_a &= C_{Z_a} \cdot \frac{1}{2} \rho V^2 S \end{aligned} \quad (2-6a)$$

and:

$$\begin{aligned} L_a &= C_{l_a} \cdot \frac{1}{2} \rho V^2 S b \\ M_a &= C_{m_a} \cdot \frac{1}{2} \rho V^2 S \bar{c} \\ N_a &= C_{n_a} \cdot \frac{1}{2} \rho V^2 S b \end{aligned} \quad (2-6b)$$

C_X		C_Y		C_Z	
parameter	value	parameter	value	parameter	value
0	-0.03554	0	-0.002226	0	-0.05504
α	0.002920	β	-0.7678	α	-5.578
α^2	5.459	$\frac{pb}{2V}$	-0.1240	α^3	3.442
α^3	-5.162	$\frac{rb}{2V}$	0.3666	$\frac{qc}{V}$	-2.988
$\frac{qc}{V}$	-0.6748	δ_a	-0.02956	δ_e	-0.3980
δ_r	0.03412	δ_r	0.1158	$\delta_e \beta^2$	-15.93
δ_f	-0.09447	$\delta_r \alpha$	0.5238	δ_f	-1.377
$\alpha \delta_f$	1.106	$\frac{\dot{b}}{2V}$	-0.1600	$\alpha \delta_f$	-1.261

**Table 2-3 (i). Coefficients in the nonlinear aerodynamic model of the DHC-2 'Beaver', valid within the 35-55 m/s TAS-range
(continued on next page).**

C_l		C_m		C_n	
parameter	value	parameter	value	parameter	value
0	0.0005910	0	0.09448	0	-0.003117
β	-0.06180	α	-0.6028	β	0.006719
$\frac{pb}{2V}$	-0.5045	α^2	-2.140	$\frac{pb}{2V}$	-0.1585
$\frac{rb}{2V}$	0.1695	$\frac{qc}{V}$	-15.56	$\frac{rb}{2V}$	-0.1112
δ_a	-0.09917	δ_e	-1.921	δ_a	-0.003872
δ_r	0.006934	β^2	0.6921	δ_r	-0.08265
$\delta_a \alpha$	-0.08269	$\frac{rb}{2V}$	-0.3118	$\frac{qc}{V}$	0.1595
		δ_f	0.4072	β^3	0.1373

Table 2-3 (ii). Coefficients in the nonlinear aerodynamic model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-range.

Often, aerodynamic forces are expressed in terms of lift, drag, and sideforce, rather than the body-axes components:

$$\begin{bmatrix} \bar{D} \\ \bar{Y} \\ \bar{L} \end{bmatrix} = \begin{bmatrix} -\cos \alpha & 0 & -\sin \alpha \\ 0 & 1 & 0 \\ \sin \alpha & 0 & -\cos \alpha \end{bmatrix} \cdot \begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix} \quad (2-7)$$

If this conversion is not made, the lift, drag, and sideforce in the state equations (2-1) need to be exchanged for the body-axes aerodynamic forces, using the inverse transform.

Engine forces & moments model for the DHC-2 'Beaver'.

The contributions from the engine to the external forces and moments, and the influence of changes in airspeed, can be expressed in terms of changes of $dpt = \frac{\Delta p_t}{\frac{1}{2}\rho V^2}$, see ref.[24]. Δp_t equals the difference between the total pressure in front of the propeller and the total pressure behind the propeller. For the



'Beaver', the relation between dpt , the airspeed V , and the engine power P can be written as:

$$\frac{\Delta p_t}{\frac{1}{2}\rho V^2} = a + b \left(\frac{P}{\frac{1}{2}\rho V^3} \right) \quad (2-8)$$

with $\frac{P}{\frac{1}{2}\rho V^3}$ in $\left[\frac{kW}{kg/s^3} \right]$ and: $a = 0.08696$, $b = 191.18$, see ref.[24]. The engine power in [Nm/s] can be calculated with the following expression:

$$P = 0.7355 \cdot \left\{ -326.5 + \left(0.00412(p_z + 7.4) \cdot (n + 2010) + (408.0 - 0.0965n) \cdot \left(1.0 - \frac{\rho}{\rho_0} \right) \right) \right\} \quad (2-9)$$

The dimensionless force and moments coefficients along the body axes can now be expressed in terms of dpt . These coefficients include slipstream effects, which are quite large for the 'Beaver' aircraft, as well as the gyroscopic effect of the propeller. The engine force and moment coefficients are:

$$\begin{aligned} C_{X_t} &= C_{X_{\Delta p_t}} \left(\frac{\Delta p_t}{\frac{1}{2}\rho V^2} \right) + C_{X_{\alpha \Delta p_t^2}} \cdot \alpha \left(\frac{\Delta p_t}{\frac{1}{2}\rho V^2} \right)^2 \\ C_{Y_t} &= 0 \\ C_{Z_t} &= C_{Z_{\Delta p_t}} \left(\frac{\Delta p_t}{\frac{1}{2}\rho V^2} \right) \\ C_{l_t} &= C_{l_{\alpha^2 \Delta p_t}} \cdot \alpha^2 \left(\frac{\Delta p_t}{\frac{1}{2}\rho V^2} \right) \quad (2-10) \\ C_{m_t} &= C_{m_{\Delta p_t}} \left(\frac{\Delta p_t}{\frac{1}{2}\rho V^2} \right) \\ C_{n_t} &= C_{n_{\Delta p_t^3}} \left(\frac{\Delta p_t}{\frac{1}{2}\rho V^2} \right)^3 \end{aligned}$$

The values of the coefficients are listed in table 2-4. Notice the absence of dynamics of the powerplant itself within these equations! Again we need to multiply the dimensionless coefficients by $\frac{1}{2}\rho V^2 S$, $\frac{1}{2}\rho V^2 S b$, or $\frac{1}{2}\rho V^2 S \bar{c}$, to obtain the actual values of the engine forces and moments; hence:

$$\begin{aligned} X_t &= C_{X_t} \cdot \frac{1}{2}\rho V^2 S \\ Y_t &= C_{Y_t} \cdot \frac{1}{2}\rho V^2 S \\ Z_t &= C_{Z_t} \cdot \frac{1}{2}\rho V^2 S \end{aligned} \quad (2-11a)$$

and:

$$\begin{aligned} L_t &= C_{l_t} \cdot \frac{1}{2}\rho V^2 S b \\ M_t &= C_{m_t} \cdot \frac{1}{2}\rho V^2 S \bar{c} \\ N_t &= C_{n_t} \cdot \frac{1}{2}\rho V^2 S b \end{aligned} \quad (2-11b)$$

C_X		C_Y		C_Z	
parameter	value	parameter	value	parameter	value
Δp_t	0.1161	-	-	Δp_t	-0.1563
$\alpha \cdot \Delta p_t^2$	0.1453				

C_l		C_m		C_n	
parameter	value	parameter	value	parameter	value
$\alpha^2 \cdot \Delta p_t$	-0.01406	Δp_t	-0.07895	Δp_t^3	-0.003026

$$\frac{\Delta p_t}{\frac{1}{2}\rho V^2} = \alpha + b \cdot \frac{P}{\frac{1}{2}\rho V^3} \quad \text{with } \alpha = 0.08696, b = 191.18.$$

Table 2-4. Coefficients in the nonlinear engine model of the DHC-2 'Beaver' aircraft, valid within the 35-55 m/s TAS-range.



Gravity forces model.

The contribution of the aircraft's weight W to the forces along the body-axes of the aircraft can be calculated if the Euler angles ψ , θ , and φ are known. For this reason, the relations (2-3) are essential for solving the equations of motion. The contributions of the weight to the forces along the body-axes are:

$$\begin{aligned} X_{gr} &= -W \cdot \sin \theta \\ Y_{gr} &= W \cdot \cos \theta \sin \varphi \\ Z_{gr} &= W \cdot \cos \theta \cos \varphi \end{aligned} \quad (2-12)$$

Additional contributions to the forces due to a non-steady atmosphere.

If the aircraft flies in non-steady atmosphere, corrections to the body-axes forces are necessary, see appendix A. These corrections are:

$$\begin{aligned} X_w &= -m(\dot{u}_w + q w_w - r v_w) \\ Y_w &= -m(\dot{v}_w - p w_w + r u_w) \\ Z_w &= -m(\dot{w}_w + p v_w - q u_w) \end{aligned} \quad (2-13)$$

where u_w , v_w , and w_w are the components of the wind velocity along the body-axes of the aircraft, which do not need to be constant in time.

2.2.3 Writing the $\dot{\beta}$ -equation explicitly.

As can be seen from equation (2-2), the aerodynamic force Y_a partly depends on $\dot{\beta}$. Since $\dot{\beta}$ is not available until *after* the evaluation of the state equation (2-1c), and $\dot{\beta}$ itself is a function of Y_a , the $\dot{\beta}$ -equation is implicit. Although it is possible to solve this equation numerically, this is not recommended because this would reduce the speed of the computations. In this case, it is easy to write the $\dot{\beta}$ -equation explicitly. For the 'Beaver', equation (2-1c) can be written as:

$$\begin{aligned} \dot{\beta} = \frac{1}{Vm} &\left[-(X_{gr} + X_a + X_t + X_w) \cos \alpha \sin \beta + (Y_{gr} + Y_a^* + Y_t + Y_w) \cos \beta + \right. \\ &\left. - (Z_{gr} + Z_a + Z_t + Z_w) + \frac{1}{2} \rho V^2 S \cdot C_{Y_\beta} \frac{\dot{\beta} b}{2V} \cos \beta \right] + p \sin \alpha - r \cos \alpha \end{aligned} \quad (2-14)$$

where Y_a^* denotes the aerodynamic force along the Y_B -axis, *without* the influence of $\dot{\beta}$. The $\dot{\beta}$ -term on the right hand side of equation (2-14) can easily be moved to the left:

$$\begin{aligned} \dot{\beta}^* = \dot{\beta} \left(1 - \frac{\rho S b}{4m} \cdot C_{Y_\beta} \cos \beta \right) = \frac{1}{Vm} &\left[-(X_{gr} + X_a + X_t + X_w) \cos \alpha \sin \beta + \right. \\ &\left. + (Y_{gr} + Y_a^* + Y_t + Y_w) \cos \beta - (Z_{gr} + Z_a + Z_t + Z_w) \right] + p \sin \alpha - r \cos \alpha \end{aligned} \quad (2-15)$$

Now the following calculation sequence can be used:

- 1 - calculate the external forces and moments, but neglect the $\dot{\beta}$ -term in the equation for C_{Y_α} ,
- 2 - calculate $\dot{\beta}^*$ by applying (2-15),
- 3 - calculate the true value of $\dot{\beta}$ with the following equation:

$$\dot{\beta} = \frac{\dot{\beta}^*}{1 - \frac{\rho S b}{4m} \cdot C_{Y_\beta} \cos \beta} \quad (2-16)$$

The aircraft-dependent corrections to the state equations can thence be separated from the aircraft-independent terms, which is desirable if a certain standardization of the aircraft models is required. See chapter 4 and appendix F for a treatment of the implementation of the 'Beaver' models in SIMULINK.

2.2.4 Atmosphere model.

A multiplication with $\frac{1}{2}\rho V^2 S$ was necessary to calculate the actual forces and moments. Hence, the air density ρ is needed for solving the equations of motion. For other aircraft, flying at higher speeds and experiencing more compressibility effects, the forces and moments may also depend on the Mach number. These quantities will be calculated along with a number of other variables such as the equivalent and calibrated airspeeds and Reynolds numbers.

For this purpose, the 'US Standard Atmosphere 1962' model will be used. A description of this idealized model of the earth's atmosphere can, for instance, be found in ref.[27]. The 'Beaver' flies in the troposphere (i.e., at altitudes between 0 and 11,000 metres above sea level). In the SA model, the air temperature in troposphere decreases linearly with increasing altitude:

$$T = T_0 + \lambda h \quad (2-17)$$

where:

- T = air temperature (in [K]),
- h = altitude above sea level (in [m]),
- T_0 = air temperature on sea level ($T_0 = 288.15$ [K]),
- λ = temperature gradient in troposphere ($\lambda = -0.0065$ [K/m]).

Combining this equation with the basic hydrostatic equation:

$$dp_s = -\rho g dh \quad (2-18)$$

and the ideal gas law:

$$\frac{p_s}{\rho} = \frac{R_a}{M} T \quad (2-19)$$



yields:

$$\frac{dp_s}{p_s} = - \frac{Mg}{R_a T} dh \quad (2-20)$$

where:

p_s = air pressure,
 g = gravitational acceleration ($g = g_0 = 9.80665 \text{ [m/s}^2\text{]}$),
 M = molecular weight of the air ([kg/kmol]),
 R_a = molar gas constant ($R_a = 8314.32 \text{ [J/K}\cdot\text{kmol]}$).

The static air pressure p_s can be found by integrating (2-20), which yields:

$$\ln\left(\frac{p_s}{p_0}\right) = - \frac{g}{\lambda R} \ln\left(\frac{T_0 + \lambda h}{T_0}\right) \quad (2-21)$$

which can be written as:

$$\frac{p_s}{p_0} = \left(1 + \frac{\lambda h}{T_0}\right)^{-\frac{g}{\lambda R}} = \left(\frac{T_0}{T}\right)^{\frac{g}{\lambda R}} \quad (2-22)$$

where:

p_0 = air pressure on sea level ($p_0 = 101325 \text{ [N/m}^2\text{]}$),
 R = specific gas constant ($R = R_a / M_0$, with $M_0 = 28.9644 \text{ [kg/kmol]}$ = molecular weight of the air on sea level).

The acceleration of gravity g was held constant during this integration. It is actually necessary to replace the geometrical altitude h by the geopotential altitude H ¹⁾, but the slight distinction between h and H will be neglected here, in view of the altitudes considered. In other words: we will assume that $g = g_0$ throughout the whole flight envelope of the aircraft.

The airdensity ρ (in kg/m^3) is calculated from p_s and T using the ideal gas law (2-19), which yields:

$$\rho = \frac{p_s M}{R_a T} = \frac{p_s}{RT} \quad (2-23)$$

¹⁾ $H = \int_0^h \frac{g}{g_0} dh$

The dynamic pressure q_{dyn} is defined as:

$$q_{dyn} = \frac{1}{2} \rho V^2 \quad (2-24)$$

The impact pressure q_c can be calculated with the following equation:

$$q_c = p_s \left\{ \left(1 + \frac{\gamma - 1}{2\gamma} \frac{\rho}{p_s} V^2 \right)^{\frac{\gamma}{\gamma-1}} - 1 \right\} \quad (2-25)$$

where:

γ = ratio of the specific heats of air with constant pressure and constant volume, respectively ($\gamma = 1.4$).

Next, the total temperature T_t is calculated:

$$T_t = T \left(1 + \frac{\gamma - 1}{2} M^2 \right) \quad (2-26)$$

Some other variables which are related to the aircraft velocity will be calculated too. The Mach number is defined as:

$$M = \frac{V}{a} \quad (2-27)$$

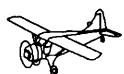
where a is the speed of sound (in [m/s]):

$$a = \sqrt{\gamma R T} \quad (2-28)$$

The calibrated and equivalent airspeed (V_c and V_e) can be calculated with the following equations:

$$V_c = \sqrt{\frac{2\gamma}{\gamma-1} \frac{p_0}{\rho_0} \left\{ \left(1 + \frac{q_c}{p_0} \right)^{\frac{\gamma}{\gamma-1}} - 1 \right\}} \quad (2-29)$$

$$V_e = V \cdot \sqrt{\frac{\rho}{\rho_0}} = \sqrt{\frac{2q_{dyn}}{\rho_0}} \quad (2-30)$$



If scale effects are taken into account, the Reynolds number needs to be known. The Reynolds number with respect to \bar{c} equals:

$$R_c = \frac{\rho V \bar{c}}{\mu} \quad (2-31)$$

and the Reynolds number per unit length (in [1/m]) is:

$$R_e = \frac{\rho V}{\mu} \quad (2-32)$$

The coefficient of the dynamic viscosity μ , which appears in the equations (2-31) and (2-32) can be calculated with Sutherland's equation [27]:

$$\mu = \frac{1.458 \cdot 10^{-6} T^{3/2}}{T + 110.4} \quad (2-33)$$

For solving the equations of motion for the 'Beaver', the dynamic pressure is needed to calculate the external forces and moments acting upon the airplane. This means that p , T , ρ , and q_{dyn} always have to be calculated to solve the equations of motion. Inclusion of other variables may be necessary to calculate the external forces and moments for other aircraft. For the implementation within SIMULINK, these equations will be logically subdivided in a number of 'atmosphere' and 'airdata'-groups, see appendix F.

2.2.5 Additional output equations.

It is possible to include a number of additional output equations to the state variables, state derivatives, forces and moments, and the atmosphere and airdata variables. Here, a list of normalized accelerations and specific forces will be included, as well as some flightpath (-related) variables.

Accelerations and specific forces.

It is possible to calculate a number of different accelerations and accelerometer outputs, which are often important in the aircraft control analysis and design problem (consider for example turncoordination by applying feedback of the acceleration along the Y_B -axis, or manoeuvre load limiting). Here, accelerations in the vehicle's centre of gravity will be considered. The expressions are reproduced from ref.[11].

The body axis acceleration vector \mathbf{a} can be expressed as:

$$\mathbf{a} = \dot{\mathbf{V}} = \frac{\partial \mathbf{V}}{\partial t} + \boldsymbol{\omega} \times \mathbf{V} \quad (2-34)$$

where ω is the rotational velocity vector of the aircraft. One should note that the body-axes velocity rates \dot{u} , \dot{v} , and \dot{w} differ from the body axis accelerations: they contain not only the body axis velocity rates \dot{u} , \dot{v} , and \dot{w} , but also rotational velocity and translational velocity cross-product terms. Expanding (2-34) into its components along the X_B , Y_B , and Z_B -axis and substituting for \dot{u} , \dot{v} , and \dot{w} (see section A.2.4 of appendix A) yields:

$$\begin{aligned} a_{x,k} &= \frac{1}{g_0} (\dot{u} + q w - r v) = \frac{1}{W} (X_{gr} + X_t + X_a + X_w) \\ a_{y,k} &= \frac{1}{g_0} (\dot{v} + r u - p q) = \frac{1}{W} (Y_{gr} + Y_t + Y_a + Y_w) \\ a_{z,k} &= \frac{1}{g_0} (\dot{w} + p v - q u) = \frac{1}{W} (Z_{gr} + Z_t + Z_a + Z_w) \end{aligned} \quad (2-35)$$

The accelerations are measured in units of g , hence the division by g_0 (notice that the acceleration due to gravity is assumed to be constant throughout this report). W is the aircraft's weight. The index k is used to define the *kinematic* accelerations in the body X_B , Y_B , and Z_B axis, respectively.

The outputs of body axis accelerometers (usually called '*specific forces*') at the vehicle center of gravity are simply equal to the body accelerations due to the thrust and aerodynamic forces and the forces due to non-steady atmosphere. These are derived from equation (2-35) by eliminating the gravity terms:

$$\begin{aligned} A_x &= a_{x,k} + \sin \theta &= \frac{1}{W} (X_t + X_a + X_w) \\ A_y &= a_{y,k} - \cos \theta \cdot \sin \varphi &= \frac{1}{W} (Y_t + Y_a + Z_w) \\ A_z &= a_{z,k} + \cos \theta \cdot \cos \varphi &= \frac{1}{W} (Z_t + Z_a + Z_w) \end{aligned} \quad (2-36)$$

A_x , A_y , and A_z are also measured in units of g .

Flightpath (-related) variables.

The flightpath angle γ is added to the list of outputs too. It is calculated with the following expression:

$$\gamma = \arcsin \left(\frac{\dot{H}}{V} \right) \quad (2-37)$$

The flightpath acceleration fpa is the acceleration in the direction of \mathbf{V} , measured in units of g :

$$fpa = \frac{\sqrt{\dot{u}^2 + \dot{v}^2 + \dot{w}^2}}{g_0} \quad (2-38)$$



Finally, the azimuth angle χ and the bank angle Φ are added to the list of outputs, using the following equations:

$$\begin{aligned}\chi &= \beta + \psi \\ \Phi &= \varphi \cdot \cos(\theta - \alpha_0)\end{aligned}\quad (2-39)$$

2.3 Wind and atmospheric turbulence.

For the evaluation of aircraft control systems, it is necessary to include simulations of the aircraft responses to wind and atmospheric turbulence. In this section, only the most important expressions will be described. A more detailed treatment of wind and atmospheric turbulence can be found in refs.[1] and [24] and in appendix B of this report.

Wind and wind shear.

First of all, it is necessary to distinguish between wind and atmospheric turbulence. Wind is the mean or steady-state velocity of the atmosphere with respect to the earth at a given position. Usually, the mean wind is measured over a certain time interval of several minutes. The remaining fluctuating part of the wind velocity is then defined as atmospheric turbulence.

The wind velocity and direction with respect to the ground is usually not constant along the flightpath. This wind variation is called wind shear. Since the wind velocity varies most strongly as a function of altitude in the earth's boundary layer ($H <$ approximately 300 m), it is particularly important to consider wind shear during the simulation of approach and landing or take-off and climb. Ref.[1] gives the following idealized equation of the wind velocity as a function of altitude:

$$\begin{aligned}V_w &= V_{w_{9.15}} \frac{h^{0.2545} - 0.4097}{1.3470} & (0 < h < 300\text{ m}) \\ V_w &= 0.2545 & (h \geq 300\text{ m})\end{aligned}\quad (2-40)$$

where $V_{w_{9.15}}$ is the wind speed at 9.15 m altitude. More extreme wind profiles in lower atmosphere have often been measured [1] and have sometimes resulted in accidents. Actual measurements of extreme wind profiles can therefore play an important role in the assessment of approach and landing or take-off and climb control functions of AACSSs.

In equation (2-13), the wind velocity components along the aircraft's body-axes were needed. If the direction of the wind, measured relatively to the earth, is denoted as ψ_w , where $\psi_w = 0^\circ$ if the wind is directed southwards, the following transformation equations can be derived (see section B.2):

$$\begin{aligned}u_w &= V_w \cos(\psi_w - \pi) \cos \psi + V_w \sin(\psi_w - \pi) \sin \psi \\ v_w &= -V_w \cos(\psi_w - \pi) \sin \psi + V_w \sin(\psi_w - \pi) \cos \psi\end{aligned}\quad (2-41)$$

If the influence of atmospheric turbulence is considered too, the velocity components of the turbulence along the body-axes should be added to these wind velocity components.

Atmospheric turbulence.

It is possible to model the components of atmospheric turbulence along the aircraft's body-axes as white noise, passing through linear filters. A detailed description of such turbulence filters can be found in refs.[1] and [24], and in appendix B of this report. The following forming filters are based upon the Dryden turbulence spectra, which have been derived from actual measurements of turbulence velocities, see appendix B:

$$H_{u_g w_1}(\omega) = \sigma_u \sqrt{\frac{2L_u}{V}} \frac{1}{1 + \frac{L_u}{V} j \omega} \quad (2-42)$$

$$H_{v_g w_2}(\omega) = \sigma_v \sqrt{\frac{L_v}{V}} \frac{1 + \sqrt{3} \frac{L_v}{V} j \omega}{\left(1 + \frac{L_v}{V} j \omega\right)^2} \quad (2-43)$$

$$H_{w_g w_3}(\omega) = \sigma_w \sqrt{\frac{L_w}{V}} \frac{1 + \sqrt{3} \frac{L_w}{V} j \omega}{\left(1 + \frac{L_w}{V} j \omega\right)^2} \quad (2-43)$$

w_1 , w_2 , and w_3 are independent white noise signals, L_u , L_v , and L_w are scale lengths, and σ_u , σ_v , and σ_w are standard deviations. Better approximations of the turbulence velocities can be made, see ref.[1], but these filters can be easily implemented within simulation models. It is also possible to use actual measured turbulence velocities, if very high accuracy is needed.

2.4 VOR navigation and ILS approach system signals.

If an AACCS with navigational functions is to be evaluated, it is necessary that signals from navigational aids, such as VOR and ILS stations, are calculated. The characteristics of these simulated signals should match the signals as they would be measured within the real aircraft. This means, for instance, that signal noise and steady-state errors must be included. In this report, only ILS and VOR signals will be treated, because the 'Beaver' autopilot does not use other navigational aids. In the future, MLS and GPS models should be added. A detailed description of the ILS and VOR systems can be found in refs.[1] and [5], and in appendix C of this report.

ILS signals.

To calculate the signals from an ILS transmitter, it is first necessary to know the position of the aircraft relatively to the runway. For this purpose, a runway-fixed reference frame $F_F = X_F Y_F Z_F$ is introduced. X_F is directed along the



runway centerline in landing direction, Z_F points downwards, and Y_F points to the right, as seen from an approaching aircraft. The origin of the earth-fixed reference frame F_E is chosen to coincide with the projection of the point of intersection of the runway threshold and the centerline of the runway (the origin of F_F) on the horizontal plane at sea level. The transformation from F_E to F_F then becomes:

$$\begin{aligned}x_f &= x_e \cos \psi_{RW} + y_e \sin \psi_{RW} \\y_f &= -x_e \sin \psi_{RW} + y_e \cos \psi_{RW} \\H_f &= H - H_{RW} \quad (= -z_f)\end{aligned}\tag{2-44}$$

where x_f , y_f , and z_f are the coordinates in the reference frame F_F , H_f is the altitude of the aircraft above sea level, H_{RW} is the altitude of the runway above sea level, and ψ_{RW} is the heading of the runway.

The ILS system provides guidance in two planes. The *localizer* reference plane is a vertical plane, passing through the runway centerline, which provides guidance in lateral direction. The instruments in the aircraft measure the angle between the line through the projection of the aircraft on the ground and the localizer antenna, and the runway centerline Γ_{loc} , which is equal to:

$$\Gamma_{loc} = \arcsin\left(\frac{d_{loc}}{R_{loc}}\right)\tag{2-45}$$

where:

$$R_{loc} = \sqrt{y_f^2 + (x_{loc} - x_f)^2}\tag{2-46}$$

and:

$$d_{loc} = y_f\tag{2-47}$$

R_{loc} is the ground distance from aircraft to localizer antenna, d_{loc} is the lateral deviation of the aircraft from the localizer reference plane, and x_{loc} is the distance from the runway threshold to the localizer antenna (measured along the X_F -axis).

The *glideslope* reference plane, which is actually a cone, provides guidance in longitudinal direction. The line of intersection of the glideslope reference cone and the localizer reference plane should provide a straight glide path with a reference glide path angle of about 3° . In practice, however, the glideslope antenna is located at some 300 metres besides the runway centerline, so the glideslope actually looks like a hyperbola. See section C.2 of appendix C for more details.

The angle ϵ_{gs} between the line through the aircraft and glideslope transmitter, and the reference cone is measured. If γ_{gs} is the reference glide

path angle, ϵ_{gs} can be calculated as follows:

$$\epsilon_{gs} = \gamma_{gs} + \arctan\left(\frac{H_f}{R_{gs}}\right) \quad (2-48)$$

where:

$$R_{gs} = \sqrt{(x_{gs} - x_f)^2 + (-y_{gs} + y_f)^2} \quad (2-49)$$

R_{gs} is the ground-distance from the aircraft to the glideslope antenna, x_{gs} is the X_F-coordinate of the glideslope antenna, and y_{gs} is the Y_F-coordinate of the glideslope antenna.

In section C.2 of appendix C, it is shown that the strength of the glideslope and localizer reference signals is only satisfactory in a limited area around the antennas on the ground. Furthermore, appendix C gives some ICAO-guidelines for maximum permissible localizer and glideslope steady-state errors and some ILS noise characteristics. Just like atmospheric turbulence, ILS noise can be modelled as white noise, passing through linear filters. Two sets of filters for localizer and glideslope noise have been given in appendix C:

From ref.[1]:

$$H_{loc}(\omega) = \sigma_{loc} \sqrt{\frac{2L_{loc}}{V}} \frac{1}{1 + \frac{L_{loc}}{V} j\omega} \quad (2-50)$$

$$H_{gs}(\omega) = \sigma_{gs} \sqrt{\frac{2L_{gs}}{V}} \frac{1}{1 + \frac{L_{gs}}{V} j\omega} \quad (2-51)$$

and from ref.[20]:

$$S_{loc}(\omega) = |H_{loc}(\omega)|^2 = \frac{25 (1.5 + j\omega)^2}{(0.35 + j\omega)^2 (10 + j\omega)^2} \left[\frac{\mu A^2}{rad/s} \right] \quad (2-52)$$

$$S_{gs}(\omega) = |H_{gs}(\omega)|^2 = \frac{15.9}{(0.25 + j\omega)^2} \left[\frac{\mu A^2}{rad/s} \right] \quad (2-53)$$

See section C.2 of appendix C for more details.



The VOR signals.

The VOR system is the standard short-range radio navigation aid. The VOR signal, received by the aircraft, depends linearly on the angle Γ_{VOR} between some reference VOR-radian, called the *Course Datum* (CD) and the actual bearing of the aircraft, denoted as QDR (a radio navigation term). The angle Γ_{VOR} now equals:

$$\Gamma_{VOR} = CD - QDR \quad (2-54)$$

where:

$$QDR = \arctan \left(\frac{y_e - y_{VOR}}{x_e - x_{VOR}} \right) \quad (2-55)$$

In this case, the coordinates of the aircraft, relative to the earth-fixed reference frame F_E are x_e , y_e , and z_e (H_e). The coordinates of the VOR station, relative to F_E are x_{VOR} , y_{VOR} , and z_{VOR} (H_{VOR}).

The VOR signals have also a limited coverage, and can not be received accurately if the aircraft enters the *cone of silence*. See section C.3.2 of appendix C. In section C.3.3, some remarks about VOR steady-state errors are made.

2.5 Conclusions.

To simulate an aircraft for the evaluation of automatic control systems, mathematical models for all relevant (sub-) systems are needed. This chapter has presented a survey of these models¹⁾, starting with the general nonlinear equations of motion, models for the external forces and moments, and a large number of output equations. Furthermore, a description of wind and atmospheric turbulence has been added, so that it is possible to evaluate control systems in conditions where external disturbances affect the flight path of the aircraft. Finally, a model of VOR navigation and ILS approach signals has been treated. See appendices A, B, and C for more details.

¹⁾ Models for the Flight Control System and engine dynamics of the 'Beaver' will be treated in part II of this report.



Chapter 3. Analysis of nonlinear dynamical systems with SIMULINK.

3.1 Introduction.

This chapter gives a short description of the basic functions of SIMULINK (version 1.2), that will be necessary to understand how the nonlinear aircraft models from chapter 2 can be implemented within the SIMULINK environment. Here, only a short introduction will be given; this chapter it is not intended to serve as a replacement for the SIMULINK User's Guide, ref.[4]. Also, a short lists of advantages and disadvantages of SIMULINK, compared with other simulation environments, will be given. Some theoretical aspects regarding simulation of nonlinear systems are treated in appendix D. A list of frequently used SIMULINK blocks is included in appendix G.

3.2 The programs SIMULINK and MATLAB.

MATLAB (ref.[3]) is a software package from The MathWorks Inc., conceived especially for matrix and vector calculations. It contains some standard commands, which can be called by the user from the command line of the MATLAB workspace in an interactive way. It is also possible to write MATLAB programs or functions, so called 'M-files', which can be treated as standard MATLAB commands. Finally, it is also possible to use Fortran and/or C programs from the MATLAB command line or from within other MATLAB programs, if these programs are structured as MATLAB executables, called 'MEX-files'.

Many MATLAB software libraries ('toolboxes'), for many special purposes are available. The CONTROL SYSTEM TOOLBOX contains a library of tools for the design analysis of 'classical' control systems, including routines for the calculation of root-loci, Bode and Nyquist diagrams, and time responses of feedback systems. However, classical control theory requires linearity of the systems considered, whereas the systems to be controlled are often nonlinear. Evaluation of nonlinear systems is possible with the simulation program SIMULINK (ref.[4]), which is an addition to standard MATLAB with functions that reach beyond the standard toolbox structure.

SIMULINK adds tools for the analysis of nonlinear dynamical systems, and a new graphical user environment, running under Microsoft WINDOWS when used on a PC. Nonlinear systems can be constructed as block-diagrams, using a number of standard graphical SIMULINK blocks (see appendix G). It is possible to call M or MEX-files from within SIMULINK systems, and to connect several SIMULINK systems to eachother. The combination of SIMULINK, and MATLAB yields a very powerful environment for (control) system research.

Figure 3-1 shows the relations between the programs. All variables are available in the MATLAB workspace, either locally within a MATLAB subroutine, or globally. The SIMULINK functions can be accessed from the MATLAB workspace and the graphical SIMULINK environment. The SIMULINK systems themselves can be entered as block-diagrams (accessible from the graphical environment) or they can be programmed as M or MEX-file.



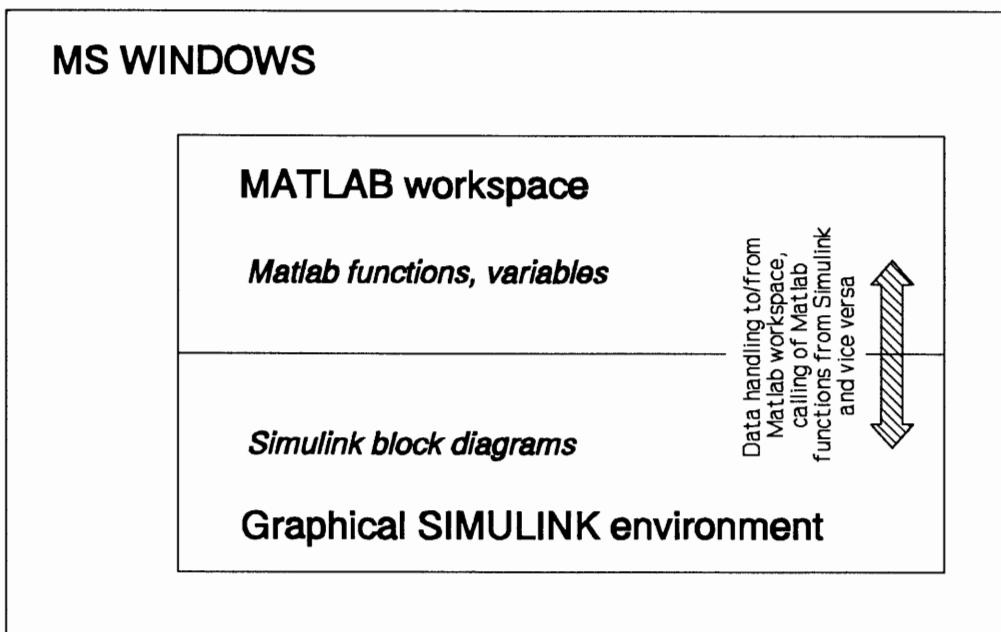


Figure 3-1. Functional relations between MATLAB and SIMULINK.

3.3 Advantages and disadvantages of SIMULINK, compared with other simulation environments.

Until a year ago, all models for simulation of nonlinear systems, used in the Disciplinary Group for Stability and Control were programmed in languages like Fortran, C/C++, and ADA. Although MATLAB was used for control system design and analysis, as well as system identification purposes, the more demanding computing tasks were all performed on the ENCORE minicomputer. For some purposes, PSIE and MatrixX were also used.

Due to the increase in 'desktop computing power', it is nowadays possible to do number-crunching on powerful PCs and workstations. Different phases in the system analysis and design can be integrated, using the combination of MATLAB and SIMULINK. The graphical system representation of SIMULINK quite naturally leads to modular, block-oriented simulation models. This is particularly useful for the research environment of a university, because it creates the possibility of linking different models, developed by several students for their graduation research to each other with a minimum of programming efforts.

The MATLAB/SIMULINK combination has several other advantages:

- MATLAB doesn't require size declarations or global definitions of variables.
 - MATLAB contains many built-in functions for matrix manipulation; it is not necessary to define the size of the matrices, used in MATLAB.
 - There are many MATLAB toolboxes with specialized functions for system analysis, control system design, parameter estimation, etc.

- MATLAB contains several functions for creating graphs.
- In the Disciplinary Group for Stability and Control, MATLAB is already used frequently, so a lot of experience with this programming language has already been gained.

In addition,

- The SIMULINK environment is perfectly suited for modular system design.
- It is possible to implement systems in SIMULINK as *black boxes*; in such case, the user only has to be concerned about the right inputs to the subsystem, in order to get the (hopefully right) output without having to know all details about the implementation of the subsystem.
- SIMULINK provides direct access to all variables during simulation.
- SIMULINK minimizes syntax-errors, caused by typing errors, saving a lot of debugging time.
- SIMULINK contains six numerical integration methods.
- Since SIMULINK operates under WINDOWS, it is possible to access other applications while working on a SIMULINK system. This saves a lot of time.
- Theoretically, it must be possible to implement SIMULINK systems in real-time simulation facilities, and to transfer control laws from the simulation model to the Flight Control Computer of the aircraft, with a minimum of programming efforts.

Some disadvantages of the MATLAB/SIMULINK combination are:

- MATLAB programs are considerably slower than Fortran or C programs. However, graphical SIMULINK systems are relatively fast, and it is always possible to implement very time-consuming algorithms as MEX-files.
- SIMULINK V1.2 isn't very stable under WINDOWS (although it already forms a significant improvement over SIMULAB V1.1). The user will frequently encounter 'Application Errors' during simulations and consequently, lose his/her data. It seems unlikely that such errors will arise in a UNIX environment.
- The user must be very careful in checking the connections of all signals. SIMULINK will not report errors if the user connects signals to wrong blocks, or if he/she forgets to draw some signal lines. This partly undoes the time-benefits from the reduced risk of making typing errors.

The benefits of SIMULINK and MATLAB over the conventional Fortran, ADA, or C/C++ environments outbalance the disadvantages mentioned above. This will be illustrated clearly by the flexibility and power of the 'Beaver' models from this report. The SIMULINK environment is perfectly suited for future research in the Disciplinary Group for Stability and Control, aimed at the National Fly-by-wire Testbed and the new six degree-of-freedom flightsimulator 'BARESIM'.

3.4 System representation in SIMULINK.

Simulation programs such as SIMULINK can handle *nonlinear* dynamical systems, if these are available in nonlinear continuous or discrete *state-space format*. For continuous systems, the state derivatives $\dot{\mathbf{x}}$ must be written as explicit functions of the continuous states $\mathbf{x}(t)$ and inputs $\mathbf{u}(t)$ at time t . For dis-



crete-time systems, the discrete state update $\mathbf{x}^*(k+1)$ must be written as an explicit function of the discrete states $\mathbf{x}^*(k)$ and inputs $\mathbf{u}^*(k)$ at time $t = k \cdot T_s$, where T_s is the sample time. If $\mathbf{y}(t)$ and $\mathbf{y}^*(k)$ denote the continuous and discrete output vectors respectively, the following equations fully describe the system dynamics:

Continuous system:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{y}(t) &= \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{x}(0) &= \mathbf{x}_0\end{aligned}\tag{3-1}$$

Discrete system:

$$\begin{aligned}\mathbf{x}^*(k+1) &= \mathbf{f}^*(\mathbf{x}^*(k), \mathbf{u}^*(k), k) \\ \mathbf{y}^*(k) &= \mathbf{g}^*(\mathbf{x}^*(k), \mathbf{u}^*(k), k) \\ \mathbf{x}^*(0) &= \mathbf{x}_0^*\end{aligned}\tag{3-2}$$

To simulate systems with a mixture of continuous and discrete dynamics (*hybrid systems*), SIMULINK uses a Zero Order Hold filter at the output side of discrete blocks. It is also possible to use a number of discrete blocks with multiple sampling rates (*multi-rate systems*).

Dynamical systems in state space format can be implemented in SIMULINK in a number of ways. SIMULINK systems are called 'S-functions'. An 'S-function' may be written as an M-file, a MEX-file, or it may be entered in block-diagram format, using the graphical SIMULINK environment. In order to function correctly, the S-function must have a certain required input/output structure.

Consider the S-function '*sfun*' (implemented as either an M-file, a MEX-file, or a SIMULINK block diagram). If this function is called by the user from the MATLAB workspace with the command:

```
sys = sfun( t, x, u, flag )
```

where t is the time, x is the state vector at time t , and u is the input vector at time t , *sfun* must return certain required system parameters in the return variable *sys*, depending on the value of *flag*:

flag	sys	description
0	sizes	see below
1	xdot	time derivatives of continuous states ($d\mathbf{x}/dt$)
2	xnew	discrete states $\mathbf{x}(k+1)$
3	y	output variables \mathbf{y}
4	tnext	next time interval for updates of \mathbf{x} and \mathbf{y} (used for discrete systems only)
5	r	values of the root functions (singularities)

Two other options, added for reasons of efficiency, are:

flag	sys	description
-1	xdot	time derivatives of continuous states ($d\mathbf{x}/dt$)
-2	xnew	discrete states $\mathbf{x}(k+1)$

Negative values of *flag* indicate that \mathbf{x} , $\mathbf{x}(k+1)$, t , and \mathbf{u} are unchanged from the last call. The state vector \mathbf{x} is partitioned into continuous and discrete states; the first states are the continuous ones and the last states are discrete.

If the S-function is called with *flag* = 0, a number of system properties are determined. Empty matrices may be used for t , \mathbf{x} , and \mathbf{u} :

```
sizes = sfun( [ ], [ ], [ ], 0 )
```

returns the vector *sizes*, with:

<i>sizes</i> (1)	number of continuous states,
<i>sizes</i> (2)	number of discrete states,
<i>sizes</i> (3)	number of output variables,
<i>sizes</i> (4)	number of input signals,
<i>sizes</i> (5)	number of discontinuous roots in the system,
<i>sizes</i> (6)	equals 1 when direct feedthrough of the input signals occurs in the system ¹⁾ (used for systems within systems).

More details about S-functions can be found in the SIMULINK manual, ref.[4]. An M or MEX-file is a valid S-function if it satisfies these input/output requirements; graphical block-diagrams automatically conform to these I/O relations. All analytical functions of SIMULINK (i.e. integration, linearization, and trim routines) use this call-format.

Since all S-functions (including block-diagrams!) behave like MATLAB functions, it is possible to call S-functions from M-files or from the MATLAB command line. This makes it possible to write customized routines for simulation, linearization, or parameter estimation. An example is given in appendix E and section F.2.7 of appendix F, where a custom aircraft trim routine is treated. Using the S-function call-format from the command line can be useful for extracting information like initial conditions and the number of states and inputs from SIMULINK systems, and for validation of the input/output relations of S-functions, are written as M or MEX-files.

3.5 The analytical functions of SIMULINK.

SIMULINK adds three powerful new functions to MATLAB, for analyzing linear or nonlinear, discrete-time or continuous dynamical systems in S-function format. These analytical functions are:

1 - simulation of (nonlinear) dynamical systems,

¹⁾ Gain blocks, transfer function blocks where the numerator is of the same order as the denominator, zero-pole-gain blocks with as many poles as zeros, and state-space blocks with a non-zero D matrix all have direct feedthrough. An algebraic or implicit loop occurs when two or more blocks with direct feedthrough form a feedback loop. See also section D.4.



- 2 - steady-state trim of dynamical systems (find the equilibrium points),
- 3 - linearization of nonlinear dynamical systems.

Some theory behind the simulation of nonlinear dynamical systems can be found in appendix D. The three classes of analytical functions will now be discussed briefly.

3.5.1 The integration routines.

SIMULINK contains six numerical integration routines, which can be applied to integrate the system equations (3-1):

- 1 - **EULER**. This is a one step integration method, which simply multiplies the state derivatives with the stepsize. It must take much smaller step sizes than the other methods in order to achieve the same accuracy, so this method is not recommended for the majority of problems. It is contained in SIMULINK primarily for historical reasons.
- 2 - **LINSIM**. This method first divides the system in linear and nonlinear subsystems. The discrete dynamics can be solved in a straightforward way, which leaves only the nonlinear system dynamics to be solved numerically. The method works quite well for systems which are 'relatively linear'. The method works very well for systems with a mixture of very fast and slow dynamics ('stiff systems', see appendix D).
- 3 - **RK23**. This is a third order Runge-Kutta method, which uses a second order method for step size control. For the generation of one output point, three evaluations of the state functions are needed. *RK23* is a general purpose method, which works well for a large range of problems. The method is well suited for the simulation of systems with discontinuities.
- 4 - **RK45**. This is a fifth order Runge-Kutta method, which uses a fourth order method for controlling the step size. For the generation of one output point, six state function evaluations, unequally distributed between two output points, are used. The method is usually faster and more accurate than *RK23*, and it produces fewer output points.
- 5 - **ADAMS**. This predictor-corrector method uses a variable number of points for the generation of one output point. It works well for systems with smooth state trajectories.
- 6 - **GEAR**. This method is primarily designed for the simulation of stiff systems. This too is a predictor-corrector method.

All integrators except *EULER* will take steps back in time if the predicted error exceeds the user-defined limit for the relative error. If the time interval is larger than the specified minimum step size, it will automatically be shortened. The user can specify such integration options as relative error, minimum and maximum stepsize, and, of course, the integration method itself. See ref.[4] for more details. In SIMULINK, it is possible to start simulations from WINDOWS menus within the graphical user-environment. The user can also call the integrators directly from the MATLAB workspace.

3.5.2 The linearization routines.

SIMULINK also contains some linearization routines. The command LINMOD extracts a linear state space model of a nonlinear continuous system in S-function format, in a certain operation point. The resulting linear system is given in linear state-space format:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du}\end{aligned}\tag{3-3}$$

LINMOD obtains a linear model around an operation point with the state variables \mathbf{x} and the input \mathbf{u} set to zero. It is possible to specify other values for \mathbf{x} and \mathbf{u} in the operating point if necessary. LINMOD perturbs the states around an operating point to find the rate of change of \mathbf{x} and \mathbf{u} (Jacobians). The perturbation factor can be defined by the user for every individual state or input variable.

DLINMOD does the same for discrete, multirate, and hybrid systems. It returns a discrete state model. The user must define the sample time T_s at which to perform the linearization. By setting T_s to zero, a continuous model approximation can be found. DLINMOD will give linear state space models with equal time and frequency responses at the converted sample time T_s , if T_s satisfies certain requirements, see ref.[4].

In chapter 6, the application of the SIMULINK linearization routines to linearization of the aircraft model will be illustrated briefly, but this report doesn't contain a detailed treatment of linearization theory (covering such topics as how to select perturbation levels). The 'Beaver' autopilot, which will be treated in part II, was designed with the linear models from ref.[30], because at that moment, the current SIMULINK models were not available yet. See ref.[4] for more details about the linearization tools, included in SIMULINK. More information about linearization of nonlinear aircraft models can be found in ref.[29].

3.5.3 The trim routines.

The last class of SIMULINK tools are some trim routines, which can be used to find equilibrium points of nonlinear dynamical systems. The command TRIM determines stationary values for the states, satisfying the input, output, and state equations of the system. TRIM searches for values of the state and input vectors for which $d\mathbf{x}/dt$ equals zero. According to ref.[4] there should be more versions of TRIM, which use different optimization routines, of which some are contained in the MATLAB OPTIMIZATION TOOLBOX, but only one version of TRIM could be found when using SIMULINK! It is possible to specify starting guesses for \mathbf{x} and \mathbf{u} , and to fix individual states variables (elements of \mathbf{x}), inputs (elements of \mathbf{u}), and outputs (elements of \mathbf{y}). See ref.[4] for more information about the SIMULINK TRIM commands.

In appendix E, a trim algorithm, which was designed especially for trimming nonlinear aircraft models is described, and applied to the aircraft model in SIMULINK. See also section F.2.7 of appendix F and ref.[29].



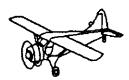
3.6 SIMULINK block diagrams.

SIMULINK contains libraries with graphical blocks, which can be connected to build block-diagrams. It is possible (and highly recommended) to subdivide large systems into a number of smaller subsystems, which can be arranged in a neat way and validated separately. The use of block-diagrams greatly facilitates computer representation of dynamical systems. In some cases, it is more convenient to program entire systems as M or MEX-files, but quite often, the graphical system representation is more flexible.

The most important graphical SIMULINK blocks are listed in appendix G. A list of all basic blocks can be found in the SIMULINK manual [4]. However, it is also very useful to examine the examples which are included in the SIMULINK package, because some useful undocumented tricks can be found in this way. Also, a large library of very useful but undocumented 'masked' blocks is available. See appendix G or ref.[4] for more information about subsystems and the *mask* function.

3.7 Conclusions.

The basic functions of the simulation program SIMULINK and the general structure of SIMULINK systems has been outlined in this chapter. SIMULINK contains tools for simulation, linearization, and steady-state trim of nonlinear dynamical systems. The systems can be either implemented as M or MEX-files, or entered as block-diagrams. The combination of MATLAB and SIMULINK yields a flexible and powerful environment for system research, which will be applied later in this report to the simulation of the nonlinear 'Beaver' model.



Chapter 4. Implementation of the nonlinear 'Beaver' model in SIMULINK.

4.1 Introduction.

The models described in chapter 2 and appendices A to C can now be transferred into a SIMULINK system. There are different possible ways to do this. Ref.[25] described how the models can be programmed as a set of MATLAB functions. This yielded a structure which was quite flexible, but only for experienced users. In this chapter, the systems will be implemented as SIMULINK block-diagrams, which are much faster, more flexible, and more user-friendly than the models from ref.[25]. Adding new output equations, deleting unwanted relations, or changing submodels to implement other aircraft is now really straightforward. In this chapter, the general structure of the models will be outlined. Appendix F gives a detailed description of all blocks from the SIMULINK implementation. See chapter 6 for some examples which illustrate how to use the SIMULINK models in practice.

4.2 Modular structure of the aircraft models.

The dynamics of an aircraft can be written as a set of nonlinear state equations, which can be derived from basic Newton mechanics and some kinematic relations. In appendix A, this derivation is described in detail; chapter 2 gives a short summary of the resulting equations. It will now be shown how the dynamic model of the aircraft dynamics can be divided into a number of submodels, which, in turn, can be implemented as different blocks in SIMULINK.

There are twelve state equations: three equations for the *velocities* along the aircraft's body axes (in this report, these expressions have been replaced by equations for the angle of attack, sideslip angle, and true airspeed), three equations for the *rotational velocities* along the aircraft's body axes, three equations which define the *attitude* of the aircraft relatively to the earth, and finally, three equations which define the *position* of the aircraft relatively to the earth. Consequently, there are twelve state variables.

The time-derivatives of the state variables are nonlinear functions of the states themselves and of external forces and moments acting upon the aircraft. The most important contributions to the total forces and moments are caused by aerodynamic influences, the operation of the powerplant (including slipstream effects), the influence of gravity, and additional terms which must be taken into account in non-steady atmosphere. These contributions can be modelled separately and implemented as separate SIMULINK blocks.

The forces and moments are functions of the state variables and external inputs and/or disturbances, such as deflections of control surfaces, engine power settings, and components of wind and atmospheric turbulence. They also depend upon a number of airdata variables, which, in turn, are functions of some atmospheric properties, such as temperature, air density, and pressure. Besides the equations for the forces and moments themselves, all other equations can be implemented as separate SIMULINK function blocks, and the functions which are closely related can be *grouped* together.



Before examining the implementation of these models in SIMULINK, it is useful to consider the model structure used in the DUT-flightsimulator first. Figure 4-1 shows the sequence of calculations used in this simulator¹⁾, see ref.[19]. A number of subroutines (aircraft module, engine module, ...) have to be called to calculate the time derivatives, i.e., to evaluate the state equations. The time derivatives of the state variables are integrated numerically with integration routines which have been included centrally within the simulation program. Before starting the actual simulations, a numerical trim routine is used to calculate steady-state values of the state and input variables, valid for a flight-condition, which must be specified by the user (airspeed, altitude, flap setting, engine speed). Refs.[6] and [18] describe the structure of this simulation software in more detail.

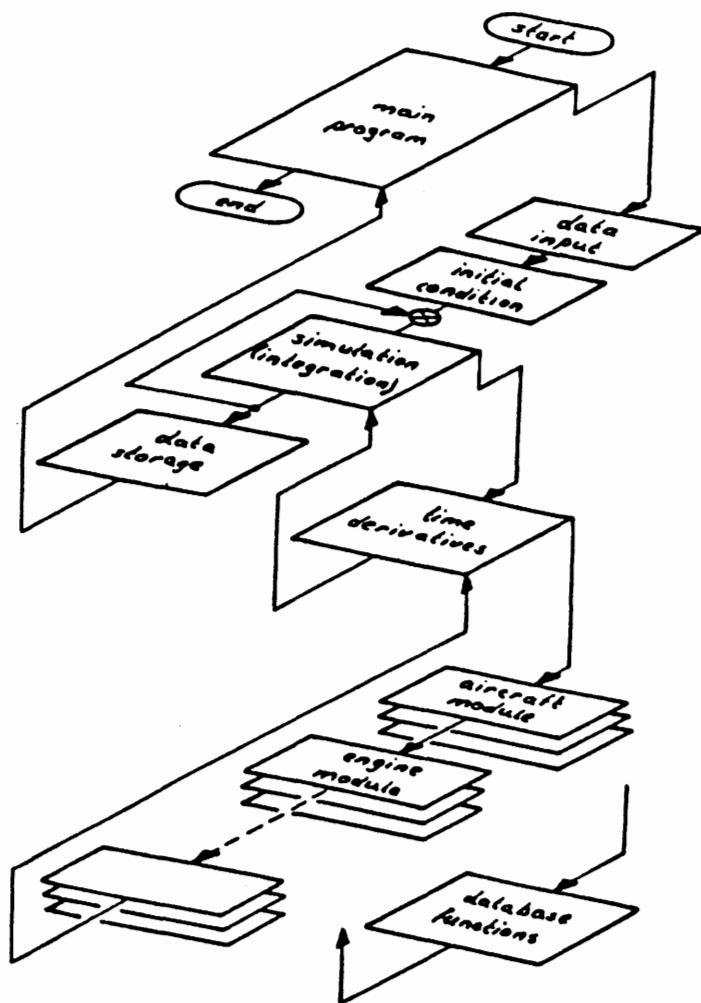


Figure 4-1. Modular structure of the DUT-flightsimulator software (sequence of computations).

¹⁾ The figure shows the calculation sequence, used for *off-line* simulation. It is necessary to include additional modules to determine the deflections of the actuators of the motion system, the indications of the cockpit instruments, and the visual system display for *on-line* piloted flightsimulation.

In ref.[25], it was described how the structure from figure 4-1 can be transferred to a SIMULINK system in M-file format. Contrary to the flightsimulator software, the numerical integrators were *not* included in the model from ref.[25], because SIMULINK itself takes care of the numerical integration. Here, the simulation model will be implemented as a SIMULINK block-diagram, which doesn't include numerical integration routines either, because it also uses the built-in SIMULINK integrators.

The structure of the graphical model still looks a lot like figure 4-1, but it is much more flexible than the Fortran or M-file implementation. The model was implemented by first creating blocks and *groups* containing closely related sets of equations, and then connecting these elements to eachother, to obtain the complete system. It is not necessary to edit any programs in order to add or delete equations, although it is still possible to include calls to Fortran, C, or Matlab subroutines if very complicated equations are to be implemented. The models from this report have been constructed entirely as block-diagrams!

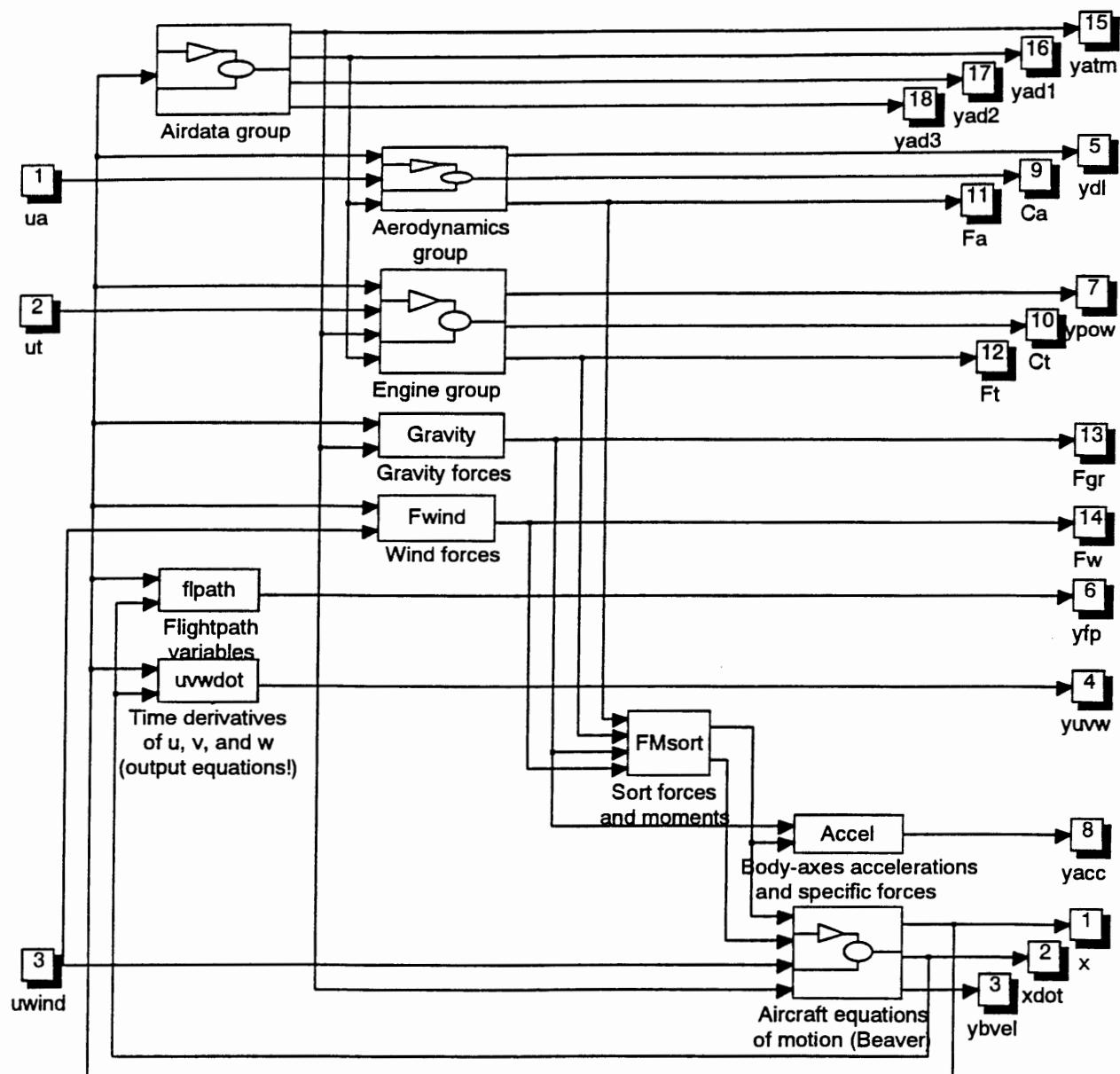


Figure 4-2. Structure of main level of 'Beaver' model in SIMULINK.



Figure 4-2 shows the global structure of the 'Beaver' models in SIMULINK block-diagram format. If some additional input and output ports are added to this system, it is possible to connect the nonlinear aircraft model to other systems by means of a SIMULINK *S-function* block (see chapter 3). Figure 4-2 actually shows the second level of the graphical 'Beaver' model; the first level, which only consists of additional input and output ports and a data storage block, will be treated in detail in appendix F.

There are two block types: (i) 'basic' blocks with the blockname displayed in the block itself, and (ii) subsystem blocks, with the name of the subsystem displayed underneath the block. The internal structure of the first block type can't be accessed directly by the user; these blocks behave like black-boxes¹⁾. In figure 4-2, the blocks *Gravity*, *Fwind*, *Flpath*, *Accel*, *FMsort*, and *uvwdot* are of type 1. All subsystem blocks (type 2) have been built up from a number of blocks of type 1. The user can zoom in to such subsystems, until the deepest level is reached. In figure 4-2, the blocks *Airdata group*, *Aerodynamics group*, *Engine group*, and *Aircraft equations of motion* are of type 2. Figures 4-3 to 4-6 show the structure of these submodels. The internal structure of the block *State derivatives* from figure 4-6, which forms another subsystem to which the user can zoom in further, is shown in figure 4-7.

The blocks *Atmosph*, *Airdata1*, *Airdata2*, *Airdata3*, *Dimless*, *Aeromod (Beaver)*, *FMdims*, *Power (Beaver)*, *Engmod (Beaver)*, *xdotcorr (Beaver)*, *Helpfcn*, *Vabdot*, *pqrdat*, *Eulerdot*, *uvw*, and *xyHdot* are blocks of type 1. The internal structure of all blocks of type 1 will be discussed in detail in appendix F. The *Integrator* and *Gain* blocks are standard SIMULINK blocks, see appendix E. The *Gain* block is used to fix certain state variables, for instance to exclude symmetrical or asymmetrical motions, to fix the airspeed arbitrarily, etc.

The following input and output vectors are used (the acronyms, used in the programs, and block-diagrams are shown between brackets):

Inputvectors:

\mathbf{u}_a	external inputs for the aerodynamic model (ua)
\mathbf{u}_t	external inputs for the engine model (ut)
\mathbf{u}_{wind}	wind velocities and time derivatives of wind velocities, necessary to compute wind forces and the position of the aircraft relatively to the earth (uwind)

Outputvectors:

\mathbf{y}_{atm}	atmosphere (-related) variables (yatm)
$\mathbf{y}_{ad\ 1}$	airdata (-related) variables group 1 (yad1)
$\mathbf{y}_{ad\ 2}$	airdata (-related) variables group 2 (yad2)
$\mathbf{y}_{ad\ 3}$	airdata (-related) variables group 3 (yad3)
\mathbf{y}_{dl}	dimensionless body-axes rotational velocities (ydl)

¹⁾ The user only needs to know the definitions of the input and output vectors for these blocks. Since the internal structure has been hidden from the user, it is possible to use the blocks in the same way as standard SIMULINK blocks. This was achieved by applying the 'mask'-function of SIMULINK, see appendix E. The internal structure of the blocks will be treated in detail in appendix F.

y_{pow}	engine power and dimensionless pressure increase across the propeller (y_{pow})
C_a	aerodynamic force and moment coefficients (C_a)
C_t	engine force and moment coefficients (C_t)
F_a	aerodynamic forces and moments (F_a)
F_t	engine forces and moments (F_t)
F_{gr}	gravity forces (F_{gr})
F_w	forces due to nonsteady atmosphere (F_w)
y_{fp}	flightpath variables (y_{fp})
y_{acc}	accelerations and specific forces (y_{acc})
y_{bvel}	body-axes velocities relatively to the surrounding atmosphere (y_{bvel})

Other vectors:

- \mathbf{x} state vector
 $\dot{\mathbf{x}}$ time derivative of the state vector (x_{dot})

See appendix F for detailed definitions.

The overall structure of the aircraft model is clearly visible in figure 4-2: the state derivatives are functions of external forces and moments, which have been divided in different categories. Blocks to calculate some atmosphere and airdata variables have been included to compute the forces and moments, and a number of additional airdata variables, flightpath variables, and accelerations have been added as additional output variables.

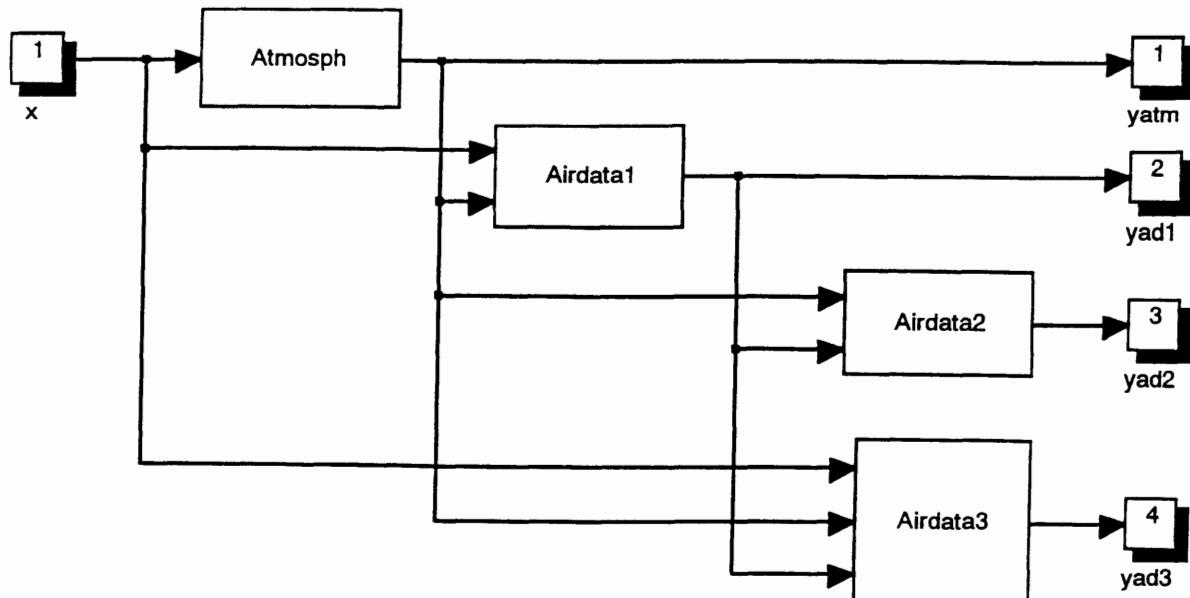


Figure 4-3. Structure of the block 'Airdata group'.



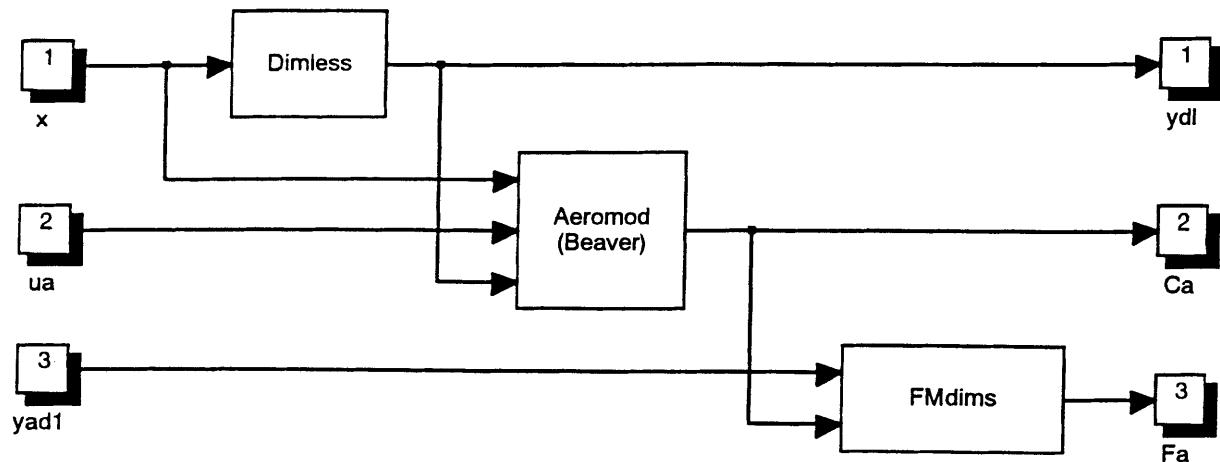


Figure 4-4. Structure of the block 'Aerodynamics group'.

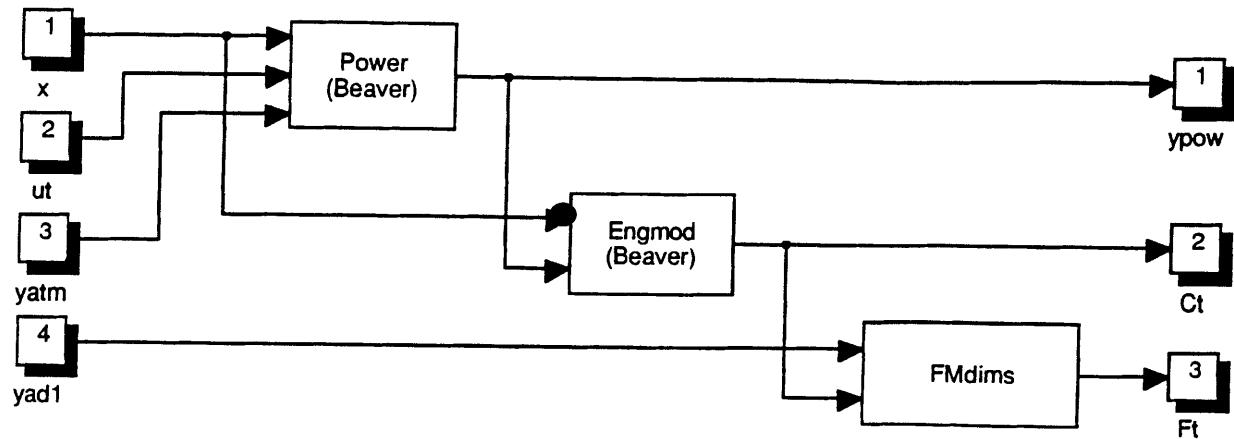


Figure 4-5. Structure of the block 'Engine group'.

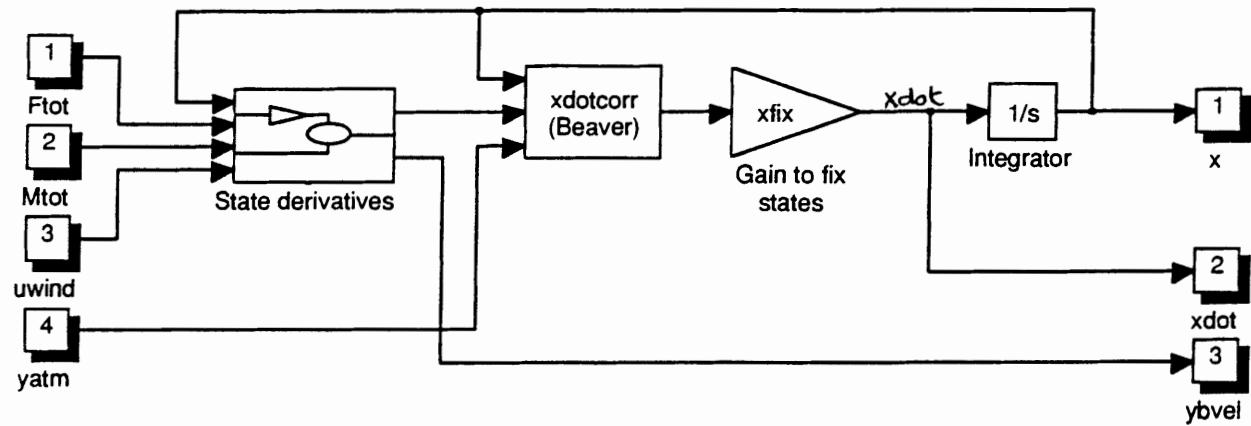


Figure 4-6. Structure of the block '*Aircraft equations of motion*'.

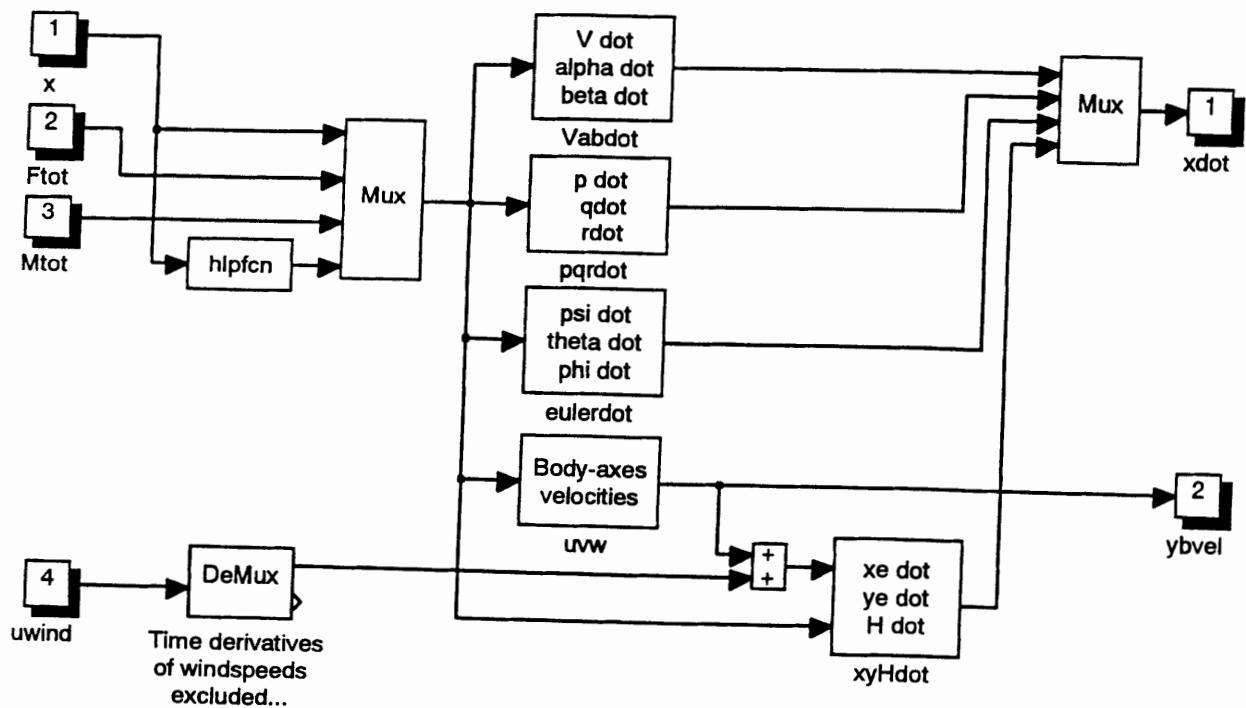


Figure 4-7. Structure of the block '*State derivatives*'.



If a part of the aircraft model needs to be changed, most other blocks can remain the same. For instance, removing all 'additional outputs' from this block-diagram can be done within only a few seconds. Adding other output-variables takes some more time, since a new output-function block must be constructed, but implementing such blocks within this structure shouldn't be too difficult if the definitions of all input and outputvectors are known. Replacing the engine forces and moments model and the aerodynamic model of the 'Beaver' for similar models of other aircraft is also quite simple.

The *parameters* of the different submodels, such as coefficients for the aerodynamic and engine forces and moments models, have to be defined within the MATLAB workspace, using the M-files MODBUILD and LOADER. See appendix F for more details.

It is necessary to define the initial values of the state and input variables in the vectors x_{inco} ($=\mathbf{x}_0$), $ua0$ ($=\mathbf{u}_a(0)$), and $ut0$ ($=\mathbf{u}_t(0)$) in the MATLAB workspace. Normally, states and inputs which yield a certain user-specified steady-state flight condition are used as initial conditions. They need to be calculated before the actual simulation is started, using a numerical *aircraft trim* routine, similarly to the DUT-flightsimulator calculating sequence shown in figure 4-1.

In appendix E, it will be shown how the steady-state values of the input and state variables can be found numerically. The resulting aircraft trim routine is an M-file called ACTRIM, which will be treated in section F.2.7 of appendix F. The user can save trimmed flight conditions to MAT-files, so it will not be necessary to recompute the steady-state condition every time a simulation is started.

In order to further integrate linear and nonlinear analysis, it is useful to linearize the aircraft models in several working points. Linearized aircraft models play a very important role in the AACCS design and analysis, because the whole linear system theory can be applied to such models. In this report, it will be illustrated that the standard SIMULINK linearization routine *LINMOD* can be used for this purpose. The working point is again defined by the values of \mathbf{x} and \mathbf{u} which yield a steady-state flight condition.

The combination of the nonlinear aircraft model in SIMULINK, the aircraft trim routine, and the linearization tools from SIMULINK makes it possible to do the whole linear and nonlinear off-line (control-) system analysis in the same working environment. In this way, it is much easier to make the step from linear to nonlinear system analysis, which will encourage the users to do more experiments to analyze the systems, and at the same time reduce the risk of making errors.

4.3 Adding external disturbances, navigation and approach models, and control loops.

If some additional input and output ports are included in the system shown in figure 4-2, it is possible to implement the nonlinear aircraft model as a sub-system within larger simulation models. In this way, other SIMULINK models can be connected to the 'Beaver' model. Examples of such models are: blocks to generate external disturbances and navigation signals (which may be needed for some control loops), control structures, and models of actuators and sensors. To keep the file size within limits, and to ensure a clear, modular structure, it is recommended to implement each model as a separate S-function. Different

S-functions can be connected to each other in a separate block-diagram, and each (sub-) model can easily be updated without affecting the total simulation structure. The use of different S-functions is particularly recommended for large models, which contain many levels. In the previous section, it was shown that the model of the aircraft dynamics itself is indeed very complicated.

In this report, separate libraries with models of wind, atmospheric turbulence, ILS signals, and VOR signals will be created. All models will be described in detail in appendix F. Similarly to the aircraft model, the equations of the other models will be implemented as SIMULINK *Function* blocks, which will be grouped together in 'basic' blocks or subsystems. Figure 4-8 shows how the models of the aircraft dynamics, disturbances, navigation signals, and control loops can be connected together in a modular way. In part II of this report, a similar structure, containing the control laws of the 'Beaver' autopilot, the model of the 'Beaver' dynamics, Flight Control System models, and external disturbances, will be treated in more detail.

Contrary to the nonlinear aircraft model of figure 4-2, control laws are often based upon *deviations* from nominal values of the control variables instead of the actual values of these signals. Thus, control loops use the difference between the outputvector from the aircraft model and the initial (= trimmed) value of this outputvector, $\mathbf{y} - \mathbf{y}_0$, instead of \mathbf{y} itself to compute the control signals $\Delta\mathbf{u}$. If the control signals are fed back to the aircraft model, it is necessary to add the initial values of the input signals \mathbf{u}_0 to $\Delta\mathbf{u}$. This is not necessary if a linear *small-perturbations* model of the aircraft is used instead of the complete nonlinear aircraft model. In that case, the whole structure presented in section 4.2 can be replaced by one linear state-space block only! The use of trimmed values of the states and inputs for linear and nonlinear aircraft models will be illustrated in chapter 6.

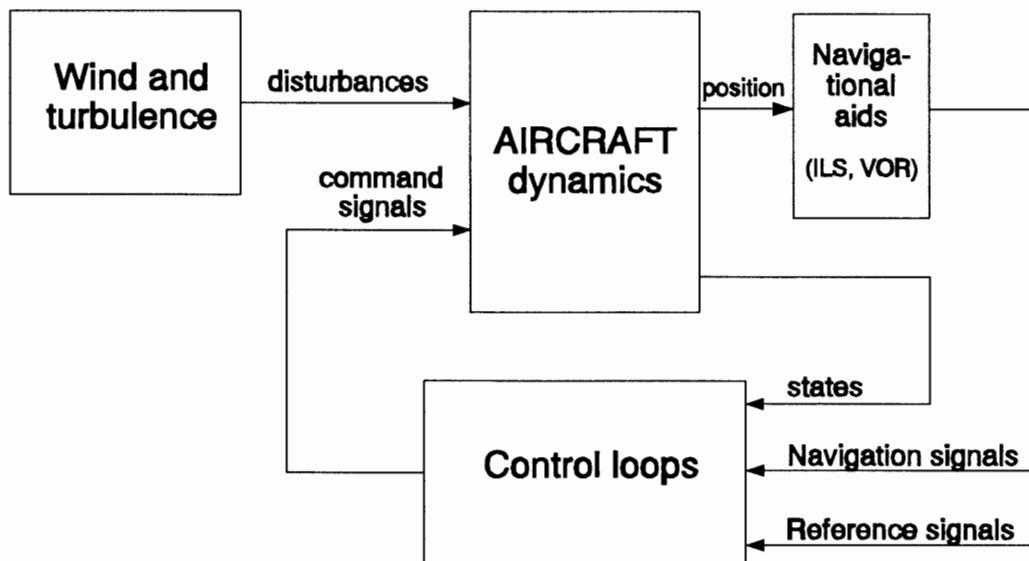


Figure 4-8. Model structure for simulation of an aircraft, equipped with an automatic controller, and affected by external disturbances.



Because of the modular structure, it is very easy to include other types of external disturbances, (e.g. wing icing, walking passengers, etc.), navigational signals (e.g. GPS signals), or additional contributions to the forces and moments (e.g. wing flexibility) into the simulation model. Moreover, the whole collection of MATLAB toolboxes can be accessed, so the model structure can also be used for other applications, such as parameter estimation, using the SYSTEM IDENTIFICATION TOOLBOX.

4.4 Conclusions.

The model of the aircraft dynamics can be divided into a number of submodels. Using the block-diagram representation of SIMULINK, a very flexible model structure has been created. Some models of external disturbances and navigational signals have been added to the simulation structure because they are often used for the assessment of the flying qualities of aircraft and the performance of automatic controllers. Models of other external disturbances or navigational aids can be included in a straightforward way. The block-oriented environment makes it easy to evaluate different solutions to typical control problems in a short time, without many programming efforts. Moreover, the aircraft trim and linearization routines have further integrated nonlinear and linear control system design and analysis.



Chapter 5. Assessment of the SIMULINK implementation of the 'Beaver' model.

5.1 Introduction

In this chapter, some open-loop results, obtained with the SIMULINK models of the 'Beaver' will be analyzed. First, some basic properties of the aerodynamic and engine models will be shown. Section 5.3 gives some linear and nonlinear open-loop responses, which should match similar results from refs.[6] and [18]. In section 5.4, open-loop turbulence responses will be shown, and finally, some results of basic open-loop analysis of the linearized 'Beaver' model will be given. Chapter 6 shows how the results from this chapter were created.

5.2 Results from the aerodynamic and engine models.

Of course, the primary contributions to the aerodynamic forces and moments are functions of the angle of attack α and the sideslip angle β . In ref.[30], graphs in which the aerodynamic forces and moments have been plotted as a

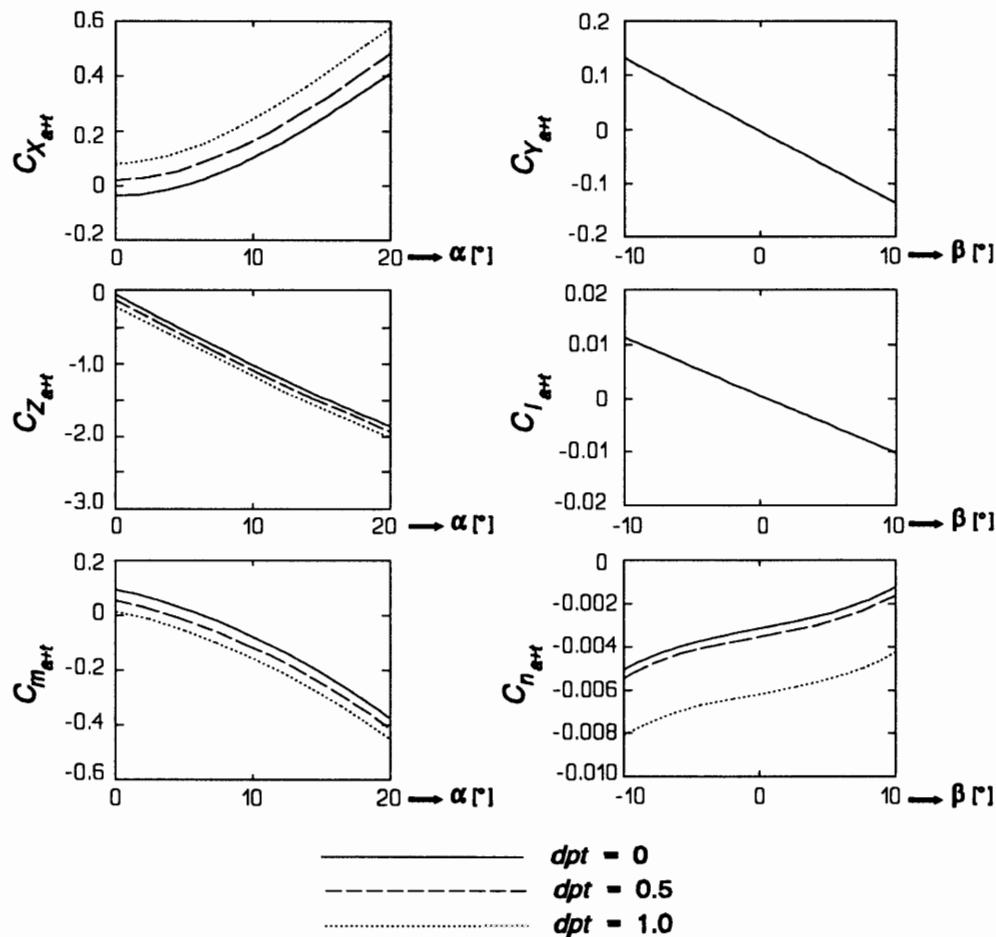


Figure 5-1. Force and moment coefficients of the nonlinear 'Beaver' model as functions of α and β . Compare with figures 4 and 5 of ref.[30].



function of α and β have been given. The engine forces and moments depend largely upon the dimensionless pressure increase across the propeller, dpt . In the plots of ref.[30], this influence has been shown too. Both results from the Fortran implementation of the aerodynamic and engine forces & moments models, which form a part of the DUT-flightsimulator software, and results from windtunnel experiments have been given in ref.[30].

Figure 5-1 shows the same graphs, calculated with the SIMULINK version of these models. The symmetric force and moment coefficients $C_{X_a} + C_{X_t}$, $C_{Z_a} + C_{Z_t}$, and $C_{m_a} + C_{m_t}$, have been plotted against α and the asymmetric force and moment coefficients $C_{Y_a} + C_{Y_t}$, $C_{l_a} + C_{l_t}$, and $C_{n_a} + C_{n_t}$, have been plotted against β for three different values of dpt . These graphs differ from the results obtained with the DUT-flightsimulator models, because in ref.[30], the angle of attack and sideslip angle have been incorrectly converted from radians to degrees before the plots were made¹⁾. Figure 5-1 shows the correct graphs, which also better match the results from windtunnel measurements (ref.[30])!

5.3 Assessment of open-loop responses computed with the new SIMULINK models.

In refs.[6] and [18], some open-loop responses of the 'Beaver', calculated with the DUT-flightsimulator software, were presented. Figures 5-2 to 5-4 show the same responses, created with the nonlinear SIMULINK 'Beaver' model and a version of this model which was linearized from within SIMULINK. The trim values of the input and state variables were computed with the aircraft trim routine *ACTRIM*. Table 5-1 lists the initial conditions for steady level flight, $V_0 = 45$ m/s, $H_0 = 6000$ ft. This trim routine, which can be applied to the SIMULINK implementation of the 'Beaver' model will be described in appendix E and section F.2.7 of appendix F.

Figures 5-5 to 5-7 show open-loop responses to a flap deflection of 3 degrees within 3 seconds, a change in engine speed of 200 rotations per minute within 4 seconds, and a change in manifold pressure of 2 inch Hg within 2 seconds, respectively. Ref.[18] gives the same responses, calculated with the flightsimulator software. Although the scaling of the plots differs from refs.[6] and [18], to show more details of the aircraft responses, the results are exactly the same. This means that the most important blocks in the 'Beaver' model, i.e., the aerodynamic model, atmosphere model, basic airdata equations, engine forces and moments model, and the state equations themselves have been implemented correctly. The new SIMULINK model gives the same results as the M-file implementation from ref.[25], but compared with that model, all open-loop calculations are approximately 3 to 5 times as fast (depending upon the integration method and required stop-time of the simulations).

The linear open-loop responses from figures 5-2 to 5-7, have been created by applying the SIMULINK linearization tool *LINMOD* to the nonlinear 'Beaver' model in SIMULINK. The results match the nonlinear responses remarkably well, even though no efforts have been made to find out which linearization

¹⁾ Indeed, the software, which was used to create the figures in ref.[30] converted the results from radians to degrees by multiplying with $180/2\pi$ in stead of $180/\pi$! This error was made when preparing the plots from ref.[30], so the flightsimulator models themselves are correct, giving the same results as figure 5-1!

options (perturbation levels, linearization algorithm, etc.) give the best performance¹⁾. The largest differences can be found in the lateral-longitudinal cross-coupling effects from figures 5-3 and 5-4. The longitudinal-lateral cross-coupling is quite good, especially in figure 5-2. Although there is often some phase shift of the phugoid in linear responses to *lateral* inputs (compared with the nonlinear results), the lateral responses to lateral inputs match perfectly, as do longitudinal responses to longitudinal inputs.

Note that the time-trajectory of x_e is not accurate. This is due to the small-perturbation character of the linear responses: integrating the *deviation* of the nominal airspeed yields a result which differs a lot from the integral of the actual airspeed! Since the time-trajectory of y_e is also not very accurate, the linearized x_e and y_e -equations shouldn't be used for navigation purposes. For non-turning flights, x_e can be approximated by: $x_e(t) = y(10) + V_0 \cdot t$, where $y(10)$ is the time-trajectory of the tenth output from the state-space model, Δx_e (use $x_e(t) = y(10) + (V_0 \cos \alpha_0 \sin \beta_0) \cdot t$ if the equilibrium values of α and/or β differ a lot from zero).

The initial condition, which was computed with the trim routine *ACTRIM*, (see appendix E and section F.2.7 of appendix F) yielded the same results as ref.[30]. This gives us an additional validation of the SIMULINK 'Beaver' model. And since the SIMULINK and Fortran versions of the same model have been developed fully independently, this check also raises the confidence in the Fortran programs. It is highly unlikely that the same mistakes are made twice!

Figures 5-2 to 5-7 clearly show the characteristic motions of the 'Beaver'. In figure 5-2, the short-period mode is visible in the α , q , and θ plots and also in the β , p , r , and φ plots, due to longitudinal-lateral cross-coupling. For the 'Beaver', the *lateral* responses to the *longitudinal* block-shaped elevator deflection $\Delta \delta_e$ are quite large. One should realize this when using simplified 'Beaver' models, which assume that the longitudinal and lateral dynamics are independent of each other. The phugoid, an oscillation with a period of about 23 seconds, is clearly visible in the responses to the elevator deflection, shown in figure 5-2. It is also clearly present in the responses to aileron and rudder deflections, shown in figures 5-3 and 5-4, which again proves the significance of cross-coupling effects for the 'Beaver'.

The Dutch-roll and spiral modes can be distinguished easily in figures 5-3 and 5-4. The 'Beaver' behaves quite well in this respect, since the Dutch-roll mode is strongly damped and the spiral mode is (just) stable. The roll subsidence mode is hardly visible in these plots; only the p and φ -responses in figure 5-3 show some influence of this characteristic motion. The influence of the roll subsidence mode would probably have been more clearly in these plots if a step-shaped input to the ailerons would have been applied instead of a block-shaped deflection of the ailerons.

In figure 5-5, the response of the 'Beaver' to a flap deflection of 3 degrees within 3 seconds is shown. Again, the phugoid is visible in both longitudinal and lateral motion variables. The airspeed varies significantly after application of the flap deflection. The mean value of the airspeed drops slightly, and the

¹⁾ In this report, the linearization blocks and plots with responses of the linear aircraft model have been included to illustrate the principle of linearization in SIMULINK, not to find out which linearization method or options give the best results. However, the differences between linear and nonlinear responses are already quite small!



final value of the angle of attack is larger than without flap deflection. This figure again clearly proves the significance of the longitudinal-lateral cross-coupling for the 'Beaver': due to the flap deflection, the aircraft starts to roll with a roll angle of approximately 10 degrees, and the sideslip angle becomes more negative. The change in φ is due to the negative value of $C_{l_{\alpha \Delta p_t}}$ (both α and dpt increase after the flap deflection).

Figures 5-6 and 5-7 show the responses of the 'Beaver' to an increase in engine speed and manifold pressure, respectively. These two inputs yield almost identical responses, as can be expected, because in both cases the resulting increase in engine power P , and hence, the increase in manifold pressure Δp_t , causes the changes of the forces and moments. The mean airspeed grows somewhat, the angle of attack becomes smaller. The value of dpt increases due to the larger engine power, even though the airspeed increases too. Again, this yields a larger sideslip angle (more negative), and the aircraft starts to roll.

Variable:	Value at $t = 0$ sec:	
V	45	m/s
α	0.1444	rad
β	-0.0147	rad
p	0	rad/s
q	0	rad/s
r	0	rad/s
ψ	0	rad
θ	0.1444	rad
φ	0	rad
x_e	0	m
y_e	0	m
H	6000 ft = 1828.8 m	
δ_e	-0.0425	rad
δ_a	0.0091	rad
δ_r	-0.0460	rad
δ_f	0	rad
p_z	21.072	"Hg
n	1800	RPM

Table 5-1. Initial conditions for $V = 45$ m/s, $H = 6000$ ft, obtained with the trim routine ACTRIM (compare with ref.[30]).

The responses illustrate the most important characteristics of the motions of the 'Beaver'. In particular, the strong longitudinal-lateral cross-coupling, and the strong influence of the engine and slipstream effects, are clearly visible. As stated in ref.[18], the responses obtained with the nonlinear 'Beaver' model do resemble the actual motions of the aircraft quite well.

5.4 Open-loop responses to atmospheric turbulence.

Figures 5-8 to 5-11 show open-loop responses of the 'Beaver' to atmospheric turbulence. The turbulence has been modelled as white noise, passing through Dryden filters, with $L_g = 300$ m, and $\sigma_g = 1$ m/s. The coefficients of these filters are a function of the airspeed V , but for these plots, the values of the filter coefficients for $V = 45$ m/s were used. The initial altitude H_0 was equalled to 2000 ft instead of the 6000 ft used in section 5.3, but this doesn't really matter for the evaluation of the responses.

It is not possible to compare the results from figures 5-8 to 5-10 right away with similar responses presented in ref.[24], because there are some significant differences in the definitions of the variables. Firstly, ref.[24] uses linear aircraft models, which express the motions of the aircraft in terms of small perturbations from an equilibrium condition. Furthermore, in ref.[24] the contributions of the turbulence velocity components to the angle of attack, sideslip angle, and true airspeed are treated as separate state variables (called α_g , β_g , and u_g). Here these contributions are simply included in α , β , and V .

If we now compare figures 5-8 to 5-11 with responses given in ref.[24], we find some differences:

- The frequency of the r -response to vertical or longitudinal turbulence (figures 5-8 and 5-10) is larger than the frequency of the p -response, contrary to ref.[24], where the frequency of the p -response was always larger than the frequency of the r -response. However, the responses to lateral turbulence (figure 5-9) are consistent with the results from ref.[24].
- The influence of vertical turbulence to the φ -response (figure 5-10) and the amplitude of the high-frequency fluctuations in φ (figure 5-9) are both larger than in ref.[24].
- The ψ , φ , and r -responses to longitudinal, vertical, and lateral turbulence, clearly show the influence of the spiral mode, whereas this influence is not so obvious in the results from ref.[24].
- The frequency of the q -response to longitudinal and vertical turbulence is quite large, compared with ref.[24].

Most results, however, are consistent with ref.[24]. The longitudinal variables H and θ clearly show the influence of the phugoid, which is initiated by longitudinal, vertical, and lateral turbulence. In figures 5-8 to 5-11, as well as in ref.[24], the amplitude of the phugoid tends to grow in time. As can be expected, the phugoid is most strongly excited by vertical turbulence, the longitudinal motion variables vary most strongly for vertical or longitudinal turbulence, and the lateral motion variables vary most strongly for lateral turbulence.



Furthermore, the longitudinal responses to longitudinal or vertical turbulence, and the lateral responses to lateral turbulence are of the same order of magnitude as the corresponding results from ref.[24]. The remaining differences are largely caused by the cross-coupling of longitudinal and lateral motions of the aircraft, which has been neglected in ref.[24].

The responses of the 'Beaver' to atmospheric turbulence, calculated with the nonlinear aircraft model from appendix A and the Dryden turbulence model from appendix B, seem to be quite reasonable. For this reason, the current models can be used to analyze the behaviour of an AAC under external atmospheric disturbances. But remember that a large number of approximations have been made in the derivation of the mathematical models, so it might be necessary to use actual measurements of turbulence or more accurate models for some purposes.

5.5 Open-loop analysis of the linearized 'Beaver' model.

It is very easy to create a linear state-space aircraft model, based upon the nonlinear SIMULINK implementation of the aircraft model, by first trimming the system with the routine *ACTRIM* ('Aircraft trim', see chapter 6, appendix E, and section F.2.7) and then using the routine *ACLIN* ('Aircraft linearization', see chapter 6 and section F.2.7), which is based upon the standard SIMULINK linearization command. The resulting linear aircraft model can be used for control system design, but here, we will limit ourselves to the determination of some basic open-loop properties.

The flight-condition from table 5-1 will be used here to further analyze the 'Beaver' dynamics. This flight-condition has been defined before applying the linearization tool *ACLIN* to obtain linear models for the longitudinal and lateral aircraft dynamics. The resulting state-space matrices are given in table 5-2; table 5-3 shows some basic properties of this system. Apart from the cross-coupling, which was neglected to produce these results, it can be concluded that the results are in accordance with the time-responses of the system. The phugoid and the spiral mode are both marginally stable. Also notice the relatively large damping constant of the Dutch-roll!

$$\begin{aligned} A_{\text{symm}} &= \begin{bmatrix} -3.8930 \cdot 10^{-2} & 5.4535 \cdot 10^0 & -4.0758 \cdot 10^{-1} & -9.8000 \cdot 10^0 \\ -8.3760 \cdot 10^{-3} & -1.2854 \cdot 10^0 & 9.7640 \cdot 10^{-1} & -1.0893 \cdot 10^{-6} \\ 1.3905 \cdot 10^{-2} & -6.7369 \cdot 10^0 & -3.0292 \cdot 10^0 & 0 \\ 0 & 0 & 1.0000 \cdot 10^0 & 0 \end{bmatrix}; \quad B_{\text{symm}} = \begin{bmatrix} -6.0796 \cdot 10^{-1} & -1.6817 \cdot 10^0 \\ -9.2937 \cdot 10^{-2} & -3.6313 \cdot 10^{-1} \\ -1.0601 \cdot 10^1 & 2.2471 \cdot 10^0 \\ 0 & 0 \end{bmatrix} \\ A_{\text{asym}} &= \begin{bmatrix} -1.8068 \cdot 10^{-1} & 1.4002 \cdot 10^{-1} & -9.8159 \cdot 10^{-1} & 2.1682 \cdot 10^{-1} \\ -4.0528 \cdot 10^0 & -5.4022 \cdot 10^0 & 1.7965 \cdot 10^0 & 0 \\ 1.7227 \cdot 10^{-1} & -8.7058 \cdot 10^{-1} & -5.5189 \cdot 10^{-1} & 0 \\ 0 & 1.0000 \cdot 10^0 & 1.4540 \cdot 10^{-1} & 0 \end{bmatrix}; \quad B_{\text{asym}} = \begin{bmatrix} -6.9558 \cdot 10^{-3} & 4.5163 \cdot 10^{-2} \\ -7.2976 \cdot 10^0 & 3.9805 \cdot 10^{-1} \\ -1.9921 \cdot 10^{-1} & -2.6058 \cdot 10^0 \\ 0 & 0 \end{bmatrix} \end{aligned}$$

(Here: $x_{\text{symm}} = [V \ \alpha \ q \ \theta]^T$; $u_{\text{symm}} = [\delta_e \ \delta_r]^T$; $y_{\text{symm}} = x_{\text{symm}}$)

Table 5-2. State-space matrices of linear 'Beaver' model, derived from the nonlinear SIMULINK model with *ACLIN*.

	Mode	$\lambda = \xi \pm \eta i$	$P = \frac{2\pi}{\eta}$ [s]	$T_{1/2} = -\frac{0.663}{\xi}$ [s]	$\zeta = \frac{-\xi}{\sqrt{\xi^2 + \eta^2}}$	$\omega_n = \eta$ [rad/s]	$\omega_0 = \frac{\omega_n}{\sqrt{1 - \zeta^2}}$ [rad/s]
Symm	phugoid	-0.01614 ± 0.2631 <i>i</i>	23.8	42.94	0.0612	0.2631	0.2636
	short period	-2.1606 ± 2.4112 <i>i</i>	2.606	0.32	0.6673	2.4112	3.2374
Asymm	Dutch-roll	-0.4770 ± 0.9763 <i>i</i>	6.436	1.453	0.4390	0.9763	1.0866
	roll subsidence	-5.1351	-	0.135	-	-	-
	spiral	-0.04573	-	15.15	-	-	-

Table 5-3. Basic open-loop properties of the linear 'Beaver' model, derived from the nonlinear SIMULINK model with ACLIN.

Figure 5-12, which has been included at the end of this chapter, shows some Bode plots for the *longitudinal* open-loop system.¹⁾ The amplitude-ratios and phase-angles have all been determined for an input signal upon the elevator. The resulting plots have been compared with 'typical control-input Bode plots for conventional airplanes', given in ref.[23] (fig.5-2), and the results match remarkably well. Hence, the SIMULINK linearization tool, applied with *default* linearization options (perturbation level, etc., see ref.[4]) gives good results, both in time *and* in frequency domain.

5.6 Conclusions.

The SIMULINK models, presented in this report, yield the same results as the flightsimulator software, presented in refs.[6], [18], and [30], so the most critical parts of the models, i.e., the aerodynamic model, engine model, atmosphere and airdata model, and the state equations themselves, have been implemented correctly. The open-loop responses to atmospheric turbulence, given in this chapter, are not exactly equal to the results from ref.[24], but the aircraft model used here differs a lot from the small perturbation models in ref.[24]. For instance, the cross-coupling of longitudinal and lateral motion variables, which is really significant for the 'Beaver', has been neglected totally in ref.[24]. The turbulence models seem to be good enough for most AACs research purposes.

¹⁾ The definition of the phase-angle, used in the MATLAB command *BODE* has been maintained, to make sure that the user can reproduce figure 5-12 easily. However, often other definitions for the initial value of the phase-angles are used!



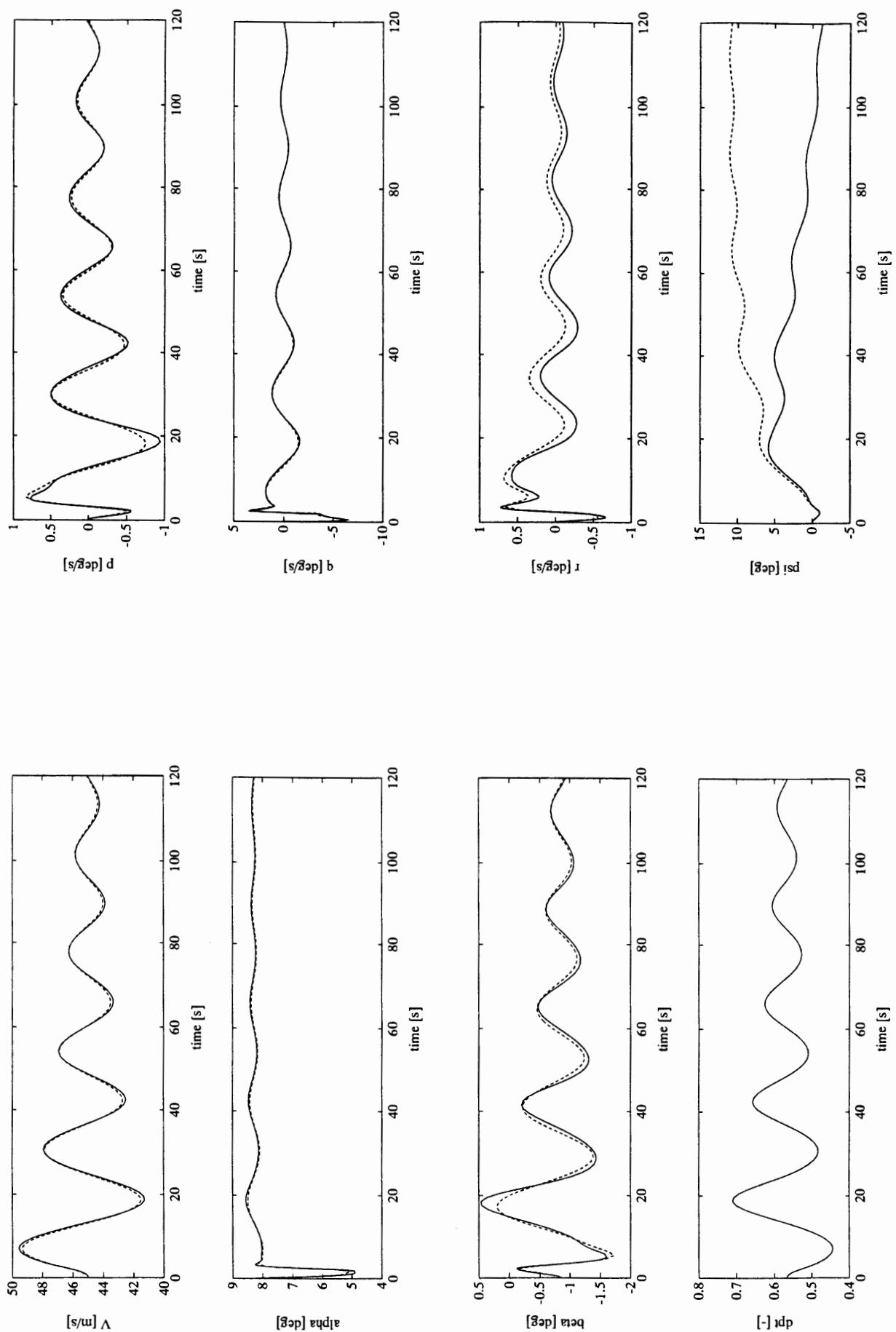
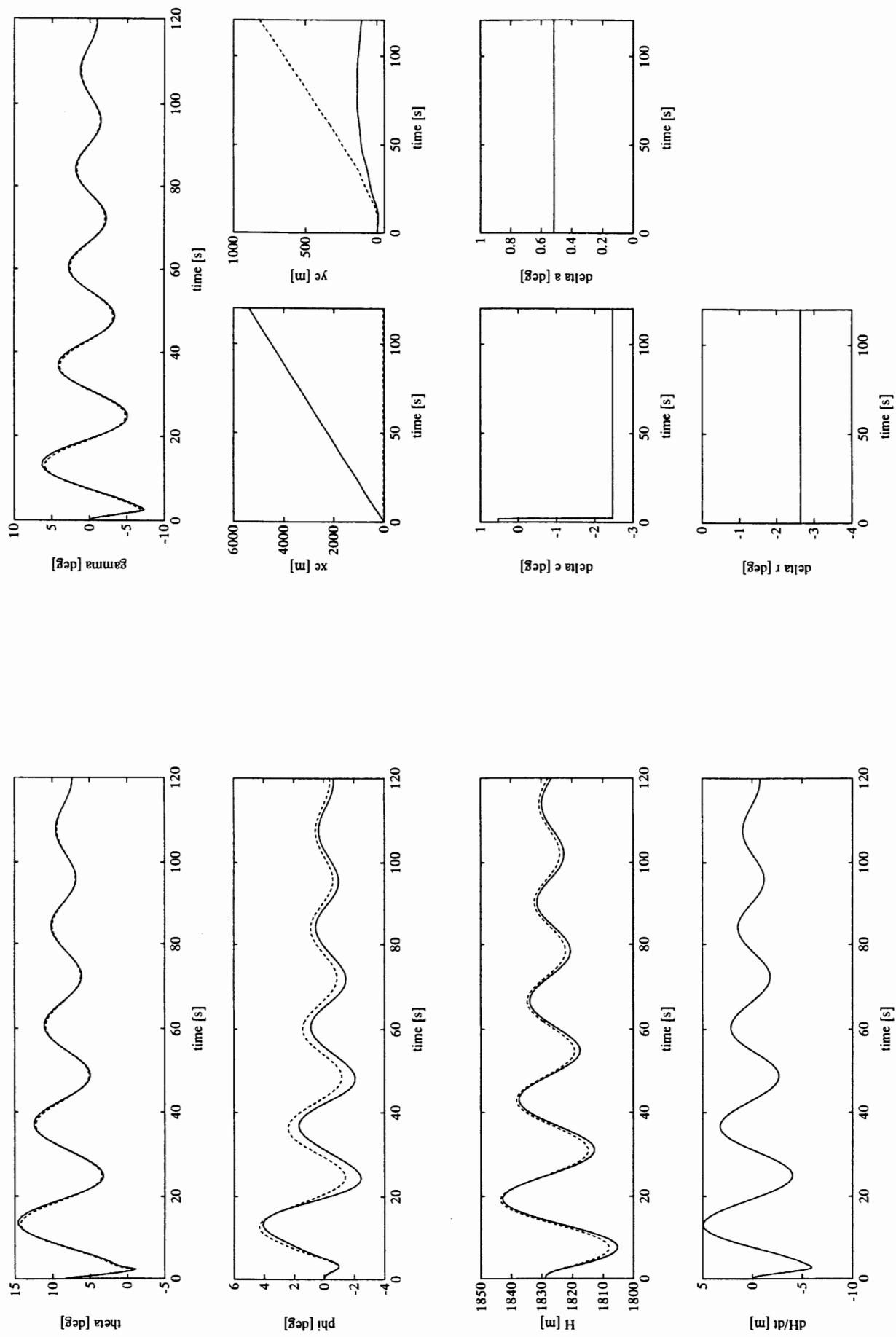


Figure 5-2. Open-loop responses of the 'Beaver' to a block-shaped elevator input $\Delta\delta_e = 3^\circ$ during 2 seconds. $V_0 = 45 \text{ m/s}$, $H_0 = 6000 \text{ ft}$.

**Figure 5-2 (continued).**

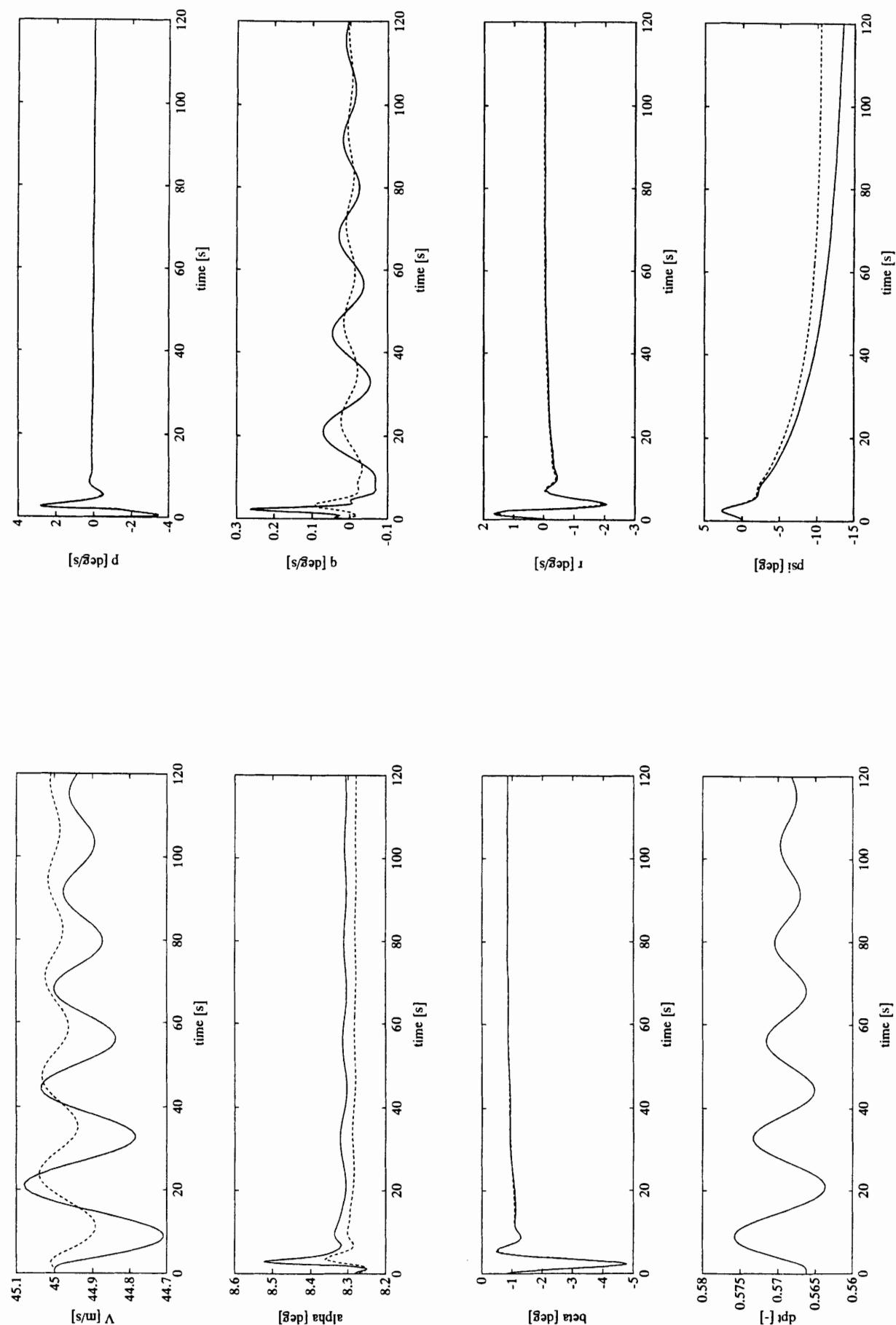
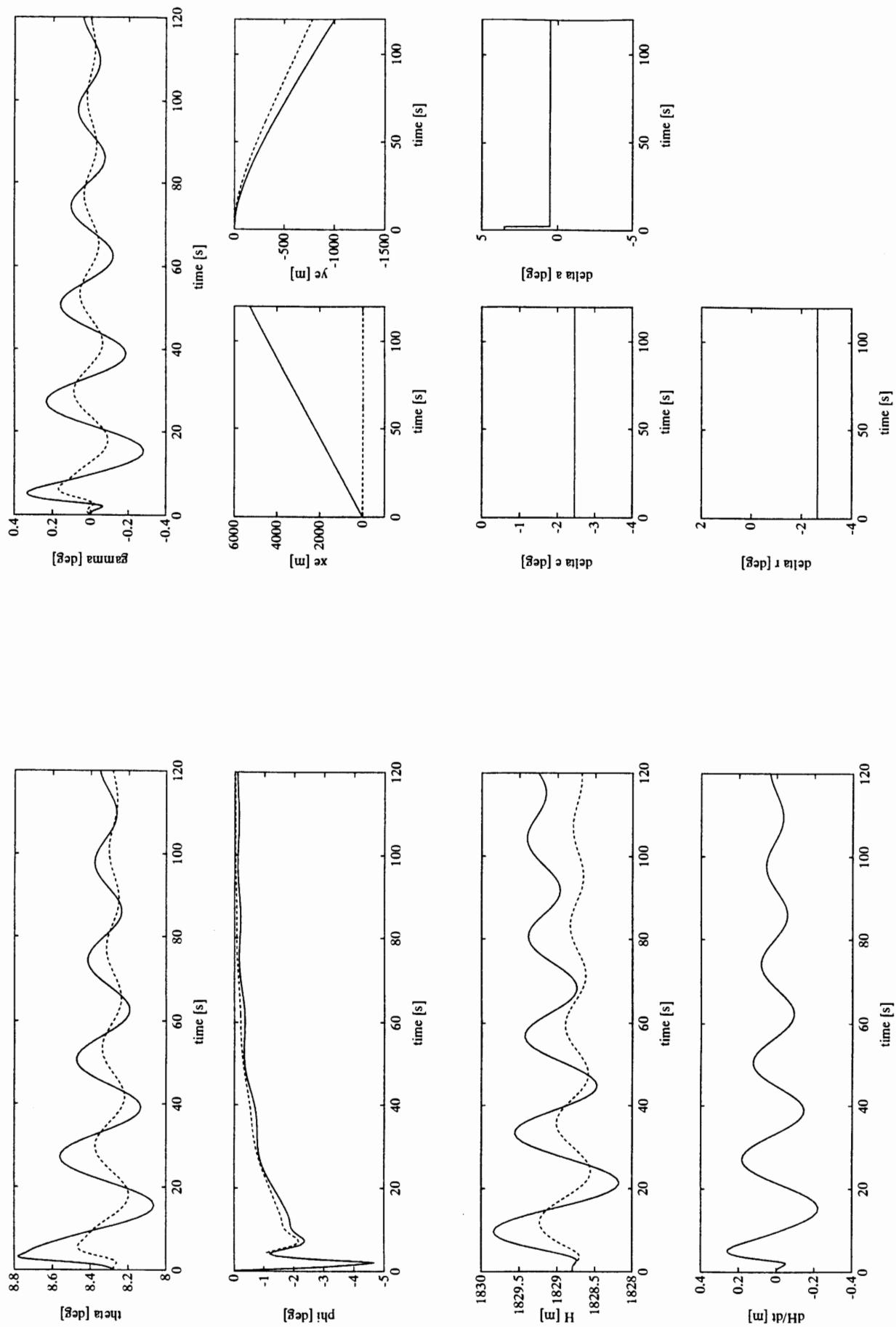


Figure 5-3. Open-loop responses of the 'Beaver' to a block-shaped aileron input $\Delta\delta_a = 3^\circ$ during 2 seconds. $V_0 = 45 \text{ m/s}$, $H_0 = 6000 \text{ ft}$.

**Figure 5-3 (continued).**

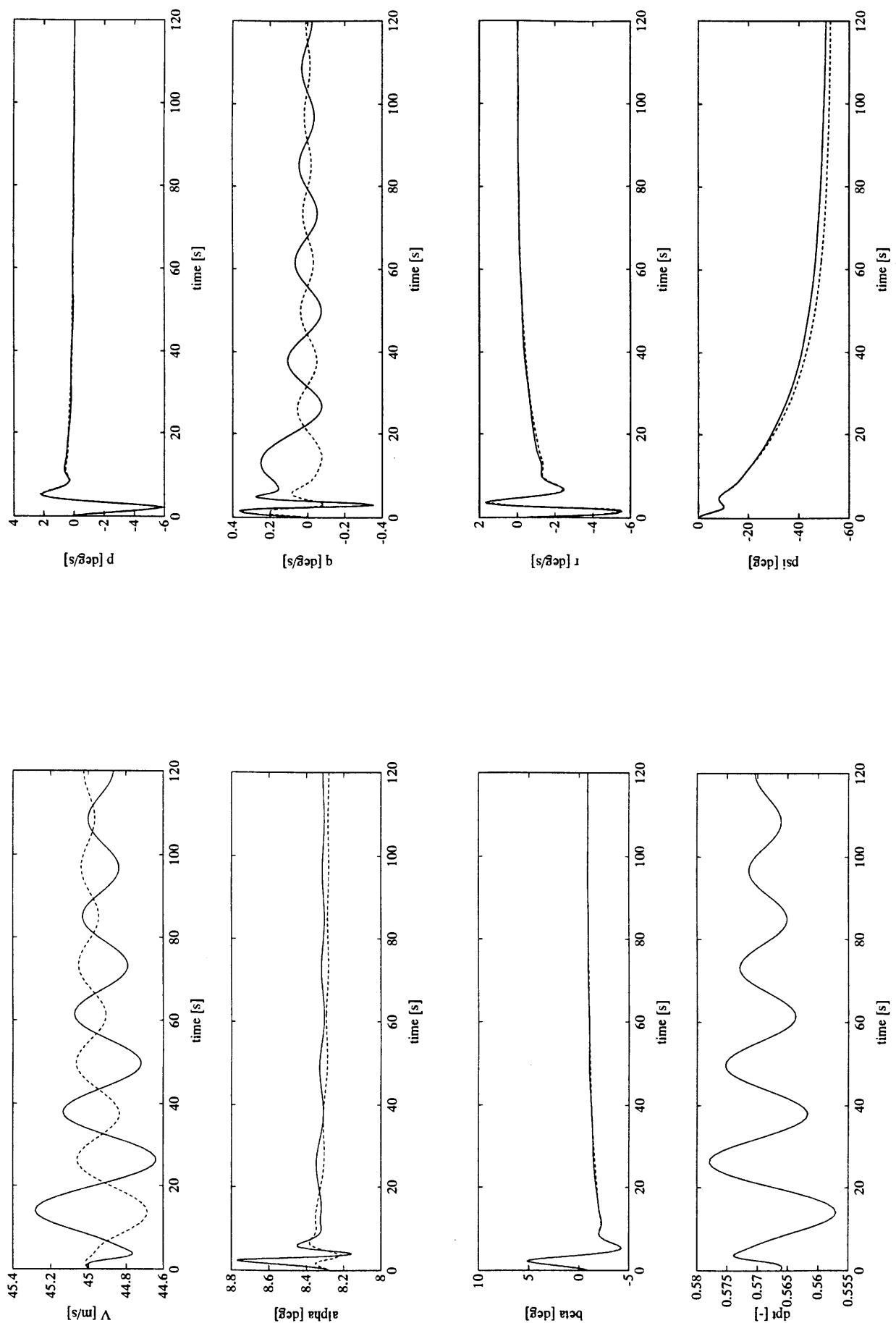
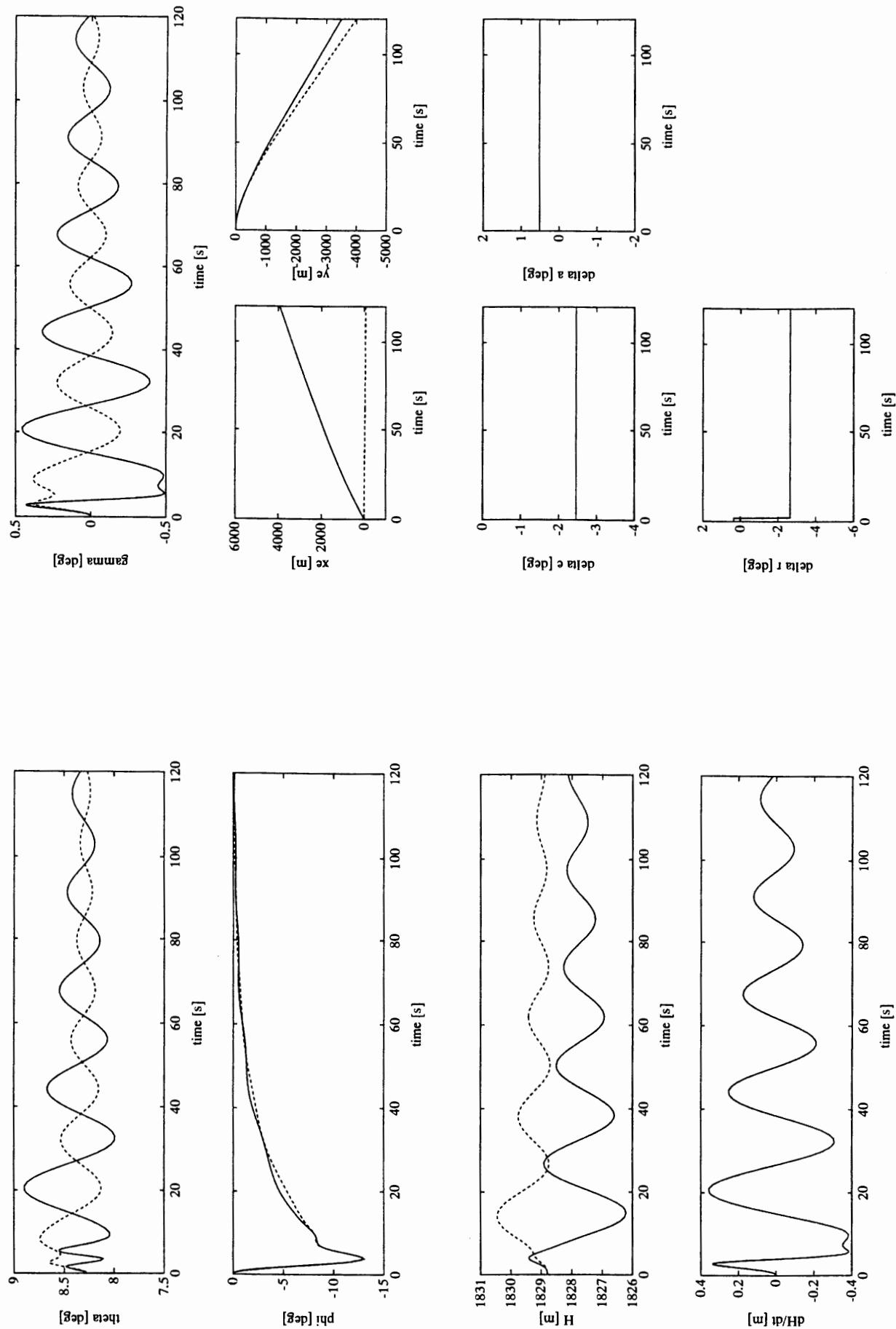


Figure 5-4. Open-loop responses of the 'Beaver' to a block-shaped rudder input $\Delta\delta_r = 3^\circ$ during 2 seconds. $V_0 = 45 \text{ m/s}$, $H_0 = 6000 \text{ ft}$.

**Figure 5-4 (continued).**

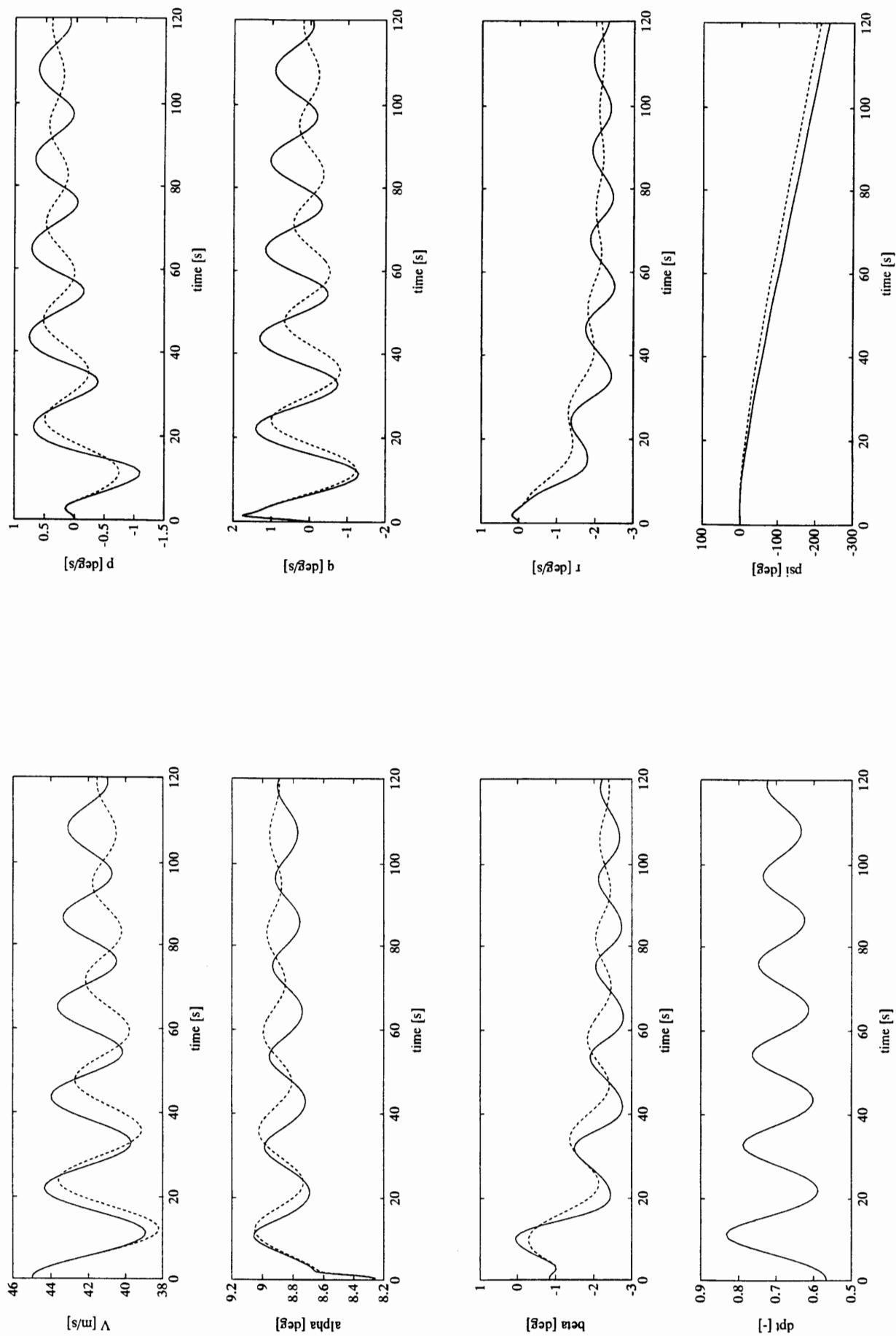


Figure 5-5. Open-loop responses of the 'Beaver' to a flap deflection
 $\Delta\delta_f = 3^\circ$ in 3 seconds. $V_0 = 45$ m/s, $H_0 = 6000$ ft.

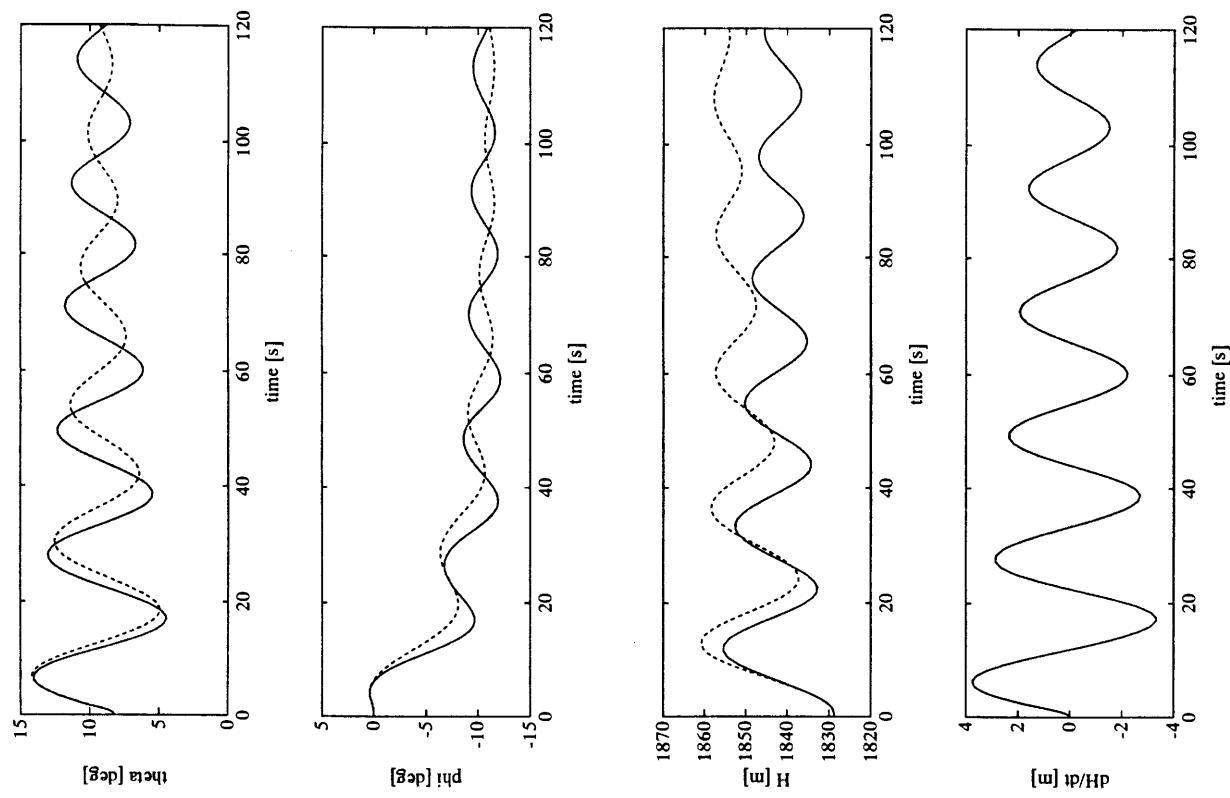
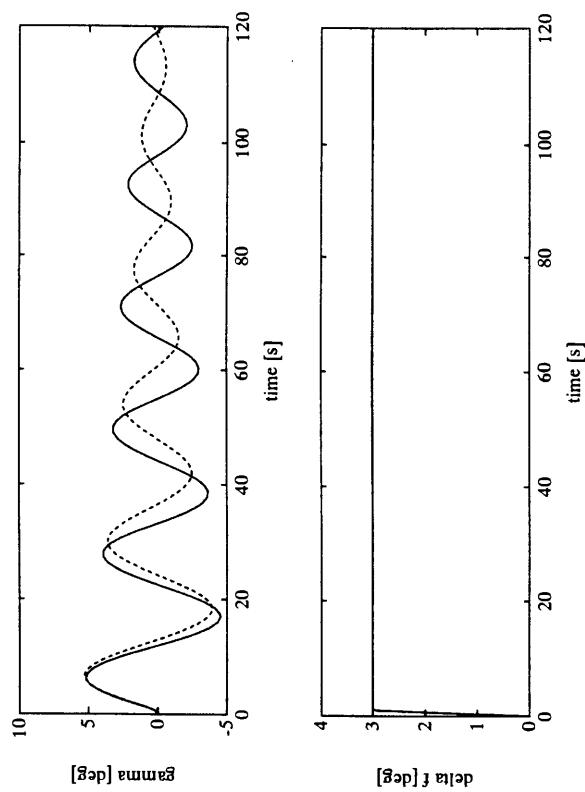


Figure 5-5 (continued).



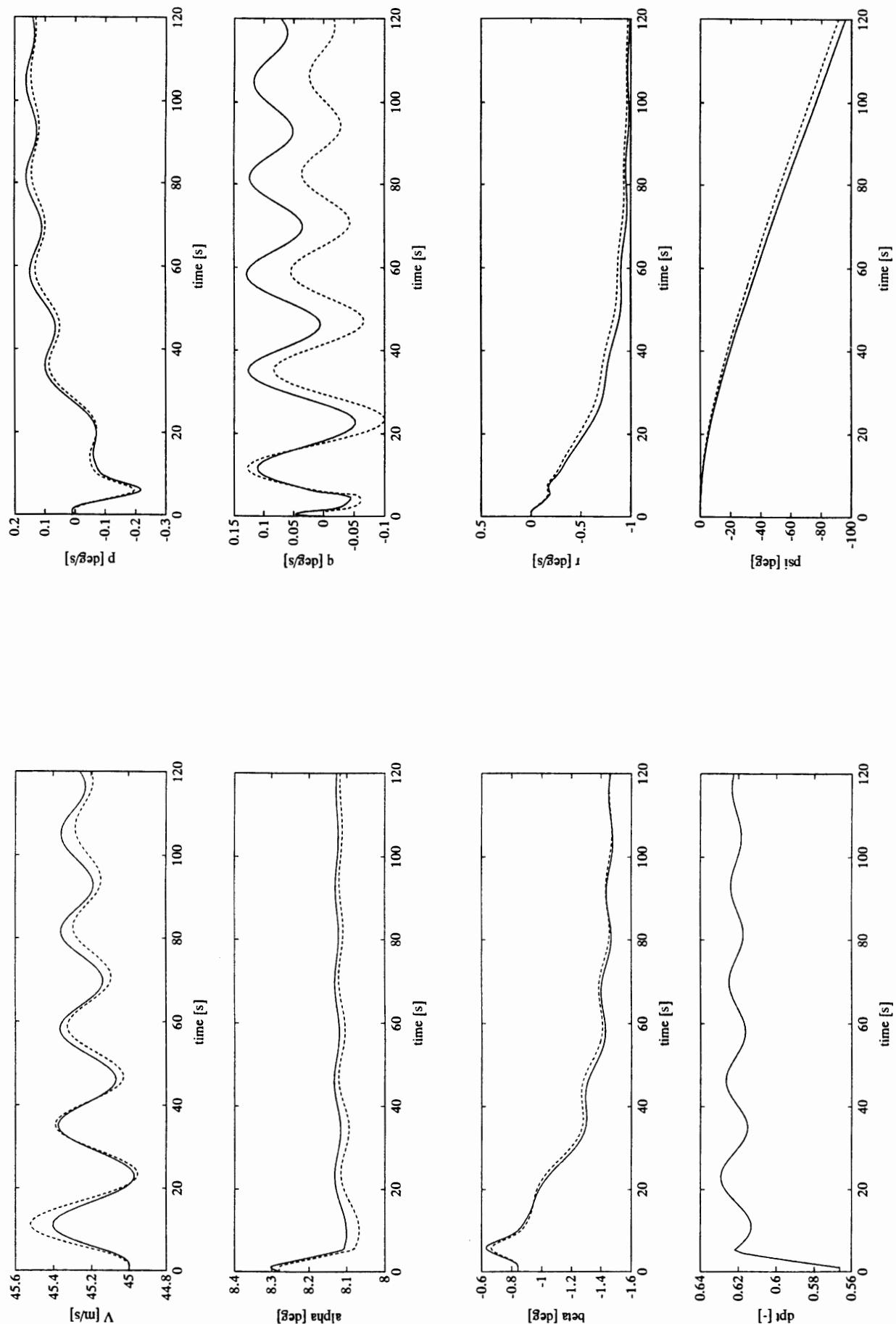


Figure 5-6. Open-loop responses of the 'Beaver' to a change in engine speed $\Delta n = 200 \text{ RPM}$ in 4 seconds. $V_0 = 45 \text{ m/s}$, $H_0 = 6000 \text{ ft}$.

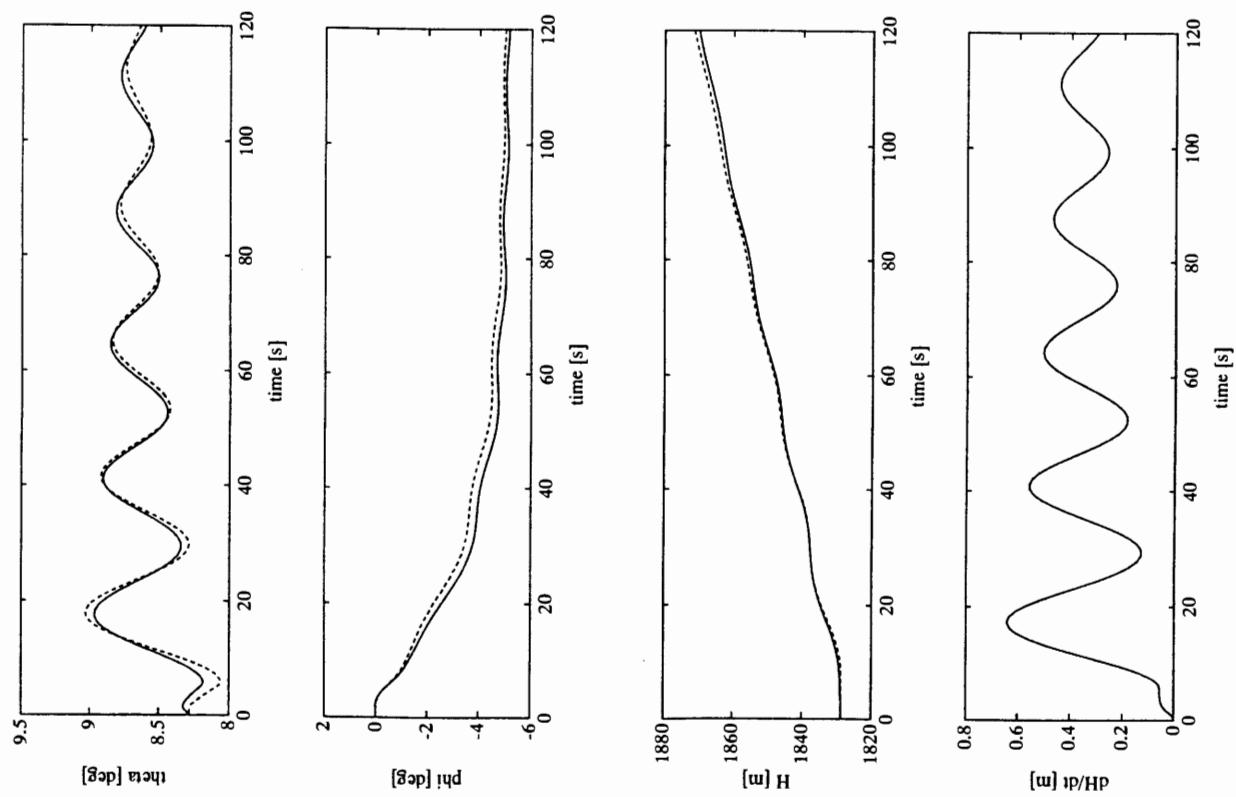
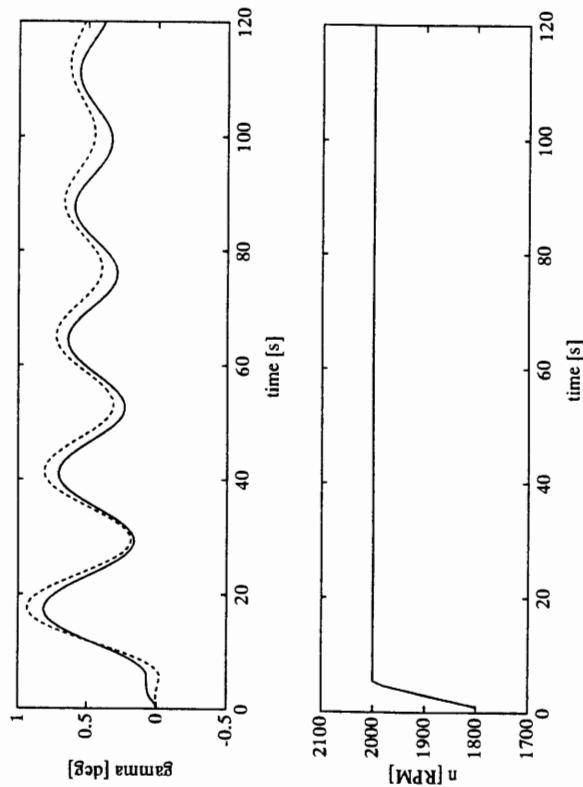


Figure 5-6 (continued).



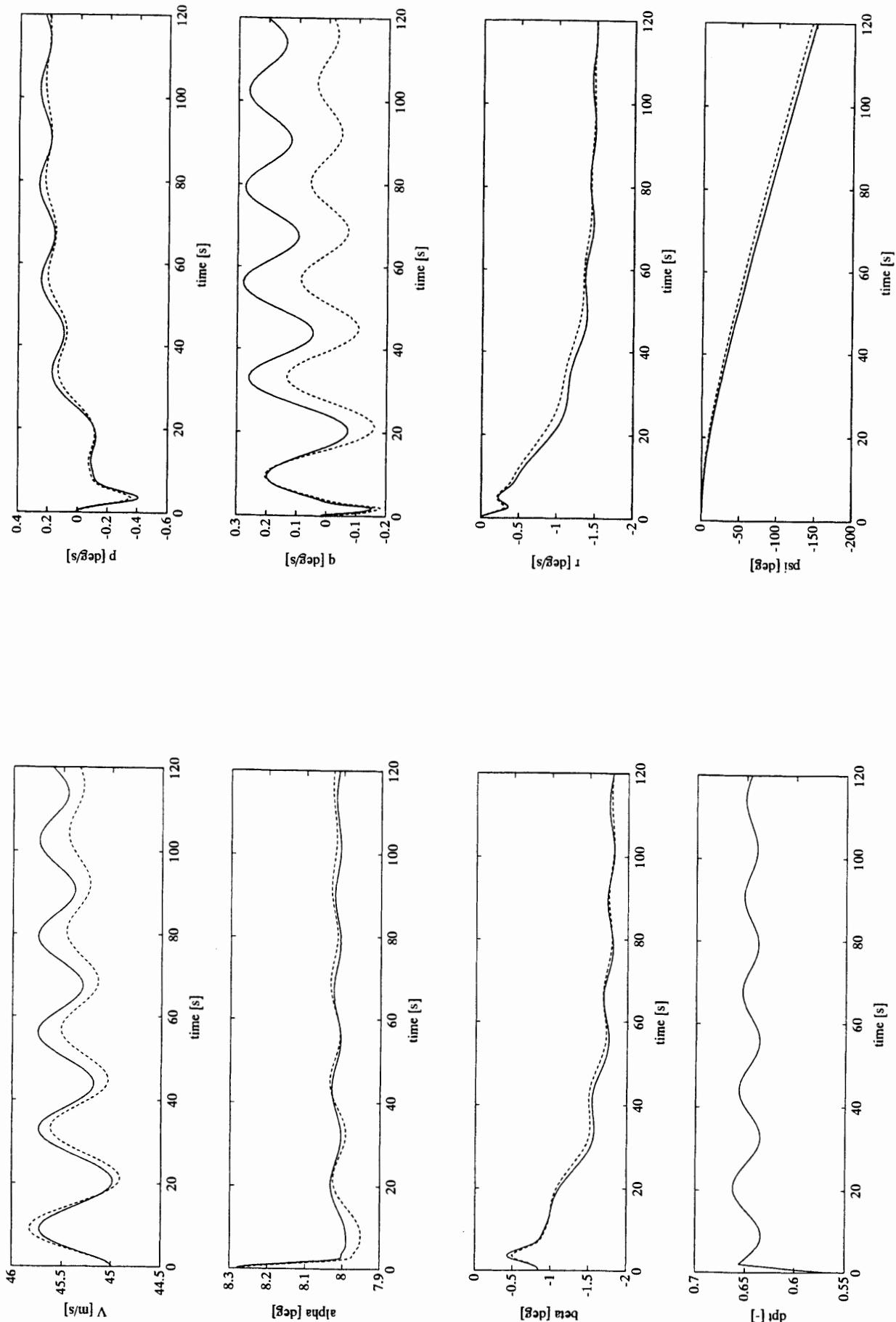


Figure 5-7. Open-loop responses of the 'Beaver' to a change in manifold pressure $\Delta p_z = 2$ inch Hg in 2 seconds. $V_0 = 45$ m/s, $H_0 = 6000$ ft.

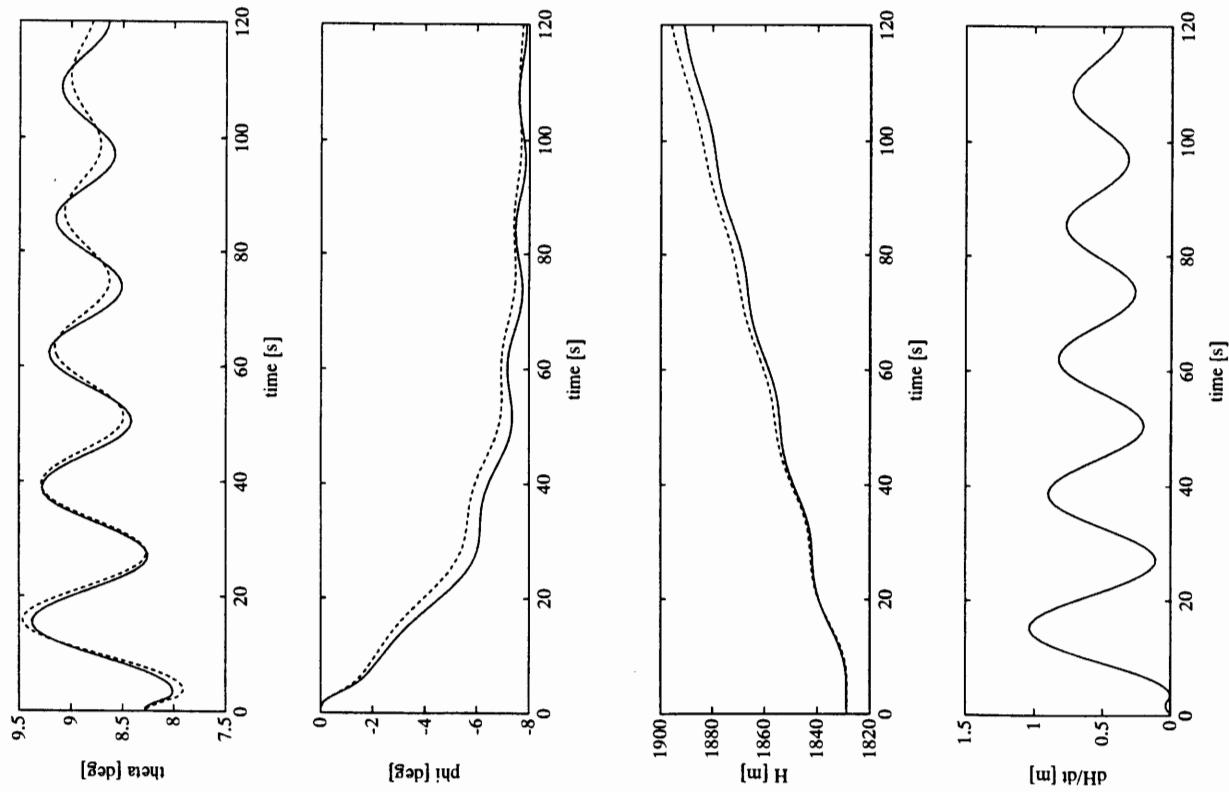
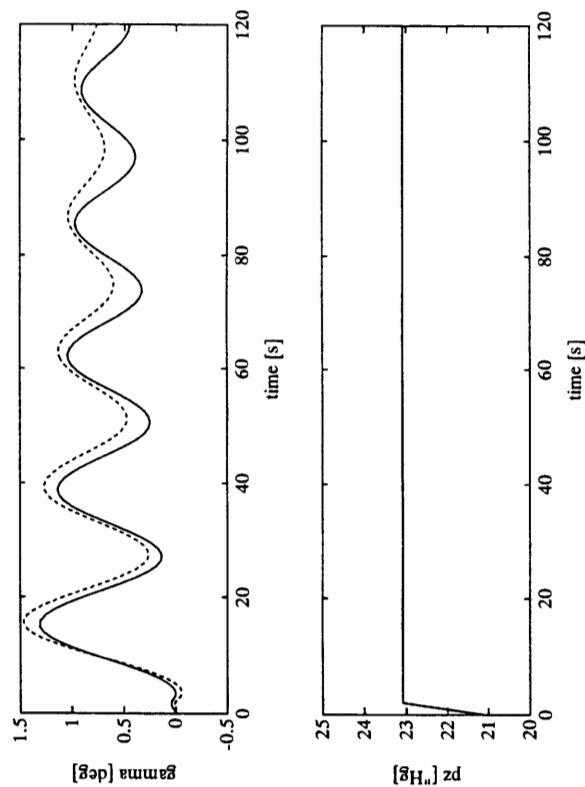


Figure 5-7 (continued).



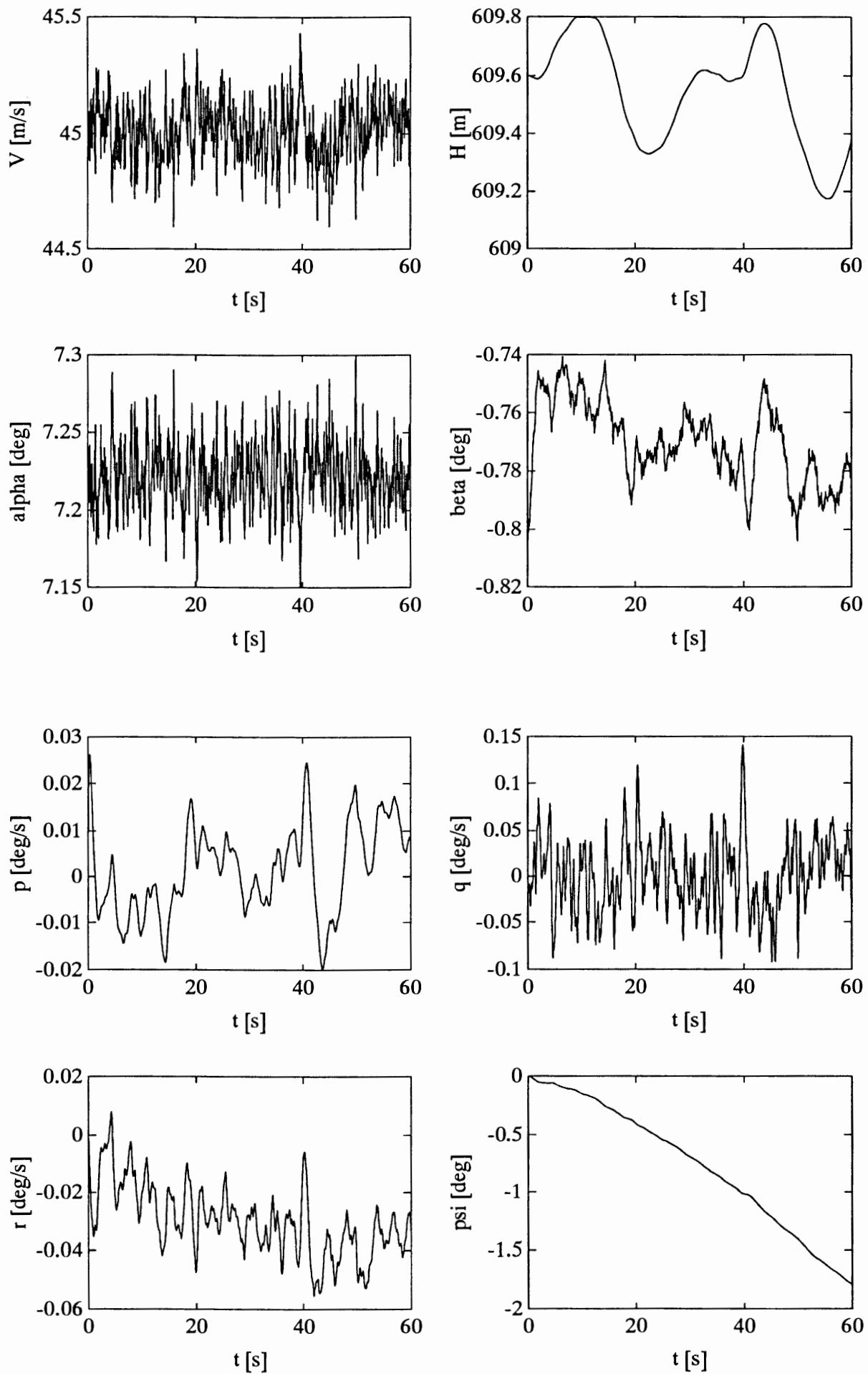


Figure 5-8. Open-loop responses to longitudinal turbulence
(Dryden spectrum, $L_g = 300$ m, $\sigma_g = 1$ m/s, $V_0 = 45$ m/s, $H_0 = 2000$ ft)

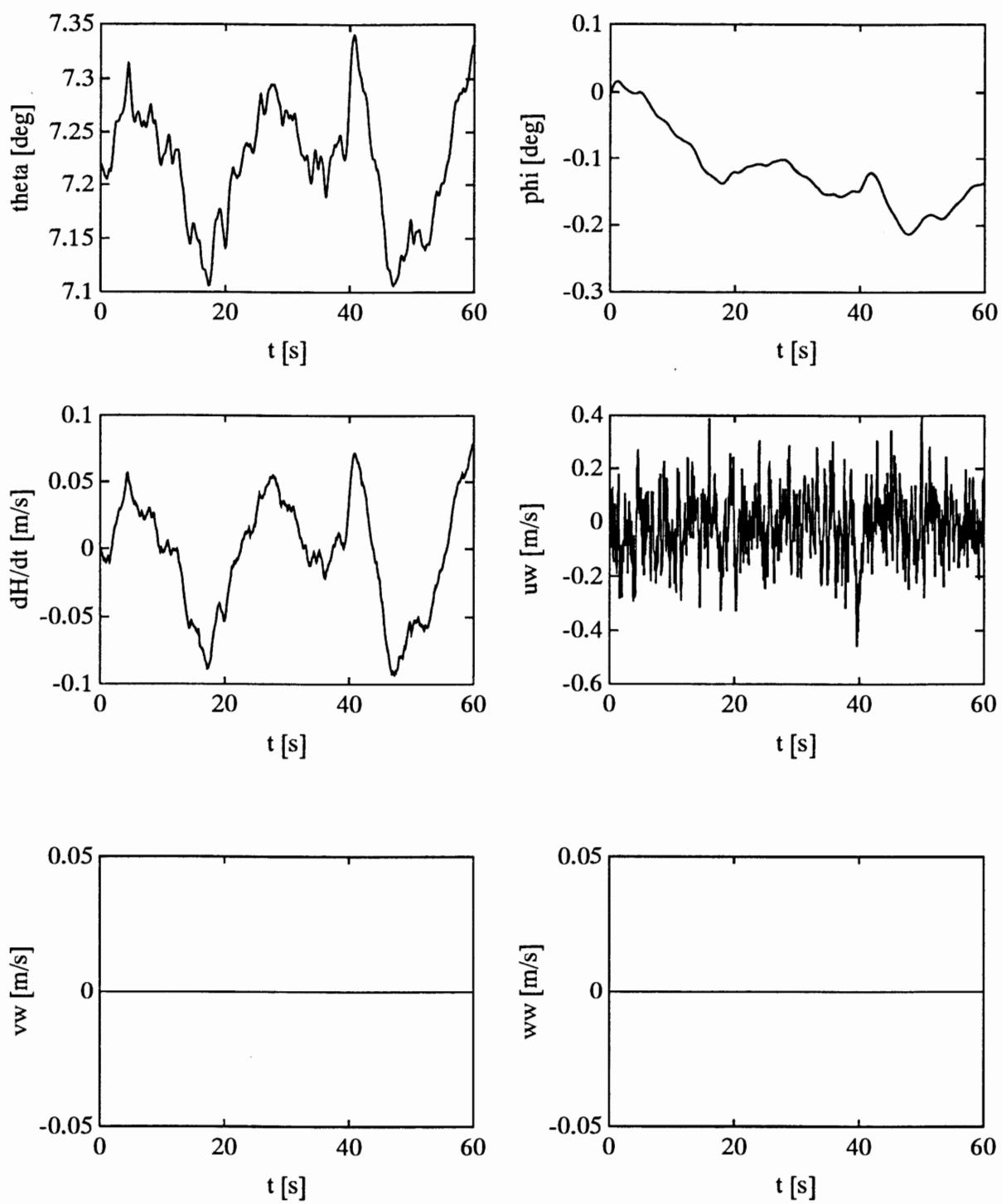


Figure 5-8 (continued).



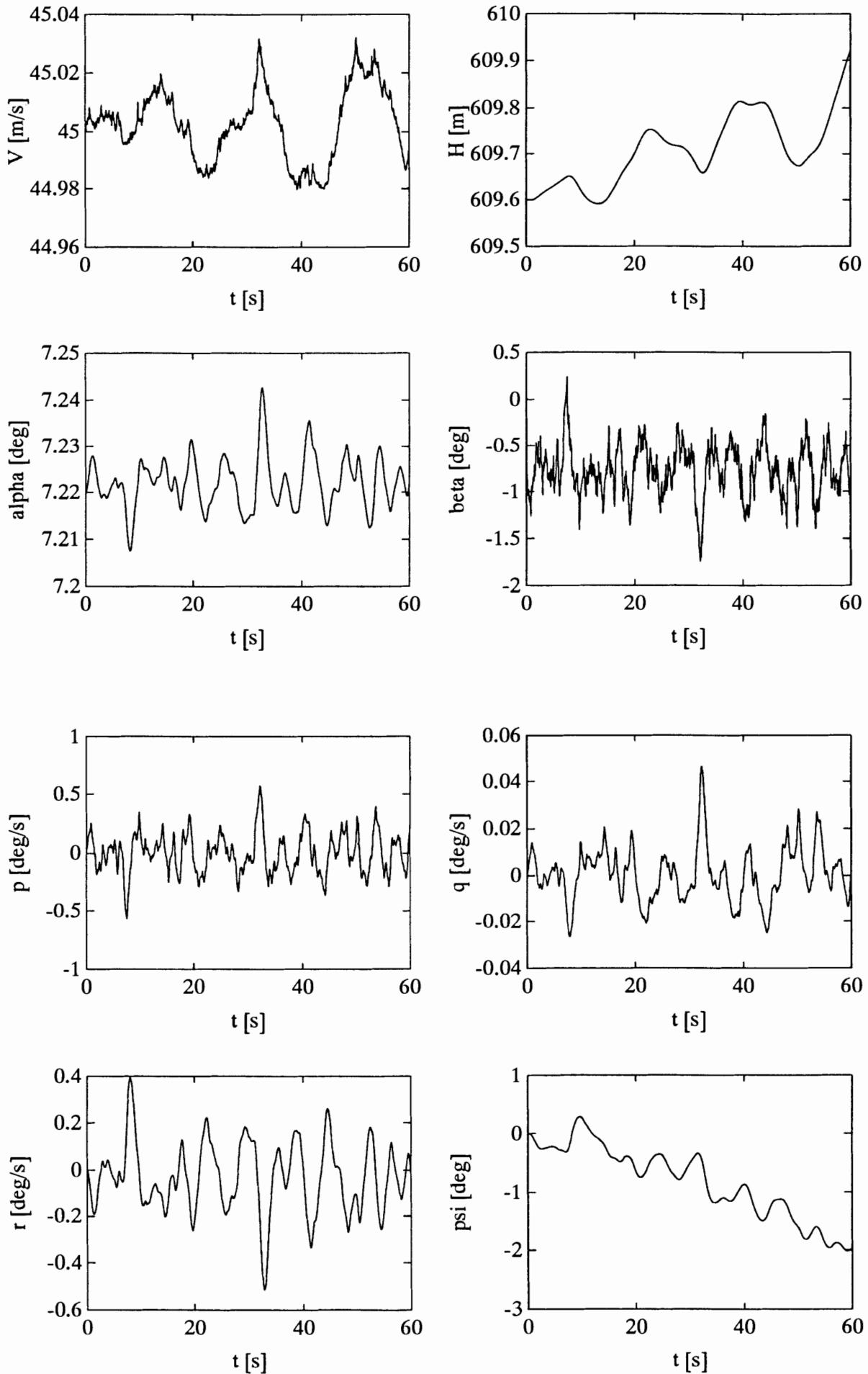


Figure 5-9. Open-loop responses to lateral turbulence
(Dryden spectrum, $L_g = 300$ m, $\sigma_g = 1$ m/s, $V_0 = 45$ m/s, $H_0 = 2000$ ft)

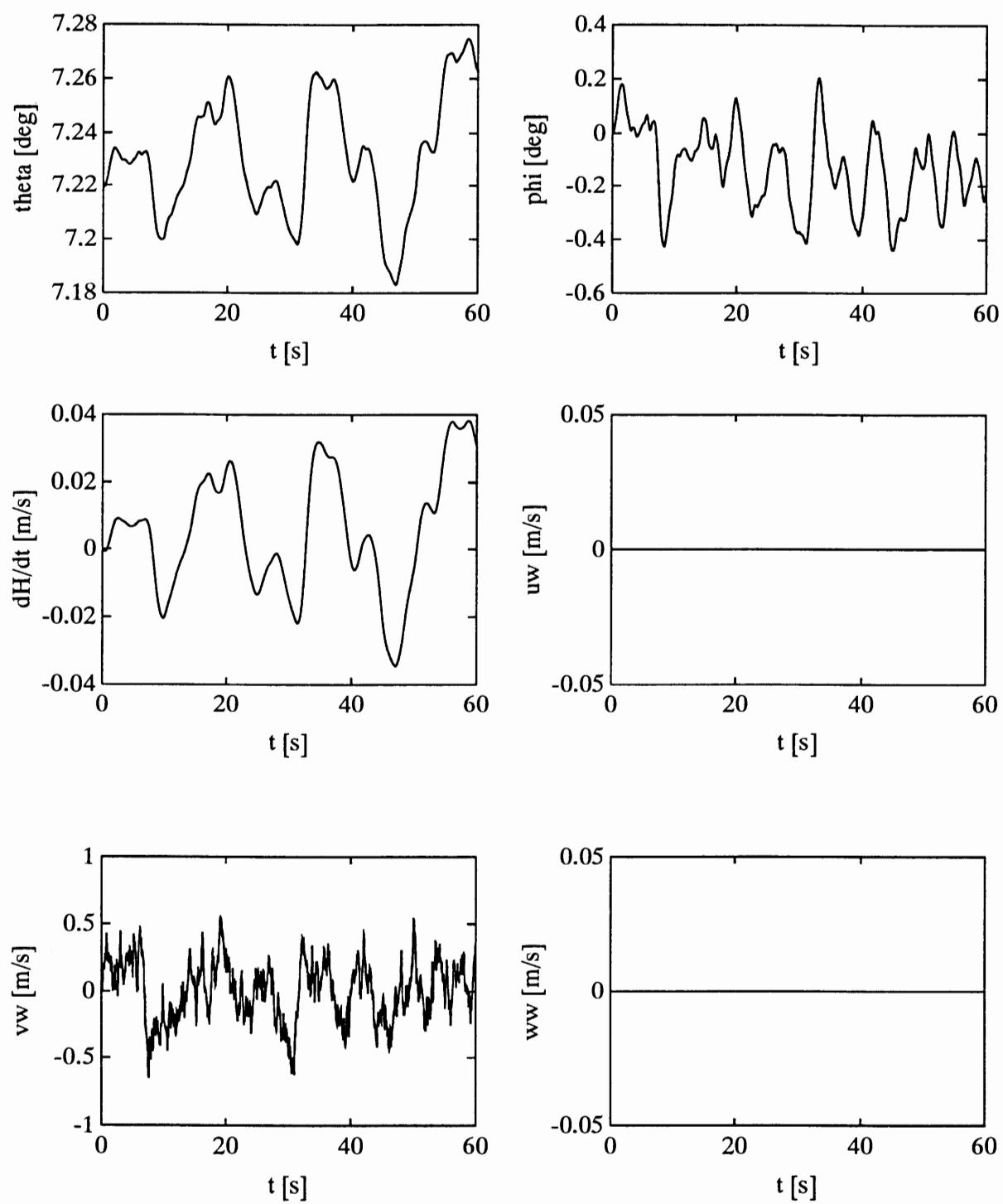


Figure 5-9 (continued).



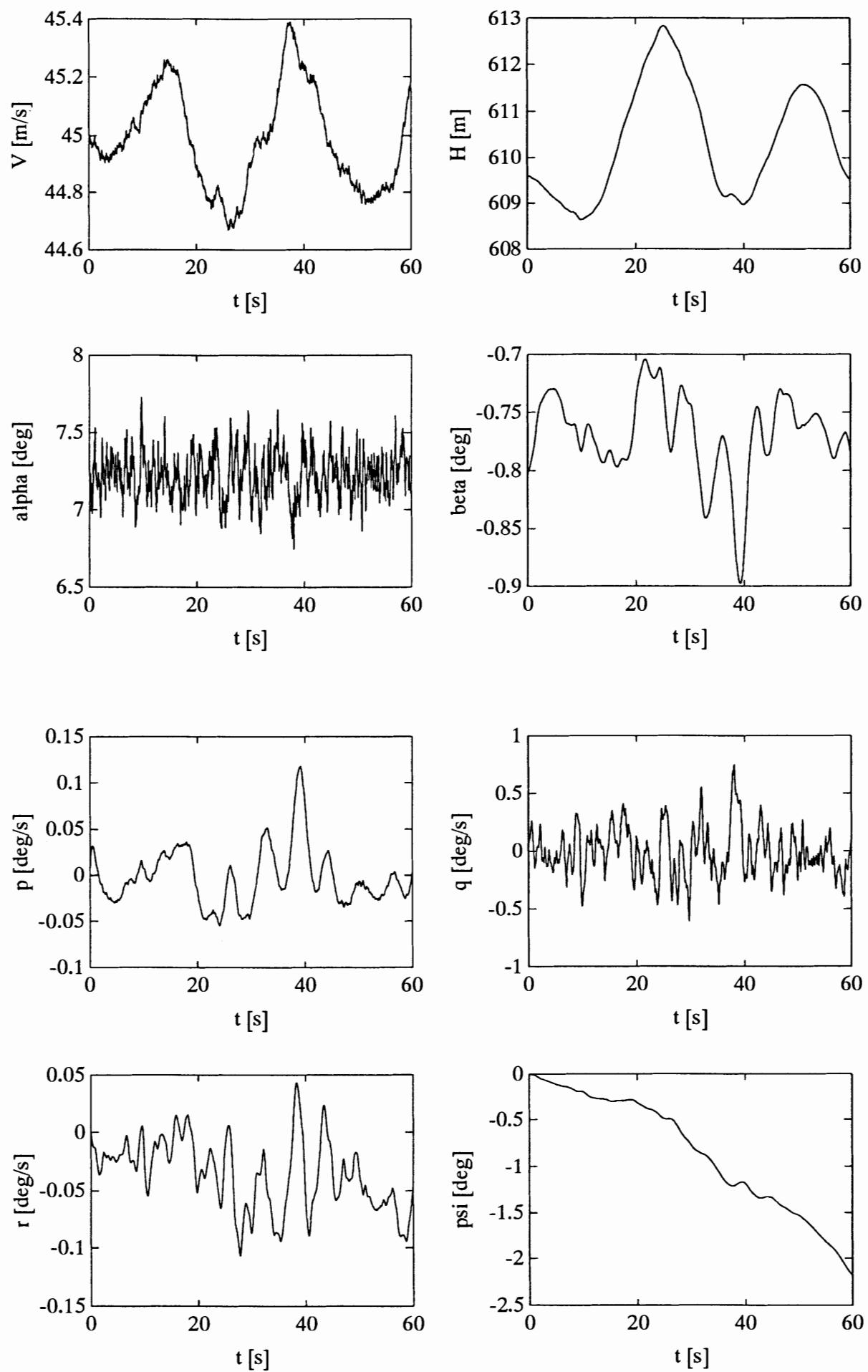


Figure 5-10. Open-loop responses to vertical turbulence
(Dryden spectrum, $L_g = 300$ m, $\sigma_g = 1$ m/s, $V_0 = 45$ m/s, $H_0 = 2000$ ft)

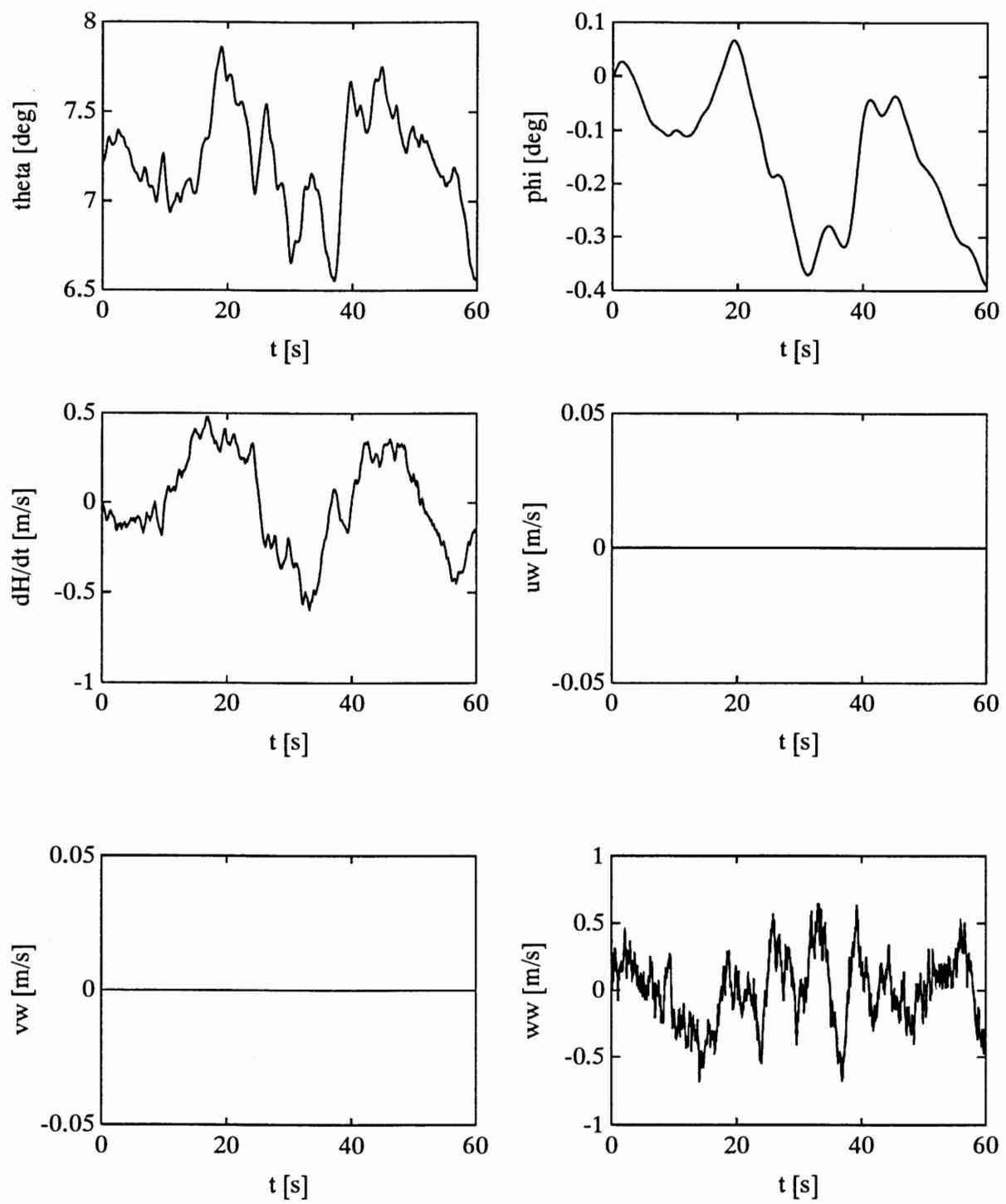


Figure 5-10 (continued).



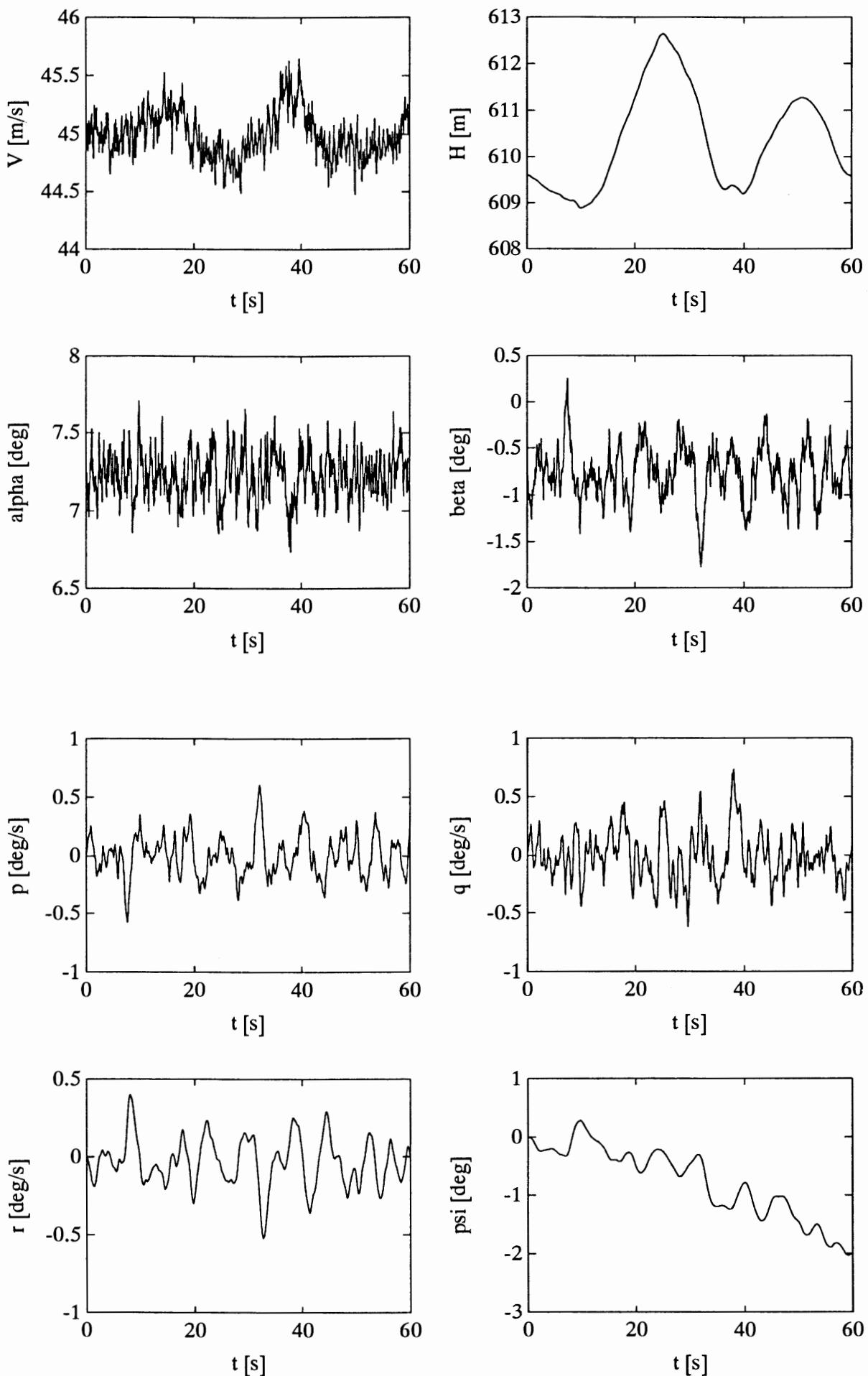


Figure 5-11. Open-loop responses to longitudinal + lateral + vertical turbulence (Dryden, $L_g = 300$ m, $\sigma_g = 1$ m/s, $V_0 = 45$ m/s, $H_0 = 2000$ ft)

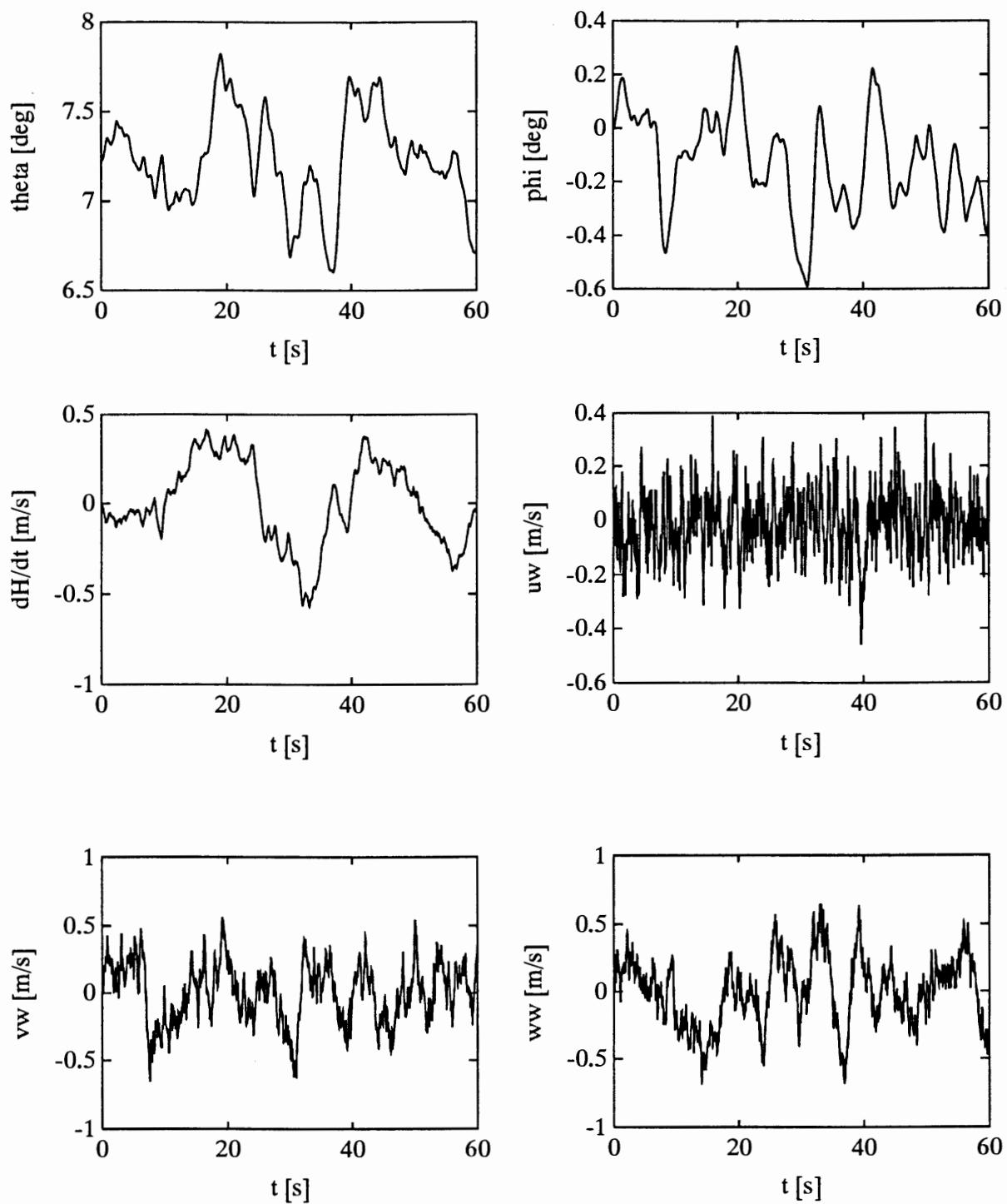


Figure 5-11 (continued).



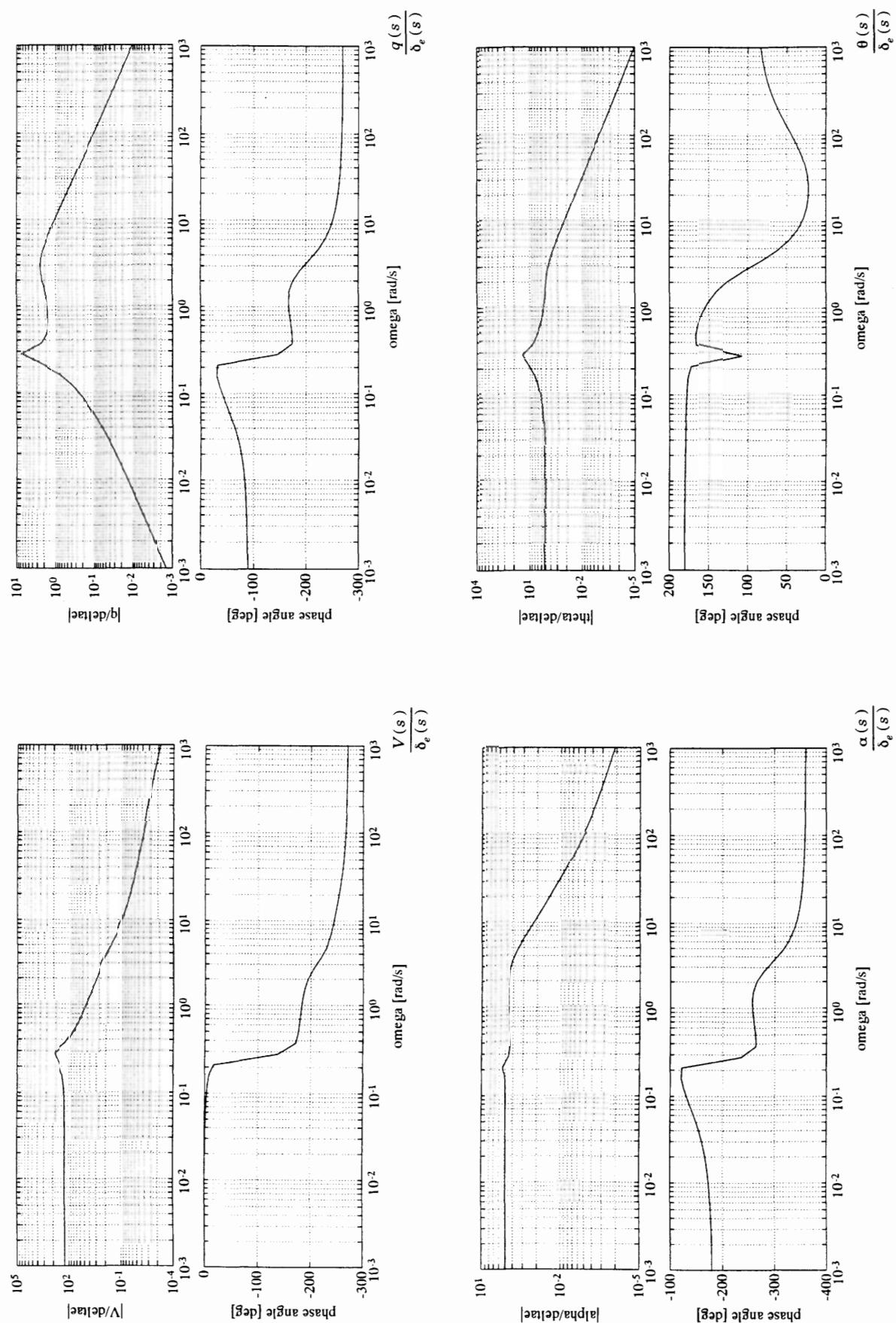


Figure 5-12. Control-input Bode plots for longitudinal linearized 'Beaver' model (input signal: δ_e)



Chapter 6. Using the nonlinear SIMULINK 'Beaver' model in practice.

6.1 Introduction.

In the preceding chapter, the SIMULINK implementation of the nonlinear 'Beaver' model, described in chapter 4, was verified by comparing some open-loop results with other literature. This chapter will show *how* these open-loop results were created. It is recommended to read this chapter first before using the 'Beaver' model for other AACs research purposes, because it will illustrate the main principles. It is advisable to read part II of this report too if you want to design and/or analyze AACs, because there it will be illustrated how the complete 'Beaver' autopilot logic was connected to the 'Beaver' model in SIMULINK. In appendix F, all models and analytical tools will be shown in detail.

All systems and helpfiles mentioned in this chapter have been stored on a floppy disk which should be included to this report (contact the author at the address on the cover page if this floppy disk is *not* included). The systems are contained in the subdirectory EXAMPLES. See appendix H for more information about the directory structure and installation procedure of the models. All model-directories can be added to the MATLAB path by running MODINIT.M first.

6.2 Creating nonlinear open-loop responses to deflections of control surfaces engine inputs.

The nonlinear open-loop responses from section 5.3 were created with the system *OLOOP1*, which is shown in figure 6-1. On the left side of this figure, the input signals are generated. The *S-function* block which calls the nonlinear 'Beaver' model, is contained in the centre of this figure, and on the right side of the figure, some blocks to visualize outputs and/or send outputs to the MATLAB workspace have been included.

To create open-loop responses to deflections of control surfaces or changes in engine-setting, the user must first define the appropriate input signals. In this case, we wanted to examine block-shaped inputs to the elevator, ailerons, and rudder, and ramp-shaped changes of flap angle, engine speed, and manifold pressure. Block and ramp signals are not included in the standard *Sources*-library of SIMULINK, but they can easily be created easily with some *Step-Fcn* blocks and *Rate-limiters*.

The input signals to the control surfaces and flaps are *Mixed* together, yielding the deviation from the nominal inputvector to the aerodynamic model, $\Delta \mathbf{u}_a$. The nominal value $\mathbf{u}_a(0)$ must be added to get the total value of the aerodynamic inputvector \mathbf{u}_a . In a similar way, the input signals to the engine are *Mixed*, yielding the deviation from the nominal inputvector to the engine model, $\Delta \mathbf{u}_t$. The initial value $\mathbf{u}_t(0)$ is added to get \mathbf{u}_t . Finally, \mathbf{u}_a and \mathbf{u}_t are *Mixed* together with the wind and turbulence velocities and accelerations, which in this case are zero. The resulting inputvector to the S-function *BEAVER* thus equals: $\mathbf{u} = [\mathbf{u}_a^T \mathbf{u}_t^T 0 0 0 0 0]^T$.



During the simulation, the time-trajectories of the outputs of the system *BEAVER* are send to the columns of the table *Out* in the MATLAB workspace, the inputs are sent to the columns of the table *In*, and a time-axis is created in the vector *time*. However, as explained in section F.2.4 of appendix F, the outputvector from the *S-function* block is defined differently than the definition of *Out*, see table F-5. The outputvector from the S-function *BEAVER* equals:

$$\mathbf{y} = [V \ \alpha \ \beta \ p \ q \ r \ \psi \ \theta \ \varphi \ x_e \ y_e \ H \ \dot{H} \ \frac{pb}{2V} \ \frac{qc}{V} \ \frac{rb}{2V}]^T$$

So the signals which leave the *Demux* block on the right of the *S-function* block must have sixteen elements in total. The user can connect other blocks to the sixteen outputs from this *Demux* block, for instance auto-scaling plotting blocks or *Scopes*.

Before starting simulations, it is necessary to connect the right input signal(s) to the *Mux* block (in figure 6-1, only a block-shaped input signal to the elevator is considered; all other signals are generated too, but they are not yet connected to the 'Beaver' model). It is also necessary to load the model parameters into the MATLAB workspace and to define the initial conditions. This can be done by double-clicking blocks no. 2 and 3a or 3b on top of the system *OLOOP1*. If the datafiles with model parameters AEROMOD.MAT, AIRCRAFT.MAT, and ENGMOD.MAT are not present in the model directories, they can be generated by clicking block no.1 on top of the system. Finally, it may be useful to fix some state variables, which can be done by clicking block no. 4 on top of the system.

Step no.	Action
1	Connect the required input block(s) to the <i>Mux</i> blocks on the left side of the system.
2	Load model parameters into the MATLAB workspace (click block 2 and if necessary block 1).
3	Define initial conditions <i>xinco</i> , <i>ua0</i> , and <i>ut0</i> in the MATLAB workspace (click block 3a or 3b, or define these vectors manually).
4	If it is necessary to fix some states, click block 4.
5	Set simulation options, and start simulation. Use a <i>Scope</i> or plot block if you want to visualize signals during simulation.
6	The results of the simulation are stored in the columns of the tables <i>In</i> and <i>Out</i> , and a time-axis is stored in the vector <i>time</i> in the MATLAB workspace. The results can be saved, plotted, used for other calculations, etc.

Table 6-1. Steps to be taken to simulate the system from figure 6-1.

Block 3a calls the aircraft trim routine ACTRIM.M, and block 3b calls the load routine INCOLOAD.M (see appendix F.2.7), which can be used to load initial conditions from a MAT-file, which should contain the variables $ua0 = \mathbf{u}_a(0)$, $ut0 = \mathbf{u}_t(0)$, and the initial value of the state vector $xinco = \mathbf{x}(0)$. Of course, it is also possible to create these vectors manually in the MATLAB workspace.

Table 6-1 summarizes the steps which must be taken when performing simulations with the system *OLOOP1* from figure 6-1. Only if the user is not satisfied with the definitions of the input and outputvectors, it is necessary to change the system *BEAVER* (see appendix F). For the simulation of this system, the integrators *ADAMS/GEAR* are best suited, even though they have starting difficulties due to the step-inputs. These integrators give accurate results for an error tolerance of 10^{-3} . Use *RK45* or *RK23* with an error tolerance of 10^{-7} if the simulation spans a small time-interval, or if input signals with many large discontinuities are created.

6.3 Creating linear open-loop responses to deflections of control surfaces and engine inputs.

The system *OLOOP1_L* from figure 6-2 looks similar to the system *OLOOP1* from figure 6-1, but it uses a linear aircraft model in stead of the nonlinear aircraft model *BEAVER*. The same input signals as in *OLOOP1* are generated and *Mixed* on the left of the system. However, in this case, it is not necessary to add the initial values of the inputvectors \mathbf{u}_a and \mathbf{u}_t to these inputsignals, because the linearized aircraft model is a *small-perturbation* model, which expresses all inputs, outputs, and state variables as *deviations* from their nominal values.

Before starting a simulation of *OLOOP1_L*, it is necessary to define the linear state-space matrices *Abeav*, *Bbeav*, *Cbeav*, and *Dbeav* in the MATLAB workspace. This can be done by clicking block 5a on top of the system (in that case, the state-space matrices will be loaded from a MAT-file), by manually defining the matrices (which is not recommendable, because they are quite large), or by using block 5 to linearize the system in the working point, defined by the initial conditions *xinco*, *ua0*, and *ut0*.

When using block 5, the parameters of the 'Beaver' model need to be loaded to the MATLAB workspace first by clicking block no. 2. If SIMULINK can't find the datafiles AEROMOD.MAT, ENGMOD.MAT, and AIRCRAFT.MAT, use block 1 to create them. It is also necessary to define the initial conditions *xinco*, *ua0*, and *ut0* in the MATLAB workspace by clicking block 4 to trim the aircraft model or block 4a to load the initial conditions from a MAT-file before starting the linearization process. Use block 3 if you want to fix some states to their initial values (use this block *before* linearizing!). Table 6-2 summarizes the steps which should be taken when simulating the system *OLOOP1_L* from figure 6-2.

Note that in this case, the matrices *In* and *Out* and the time-axis *time* will not be generated in the MATLAB workspace, because those variables were sent to the workspace internally within the nonlinear aircraft model *BEAVER!* In stead, three new *To Workspace* blocks have been included in the system *OLOOP1_L* to send the time-trajectories of the inputs and outputs to the columns of the matrices *yin* and *yout*, respectively. A time-axis is created too,



Step no.	Action
1	Connect the required input signal(s) to the <i>Mux</i> block on the left side of the system
2	Load the state-space matrices into the MATLAB workspace from file (click block 4a), or: 2a Load model parameters into the MATLAB workspace from file (click block 2 and if the datafiles don't exist yet, click block 1). 2b Define initial conditions $xinco$, $ua0$, and $ut0$, by typing them in, loading them from file (click block 3a), or by trimming the aircraft model (click block 3). 2c Linearize the aircraft model in the workpoint, defined by $xinco$, $ua0$, and $ut0$ (click block 4).
3	Set simulation options and start simulation. Use a <i>Scope</i> block to visualize the input or output signals during simulation.
4	After simulating, a time-axis has been stored in the vector t in the MATLAB workspace, the time-trajectories of the input variables are stored in the matrix yin , and the time-trajectories of the outputs are stored in the matrix $yout$ (see table F-5 for definitions of the S-function inputs and outputs).

Table 6-2. Steps to be taken to simulate the system from figure 6-2.

in the vector t . Table F-5 of appendix F gives the definition of the S-function inputs and outputs, which correspond with the inputs and outputs yin and $yout$, used by the linearized aircraft model. Note: the time-trajectories of x_e and y_e , computed with the linear state-space model are inaccurate, so the linear model is not suitable for navigation purposes (see chapter 5)!

The SIMULINK integrator *LINSIM* is perfectly suited for the simulation of this linear system. It gives accurate results if the error tolerance is set to 10^{-3} . If other, nonlinear system dynamics are added to the system, use *RK45* with an accuracy of 10^{-7} or *ADAMS/GEAR* with an accuracy of 10^{-3} . It might be necessary to use smaller values of the error-tolerance for some purposes.

6.4 Creating nonlinear open-loop responses to atmospheric turbulence.

Figure 6-3 shows the system *OLOOP2*, which was used to create the open-loop responses to atmospheric turbulence, shown in chapter 5. The system is equal to the system from figure 6-1, but this time no external inputs to the control

surfaces and engine have been given, except for the initial (trimmed) values of the input vectors \mathbf{u}_a and \mathbf{u}_t .

The turbulence inputs are generated with the blocks *UDRYD1*, *VDRYD1*, and *WDRYD1*, which will be described in section F.3 of appendix F. These blocks generate turbulence velocities and their time-derivatives according to the Dryden turbulence models, see appendix B. The user must specify the scale lengths L_u , L_v , and L_w , the standard deviations σ_u , σ_v , and σ_w , and the mean velocity. Although the coefficients of the Dryden filters actually depend upon the airspeed, this dependence is neglected here, because the speed-range of the 'Beaver' is quite limited. In section F.3 it is shown how Dryden filters with varying coefficients can be implemented in SIMULINK (giving the blocks *UDRYD2*, *VDRYD2*, and *WDRYD2*). In practice, the filters with constant coefficients will give satisfying results, see section F.3.

Before starting simulations of *OLOOP2*, the model parameters must be loaded into the MATLAB workspace, and the initial conditions $xinco$, $ua0$, and $ut0$ must be defined. Also, it might be useful to neglect some of the aircraft dynamics, by fixing some states to their initial values. The same system initialization blocks as in figure 6-1 have been used here, so the actions, listed in table 6-1 are still valid (only this time, atmospheric disturbances are used as inputs instead of inputs to the engine and/or control surfaces).

The minimum stepsize must be limited to a value which is not too small, because otherwise, the simulations will become very slow. This is caused by the white noise input, which is actually a discrete signal which changes in every time-step taken by the integration method. The plots from chapter 5 were created with a minimum time-step of 0.02 seconds. In this case, the only suitable integration method is *RK45* (or alternatively *RK23*), because the other integrators work efficiently for smooth signals only.

6.5 Linearization of the aircraft model and applications for the linear model.

In the system *OLOOP1_L* from figure 6-2, block 4 was used to linearize the aircraft model in the workpoint, defined by the state vector $xinco$ and the input vectors $ua0$ and $ut0$. The resulting linear aircraft model is obtained directly by applying the SIMULINK routine *LINMOD* to the system *BEAVER*, which yields a linear state-space model which uses the small-perturbation versions of the input and state variables of the system *BEAVER*. Block 4 is therefore called *Tiny linearization of Beaver model*, implying that the linear model is obtained directly with the command *LINMOD*. However, sometimes it may be more convenient to derive separate models for the longitudinal and lateral motions of the aircraft, for which reason, a more sophisticated linearization routine, called *ACLIN*, has been developed. See appendix F.2.7 for details of both the *Tiny* and the *Full-blown* linearization options.

In chapter 5, the routine *ACLIN* has been used to extract some characteristics of the open-loop motions of the 'Beaver'. To do so, the flight-condition of the aircraft had to be defined first by applying the aircraft trim routine *ACTRIM* (this time starting from the MATLAB command-line by typing *actrim*) and then applying *ACLIN* itself (by typing *aclin*).



First the symmetrical state-space models were derived, neglecting the longitudinal-lateral cross-coupling (option 1 in the *Fix States* menu). After linearization, separate models for longitudinal and lateral aircraft dynamics were created (answer Y to the respective question), without taking into account the engine inputs and atmospheric disturbances. The resulting symmetrical state matrices A_{symm} and B_{symm} have been listed in table 5-2.

The same procedure was followed once again to derive the asymmetrical state model, this time using option 2 in the *Fix States* menu. This yielded the asymmetrical state model A_{asym} , B_{asym} from table 5-2. Applying the *eig* command of MATLAB, the eigenvalues of the systems were obtained, from which a number of characteristic quantities for the open-loop motions could easily be derived. The results of this basic open-loop system analysis have been listed in table 5-3.

Finally, the longitudinal linear aircraft model was used to create the Bode-plots from figure 5-12, using the MATLAB commands *bode* and *logspace* as follows:

```
w = logspace(-3,3);
[mag,phase] = bode(Asymm,Bsymm,eye(4),zeros(4,2),1,w);

loglog(w,mag(:,1));
loglog(w,mag(:,2));
loglog(w,mag(:,3));
loglog(w,mag(:,4));

semilogx(w,phase(:,1));
semilogx(w,phase(:,2));
semilogx(w,phase(:,3));
semilogx(w,phase(:,4));
```

Here, the Bode-plots were created for the first input signal, which in this case was δ_e , and the outputs were V , α , q , and θ . When linearizing with *ACLIN*, it will always be specified which definitions of input and output vectors are used.

6.6 Conclusions.

In this chapter, the practical use of the SIMULINK 'Beaver' model has been illustrated by analyzing the systems, used for open-loop analysis of the 'Beaver' in chapter 5. The nonlinear model can be trimmed and linearized fully within the MATLAB/SIMULINK environment, so it is never necessary to move to another computer platform during the *off-line analysis*¹⁾ of the aircraft. Here, only open-loop analysis of the 'Beaver' was treated, but closing the loops in SIMULINK is really straightforward. In part II of this report, the control laws of the 'Beaver' autopilot will be implemented in SIMULINK and connected to the nonlinear 'Beaver' model in a similar way as the open-loop input signals shown in this chapter.

¹⁾ On-line analysis covers piloted simulation in a real-time flightsimulator, off-line analysis covers the whole system identification, design, and analysis, which doesn't need to be done in real-time, and which doesn't require pilot-in-the-loop simulations.

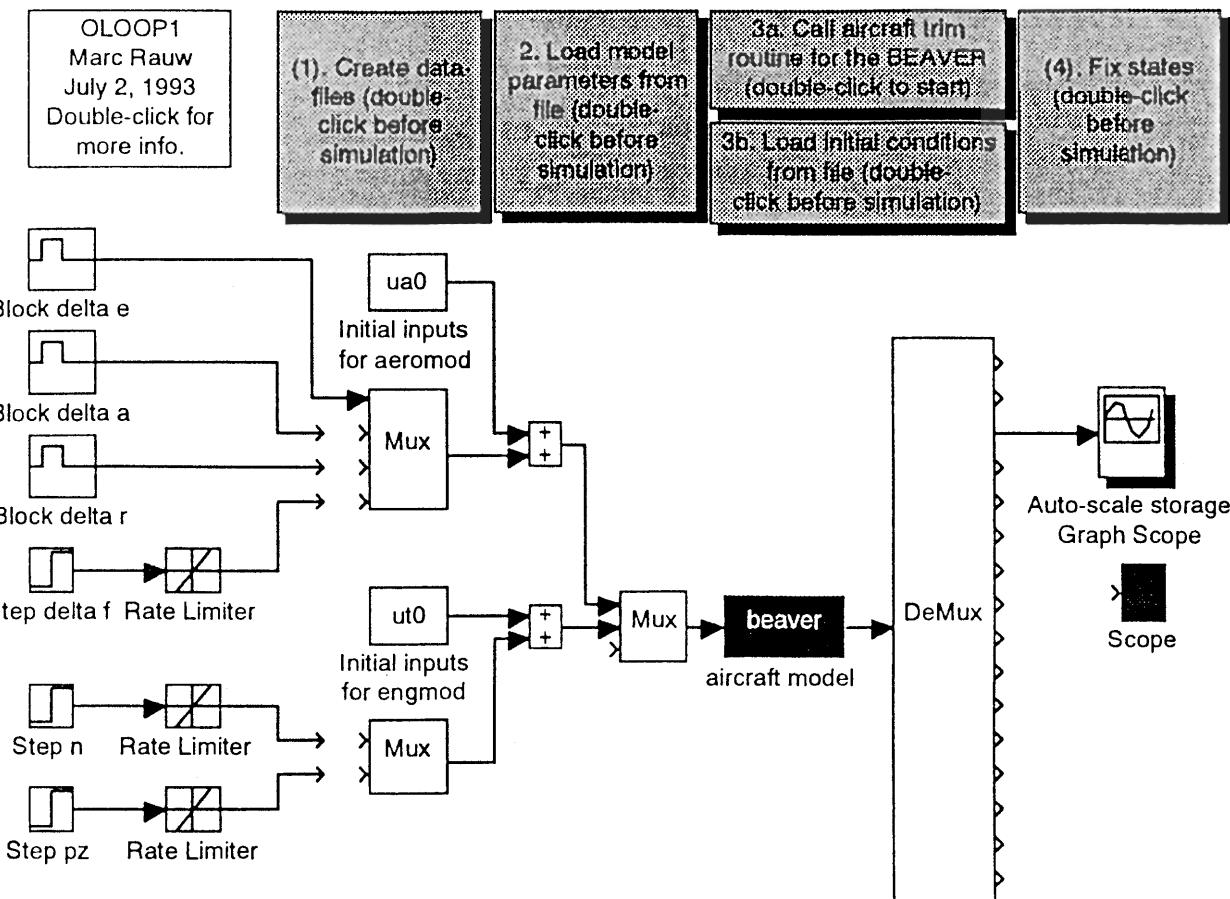


Figure 6-1. Block-diagram of the system OLOOP1 (nonlinear open-loop simulation model for external inputs)

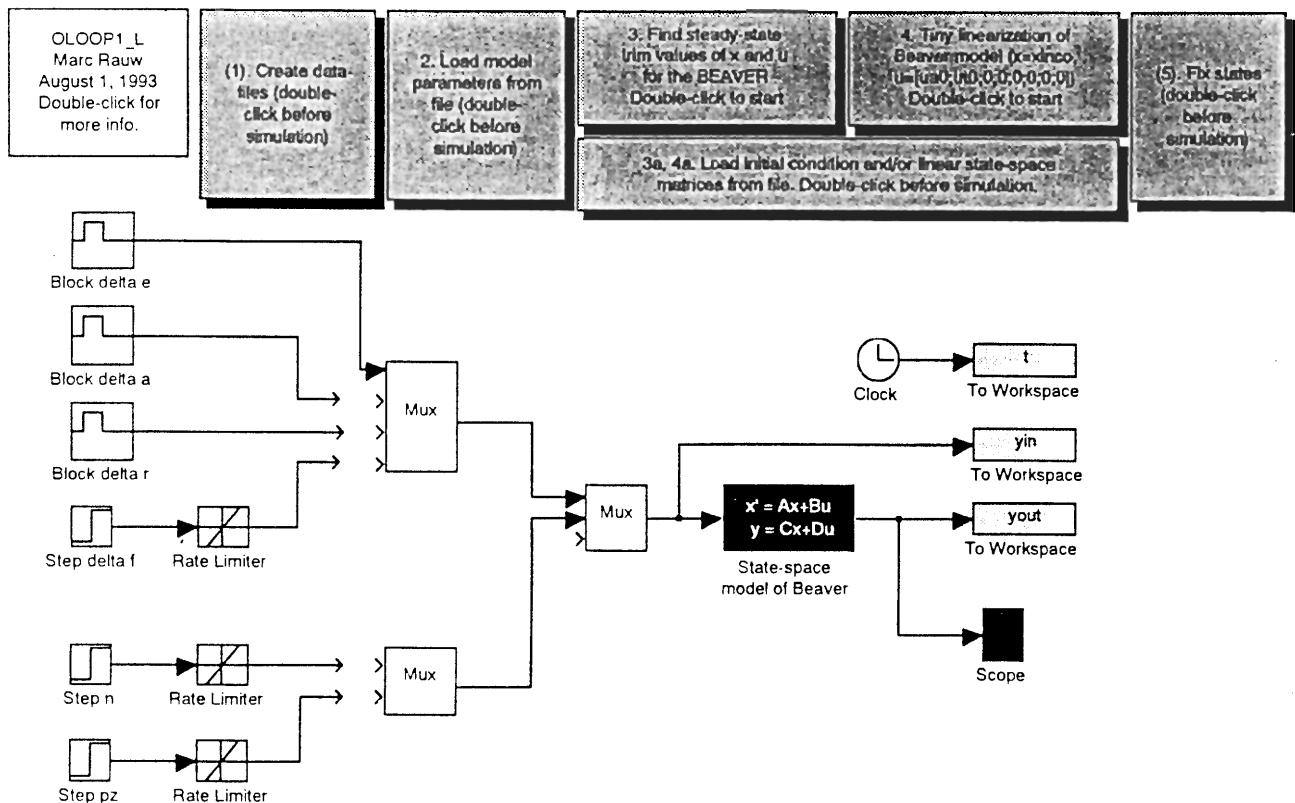


Figure 6-2. Block-diagram of the system OLOOP1_L (linear open-loop simulation model for external inputs)

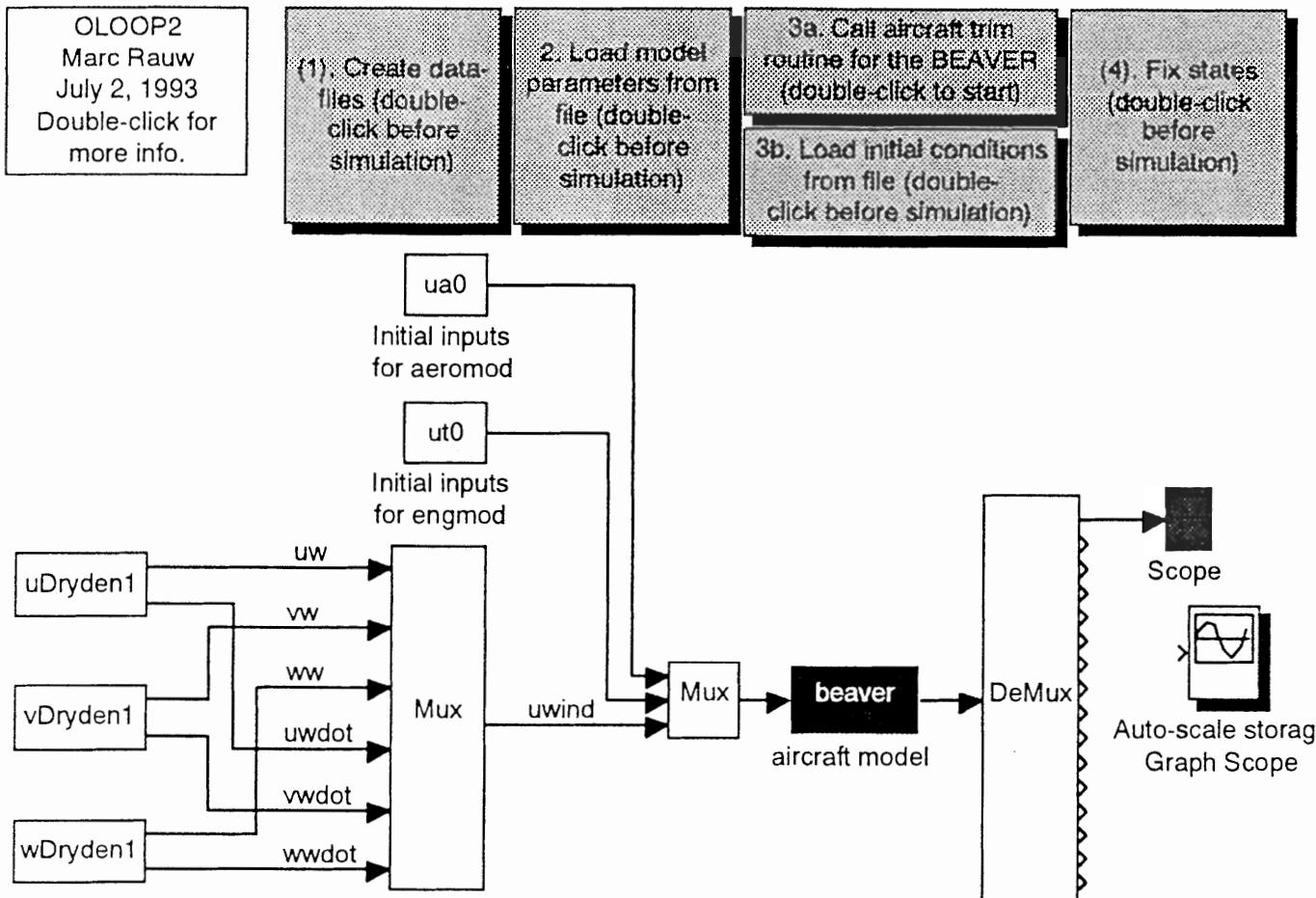


Figure 6-3. Block-diagram of the system OLOOP2 (nonlinear open-loop simulation model for atmospheric disturbances)



Conclusions part I.

In this report, a SIMULINK environment for aircraft dynamics and control analysis has been presented. This environment contains a modular non-linear aircraft model, several submodels for atmospheric disturbances and the calculation of radio-navigation signals, and some helpprograms for the determination of steady-state flight conditions and linearized state-space models of the aircraft. The environment has been applied to the 'Beaver', but due to its modular structure, it is relatively straightforward to implement models of other aircraft.

The SIMULINK tools from this report are particularly useful for application in the field of Automatic Aircraft Control Systems (AACS) design and analysis, because they fully integrate the complete linear control system design and the nonlinear *off-line* AACS-analysis. This is very important because of the iterative character of the AACS development process. Moreover, since the complete range of MATLAB toolboxes can be accessed from this environment, applications for system identification and the analysis of flight-test results are within reach.

The models have been validated by means of some open-loop responses to several inputs, including atmospheric turbulence. The responses are consistent with results from the DUT-flightsimulator software, so it can be concluded that the systems have been implemented correctly. Open-loop responses of the linearized models have been created too, to demonstrate the capabilities of the aircraft trim and linearization tools. Compared to ref.[25], the new models are far more flexible, and also much faster.

The tools do require sufficient computer capacity in order to be able to perform simulations of the aircraft, equipped with a digital controller, or affected by noisy disturbances such as atmospheric turbulence, within a reasonable time. For most purposes, a 486 PC running at 33 MHz or more will suffice. Really sophisticated models will probably need the power of modern workstations.

In the nearby future, the *National Fly-by-wire Testbed* projects for the new Cessna Citation II laboratory aircraft will certainly require the flexibility and power of the SIMULINK environment in order to be able to set new standards for fly-by-wire control research. It will not be sufficient to focus on the hardware and software equipment of the aircraft only.



Recommendations.

The environment for aircraft dynamics and control analysis, presented in this report needs to be developed further into a more general simulation structure in an arbitrary aircraft dynamics model can be implemented with a minimum of programming efforts. Due to the flexibility of the MATLAB/SIMULINK environment, time-consuming conversions from linear system analysis in MATLAB to nonlinear system analysis in Fortran will be eliminated completely, thus reducing the chances of making conversion errors.

The only conversions that will remain are the steps from *off-line* system analysis on a PC or workstation to *on-line* analysis in a real-time flightsimulator, and the step from control system design on a PC to implementation of control laws in the Flight Control Computers of the real aircraft. With regard to the assessment of control laws for autopilots or fly-by-wire systems, it is highly recommendable to smooth-out these conversions as much as possible by developing flexible interfaces between the SIMULINK environment and the flightsimulator or the FCCs respectively.

It is recommended to collect all dynamic SIMULINK models, developed in the Disciplinary group for Stability and Control in standardized model-libraries, stored on a centralized workstation which can be accessed by way of a computer network. If these models are properly managed, it can be made sure that all persons which are involved in a particular research project are always provided with the most up-to-date versions of the models. Also, it can be prevented that everyone has to re-invent the wheel again for each new project is started. If standardized definitions of reference frames and/or lists of symbols are created too, many misunderstandings can be eliminated.



References.

1. Anon. *Approach and Landing Simulation*. AGARD report 632, Ames, 1975.
2. Anon. *International Standards and Recommended Practices*. Annex 10, Volume I, Part I: *Equipment and Systems* and Attachment C to Part I, ICAO, Montreal, Canada, 1968.
3. Anon. *PC-MATLAB User's Guide*. The Mathworks Inc., Natick, Massachusetts, USA, 1989.
4. Anon. *SIMULINK, a program for simulating dynamic systems*. User's Guide, The MathWorks Inc., Natick, Massachusetts, USA, 1992.
5. Abbink, F.J.: *Vliegtuiginstrumentatie I/II* (in Dutch). Lecture Notes D-27/D-31, Delft University of Technology (Aerospace Engineering), Delft, 1984/1983.
6. Baarspul, M.: *Lecture Notes on Flight Simulation Techniques*. Report LR-596, Delft University of Technology (Aerospace Engineering), Delft, 1989.
7. Bauss, W. (editor): *Radio Navigation Systems for Aviation and Maritime Use*. AGARDograph 63, Pergamon Press, UK, 1963.
8. Bosch, P.P.J. van den, Klauw, A.C. van der: *Modelling, Identification and Simulation of Dynamical Systems*. Lecture notes l87, Delft University of Technology (Electrical Engineering), Delft, edition 1992.
9. Chen, Chi-Tsong: *Linear system theory and design*. Holt, Rinehart and Winston Inc., USA, 1984.
10. Colgren, R.D.: *A workstation for the integrated design and simulation of flight control systems*. Lockheed Aeronautical Systems Company, Burbank, California, USA.
11. Duke, E.L., Antoniewicz, R.F., Krambeer, K.D.: *Derivation and Definition of a Linear Aircraft model*. NASA Reference Publication 1207, USA, 1988.
12. Etkin, B.: *Dynamics of Flight; Stability and Control*. Wiley, New York, USA, 2nd edition, 1982.
13. Fogarty, L.E., Howe, R.M.: *Computer mechanization of six-degree of freedom flight equations*. NASA Contractor Report 1344, USA, 1969.
14. Forsythe, G.E., Malcolm, M.A., Moler, C.B.: *Computer Methods for Mathematical Computations*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1977.
15. Gear, C.W.: *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1971.
16. 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.
17. Gerlach, O.H., *Lecture Notes on Aircraft Stability and Control*. Lecture notes D-26, Delft University of Technology (Aerospace Engineering), Delft, 1981.



18. Gobardhan, S.: *Het overzetten van het niet-lineaire model van de 'Beaver' (Remmerswaal) in de structuur, zoals toegepast door Hermansyah bij de implementatie van de F28* (In Dutch). Interim report, Delft University of Technology (Aerospace Engineering), Delft, 1989.
19. Hoogstraten, J.A., Van de Moesdijk, B.: *Modular programming structure applied to the simulation of non-linear aircraft models*. In: IMACS Conference Proceedings on 'Simulation in Engineering Sciences', Nantes, France, 1983.
20. Johnson, W.A., McRuer, D.T.: *Development of a Category II Approach System Model*. NASA Contractor Report 2022, Washington D.C., USA, 1972.
21. Kendal, B.: *Manual of avionics*. BSP Professional Books, UK, 2nd edition, 1987.
22. McLean, D.: *Automatic Flight Control Systems*. Prentice Hall international, Hertfordshire, UK, 1990.
23. McRuer, D., Ashkenas, I., Graham, D.: *Aircraft Dynamics and Automatic Control*. Princeton University Press, Princeton, New Jersey, USA, 1973.
24. 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.
25. Rauw, M.O.: *Using SIMULINK for Nonlinear Analysis of Aircraft Control Systems*. Interim report, Delft University of Technology (Aerospace Engineering), Delft, 1992.
26. Rolfe, J.M., Staples, K.J. (editors): *Flight Simulation*. Aerospace Series, Cambridge University Press, Cambridge, UK, 1986.
27. Ruijgrok, G.J.J.: *Elements of airplane performance*. Delft University Press, Delft, 1990.
28. Stengel, R.F., Sircar, S.: *Computer-Aided Design of Flight Control Systems*. AIAA-91-2677-CP, Princeton, New Jersey, USA, 1991.
29. Stevens, B.L., Lewis, F.L.: *Aircraft Control and Simulation*. John Wiley & Sons Inc., 1992.
30. 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.
31. Tomlinson, B.N., Padfield, G.D., Smith, P.R.: *Computer-Aided control law research; From concept to flight test*. In: AGARD conference proceedings on 'Computer Aided System Design and Simulation', AGARD CP-473, London, 1990.



Appendices



Appendix A. Nonlinear equations of motion for a rigid aircraft.

A.1 Introduction.

The general nonlinear equations of motion for a rigid aircraft in steady atmosphere will be established in this appendix. There are many textbooks on the matter, see for instance refs.[11], [12], [17], [22], [23], or [29]. Still, a thorough knowledge of the general equations of motion is considered to be such vital for any discussion about the implementation and use of aircraft models, that a complete derivation of the equations of motion will be presented here. One should particularly take notice of all the assumptions which shall be made during the derivation.

A.2 Derivation of the equations for velocities and rotational velocities in the body-fixed reference frame.

A.2.1 General force equation for a rigid body.

Consider a mass point δm that moves with time varying velocity \mathbf{V} (with respect to a right-handed orthogonal reference frame OXYZ) under the influence of a force $\delta \mathbf{F}$. Applying Newton's second law yields:

$$\delta \mathbf{F} = \delta m \cdot \dot{\mathbf{V}} \quad (\text{A-1})$$

For a rigid body it is possible to write:

$$\sum \delta \mathbf{F} = \sum \delta m \frac{d\mathbf{V}}{dt} = \frac{d}{dt} \sum \mathbf{V} \delta m \quad (\text{A-2})$$

in which the contributions of all mass points are summed across the rigid body. Let the centre of gravity of the rigid body have a velocity $\mathbf{V}_{c.g.}$ with components u , v , and w along the X, Y, and Z-axis of the right-handed reference frame. The velocity of each mass point then equals the sum of $\mathbf{V}_{c.g.}$ and the velocity of the mass point with respect to the centre of gravity. If the position of the mass point with respect to the c.g. is denoted by the vector \mathbf{r} , the following vector equation is found:

$$\mathbf{V} = \mathbf{V}_{c.g.} + \dot{\mathbf{r}} \quad (\text{A-3})$$

so:

$$\sum \mathbf{V} \delta m = \sum (\mathbf{V}_{c.g.} + \dot{\mathbf{r}}) \delta m = m \mathbf{V}_{c.g.} + \frac{d}{dt} \sum \mathbf{r} \delta m \quad (\text{A-4})$$



In this equation, m denotes the total mass of the rigid body. In the centre of gravity we can write:

$$\sum \mathbf{r} \delta m = 0 \quad (\text{A-5})$$

so the equation for the resulting force \mathbf{F} , acting on the rigid body becomes:

$$\mathbf{F} = m \dot{\mathbf{V}}_{c.g.} \quad (\text{A-6})$$

A.2.2 General moment equation for a rigid body.

The moment $\delta\mathbf{M}$, about the centre of gravity, is equal to the time derivative of the angular momentum of the mass point relative to the c.g.:

$$\delta\mathbf{M} = \frac{d}{dt} (\mathbf{r} \times \mathbf{V}) \delta m = (\dot{\mathbf{r}} \times \mathbf{V}) \delta m + (\mathbf{r} \times \dot{\mathbf{V}}) \delta m \quad (\text{A-7})$$

where:

$$\dot{\mathbf{r}} = \mathbf{V} - \mathbf{V}_{c.g.} \quad (\text{A-8})$$

and:

$$(\mathbf{r} \times \dot{\mathbf{V}}) \delta m = \mathbf{r} \times \delta\mathbf{F} = \delta\mathbf{G} \quad (\text{A-9})$$

Here $\delta\mathbf{G}$ denotes the moment of the force $\delta\mathbf{F}$ about the centre of gravity. The angular momentum of the mass point relative to the c.g. will be denoted by $\delta\mathbf{h}$ (hence: $\delta\mathbf{h} = (\mathbf{r} \times \mathbf{V}) \delta m$). Writing this out yields:

$$\delta\mathbf{G} = \dot{\mathbf{h}} - (\mathbf{V} - \mathbf{V}_{c.g.}) \times \mathbf{V} \delta m = \dot{\mathbf{h}} + \mathbf{V}_{c.g.} \times \mathbf{V} \delta m \quad (\text{A-10})$$

The contributions of all mass points are once again summed across the whole rigid body, yielding:

$$\sum \delta\mathbf{G} = \frac{d}{dt} \sum \dot{\mathbf{h}} + \mathbf{V}_{c.g.} \times \sum \mathbf{V} \delta m \quad (\text{A-11})$$

The equation for the resulting moment \mathbf{G} , about the c.g. becomes:

$$\mathbf{G} = \dot{\mathbf{h}} \quad (\text{A-12})$$

where \mathbf{h} denotes the resulting angular momentum of the body about the c.g.

A.2.3 Angular momentum about the centre of gravity.

If the body has an angular velocity $\boldsymbol{\omega}$, with components p , q , and r about the X, Y, and Z axes of the right-handed reference frame respectively:

$$\boldsymbol{\omega} = \mathbf{i}p + \mathbf{j}q + \mathbf{k}r \quad (\text{A-13})$$

(where \mathbf{i} , \mathbf{j} , and \mathbf{k} are unity vectors about the X, Y, and Z axes), the velocity of a mass point of the rotating body becomes:

$$\mathbf{V} = \mathbf{V}_{c.g.} + \boldsymbol{\omega} \times \mathbf{r} \quad (\text{A-14})$$

hence, the angular momentum of the rigid body about the c.g. can be written as:

$$\mathbf{h} = \sum \mathbf{r} \times (\mathbf{V}_{c.g.} + \boldsymbol{\omega} \times \mathbf{r}) \delta m = \sum \mathbf{r} \times \mathbf{V}_{c.g.} \delta m + \sum \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r}) \delta m \quad (\text{A-15})$$

The first term of the right hand side of (A-15) can be written as:

$$(\sum \mathbf{r} \delta m) \times \mathbf{V}_{c.g.} = 0 \quad (\text{A-16})$$

and for the second term we can write:

$$\sum \mathbf{r} \times (\boldsymbol{\omega} \times \mathbf{r}) \delta m = \sum (\boldsymbol{\omega}(\mathbf{r} \cdot \mathbf{r}) - \mathbf{r}(\boldsymbol{\omega} \cdot \mathbf{r})) \delta m = \sum (\boldsymbol{\omega} r^2 - \mathbf{r}(\boldsymbol{\omega} \cdot \mathbf{r})) \delta m \quad (\text{A-17})$$

Substitution of $\mathbf{r} = \mathbf{i}x + \mathbf{j}y + \mathbf{k}z$, (A-16), and (A-17) in (A-15) yields:

$$\mathbf{h} = \boldsymbol{\omega} \sum (x^2 + y^2 + z^2) \delta m - \sum \mathbf{r} (px + qy + rz) \delta m \quad (\text{A-18})$$



symbol	definition
I_x	$\sum (y^2 + z^2) \delta m$
I_y	$\sum (x^2 + z^2) \delta m$
I_z	$\sum (x^2 + y^2) \delta m$
J_{xy}	$\sum xy \delta m$
J_{xz}	$\sum xz \delta m$
J_{yz}	$\sum yz \delta m$

Table A-1. Moments and products of inertia.

The components of \mathbf{h} along the X, Y, and Z axes will be denoted as h_x , h_y , and h_z , respectively, yielding:

$$\begin{aligned} h_x &= p \sum (y^2 + z^2) \delta m - q \sum xy \delta m - r \sum xz \delta m \\ h_y &= -p \sum xy \delta m + q \sum (x^2 + z^2) \delta m - r \sum yz \delta m \\ h_z &= -p \sum xz \delta m - q \sum yz \delta m + r \sum (x^2 + y^2) \delta m \end{aligned} \quad (\text{A-19})$$

The summations appearing in these equations are defined as the inertial moments and products about the X, Y, and Z axes, respectively; see table A-1¹⁾.

A.2.4 General equations of motion for a rigid body.

When we choose a reference frame fixed to the body ($OXYZ = OX_B Y_B Z_B$) the inertial moments and products are constants. The reference frame now rotates with angular velocity ω . For an arbitrary position vector \mathbf{a} with respect to the body reference frame, we can then write:

$$\dot{\mathbf{a}} = \frac{\partial \mathbf{a}}{\partial t} + \boldsymbol{\omega} \times \mathbf{a} \quad (\text{A-20})$$

¹⁾ The summations across the body actually have to be written as integrals, but that refinement is omitted here.

Let the body now be an airplane that is considered to be rigid. Applying (A-20) to (A-6) and (A-12), we find:

$$\mathbf{F} = m \frac{\partial \mathbf{V}_{c.g.}}{\partial t} + m\boldsymbol{\omega} \times \mathbf{V}_{c.g.} \quad (\text{A-21})$$

and:

$$\mathbf{G} = \frac{\partial \mathbf{h}}{\partial t} + \boldsymbol{\omega} \times \mathbf{h} \quad (\text{A-22})$$

The force and moment equations (A-6) and (A-12) can be written out into their components along the X_B , Y_B , and Z_B -axes (i.e. $\mathbf{F} = \mathbf{i} F_x + \mathbf{j} F_y + \mathbf{k} F_z$ and $\mathbf{G} = \mathbf{i} L + \mathbf{j} M + \mathbf{k} N$), using equations (A-21) and (A-22). This yields:

$$\begin{aligned} F_x &= m(\dot{u} + qw - rv) \\ F_y &= m(\dot{v} + ru - pw) \\ F_z &= m(\dot{w} + pv - qu) \end{aligned} \quad (\text{A-23a})$$

and:

$$\begin{aligned} L &= \dot{h}_x + qh_z - rh_y \\ M &= \dot{h}_y + rh_x - ph_z \\ N &= \dot{h}_z + ph_y - qh_x \end{aligned} \quad (\text{A-23b})$$

It should be noted that the derivation of these equations is only valid when the following restrictive assumptions are made:

- 1 - the airframe is assumed to be a rigid body in the motion under consideration,
- 2 - the airplane's mass is assumed to be constant during the time interval in which its motions are studied,
- 3 - the earth is assumed to be fixed in space, i.e. its rotation is neglected,
- 4 - the curvature of the earth is neglected.

Assumptions 3 and 4 were made in the definition of the inertial reference frame in which the aircraft's motions will be considered. The description of the vehicle motion under assumptions 3 and 4 is accurate for relatively short-term guidance and control analysis purposes. It does have practical limitations when very long term navigation or extra-atmosphere operations are of interest [23].



For aircraft flying in steady atmosphere, the forces about the X_B , Y_B , and Z_B -axes will be divided in three components:

- 1 - forces resulting from the weight of the aircraft $W = m \cdot g$ (parallel to the Z_E -axis of the earth-fixed reference frame), which will be denoted as X_{gr} , Y_{gr} , and Z_{gr} ,
- 2 - forces resulting from operation of the engine(s) of the aircraft, which will be denoted as X_t , Y_t , and Z_t ,
- 3 - aerodynamic forces, which will be denoted as X_a , Y_a , and Z_a .

In principle, it is possible to add other contributions, such as forces coming from the landing gear if the aircraft is taxiing, but only the contributions mentioned above will be considered here.

Gravity does not contribute to the moments about the body-axes, because the centre of gravity is located in the origin of this reference frame. So the moments can be divided in contributions from aerodynamic properties of the aircraft (subscript a) and contributions from operation of the engine (subscript t). With this division of the forces and moments in different components, and using (A-19), the equations of motion (A-23) can be written as:

$$\begin{aligned} X_{gr} + X_t + X_a &= m(\dot{u} + qw - rv) \\ Y_{gr} + Y_t + Y_a &= m(\dot{v} + ru - pw) \\ Z_{gr} + Z_t + Z_a &= m(\dot{w} + pv - qu) \end{aligned} \quad (\text{A-24a})$$

and:

$$\begin{aligned} L_t + L_a &= I_x \dot{p} + (I_z - I_y) qr - J_{xy}(\dot{q} - pr) - J_{xz}(\dot{r} + pq) - J_{yz}(q^2 - r^2) \\ M_t + M_a &= I_y \dot{q} + (I_x - I_z) pr - J_{xy}(\dot{p} + qr) + J_{xz}(p^2 - r^2) - J_{yz}(\dot{r} - pq) \\ N_t + N_a &= I_z \dot{r} + (I_y - I_x) pq - J_{xy}(p^2 - q^2) - J_{xz}(\dot{p} - qr) - J_{yz}(\dot{q} + pr) \end{aligned} \quad (\text{A-24b})$$

(see ref.[11]). It is possible to move the derivatives of the motion variables to the left hand side of the equations if the forces and moments don't contain any contributions from derivatives of the velocities u , v , and w , and/or the angular velocities p , q , and r . This yields a set of Ordinary Differential Equations (ODEs). If the forces and moments do contain such time derivatives, a set of implicit differential equations results. For instance: for the 'Beaver' aircraft, the time derivatives of u , v , and w still appear on the right hand side of the \dot{v} -equation, since a stability derivative to β is present in the aerodynamic model which determines the side force Y_a , see ref.[25]. We get the following expressions:

$$\begin{aligned} \dot{u} &= \frac{1}{m}(X_{gr} + X_t + X_a) - qw + rv \\ \dot{v} &= \frac{1}{m}(Y_{gr} + Y_t + Y_a) + pw - ru \\ \dot{w} &= \frac{1}{m}(Z_{gr} + Z_t + Z_a) - pv + qu \end{aligned} \quad (\text{A-25a})$$

and:

$$\begin{aligned}\dot{p} &= P_{pp}p^2 + P_{pq}pq + P_{pr}pr + P_{qq}q^2 + P_{qr}qr + P_{rr}r^2 + P_lL + P_mM + P_nN \\ \dot{q} &= Q_{pp}p^2 + Q_{pq}pq + Q_{pr}pr + Q_{qq}q^2 + Q_{qr}qr + Q_{rr}r^2 + Q_lL + Q_mM + Q_nN \\ \dot{r} &= R_{pp}p^2 + R_{pq}pq + R_{pr}pr + R_{qq}q^2 + R_{qr}qr + R_{rr}r^2 + R_lL + R_mM + R_nN\end{aligned}\quad (\text{A-25b})$$

The equations contain no assumptions of either symmetric mass distributions or aerodynamic properties and are therefore applicable to asymmetric aircraft as well as to conventional symmetric aircraft. Table A-2 gives the definition of the inertial parameters $P_{pp}, P_{pq}, \dots, R_n$ [11]. The notation is analogous to the notation that is used in ref.[6].

A.3 Using the angle of attack, sideslip angle, and total air-speed in stead of the velocity components in F_B .

A.3.1 Why use flight-path axes for the state equations?

In aerodynamics, it is more convenient to use the airspeed V (TAS), the angle of attack α , and the sideslip angle β in stead of the components of the airspeed along the body axes. For solving the equations of motion, it is possible to calculate V , α , and β from u , v , and w or vice versa. In other words, both the body-axes velocity components and the flight-path axes V , α , and β can be used as *state variables*. The question is now which set of state variables is best suited for simulation purposes.

From a physical point of view it is logical to express the aerodynamic forces and moments in terms of the flight-path variables α , β , and V . A difficulty may arise because the aerodynamic forces and moments themselves can contain

symbol	definition
$ I $	$I_x I_y I_z - 2J_{xy} J_{xz} J_{yz} - I_x J_{yz}^2 - I_y J_{xz}^2 - I_z J_{xy}^2$
I_1	$I_y I_z - J_{yz}^2$
I_2	$J_{xy} I_z + J_{yz} J_{xz}$
I_3	$J_{xy} J_{yz} + I_y J_{xz}$
I_4	$I_x I_z - J_{xz}^2$
I_5	$I_x J_{yz} + J_{xy} J_{xz}$
I_6	$I_x I_y - J_{xy}^2$

**Table A-2 (i). Definition of inertia parameters.
(See also next page).**



symbol	definition
P_l	$I_1 / I $
P_m	$I_2 / I $
P_n	$I_3 / I $
P_{pp}	$- [J_{xz} I_2 - J_{xy} I_3] / I $
P_{pq}	$[J_{xz} I_1 - J_{yz} I_2 - (I_y - I_x) I_3] / I $
P_{pr}	$- [J_{xy} I_1 + (I_x - I_z) I_2 - J_{yz} I_3] / I $
P_{qq}	$[J_{yz} I_1 - J_{xy} I_3] / I $
P_{qr}	$- [(I_z - I_y) I_1 - J_{xy} I_2 + J_{xz} I_3] / I $
P_{rr}	$- [J_{yz} I_1 - J_{xz} I_2] / I $
Q_l	$I_2 / I $
Q_m	$I_4 / I $
Q_n	$I_5 / I $
Q_{pp}	$- [J_{xz} I_4 - J_{xy} I_5] / I $
Q_{pq}	$[J_{xz} I_2 - J_{yz} I_4 - (I_y - I_x) I_5] / I $
Q_{pr}	$- [J_{xy} I_2 + (I_x - I_z) I_4 - J_{yz} I_5] / I $
Q_{qq}	$[J_{yz} I_2 - J_{xy} I_5] / I $
Q_{qr}	$- [(I_z - I_y) I_2 - J_{xy} I_4 + J_{xz} I_5] / I $
Q_{rr}	$- [J_{yz} I_2 - J_{xz} I_4] / I $
R_l	$I_3 / I $
R_m	$I_5 / I $
R_n	$I_6 / I $
R_{pp}	$- [J_{xz} I_5 - J_{xy} I_6] / I $
R_{pq}	$[J_{xz} I_3 - J_{yz} I_5 - (I_y - I_x) I_6] / I $
R_{pr}	$- [J_{xy} I_3 + (I_x - I_z) I_5 - J_{yz} I_6] / I $
R_{qq}	$[J_{yz} I_3 - J_{xy} I_6] / I $
R_{qr}	$- [(I_z - I_y) I_3 - J_{xy} I_5 + J_{xz} I_6] / I $
R_{rr}	$- [J_{yz} I_3 - J_{xz} I_5] / I $

Table A-2 (ii). Definition of inertia parameters. See previous page for definition of $|I|, I_1, \dots, I_6$.

alpha-dot or beta-dot dependence. However, $\dot{\alpha}$ and $\dot{\beta}$ are not available until after the force equations have been evaluated. It may be impossible to arrange the body-axes state equations in a form that allows an explicit solution for the state derivatives [29].

Provided that the aerodynamic forces are linearly dependent of $\dot{\alpha}$ and $\dot{\beta}$, explicit nonlinear state equations can easily be found if the differential equations for the velocity components are written out in terms of the flight-path axes instead of the body-axes. This facilitates computations, and thus reduces the possibility of making errors. In ref.[25], it is demonstrated for the 'Beaver' aircraft how complicated the equations of motion may become as a result of using body axes velocities in the state equations, but flight-path axes variables in the aerodynamic model.

Another reason for using flight-path axes is given in ref.[13]. For agile aircraft, with an upper limit of the pitch rate q of about 2 rad/s, flying at very high velocities (i.e. $V_{max} = 600$ m/s), the term $u \cdot q$ may become as large as 120 g's! On the other hand, F_z / m , the normal acceleration due to the external force (primarily gravity and aerodynamic lift) has an upper limit of several g's. Hence, artificial accelerations which are much greater than the actual accelerations are introduced, because of high rotation rates of the body-axes. In practice, this means less favourable computer scaling and hence poorer accuracy for a given computer precision if equations (A-25) are used for simulation.

A.3.2 Transformations of forces and velocities from body-axes to flight-path axes.

For simulation purposes, the equations for u , v , and w will be now transferred to equations for V , α , and β , using a derivation which closely follows ref.[11]. The following set of equations give the required transformations:

$$\begin{aligned} u &= V \cos \alpha \cos \beta \\ v &= V \sin \beta \\ w &= V \sin \alpha \cos \beta \end{aligned} \tag{A-26}$$

Hence:

$$V = \sqrt{u^2 + v^2 + w^2}$$

$$\begin{aligned} \alpha &= \arctan \left(\frac{w}{u} \right) \\ \beta &= \arctan \left(\frac{v}{\sqrt{u^2 + w^2}} \right) \end{aligned} \tag{A-27}$$



Often, the *aerodynamic* forces are expressed in lift, drag, and sideforce, in stead of the force components about the X_B , Y_B , and Z_B -axes, because of the physical origin of these forces. Forces caused by the engine(s) are generally expressed in the body-fixed reference frame F_B , and gravity forces are transferred from earth-fixed to body axes. The components of the aerodynamic forces along the body axes can be expressed in terms of lift \bar{L} , drag \bar{D} , and sideforce \bar{Y} (see figure A-1):

$$\begin{aligned} X_a &= -\bar{D} \cos \alpha + \bar{L} \sin \alpha \\ Y_a &= \bar{Y} \\ Z_a &= -\bar{D} \sin \alpha - \bar{L} \cos \alpha \end{aligned} \quad (\text{A-28})$$

The forces due to gravity will be transferred from Earth-fixed reference axes to the body axes of the aircraft in section A.4.

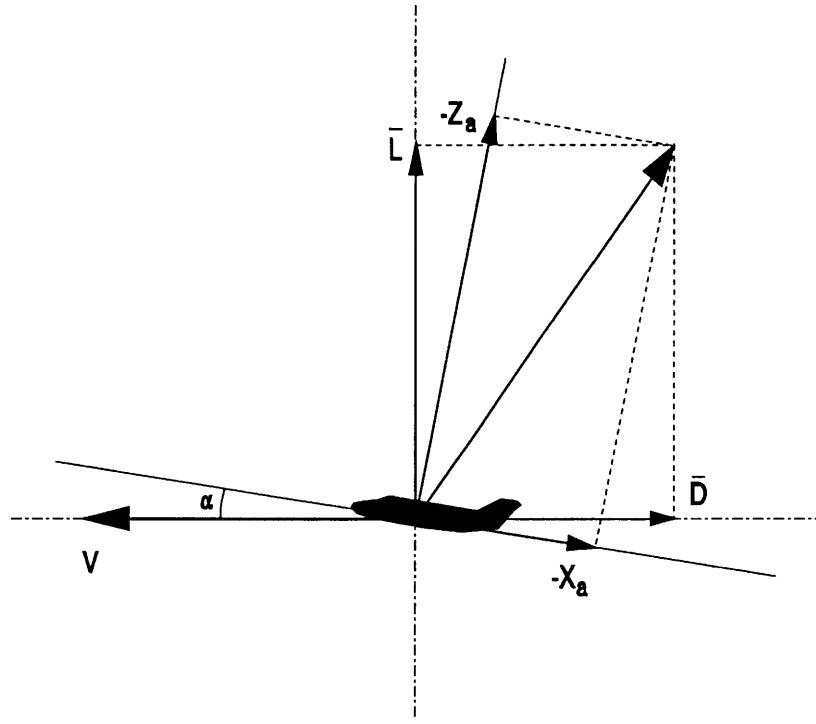


Figure A-1. Relationship between aerodynamic forces in flight-path and body axes reference frames.

A.3.3 Derivation of the \dot{V} -equation.

From equation (A-27), it can be deduced that:

$$\dot{V} = \frac{u \dot{u} + v \dot{v} + w \dot{w}}{V} \quad (\text{A-29})$$

Substituting the definitions (A-26) for u , v , and w , and cancelling terms yields:

$$\dot{V} = \dot{u} \cos \alpha \cos \beta + \dot{v} \sin \beta + \dot{w} \sin \alpha \cos \beta$$

If we now substitute the equations for \dot{u} , \dot{v} , and \dot{w} in (A-25a), the terms involving the vehicle rotational rates p , q , and r are identically zero, and the resulting equation becomes:

$$\begin{aligned} \dot{V} = \frac{1}{m} [& (X_{gr} + X_t + X_a) \cos \alpha \cos \beta + (Y_{gr} + Y_t + Y_a) \sin \beta + \\ & + (Z_{gr} + Z_t + Z_a) \sin \alpha \cos \beta] \end{aligned} \quad (\text{A-31})$$

and using the lift and drag:

$$\begin{aligned} \dot{V} = \frac{1}{m} [& -\bar{D} \cos \beta + \bar{Y} \sin \beta + (X_{gr} + X_t) \cos \alpha \cos \beta + \\ & + (Y_{gr} + Y_t) \sin \beta + (Z_{gr} + Z_t) \sin \alpha \cos \beta] \end{aligned} \quad (\text{A-32})$$

A.3.4 Derivation of the $\dot{\alpha}$ -equation.

Using (A-27), the time derivative of α can be written as:

$$\dot{\alpha} = \frac{u \dot{w} - \dot{u} w}{u^2 + w^2} \quad (\text{A-33})$$

(see ref.[25]). Substituting for u and w and using:

$$u^2 + v^2 = V^2 - v^2 = V^2(1 - \sin^2 \beta) = V^2 \cos^2 \beta \quad (\text{A-34})$$

yields:

$$\dot{\alpha} = \frac{\dot{w} \cos \alpha - \dot{u} \sin \alpha}{V \cos \beta} \quad (\text{A-35})$$

Substituting for \dot{w} and \dot{u} and rewriting terms now yields:

$$\begin{aligned} \dot{\alpha} = \frac{1}{V \cos \beta} \{ & \frac{1}{m} [(Z_{gr} + Z_t + Z_a) \cos \alpha - (X_{gr} + X_t + X_a) \sin \alpha] + \\ & - p v \cos \alpha + q u \cos \alpha + q w \sin \alpha - r v \sin \alpha \} \end{aligned} \quad (\text{A-36})$$



Using equations (A-26) for u , v , and w , we find:

$$\dot{\alpha} = \frac{1}{V \cos \beta} \left\{ \frac{1}{m} \left[(Z_{gr} + Z_t + Z_a) \cos \alpha - (X_{gr} + X_t + X_a) \sin \alpha \right] \right\} + q - (p \cos \alpha + r \sin \alpha) \tan \beta \quad (\text{A-37})$$

The aerodynamic forces can again be written in terms of aerodynamic lift and drag, yielding:

$$\dot{\alpha} = \frac{1}{V \cos \beta} \left\{ \frac{1}{m} \left[-\bar{L} + (Z_{gr} + Z_t) \cos \alpha - (X_{gr} + X_t) \sin \alpha \right] \right\} + q - (p \cos \alpha + r \sin \alpha) \tan \beta \quad (\text{A-38})$$

A.3.5 Derivation of the $\dot{\beta}$ -equation.

From (A-27) we find:

$$\dot{\beta} = \frac{\dot{v}(u^2 + w^2) - v(u\dot{u} + w\dot{w})}{V^2 \sqrt{u^2 + w^2}} \quad (\text{A-39})$$

(see ref.[25]). If we substitute:

$$\begin{aligned} u^2 + w^2 &= V^2 \cos^2 \beta \\ uv &= V^2 \sin \beta \cos \beta \cos \alpha \\ vw &= V^2 \sin \beta \cos \beta \sin \alpha \end{aligned} \quad (\text{A-40})$$

which can be derived from equations (A-26), the following expression for $\dot{\beta}$ can be found:

$$\dot{\beta} = \frac{1}{V} (-\dot{u} \cos \alpha \sin \beta + \dot{v} \cos \beta - \dot{w} \sin \alpha \sin \beta) \quad (\text{A-41})$$

Substituting for \dot{u} , \dot{v} , and \dot{w} yields:

$$\begin{aligned} \dot{\beta} = \frac{1}{V} \left\{ \frac{1}{m} \left[- (X_{gr} + X_t + X_a) \cos \alpha \sin \beta + (Y_{gr} + Y_t + Y_a) \cos \beta + \right. \right. \\ \left. \left. - (Z_{gr} + Z_t + Z_a) \sin \alpha \right] + qw \cos \alpha \sin \beta - rv \cos \alpha \sin \beta + \right. \\ \left. + pw \cos \beta - ru \cos \beta + pv \sin \alpha \sin \beta - qu \sin \alpha \sin \beta \right\} \end{aligned} \quad (\text{A-42})$$

If we substitute equations (A-26), many terms can be cancelled, and we find:

$$\dot{\beta} = \frac{1}{V} \left\{ \frac{1}{m} \left[- (X_{gr} + X_t + X_a) \cos \alpha \sin \beta + (Y_{gr} + Y_t + Y_a) \cos \beta + \right. \right. \\ \left. \left. - (Z_{gr} + Z_t + Z_a) \sin \alpha \right] \right\} + p \sin \alpha - r \cos \alpha \quad (\text{A-43})$$

which yields:

$$\dot{\beta} = \frac{1}{V} \left\{ \frac{1}{m} \left[\bar{D} \sin \beta + \bar{Y} \cos \beta - (X_{gr} + X_t) \cos \alpha \sin \beta + \right. \right. \\ \left. \left. + (Y_{gr} + Y_t) \cos \beta - (Z_{gr} + Z_t) \sin \alpha \sin \beta \right] \right\} + p \sin \alpha - r \cos \alpha \quad (\text{A-44})$$

A.4 Equations of motion in a non-steady atmosphere.

The equations of motion (A-25) are valid only if the components u , v , and w of the translational velocity $V_{c.g.}$ of the aircraft's centre of gravity are measured relative to a non-rotating system of reference axes having a constant translational speed in inertial space. Under the assumptions 3 and 4 of section A.2.4, it is possible to select a reference frame that is fixed to the surrounding atmosphere, as long as the wind velocity vector \mathbf{V}_w is constant. In that case, the components u , v , and w of the velocity vector $\mathbf{V}_{c.g.} = \mathbf{V}_a$ express the aircraft's velocity with respect to the surrounding atmosphere.

However, if the wind velocity vector \mathbf{V}_w is not constant during the time interval over which the aircraft's motions are studied, it is not possible to fix the frame of reference to the surrounding atmosphere. This situation can, for instance, arise during approach and landing of an aircraft, due to variation of wind velocity with altitude. Again using assumptions 3 and 4 of section A.2.4, the most obvious choice of the reference frame in this case turns out to be the earth-fixed reference frame F_E [16].

In the subsequent part of this section, the symbols u , v , and w will be reserved for the velocity components of the vector \mathbf{V}_a , i.e., the velocity of the aircraft's c.g. with respect to the surrounding atmosphere. The vector $\mathbf{V}_{c.g.} = \mathbf{V}_e$ is used within the equations of motion. \mathbf{V}_e denotes the velocity of the aircraft's c.g. with respect to the earth-fixed reference frame. The wind velocity \mathbf{V}_w is also measured with respect to F_E . Hence:

$$\mathbf{V}_e = \mathbf{V}_a + \mathbf{V}_w \quad (\text{A-45})$$

or:

$$\begin{aligned} u_e &= u + u_w \\ v_e &= v + v_w \\ w_e &= w + w_w \end{aligned} \quad (\text{A-46})$$



where u_w , v_w , and w_w are the components of \mathbf{V}_w along the body axes of the aircraft and u_e , v_e , and w_e are the components of \mathbf{V}_e along \mathbf{F}_B . The equations of motion (A-23a) now become:

$$\begin{aligned} F_x &= m(\dot{u}_e + q w_e - r v_e) \\ F_y &= m(\dot{v}_e + r u_e - p w_e) \\ F_z &= m(\dot{w}_e + p v_e - q u_e) \end{aligned} \quad (\text{A-47})$$

We want to express the equations of motion in terms of V_a , α , and β in the sense of equations (A-32), (A-38), and (A-44), because these three quantities are required for calculating the aerodynamic forces and moments. Expressions (A-26) and (A-27) are still valid if V is set equal to the velocity with respect to the atmosphere, V_a . Rewriting (A-47) yields:

$$\begin{aligned} \dot{u} &= \frac{F_x}{m} - q w_e + r v_e - \dot{u}_w \\ \dot{v} &= \frac{F_y}{m} + p w_e - r u_e - \dot{v}_w \\ \dot{w} &= \frac{F_z}{m} + p v_e + q u_e - \dot{w}_w \end{aligned} \quad (\text{A-48})$$

In a manner analogous to sections A.3.3, A.3.4, and A.3.5, expressions for the time derivatives of V_a , α , and β can be found:

$$\begin{aligned} \dot{V}_a &= \frac{1}{m} [-\bar{D} \cos \beta + \bar{Y} \sin \beta + (X_{gr} + X_t) \cos \alpha \cos \beta + \\ &\quad + (Y_{gr} + Y_t) \sin \beta + (Z_{gr} + Z_t) \sin \alpha \cos \beta] + \\ &\quad - (q w_w - r v_w + \dot{u}_w) \cos \alpha \cos \beta + \\ &\quad + (p w_w - r u_w - \dot{v}_w) \sin \beta - (p v_w - q u_w + \dot{w}_w) \sin \alpha \cos \beta \end{aligned} \quad (\text{A-49})$$

$$\begin{aligned} \dot{\alpha} &= \frac{1}{V_a \cos \beta} \left\{ \frac{1}{m} \left[-\bar{L} + (Z_{gr} + Z_t) \cos \alpha - (X_{gr} + X_t) \sin \alpha \right] + \right. \\ &\quad \left. - (p v_w - q u_w + \dot{w}_w) \cos \alpha + (q w_w - r v_w + \dot{u}_w) \sin \alpha \right\} + \\ &\quad + q - (p \cos \alpha + r \sin \alpha) \tan \beta \end{aligned} \quad (\text{A-50})$$

$$\begin{aligned} \dot{\beta} &= \frac{1}{V_a} \left\{ \frac{1}{m} \left[\bar{D} \sin \beta + \bar{Y} \cos \beta - (X_{gr} + X_t) \cos \alpha \sin \beta + \right. \right. \\ &\quad \left. + (Y_{gr} + Y_t) \cos \beta - (Z_{gr} + Z_t) \sin \alpha \sin \beta \right] + \\ &\quad + (q w_w - r v_w + \dot{u}_w) \cos \alpha \sin \beta + (p w_w - r u_w - \dot{v}_w) \cos \beta + \\ &\quad \left. + (p v_w - q u_w + \dot{w}_w) \sin \alpha \sin \beta \right\} + p \sin \alpha - r \cos \alpha \end{aligned} \quad (\text{A-51})$$

The differences between these expressions and (A-32), (A-38), and (A-44) can be modelled by adding a 'wind' component to the forces along the aircraft's body-axes. The total forces and moments along the body-axes now become:

$$\begin{aligned} F_x &= X_{gr} + X_t + X_a + X_w \\ F_y &= Y_{gr} + Y_t + Y_a + Y_w \\ F_z &= Z_{gr} + Z_t + Z_a + Z_w \end{aligned} \quad (\text{A-52})$$

where X_w , Y_w , and Z_w represent corrections to the body-axes forces due to nonsteady atmosphere, according to the following equations:

$$\begin{aligned} X_w &= -m(\dot{u}_w + q w_w - r v_w) \\ Y_w &= -m(\dot{v}_w - p w_w + r u_w) \\ Z_w &= -m(\dot{w}_w + p v_w - q u_w) \end{aligned} \quad (\text{A-53})$$

Due to these additional force components, the responses of V , α , and β in non-steady atmosphere will be different from the responses in steady atmosphere. The aerodynamic forces and moments can be expressed as functions of V , α , β , etc., which implies that these forces and moments will also differ from the results that would have been obtained in steady atmosphere.

If the wind velocity or direction changes very fast, for instance in atmospheric turbulence, the aerodynamic model sometimes needs to be extended with terms, which bring into account aerodynamic lags, such as the *gust penetration effect*, which is caused by the finite dimensions of the aircraft. This effect is described and modelled in ref.[24], but it will be neglected in this report¹⁾.

Appendix B outlines some common methods to model atmospheric turbulence.

The corrections (A-53) again contain terms, involving the vehicle rotational rates p , q , and r . Unlike the term $u \cdot q$, the magnitude of $u_w \cdot q$ will not become large with respect to the normal acceleration F_z/m , because the maximum windspeed under which it is allowed to fly is limited. Hence, expressions (A-53) can be used without any problems regarding computer precision. Often, windspeeds are expressed with respect to the earth-fixed reference frame, which means that a transformation of axes from F_E to F_B is necessary.

¹⁾ In ref.[24] the responses of an aircraft to atmospheric turbulence are modelled by adding 'gust corrections' to u , α , and β , which means that α and β themselves are not measured relatively to the surrounding atmosphere as within this report. Furthermore, the most recent version of ref.[24] uses other expressions to model the gust penetration effect than earlier versions of these Lecture Notes, but it is not clear which version is the best. To model the gust penetration effect, knowledge about contributions of certain specific parts of the airframe to the stability derivatives is required, but this knowledge is not readily available. It is possible to approximate these contributions, but this introduces errors. Also, care has to be taken if a *nonlinear* aerodynamic model is used, because the expressions are usually derived assuming linearity. For these reasons, this report will not model the responses to atmospheric turbulence in the fashion of ref.[24]. This is considered to be beyond the scope of this report. Modifications to the aerodynamic models might, however, be needed in the future.



A.5 Kinematic relations and transformation of gravity forces from earth-axes to body-axes.

A.5.1 The Euler angles.

The heading of the body reference frame with respect to the Earth-fixed reference frame is defined by the Euler angles ψ , θ , and φ , see section 0.5. In order to calculate the contribution of the aircraft weight to the external forces, it is necessary that these angles are known. The kinematic relations needed for this calculation are summarized in the following formulas (see for instance refs.[12] or [17]):

$$\begin{aligned}\dot{\varphi} &= p + q \sin \varphi \cdot \tan \theta + r \cos \varphi \cdot \tan \theta \\ \dot{\theta} &= q \cos \varphi - r \sin \varphi \\ \dot{\psi} &= q \frac{\sin \varphi}{\cos \theta} + r \frac{\cos \varphi}{\cos \theta}\end{aligned}\tag{A-54}$$

A.5.2 Gravity forces in the body-fixed reference frame.

Using the Euler angles, the following expressions for the components of the aircraft's weight $W = m \cdot g$ in the body-fixed reference frame F_B can be derived:

$$\begin{aligned}X_{gr} &= -W \cdot \sin \theta \\ Y_{gr} &= W \cdot \cos \theta \sin \varphi \\ Z_{gr} &= W \cdot \cos \theta \cos \varphi\end{aligned}\tag{A-55}$$

A.5.3 The position of the aircraft.

The position of the aircraft with respect to the earth bounded reference frame is given by the coordinates x_e , y_e , and z_e , defined by the following equations:

$$\begin{aligned}\dot{x}_e &= \{u_e \cos \theta + (v_e \sin \varphi + w_e \cos \varphi) \sin \theta\} \cos \psi - (v_e \cos \varphi - w_e \sin \varphi) \sin \psi \\ \dot{y}_e &= \{u_e \cos \theta + (v_e \sin \varphi + w_e \cos \varphi) \sin \theta\} \sin \psi + (v_e \cos \varphi - w_e \sin \varphi) \cos \psi \\ \dot{z}_e &= -u_e \sin \theta + (v_e \sin \varphi + w_e \cos \varphi) \cos \theta\end{aligned}\tag{A-56}$$

which results after a transformation of axes from F_B to F_E :

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \end{bmatrix} = T_{BE} \cdot \begin{bmatrix} u_e \\ v_e \\ w_e \end{bmatrix}\tag{A-57}$$

where $T_{BE} = T_{BV} = T_{VB}^{-1}$ is the transformation matrix from F_B to F_E (see definition in section 0.5.2). Usually, the aircraft's altitude H is used in stead of the coordinate z_e . The following relation holds:

$$\dot{H} = -z_e \quad (\text{A-58})$$

A.6 Summary of the state equations.

The equations of motion can thus be written as a set of first order ordinary differential equations, which is particulary useful for such purposes as simulation, linearization and steady-state trim [29]. The following variables will be used as states: V , α , β , p , q , r , ψ , θ , and φ . The set x_e , y_e , and H is added to this list, but contrary to the Euler angles ψ , θ , and φ , these variables are not necessary to solve the equations for the other state variables. In this appendix, the following state equations have been derived:

$$\begin{aligned} \dot{V}_a &= \frac{1}{m} [-\bar{D} \cos \beta + \bar{Y} \sin \beta + (X_{gr} + X_t + X_w) \cos \alpha \cos \beta + \\ &\quad + (Y_{gr} + Y_t + Y_w) \sin \beta + (Z_{gr} + Z_t + Z_w) \sin \alpha \cos \beta] \end{aligned} \quad (\text{A-59a})$$

$$\begin{aligned} \dot{\alpha} &= \frac{1}{V_a \cos \beta} \left\{ \frac{1}{m} \left[-\bar{L} - (X_{gr} + X_t + X_w) \cos \alpha + (Z_{gr} + Z_t + Z_w) \sin \alpha \right] \right\} + \\ &\quad + q - (p \cos \alpha + r \sin \alpha) \tan \beta \end{aligned} \quad (\text{A-59b})$$

$$\begin{aligned} \dot{\beta} &= \frac{1}{V_a} \left\{ \frac{1}{m} \left[\bar{D} \sin \beta + \bar{Y} \cos \beta - (X_{gr} + X_t + X_w) \cos \alpha \sin \beta + \right. \right. \\ &\quad \left. \left. + (Y_{gr} + Y_t + Y_w) \cos \beta - (Z_{gr} + Z_t + Z_w) \sin \alpha \sin \beta \right] \right\} + p \sin \alpha - r \cos \alpha \end{aligned} \quad (\text{A-59c})$$

$$\begin{aligned} \dot{p} &= P_{pp} p^2 + P_{pq} p q + P_{pr} p r + P_{qq} q^2 + P_{qr} q r + P_{rr} r^2 + P_l L + P_m M + P_n N \\ \dot{q} &= Q_{pp} p^2 + Q_{pq} p q + Q_{pr} p r + Q_{qq} q^2 + Q_{qr} q r + Q_{rr} r^2 + Q_l L + Q_m M + Q_n N \\ \dot{r} &= R_{pp} p^2 + R_{pq} p q + R_{pr} p r + R_{qq} q^2 + R_{qr} q r + R_{rr} r^2 + R_l L + R_m M + R_n N \end{aligned} \quad (\text{A-60})$$

$$\begin{aligned} \dot{\psi} &= p + q \sin \varphi \cdot \tan \theta + r \cos \varphi \cdot \tan \theta \\ \dot{\theta} &= q \cos \varphi - r \sin \varphi \\ \dot{\varphi} &= q \frac{\sin \varphi}{\cos \theta} + r \frac{\cos \varphi}{\cos \theta} \end{aligned} \quad (\text{A-61})$$



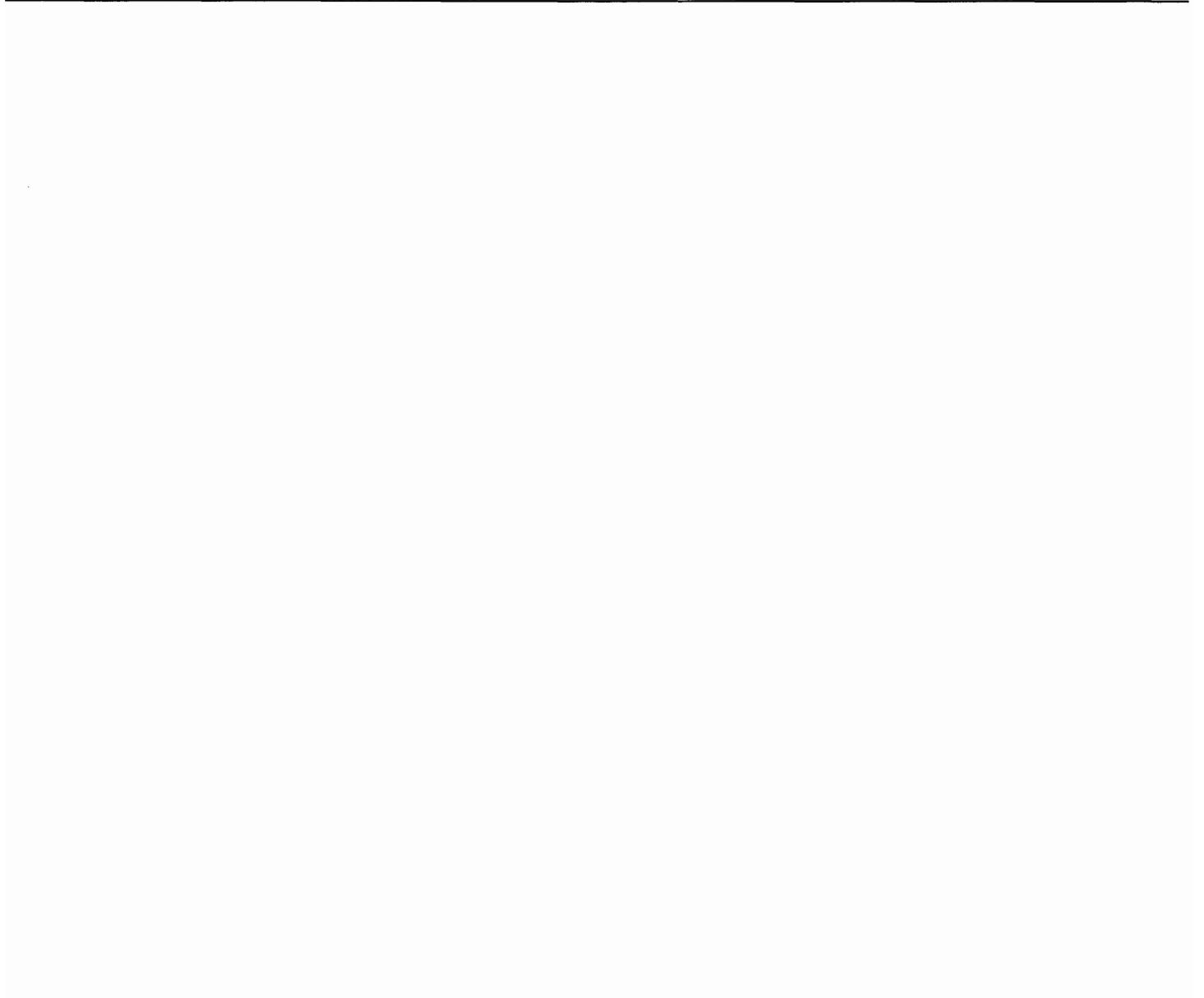
$$\begin{aligned}
 \dot{x}_e &= \{u_e \cos \theta + (v_e \sin \varphi + w_e \cos \varphi) \sin \theta\} \cos \psi - (v_e \cos \varphi - w_e \sin \varphi) \sin \psi \\
 \dot{y}_e &= \{u_e \cos \theta + (v_e \sin \varphi + w_e \cos \varphi) \sin \theta\} \sin \psi + (v_e \cos \varphi - w_e \sin \varphi) \cos \psi \\
 \dot{z}_e &= -u_e \sin \theta + (v_e \sin \varphi + w_e \cos \varphi) \cos \theta
 \end{aligned} \tag{A-62}$$

Equations (A-53) and (A-54) express the forces from gravity and winds as functions of states and (wind-) inputs, respectively. The other forces and moments can also be written in terms of states, input signals, and sometimes state-derivatives, but contrary to the gravity and wind terms, these forces and moments depend on the aircraft considered. For this reason no expressions for aerodynamic and engine forces and moments are given here.

A.7 Conclusions.

In this appendix, the equations of motion for steady *and* nonsteady atmosphere have been derived. The nonlinear state-space formulation that was used is particularly suited for simulation, linearization, and for trimming nonlinear aircraft models for steady-state flight. Aircraft velocity components along the body-axes have been replaced by the true airspeed, angle of attack, and sideslip angle, because these variables are commonly used within aerodynamic models. The equations in nonsteady atmosphere are very general, but it might be necessary to enhance the aerodynamic model with additional contributions to account for rapid changes of windspeeds, i.e. atmospheric turbulence, which may introduce aerodynamic lags due to the finite dimensions of the aircraft.





Appendix B. Mathematical models of wind and atmospheric turbulence.

B.1 Introduction.

For the assessment of the flying qualities of an aircraft, its responses to external disturbances such as wind and atmospheric turbulence should be considered. It is therefore necessary to develop mathematical models of these phenomenae. This appendix will give a short description of wind(-shear) and atmospheric turbulence. See refs.[1] and [24] for more details.

B.2 Wind profiles and wind shear.

In this section, a distinction between wind and atmospheric turbulence will be made. Wind is defined here as the mean or steady-state velocity of the atmosphere with respect to the earth at a given position. Usually, the mean wind is measured over a certain time interval of a few minutes. The remaining fluctuating part of the wind velocity is then the contribution of atmospheric turbulence.

The velocity and direction of the mean wind with respect to the ground usually is not constant along the flightpath. This variation of mean wind velocity and direction along the flightpath is called wind shear¹⁾. The influence of wind shear is particulary important during approach and landing, or take-off and climb. An idealized profile of the mean wind as a function of altitude is shown in figure B-1. More extreme wind profiles in lower atmosphere have been measured [1] and have sometimes resulted in accidents. A very serious type of wind shear is encountered in *microbursts*, where large nose winds are followed by large tail winds in just a couple of seconds.

In ref.[1], a couple of idealized wind profiles for different atmospheric conditions are presented. Since this report limits itself to the use of the US Standard Atmosphere (see for instance ref.[27]), the standard temperature lapse rate $\lambda = \frac{dT}{dh} = -0.0065$ will be used. For this lapse rate, the wind profile can be represented by the following expression [1]:

$$V_w = V_{w_{9.15}} \frac{h^{0.2545} - 0.4097}{1.3470} \quad (0 < h < 300\text{ m}) \quad (\text{B-1})$$

$$V_w = 0.2545 \quad (h \geq 300\text{ m})$$

$V_{w_{9.15}}$ is the wind speed at 9.15 m altitude. The wind profile that is shown in figure B-1 uses $V_{w_{9.15}} = 1$ [m/s].

¹⁾ Sometimes, the local variations of the velocity of the atmosphere with respect to the ground, *including atmospheric turbulence*, are noted as wind shear. In this report, the term will be reserved for a description of the variations of the mean wind.



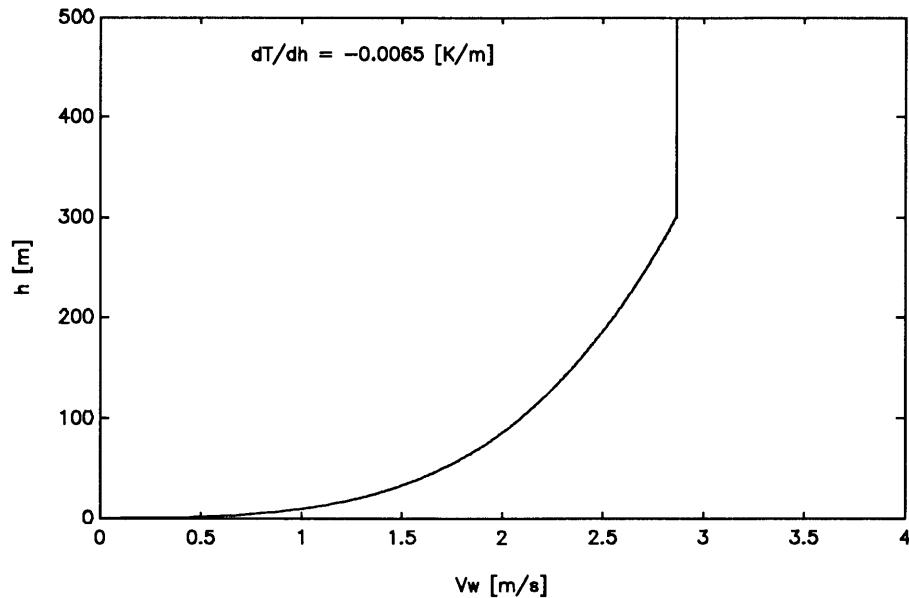


Figure B-1. Wind profile for $\lambda = -0.0065 \text{ [K/m]}$ and $V_{w_{0.15}} = 1 \text{ [m/s]}$.

This model of the earth's boundary layer wind profile is inadequate to represent the extreme wind profiles which can occur in the lower atmosphere. Thence, actual measurements of extreme wind profiles may have to be used for the assessment of automatic aircraft control systems.

The aircraft model presented in appendix A uses the wind velocities along the three body-axes. If the vertical wind velocity w_w is assumed to be zero, we have the situation that is sketched in figure B-2. The wind direction with respect to the earth is denoted as ψ_w and the wind velocity is denoted as V_w . In this figure, the usual convention, with $\psi_w = 0^\circ$ if the wind is directed *southwards* is used. For the components along the X_B and Y_B -axes, we can now write:

$$\begin{aligned} u_w &= V_w \cos(\psi_w - \pi) \cos \psi + V_w \sin(\psi_w - \pi) \sin \psi \\ v_w &= -V_w \cos(\psi_w - \pi) \sin \psi + V_w \sin(\psi_w - \pi) \cos \psi \end{aligned} \quad (\text{B-2})$$

If atmospheric turbulence is considered too, the turbulence velocities must be added to these wind velocity components.

B.3 Modelling atmospheric turbulence.

It is possible to create a mathematical model to describe atmospheric turbulence. For purposes where very high accuracy is required, it might be necessary to use actual measurements of atmospheric turbulence instead of the models. The theory of stochastic processes provides a convenient means to describe atmospheric turbulence. Auto power density spectra form the basic elements of the resulting mathematical model. A number of alternative sets of these spectra can be found in the literature. They all require the selection of intensity levels and scale lengths, before they can be applied in simulations.

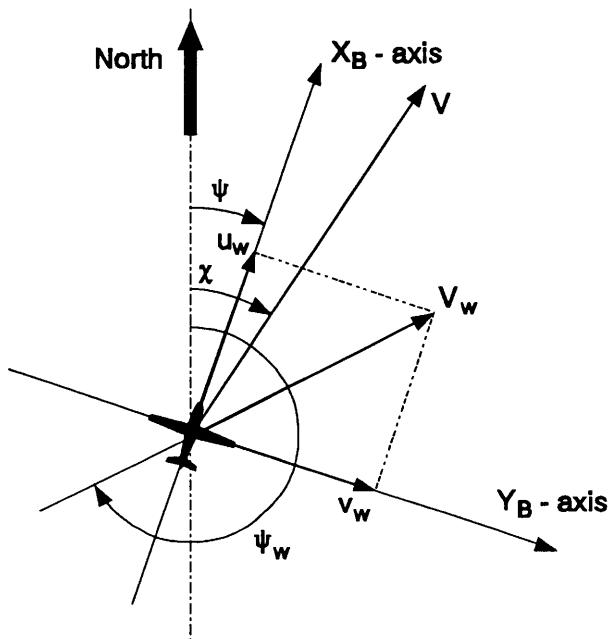


Figure B-2. Wind velocity components along the aircraft's body axes.

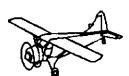
The following six assumptions concerning stochastic processes are often made when they are applied to atmospheric turbulence (ref.[1]):

- 1 - Ergodicity, which means that time averages in the process are equal to corresponding ensemble averages. This assumption makes it possible to determine all required statistical properties related to a given set of atmospheric conditions from a single time history of sufficient length.
- 2 - Stationarity, dealing with temporal properties of turbulence. If the statistical properties of a process are not affected by a shift in the time origin, this process is called stationary.
- 3 - Homogeneity, dealing with spatial properties of turbulence. The statistical properties of homogeneous turbulence are not affected by a spatial translation of the reference frame.
- 4 - Isotropy, which means that the statistical properties are not changed by a rotation or a reflection of the frame of reference. Complete isotropy implies homogeneity. Because of isotropy, the three mean-square velocity components are equal:

$$\sigma_u^2 = \sigma_v^2 = \sigma_w^2 = \sigma^2 \quad (\text{B-3})$$

and the scale lengths for the turbulence velocities along the three body axes satisfy:

$$L_u = L_v = L_w = L_g \quad (\text{B-4})$$



- 5 - Taylor's hypothesis of 'frozen atmosphere', which implies that gust velocities are functions of the position in the atmosphere only. During the short time interval in which the aircraft is under the influence of the velocities at a certain point in the atmosphere, these velocities are assumed not to change with time. This hypothesis allows spatial correlation functions and frequencies to be related to correlation functions and frequencies in the time domain. The following relations are used:

$$\Delta x = V\tau \quad (\text{B-5})$$

and:

$$\omega = \Omega V \quad (\text{B-6})$$

- Δx : distance between two points in space (in [m]),
- V : velocity of the aircraft relative to the air (in [m/s]),
- τ : time taken by the aircraft to cover the distance Δx (in [s]),
- Ω : spatial frequency (in [rad/m]),
- ω : frequency (in [rad/s]).

- 6 - Normality, meaning that the probability density function of each turbulence velocity component is Gaussian. With this assumption, the information of the covariance matrix only suffices for a total statistical description of atmospheric turbulence (ref.[24]).

Experimental data on atmospheric turbulence at low altitudes, i.e., in the boundary layer at the earth's surface, do not satisfy all these assumptions (ref.[1]). At low altitudes, due to the proximity of the ground, the assumptions of homogeneity and isotropy are not very valid. Both are affected by terrain roughness and the height above the ground. The assumption of stationarity is satisfied only over the short periods of time during which the meteorological conditions remain reasonably constant. Stationarity is also affected by the shape and roughness of the ground surface below the aircraft.

Taylor's hypothesis seems to be valid as long as the aircraft's velocity is large relative to the encountered turbulence velocities. For this reason it is somewhat doubtful that the hypothesis is fully valid when simulating the final approach and landing of S/VTOL aircraft. Finally, measurements have provided mounting evidence that atmospheric turbulence is not perfectly Gaussian. The measured departures from a normal amplitude distribution are small, but pilots seem to be quite sensitive to these effects. Actual atmospheric turbulence possess what is sometimes called a 'patchy structure' [1].

In this report, assumptions 1 to 6 will all be maintained, but the turbulence models might have to be enhanced in the future to assure a more accurate description of actual atmospheric turbulence, particularly for the simulation of aircraft approach and landing.

B.4 Power spectra of atmospheric turbulence.

B.4.1 The von Kármán spectra.

Several analytical power spectral density functions have been obtained from measured data. The von Kármán spectra functions yield spectra that seem to best fit the available theoretical and experimental data on atmospheric turbulence, particularly at higher spatial frequencies [24]. The von Kármán spectra for the three components of the turbulence velocity are:

$$S_{u_g u_g}(\Omega) = 2\sigma_u^2 L_u \frac{1}{[1 + (1.339 L_u \Omega)^2]^{5/6}} \quad (\text{B-7})$$

$$S_{v_g v_g}(\Omega) = \sigma_v^2 L_v \frac{1 + \frac{8}{3}(1.339 L_v \Omega)^2}{[1 + (1.339 L_v \Omega)^2]^{11/6}} \quad (\text{B-8})$$

$$S_{w_g w_g}(\Omega) = \sigma_w^2 L_w \frac{1 + \frac{8}{3}(1.339 L_w \Omega)^2}{[1 + (1.339 L_w \Omega)^2]^{11/6}} \quad (\text{B-9})$$

The cross spectral density functions are zero in isotropic turbulence at any point in space. Although this approximation is not very valid at low altitudes, the cross covariances, and hence, the cross power spectral densities are usually neglected (ref.[16]). The von Kármán spectrum yields an asymptotic behaviour of $S(\Omega) \sim \Omega^{-5/3}$ as Ω approaches infinity. See figure B-3.

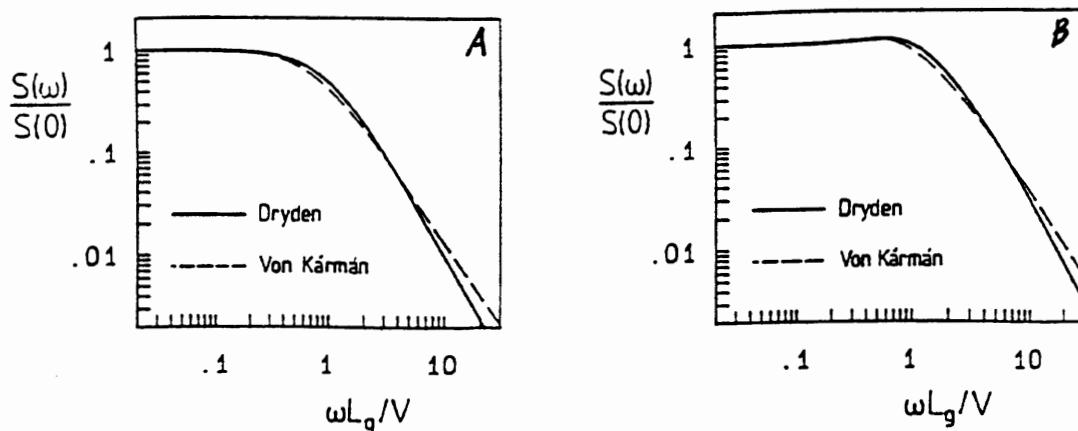


Figure B-3. Von Kármán and Dryden spectra.
A: longitudinal, B: lateral/vertical.

B.4.2 The Dryden spectra.

A major drawback of the von Kármán spectral densities is that they are not rational functions of Ω . For this reason, the Dryden power spectrum model is



often used for flight simulation purposes. The Dryden spectra are:

$$S_{u_g u_g}(\Omega) = 2\sigma_u^2 L_u \frac{1}{1 + (\Omega L_u)^2} \quad (\text{B-10})$$

$$S_{v_g v_g}(\Omega) = \sigma_v^2 L_v \frac{1 + 3(\Omega L_v)^2}{[1 + (\Omega L_v)^2]^2} \quad (\text{B-11})$$

$$S_{w_g w_g}(\Omega) = \sigma_w^2 L_w \frac{1 + 3(\Omega L_w)^2}{[1 + (\Omega L_w)^2]^2} \quad (\text{B-12})$$

The von Kármán and Dryden spectra are shown in figure B-3. The most obvious difference is the asymptotic behaviour at large values of the spatial frequency, the former having a slope of -5/3 and the latter a slope of -2.

B.5 Filter design for atmospheric turbulence.

B.5.1 Modelling atmospheric turbulence as filtered white noise.

For simulation purposes, it would be practical to model atmospheric turbulence as white noise passing through a linear, rational filter, see figure B-4. The relationship between the auto-spectral density of the output signal and the auto-spectral density of the input signal of a *linear* filter can be written as:

$$S_{yy}(\omega) = |H_{yu}(\omega)|^2 S_{uu}(\omega) \quad (\text{B-13})$$

If the input signal u is white noise, its spectral density satisfies:

$$S_{uu}(\omega) = 1 \quad (\text{B-14})$$

so for white noise relation (B-11) simplifies to:

$$S_{yy}(\omega) = |H_{yu}(\omega)|^2 \quad (\text{B-15})$$

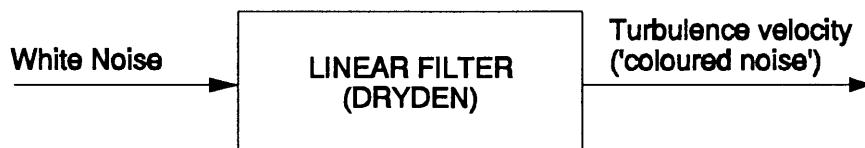


Figure B-4. Modelling atmospheric turbulence as filtered white noise.

To apply these relations, the spectral density functions of the turbulence velocities must be transformed to functions of ω , which may be done, because we have assumed that Taylor's hypothesis holds. The transformation between spatial frequency Ω and temporal frequency ω is given by:

$$S(\omega) = \frac{1}{V} S\left(\Omega = \frac{\omega}{V}\right) \quad (\text{B-16})$$

Notice the term $1/V$ arising in the spectral density function (see ref.[24])!

B.5.2 Filter design for the Dryden spectra.

The Dryden spectra were developed to approximate the von Kármán turbulence spectra by means of rational functions. This makes it possible to use relation (B-15) for the generation of the turbulence velocities from white noise. From the definitions of the Dryden spectra (B-10) to (B-11) and relation (B-16), the following expressions are found:

$$\left|H_{u_g w_1}(\omega)\right|^2 = 2\sigma_u^2 \frac{L_u}{V} \frac{1}{1 + \left(L_u \frac{\omega}{V}\right)^2} \quad (\text{B-17})$$

$$\left|H_{v_g w_2}(\omega)\right|^2 = 2\sigma_v^2 \frac{L_v}{V} \frac{1 + 3\left(L_v \frac{\omega}{V}\right)^2}{1 + \left(L_v \frac{\omega}{V}\right)^2} \quad (\text{B-18})$$

$$\left|H_{w_g w_3}(\omega)\right|^2 = 2\sigma_w^2 \frac{L_w}{V} \frac{1 + 3\left(L_w \frac{\omega}{V}\right)^2}{1 + \left(L_w \frac{\omega}{V}\right)^2} \quad (\text{B-19})$$

Solving equations (B-17), (B-18), and (B-19) yields the following candidate frequency response functions:

$$H_{u_g w_1}(\omega) = \sigma_u \sqrt{\frac{2L_u}{V}} \frac{1}{1 \pm \frac{L_u}{V} j \omega} \quad (\text{B-20})$$

$$H_{v_g w_2}(\omega) = \sigma_v \sqrt{\frac{L_v}{V}} \frac{1 \pm \sqrt{3} \frac{L_v}{V} j \omega}{\left(1 \pm \frac{L_v}{V} j \omega\right)^2} \quad (\text{B-21})$$

$$H_{w_g w_3}(\omega) = \sigma_w \sqrt{\frac{L_w}{V}} \frac{1 \pm \sqrt{3} \frac{L_w}{V} j \omega}{\left(1 \pm \frac{L_w}{V} j \omega\right)^2} \quad (\text{B-22})$$



w_1 , w_2 , and w_3 are independent white noise signals. Choosing the minus sign in the denominators would lead to instable filters and hence should be rejected for physical reasons. Choosing the minus sign in the numerators leads to a non-minimum phase system [24]. Therefore, we shall use positive signs in both the numerator and denominator.

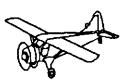
It is easy to implement these filters in a simulation package like SIMULINK. If white noise is approximated by a sequence of Gaussian distributed random numbers, it is then very easy to obtain the required turbulence velocities. These random sequences should be totally independent, which may not be obvious if the simulation software uses some initial starting value.

B.5.3 Filter design for the von Kármán spectra.

Although an exact implementation of the higher spatial frequency asymptotic behaviour of the von Kármán spectra cannot be attained by a linear filter, it is possible to closely approximate this behaviour up to any frequency of interest, by using a series of additional lead-lag networks, adding differentiating and integrating terms to the basic first order Dryden-filters. A full derivation of these approximations of the von Kármán filters can be found in ref.[1]. In the remainder of this report, only linear filters for the Dryden spectra will be used.

B.6 Conclusions.

In this appendix, some models of wind and atmospheric turbulence have been presented. Wind and atmospheric turbulence can be considered as external disturbances acting on the aircraft, which should be considered for the assessment of aircraft control systems. Atmospheric turbulence has been modelled as white noise, passing through a linear filter. During the final assessment of aircraft control systems, for instance in piloted on-line simulations, more sophisticated models or recordings of actual measured turbulence may be needed.



Appendix C. The Instrument Landing System and the VOR navigation system.

C.1 Introduction.

In this appendix, a short description of the Instrument Landing System (ILS) and the VOR navigation system will be given. First, the ILS system will be described and equations for the nominal ILS guidance signals will be derived. Then expressions for steady-state errors and ILS noise will be given, and finally, the VOR navigation system will be discussed briefly. See refs.[5], [7], and [21], for more details about navigation systems.

C.2 The Instrument Landing System.

C.2.1 Nominal ILS signals.

The Instrument Landing System (ILS) is the standard aid for non-visual approaches to landing, in use throughout the world today. Under certain circumstances, it can provide guidance data of such integrity that fully coupled approaches and landings may be achieved. The system comprises three distinct equipments:

- 1 - the localizer transmitter, which gives guidance in the horizontal plane,
- 2 - the glideslope (or glide path) transmitter, which supplies vertical guidance,
- 3 - two or three marker beacons, situated on the approach line, which give an indication of the distance to the runway to the approaching aircraft.

Only the localizer and glideslope signals will be considered here. Figures C-1 and C-2 show the basic layout of the ILS system and the approach path. The localizer signal is emitted by an antenna situated beyond the up-wind end of the runway. Operating on a frequency in the 108.0 to 112 MHz frequency band, it radiates a signal modulated by 90 and 150 Hz tones, in which the 90 Hz predominates to the left hand (to the aircraft) of the approach path, and 150 Hz to the right. Figure C-3 shows the required coverage of the localizer signals.

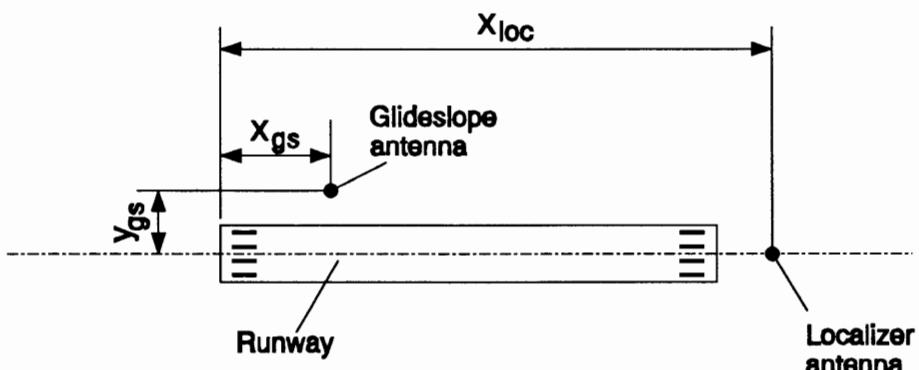


Figure C-1. Positions of the ILS system components.



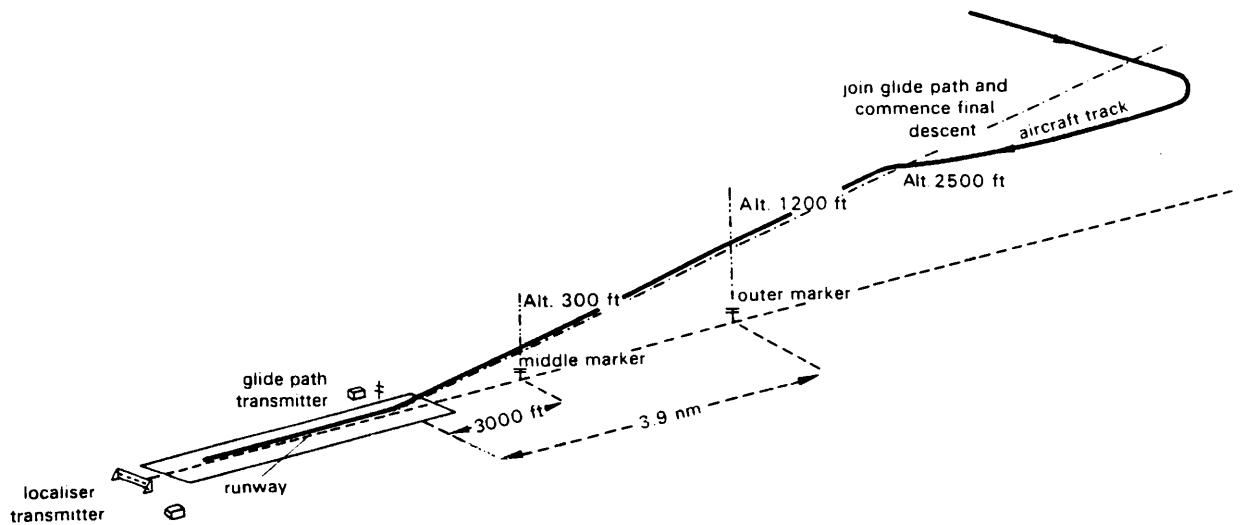


Figure C-2. Layout of the approach path.

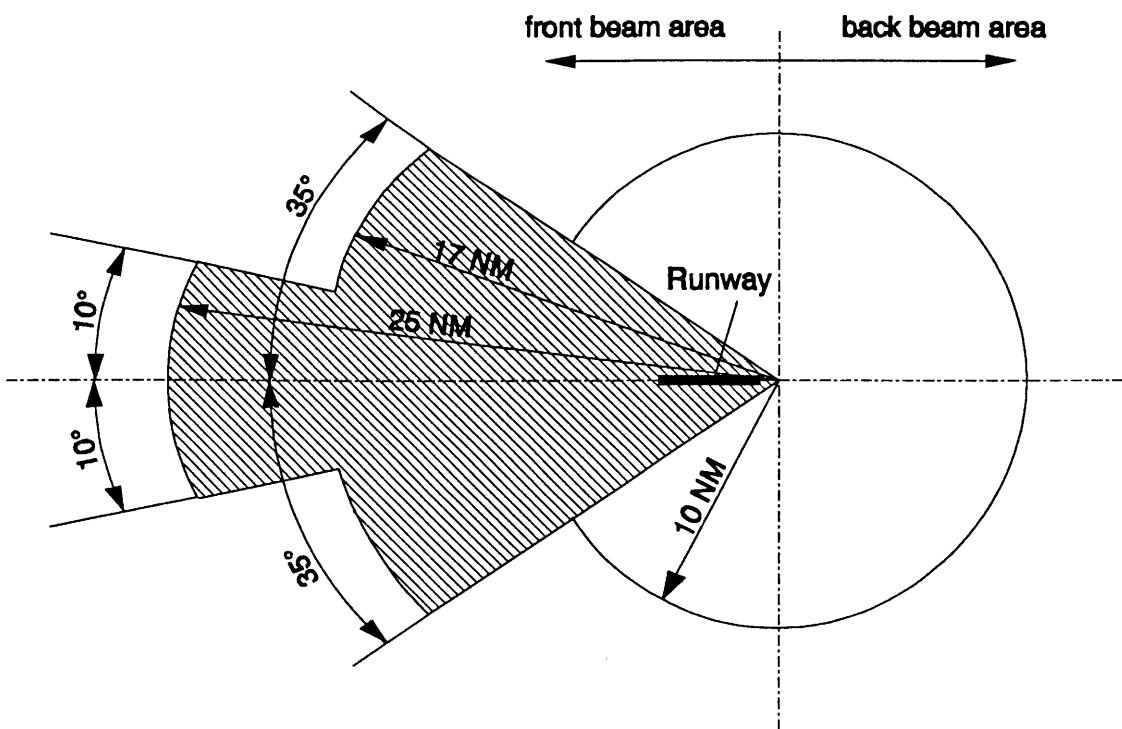


Figure C-3. Required coverage of localizer signals.

The glideslope antenna is located some 300 m beyond the runway threshold (approximately adjacent to the touch-down point) and about 120 to 150 m from the runway centerline. The frequency of the signal lies in the 328.6 to 335.0 MHz band. The signal is modulated by 90 and 150 Hz tones, in which the 90 Hz is predominant above the desired approach line, and 150 Hz beneath the glide path. Due to the position of the glideslope antenna, the intersection of the localizer reference plane and the glideslope reference cone is actually a hyperbola located a small distance above the idealized straight glide path. This is shown in figure C-4. The required coverage of the glideslope signals is shown in figure C-5.

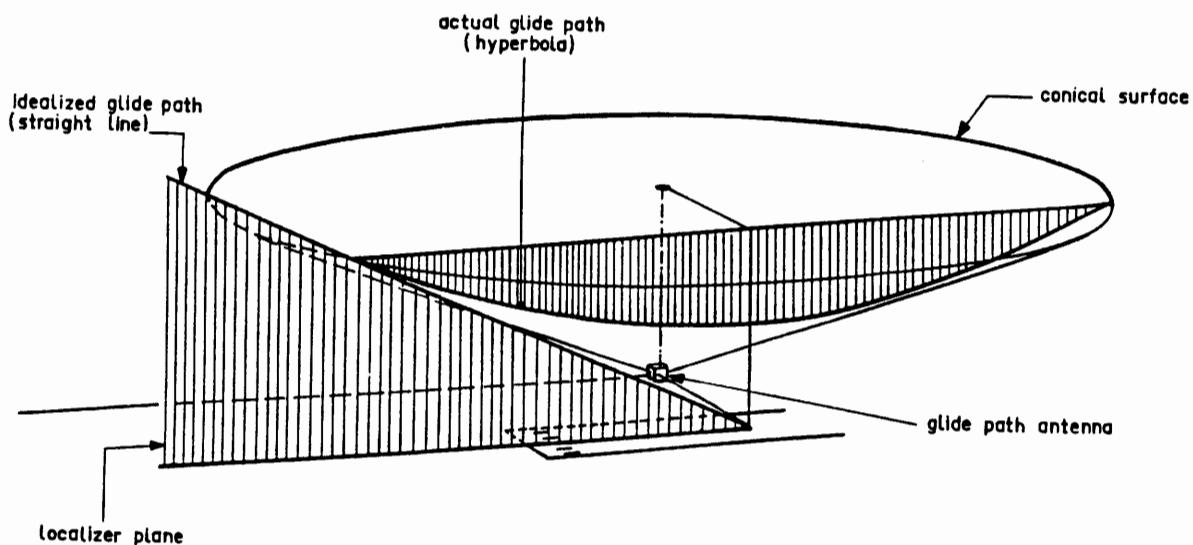


Figure C-4. Hyperbolic intersection of localizer and glideslope reference planes.

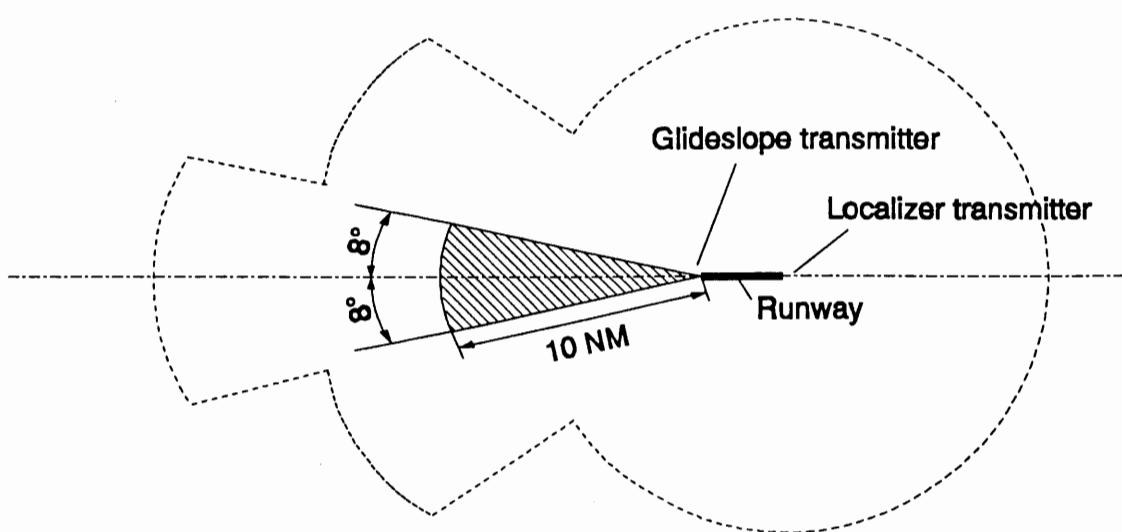
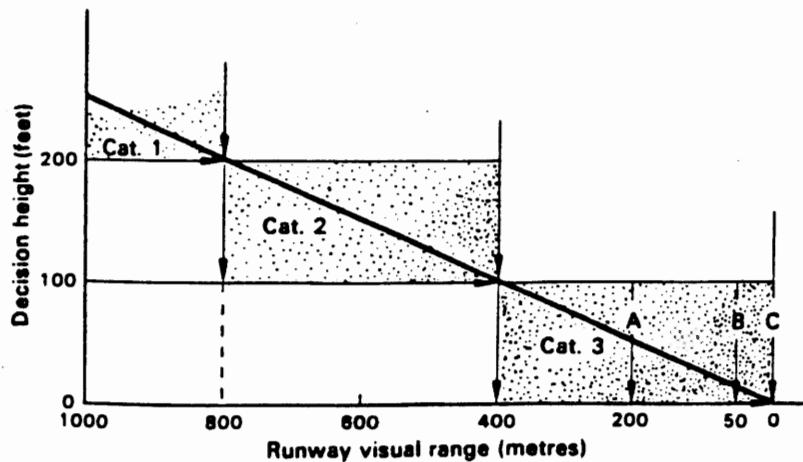


Figure C-5. Required coverage of glideslope signals compared with required coverage of localizer signals.



An ILS installation is said to belong to a certain performance category, according the meteorological conditions under which it is to be used. These conditions are summarized in figure C-6. An ILS installation of category I is intended to provide guidance down to an altitude of 200 ft, an installation of category II provides guidance down to 100 ft, and an installation of category III should provide guidance down to the runway surface, hence, its signals can be used by automatic landing systems. This means that if the aircraft is making an approach under cat. I conditions, the pilot should see the runway lights at an altitude of 200 ft. If not, the approach must be cancelled¹⁾.



- Category 1:** Operation down to minima of 200 ft decision height and runway visual range of 800 m with a high probability of approach success.
- Category 2:** Operation down to minima below 200 ft decision height and runway visual range of 800 m, and to as low as 100 ft decision height and runway visual range of 400 m with a high probability of approach success.
- Category 3A:** Operation down to and along the surface of the runway, with external visual reference during the final phase of the landing down to runway visual range minima of 200 m.
- Category 3B:** Operation to and along the surface of the runway and taxiways with visibility sufficient only for visual taxiing comparable to runway visual range value in the order of 50 m.
- Category 3C:** Operation to and along the surface of the runway and taxiways without external visual reference.

Figure C-6. ILS performance categories.

The localizer and glideslope signals are received on board the approaching aircraft. They are displayed in appropriate form to the pilot, and may be fed directly to an automatic pilot as well. The nominal ILS signals on board the aircraft are expressed in terms of the currents supplied to the pilot's cockpit instrument. The magnitude of the localizer current i_{loc} depends on the angle Γ_{loc} (in [rad]) between the localizer reference plane and a vertical plane passing through the localizer antenna, see figure C-8.

¹⁾ The altitude at which the runway lights should be visible is called the 'decision height'. Some airlines may use larger decision heights than the ones listed in figure C-6.

The localizer current is:

$$i_{loc} = S_{loc} \Gamma_{loc} \quad [\mu\text{A}] \quad (\text{C-1})$$

where S_{loc} is the sensitivity of the localizer system. According to ref.[2], S_{loc} has to satisfy the following equation:

$$S_{loc} = 1.40 \cdot x_{loc} \quad [\mu\text{A}/\text{rad}] \quad (\text{C-2})$$

where x_{loc} is the distance from the localizer antenna to the runway threshold (measured in [m]), see figure C-2.

For the simulation of ILS-approaches, a new *runway reference frame* $F_F = O_F X_F Y_F Z_F$ will be introduced. The X_F -axis is directed along the runway centerline in the direction of take-off and landing. Z_F points downwards, Y_F points rightwards as seen from an approaching aircraft. At $t = 0$, the position of the aircraft's c.g. coincides with the earth-fixed reference frame, hence: $x_e = 0$ and $y_e = 0$; $H = H_0$. The position of the origin O_F of the runway reference frame at $t = 0$, is given by the coordinates: x_{RW} and y_{RW} , measured relatively to F_E , and the altitude of the runway above sea-level, H_{RW} . From figures C-7 and C-8, the following transformations between F_E and F_F can be deduced:

$$\begin{aligned} x_f &= (x_e - x_{RW}) \cos \psi_{RW} + (y_e - y_{RW}) \sin \psi_{RW} \\ y_f &= -(x_e - x_{RW}) \sin \psi_{RW} + (y_e - y_{RW}) \cos \psi_{RW} \\ H_f &= H - H_{RW} \quad (= -z_f) \end{aligned} \quad (\text{C-3})$$

where x_f , y_f , and z_f are coordinates in the runway reference frame F_F , and ψ_{RW} is the heading of the start and landing direction of the runway ($\psi_{RW} = 0$ if it points to the north). The reference frame $\tilde{F}_E = \tilde{O}_E \tilde{X}_E \tilde{Y}_E \tilde{Z}_E$ in figure C-7 is an intermediate frame of reference, which has the same orientation as F_E , but an origin that has been moved to the projection point of O_F on the horizontal plane at sea-level. Hence: $\tilde{x}_e = x_e - x_{RW}$ and $\tilde{y}_e = y_e - y_{RW}$ (see also figure C-8).

As can be seen from figure C-8, Γ_{loc} can be calculated from the coordinates x_f , and y_f using the equations:

$$R_{loc} = \sqrt{y_f^2 + (x_{loc} - x_f)^2} \quad (\text{C-4})$$

$$d_{loc} = y_f \quad (\text{C-5})$$

and:

$$\Gamma_{loc} = \arcsin \left(\frac{d_{loc}}{R_{loc}} \right) \quad (\text{C-6})$$

Γ_{loc} and d_{loc} are positive if the aircraft flies at the right hand side of the localizer reference plane if it is headed towards the runway.



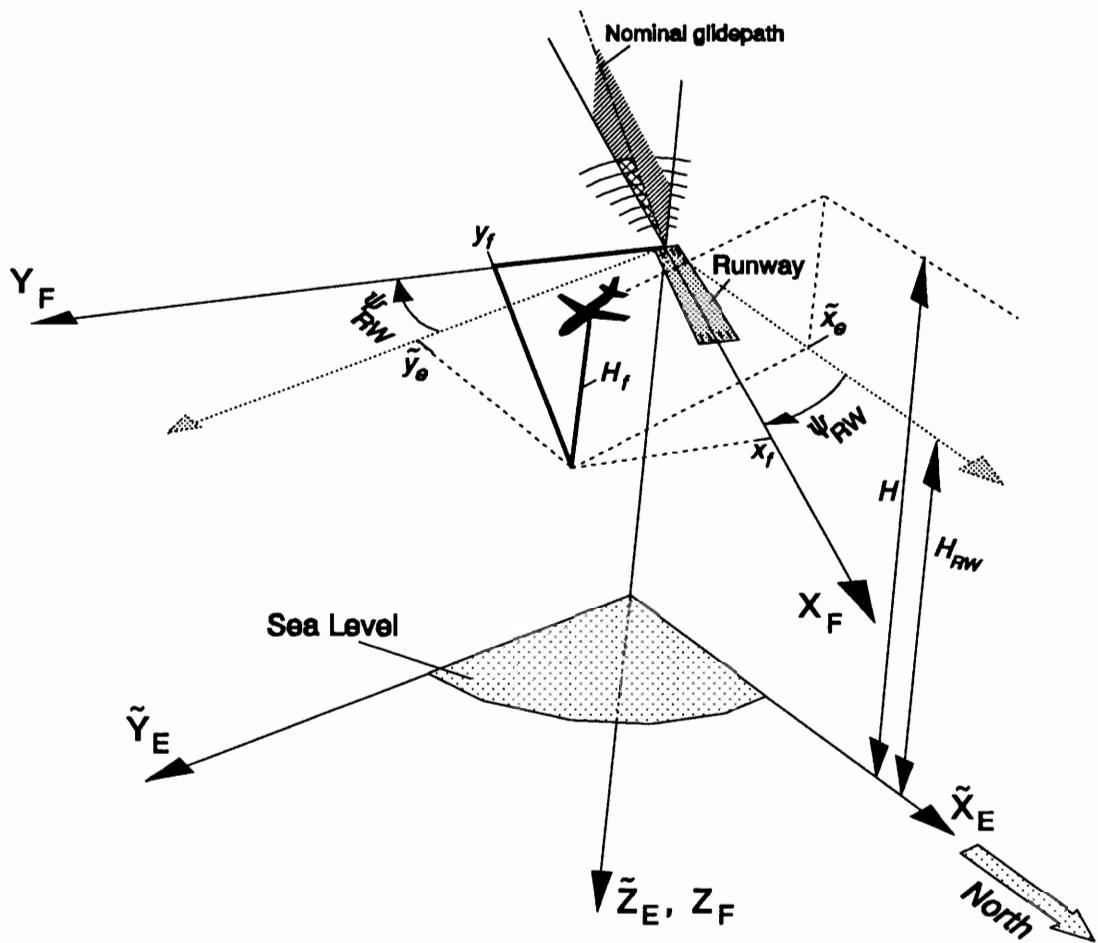


Figure C-7. Definition of earth-fixed and runway axes, used for approach simulation (here, the aircraft turns right after a missed approach).

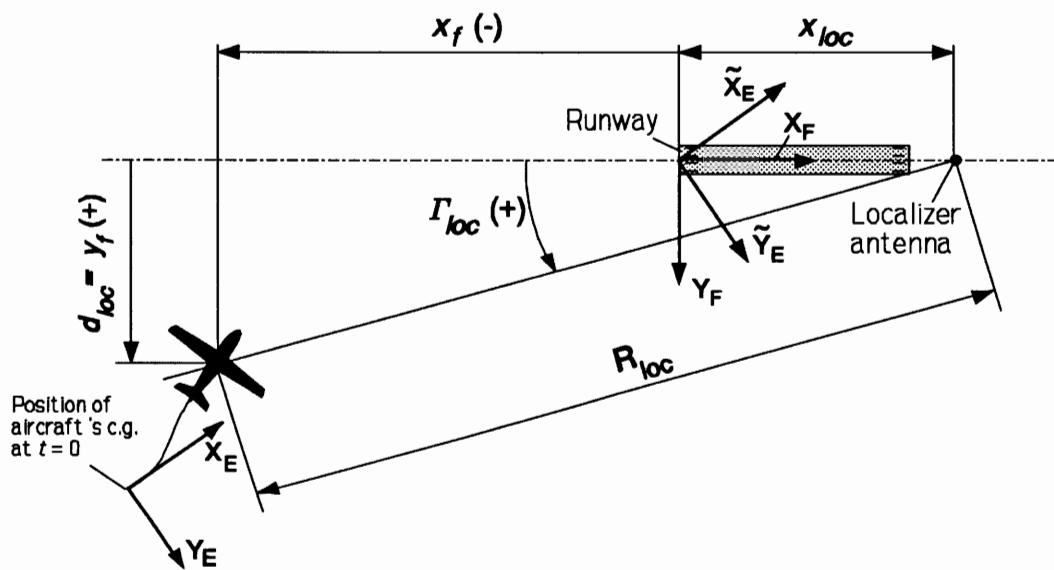


Figure C-8. Localizer geometry and definition of \tilde{X}_E , \tilde{Y}_E , X_F , and Y_F .

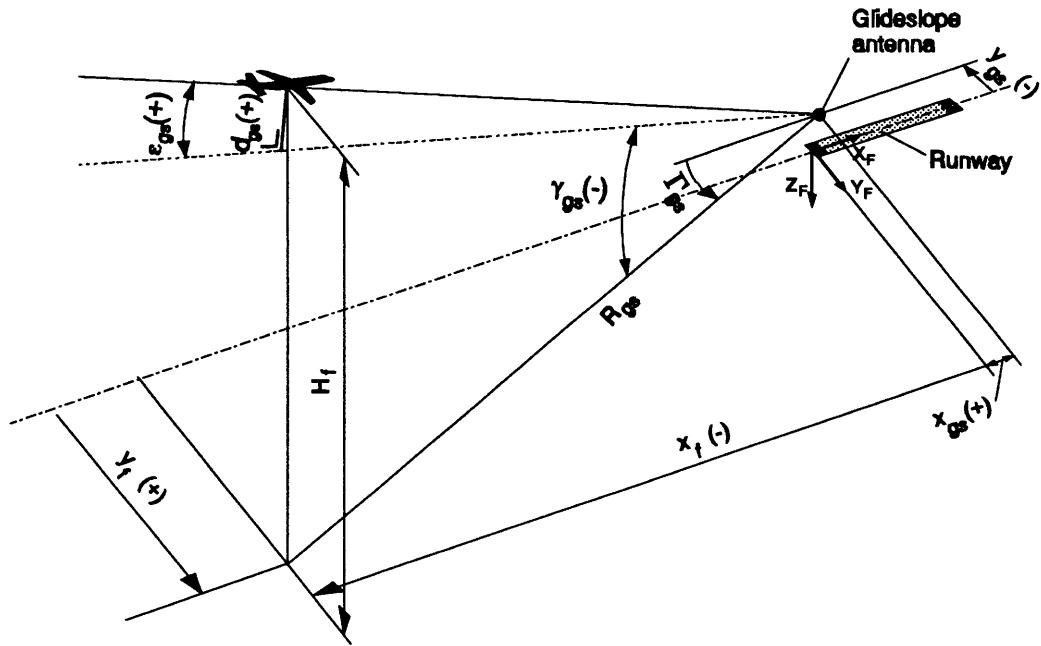


Figure C-9. Glideslope geometry.

The locations which provide a constant glideslope current lie on a cone, as shown in figures C-4 and C-9. The magnitude of the glideslope current is proportional to the glideslope error angle ϵ_{gs} (in [rad]):

$$i_{gs} = S_{gs} \epsilon_{gs} \quad [\mu\text{A}] \quad (\text{C-7})$$

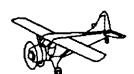
where S_{gs} is the sensitivity of the glideslope system, which follows from ref.[2]:

$$S_{gs} = \frac{625}{|\gamma_{gs}|} \quad \left[\frac{\mu\text{A}}{\text{rad}} \right] \quad (\text{C-8})$$

The nominal glide path elevation angle γ_{gs} equals some angle between -2° and -4° (the minus sign resulting from the fact that an approaching aircraft is *descending*). From figure C-9, it can be seen that the error angle ϵ_{gs} can be calculated from the coordinates x_f, y_f , and the altitude above the runway H_f with the following expressions:

$$R_{gs} = \sqrt{(x_{gs} - x_f)^2 + (-y_{gs} + y_f)^2} \quad (\text{C-9})$$

$$\epsilon_{gs} = \gamma_{gs} + \arctan \left(\frac{H_f}{R_{gs}} \right) \quad (\text{C-10})$$



and the distance from the aircraft to the glideslope is:

$$d_{gs} = (R_{gs} \tan \gamma_{gs} + H_f) \cdot \cos \gamma_{gs} \quad (\text{C-11})$$

In these expressions, ϵ_{gs} and d_{gs} are positive if the aircraft flies *above* the glide-slope reference line. The angle γ_{gs} is always negative!

To check if the aircraft flies in the area where the glideslope signals can be received (figure C-5), the angle Γ_{gs} is calculated too. Due to the different position of the glideslope antenna, this angle is not equal to Γ_{loc} . Γ_{gs} can be calculated using the following equation:

$$\Gamma_{gs} = \arcsin\left(\frac{y_f - y_{gs}}{R_{gs}}\right) \quad (\text{C-12})$$

C.2.2 Steady-state ILS offset errors.

ICAO has established limits for ILS steady-state offset errors introduced by the ground equipment, see ref.[2]. These limits are, of course, most stringent for cat.III approaches. Tables C-1 and C-2 gives these limits for localizer and glideslope transmitters.

performance category of ILS system	maximum deviation from nominal localizer sensitivity	maximum deviation of localizer reference plane from runway centerline at runway threshold [m]
I	± 17 percent	± 10.5
II	± 17 percent (± 10 percent where practicable)	± 7.5 (± 4.5 for new installations)
III	± 10 percent	± 3

Table C-1. Maximum permissible localizer steady state errors [2].

performance category of ILS system	maximum deviation from nominal glideslope sensitivity	maximum deviation of nominal glideslope elevation angle
I	± 25 percent	$\pm 0.075 \gamma_{gs}$
II	± 20 percent	$\pm 0.075 \gamma_{gs}$
III	± 15 percent	$\pm 0.04 \gamma_{gs}$

Table C-2. Maximum permissible glideslope steady state errors [2].

The nominal glide path must pass over the runway threshold at 15 ± 3 m. The maximum values of the localizer current i_{loc} and the glideslope current i_{gs} are limited to $\pm 150 \mu\text{A}$ [2], hence, $\pm 150 \mu\text{A}$ represents a full-scale deflection on the cockpit instrument.

C.2.3 ILS Noise characteristics.

Due to interference effects caused by buildings, high voltage cables, etc., the actual ILS signals become distorted in the spatial and time domains. To an approaching aircraft, these distortions appear as noise in the time domain, superimposed on the nominal ILS signals. Based on available experimental data, localizer and glideslope noise may be approximated by stochastic signals having rather simple power spectral density functions.

Refs.[1] and [16] present power spectra for ILS noise, which are expressed in the same general form as the Dryden model for longitudinal atmospheric turbulence. The power spectral density function for localizer noise can be approximated by¹⁾:

$$S_{loc}(\Omega) = 2 \sigma_{loc}^2 L_{loc} \frac{1}{1 + \Omega^2 L_{loc}^2} \quad \left[\frac{\mu\text{A}^2}{\text{rad/m}} \right] \quad (\text{C-13})$$

where:

- σ_{loc} = standard deviation of the localizer noise,
- L_{loc} = 'scale' of the localizer noise, approximately 130 m,
- Ω = spatial frequency (in [rad/m]).

The power spectral density of the glide path noise appears to be similar to the localizer noise and may be approximated by:

$$S_{gs}(\Omega) = 2 \sigma_{gs}^2 L_{gs} \frac{1}{1 + \Omega^2 L_{gs}^2} \quad \left[\frac{\mu\text{A}^2}{\text{rad/m}} \right] \quad (\text{C-14})$$

¹⁾ Note: the expressions for ILS noise and atmospheric turbulence, given in refs.[1] and [16], have an additional term π in the denominator. The Dryden spectra from ref.[24] do not contain this term, due to a different definition of the Fourier transform. In this report, the definition of the Dryden filters from [24] is used, and due to the similarity of the expressions for ILS noise, the term π will be omitted here too.

The definitions of the Fourier Transform and the Inverse Fourier Transform, used in ref.[24] are:

$$X(\omega) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt ; \quad x(t) = F^{-1}\{X(\omega)\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega$$



where:

$$\begin{aligned}\sigma_{gs} &= \text{standard deviation of the glideslope noise,} \\ L_{gs} &= \text{'scale' of the glideslope noise, approximately 85 m.}\end{aligned}$$

For atmospheric turbulence, it is often assumed that the turbulence velocities are functions only of the position in the atmosphere (the *frozen field concept* or *Taylor's hypothesis*). This assumption can be made, because aircraft usually fly at speeds large compared to turbulence velocities. However, it is somewhat doubtful if this assumption is fully valid when simulating the final approach and landing of V/STOL aircraft.

Using Taylor's hypothesis for the ILS noise will probably also induce errors, especially for aircraft with low approach speeds, such as the DHC-2 'Beaver'. Still, by using this assumption, practical expressions for the power spectral density functions, expressed in ω will result. One should remember that the power spectral density functions are in any case approximations of the actual ILS noise, so if a true representation of ILS noise is required for simulation purposes, an actual calibration of the localizer and glideslope course structure for the site in question should be used.

With Taylor's hypothesis, it is possible to use $\omega = \Omega V$. It is then possible to model the ILS noise as a white noise signal, passing through linear forming filters, which can be derived in a similar way as the forming filters for atmospheric turbulence, see appendix B. The resulting filters are:

$$H_{loc}(\omega) = \sigma_{loc} \sqrt{\frac{2L_{loc}}{V}} \frac{1}{1 + \frac{L_{loc}}{V} j\omega} \quad (C-15)$$

and:

$$H_{gs}(\omega) = \sigma_{gs} \sqrt{\frac{2L_{gs}}{V}} \frac{1}{1 + \frac{L_{gs}}{V} j\omega} \quad (C-16)$$

Ref.[20] gives somewhat different shapes for the power spectral density functions of the localizer and glideslope noise, based on average power spectral density plots of beam noise at several airports:

$$S_{loc}(\omega) = |H_{loc}(\omega)|^2 = \frac{25 (1.5 + j\omega)^2}{(0.35 + j\omega)^2 (10 + j\omega)^2} \left[\frac{\mu A^2}{rad/s} \right] \quad (C-17)$$

and:

$$S_{gs}(\omega) = |H_{gs}(\omega)|^2 = \frac{15.9}{(0.25 + j\omega)^2} \left[\frac{\mu A^2}{rad/s} \right] \quad (C-18)$$

Hence:

$$H_{loc}(\omega) = \pm \frac{5(1.5 + j\omega)}{(0.35 + j\omega)(10 + j\omega)} \quad (C-19)$$

and:

$$H_{gs}(\omega) = \pm \frac{3.9875}{(0.25 + j\omega)} \quad (C-20)$$

The maximum allowable values of the standard deviations of localizer and glideslope noise, according to ICAO standards are given in figure C-10.

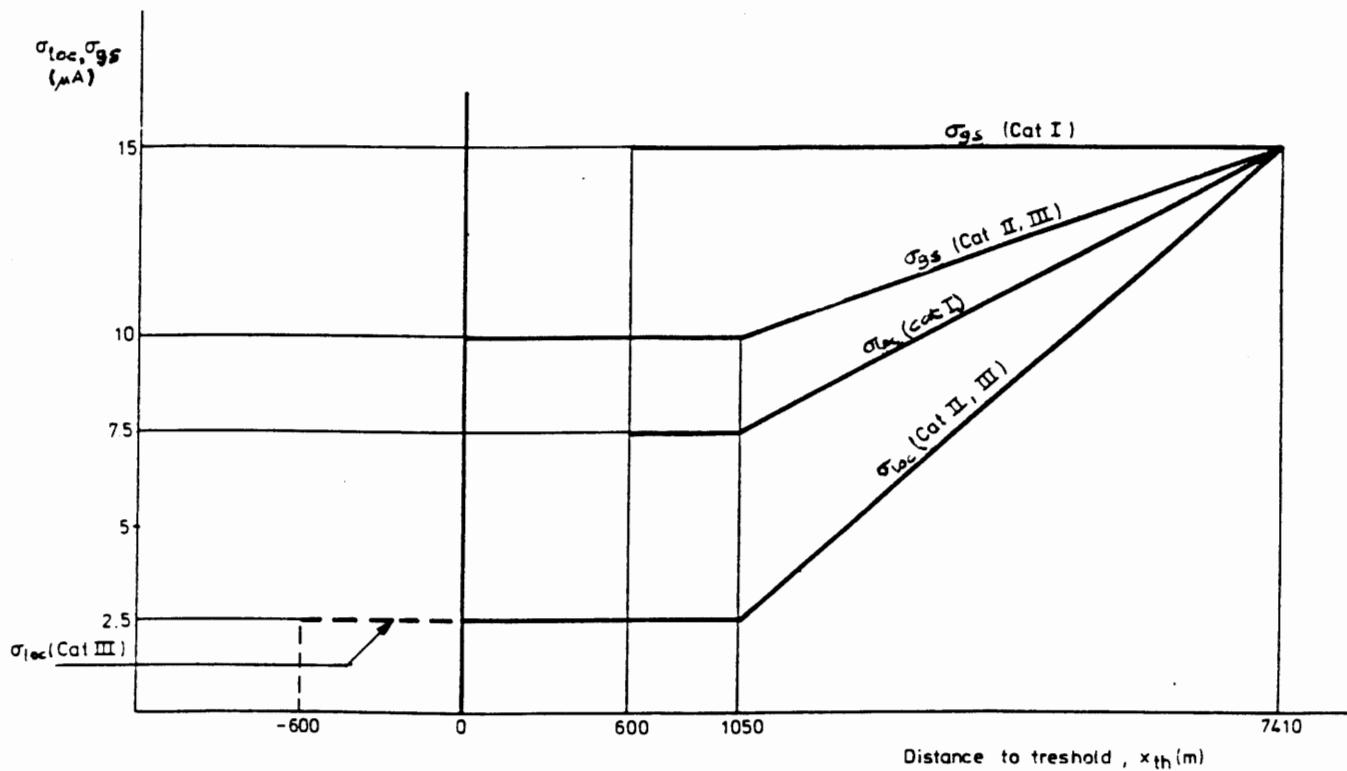


Figure C-10. Maximum allowable ILS localizer and glideslope noise.

In addition to the ILS noise and steady-state errors, specific deterministic interference patterns may occur due to signal reflections from aircraft flying in the vicinity of the transmitter antennae. It is possible to construct relatively simple models of these disturbances, which can be quite severe (ref.[1]). However, in this report such interference effects will not be considered.



C.3 The VOR navigation system.

C.3.1 Nominal VOR signals.

In 1949, the Very high frequency Omnidirectional Radio range (VOR) system was internationally recognized as a standard short-range radio navigation aid [7], [21]. The system uses the 108-118 MHz frequency range. The VOR ground station radiates a cardoid pattern that rotates at 30 RPS, generating a 30 Hz sine wave at the output of the airborne VOR receiver. The ground station also radiates an omnidirectional signal, modulated with a 30 Hz reference tone. The phase difference between the two 30 Hz tones varies directly with the bearing of the aircraft. The exact position of the aircraft can be determined by using two or more VORs, or a combination of VOR and DME information (where DME stands for Distance Measuring Equipment). See refs.[5], [7], or [21].

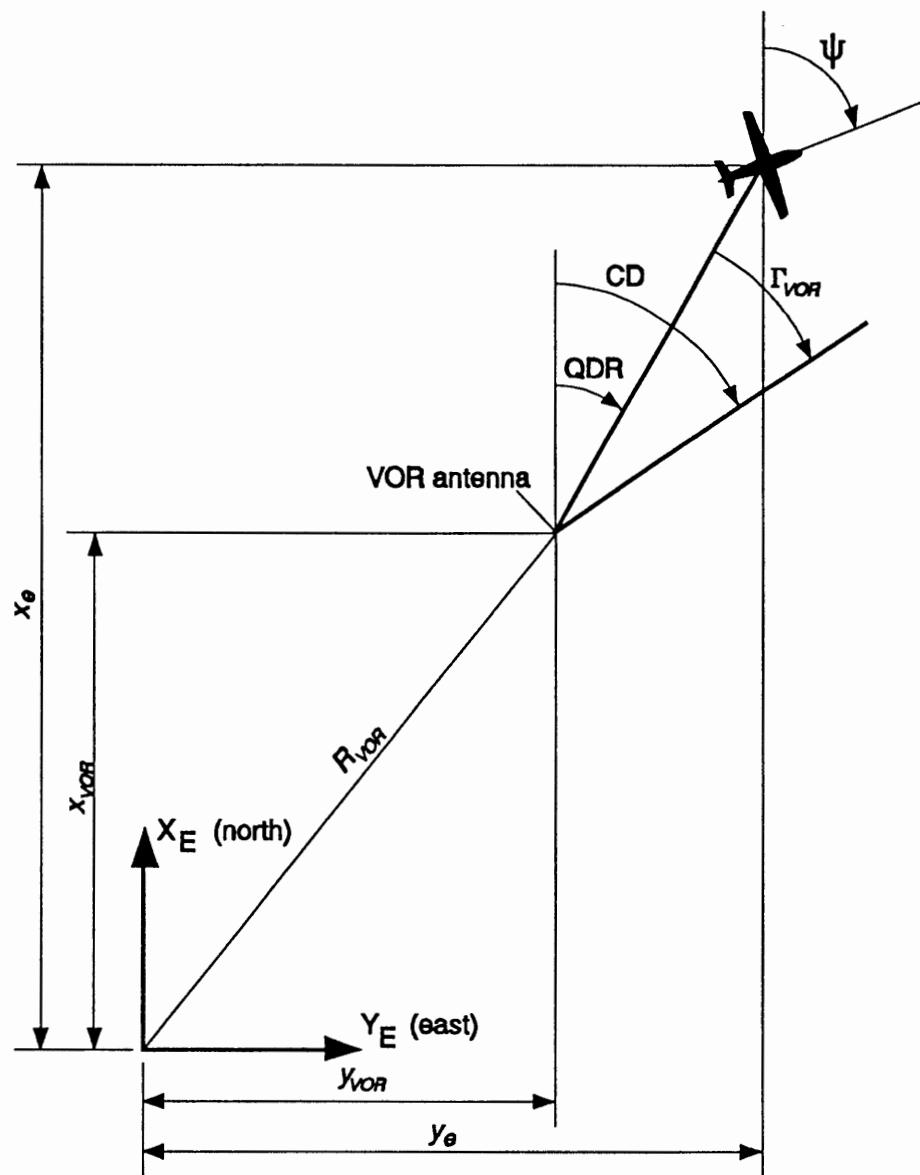


Figure C-11. Geometry of the VOR navigation.

Figure C-11 shows the VOR geometry. The pilot selects a VOR bearing with the 'Omni Bearing Selector' (OBS). This reference radian is called 'Course Datum' (CD). The bearing where the aircraft is actually flying is denoted by QDR (a radio navigation term). The course deviation angle Γ_{VOR} is depicted on the cockpit instrument and can be used for automatic control purposes. If the position of the VOR station with respect to the earth-fixed reference frame is given by $(x_{VOR}, y_{VOR}, H_{VOR})$ and the position of the aircraft by (x_e, y_e, H) , the following equations are used to calculate Γ_{VOR} :

$$QDR = \arctan\left(\frac{y_e - y_{VOR}}{x_e - x_{VOR}}\right) \quad (\text{C-21})$$

$$\Gamma_{VOR} = CD - QDR \quad (\text{C-22})$$

A typical value for a full-scale deflection of the cockpit instruments is $\Gamma_{VOR} = 10^\circ$.

It is necessary to know if the aircraft flies *to* or *from* the localizer transmitter. This information is visualized in the cockpit by means of a 'TO-FROM' indicator. If $|\psi - QDR| > 90^\circ$, the *TO* indicator should be visible; if $|\psi - QDR| < 90^\circ$, the *FROM* indicator must be set. In figure C-11, the aircraft flies *from* the VOR transmitter, and $\psi - QDR \approx 40^\circ < 90^\circ$.

C.3.2 VOR coverage and cone of silence.

The ground distance R_{VOR} can be used to determine if the aircraft flies in the area where the VOR signals can be received with appropriate reliability:

$$R_{VOR} = \sqrt{(x_e - x_{VOR})^2 + (y_e - y_{VOR})^2} \quad (\text{C-23})$$

(see figure C-12). Table C-3 gives the coverage of the VOR signals as a function of the altitude above ground level, according to ref.[7].

Altitude (ft)	VOR range (nautical miles)
1000	50
5000	92
20000	182
30000	220
3000	75
5000	95

Table C-3. VOR coverage as a function of altitude above the ground, based on two different flight tests [7].



Using the MATLAB function *POLYFIT*, the following approximative function for the VOR coverage as a function of altitude (in [m]!) was found:

$$\text{Range} = 1000 \cdot (-2.3570 \cdot 10^{-6} (H - H_{VOR})^2 + 5.7087 \cdot 10^{-2} (H - H_{VOR}) + 80.8612) \quad (\text{C-24})$$

If the aircraft flies outside the area defined by (C-24), a flag must be set. In practice, a warning flag is visible on the Course Deviation Indicator if the VOR signal is too weak. Notice that the distance R_{VOR} differs from the distance which an aircraft would receive from a DME system, if DME information is available! The DME provides the three-dimensional distance R_{DME} (figure C-12), which equals:

$$R_{DME} = \sqrt{R_{VOR}^2 + (H - H_{VOR})^2} \quad (\text{C-25})$$

If the aircraft flies in a certain area in the direct neighbourhood of the VOR transmitter, the signals are not accurate. This area is formed by a cone with a top-angle of approximately 80 to 120 degrees, the so-called 'cone of silence', see figure C-12. See also ref.[5].

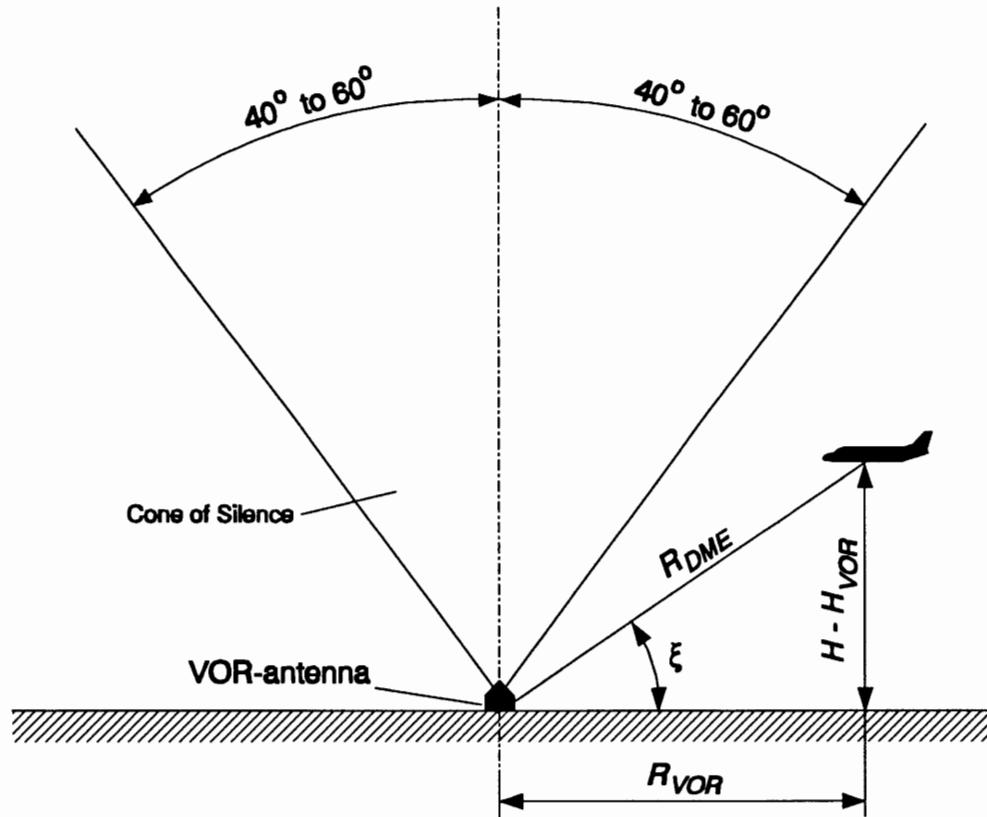


Figure C-12. The 'cone of silence'.

The VOR signals are thus received with appropriate reliability if:

$$\xi = \arctan \left(\frac{H - H_{VOR}}{R_{VOR}} \right) \leq 90^\circ - (40^\circ \text{ to } 60^\circ) \quad (\text{C-26})$$

and: $R_{VOR} \leq Range$

C.3.3 VOR steady-state errors.

The nominal VOR signals become distorted by VOR noise and steady-state errors. There are two types of systematic errors: ground station errors and airborne equipment errors. Each of these errors comprises both the equipment and antenna errors and site or location errors. ICAO has established the following error limits (refs.[2] and [7]):

- 1 - The error of the airborne equipment must be smaller than $\pm 2^\circ$ at a distance from the antenna of 4 times the wavelength and at an elevation of 0° - 40° .
- 2 - The maximum ground station error is $\pm 3.5^\circ$.

In ref.[7], some results of measurements of ground equipment errors are presented. Typical measured values are errors of ± 1.4 to ± 2.5 degrees. Besides the constant steady-state errors of the ground and airborne equipment, there are also random errors, such as variations of supply voltage of the ground and/or airborne equipment, temperature changes, inaccurate instrument reading, etc. According to ref.[7], the following values for the overall VOR system error were found from flight tests using commercial aircraft:

$$\begin{aligned} \sigma < \pm 1.7^\circ & \quad (68\% \text{ of the tests}) \\ \sigma < \pm 3.4^\circ & \quad (95\% \text{ of the tests}) \\ \sigma < \pm 5.1^\circ & \quad (99.7\% \text{ of the tests}) \end{aligned}$$

Since ref.[7] is already somewhat outdated, it can safely be assumed that modern VOR stations and receivers are more accurate.

C.4 Conclusions.

This appendix has presented some basic equations, needed for the calculation of signals from ILS and VOR systems for flight simulation purposes. The Instrument Landing System is currently the standard guidance system for non-visual approaches. Expressions for the nominal signals and for ILS steady-state and random errors have been given in this appendix. The VOR system is used for short-range navigation. The nominal VOR signals and the order of magnitude of steady-state VOR errors have also been indicated in this appendix.



Appendix D. Simulation of dynamic systems in SIMULINK - theoretical backgrounds¹⁾.

D.1 Introduction.

In this appendix some theoretical aspects of the simulation routines of SIMULINK will be outlined. The reader is advised to consult refs.[14] and [15] for a more extensive treatment of numerical analysis. A useful book, which demonstrates the practical use of these routines specifically for aircraft simulation is ref.[29]. This reference also treats the subject of linearization and trimming of nonlinear aircraft models²⁾, which will not be considered here. Since the user's manual [4] does not give details about the theory behind the integration routines in SIMULINK and the software codes of the integrator routines are not accessible to the user, the equations in this appendix were reproduced from the sources mentioned above. Some differences between the expressions in this appendix and the implementation in SIMULINK may therefore exist.

D.2 The integrators.

SIMULINK contains six numerical integration routines: *LINSIM*, *EULER*, *RK23*, *RK45*, *ADAMS*, and *GEAR* (see also chapter 2). This section gives some general information about numerical integration methods.

D.2.1 Some fundamental concepts.

The type of problems considered.

Systems in SIMULINK need to be described by a set of ordinary differential equations (ODEs) or ordinary difference equations:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t); & \mathbf{x}(0) &= \mathbf{x}_0 \\ \mathbf{x}^*(k+1) &= \mathbf{g}(\mathbf{x}^*(k), \mathbf{u}^*(k), k); & \mathbf{x}^*(0) &= \mathbf{x}_0^*\end{aligned}\tag{D-1}$$

Since few differential equations can be solved exactly, it is necessary to approximate the solutions of these ODEs numerically. Numerical integrators have been developed specifically for solving so called *initial value problems*:

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t); \quad \mathbf{y}(t_0) = \mathbf{y}_0\tag{D-2}$$

¹⁾ This appendix is largely copied from ref.[25]. Some parts have been rewritten and some errors have been fixed.

²⁾ Although trim and linearization functions are very important for flight simulation and control system design, they will not be treated in detail in this report. See refs.[11], or [29], for more information about these subjects.



Equation (D-2) does contain input signals $u(t)$, but the methods for solving (D-1) are similar to the methods designed for solving (D-2). Problem (D-2) will be considered for the remainder of this section.

Numerical solution of ordinary differential equations (refs.[14] and [15]).

Equation (D-2) is a vector equation. If y has N elements, N constants of integration appear in the solution. A unique solution to the system is obtained when the values of the elements of y are specified at a certain initial time t_0 (in this report, the time t is used as the independent variable, and $t_0 = 0$). From now on, only scalar expressions will be considered. The equations can easily be transferred to more general vector equations for sets of ODEs.

The SIMULINK integrators use so called step-by-step methods, also called difference methods or discrete variable methods, to solve the set of ODEs numerically. These methods generate a sequence of discrete points t_0, t_1, \dots , possibly with variable spacing $h_n = t_{n+1} - t_n$ (the *stepwidth*). At each point t_n , the solution $y(t_n)$ is approximated by y_n , which is calculated from earlier values. When k earlier values $y_n, y_{n-1}, \dots, y_{n-k+1}$ are used for the computation of y_{n+1} , the method is called a ' k -step method'. The Euler method (the integrator *EULER* in SIMULINK), for instance, is a single-step method:

$$y_{n+1} = y_n + h f(y_n, t_n) \quad (\text{D-3})$$

Stability and errors.

Figure D-1 shows a typical family of solutions of a first order differential equation. The actual solution depends on the initial condition $y(t_0) = y_0$. In figure D-1, a wrong value of y_0 leads to an error, which increases in time. This is called *instability of the differential equation*.

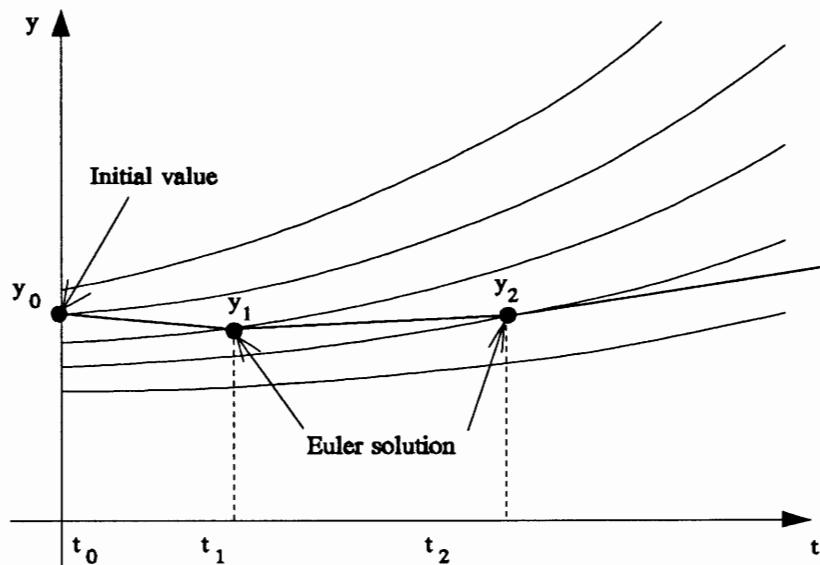


Figure D-1. Family of solutions of instable ODE. The Euler method is displayed graphically.

In figure D-1, it is demonstrated how the Euler approximation generally crosses from one solution to another between two time steps. In this case, this leads to an error which increases in time. In figure D-2, this growth of the error in time does not occur, because the different solutions come nearer to each other as time proceeds. The ODE from figure D-2 is therefore stable. Nonlinear differential equations may be unstable in some regions and stable in others. For systems of (nonlinear) ODEs, the situation is even more complex. One should therefore always be aware of possible instabilities of the system considered.

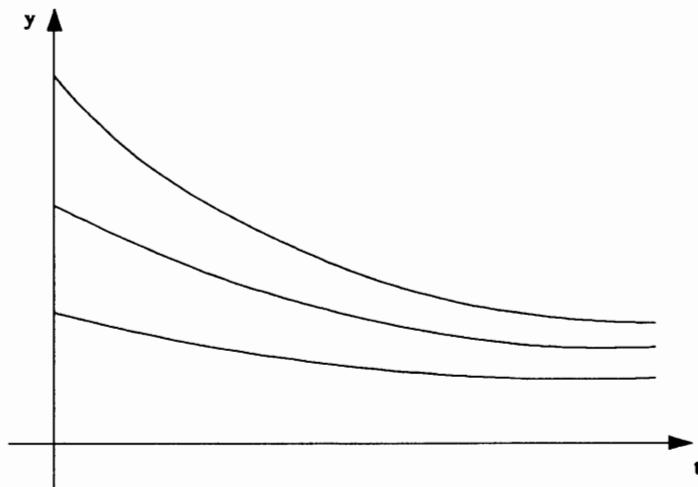
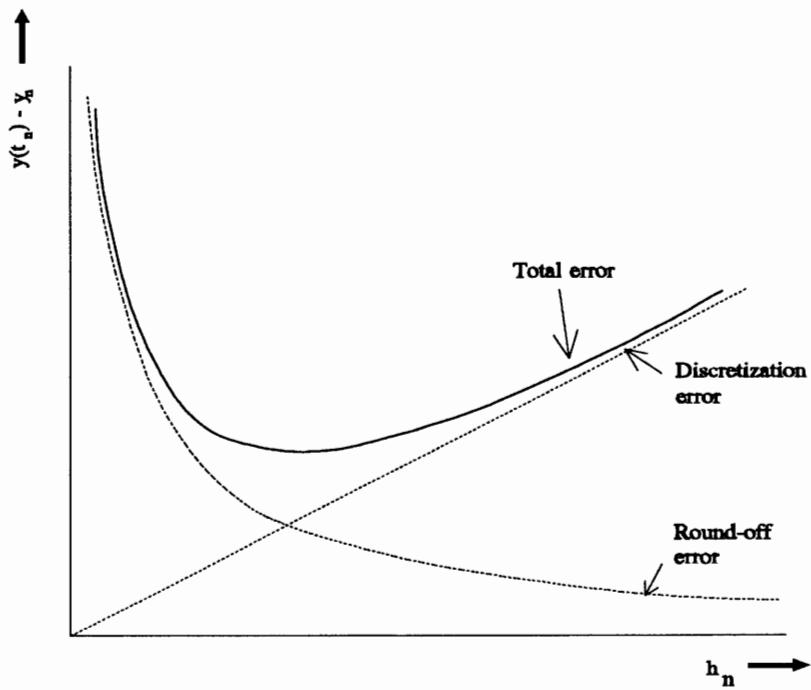


Figure D-2. Family of solutions of stable ODE.



D-3. Discretization error, roundoff error, and total error as a function of h_n .



There are two types of errors:

- 1 - discretization errors (also called truncation errors),
- 2 - roundoff errors.

Discretization errors are a property of the numerical integration method. Roundoff errors occur due to the finite number of digits, used in the calculations, hence, they are a property of the computer and the program that is used. In general, the total error decreases as h decreases, until a point where the round-off error becomes dominant. This is illustrated in figure D-3. Due to the errors, even for stable ODEs, the numerical solution may become unstable. See for instance figure D-4 and section D.2.3.

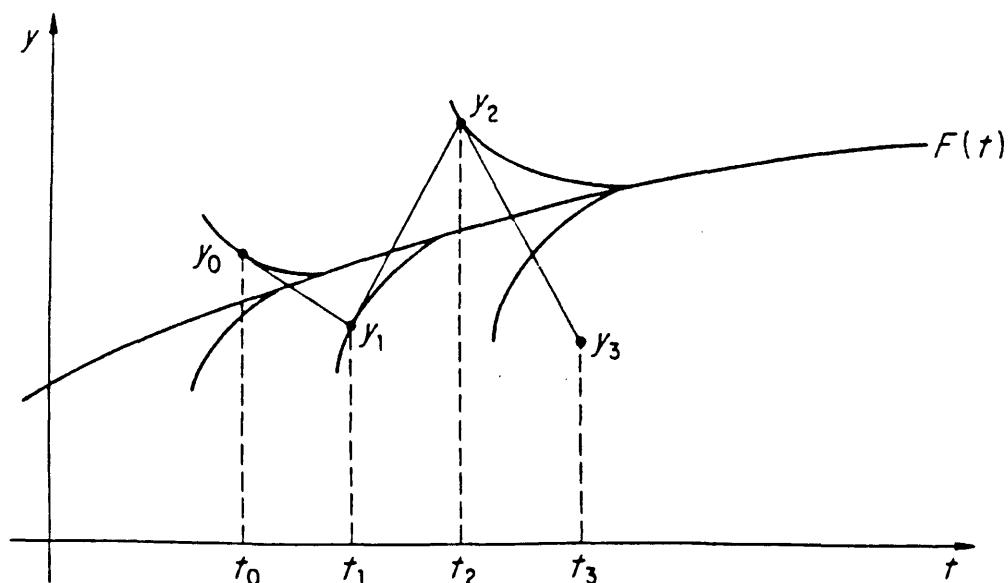


Figure D-4. Euler's method for stiff ODE.

Order of the numerical integration method.

The *order* of a numerical method is defined in terms of the local discretization error δ , obtained when the method is applied to problems with smooth solutions. A method is of order p if a number C exists so that:

$$|\delta_n| \leq C h_n^{p+1} \quad (\text{D-4})$$

C may depend on the derivatives of the function which defines the differential equation and on the length of the interval over which the solution is sought, but it should be independent of the step number n and the step size h_n . Inequality (D-4) may be written in 'big Oh notation' as:

$$\delta_n = O(h_n^{p+1}) \quad (\text{D-5})$$

D.2.2 A review of several numerical integration methods.

It is possible to distinguish between four general categories of step-by-step methods [14], which will briefly be discussed here. All results can easily be converted to vector notations for sets of ODEs.

a. Taylor series methods.

A smooth solution $y(t)$ can be approximated by the Taylor series expansion:

$$y(t+h) = y(t) + h\dot{y}(t) + \frac{h^2}{2!} \ddot{y}(t) + \dots \quad (\text{D-6})$$

If it is possible to calculate higher-order derivatives of $y(t)$, a method of order p can be obtained by using:

$$y_{n+1} = y_n + \frac{h^2}{2!} \ddot{y}_n + \dots + \frac{h^p}{p!} y_n^{(p)} \quad (\text{D-7})$$

The first neglected term provides an estimate of the local discretization error. The Euler method is an example of a Taylor series method, where all derivatives of order two and higher have been neglected. Hence, the local discretization error of the Euler method is $O(h_n^2)$. Higher-order Taylor series methods involve increasingly complex computations of the higher order derivatives. For this reason, the general applicability of these methods is quite limited [14].

It is not recommended to use the *EULER* integrator, because it needs much smaller step sizes than the other routines in order to obtain the same accuracy. It is included in SIMULINK mainly for historical reasons [4]: it forms the base of every textbook on ODEs, because it can easily be illustrated graphically, see figure D-1. The method contained in SIMULINK uses variable step sizes, but it does not take steps back in time if the user-defined error tolerance is exceeded. The method can be converted to a fixed step size method by setting the minimum step size equal to the maximum step size.

b. Runge-Kutta methods.

Runge-Kutta methods approximate Taylor series methods, without evaluating derivatives beyond the first. However, several evaluations of the function f are needed to replace the higher-order derivatives. Modern Runge-Kutta algorithms often include techniques for estimating discretization error for step size control. The Runge-Kutta methods are *self-starting*, which means that only one value y_n is needed to calculate y_{n+1} .

SIMULINK contains two Runge-Kutta integrators, called *RK23* and *RK45*.

- *RK45* is a fifth order Runge-Kutta method, which uses a fourth order method for step size control. It takes six unevenly spaced points be-



tween two output points. Four of these function values, combined with one set of coefficients, are used to produce the 4th order method. All six function values, combined with another set of coefficients, produces the 5th order method. Comparison of the two values yields the error estimate for the step size control.

- RK23 is a third order Runge-Kutta method, which uses a second order method for step size control. It needs three function evaluations to generate one output point. Two steps are needed for calculating the 2nd order method and all three function values are used for calculating the 3rd order method. The step size is again controlled by comparing the estimated error with the user-specified error tolerance.

Derivation of a second order Runge-Kutta method.

A second order Runge-Kutta method for scalar ODEs will now be investigated in detail. The results can easily be transferred to vector notation for sets of ODEs, and higher order methods can be derived in a similar way. The second order method uses two function evaluations per step. The first evaluation is:

$$k_1 = h_n f(y_n, t_n) \quad (\text{D-8})$$

Then a fractional step based on k_1 is taken:

$$k_2 = h_n f(y_n + \beta k_1, t_n + \alpha h_n) \quad (\text{D-9})$$

where α and β are two as yet unknown coefficients. The two function values are combined to make a complete step, involving two more unknown coefficients:

$$y_{n+1} = y_n + \gamma_1 k_1 + \gamma_2 k_2 \quad (\text{D-10})$$

To determine the coefficients, k_1 and k_2 are expanded in a Taylor series about (t_n, y_n) , giving:

$$\begin{aligned} k_1 &= h_n f_n \\ k_2 &= h_n \left(f_n + \beta k_1 \left(\frac{\partial f}{\partial y} \right)_n + \alpha h_n \left(\frac{\partial f}{\partial t} \right)_n + \dots \right) \\ y_{n+1} &= y_n + (\gamma_1 + \gamma_2) h_n f_n + \gamma_2 \beta h_n^2 \left(\frac{\partial f}{\partial y} \right)_n f_n + \gamma_2 \alpha h_n^2 \left(\frac{\partial f}{\partial t} \right)_n + \dots \end{aligned} \quad (\text{D-11})$$

The expansion of y_{n+1} can be compared with the expression for the true *local* solution $z_n(t)$:

$$\begin{aligned} z_n(t_{n+1}) &= z_n(t_n) + h_n z_n'(t_n) + \frac{h_n^2}{2} z_n''(t_n) + \dots \\ &= y_n + h_n f_n + \frac{h_n^2}{2} \left(\left(\frac{\partial f}{\partial y} \right)_n f_n + \left(\frac{\partial f}{\partial t} \right)_n \right) + \dots \end{aligned} \quad (\text{D-12})$$

Matching coefficients of powers h_n leads to three equations involving the unknown coefficients:

$$\begin{aligned} \gamma_1 + \gamma_2 &= 1 \\ \gamma_2 \beta &= \frac{1}{2} \\ \gamma_2 \alpha &= \frac{1}{2} \end{aligned} \quad (\text{D-13})$$

It is now possible to choose one coefficient as a parameter to produce a one-parameter family of Runge-Kutta methods. If α is chosen as a parameter this leads to:

$$\begin{aligned} k_1 &= h_n f(y_n, t_n) \\ k_2 &= h_n f(y_n + \alpha k_1, t_n + \alpha h_n) \\ y_{n+1} &= y_n + \left(1 - \frac{1}{2\alpha} \right) k_1 + \frac{1}{2\alpha} k_2 \end{aligned} \quad (\text{D-14})$$

Two obvious choices of α are $\frac{1}{2}$ (which yields a method closely related to the rectangle rule) and 1 (yielding a method related to the trapezoid rule). The method is second order, because no choice of α can eliminate the h^3 terms, neglected above (so $p = 3$).

The integrator RK45.

In a similar way, higher order Runge-Kutta methods can be derived. The fifth order method is described by the following formulas:

$$k_i = h_n f(y_n + \sum_{j=1}^{i-1} \beta_{ij} k_i, t_n + \alpha_i h_n); \quad i = 1, \dots, 6 \quad (\text{D-15a})$$

$$y_{n+1} = y_n + \sum_{i=1}^6 \gamma_i k_i \quad (\text{D-15b})$$



There are 27 coefficients: 6 α 's, 15 β 's, and 6 γ 's. The β 's form a lower triangular array, so that each k_i is obtained from previous k 's. By expanding all k 's in Taylor series, substituting the expansions into the formula for y_{n+1} , and comparing the result with the Taylor series for $z_n(t_{n+1})$, the coefficients can be determined.

For some combinations of coefficients, the difference between two expansions starts with a term including h_n^6 , but not h_n^5 . The resulting method is consequently fifth order ($p = 6$). The combination used in the Runge-Kutta Fehlberg routine are given in table D-1, see refs.[14] and [15]. For $p = 1, 2, 3$, and 4 , a p^{th} order method with p function evaluations can be found. For $p = 5$ or $p = 6$, p function evaluations will produce a method of order ($p - 1$) only. The 'additional' function evaluation to obtain a 5th order method is not wasted, however, because a set of coefficients γ_i^* exists, with which four of the six k 's can be combined to give a second value y_{n+1}^* which is accurate to fourth order. Hence:

$$y_{n+1}^* = y_n + \sum_{i=1}^6 \gamma_i^* k_i \quad (\text{D-16})$$

These coefficients are also given in table D-1. For step size control the error estimate $\sum_{i=1}^6 (\gamma_i - \gamma_i^*) k_i$ is used. The MATLAB routine *ODE45*, for example, uses the Runge-Kutta Fehlberg algorithm, in which the estimated error δ is compared with the acceptable error τ . The estimated error δ is computed with the vector equivalents of equations (D-15) and (D-16). The acceptable error is calculated with:

$$\tau_n = tol \cdot \max\{|y_n|, 1\} \quad (\text{D-17})$$

where tol is the desired accuracy. If \mathbf{y} is a vector, equation (D-17) becomes:

$$\tau_n = tol \cdot \max\{\max\{|\mathbf{y}_n|\}, 1\} \quad (\text{D-18})$$

where $|\mathbf{y}|$ is a vector in which all elements equal the absolute value of the corresponding elements in \mathbf{y} . Knowing δ and τ , *ODE45* updates the step size as follows:

$$h_{n+1} = \min \left\{ h_{\max}, 0.8 \cdot h_n \left(\frac{\tau_n}{\delta_n} \right)^{1/5} \right\} \quad \text{with } h_{n+1} \geq h_{\min} \quad (\text{D-19})$$

ODE45 only updates the step size if $\delta_n \neq 0$. It is to be expected that the SIMULINK integrator *RK45* is based upon a similar algorithm.

α_i	β_{ij}				γ_i	γ_i^*
0					$\frac{16}{135}$	$\frac{25}{216}$
$\frac{1}{4}$	$\frac{1}{4}$				0	0
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$			$\frac{6656}{12825}$	$\frac{1408}{2565}$
$\frac{12}{13}$	$\frac{1932}{2197}$	- $\frac{7200}{2197}$	$\frac{7296}{2197}$		$\frac{28561}{56430}$	$\frac{2197}{4104}$
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	- $\frac{845}{4104}$	- $\frac{9}{50}$	- $\frac{1}{5}$
$\frac{1}{2}$	- $\frac{8}{27}$	2	- $\frac{3544}{2565}$	$\frac{1859}{4104}$	- $\frac{11}{40}$	$\frac{2}{55}$
						0

Table D-1. Fehlberg coefficients.

The integrator *RK23*.

The SIMULINK integrator *RK23* can be derived in a similar way. The software source code is not accessible for the user, but the algorithm is probably the same as the one that is used in the MATLAB function *ODE23*. This routine uses the following expressions to obtain an approximation of $y(t_n)$:

$$\begin{aligned}
 k_1 &= f(y, t) \\
 k_2 &= f(y + h_n k_1, t + h_n) \\
 k_3 &= f(y + \frac{h_n}{4} (k_1 + k_2), t + \frac{h_n}{2}) \\
 y_{n+1} &= y_n + \frac{h_n}{6} (k_1 + 4k_3 + k_2)
 \end{aligned} \tag{D-20}$$

ODE23 updates y if the estimated error δ is smaller than or equal to the acceptable error τ . These are estimated with the following expressions:

$$\begin{aligned}
 \delta_n &= \left| \frac{h_n}{3} (k_1 - 2k_3 + k_2) \right| \\
 \tau_n &= tol \cdot \max\{|y_n|, 1\}
 \end{aligned} \tag{D-21}$$



or if \mathbf{y} and \mathbf{k} are vectors:

$$\begin{aligned}\delta_n &= \max \left\{ \left| \frac{h_n}{3} (\mathbf{k}_1 - 2\mathbf{k}_3 + \mathbf{k}_2) \right| \right\} \\ \tau_n &= tol \cdot \max \left\{ \max \{ |\mathbf{y}_n| \}, 1 \right\}\end{aligned}\quad (\text{D-22})$$

The estimate of δ involves another Runge-Kutta method than used for the update y_{n+1} , just as was the case with *RK45*. The step size is chosen as large as possible, within the user-specified limits. In *ODE23*, the step size is updated with the following formula:

$$h_{n+1} = \min \left\{ h_{\max}, 0.9 h_n \left(\frac{\tau_n}{\delta_n} \right)^{1/3} \right\} \quad \text{with } h_{n+1} \geq h_{\min} \quad (\text{D-23})$$

SIMULINK uses a variable step size, but it is possible to fix the step size by setting the maximum step size equal to the minimum step size. The Runge-Kutta methods usually outperform the other methods when the system is highly nonlinear or discontinuous. They perform well for mixed continuous and discrete time systems. For systems with a very wide spread of time constants ('stiff systems'), *LINSIM* or *GEAR* should be used (refs.[4], [29]).

c. Multistep methods.

The methods considered so far calculate the value y_{n+1} by means of a function which depends only on t_n , y_n , and the step size h_n . Multistep methods use information at previous points to obtain more accuracy, namely y_{n-1} , y_{n-2} , y_{n-3} , ..., and f_{n-1} , f_{n-2} , f_{n-3} , ..., where $f_i = f(y_i, t_i)$. Multistep methods can be very effective. They usually require less function evaluations than one-step methods to obtain a high accuracy. Furthermore, an estimate of the discretization error can often be trivially obtained [14]. All linear multistep methods can be considered as special cases of the formula:

$$y_{n+1} = \sum_{i=1}^k \alpha_i y_{n+1-i} + h \sum_{i=0}^k \beta_i f_{n+1-i} \quad (\text{D-24})$$

where k is a fixed integer and either α_k or β_k is not zero. The formula defines the general k -step method. It is linear, because every f_i enters linearly; f doesn't necessarily have to be a linear function in its arguments. After the method is 'started', each step requires the calculation of y_{n+1} from the known values: y_n , y_{n-1} , ..., y_{n-k+1} , f_n , f_{n-1} , ..., f_{n-k+1} . If $\beta_0 = 0$, this method is *explicit* and the calculation is straightforward. If $\beta_0 \neq 0$, the method is *implicit*, because f_{n+1} is needed to solve for y_{n+1} .

Usually, two multistep methods are used together in computing each step of a solution: an explicit method, called predictor, is followed by one or more applications of an implicit method, which is called a corrector. These

methods are therefore called *predictor-corrector* methods. A predictor-corrector scheme for calculating y_{n+1} is as follows:

- 1 - Use the predictor to calculate $y_{n+1}^{(0)}$, an initial approximation to y_{n+1} . Set $i = 0$.
- 2 - Evaluate the derivative function and set $f_{n+1}^{(i)} = f(y_{n+1}^{(i)}, t_{n+1})$.
- 3 - Calculate a better approximation $y_{n+1}^{(i+1)}$ using the corrector formula with $f_{n+1} = f_{n+1}^{(i)}$.
- 4 - If $|y_{n+1}^{(i+1)} - y_{n+1}^{(i)}| >$ some specified error tolerance, then increment i by 1 and go to step 2; otherwise, set $y_{n+1} = y_{n+1}^{(i+1)}$.

The *ADAMS* integrator is a predictor-corrector multistep method, contained in SIMULINK. A number of different Adams methods can be found in the literature, but the most important ones are Adams-Bashforth and Adams-Moulton, which will be described here.

The Adams-Bashforth method.

The Adams-Bashforth integrator is an *explicit* multistep method. It can be derived in a number of different ways, see ref.[15]. The derivation is too complicated for a brief treatment, so only the resulting expressions will be presented here. The k -step Adams-Bashforth formula can be written as:

$$y_{n+1} = y_n + h \sum_{i=1}^k \beta_{ki} f_{n+1-i} \quad (\text{D-25})$$

where:

$$\beta_{ki} = (-1)^{i-1} \sum_{j=i-1}^{k-1} \gamma_j \binom{j}{i-1} \quad (\text{D-26})$$

and:

$$\gamma_j = (-1)^j \int_0^1 \binom{-s}{j} ds \quad (\text{D-27})$$

Although equations (D-26) and (D-27) can be used to determine the coefficients β_{kj} and γ_j , they do not give the most convenient form for this. Frequently the method of *generating functions* is used [15]. This method yields the following recursive expression:

$$\gamma_m + \frac{1}{2} \gamma_{m-1} + \frac{1}{3} \gamma_{m+2} + \dots + \frac{1}{m+1} \gamma_0 = 1 \quad (\text{D-28})$$

Table D-2 gives the resulting set of coefficients γ_j and β_{kj} .



$m:$	0	1	2	3	4	5
γ_m	1	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{3}{8}$	$\frac{251}{720}$	$\frac{95}{288}$
$i:$	1	2	3	4	5	6
β_{1i}	1					
$2 \beta_{2i}$	3	-1				
$12 \beta_{3i}$	23	-16	5			
$24 \beta_{4i}$	55	-59	37	-9		
$720 \beta_{5i}$	1901	-2774	2616	-1274	251	
$1440 \beta_{6i}$	4277	-7923	9982	-7298	2877	-475

Table D-2. Coefficients for Adams-Bashforth methods.The Adams-Moulton method.

The Adams-Moulton integrator is an *implicit* multistep method. Its derivation can be found ref.[15]. The resulting method can be written as:

$$y_{n+1} = y_{n-1} + h \sum_{i=0}^{k-1} \beta_{ki}^* f_{n+1-i} \quad (\text{D-30})$$

where:

$$\beta_{ki}^* = (-1)^j \sum_{j=i}^{k-1} \gamma_j^* \binom{j}{i} \quad (\text{D-31})$$

and:

$$\gamma_j^* = (-1)^j \int_0^1 \binom{-s+1}{j} ds \quad (\text{D-32})$$

The coefficients γ_j^* can again be found by use of generating functions. Table D-3 lists some values of γ_j^* and β_{ki}^* .

There are two important differences between the Adams-Bashforth and Adams-Moulton methods. The first is that the coefficients of the latter are smaller, which results in smaller round-off and smaller discretization errors. The second difference is that for the same order the Adams-Moulton method uses information from fewer points. The k -step Moulton method is thus of order $k + 1$, but the k -step Bashforth method is of order k only [15].

$m:$	0	1	2	3	4	5
γ_m^*	1	$-\frac{1}{2}$	$-\frac{1}{12}$	$-\frac{1}{24}$	$-\frac{19}{720}$	$-\frac{3}{160}$
$i:$	1	2	3	4	5	6
β_{1i}^*	1					
$2 \beta_{2i}^*$	1	1				
$12 \beta_{3i}^*$	5	8	-1			
$24 \beta_{4i}^*$	9	19	-5	1		
$720 \beta_{5i}^*$	251	646	-264	106	-19	
$1440 \beta_{6i}^*$	475	1427	-798	482	-173	27

Table D-3. Coefficients for Adams-Moulton method.Predictor-corrector Adams method.

Often, the Adams-Bashforth method is used as predictor and the Adams-Moulton method as corrector. An example of a good predictor-corrector method is the fourth order Adams method, which uses the fourth order Bashforth and Moulton formulas:

$$\text{predictor: } y_{n+1} = y_n + \frac{h}{24} (55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) \quad (\text{D-33})$$

$$\text{corrector: } y_{n+1} = y_n + \frac{h}{24} (9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2})$$

The SIMULINK integrator *ADAMS* most likely uses some combination of Bashforth and Moulton methods. This integrator is particularly suited for systems which are smooth and nonlinear, but do not have widely varying time constants [4]. The Adams routines do not give enough accuracy for digital control purposes (systems with mixtures of continuous and discrete dynamics). This is especially true when advanced adaptive and parameter estimation techniques are used [29].

ADAMS can take an undetermined number of steps between two output points. It is therefore not possible to convert this method to a fixed step size method by setting the maximum step size equal to the minimum step size in SIMULINK. The accuracy of the method is not affected by maximum and minimum step sizes; for *ADAMS* these user-defined parameters are used only to control the output points for plotting the results.



d. Extrapolation methods.

The predictor formulas, described earlier, can be regarded as methods, where y_{n+1} is extrapolated from known previous values of y_i and f_i . Other kinds of extrapolation have been implemented in various *extrapolation methods*. These methods will not be discussed here, because they are not used in SIMULINK. See ref.[15] for more information.

D.2.3 Stiff differential equations.

Some problems are not handled very efficiently by the methods mentioned so far. A special class of differential equations, which occurs in many physically important situations, is called 'stiff'. Stiffness can roughly be defined as the presence of one or more fast decay processes in time, with a time constant that is short compared to the time span of interest (refs.[14] and [15]). The *time constant* is defined as the time in which a solution to a differential equation decays by a factor $1/e$.

In a physical system, different elements often have different time constants, which means that some solutions to differential equations decay much faster than others. In such cases, the signals with fast dynamics will determine the stability of the integration method, even though these components may have decayed to insignificant levels. After a short time, the fast components are negligibly small compared to the slow components. From this point on conventional numerical methods must take very small time steps because of the fast components, although only the slow components contain any significant information at that time. This is the problem of stiff equations.

Example D-1 (reproduced from refs.[14] and [15]).

Consider the system:

$$\dot{u} = 998u + 1998v$$

$$\dot{v} = -999u - 1999v$$

The eigenvalues of the matrix of coefficients:

$$\begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix}$$

are -1 and -1000. If $u(0) = 1$ and $v(0) = 1$, the solution is:

$$u = 4e^{-t} - 3e^{-1000t}$$

$$v = -2e^{-t} + 3e^{-1000t}$$

After a very short time, the solution can be approximated by:

$$u = 4e^{-t}$$

$$v = -2e^{-t}$$

When using Euler's method to solve this system, the discretized solution becomes:

$$u_{n+1} = u_n + h(990u_n + 1998v_n)$$

$$v_{n+1} = v_n + h(-999u_n + 1999v_n)$$

For $h = 0.01$, the numerical solution behaves as disastrous as shown in figure D-4, which shows the family of solutions of a stiff problem. As in figure D-1, the numerical solution moves from one curve to another. Although the system is stable, the numerical solution diverges rapidly. The only way to solve this problem is reducing the step size, but eventually, roundoff and discretization errors will accumulate enough to result in another instability. The transient part of the solution, which decays very fast, prevents an increase in step size, although the solution is very smooth after only a few seconds! \square

SIMULINK contains two integrators which are particularly suited for solving stiff ODEs: *GEAR*, and *LINSIM* [4].

The integrator *LINSIM*.

LINSIM works well for linear or 'almost' linear systems, with a very wide range of time constants. The method divides the system in linear and nonlinear subsystems and simulates only the nonlinear system dynamics. The linear part of the system is then solved separately, in a much faster way. The method uses variable step sizes, but setting the maximum step size equal to the minimum step size yields a fixed step method.

The integrator *GEAR*.

GEAR works well with systems which are smooth, nonlinear, and stiff. The method is less efficient than others for non-stiff systems. It may not work well when there are singularities in the system or if the system is perturbed by rapidly changing inputs. *GEAR* uses a predictor-corrector method which takes a variable number of steps between two output points. It can not be converted to a fixed step method by setting the maximum step size equal to the minimum step size. In ref.[15], a multistep method that is suited for solving stiff systems is introduced. A detailed discussion of concepts such as *stiff stability* is also included in ref.[15]. The stiffly stable multistep method can be written as:

$$y_n = \sum_{i=1}^k \alpha_i y_{n-i} + h \beta_0 f_n \quad (\text{D-34})$$



The coefficients α_i and β_0 are listed in table D-4. The method is closely related to the Adams method. Many different implementations of this stiffly stable multistep method exist nowadays. They differ in the ways in which the implicit expression is solved. It is not known how SIMULINK handles this problem, because of the inaccessibility of the source codes. Nevertheless, all implementations are based upon equation (D-34).

k	2	3	4	5	6
β_0	$\frac{2}{3}$	$\frac{6}{11}$	$\frac{12}{25}$	$\frac{60}{137}$	$\frac{60}{147}$
α_1	$\frac{4}{3}$	$\frac{18}{11}$	$\frac{48}{25}$	$\frac{300}{137}$	$\frac{360}{147}$
α_2	$-\frac{1}{3}$	$-\frac{9}{11}$	$-\frac{36}{25}$	$-\frac{300}{137}$	$-\frac{450}{147}$
α_3		$\frac{2}{11}$	$\frac{16}{25}$	$\frac{200}{137}$	$\frac{400}{147}$
α_4			$-\frac{3}{25}$	$-\frac{75}{137}$	$-\frac{225}{147}$
α_5				$\frac{12}{137}$	$\frac{72}{147}$
α_6					$-\frac{10}{147}$

Table D-4. Coefficients for stiffly stable method (*GEAR*).

In SIMULINK, it is possible to use a combination of *ADAMS* and *GEAR*. SIMULINK will automatically determine if the system is stiff, in which case it will use *GEAR*, or not, in which case it will use *ADAMS*.

D.2.4 Differential algebraic equations.

The most generalized system equations of an aircraft can be written as:

$$\begin{aligned} T\ddot{\mathbf{x}}(t) &= \mathbf{f}[\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t)] \\ \mathbf{y}(t) &= \mathbf{g}[\mathbf{x}(t), \dot{\mathbf{x}}(t), \mathbf{u}(t)] \end{aligned} \quad (\text{D-35})$$

(see ref.[11]). These equations can be formulated as implicit differential equations:

$$\mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{u}(t)) = 0 \quad (\text{D-36})$$

The presence of algebraic equations in variables of which the derivatives are defined in an explicit differential equation also makes these equations implicit. Such equations are called '*Differential Algebraic Equations*' (DAEs). It is

not always possible to transfer these DAEs to ODEs. The integration method of Gear could, in theory, be used for solution of DAEs [8], but unfortunately, this is not (yet?) possible in SIMULINK.

D.3 Simulation of digital controllers.

The time responses of the linear time-invariant continuous system:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \quad (\text{D-37})$$

can easily be obtained by discretizing and then using the resulting recursion:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, k) \quad (\text{D-38})$$

This method is, for instance, used by the MATLAB command *LSIM* from the CONTROL SYSTEMS TOOLBOX. This approach is fast, but it generates information on the sample times only. For the general nonlinear time-varying system, numerical integration routines as described in section D.2 must be used. If a digital controller is used to control continuous-time systems, the scheme shown in figure D-5 should be used. This figure assumes a Zero Order Hold; thus, the control input $u(t)$ is updated at each $t = kT_s$ and then hold constant until $t = (k+1)T_s$. The continuous dynamics are contained in the block $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, t)$. They are integrated using a Runge-Kutta integrator (the Adams methods are not accurate enough for simulating discrete systems).

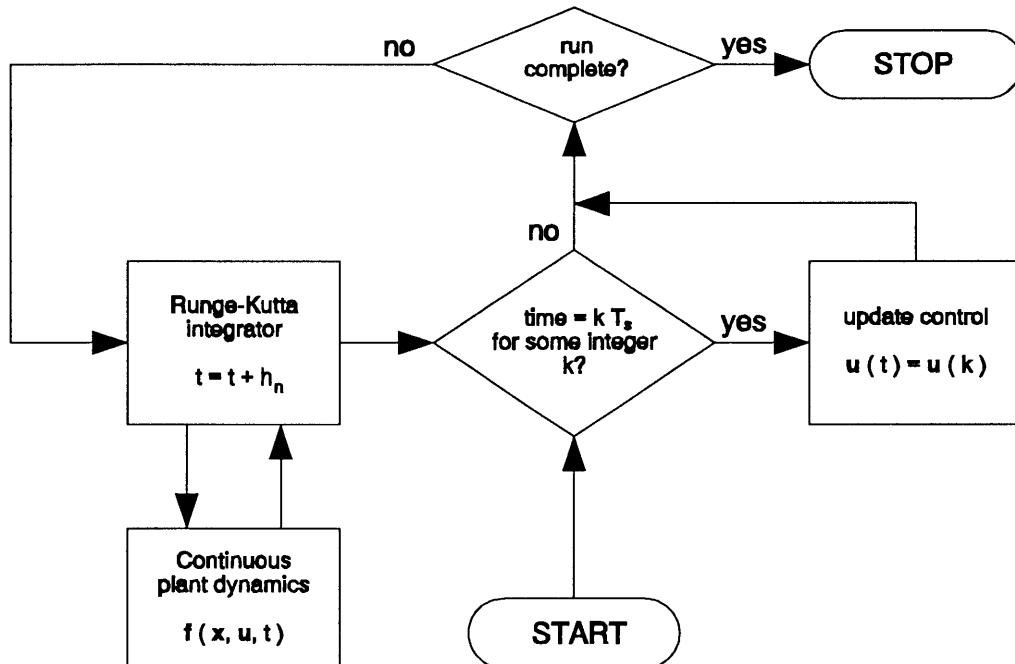


Figure D-5. Digital control simulation scheme (see ref.[29]).



Note that two time intervals are involved: the sampling period T_s and the Runge-Kutta integration period $h_n \ll T_s$. This technique provides $\mathbf{x}(t)$ even at points between sample instants (in fact, it provides $\mathbf{x}(t)$ at multiples of T_s), which is essential in verifying acceptable *intersample behaviour* of the closed-loop system prior to implementing the digital controller on the actual plant. Even though the closed-loop behaviour is acceptable at the sample points, with improper digital control system design there can be serious problems between the samples, because a badly designed controller can destroy observability, so that poor intersample behaviour is not apparent at the sample points [29].

D.4 Obtaining state models from transfer functions.

Simulation programs like SIMULINK use numerical integration routines to obtain *state trajectories*, so it is necessary that the systems are written as a set of linear or nonlinear state equations. For this reason, transfer functions need to be transformed into state models. Since this transformation is normally performed automatically by the simulation program, the user doesn't need to worry about it. However, it is still useful to know the principles behind this transformation, and at the end of this section it will be shown that it is sometimes convenient to do this transformation by hand.

A number of different transformations from transfer function to state space models can be found in the literature (see for instance refs.[8], [9], or [29]). Only one method will be shown here. Consider the following transfer function:

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_{n-1}s^{n-1} + \dots + b_2s^2 + b_1s + b_0}{s^n + a_{n-1}s^{n-1} + \dots + a_2s^2 + a_1s + a_0} \quad (\text{D-39})$$

To transform this equation into a state model, the equation will first be rewritten and a helpvariable $V(s)$ will be introduced:

$$\frac{Y(s)}{b_{n-1}s^{n-1} + \dots + b_1s + b_0} = \frac{U(s)}{s^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0} = V(s) \quad (\text{D-40})$$

This yields the following equations:

$$Y(s) = \{b_{n-1}s^{n-1} + \dots + b_1s + b_0\} \cdot V(s) \quad (\text{D-41})$$

and:

$$s^n V(s) = U(s) - a_{n-1}s^{n-1}V(s) - \dots - a_1sV(s) - a_0V(s) \quad (\text{D-42})$$

Equations (D-41) and (D-42) can be used to obtain a state model. Figure D-6 shows the block-diagram equivalent of the transfer function (D-39). The

diagram is built up from first order integrals ($1/s$ blocks) and gain blocks only. If the state vector for this transfer function block is defined as:

$$\mathbf{x} = \begin{bmatrix} 1 \\ s \\ s^2 \\ \vdots \\ s^{n-1} \end{bmatrix} \cdot V(s)$$

the following linear state equation is found:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u \quad (\text{D-43})$$

$$\mathbf{y} = \mathbf{c}\mathbf{x}$$

with:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}; \quad \mathbf{c} = [b_0 \ b_1 \ b_2 \ \dots \ b_{n-1}]$$

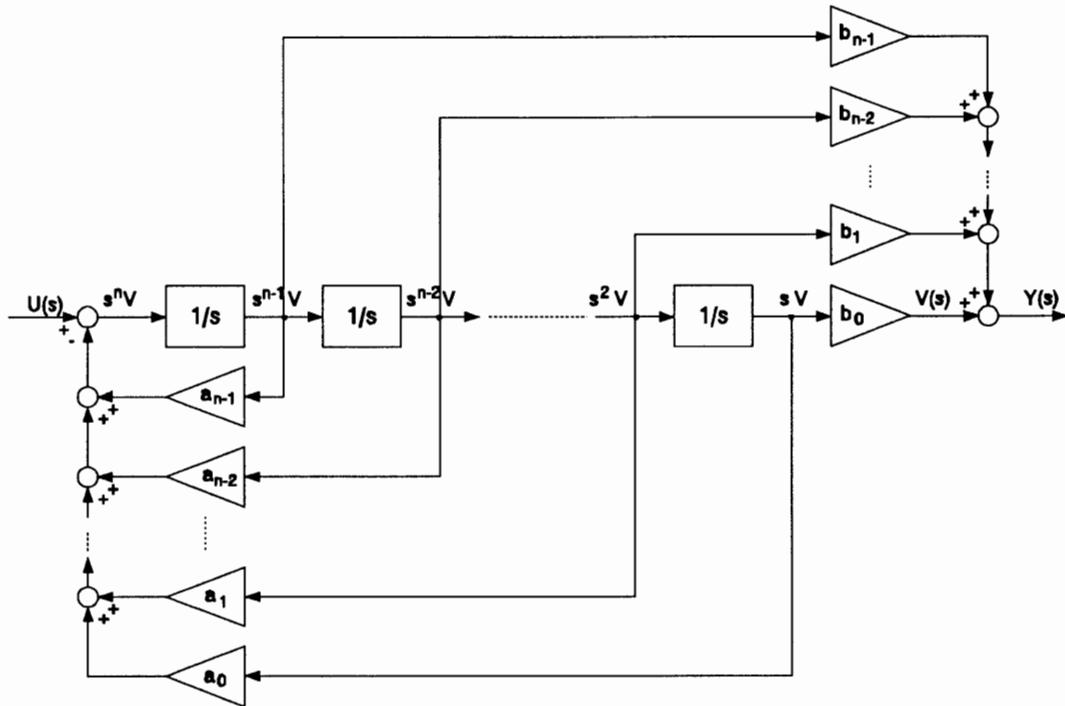


Figure D-6. Block-diagram equivalent of transfer function (D-39).



As said before, the simulation package will usually do this transformation automatically. However, if the coefficients $a_0, \dots, a_{n-2}, a_{n-1}$, or $b_0, \dots, b_{n-2}, b_{n-1}$ are varying during the simulation (see for instance the Dryden turbulence filters (B-20) to (B-22), where the coefficients of the filters are dependent on $L_g = L_g(H)$, $\sigma_g = \sigma_g(H)$, and V), it may be necessary to perform the transformation by hand, using gain scheduling within the block-diagram D-6¹⁾.

D.5 Algebraic loops.

One difficulty which can arise during simulation of a continuous time system on a digital computer is the occurrence of Algebraic Loops. Consider a system in block-diagram representation. Algebraic, or implicit, loops occur when two or more blocks with direct feedthrough of their inputs form a feedback loop. It shall be shown that these loops have to be solved with an iterative algorithm, which considerably slows down the simulation.

The model which has to be simulated is in fact a parallel system: all variables change simultaneously. But the calculation of responses of a parallel system on a *digital* computer system (with one microprocessor) can only be done sequentially. The ordering of the different blocks then becomes important. This will be illustrated with an example from ref.[8].

Example D-2.

Consider the system in figure D-7, consisting of three gains: $A = 1$, $B = 2$, and $C = 3$. Two calculation sequences will be evaluated: $uABC$ and $uCBA$. The signals u , u_A , u_B , and u_C are calculated as follows:

$$\begin{array}{llll} uABC: & u & = & u \\ & u_A = u^*A & = & u \\ & u_B = u_A^*B = u^*A^*B & = & 2u \\ & u_C = u_B^*C = u^*A^*B^*C & = & 6u \\ \\ uCBA: & u & = & u \\ & u_C = u_B^*C = 3u_B & = & 3u \\ & u_B = u_A^*B = 2u_A & = & 2u \\ & u_A = u^*A & = & u \end{array}$$

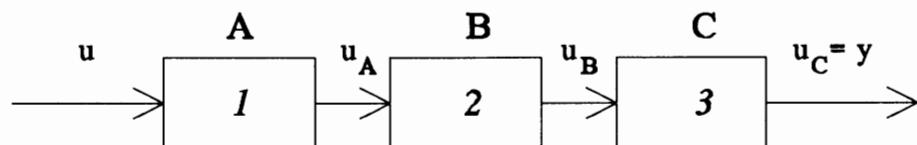


Figure D-7. Model, consisting of three gains.

¹⁾ Gain scheduling can easily be implemented in SIMULINK, using a (nonlinear) function block, representing the varying gain, and a product block. See appendix F.

Suppose that $u = 1$ for $k \geq 0$, k being the period on which the signals are calculated (by the numerical integrator). Then we get the following sequence of the signals:

period k	uABC				uCBA			
	u	u_A	u_B	u_C	u	u_A	u_B	u_C
0	1	1	2	6	1	1	0	0
1	1	1	2	6	1	1	2	0
2	1	1	2	6	1	1	2	6
3	1	1	2	6	1	1	2	6

Table D-5. Magnitude of the signals for two calculating sequences.

In table D-5 it is made clear that the sequence uCBA introduces an extra time delay of two periods, caused by the fact that u_C needs the value of u_B , and u_B needs the value of u_A , which are not yet available at $k = 0!$ This time delay is not present in the original model. \square

SIMULINK automatically chooses the ordering of the calculations, so that the problems of example D-2 will not occur. But when feedback is applied, it is possible that no suitable calculating sequence can be found. This is illustrated with the next example, also coming from ref.[8].

Example D-3.

Consider the system in figure D-8, consisting of a gain A with negative unity feedback.

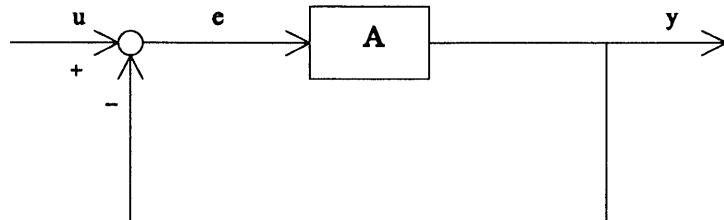
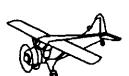


Figure D-8. Gain with unity feedback: an algebraic loop.

If this system is analysed with an integrator with stepwith h_n , it is possible to write:

$$e_n = u_n - y_{n-1}$$

$$y_n = Ae_n$$



where $y_n = y(n \cdot h_n)$, $n \in \mathbb{Z}$, etc. Taking the Z-transform of these equations, yields:

$$\frac{Y(z)}{U(z)} = \frac{Az}{z + A}$$

which indicate that additional dynamics, due to the sequential calculation of the parallel feedback system, have been introduced (see figure D-9). \square

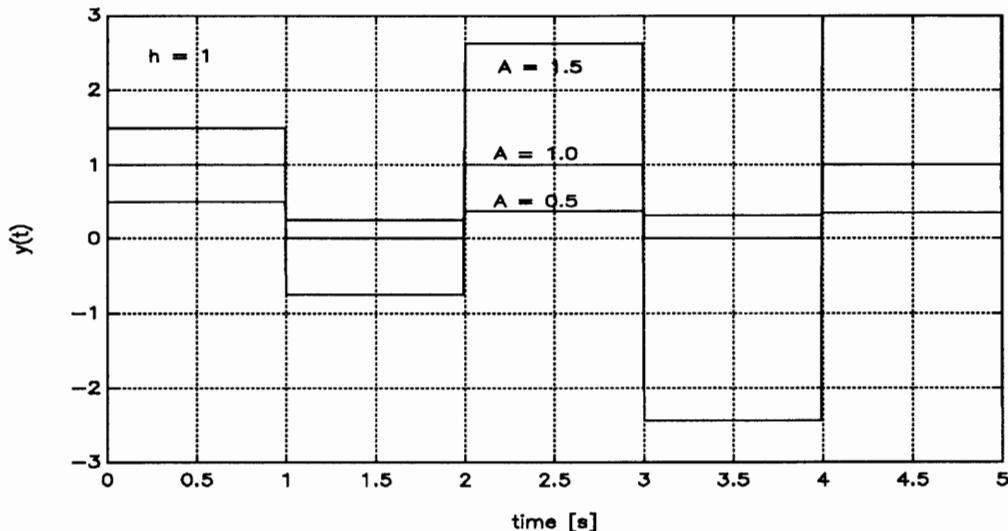


Figure D-9. Dynamics,introduced by an algebraic loop.

One should try to avoid algebraic loops whenever possible. When SIMULINK does detect an algebraic loop, this will be solved with the iterative Newton-Raphson method, see refs.[4] or [8]. To the system with algebraic equations, a new block ALB is added, see figure D-10. It has an input $g(x)$ and output x . The block tries to find a value of x , such that $g(x) = x$. This means that the nonlinear equation:

$$G(x) = g(x) - x = 0 \quad (\text{D-44})$$

has to be solved. This is done by applying the Newton-Raphson method:

$$x_{i+1} = x_i - [G'(x_i)]^{-1} G(x_i) \quad (\text{D-45})$$

where:

$$G'(x_i) = \frac{G(x_i + \Delta x) - G(x_i)}{\Delta x}; \quad \Delta x \text{ small.} \quad (\text{D-46})$$

The subscript i indicates the iteration number, not the time. SIMULINK returns an error if it can't solve an algebraic loop in 200 iterations. Note that the Newton-Raphson iterations have to be carried out for every time step, so the integration will inevitably slow down.

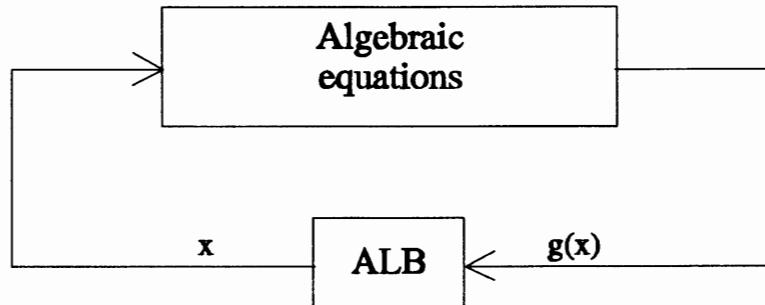


Figure D-10. Iterative solution of an algebraic loop.

D.6 Conclusions.

In this appendix, the basic theory behind a number of numerical integration methods, which can be applied to ordinary differential equations, has been presented. These methods are used in simulation programs like SIMULINK. One should always be careful in selecting step size and error-tolerance, because numerical solutions may become instable if these parameters are not properly set. The Runge-Kutta integrators are convenient for many purposes, including the simulation of a continuous system that is controlled by a digital controller. However, they are not very efficient for 'stiff' systems. Finally, the transformation of transfer functions into state models and the concept of algebraic loops have been explained.



Appendix E. Steady-state trimmed flight.

E.1 Introduction.

In this appendix, it will be shown how steady-state trimmed flight conditions can be found, using a numerical trim algorithm, and how this trim algorithm can be implemented in MATLAB/SIMULINK. Although SIMULINK contains general trim routines, the special problem of trimming a nonlinear aircraft model can better be solved by applying a specialized aircraft trim routine. In section F.2.7 of appendix F, the resulting aircraft trim program will be described in detail. The theory of this appendix is fully based upon ref.[29].

E.2 Definition of steady-state flight.

In appendix A, the general nonlinear equations of motion for a rigid aircraft were derived. These equations could be written as a set of twelve Ordinary Differential Equations (ODEs), which in general look like:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \quad (\text{E-1})$$

Here, we will consider the more general equation:

$$\mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{u}(t), t) = 0 \quad (\text{E-2})$$

In ref.[29], a *singular point*, or *equilibrium point* of an autonomous (no external control inputs), time-invariant system is defined as:

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{u}) = 0, \quad \text{with: } \dot{\mathbf{x}} = 0; \mathbf{u} = 0 \text{ or constant} \quad (\text{E-3})$$

The system is 'at rest' when all of the time derivatives are identically zero.

Steady-state aircraft flight can be defined as a condition in which all of the motion variables are constant or zero, and all acceleration components are zero. This definition is very restrictive, unless some simplifying assumptions are made. The definition allows steady wings-level flight and steady turning flight if the flat-Earth equations of motion are used, and if it is assumed that the aircraft's mass remains constant during the motions of consideration.

If the change in atmospheric density with altitude is neglected during the trim process, a wings-level climb and a climbing turn are permitted as steady-state flight conditions too. The equations for x_e , y_e , and H then do not couple back into the equations of motion and don't need to be used in finding a steady-state equation. So steady-state flight can be defined in terms of the remaining nine state variables of the flat-Earth equations:

$$\dot{p}, \dot{q}, \dot{r}, \dot{V}, \dot{\alpha}, \dot{\beta} = 0, \quad \mathbf{u} = \text{const.} \quad (\text{E-4})$$

where \mathbf{u} is the input vector to the aircraft model.



Additional constraints have to be made to define the flight condition. Here we consider steady wings-level flight, steady-turning flight, steady pull-up or push-over, and steady rolls, defined by the following constraints:

<i>steady wings-level flight:</i>	$\varphi, \dot{\varphi}, \dot{\theta}, \dot{\psi} = 0$	($\therefore p, q, r = 0$)
<i>steady turning flight:</i>	$\dot{\varphi}, \dot{\theta} = 0,$	$\dot{\psi} = \text{turn rate}$
<i>steady pull-up:</i>	$\varphi, \dot{\varphi}, \dot{\psi} = 0,$	$\dot{\theta} = \text{pull-up rate}$
<i>steady roll:</i>	$\dot{\theta}, \dot{\psi} = 0,$	$\dot{\varphi} = \text{roll rate}$

The conditions $\dot{p}, \dot{q}, \dot{r} = 0$ require the angular rates to be zero or constant, and therefore, the aerodynamic and thrust moments must be zero or constant. The conditions $\dot{V}, \dot{\alpha}, \dot{\beta} = 0$ require the aerodynamic forces to be zero or constant. Therefore, the steady-state pull-up (or push-over) and steady-state roll conditions can only exist instantaneously. Still, it is useful to trim the aircraft dynamics in such flight conditions (and use the resulting trim values of \mathbf{x} and \mathbf{u} to linearize the aircraft model in these flight conditions) because the control systems must operate there.

To find a steady-state flight condition, a set of nonlinear simultaneous equations, derived from the state model, must be solved. Due to the very complex functional dependence of the aerodynamic data, it is in general not possible to solve these equations analytically. Instead, a numerical algorithm must be used to iteratively adjust the independent variables until some solution criterion is met. The solution will be approximate, but can be made arbitrarily close to the exact solution by tightening up the criterion. Also, the solution may not be unique; for example, steady-state level flight at a given engine power level can correspond to two different airspeeds and angles of attack.

Our knowledge of aircraft behaviour makes it possible to specify the required steady-state condition so that the trim algorithm will converge on an appropriate solution. The trim algorithm presented here will deal with the aircraft model only through its input and output signals. It shall not work *within* the model to balance forces and moments separately. Hence, any aircraft model which uses the same input and state vectors as the model from this report can be trimmed with the same program; the internal structure of the aircraft model doesn't matter.

In the next three sections, first it will be shown how the steady-state condition can be specified, how many of the control variables may be chosen independently, and what constraints exist on the remaining variables. Then, an algorithm which solves the nonlinear equations numerically will be developed, and finally, this algorithm will be implemented in a computer program which can be applied to the 'Beaver' model from this report or any other model which uses the same states and inputs.

E.3 Specification of flight condition.

The (initial) values of the altitude, airspeed, and climb angle for steady-state flight can be specified by the user, within the limits imposed by the engine power. Assuming that the configuration of the aircraft (flap setting, landing gear up or down, etc.) is prespecified, it can be expected that a unique combination of control inputs and remaining state variables will exist. For the 'Beaver' model, the flap setting δ_f and the engine speed n will be prespecified. In general, it is not possible to determine any analytical constraints on the

remaining control variables δ_e , δ_a , δ_r , and p_z , so these control inputs must be adjusted by the numerical trim algorithm. This is not the case for all state variables.

The three position states can temporarily be eliminated from consideration, since the only relevant component of the position vector is the (initial) altitude H , which can be prespecified. For steady translational flight, the state variables φ , p , q , and r are identically zero and ψ can be selected freely by the user. This leaves V , α , β , and θ to be considered. The sideslip angle β must be adjusted by the trim algorithm to zero out the sideforce F_y , which leaves V , α , and θ . It is customary to impose a flightpath-angle constraint on the steady-state condition, so finally, the variables V , and γ remain to be specified by the user. In the next section, a general rate-of-climb constraint, which allows nonzero values of the roll gnale, will be given.

In steady-state turning flight, φ , p , q , and r will differ from zero. The turn can be specified by the yaw rate $\dot{\psi}$ or by the turn radius R ($\dot{\psi} = V/R$); the initial heading can still be specified freely. Then p , q , and r can be determined from the kinematic relations (2-3) or (A-61), given the attitude angles θ and φ . The required pitch angle θ can be obtained from the rate-of-climb constraint from the next section if the roll angle φ is known. It is possible to specify the roll angle freely, but then, in general, a significant sideslip angle β will occur and the turn will be a *skidding turn*. Therefore, a constraint for *coordinated turns* will be included, which will compute the roll and sideslip angles which lead to turns in which the aircraft is banked at an angle such that there is no component of the aerodynamic force along the Y_B -axis.

E.4 The rate-of-climb and turncoordination constraints.

In this section, the rate-of-climb and turncoordination constraints will be defined. Both constraints must be solved simultaneously, because the rate-of-climb constraint involves θ , and the turncoordination constraint involves both θ and φ .

According to ref.[29], the flight-condition must satisfy the following equation:

$$\sin \gamma = a \sin \theta - b \cos \theta \quad (\text{E-5})$$

where:

$$\begin{aligned} a &= \cos \alpha \cos \beta \\ b &= \sin \varphi \sin \beta + \cos \varphi \sin \alpha \cos \beta \end{aligned} \quad (\text{E-6})$$

Solving for θ , the resulting *rate-of-climb constraint* is found to be:

$$\tan \theta = \frac{ab + \sin \gamma \sqrt{a^2 - \sin^2 \gamma + b^2}}{a^2 - \sin^2 \gamma}, \quad \theta \neq \pm \frac{\pi}{2} \quad (\text{E-7})$$

The *coordinated turn constraint* can be written as:

$$\sin \varphi = G \cos \beta (\sin \alpha \tan \theta + \cos \alpha \cos \varphi) \quad (\text{E-8})$$



(ref.[29]). This equation must be used in conjunction with (E-7) to trim the aircraft for turning flight with a specified rate of climb. If the equations (E-7) and (E-8) are solved simultaneously, the only remaining variables to be adjusted by the numerical trim algorithm are the angle of attack and sideslip angle and the control inputs. According to ref.[29], the simultaneous solution equals:

$$\tan \varphi = G \frac{\cos \beta}{\cos \alpha} \frac{(\tilde{a} - b^2) + b \tan \alpha \sqrt{\tilde{c}(1 - b^2) + G^2 \sin^2 \beta}}{\tilde{a}^2 - b^2(1 + \tilde{c} \tan^2 \alpha)} \quad (\text{E-9})$$

where:

$$\tilde{a} = 1 - G \tan \alpha \sin \beta$$

$$b = \frac{\sin \gamma}{\cos \beta} \quad (\text{E-10})$$

$$\tilde{c} = 1 + G^2 \cos^2 \beta$$

and:

$$G = \frac{\dot{\psi}V}{g_0} \quad (\text{E-11})$$

The value of φ , given by (E-10) can be substituted in (E-7) to solve for v . For skidding turns, φ can be selected by the user, so then only the rate-of-climb constraint remains to be solved. When the flight-path angle is zero, (E-10) reduces to:

$$\tan \varphi = \frac{G \cos \beta}{\cos \alpha - G \sin \alpha \sin \beta} \quad (\text{E-12})$$

Now that we know the flight-path constraints, it is possible to develop a general aircraft-trim algorithm, which will be described next.

E.5 The steady-state aircraft trim algorithm.

The trim algorithm determines steady-state flight conditions by solving the nonlinear state equations for the state and control vectors such that the state derivatives \dot{V} , $\dot{\alpha}$, $\dot{\beta}$, \dot{p} , \dot{q} , and \dot{r} are identically zero. This will be done here by applying a numerical minimization algorithm to a scalar *cost function* J , which equals:

$$J = c_1 V^2 + c_2 \dot{\alpha}^2 + c_3 \dot{\beta}^2 + c_4 \dot{p}^2 + c_5 \dot{q}^2 + c_6 \dot{r}^2 \quad (\text{E-13})$$

where c_i , $i \in \{1, 2, \dots, 6\}$, are weighting constants. According to ref.[29], the *simplex* algorithm generally performs well on the aircraft-trim problem.

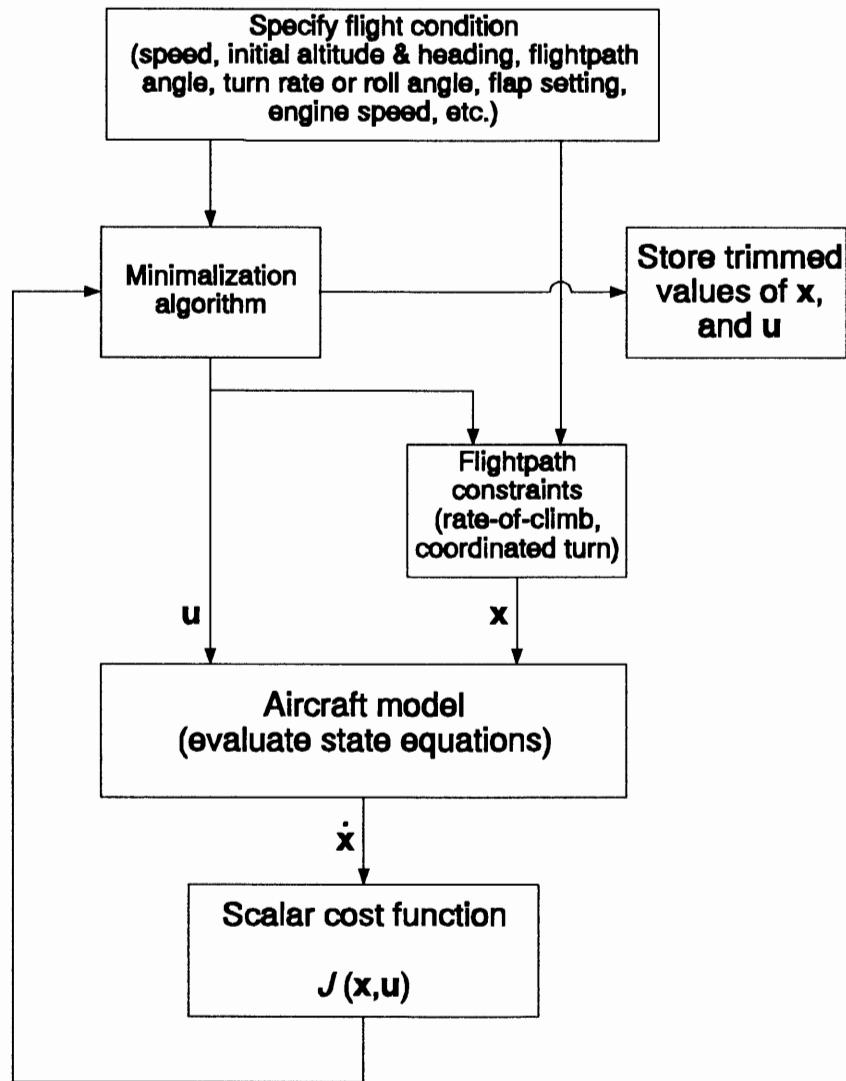


Figure E-1. Trim algorithm.

The trim program will be organized as shown in figure E-1. First, the flight condition must be specified. The trim program will make an initial guess for the independent state variables and the control variables which will be adjusted during the trim process. Then the minimization routine, which searches for the values of x and u which minimize the scalar cost function J , will be started. The elements of these vectors which are adjusted by the trim algorithm or constraints are updated once for every iteration step.

In each iteration, x and u are updated, using the flight-condition specification variables, the states which result from flightpath constraints and kinematic relations and the independent input and state variables which are adjusted by the optimization routine. The state equations are evaluated to find the time-derivative of the state vector, \dot{x} . Equation (E-13) then defines the value of J , which is returned to the minimization routine. A stop criterion, which depends upon the change of J between two iterations is used. Also, the maximum number of iterations is limited, so the process will stop if no minimum is found.

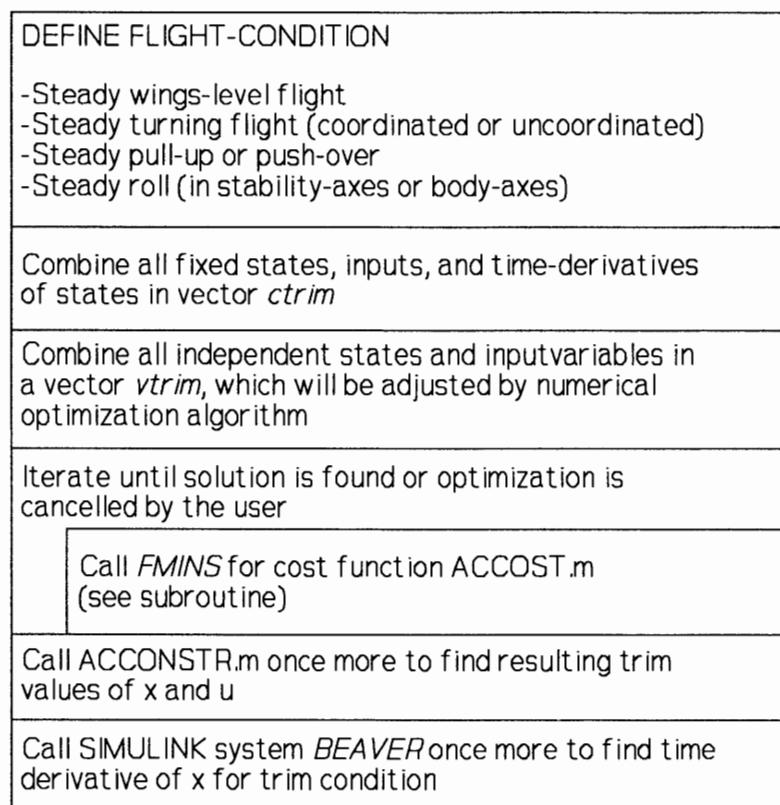


E.6 Implementation in MATLAB/SIMULINK: basic principles.

The trim algorithm can be implemented in MATLAB and applied to the aircraft models in SIMULINK in a straightforward way, using the MATLAB minimization routine *FMINS*, which is based upon the simplex algorithm. Figures E-2 to E-4 show the basic structure of the MATLAB implementation of the trim algorithm, ACTRIM.M ('aircraft trim'), and the subroutines ACCONSTR.M and ACCOST.M. The structure closely resembles the diagram from figure E-1.

First, a menu with possible flight conditions is displayed. The user must select between steady-state level flight, steady-state turns (coordinated or uncoordinated), steady pull-ups or push-overs, and steady body-axes or stability-axes roll. The user must then define those specification variables which do not follow automatically from the flight-condition. Two vectors will be created: one containing the user-specified values of the states, inputs, and some state derivatives, and another, containing the independent input and state variables which will be adjusted by the trim algorithm.

Next, the minimization routine *FMINS* starts searching for the value of the vector with independent variables which minimizes the cost function J , contained in the subroutine ACCOST.M. The flightpath constraints and kinematic relations are contained in another subroutine, called ACCONSTR.M, which is called every iteration from the cost function. ACCOST.M computes J by first computing the constraints, then calling the SIMULINK system *BEAVER* to find the current value of x_{dot} , and finally employing equation (E-13).



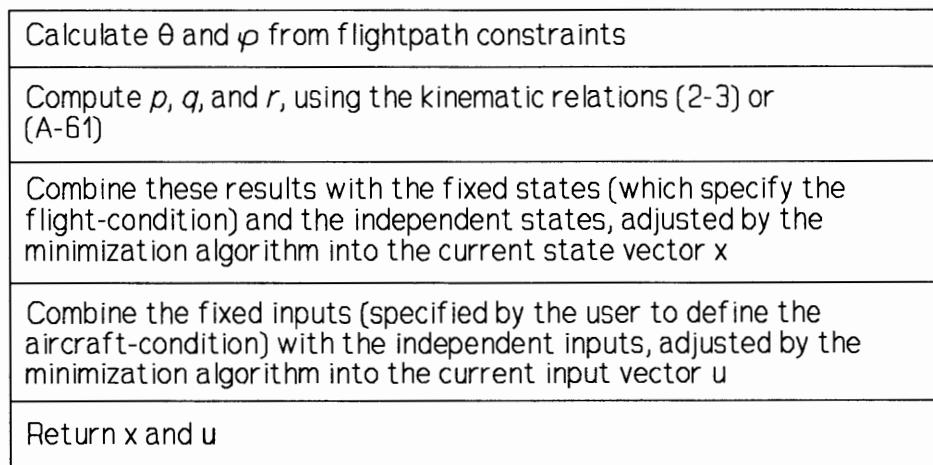
**Figure E-2. Program-structure diagram of ACTRIM.M
(main program of aircraft trim routines)**

In chapter 3, it was shown that time-derivatives of the state variables of any S-function can be extracted from the MATLAB command line with the following command:

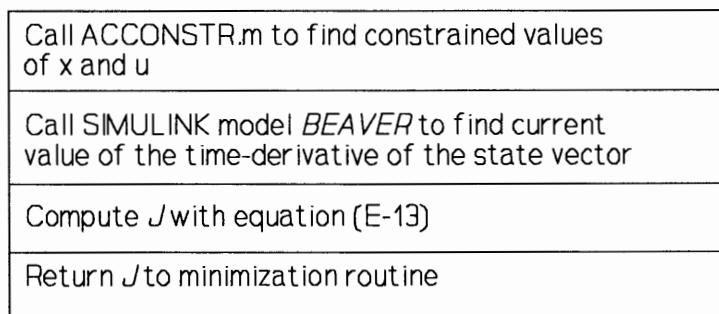
$$xdot = sfun(t, x, u, 1)$$

where *sfun* is the name of the S-function. Here, this command will be applied to the S-function *BEAVER* to find the value of *xdot*, which will be used to compute *J*. This clearly illustrates how customized analytical tools for S-functions can be constructed in practice.

The program listing of ACTRIM, ACCOST, and ACCONSTR is contained in section F.2.7 of appendix F, which also contains a list of variables.



**Figure E-3. Program-structure diagram of ACCONSTR.M
(subroutine with flightpath constraints and kinematic relations)**



**Figure E-4. Program-structure diagram of ACCOST.M
(subroutine with cost function for aircraft trim)**



E.7 Conclusions.

A general aircraft trim algorithm has been derived in this appendix. It can be implemented straightforwardly as a MATLAB routine, which uses the SIMULINK system *BEAVER* to calculate the time-derivatives of the state variables. The trim routine can be used to find steady-state level-flight conditions, steady turns (skidding or coordinated), steady pull-ups (or push-overs), and steady rolls. It can be applied to every model which uses the same inputs and states as the system *BEAVER*, independently from the internal structure of the aircraft model. Moreover, it shouldn't be too difficult to change the programs for application to aircraft models which use other state or input vectors.



Appendix F. The SIMULINK simulation models.

F.1 Introduction.

In chapter 4, the overall structure of the simulation models was outlined. This appendix will zoom in further, to show all details about the SIMULINK implementation of the mathematical models of the aircraft dynamics, wind and turbulence, and navigation signals from VOR and ILS equipment. The aircraft dynamics model forms the heart of the model library. Although some of its sub-models are valid only for the DHC-2 'Beaver' aircraft, the model itself is such general, and the models are such modular that it is relatively easy to adapt and update the 'Beaver' model for other aircraft. Don't be impressed too much by the very detailed model description in this appendix or by the sheer size of it! To use the models in practice, it is often not necessary to know this appendix in detail. If you just want to use the existing model without making any changes, reading chapters 4 and 6 will probably suffice.

There are three main model-libraries: the aircraft dynamics library, an ILS/VOR library, and a wind/turbulence library. The aircraft dynamics library contains blocks with state equations and many output equations, see section F.2. The ILS/VOR library will be described in section F.3. It contains some blocks to compute navigation signals from ILS and VOR equipment, which may be used for the assessment of automatic controllers. The wind/turbulence library (section F.4) contains blocks to evaluate the performance of the aircraft and controllers in non-steady atmosphere.

In the future, models of other navigation tools and external disturbances need to be added to this list. Also, standardized lists with models of sensor and/or actuator dynamics need to be implemented in SIMULINK. Almost every dynamical model, developed in the Disciplinary Group for Stability and Control can be implemented in SIMULINK, where it can be connected to other systems. This will lead to more insight in the actual behaviour of the systems, and to better control design solutions.

Section F.2.7 contains a description of a numerical trim routine for the 'Beaver' model, which searches for steady-state equilibrium values of the state and input variables. It can be applied to any aircraft model in SIMULINK which uses the same definitions of the state and input vectors as the 'Beaver' model. Together with the linearization routines of SIMULINK, this trim routine will further integrate linear system analysis with MATLAB and nonlinear system analysis with SIMULINK.

In appendices A to C and chapter 2, theoretical backgrounds to the models are given. Chapter 6 and part II of this report both illustrate how the models can be used in practice. A description of the implementation of the control laws of the 'Beaver' autopilot in SIMULINK can be found in Part II of this report. The models, helpfiles, and examples have been stored on a floppy disk which should accompany this report. If this floppy is not present, contact the author (see address at cover page).

Appendix H describes how to install all models, programs, and help texts on your own computer!



F.2 Nonlinear model of the aircraft dynamics: state and output equations.

F.2.1 Introduction.

The aircraft dynamics model contains the nonlinear Ordinary Differential Equations (ODEs) for the twelve state variables, all equations which must be solved to calculate the external forces and moments, and a large number of additional output equations, which are not really necessary for solving the state equations, but which can be applied for many different applications involving flight dynamics analysis. It is not difficult to enhance this list of output variables even further.

The equations have been implemented as SIMULINK *Function* blocks. Small sets of interrelated variables have been grouped together into a limited number of 'basic' blocks, which together build the whole simulation model. To prevent the system from becoming really inextricable, a limited number of output vectors are used. Every output vector is created by one 'basic' block, which *Muxes* different output variables into one output vector. Some 'basic' blocks use output vectors from other blocks as inputs, in which case often only a few elements are actually used in the equations of that block. This seems somewhat confusing, but it keeps the number of output lines within reasonable limits.

To add new blocks to the aircraft model, one should know exactly which variables are calculated by the other blocks, but how that is done internally doesn't really matter, as long as the results are correct. For this reason, all 'basic' blocks have been *Masked*, thus hiding their internal structure from the user (*black-box approach*). See section E.8 of appendix E for more details about the *Mask* option.

Hiding the internal structure from the user is particularly useful for implementing aircraft-dependent blocks such as the aerodynamic model of an aircraft. For instance: the aerodynamic model of the 'Citation 500' aircraft as used in the DUT-flightimulator, interpolates multi-dimensional tables to obtain the aerodynamic forces and moments, whereas the 'Beaver' model expresses these forces and moments as nonlinear polynomials. Both models can be implemented in 'basic' blocks, which look exactly the same from the outside, but completely different from the inside.

This section will show the internal structure of all 'basic' blocks from the 'Beaver' model, which already has been outlined in general terms in chapter 4. First, a list of all variables will be given. Then, the internal structure of all blocks will be shown and explained. Finally, some routines to define model parameters and routines to compute steady-state trim values or define arbitrary initial conditions will be presented.

F.2.2 List of variables, used for the aircraft state and output models.

Although the system itself has been built up as a block-diagram, still a standardized list of acronyms has been used within the block-diagrams of the aircraft model, the help files, etc. to make the diagrams better readable for the user. Note that the SIMULINK *Function* blocks use u as standard input variable. If u is a vector - this is true for most *Function* blocks of the 'Beaver' model - the

elements of the vectors are denoted as $u[1]$, $u[2]$, . . . within the *Function* blocks (yes, those brackets are square!). The following variables or acronyms appear within the helpfiles, block-diagrams, and *Mask* initialization lines of the 'Beaver' model¹⁾:

a	a	= speed of sound; [m/s]
alpha	α	= angle of attack; [rad]
alphadot	$\dot{\alpha}$	= time derivative of angle of attack; [rad/s]
Ax	A_x	= specific force along X_B -axis in c.g.; [g]
axk	$a_{x,k}$	= acceleration along X_B -axis in c.g.; [g]
Ay	A_y	= specific force along Y_B -axis in c.g.; [g]
ayk	$a_{y,k}$	= acceleration along Y_B -axis in c.g.; [g]
Az	A_z	= specific forces along Z_B -axis in c.g.; [g]
azk	$a_{z,k}$	= acceleration along Z_B -axis in c.g.; [g]
beta	β	= sideslip angle; [rad]
betadot	$\dot{\beta}$	= time derivative of sideslip angle; [rad/s]
chi	χ	= azimuth angle; [rad]
Cla	C_{l_a}	= aerodynamic moment coefficient around X_B -axis; [-]
Clt	C_{l_t}	= engine moment coefficient around X_B -axis; [-]
Cma	C_{m_a}	= aerodynamic moment coefficient around Y_B -axis; [-]
Cmt	C_{m_t}	= engine moment coefficient around Y_B -axis; [-]
Cna	C_{n_a}	= aerodynamic moment coefficient around Z_B -axis; [-]
Cnt	C_{n_t}	= engine moment coefficient around Z_B -axis; [-]
CXa	C_{X_a}	= aerodynamic moment coefficient along X_B -axis; [-]
CXt	C_{X_t}	= engine moment coefficient along X_B -axis; [-]
CYa	C_{Y_a}	= aerodynamic moment coefficient along Y_B -axis; [-]
CYt	C_{Y_t}	= engine moment coefficient along Y_B -axis; [-]
CZa	C_{Z_a}	= aerodynamic moment coefficient along Z_B -axis; [-]
CZt	C_{Z_t}	= engine moment coefficient along Z_B -axis; [-]
deltaa	δ_a	= aileron deflection; [rad]
deltae	δ_e	= elevator deflection; [rad]
deltaf	δ_f	= flap setting; [rad]
deltar	δ_r	= rudder deflection; [rad]
dpt	dpt	= dimensionless increase of pressure across the propeller; [-]
fpa	fpa	= flightpath acceleration; [m/s^2]
Fx	F_x	= resultant force along the X_B -axis; [N]
Fy	F_y	= resultant force along the Y_B -axis; [N]
Fz	F_z	= resultant force along the Z_B -axis; [N]
g	g	= acceleration of gravity; [m/s^2]
gamma	γ	= flightpath angle; [rad]
H	H	= altitude of the aircraft; [m]
Hdot	\dot{H}	= rate of climb; [m/s]
L	L	= resultant moment around the X_B -axis; [N]

¹⁾ It is sometimes possible that the same names are used for different variables, because most variables are only *locally* available within the different subsystems. Notice also that MATLAB can distinguish upper-case from lower-case variables!

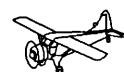


M	M	= resultant moment around the Y_B -axis; [N]
M	M	= Mach number; [-]
mu	μ	= dynamic viscosity; [$\text{kg}/\text{m}\cdot\text{s}$]
n	n	= engine speed; [RPM]
N	N	= resultant moment around the Z_B -axis; [N]
p	p	= angular velocity around the X_B -axis; [rad/s]
P	P	= engine power; [N·m/s]
pb_2V	$\frac{pb}{2V}$	= dimensionless angular velocity around X_B -axis; [-]
pdot	\dot{p}	= time derivative of roll rate; [rad/s ²]
phi	ϕ	= roll angle; [rad]
Phi	Φ	= bank angle; [rad]
phidot	$\dot{\phi}$	= time derivative of roll angle; [rad]
ps	p_s	= static pressure; [N/m ²]
psi	ψ	= yaw angle; [rad]
psidot	$\dot{\psi}$	= time derivative of yaw angle; [rad/s]
pz	p_z	= manifold pressure; ["Hg]
q	q	= angular velocity around the Y_B -axis; [rad/s]
qc	q_c	= impact pressure; [N/m ²]
qc_V	$\frac{qc}{V}$	= dimensionless angular velocity around Y_B -axis; [-]
qdot	\dot{q}	= time derivative of pitch rate; [rad/s ²]
qdyn	q_{dyn}	= dynamical pressure; [N/m ²]
r	r	= angular velocity around the Z_B -axis; [rad/s]
rb_2V	$\frac{rb}{2V}$	= dimensionless angular velocity around Z_B -axis; [-]
Rc	R_c	= Reynolds number with respect to \bar{c}
rdot	\dot{r}	= time derivative of yaw rate; [rad/s ²]
Re	R_e	= Reynolds number per unit length; [m ⁻¹]
rho	ρ	= air density; [kg/m ³]
T	T	= air temperature; [K]
theta	θ	= pitch angle; [rad]
thetadot	$\dot{\theta}$	= time derivative of pitch angle; [rad/s]
Tt	T_t	= total temperature; [K]
u	u	= component of V along X_B -axis; [m/s]
udot	\dot{u}	= time derivative of u ; [m/s ²]
uw	u_w	= component of wind velocity along X_B -axis; [m/s]
uwdot	\dot{u}_w	= time derivative of u_w ; [m/s ²]
v	v	= component of V along Y_B -axis; [m/s]
V	V	= true airspeed; [m/s]
Vc	V_c	= calibrated airspeed; [m/s]
vdot	\dot{v}	= time derivative of v ; [m/s ²]
Vdot	\dot{V}	= time derivative of true airspeed; [m/s ²]
Ve	V_e	= equivalent airspeed; [m/s]
vw	v_w	= component of wind velocity along Y_B -axis; [m/s]
vwdot	\dot{v}_w	= time derivative of v_w ; [m/s ²]
w	w	= component of V along Z_B -axis; [m/s]

wdot	\dot{w}	= time derivative of w ; [m/s ²]
ww	w_w	= component of wind velocity along Z_B -axis; [m/s]
wwdot	\dot{w}_w	= time derivative of w_w ; [m/s ²]
Xa	X_a	= aerodynamic force along X_B -axis; [N]
xe	x_e	= X-coordinate of aircraft relatively to F_E ; [m]
xedot	\dot{x}_e	= time derivative of x_e ; [m/s]
Xgr	X_{gr}	= gravity force along X_B -axis; [N]
Xt	X_t	= engine force along X_B -axis; [N]
Xw	X_w	= wind force along X_B -axis; [N]
Ya	Y_a	= aerodynamic force along Y_B -axis; [N]
ye	y_e	= Y-coordinate of aircraft relatively to F_E ; [m]
yedot	\dot{y}_e	= time derivative of y_e ; [m/s]
Ygr	Y_{gr}	= gravity force along Y_B -axis; [N]
Yt	Y_t	= engine force along Y_B -axis; [N]
Yw	Y_w	= wind force along Y_B -axis; [N]
Za	Z_a	= aerodynamic force along Z_B -axis; [N]
ze	z_e	= Z-coordinate of aircraft relatively to F_E ; [m]
Zgr	Z_{gr}	= gravity force along Z_B -axis; [N]
Zt	Z_t	= engine force along Z_B -axis; [N]
Zw	Z_w	= wind force along Z_B -axis; [N]

The following input and output vectors are used for the different blocks from the blocklibrary *ACLIB* and within the system *BEAVER*:

Ca	C_a	= [$C_{X_a} \ C_{y_a} \ C_{z_a} \ C_{l_a} \ C_{m_a} \ C_{n_a}$] ^T
Ct	C_t	= [$C_{X_t} \ C_{y_t} \ C_{z_t} \ C_{l_t} \ C_{m_t} \ C_{n_t}$] ^T
Fa	F_a	= [$X_a \ Y_a \ Z_a \ L_a \ M_a \ N_a$] ^T
Fgr	F_{gr}	= [$X_{gr} \ Y_{gr} \ Z_{gr}$] ^T
Ft	F_t	= [$X_t \ Y_t \ Z_t \ L_t \ M_t \ N_t$] ^T
Ftot	F_{tot}	= [$F_x \ F_y \ F_z$] ^T
Fw	F_w	= [$X_w \ Y_w \ Z_w$] ^T
Mtot	M_{tot}	= [$L \ M \ N$] ^T
ua	u_a	= [$\delta_e \ \delta_a \ \delta_r \ \delta_f$] ^T (inputs to the aerodynamic model)
ut	u_t	= [$n \ p_z$] ^T (inputs to the engine model)
uwind	u_{wind}	= [$u_w \ v_w \ w_w \ \dot{u}_w \ \dot{v}_w \ \dot{w}_w$] ^T (wind and turbulence inputs)
x	x	= [$V \ \alpha \ \beta \ p \ q \ r \ \psi \ \theta \ \varphi \ x_e \ y_e \ z_e$] ^T (state vector)
xdot	\dot{x}	= [$\dot{V} \ \dot{\alpha} \ \dot{\beta} \ \dot{p} \ \dot{q} \ \dot{r} \ \dot{\psi} \ \dot{\theta} \ \dot{\varphi} \ \dot{x}_e \ \dot{y}_e \ \dot{H}$] ^T
yacc	y_{acc}	= [$A_x \ A_y \ A_z \ a_{x,k} \ a_{y,k} \ a_{z,k}$] ^T
yad1	y_{ad1}	= [$a \ M \ q_{dyn}$] ^T
yad2	y_{ad2}	= [$q_c \ V_e \ V_c$] ^T
yad3	y_{ad3}	= [$T_t \ R_e \ R_c$] ^T
yatm	y_{atm}	= [$\rho \ p_s \ T \ \mu \ g$] ^T
ybvel	y_{bvel}	= [$u \ v \ w$] ^T
ydl	y_{dl}	= [$\frac{pb}{2V} \ \frac{qc}{V} \ \frac{rb}{2V}$] ^T
yeul	y_{eul}	= [$\dot{\psi} \ \dot{\theta} \ \dot{\phi}$] ^T
yfp	y_{fp}	= [$\gamma \ fpa \ \chi \ \Phi$] ^T



y_{hlp}	\mathbf{y}_{hlp}	$= [\cos(\alpha) \sin(\alpha) \cos(\beta) \sin(\beta) \tan(\beta) \sin(\psi) \cos(\psi) \dots \sin(\theta) \cos(\theta) \sin(\phi) \cos(\phi)]^T$
y_{pow}	\mathbf{y}_{pow}	$= [dpt \ P]^T$
y_{pqr}	\mathbf{y}_{pqr}	$= [\dot{p} \ \dot{q} \ \dot{r}]^T$
y_{uvw}	\mathbf{y}_{uvw}	$= [\dot{u} \ \dot{v} \ \dot{w}]^T$
y_{vab}	\mathbf{y}_{vab}	$= [\dot{V} \ \dot{\alpha} \ \dot{\beta}]^T$
y_{xyh}	\mathbf{y}_{xyh}	$= [\dot{x}_e \ \dot{y}_e \ \dot{H}]^T$

The data matrix AM, which can be loaded from the file AEROMOD.MAT, contains all parameters of the aerodynamic model of the 'Beaver', see equations (2-5) of chapter 2. The parameters of the 'Beaver' engine forces and moments model (equations (2-10), chapter 2) have been included in the matrix EM, which has been stored in the file ENGMOD.MAT. The matrices GM1 and GM2, which can be loaded from the file AIRCRAFT.MAT, contain data about the aircraft geometry and inertia parameters (GM stands for 'Geometry and Mass').

The MAT-files must be loaded into the MATLAB workspace before starting a simulation by calling the function *LOADER*. If the MAT-files are not present in the workdirectory, it is necessary to run *MODBUILD* first. The MATLAB functions *LOADER* and *MODBUILD* will be described in section F.2.6. The data matrices are only valid for the models of the 'Beaver'. The aircraft geometry and mass distribution are considered to remain constant during the motions of interest. The definition of these matrices can be found in tables F-1 to F-3 at the end of this appendix.

F.2.3 The aircraft model library and the complete aircraft model.

All basic blocks of the aircraft model have been logically stored in the library *ACLIB* ('AC' = 'aircraft'). This library includes some examples which illustrate how different blocks can be combined into 'groups'. The system *BEAVER* contains the complete aircraft model, which results after connecting all blocks and groups from the library *ACLIB* to eachother. Here, we'll use the system *BEAVER* as a starting point for the discussion of all blocks from *ACLIB*, since this immediately explains the relations between the different blocks. A list with available blocks in *BEAVER* and *ACLIB* is included in table F-4 at the end of this chapter. Note: *ACLIB1* to *ACLIB8* are sublibraries of *ACLIB*.

F.2.4 First level of the simulation model *BEAVER*.

Figure F-1 shows the first level of the system *BEAVER*, which will be called the *main level* in the figures. This level is used to arrange the inputs and outputs in such way that other systems can be connected to *BEAVER*. The large subsystem block, called *Beaver dynamics and output equations*, contains the actual aircraft model, i.e., the state and output equations themselves.

All outputs from the subsystem *Beaver dynamics and output equations* are sent to the matrix *Out* in the MATLAB workspace (notice the use of upper-case!). The inputs to this subsystem are sent to the matrix *In*. See table F-5 at the end of this report for the current definitions of *In* and *Out* (it is quite easy to change these definitions if more output equations are added or if the outputs are reshuffled), or type *help inputs* or *help outputs* at the MATLAB command line. The system *BEAVER* also contains a *Clock* block, connected to a *To Workspace* block, which will create a time-axis in the MATLAB workspace for plotting

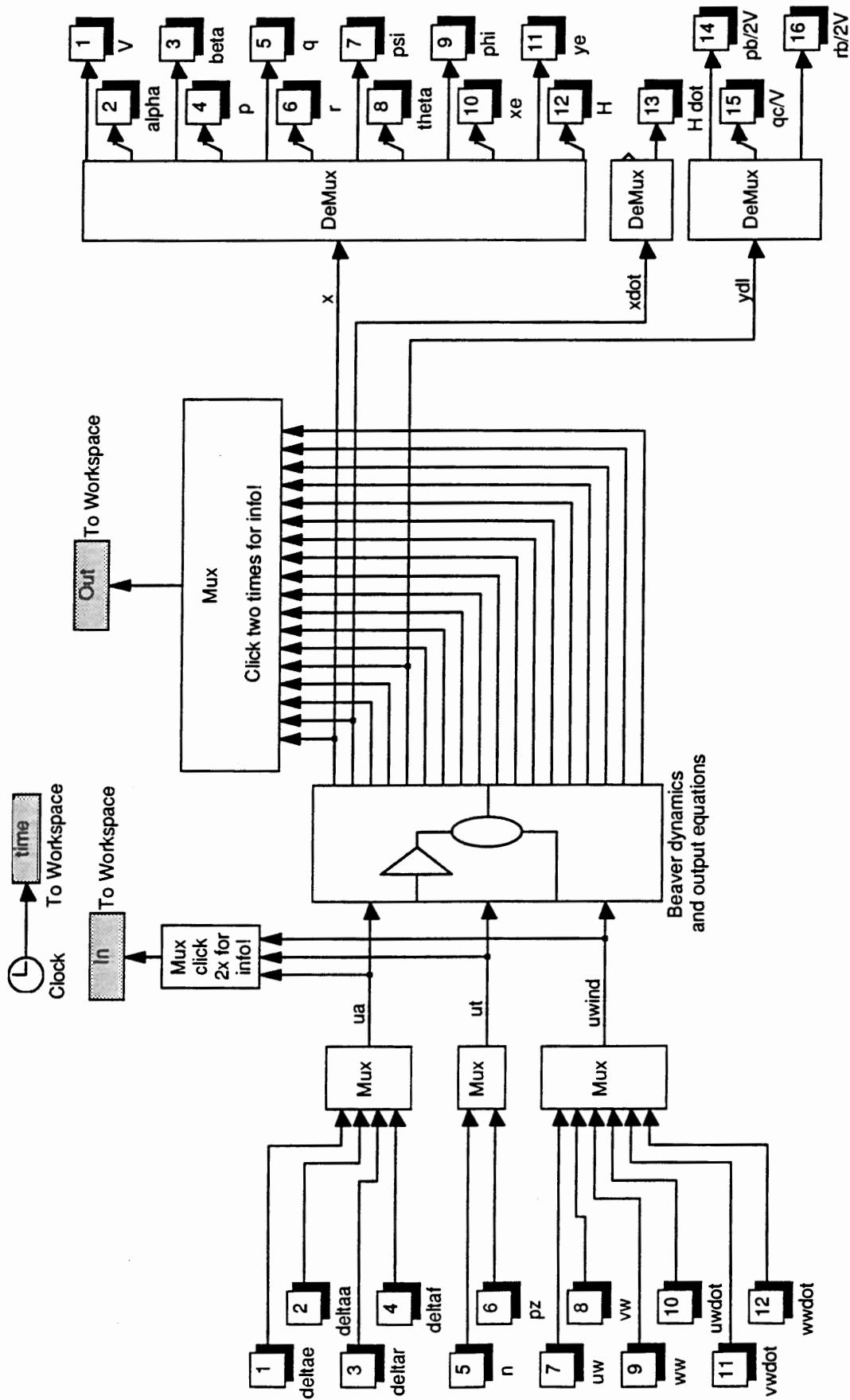
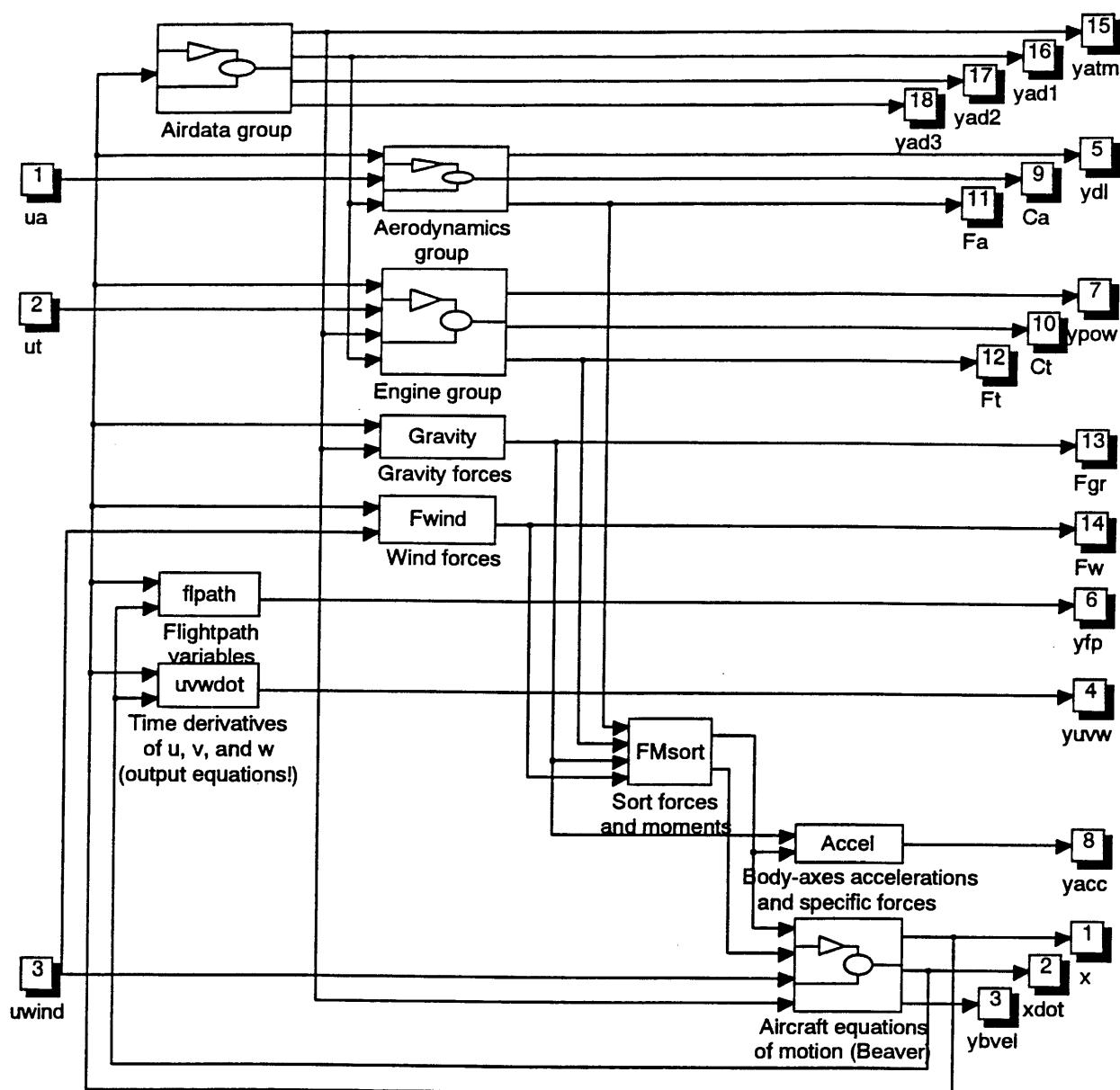


Figure F-1. Main level of the system *BEAVER*.
Main





**Figure F-2. Internal structure of the subsystem block
Beaver dynamics and output equations.**

Main | Beaver dynamics

purposes. The time values are sent to the vector *time* in the MATLAB workspace.

A limited number of output signals have been connected to *Outport* blocks (the shaded square blocks which have been numbered sequentially), which makes it possible to connect these signals to other systems. Notice that it is not possible to connect a vector line to a single *Outport* block; output vectors need to be demultiplexed with a *DeMux* block first! Here, sixteen output signals have been coupled to *Outport* blocks. These signals are:

V , α , β , p , q , r , ψ , θ , φ , x_e , y_e , H , and \dot{H} (these variables are used by the control laws of the 'Beaver' autopilot), and:

$\frac{pb}{2V}$, $\frac{qc}{V}$, and $\frac{rb}{2V}$ (these variables are used by the models of the Flight Control System of the 'Beaver', see part II)

In this report, only the output variables mentioned above will be connected to other subsystems. If the system *BEAVER* is to be used for other purposes, it might be necessary to include other *Outport* blocks to the first level of the system *BEAVER*. From now on, the variables which have been connected to *Outport* blocks in the first level of the system *BEAVER* will be called *S-function outputs*. Notice that the number of *S-function outputs* is much smaller than the number of outputs which are sent to the matrix *Out* in the MATLAB workspace. The latter variables can not be used by other subsystems on-line during the simulation!

The number of *S-function outputs* must be in accordance with the number of inputs of subsystems which have to be connected to *BEAVER*. The system *BEAVER* can be connected to other systems with an *S-function* block from the SIMULINK *Nonlinear* library (see appendix E). Chapter 6 and part II of this report contain some practical examples which illustrate the inclusion of *BEAVER* within other systems.

There are twelve input variables: four aerodynamic inputs (δ_e , δ_a , δ_r , and δ_f) which are combined into the vector \mathbf{u}_a , two engine inputs (n , and p_z) which are combined into the vector \mathbf{u}_e , three wind velocity components along the body axes of the aircraft (u_w , v_w , and w_w), which form the first half of the vector \mathbf{u}_w , and the three time derivatives of these wind velocity components, which form the second half of \mathbf{u}_w .

Table F-5 contains a list of all input variables, which are stored in the vector *In*, all output variables which are stored *Out*, and all variables that have been coupled to *Outport* blocks (the *S-function outputs*). The variables which are sent to the vectors *In* and *Out* can be transferred to *S-function outputs* if they are also connected to *Outport* blocks.

F.2.5 Beaver dynamics and output equations (second level of *BEAVER*).

Figure F-2 shows the general structure of the 'Beaver' simulation model. There are a number of *subsystem* blocks and a number of blocks with the blockname displayed within the block itself. The latter ones will be called *basic blocks* in this appendix. The internal structure of the basic blocks cannot be accessed by



the user directly, unless they are 'unmasked'¹⁾. We will now zoom in to the different blocks and groups which are visible in figure 5-2.

a. Airdata Group.

Figure F-3 shows the internal structure of the subsystem block *Airdata Group*. This subsystem contains the following basic blocks:

- 1 - Atmosph. In this block, some atmosphere variables are calculated, using the US Standard Atmosphere model. Although the acceleration of gravity is assumed to be constant ($g = g_0$), g has been included in the outputvector which makes it possible to implement g as a function of altitude in the future. Type *help atmosph* at the MATLAB command line for on-line help. The structure of *Atmosph* is shown in figure F-4.

Inputvector: \mathbf{x}
 Outputvector: \mathbf{y}_{atm}

- 2 - Airdata1. In this block, the dynamic pressure q_{dyn} , the speed of sound a , and the Mach number M are calculated. Type *help airdata1* at the MATLAB command line for on-line help. The internal structure of *Airdata1* is shown in figure F-5.

Inputvectors: \mathbf{x} and \mathbf{y}_{atm}
 Outputvector: \mathbf{y}_{ad1}

- 3 - Airdata2. This block is used to calculate the impact pressure q_c , the equivalent airspeed V_e , and the calibrated airspeed V_c . Type *help airdata2* at the MATLAB command line for on-line help. Figure F-6 shows the internal structure of *Airdata2*.

Inputvectors: \mathbf{y}_{atm} and \mathbf{y}_{ad1}
 Outputvector: \mathbf{y}_{ad2}

- 4 - Airdata3. This block calculates the total temperature T_t and the Reynolds numbers R_e and R_c . Type *help airdata3* at the MATLAB command line for on-line help. The structure of *Airdata3* is shown in figure F-7.

Inputvectors: \mathbf{x} , \mathbf{y}_{atm} , and \mathbf{y}_{ad1}
 Outputvector: \mathbf{y}_{ad3}

The variable *cbar*, used by *Airdata3* is defined in the Mask-definition line:

```
cbar = GM1(10);
```

See table F-3 for the definition of GM1.

¹⁾ See section E.8 of appendix E for more details about the *Mask* utility.

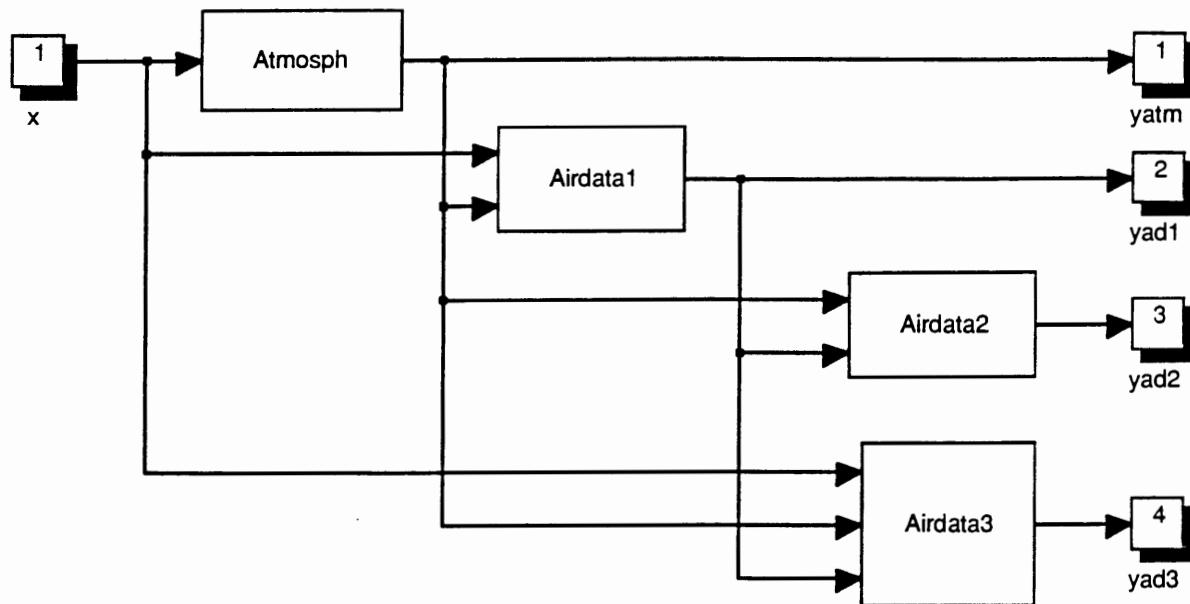


Figure F-3. Internal structure of the subsystem block *Airdata Group*.
Main | Beaver dynamics | Airdata Group

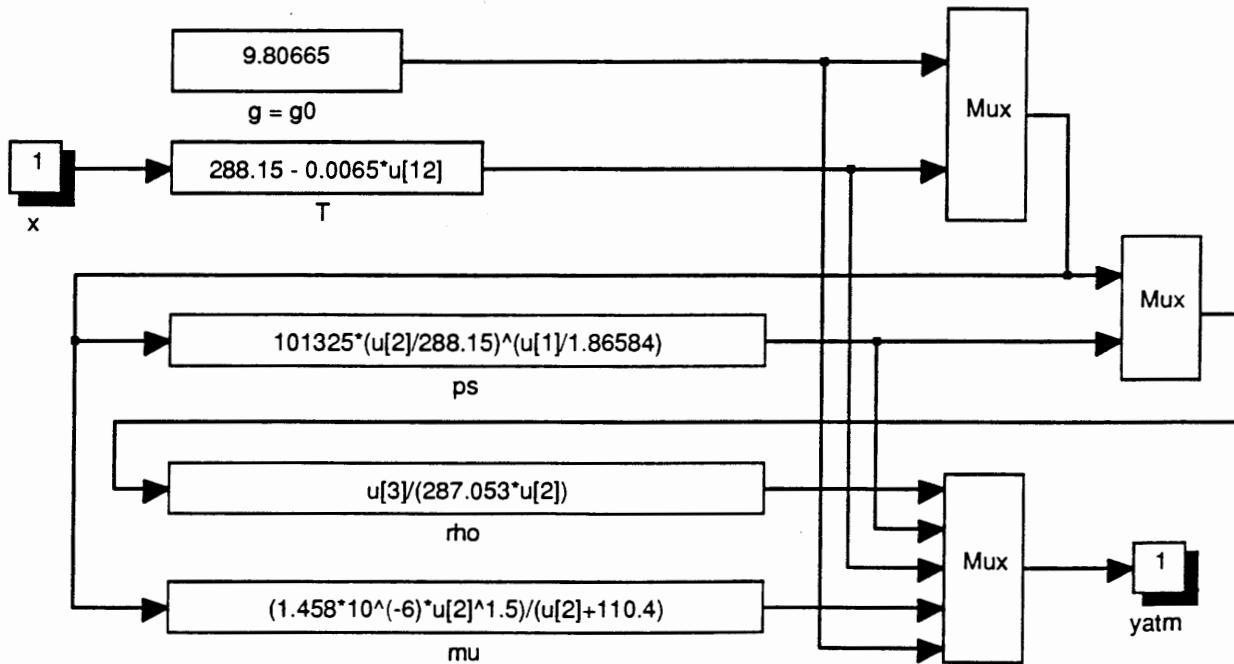


Figure F-4. Internal structure of the 'basic' block *Atmosph*.
Main | Beaver dynamics | Airdata Group | Atmosph



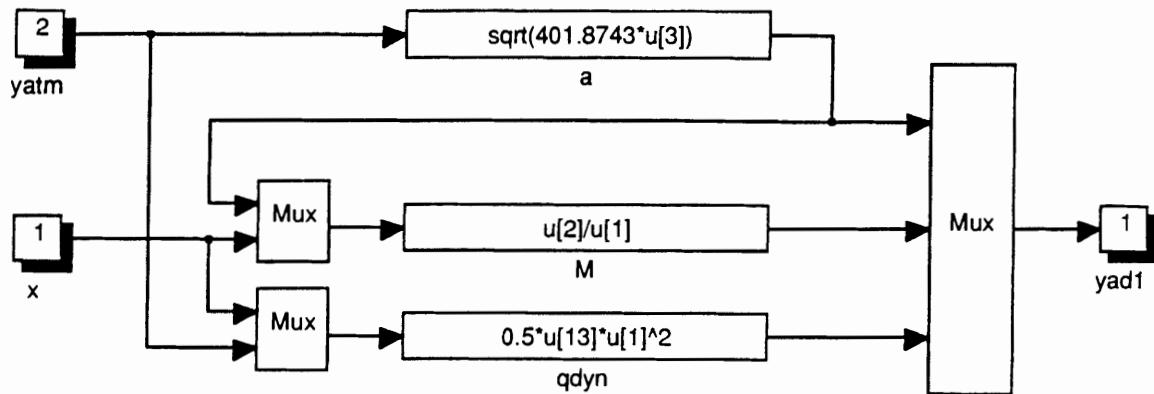


Figure F-5. Internal structure of the 'basic' block Airdata1.

Main | Beaver dynamics | Airdata Group | Airdata1

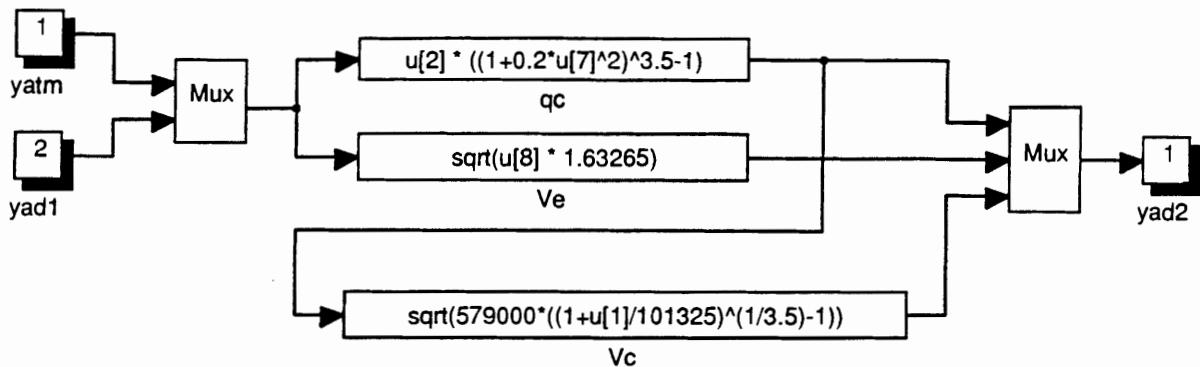


Figure F-6. Internal structure of the 'basic' block Airdata2.

Main | Beaver dynamics | Airdata Group | Airdata2

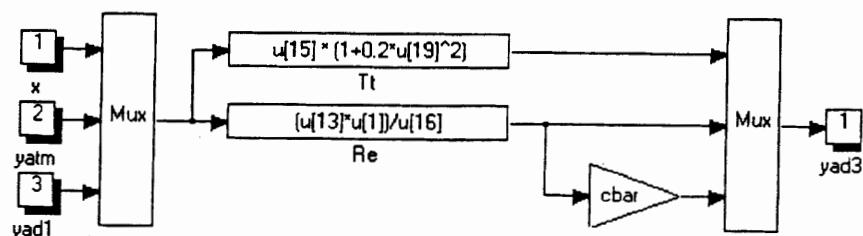


Figure F-7. Internal structure of the 'basic' block Airdata3.

Main | Beaver dynamics | Airdata Group | Airdata3

b. Aerodynamics Group.

Figure F-8 shows the internal structure of the subsystem *Aerodynamics Group*, which contains the following basic blocks:

- 1 - **Dimless.** This block is used to calculate the dimensionless rotational velocities along the aircraft's body-axes. Type *help dimless* for on-line help at the MATLAB command line. Figure F-9 shows the internal structure of *Dimless*.

Inputvector: \mathbf{x}
 Outputvector: \mathbf{y}_{dl}

The variables b and $cbar$ have been defined in the Mask-definition line:

```
cbar = GM1(1); b = GM1(2);
```

See table F-3 at the end of this appendix for the definition of GM1.

- 2 - **Aeromod (Beaver).** In this block the aerodynamic forces and moment coefficients along the body axes are calculated. This aerodynamic model is valid for the DHC-2 'Beaver' aircraft, see equations (2-5) in section 2.2.2 and ref.[30]. To implement another aerodynamic model which expresses the aerodynamic forces in terms of aerodynamic lift, drag, and sideforce, instead of forces along the body axes, it is necessary to make some changes in the equations for V , α , and β , which are contained in the block *Equations of motion\State derivatives\Vabdot*. The appropriate equations have been derived in appendix A.

On-line help is available at the MATLAB command line (type *help aeromod*). See figure F-10 for the internal structure of *Aeromod (Beaver)*.

Inputvectors: \mathbf{x} , \mathbf{u}_a , and \mathbf{y}_{dl}
 Outputvector: \mathbf{C}_a

The block *Matrix product* has been created with the SIMULINK *Mask* utility. This block has been stored in the library *TOOLS*, containing some useful *Masked* SIMULINK blocks which are not included in the standard SIMULINK libraries. Here, the *Matrix product* is used to multiply the help vector \mathbf{y}_{tmp} which leaves the *Mux* block:

$$\mathbf{y}_{tmp} = [1 \ \alpha \ \alpha^2 \ \alpha^3 \ \beta \ \beta^2 \ \beta^3 \ \frac{pb}{2V} \ \frac{qc}{V} \ \frac{rb}{2V} \ \dots \ \dots \ \delta_e \ \delta_f \ \delta_a \ \delta_r \ \alpha \cdot \delta_f \ \alpha \cdot \delta_r \ \alpha \cdot \delta_a \ \beta^2 \cdot \delta_e \ 0]^T$$

with the matrix *AM* to obtain the aerodynamic force and moment coefficients. *AM* contains all stability and control derivatives of the nonlinear aerodynamic model of the 'Beaver', see table F-1 at the end of this chapter for its definition.

- 3 - **FMdims.** This block ('Force and Moment dimensions') is used to calculate non-dimensionless forces and moments from the dimensionless coefficients.



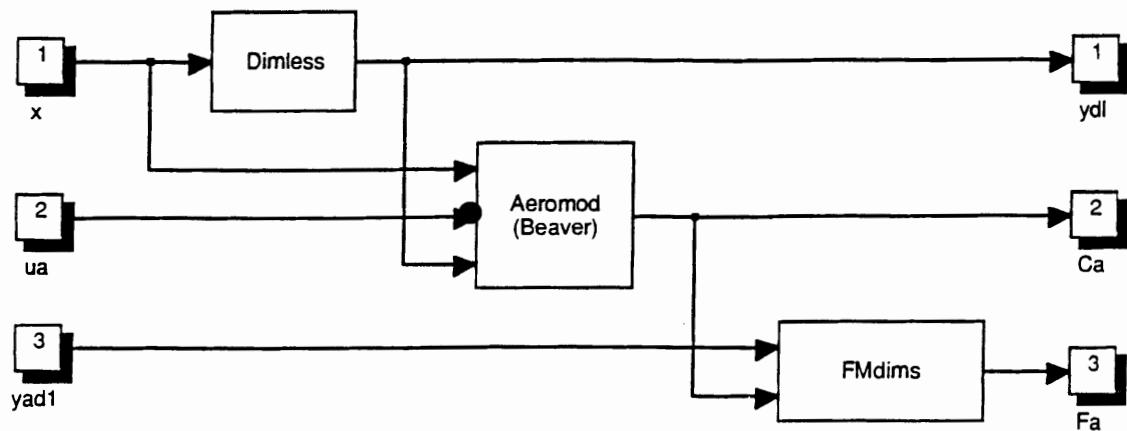


Figure F-8. Internal structure of the subsystem block Aerodynamics Group.
[Main](#) | [Beaver dynamics](#) | [Aerodynamics Group](#)

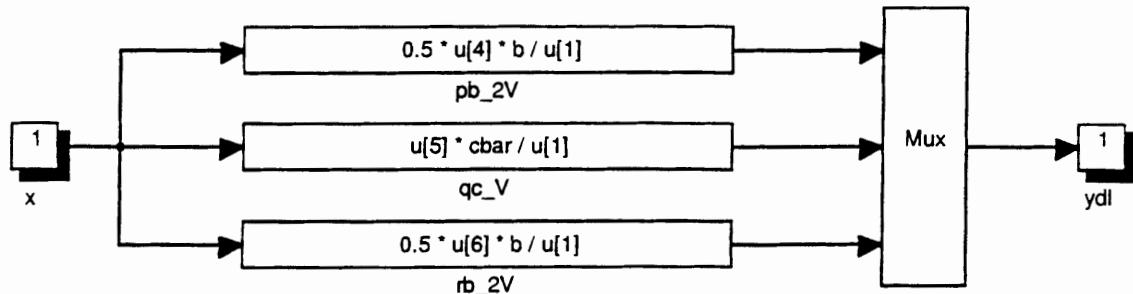


Figure F-9. Internal structure of the 'basic' block Dimless.
[Main](#) | [Beaver dynamics](#) | [Aerodynamics Group](#) | [Dimless](#)

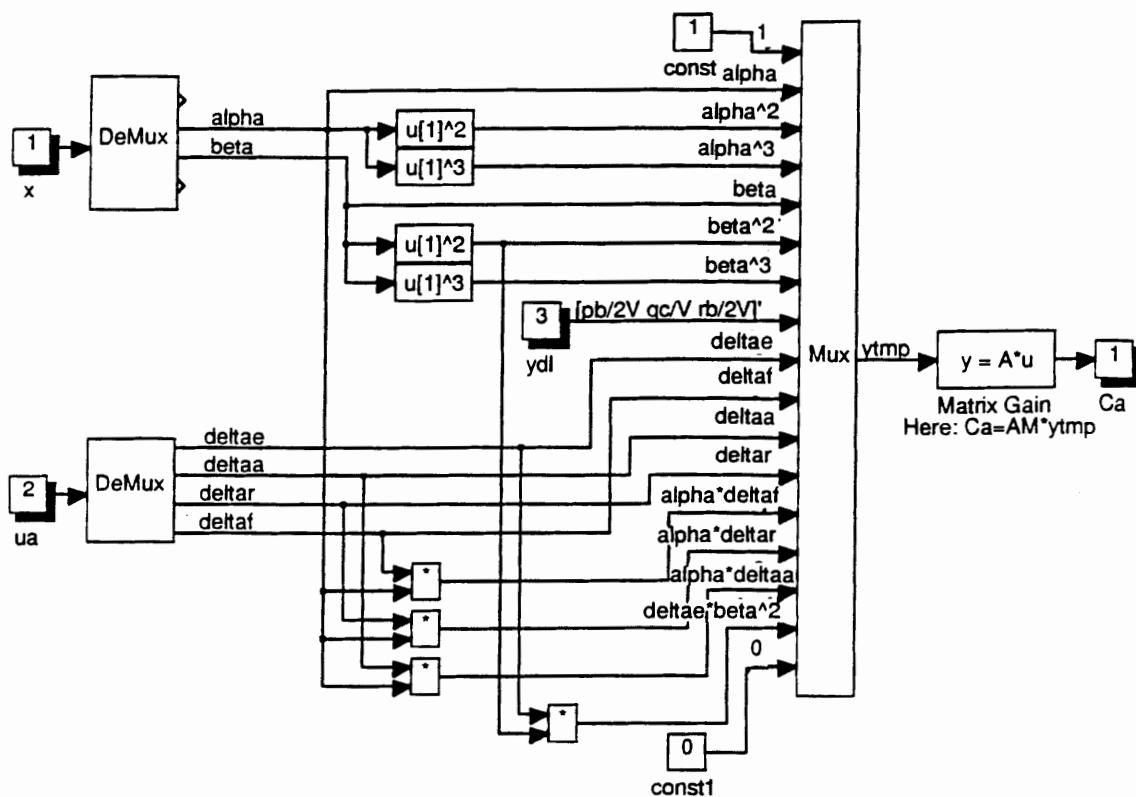


Figure F-10. Internal structure of the 'basic' block Aeromod (Beaver).
[Main](#) | [Beaver dynamics](#) | [Aerodynamics Group](#) | [Aeromod \(Beaver\)](#)

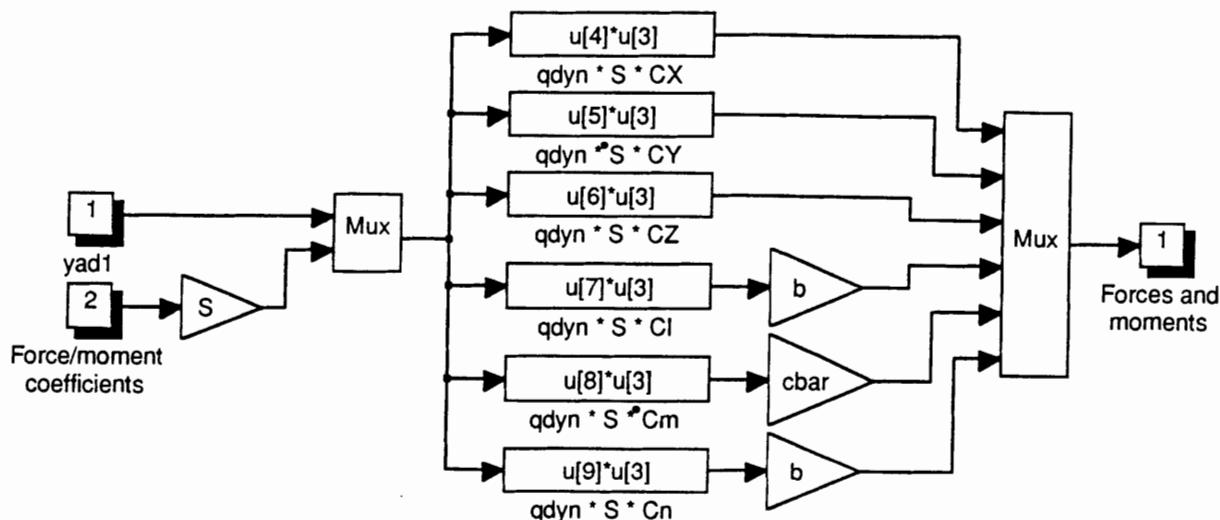


Figure F-11. Internal structure of the 'basic' block *FMdims*.

Main | Beaver dynamics | Aerodynamics Group | *FMdims*, or:
Main | Beaver dynamics | Engine Group | *FMdims*

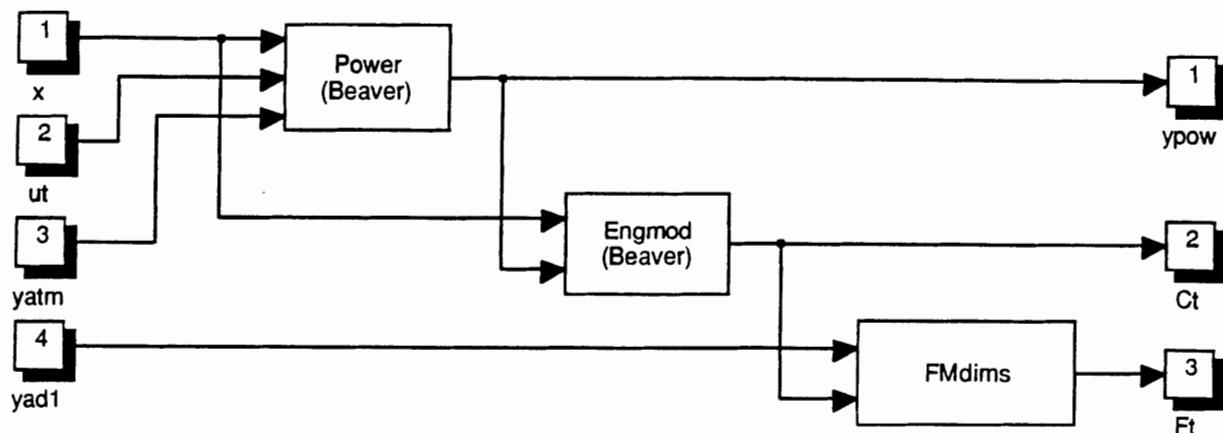


Figure F-12. Internal structure of the subsystem block *Engine Group*.

Main | Beaver dynamics | Engine Group

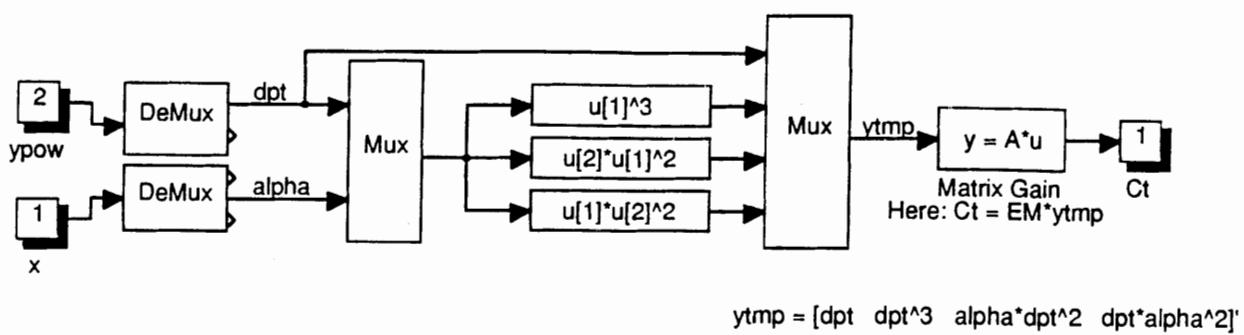


Figure F-14. Internal structure of the 'basic' block *Engmod (Beaver)*.

Main | Beaver dynamics | Engine Group | *Engmod (Beaver)*



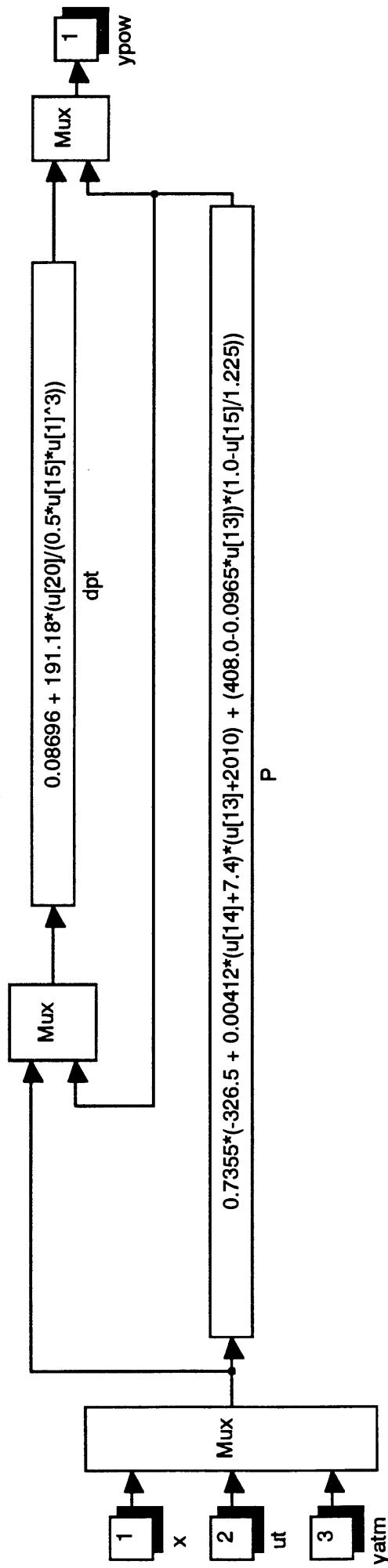


Figure F-13. Internal structure of the 'basic' block **Power (Beaver)**.

Main | Beaver dynamics | Engine Group | Power (Beaver)

Type *help fmdims* for on-line help at the MATLAB command line. The internal structure of *FMDims* is shown in figure F-11.

Inputvectors: \mathbf{y}_{adl} and \mathbf{C}_a
 Outputvector: \mathbf{F}_a

It is possible to use *FMDims* for other types of forces and moments too. For instance, this block is also used within the subsystem *Engine Group*, where instead of \mathbf{C}_a and \mathbf{F}_a , \mathbf{C}_t and \mathbf{F}_t are used.

The variables $cbar$, b , and S are defined in the Mask-definitions line:

```
cbar = GM1(1); b = GM1(2); S = GM1(3);
```

See table F-3 at the end of this appendix for the definition of GM1.

c. Engine Group.

The following 'basic' blocks are available in the subsystem block *Engine Group*, which is depicted in figure F-12:

- 1 - Power (Beaver). This block is used to calculate the engine power P and the dimensionless increase of pressure across the propeller for the DHC-2 'Beaver'. Type *help power* at the MATLAB command line for on-line help. The internal structure of *Power (Beaver)* is shown in figure F-13.

Inputvectors: \mathbf{x} , \mathbf{u}_t , and \mathbf{y}_{atm}
 Outputvector: \mathbf{y}_{pow}

- 2 - Engmod (Beaver). This block calculates the force and moment coefficients due to the operation of the powerplant (including slipstream effects), according to the model from ref.[30], see equations (2-10) in section 2.2.2. Type *help engmod* at the MATLAB command line for on-line help. The structure of *Engmod (Beaver)* is shown in figure F-14.

Inputvectors: \mathbf{x} and \mathbf{y}_{pow}
 Outputvector: \mathbf{C}_t

The block *Matrix Product*, which has been stored in the library *TOOLS*, is used to multiply the help vector \mathbf{y}_{tmp} , which is defined as:

$$\mathbf{y}_{tmp} = [dpt \ dpt^3 \ \alpha \cdot dpt^2 \ \alpha^2 \cdot dpt]^T$$

with the matrix EM. This matrix contains the coefficients of the engine forces and moments model, see the definition in table F-2 at the end of this appendix.

- 3 - FMDims. This block computes the actual forces and moments, using the dimensionless coefficients and the dynamic pressure. It already has been described at the 'Aerodynamics Group'. See figure F-11.



d. Gravity forces.

The forces due to gravity are calculated with the block from figure F-15:

Gravity. Assuming that the mass of the aircraft remains constant under the motions considered, the body axes gravity forces are calculated. If the mass is not constant, it is necessary to implement the mass as an additional input to this block. Type *help gravity* for on-line help at the MATLAB command line.

Inputvectors: \mathbf{x} and \mathbf{y}_{atm}
 Outputvector: \mathbf{F}_{gr}

The variable m has been defined at the Mask-definitions line:

$m = GM1(10);$

See table F-3 for the definition of GM1.

e. Wind forces.

It is possible to use the nonlinear equations of motion in non-steady atmosphere, if additional forces due to the wind and turbulence are included. This is done with the following block (see figure F-16):

Fwind. This block includes contributions to the body axes forces due to non-steady atmosphere. Type *help fwind* at the MATLAB command line for on-line help.

Inputvectors: \mathbf{x} and \mathbf{u}_w
 Outputvector: \mathbf{F}_w

The variable m has been defined at the Mask-definitions line:

$m = GM1(10);$

See table F-3 for the definition of GM1.

f. Total forces and moments.

The following block is used to create two vectors, with the total body axes forces and moments respectively:

EMsort. This block first separates the moments from the forces, and then adds all forces and moments to eachother. This block only functions properly if the aerodynamic forces and moments are expressed in their body axes components. If the aerodynamic lift, drag, and sideforce are used instead, this block needs to be changed. It is also necessary to change this block if more contributions to the total forces and moments are included.

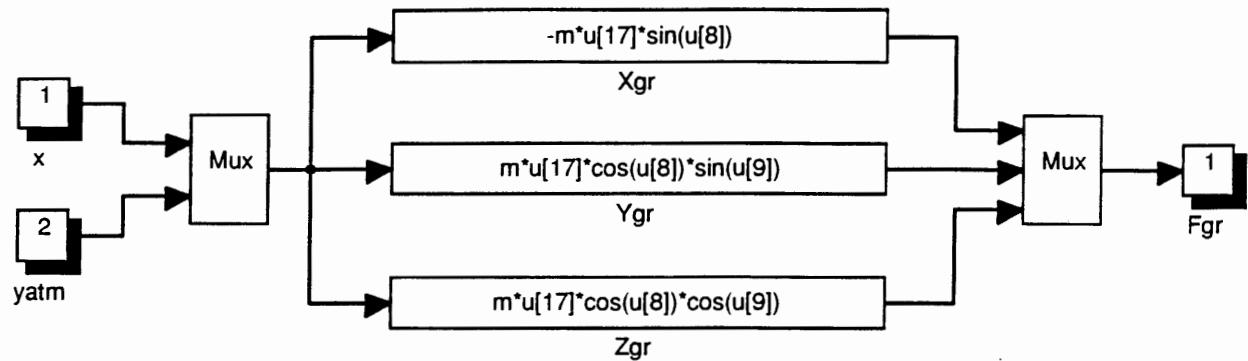


Figure F-15. Internal structure of the 'basic' block Gravity.
[Main](#) | [Beaver dynamics](#) | [Gravity](#)

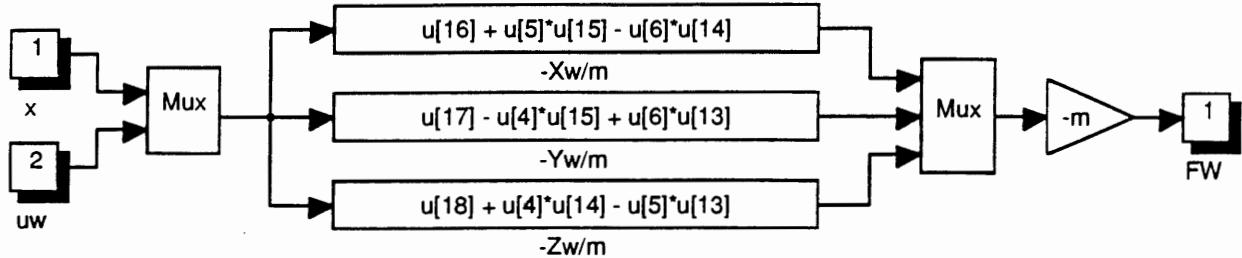


Figure F-16. Internal structure of the 'basic' block Fwind.
[Main](#) | [Beaver dynamics](#) | [Fwind](#)

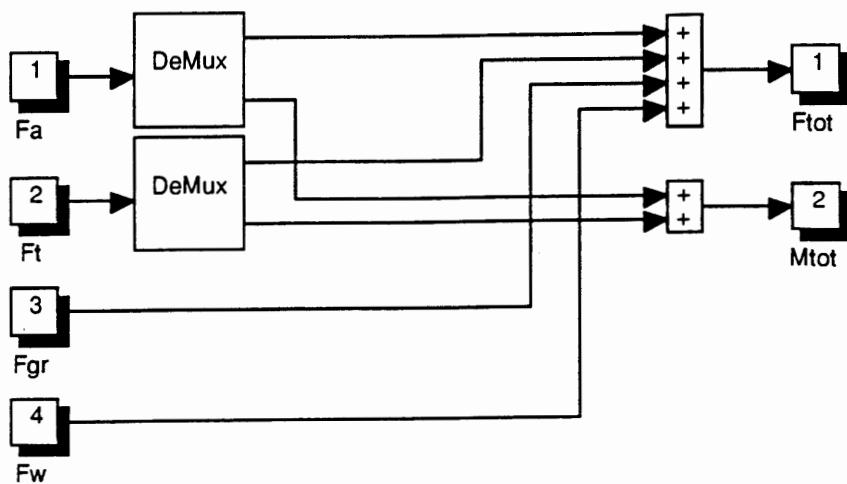


Figure F-17. Internal structure of the 'basic' block FMsort.
[Main](#) | [Beaver dynamics](#) | [FMsort](#)



Type `help fmsort` at the MATLAB command line for on-line help. The internal structure of *FMsort* is shown in figure F-17.

Inputvectors: \mathbf{F}_a , \mathbf{F}_t , \mathbf{F}_{gr} , and \mathbf{F}_w
 Outputvectors: \mathbf{F}_{tot} and \mathbf{M}_{tot}

g. Equations of motion.

The internal structure of the block called *Equations of motion* is shown in figure F-18. This block contains the subsystem *State derivatives*, which is shown in figure F-19.

First the twelve time derivatives of the state variables are computed. In order to determine the position of the aircraft relatively to the earth, the body axes velocities of the aircraft relatively to the surrounding atmosphere are calculated and added to the wind velocity components in the body-axes, u_w , v_w , and w_w . Then corrections to make the implicit equation for β explicit are made, and some state derivatives are artificially set to zero if that is required by the user. Finally, the state derivatives are integrated with respect to time, which yields the state trajectories.

The *integrator* block is a standard SIMULINK block, which uses the variable $xinco$ as initial value of the state vector ($xinco = \mathbf{x}_0$). The *Gain* block multiplies the resulting vector $\dot{\mathbf{x}}$ element-by-element with the vector $xfix$ of length 12, which has elements that equal either one or zero. The gain vector $xfix$ can be changed manually, or by calling the helpfunction *FIXSTATE* (section F.2.6), which makes it possible to fix one or more states, i.e., set the time derivatives of one or more states to zero. This may, for instance, be convenient if the user wants to neglect longitudinal or lateral motions. By default, $xfix$ is set to 1, which is equivalent to $xfix = [1 1 1 1 1 1 1 1 1 1 1 1]^T$.

The equations for the time derivatives of V , α , and β are valid only if the aerodynamic forces and moments are expressed in terms of contributions along the body axes. If the aerodynamic lift, drag, and sideforce are used instead, it is necessary to make some minor changes to these equations, see appendix A.

The subsystem *Equations of motion* consists of the following blocks:

- 1 - *State derivatives*. This is a subsystem block, which contains a number of 'basic' blocks, needed to compute the time derivatives of the state variables, see figure F-19. The following 'basic' blocks are included:

- 1.1 - *Hlpfcn*. This block is used to compute some sines and cosines which are used more than once in the calculation of the different state derivatives. See figure F-20 for the internal structure of this block. Type `help hlpfcn` at the MATLAB command line for on-line help.

Inputvector: \mathbf{x}
 Outputvector: \mathbf{y}_{hlp}

- 1.2 - *Vabdot*. This block calculates the time derivatives of V , α , and β . On-line help is available (type `help vabdot` at the MATLAB com-

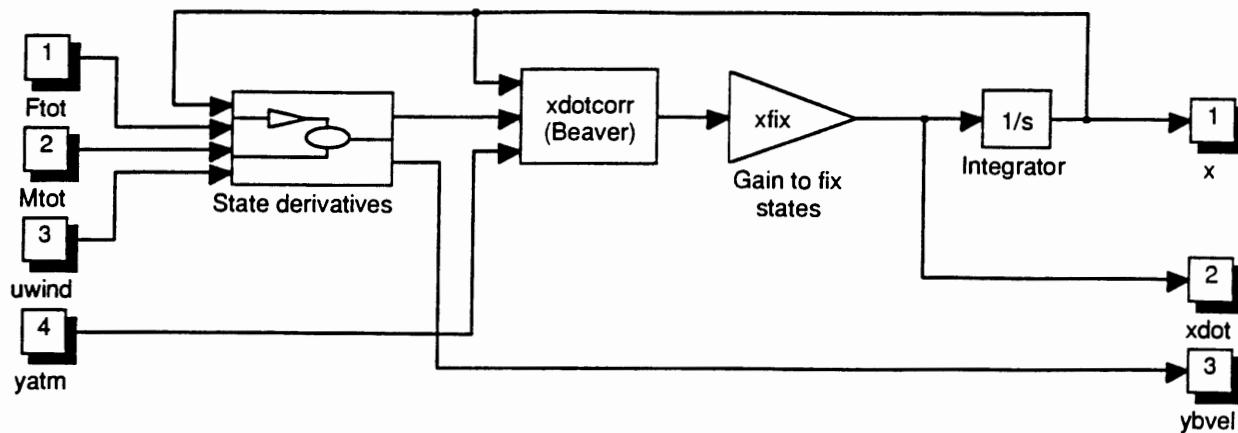


Figure F-18. Internal structure of the subsystem block *Equations of motion*.
Main | Beaver dynamics | Equations of motion

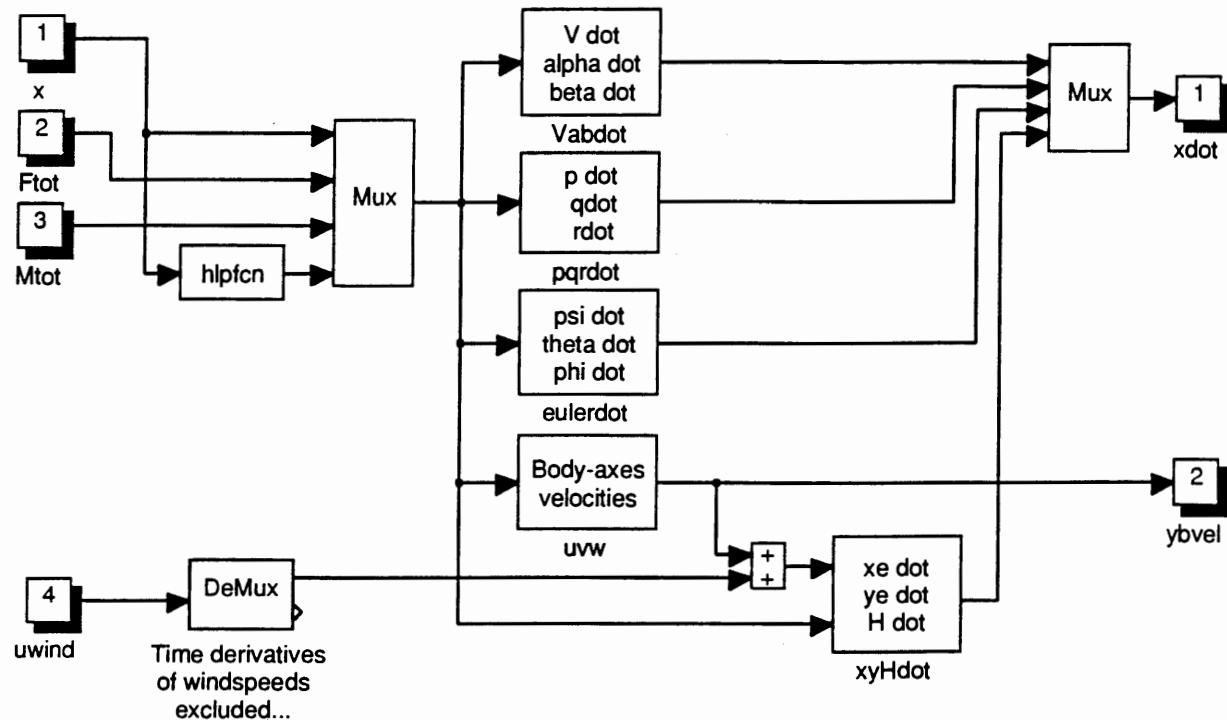


Figure F-19. Internal structure of the subsystem block *State derivatives*.
Main | Beaver dynamics | Equations of motion | State derivatives



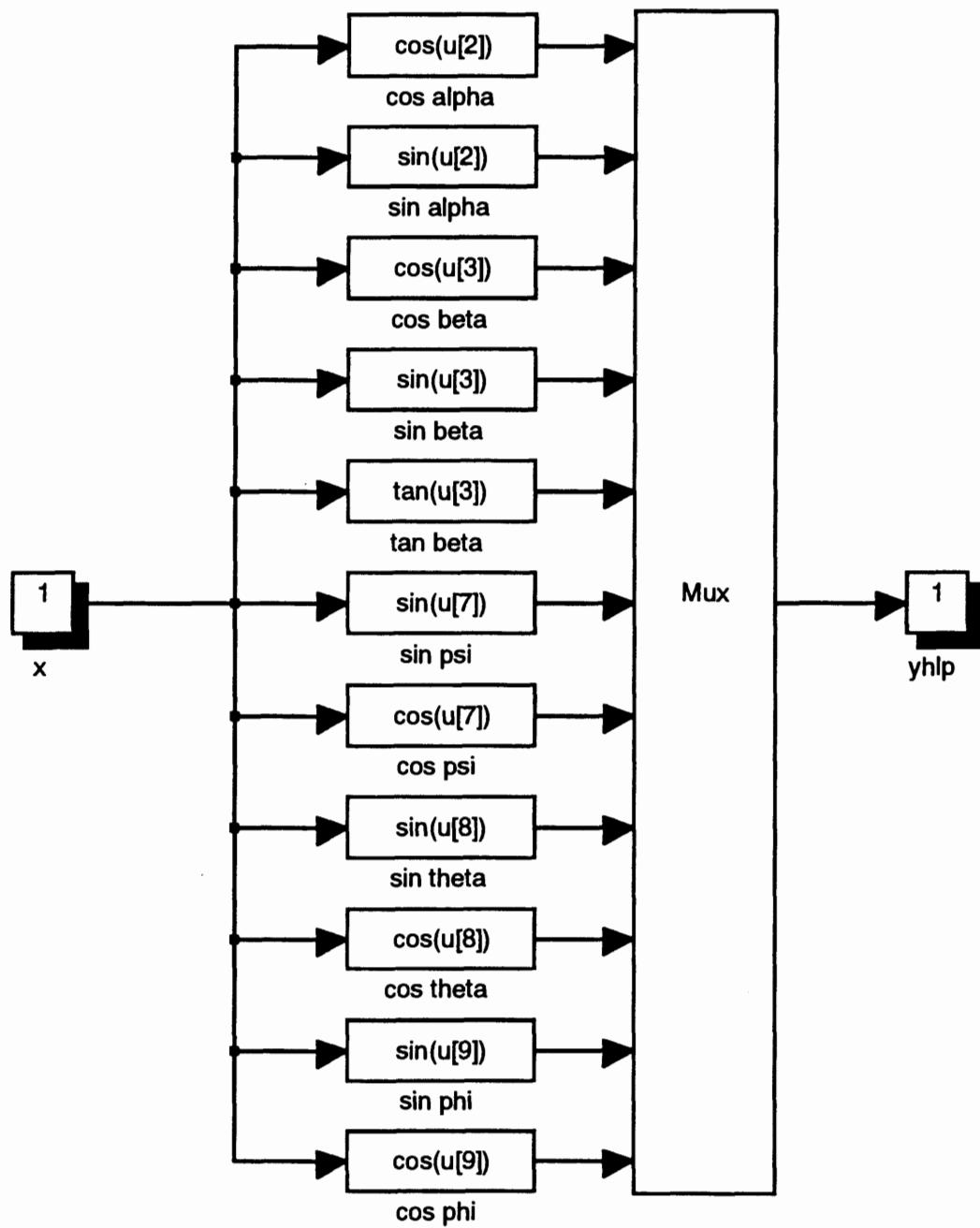


Figure F-20. Internal structure of the 'basic' block *Hlpfcn*.
[Main](#) | [Beaver dynamics](#) | [Equations of motion](#) | [State derivatives](#) | *Hlpfcn*

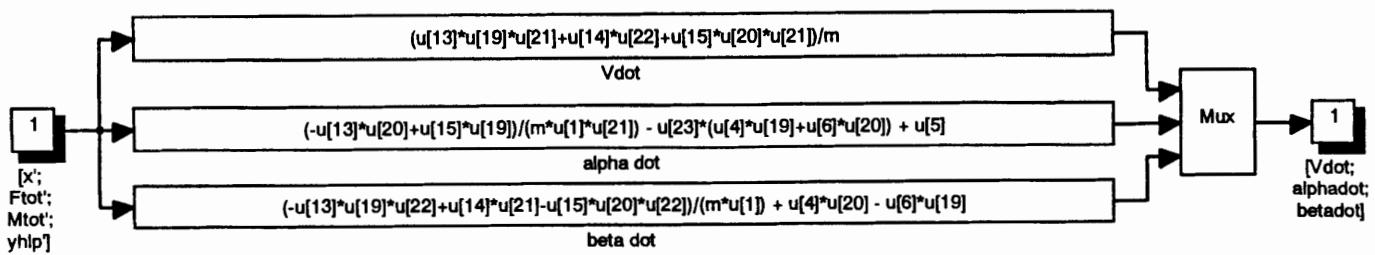
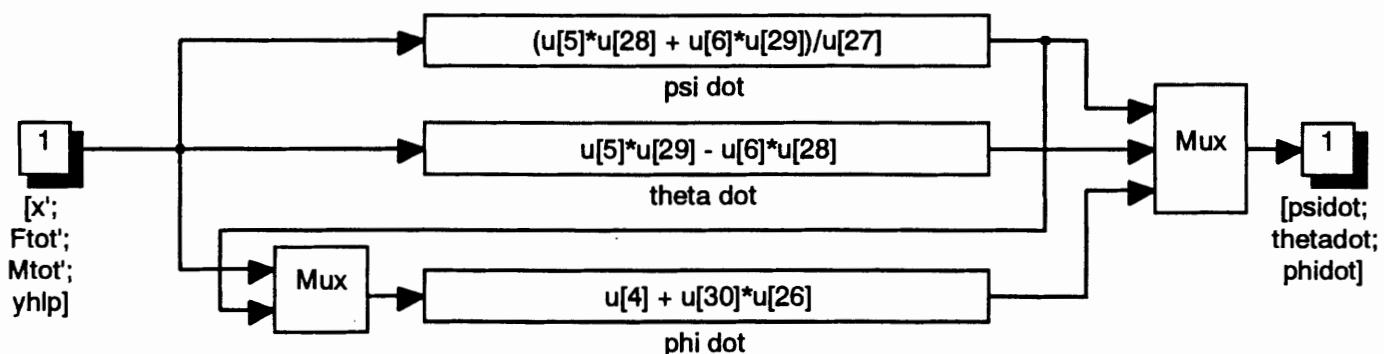
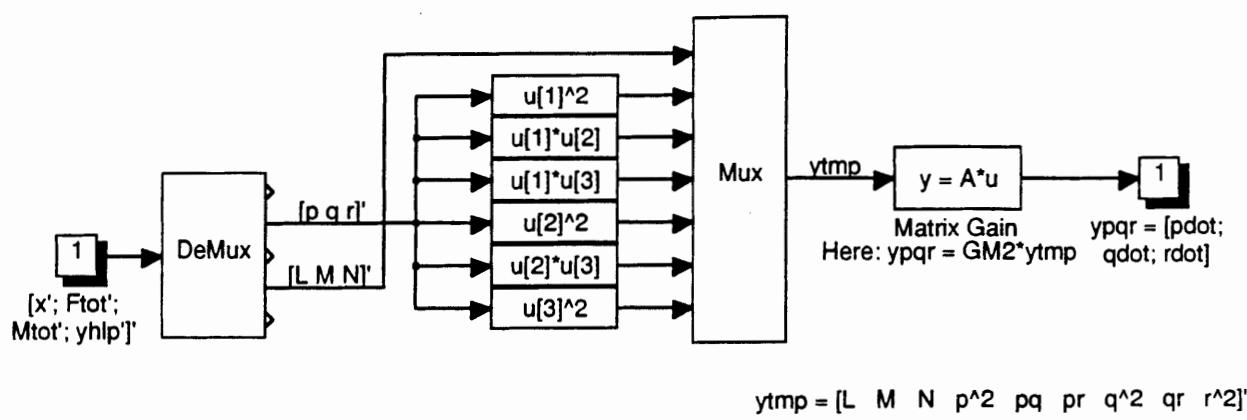


Figure F-21. Internal structure of the 'basic' block *Vabdot*.
Main | Beaver dynamics | Equations of motion | State derivatives | *Vabdot*



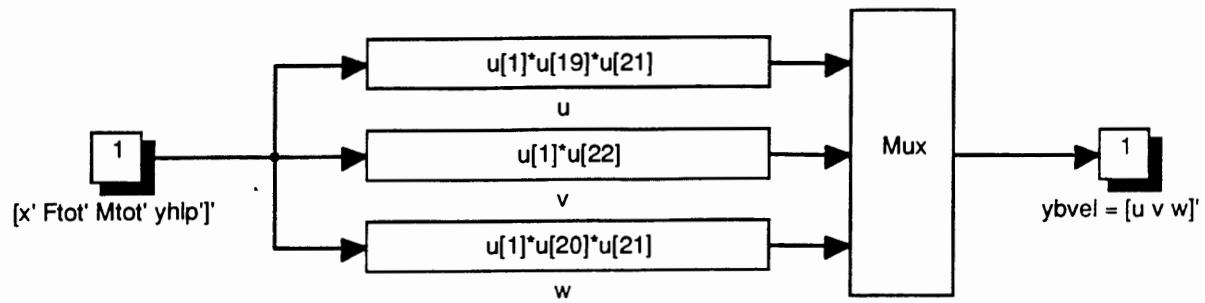


Figure F-24. Internal structure of the 'basic' block uvw .

[Main](#) | [Beaver dynamics](#) | [Equations of motion](#) | [State derivatives](#) | [uvw](#)

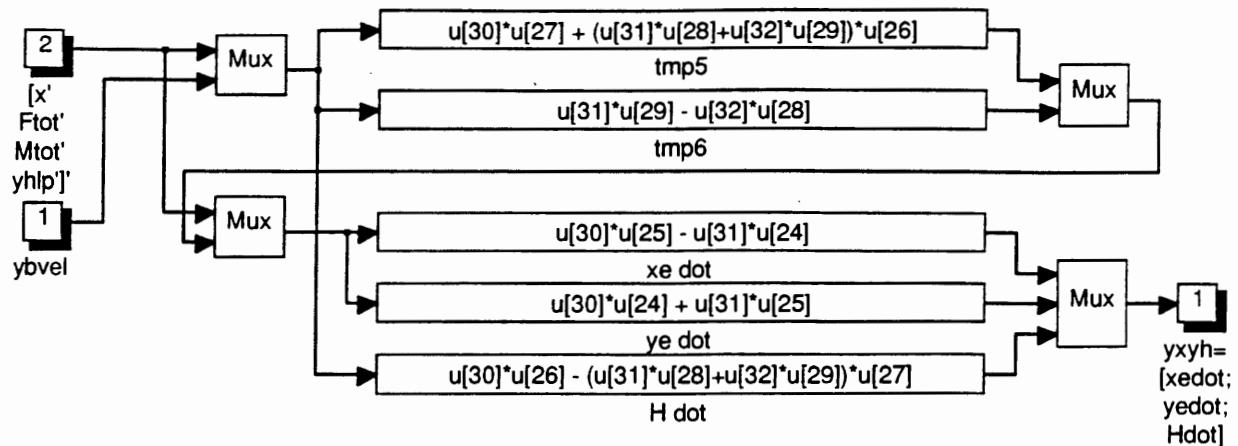


Figure F-25. Internal structure of the 'basic' block $xyHdot$.

[Main](#) | [Beaver dynamics](#) | [Equations of motion](#) | [State derivatives](#) | [xyHdot](#)

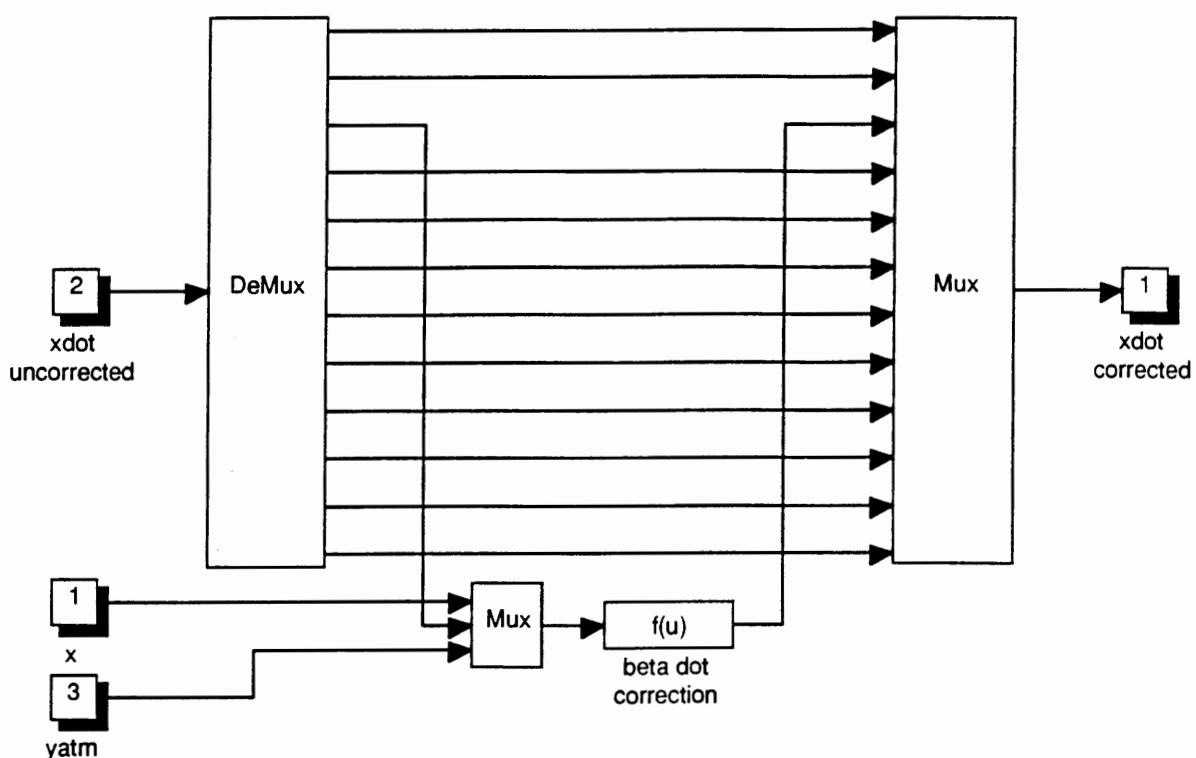


Figure F-26. Internal structure of the 'basic' block $x\dot{c}orr$.

[Main](#) | [Beaver dynamics](#) | [Equations of motion](#) | [x\dot{c}orr](#)

mand line). The structure of *Vabdot* is shown in figure F-21.

$$\begin{array}{ll} \text{Inputvector:} & [\mathbf{x}^T \quad \mathbf{F}_{tot}^T \quad \mathbf{M}_{tot}^T \quad \mathbf{y}_{hlp}^T]^T \\ \text{Outputvector:} & \mathbf{y}_{vab} \end{array}$$

The variable *m* has been defined at the Mask-definitions line:

```
m = GM1(10);
```

See table F-3 for the definition of GM1.

- 1.3 - *pqrdot*. This block calculates the time derivatives of the rotational velocities along the body axes, *p*, *q*, and *r*. Type *help pqrdat* at the MATLAB command line for on-line help. Figure F-22 shows the internal structure of *pqrdat*.

$$\begin{array}{ll} \text{Inputvector:} & [\mathbf{x}^T \quad \mathbf{F}_{tot}^T \quad \mathbf{M}_{tot}^T \quad \mathbf{y}_{hlp}^T]^T \\ \text{Outputvector:} & \mathbf{y}_{pqr} \end{array}$$

A *Matrix Product* block from the library *TOOLS*, which has been included on the floppy disk accompanying this report, is used to multiply the helpvector \mathbf{y}_{tmp} with the matrix GM2. This helpvector is defined as:

$$\mathbf{y}_{hlp} = [L \ M \ N \ p^2 \ p \cdot q \ p \cdot r \ q^2 \ q \cdot r \ r^2]^T$$

See table F-3 at the end of this appendix for the definition of the matrix GM2 with inertia parameters. Notice that the mass distribution of the aircraft is considered to be constant during the motions of interest. If the mass distribution does change during these motions, it is necessary to write out the matrix multiplication, using the mass-distribution parameters as additional variables.

- 1.4 - *Eulerdot*. This block is used to calculate the Euler angles ψ , θ , and ϕ . Type *help eulerdot* at the MATLAB command line for on-line help. The internal structure of *Eulerdot* is depicted in figure F-23.

$$\begin{array}{ll} \text{Inputvector:} & [\mathbf{x}^T \quad \mathbf{F}_{tot}^T \quad \mathbf{M}_{tot}^T \quad \mathbf{y}_{hlp}^T]^T \\ \text{Outputvector:} & \mathbf{y}_{eul} \end{array}$$

- 1.5 - *uvw*. Before the time derivatives of the aircraft's coordinates can be calculated, it is first necessary to compute the body axes velocity components. This is done in the block *uvw*. Type *help uvw* at the MATLAB command line for on-line help. The structure of *uvw* is shown in figure F-24.

$$\begin{array}{ll} \text{Inputvector:} & [\mathbf{x}^T \quad \mathbf{F}_{tot}^T \quad \mathbf{M}_{tot}^T \quad \mathbf{y}_{hlp}^T]^T \\ \text{Outputvector:} & \mathbf{y}_{bvel} \end{array}$$

The wind velocity components have to be added to *u*, *v*, and *w* to obtain correct results.



- 1.6 - *xyHdot*. This block is used to calculate the time derivatives of the coordinates of the aircraft x_e , y_e , and H ($= -z_e$). Type *help xyhdot* at the MATLAB command line for on-line help. Figure F-25 shows the internal structure of *xyHdot*.

Inputvector: $[\mathbf{x}^T \quad \mathbf{F}_{tot}^T \quad \mathbf{M}_{tot}^T \quad \mathbf{y}_{hlp}^T]^T$
 Outputvector: \mathbf{y}_{xyh}

- 2 - *xdotcorr*. This block is used to make the implicit β -equation (section 2.2.3) explicit. This correction is valid for the 'Beaver', but if models of other aircraft which also exhibit implicit state equations are implemented in this structure, it is relatively easy to change *xdotcorr*. Type *help xdotcorr* at the MATLAB command line for on-line help. The internal structure of this block is shown in figure F-26.

Inputvectors: $\mathbf{x}, \dot{\mathbf{x}}$ (uncorrected), and \mathbf{y}_{atm}
 Outputvector: $\dot{\mathbf{x}}$ (corrected to make equation explicit)

The block *beta-dot correction* contains the following expression:

```
u[13] * (1-cos(u[3])) * u[14]*b*S*CYbdot) / (4*m)
```

where the variables b , S , m , and $CYbdot$ have been defined in the Mask-definitions line:

```
b = GM1(2); S = GM1(3); m = GM1(10); CYbdot = AM(2,19);
```

See tables F-1 and F-3 for the definitions of AM and GM1.

h. Other outputs.

Three other output blocks have been added to this list, to demonstrate the flexibility of the model. The blocks calculate flightpath variables and accelerations & specific forces, respectively:

- 1 - *fpath*. This block calculates the flightpath angle γ , flightpath acceleration fpa , azimuth angle χ , and bank angle Φ . Type *help fpath* at the MATLAB command line for on-line help. The internal structure of *fpath* is shown in figure F-27.

Inputvectors: \mathbf{x} and $\dot{\mathbf{x}}$
 Outputvector: \mathbf{y}_{fp}

- 2 - *Accel*. This block can be used to calculate accelerations and specific forces (outputs of accelerometers) in the aircraft's centre of gravity. Type *help accel* at the MATLAB command line for on-line help. The internal structure of *accel* is shown in figure F-28.

Inputvectors: \mathbf{F}_{tot} and \mathbf{F}_{gr}
 Outputvector: \mathbf{y}_{acc}

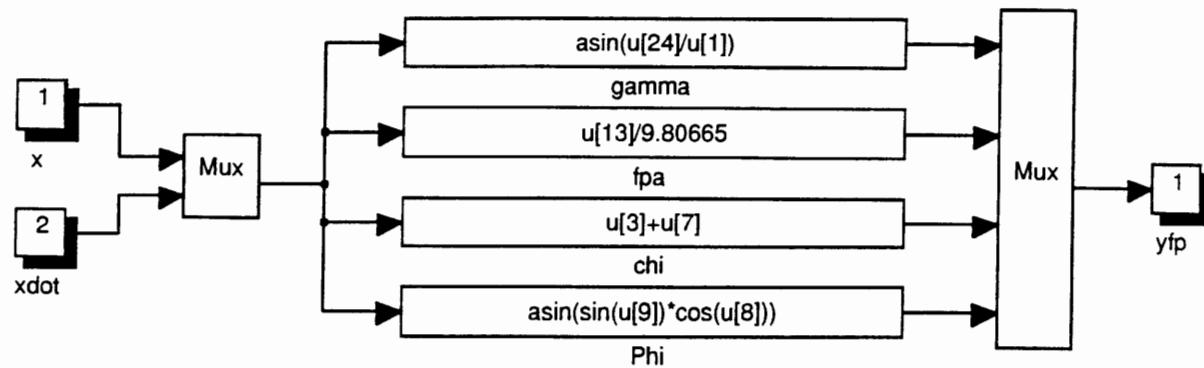


Figure F-27. Internal structure of the 'basic' block *flpath*.

Main | Beaver dynamics | *flpath*

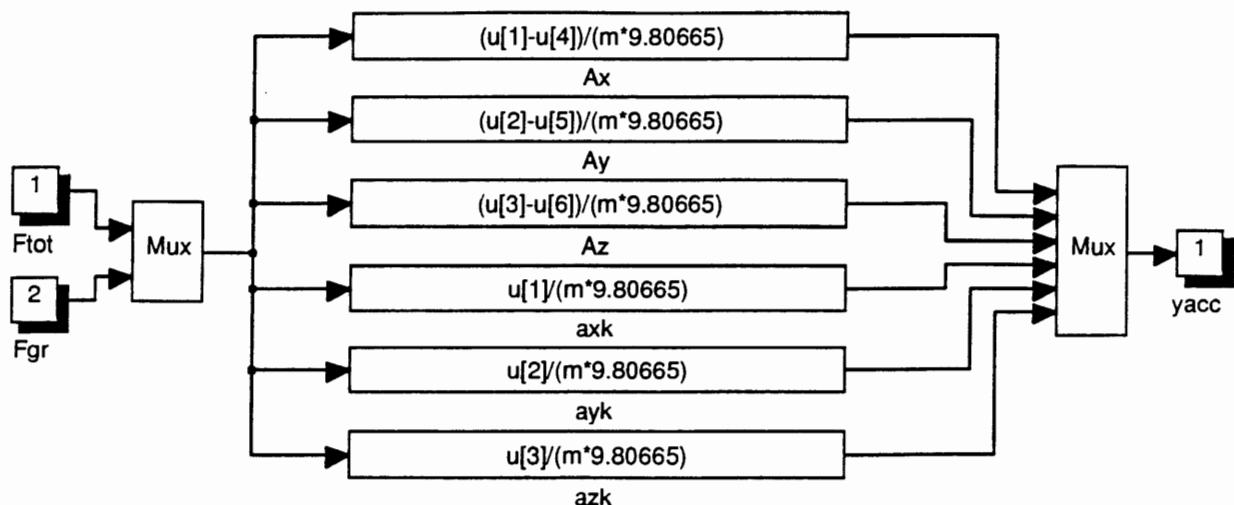


Figure F-28. Internal structure of the 'basic' block *accel*.

Main | Beaver dynamics | *accel*



The variable m has been defined in the Mask-definitions line:

```
m = GM1(10);
```

See table F-3 at the end of this appendix for the definition of GM1.

- 3 - *uvwdot*. This block can be used to calculate the time derivatives of the body-axes velocities. Note that these time-derivatives are not necessary to solve the state equations, because here V , α , and β have replaced u , v , and w as state variables. Still, \dot{u} , \dot{v} , and \dot{w} have been implemented as additional outputs, because these variables may be useful for some analytical purposes. Type *help uvwdot* at the MATLAB command line for on-line help. The internal structure of *uvwdot* is shown in figure F-29.

Inputvectors:	\mathbf{x} and $\dot{\mathbf{x}}$
Outputvector:	\mathbf{y}_{uvw}

F.2.6 Helpprograms to create and load datafiles.

The following three programs can be used to define, store, and load some model parameters which have to be present in the MATLAB workspace before starting a simulation:

- 1 - *MODBUILD*. This M-file creates datafiles with the parameters of the aerodynamic model (the matrix AM, stored in AEROMOD.MAT), the parameters of the engine model (the matrix EM, stored in ENGMOD.MAT), and matrices with data of the aircraft geometry and inertia parameters (GM1 and GM2, stored in AIRCRAFT.MAT). These data are only valid for the 'Beaver'. *MODBUILD* can be called by typing *modbuild* at the MATLAB command line, or pressing the button *Create datafiles*, which has been included in the model library *ACLIB* (sublibrary *Initialization blocks*).

For the 'Beaver', the aerodynamic and engine forces and moments are written as a set of nonlinear polynomials. This is rather unusual, because models of other aircraft often use interpolation in large multi-dimensional tables. But since the aerodynamic model and engine forces & moments model have been implemented as separate blocks, which may have any internal structure as long as the input/output relations are correct, it shouldn't be a problem to implement these submodels for other aircraft. The 'model builder' MODBUILD.M is valid only for the 'Beaver'. The listing of MODBUILD.M and a list of variables is given below:

List of variables for MODBUILD.

AM	matrix with data for Aerodynamic Model (see table F-1)
EM	matrix with data for Engine Model (see table F-2)
GM1	matrix with some data about aircraft geometry and mass-distribution (see table F-3)
GM2	matrix with inertia parameters (see table F-3)

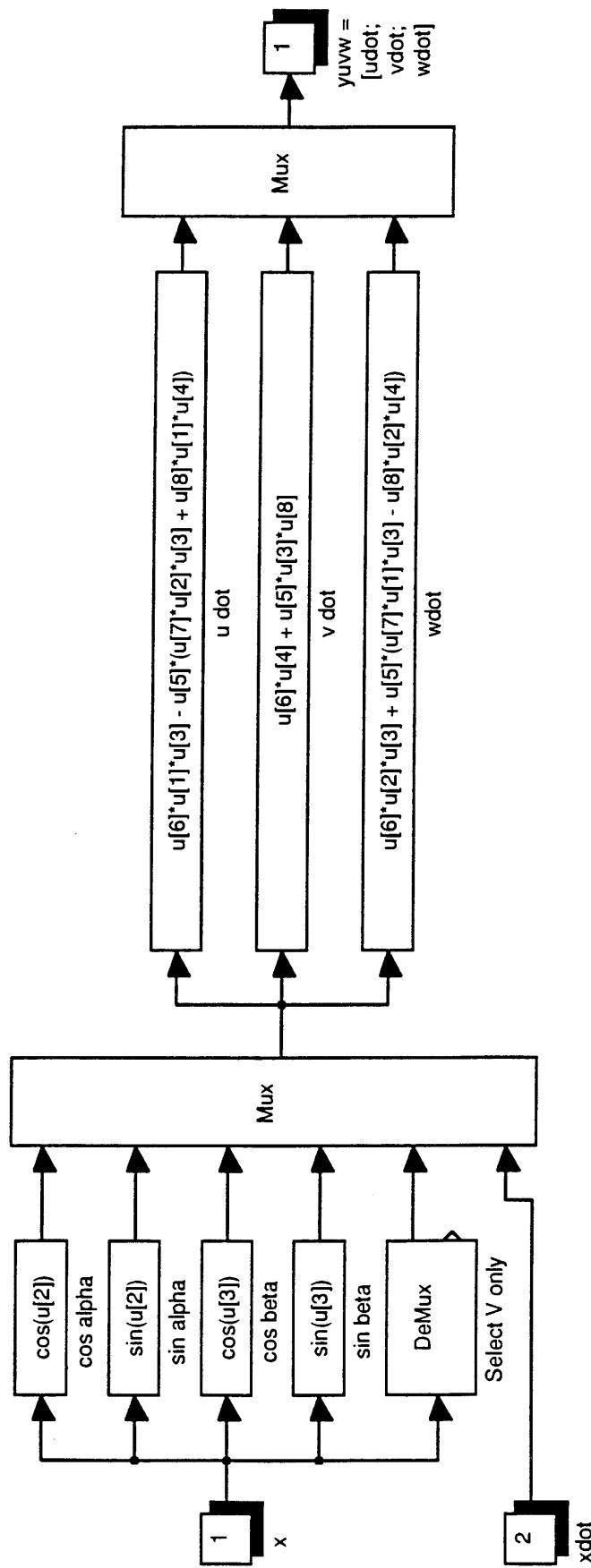


Figure F-29. Internal structure of the 'basic' block $uuvwdot$.
Main | Beaver dynamics | $uuvwdot$



Stability and control derivatives for the nonlinear 'Beaver' model:

CX0	C_{X_0}	CZ0	C_{Z_0}	Cm0	C_{m_0}
CXa	C_{X_a}	CZa	C_{Z_a}	Cma	C_{m_a}
CXa2	$C_{X_{a^2}}$	CZa3	$C_{Z_{a^3}}$	Cma2	$C_{m_{a^2}}$
CXa3	$C_{X_{a^3}}$	CZq	C_{Z_q}	Cmq	C_{m_q}
CXq	C_{X_q}	CZde	$C_{Z_{de}}$	Cmde	$C_{m_{de}}$
CXdr	$C_{X_{dr}}$	CZdeb2	$C_{Z_{deb^2}}$	Cmb2	$C_{m_{b^2}}$
CXdf	$C_{X_{df}}$	CZdf	$C_{Z_{df}}$	Cmr	C_{m_r}
CXadf	$C_{X_{adf}}$	CZadf	$C_{Z_{adf}}$	Cmdf	$C_{m_{df}}$
CXdpt	$C_{X_{dpt}}$	CZdpt	$C_{Z_{dpt}}$	Cmdpt	$C_{m_{dpt}}$
CXadpt2	$C_{X_{adpt^2}}$				

CY0	C_{Y_0}	Cl0	C_{l_0}	Cn0	C_{n_0}
CYb	C_{Y_b}	Clb	C_{l_b}	Cnb	C_{n_b}
CYp	C_{Y_p}	Clp	C_{l_p}	Cnp	C_{n_p}
CYr	C_{Y_r}	Clr	C_{l_r}	Cnr	C_{n_r}
CYda	$C_{Y_{da}}$	Clda	$C_{l_{da}}$	Cnda	$C_{n_{da}}$
CYdr	$C_{Y_{dr}}$	Cldr	$C_{l_{dr}}$	Cndr	$C_{n_{dr}}$
CYdra	$C_{Y_{dra}}$	Cldaa	$C_{l_{daa}}$	Cnq	C_{n_q}
CYbdot	$C_{Y_{bdot}}$	Cla2dpt	$C_{l_{a^2,dpt}}$	Cnb3	$C_{n_{b^3}}$
				Cndpt3	$C_{n_{dpt^3}}$

Geometric properties:

b	$b =$ wing span
cbar	$\bar{c} =$ mean aerodynamic chord
S	$S =$ wing area
m	$m =$ mass of the aircraft

Products and moments of inertia:

Ix	I_x (see table A-1, appendix A)
Iy	I_y (table A-1)
Iz	I_z (table A-1)
Jxy	J_{xy} (table A-1)
Jxz	J_{xz} (table A-1)
Jyz	J_{yz} (table A-1)

Inertia parameters:

detI	$ I $ (see table A-2)
I1	I_1 (table A-2)
I2	I_2 (table A-2)
I3	I_3 (table A-2)
I4	I_4 (table A-2)
I5	I_5 (table A-2)
I6	I_6 (table A-2)

P _l	P_l	Q _l	Q_l	R _l	R_l
P _m	P_m	Q _m	Q_m	R _m	R_m
P _n	P_n	Q _n	Q_n	R _n	R_n
P _{pp}	P_{pp}	Q _{pp}	Q_{pp}	R _{pp}	R_{pp}
P _{pq}	P_{pq}	Q _{pq}	Q_{pq}	R _{pq}	R_{pq}
P _{pr}	P_{pr}	Q _{pr}	Q_{pr}	R _{pr}	R_{pr}
P _{qq}	P_{qq}	Q _{qq}	Q_{qq}	R _{qq}	R_{qq}
P _{qr}	P_{qr}	Q _{qr}	Q_{qr}	R _{qr}	R_{qr}
P _{rr}	P_{rr}	Q _{rr}	Q_{rr}	R _{rr}	R_{rr}

Listing of MODBUILD.M.

```
% -----
% MODBUILD.m builds datafiles for the aerodynamic model, engine model,
% and aircraft geometric properties and mass distribution, for use
% within the Simulink S-function BEAVER.m. MODBUILD creates the data-
% files aeromod.mat, engmod.mat, airdata.mat, and aircraft.mat.
%
clear
clc
disp('Building matrices with data for aerodynamic model, engine model,')
disp('and mass distribution/geometry model for the DHC-2 "Beaver"')
disp('aircraft.')
disp('')
disp('Wait a moment...')

% The nonlinear aerodynamic model of the "BEAVER" aircraft valid
% within 35-55 m/s TAS range (see Tjee & Mulder).
%
CX0 = -0.03554; CZ0 = -0.05504; Cm0 = 0.09448;
CXA = 0.002920; CZa = -5.578; Cma = -0.6028;
CXA2 = 5.459; CZa3 = 3.442; Cma2 = -2.140;
CXA3 = -5.162; CZq = -2.988; Cmq = -15.56;
CXq = -0.6748; CZde = -0.3980; Cmde = -1.921;
CXdr = 0.03412; CZdeb2 = -15.93; Cmb2 = 0.6921;
CXdf = -0.09447; CZdf = -1.377; Cmr = -0.3118;
CXadf = 1.106; CZadf = -1.261; Cmdf = 0.4072;

CY0 = -0.002226; Cl0 = 0.0005910; Cn0 = -0.003117;
CYb = -0.7678; Clb = -0.06180; Cnb = 0.006719;
Cyp = -0.1240; Clp = -0.5045; Cnp = -0.1585;
CYr = 0.3666; Clr = 0.1695; Cnr = -0.1112;
CYda = -0.02956; Clda = -0.09917; Cnda = -0.003872;
CYdr = 0.1158; Cldr = 0.006934; Cndr = -0.08265;
CYdra = 0.5238; Cldaa = -0.08269; Cnq = 0.1595;
CYbdot = -0.1600; Cnb3 = 0.1373;

AM = [ CX0 CY0 CZ0 Cl0 Cm0 Cn0 ;
        CXa 0 CZa 0 Cma 0 ;
        CXa2 0 0 Cma2 0 ;
        CXa3 0 CZa3 0 0 0 ;
        0 CYb 0 Clb 0 Cnb 0 ;
        0 0 0 0 Cmb2 0 ];
```



```

0      0      0      0      0      Cnb3 ;
0      Cyp    0      Clp    0      Cnp  ;
CXq   0      CZq    0      Cmq    Cnq  ;
0      CYr    0      Clr    Cmr    Cnr  ;
0      0      CZde   0      Cmdie  0   ;
CXdf  0      CZdf   0      Cmdf   0   ;
0      CYda   0      Clda   0      Cnda ;
CXdr  CYdr   0      Cldr   0      Cndr ;
CXadf 0      CZadfc 0      0      0   ;
0      CYdra   0      0      0      0   ;
0      0      Cldaa  0      0      0   ;
0      0      CZdeb2 0      0      0   ;
0      CYbdot  0      0      0      0   ];
;

AM = AM';
save aeromod.mat AM          % data for Aerodynamic Model
clear
% The nonlinear engine model of the DHC-2 "BEAVER" aircraft
% valid within the 35-55 m/sec TAS-range (see Tjee & Mulder).
% -----
CXdpt = 0.1161;
CXadpt2 = 0.1453;
CZdpt = -0.1563;
Clad2dpt = -0.01406;
Cmdpt = -0.07895;
Cndpt3 = -0.003026;

EM = [ CXdpt 0      CZdpt 0      Cmdpt 0      Cndpt3 ;
        0      0      0      0      0      0   ;
        CXadpt2 0      0      0      0      0   ;
        0      0      0      Clad2dpt 0      0   ];
;

EM = EM';
save engmod.mat EM          % data for Engine Model
clear

% Aircraft data on which the aerodynamic model is based
% (see Tjee & Mulder).
% -----
Ix     = 5368.39;           % kgm^2 in Fr
Iy     = 6928.93;           % kgm^2 in Fr
Iz     = 11158.75;          % kgm^2 in Fr
Jxy   = 0.0;                % kgm^2 in Fr
Jxz   = 117.64;             % kgm^2 in Fr
Jyz   = 0.0;                % kgm^2 in Fr
m     = 2288.231;           % kg

% geometric data.
% -----
cbar   = 1.5875;            % m
b      = 14.63;              % m
S      = 23.23;               % m^2

% Calculate inertia parameters (see NASA TP 2768). The formula's
% are valid for symmetric and asymmetric aircraft.
% -----
detI = Ix*Iy*Iz - 2*Jxy*Jxz*Jyz - Ix*Jyz^2 - Iy*Jxz^2 - Iz*Jxy^2;
I1   = Iy*Iz - Jyz^2;
I2   = Jxy*Iz + Jyz*Jxz;
I3   = Jxy*Jyz + Iy*Jxz;
I4   = Ix*Iz - Jxz^2;
I5   = Ix*Jyz + Jxy*Jxz;
I6   = Ix*Iy - Jxy^2;

P1   = I1/detI; Pm = I2/detI; Pn = I3/detI;
Ppp = -(Jxz*I2 - Jxy*I3)/detI;
Ppq = (Jxz*I1 - Jyz*I2 - (Iy-Ix)*I3)/detI;
Ppr = -(Jxy*I1 + (Ix-Iz)*I2 - Jyz*I3)/detI;
Pqq = (Jyz*I1 - Jxy*I3)/detI;
Pqr = -((Ix-Iy)*I1 - Jxy*I2 + Jxz*I3)/detI;
Prr = -(Jyz*I1 - Jxz*I2)/detI;

Q1   = I2/detI; Qm = I4/detI; Qn = I5/detI;
Qpp = -(Jxz*I4 - Jxy*I5)/detI;

```

```

Qpq = (Jxz*I2 - Jyz*I4 - (Iy-Ix)*I5)/detI;
Qpr = -(Jxy*I2 + (Ix-Iz)*I4 - Jyz*I5)/detI;
Qqq = (Jyz*I2 - Jxy*I5)/detI;
Qqr = -((Ix-Iy)*I2 - Jxy*I4 + Jxz*I5)/detI;
Qrr = -(Jyz*I2 - Jxz*I4)/detI;

R1 = I3/detI; Rm = I5/detI; Rn = I6/detI;
Rpp = -(Jxz*I5 - Jxy*I6)/detI;
Rpq = (Jxz*I3 - Jyz*I5 - (Iy-Ix)*I6)/detI;
Rpr = -(Jxy*I3 + (Ix-Iz)*I5 - Jyz*I6)/detI;
Rqq = (Jyz*I3 - Jxy*I6)/detI;
Rqr = -((Ix-Iy)*I3 - Jxy*I5 + Jxz*I6)/detI;
Rrr = -(Jyz*I3 - Jxz*I5)/detI;

% Summarizing results in aircraft parameter matrices GM1 and GM2.
% -----
GM1 = [cbar b S Ix Iy Iz Jxy Jxz Jyz m];

GM2 = [ P1 Pm Pn Ppp Ppq Ppr Pqq Pqr Prr ;
        Q1 Qm Qn Qpp Qpq Qpr Qqq Qqr Qrr ;
        R1 Rm Rn Rpp Rpq Rpr Rqq Rqr Rrr ];

save aircraft.mat GM1 GM2
clear

disp('Ready. The data can now be loaded into the Matlab workspace.')
disp('')
disp('aerodynamic data: LOAD aeromod (contains the matrix AM),')
disp('engine data : LOAD engmod (contains the matrix EM),')
disp('aircraft data : LOAD aircraft (contains GM1 and GM2).')
disp('')
disp('The datamatrices AM, EM, GM1, and GM2 must be defined GLOBAL!')
disp('')
disp('The macro LOADER.m can also be used to load the data matrices')
disp('and define them as global variables!')
% -----
% Program written by: Marc Rauw. Version: july 2, 1993.

```

- 2 - ***LOADER***. This program must be runned before starting simulations of systems which use *BEAVER* as subsystem. *LOADER* simply loads the data-files, created by the function *MODBUILD*, and it defines the matrices AM, EM, GM1, and GM2 as global variables in the MATLAB workspace (see tables F-1 to F-3 for the definitions of these matrices). If the variable *xfix* is not defined within the MATLAB workspace, it is set to the default value (*xfix* = 1). This variable can be altered afterwards by calling *FIXSTATE*.

LOADER can be activated by typing *loader* at the MATLAB command line, or by double-clicking the *Load model parameters* button with the mouse. This button was created with the SIMULINK *Mask* function and has been included in the library *ACLIB* (sublibrary: *Initialization blocks*). The listing of *LOADER.M* is given below:

```

% LOADER.m loads the datamatrices for the aircraft models in the Matlab
% Workspace and defines them as global variables. AM contains stability
% and control derivatives for the aerodynamic model. EM contains para-
% meters of the engine model. GM1 contains data about aircraft geometry
% and mass distribution, and GM2 contains other inertia parameters.
% -----
e1 = exist('aircraft.mat');
e2 = exist('aeromod.mat');
e3 = exist('engmod.mat');
if e1~=2 | e2~=2 | e3~=2
    error('Datafile(s) not found. Run MODBUILD first!');
end
clear e1 e2 e3

load aircraft
global GM1 GM2

```



```

load aeromod
global AM
load engmod
global EM

if exist('xfix')~=1
    xfix = 1;
end

who
%
% -----%
% Programmed by Marc Rauw. Version: july 2, 1993.

```

The variables $e1$, $e2$, and $e3$ are helpvariables, used to check if the datafiles AIRCRAFT.MAT, AEROMOD.MAT, and ENGMOD.MAT exist in the workdirectory. If not, *MODBUILD* should be runned.

- 3 - ***FIXSTATE***. For some purposes, it may be necessary to fix some of the state variables, for instance, to eliminate longitudinal or lateral dynamics. The help program *FIXSTATE.M* enables the user to specify which states should be fixed and which states may vary.

If the program is started, a menu will be displayed, in which the user can select to fix symmetrical states, asymmetrical states, arbitrary states, or to reset *xfix* to its standard value (*xfix* = 1). If option 3 is chosen, the user must enter a vector with the numbers of the states which should be fixed ($\in \{1, 2, \dots, 12\}$). For instance, if the user specifies the vector: [1 3 10], states 1, 3, and 5 will be fixed ($\mathbf{x}[1] = V$, $\mathbf{x}[3] = \beta$, $\mathbf{x}[10] = x_e$). *FIXSTATE* automatically computes the corresponding value of the vector *xfix*, which in this case should equal:

```
xfix = [ 0  1  0  1  1  1  1  1  1  0  1  1 ]
```

The listing of *FIXSTATE* is given below:

```

%
% FIXSTATE.m is a helpfile, which can be used to fix state variables from the
% nonlinear aircraft model. FIXSTATE sets the value of the gain vector xfix. For
% instance, if the true airspeed (the first element of the state vector)
% is kept constant, xfix will equals [0 1 1 1 1 1 1 1 1 1 1 1], and xdot is set
% to xdot .* xfix.
%
% Without calling FIXSTATE, it is necessary to set xfix = 1, which is done auto-
% matically by the helpfile LOADER.
%
opt=menu('Options:','fix asymmetrical states','fix symmetrical states',...
         'fix arbitrary states','don''t fix any states');

if opt == 1
    xfix = [1 1 0 0 1 0 0 1 0 1 0 1];
    ok = 0;
    while ok ~= 1
        answ = input('Fix ye (y/n)? ','s');
        if answ == 'y'
            xfix(11) = 0;
            ok = 1;
        elseif answ == 'n'
            xfix(11) = 1;
            ok = 1;
        else
            disp('Enter y or n');
        end
    end
end

```

```

clear ok answ
elseif opt == 2
    disp('');
    xfix = [0 0 1 1 0 1 1 0 1 0 1 0];
    ok = 0;
    while ok~= 1
        answ = input('Fix xe (y/n)? ','s');
        if answ == 'y'
            xfix(10) = 0;
            ok = 1;
        elseif answ == 'n'
            xfix(10) = 1;
            ok = 1;
        else
            disp('Enter y or n');
        end
    end
    ok = 0;
    while ok~= 1
        answ = input('Fix H (y/n)? ','s');
        if answ == 'y'
            xfix(12) = 0;
            ok = 1;
        elseif answ == 'n'
            xfix(12) = 1;
            ok = 1;
        else
            disp('Enter y or n');
        end
    end
    clear ok answ
elseif opt == 3
    disp('The state vector equals:');
    disp('[V alpha beta p q r psi theta phi xe ye H]');
    disp('');
    disp('Specify a vector containing the element numbers of the');
    disp('states that should be fixed ( [] = don''t fix any states ): ');
    fix = input('> ');
    if min(fix) < 1 | max(fix)>12
        error('Element numbers should belong to {1,2,3,4,5,6,7,8,9,10,11,12}!');
    end
    xfix = [1 1 1 1 1 1 1 1 1 1 1 1];
    for i = 1:length(fix)
        xfix(fix(i)) = 0;
    end
    clear fix i
else
    xfix = 1;
end

% re-initialize system
% -----
[sys,x0] = beaver([],[],[],0); clear sys x0
xfix

%
% Programmed by Marc Rauw. Version: july 2, 1993.

```

The variable *fix* contains the vector with element numbers that need to be fixed; *i* is a counter; *opt* contains the menu item that is selected by the user; *answ* is used to store answers to 'yes/no' questions, and *ok* is a variable which is used to determine if while loops should be stopped or held, depending upon the user-inputs to certain questions.

FIXSTATE can be started by typing *fixstate* at the MATLAB command line, or by double-clicking the block *Fix States*, which is contained in the sublibrary *Initialization blocks* of *ACLIB*. Of course, it is also possible to define *xfix* manually.



F.2.7 Helpfunctions to compute initial conditions and to linearize the aircraft model.

The initial (flight-) conditions for the simulations are usually selected to be steady-state conditions in which all translational and rotational accelerations are zero. As stated in appendix E, a numerical optimization process is used to find such steady equilibrium conditions. In this section the implementation of the aircraft trim algorithm from appendix E will be described. Then a loader which can be used to retrieve trimmed flight conditions from disk is treated, and finally, a help program to linearize the aircraft model will be given.

- 1 - ***ACTRIM***. This M-file contains the numerical trim algorithm, described in appendix E. *ACTRIM* can be applied to any aircraft model which uses the same definitions of input and state variables as the system *BEAVER* (the variable *modfun* must contain the name of the aircraft model to be trimmed). *ACTRIM* calls the M-files ACCONSTR.M and ACCOST.M to compute flightpath constraints and the cost function for the numerical minimization routine. The resulting trimmed-flight values of the state vector **x** and the input vector **u** can be stored in a MAT-file and retrieved with the load routine *INCOLOAD*. The trim programs were based upon the theory from ref.[29] (see appendix E) and similar routines for the Citation 500, written by R. Bogers.

List of variables, used in ACTRIM.M, ACCONSTR.M, AND ACCOST.M:

answ	string variable with answer to '(y)es/(n)o - questions'
ctrim	vector, which contains all constants that specify the flight-condition and trim options ($ctrim = [V \ H \ \psi \ \gamma \ G \ \dot{\psi} \ \dot{\theta} \ \dot{\phi} \ \delta_f \ n \ \varphi]^T$)
deltaf	δ_f = flap angle; [rad]
G	$G = \frac{\dot{\psi}V}{g_0}$
gamma	desired flight path angle γ ; [rad]
H	(initial) altitude H ; [m]
Hdot	rate-of-climb $\dot{H} = V \sin \gamma$; [m/s]
modfun	string variable, containing the function call to the aircraft model, used to determine the time-derivative of the state vector (<i>modfun</i> = ' <i>xdot = beaver(0,x,u,1)</i> ', see S-function call in section 3.3)
n	engine speed n ; [RPM]
name	string variable, containing path and filename of file in which the initial condition must be saved
opt	menu item, selected by the user in the main menu of <i>ACTRIM</i> ($\in \{ 1, 2, 3, 4, 5 \}$)
opts	vector with options for minimization routine <i>FMINS</i> (type <i>help fmins</i> at the MATLAB command-line for more info)
options	vector with minimization options, returned by <i>FMINS</i> after minimization
phi	roll angle φ ; [rad]
phidot	desired roll-rate $\dot{\varphi}$; [rad/s]
psi	(initial) yaw angle ψ ; [rad]
psidot	desired yaw-rate $\dot{\psi}$
pz	manifold pressure p_z ; ["Hg]

rolltype	string variable, used to specify if a body-axis roll or a stability-axis roll is wanted (<i>rolltype</i> = ' <i>b</i> ', or <i>rolltype</i> = ' <i>s</i> ', respectively)
savecmmnd	string variable, containing the command line which is necessary to save the trim values of x and u
skip	variable, which is set to 1 if user selects menu item 5 ('quit actrim'), and otherwise equals 0
thetadot	desired pitch-rate $\dot{\theta}$
turntype	string variable, used to specify if a coordinated or an uncoordinated turn is wanted (<i>turntype</i> = ' <i>c</i> ', or <i>turntype</i> = ' <i>u</i> ', respectively)
u	u = input vector to the system <i>BEAVER</i> , see table F-4 or F-5
ua0	resulting trim value of inputvector to aerodynamic model ($\mathbf{u}_a = [\delta_e \ \delta_a \ \delta_r \ \delta_f]^T$)
ut0	resulting trim value of inputvector to engine model ($\mathbf{u}_t = [n \ p_z]^T$)
V	desired airspeed <i>V</i> ; [m/s]
vtrim	vector with all variables which are adjusted by the trim algorithm to minimize the cost function <i>J</i> (<i>vtrim</i> = [$\alpha \ \beta \ \delta_e \ \delta_a \ \delta_r \ p_z$] ^T)
vtrimmed	contains the value of <i>vtrim</i> which minimizes the cost function <i>J</i> , i.e., <i>vtrimmed</i> is the value of <i>vtrim</i> after running the minimization algorithm
x	x = state vector of the system <i>BEAVER</i> , see table F-4 or F-5
xdot	$\dot{\mathbf{x}}$ = time-derivative of state vector of the system <i>BEAVER</i> , see table F-4 or F-5
xdot0	time-derivative of state vector for the x , u combination, which was found by the trim algorithm
xinco	resulting trim value of state vector x

Listing of ACTRIM.M:

```
%=====
% ACTRIM.m is an aircraft trim routine, which was designed for the BEAVER
% model structure. The routine is based upon a trim program for the Citation
% 500 aircraft, written by R. Bogers.
%=====
format short e;

% Load model parameters and set xinco. Note: the value of xinco doesn't
% really matter, because it won't be used during the trim process. However,
% since the variable xinco is required by the integrators of the system
% BEAVER, Simulink can't define the system internally without having xinco
% available in the Matlab workspace. To avoid Division-By-Zero Errors, the
% first element of xinco (=V) has been equalled to 45 [m/s].
%
if (exist('AM')~=1 | exist('EM')~=1 | exist('GM1')~=1 | exist('GM2')~=1)
    loader
end
xinco = [45 0 0 0 0 0 0 0 0 0 0 0]; xfix = 1; [sys,x0] = beaver([],[],[],0);
clear sys x0
skip = 0;
disp('Trim routine for BEAVER model');
disp('-----');
opt = menu('Which type of steady-state flight do you want?',...
    'steady wings-level flight','steady turning flight',...
    'steady pull-up','steady roll','quit');

if opt == 1                                % STEADY WINGS-LEVEL FLIGHT
    disp('Steady wings-level flight.');
    V = input('Give desired airspeed [m/s], default = 45: ');
    if isempty(V)
        V = 45;
    end
    H = input('Give (initial) altitude [m], default = 0: ');
    if isempty(H)
        H = 0;
    end
```



```

psi = input('Give heading [deg], default = 0: ')*pi/180;
if isempty(psi)
    psi = 0;
end
gamma = input('Give desired flightpath angle [deg], default = 0: ')*pi/180;
if isempty(gamma)
    gamma = 0;
end
phidot = 0;
psidot = 0;
thetadot = 0;
rolltype = 's';
elseif opt == 2                                % STEADY TURNING FLIGHT
    disp('Steady turning flight.');
    ok = 0;
    while ok ~= 1
        turntype = input('Coordinated or Uncoordinated turn (c/u)? ','s');
        if turntype == 'c' | turntype == 'u'
            ok = 1;
        end
    end
    V = input('Give desired airspeed [m/s], default = 45: ');
    if isempty(V)
        V = 45;
    end
    H = input('Give initial altitude [m], default = 0: ');
    if isempty(H)
        H = 0;
    end
    psi = input('Give initial heading [deg], default = 0: ')*pi/180;
    if isempty(psi)
        psi = 0;
    end
    gamma = input('Give desired flightpath angle [deg], default = 0: ')*pi/180;
    if isempty(gamma)
        gamma = 0;
    end
    phidot = 0;
    thetadot = 0;
    answ = input('Do you want to specify the turnrate (1) or turnradius (2)? ');
    if answ == 1
        psidot = input('Give desired rate of turn [deg/s], default = 0: ')*pi/180;
        if isempty(psidot)
            psidot = 0;
        end
    elseif answ == 2
        R = input('Desired turn radius (>0) [m], default = straight & level: ');
        if isempty(R) ~= 1
            psidot = V/R;
        else
            psidot = 0;
        end
    end
    rolltype = 's';
elseif opt == 3                                % STEADY PULL-UP
    disp('Steady pull-up.');
    V = input('Give desired airspeed [m/s], default = 45: ');
    if isempty(V)
        V = 45;
    end
    H = input('Give initial altitude [m], default = 0: ');
    if isempty(H)
        H = 0;
    end
    psi = input('Give initial heading [deg], default = 0: ')*pi/180;
    if isempty(psi)
        psi = 0;
    end
    gamma = input('Give initial flightpath angle [deg], default = 0: ')*pi/180;
    if isempty(gamma)
        gamma = 0;
    end
    phidot = 0;
    psidot = 0;

```

```

thetadot = input('Give pull-up rate [deg/s], default = 0: ')*pi/180;
if isempty(thetadot)
    thetadot = 0;
end
rolltype = 's';
elseif opt == 4 % STEADY ROLL
    disp('Steady roll.');
    V = input('Give desired airspeed [m/s], default = 45: ');
    if isempty(V)
        V = 45;
    end
    H = input('Give initial altitude [m], default = 0: ');
    if isempty(H)
        H = 0;
    end
    psi = input('Give initial heading [deg], default = 0: ')*pi/180;
    if isempty(psi)
        psi = 0;
    end
    gamma = input('Give desired flightpath angle [deg], default = 0: ')*pi/180;
    if isempty(gamma)
        gamma = 0;
    end
    thetadot = 0;
    psidot = 0;
    phidot = input('Give desired roll-rate [deg/s], default = 0: ')*pi/180;
    if isempty(phidot)
        phidot = 0;
    end
    ok = 0;
    while ok== 1
        if phidot == 0
            rolltype = input('Roll in stability or body-axes (b/s)? ','s');
        end
        if rolltype == 'b' | rolltype == 's'
            ok = 1;
        end
    end
else
    skip = 1;
end
%%%
if skip == 1 % DEFINE FLIGHT-CONDITION, IF NOT QUITTING
    deltaf = input('Give flap angle [deg], default = 0: ')*pi/180;
    if isempty(deltaf)
        deltaf = 0;
    end
    n = input('Give engine speed [RPM], default = 1800: ');
    if isempty(n)
        n = 1800;
    end
    pz = input('Give estimation of manifold pressure pz ["Hg], default = 20: ');
    if isempty(pz)
        pz = 20;
    end
    G = psidot*V/9.80665;
    Hdot = V*sin(gamma);
    if opt == 2 & turntype == 'u'
        phi = input('Give desired roll angle phi [deg], default = 0: ')*pi/180;
    end
    if isempty(phi)
        phi = 0;
    end
% All variables which specify the flight condition will be put in the
% vector ctrim. The variables which are varied by the minimization
% routine FMINS will be put into the vector vtrim.
% Definition: vtrim = [alpha beta deltae deltaa deltar pz].
% -----
ctrim = [V H psi gamma G psidot thetadot phidot deltaf n phi]';
vtrim = [0 0 0 0 0 pz]';

```



```

% Trim options and other trim-blabla
% -----
opts      = foptions([]);
opts(2)   = 1e-10;
opts(3)   = 1e-10;
modfun   = ['xdot = beaver(0,x,u,1);'];

iterate = 1;
while iterate
    disp('Searching for stable solution. Wait a moment... ');
    disp('');
    disp('(You may now feel an urgent desire to have a coffee');
    disp(' break... Well, if you use a 386SX, you have plenty');
    disp(' of time to give in to your desire!)');

    % Call minimization routine FMINS. A scalar cost function is contained
    % in the M-file ACCOST, which also calls the constraints for coordinated
    % turns and rate of climb. FMINS returns the trim values of alpha, beta,
    % deltae, deltaaa, deltara, and pz in the vector trimmed.
    % -----
    [vtrimmed,options] = fmins('accost',vtrim,[],options, ...
                                ctrim,rolltype,turntype,modfun);
    if options(10) == options(14)
        disp('Warning: solution hasn''t converged.');
        answ = input('Perform more iterations (y/n)? ','s');
        if answ == 'y'
            iterate = 1;
            vtrimmed = vtrim;
        else
            iterate = 0;
        end
    else
        iterate = 0;
    end
end
                                         % STOP ITERATION
disp('Iteration stopped.');
disp('');
disp('<<<Press key to get results>>>');
pause

% Call constraints once more to obtain the final values of the inputvector
% and state vector.
% -----
[x,u] = acconstr(vtrimmed,ctrim,rolltype,turntype);

eval(modfun);
disp('State vector (trimmed): ');
x
disp('<<< Press key >>>');
pause
disp('Input vector (trimmed): ');
u
disp('<<< Press key >>>');
pause
disp('Time derivative of state vector: ');
xdot

% Reconfigure variables for use in aircraft simulation model BEAVER.
% -----
xinco = x;                      % Initial value of state vector
xdot0 = xdot;                    % Initial value of time-derivative of state vector
ua0   = u(1:4);                  % Initial value of input vector for aerodynamic model
ut0   = u(5:6);                  % Initial value of input vector for engine
                                    % forces and moments model
clear x xdot u
answ = input('Save trimmed condition to file (y/n)? ','s');
if answ == 'y'
    disp('Description of flight condition (1 line max.): ');
    flighttype = input('> ','s');
    disp('Enter path\filename (filename max. 8 characters): ');
    name = input('> ','s');
    savecmmnd = ['save ',name,' xinco ua0 ut0 xdot0 flighttype'];
    eval(savecmmnd);
end

```

```

clear name savecmmnd
disp('The results are stored in the following variables:');
disp('');
disp('xinco = [V alpha beta p q r psi theta phi xe ye H]'' = state vector');
disp('xdot0 = dx/dt(0)');
disp('ua0   = [deltae deltaaa deltar deltaf]'' = initial input vector for');
disp('                                aerodynamic model');
disp('ut0   = [n pz]'' = initial input vector for engine model');
disp('');
answ = input('Keep flight-condition definition in workspace (y/n)?','s');
if answ == 'n'
    clear iterate modfun ok ctrim vtrim vtrimmmed opt opts rolltype H Hdot
    clear turntype n V deltar gamma phi phidot psi psidot pz thetadot G
end
                                         % END OF TRIM-LOOP (SKIPPED IF OPTION 5 IS SELECTED)
clear skip answ
disp('Ready.');
-----
% Programmed by: Marc Rauw, August 1, 1993.

```

Listing of ACCONSTR.M:

```

function [x,u] = acconstr(vtrim,ctrim,rolltype,turntype)
%=====
% ACCONSTR.m computes the coordinated turn constraint and the rate of climb
% constraint for the aircraft trim routine. This program is based upon a trim
% program for the Citation 500, written by R. Bogers.
%=====

% vtrim = [alpha beta deltae deltaaa deltar pz]
% ctrim = [V H psi gamma G psidot thetadot phidot deltar n phi]
% If turntype = 'c', turns will be coordinated, and the value of phi from
% ctrim won't be used. If turntype = 'u' the turn is uncoordinated and the
% roll angle phi, defined in ctrim will be used.
% -----
% Helpvariables
% -----
sinalpha = sin(vtrim(1));
cosalpha = cos(vtrim(1));
tanalpha = tan(vtrim(1));
sinbeta = sin(vtrim(2));
cosbeta = cos(vtrim(2));
singamma = sin(ctrim(4));

if turntype == 'c';
    % Coordinated turn constraint on phi
    % -----
    a = 1 - ctrim(5) * sinalpha/cosalpha * sinbeta;
    b = singamma/cosbeta;
    c = 1 + (ctrim(5)*cosbeta)^2;
    num = cosbeta*((a-b^2)+b*tanalpha*sqrt(c*(1-b^2)+ctrim(5)^2*sinbeta^2));
    den = cosalpha*(a^2-b^2*(1+c*tanalpha^2));
    phi = atan(ctrim(5)*num/den);
else
    phi = ctrim(11);
end
sinphi = sin(phi);
cosphi = cos(phi);

% Rate-of-Climb constraint on theta
% -----
a = cosalpha*cosbeta;
b = sinphi*sinbeta + cosphi*sinalpha*cosbeta;
num = a*b + singamma*sqrt(a^2 - singamma^2 + b^2);
den = a^2 - singamma^2;
theta = atan(num/den);
sintheta = sin(theta);
costheta = cos(theta);
if rolltype == 'b'      % Body-axes roll
    sinalpha = 0;
    cosalpha = 1;
end

```



```

p = ctrim(8)*cosalpha - sintheta*ctrim(6);
q = cosphi*ctrim(7) + sinphi*cosheta*ctrim(6);
r = -sinphi*ctrim(7) + cosphi*cosheta*ctrim(6) + sinalpha*ctrim(8);

x = [ctrim(1) vtrim(1) vtrim(2) p q r ctrim(3) theta phi 0 0 ctrim(2)]';
u = [vtrim(3) vtrim(4) vtrim(5) ctrim(9) ctrim(10) vtrim(6) 0 0 0 0 0 0]';
-----
% Program written by: Marc Rauw, August 1, 1993.

```

Listing of ACCOST.M:

```

function J = accost(vtrim,ctrim,rolltype,turntype,modfun);
-----
% ACCOST.m computes a scalar cost function for the minimization routine of the
% aircraft trim program ACTRIM. This program is based upon a trim routine for
% the Citation 500 aircraft, written by R. Bogers.
-----
% Call constraints for rate of climb and coordinated turns.
-----
[x,u] = acconstr(vtrim,ctrim,rolltype,turntype);

% Compute the time derivative of the state vector x.
% -----
eval(modfun);

% Scalar cost function.
%
J = xdot(1)^2 + 2*(xdot(2)^2+xdot(3)^2) + 5*(xdot(4)^2+xdot(5)^2+xdot(6)^2);
%
% Program written by: Marc Rauw, August 1, 1993.

```

Note that the following cost function J has been implemented in ACCOST.M:

$$J = \dot{V}^2 + 2(\dot{\alpha}^2 + \dot{\beta}^2) + 5(\dot{p}^2 + \dot{q}^2 + \dot{r}^2)$$

This cost-function performed well for the 'Citation 500' model of R. Bogers, and the results for the 'Beaver' model were in accordance with ref.[30]. Here, the angular accelerations have been weighted most heavily, but for some purposes, it might be better to choose other weighting constants.

The trim program *ACTRIM* can be started by typing *actrim* at the MATLAB command line, or by double-clicking the block *Find steady-state trim values of x and u for the BEAVER*, which can be found in the sublibrary *Aircraft trim and linearization* in *ACLIB*. Before starting a simulation of the system *BEAVER*, the initial conditions *xinco*, *ua0*, and *ut0* must be present in the MATLAB workspace. So it is necessary to either run *ACTRIM*, define the initial conditions manually, or load them with *INCOLOAD*.

- 2 - *ACLIN*. This M-file can be used to linearize the system *BEAVER* in the working point, defined by the initial conditions *xinco*, *ua0*, and *ut0*. The program can be started by typing *aclin* at the MATLAB command line, or by clicking the block *Full-blown linearization of BEAVER model* from the sublibrary *Aircraft trim and linearization* of *ACLIB*.

In this sublibrary, there is also a block called *Tiny linearization of BEAVER model*, which gives the same result as:

```
[Abeav,Bbeav,Cbeav,Dbeav] = linmod('beaver',xinco,[ua0;ut0;0;0;0;0;0;0])
```

When using this 'tiny' linearization command, the resulting linear state-space model will use the small-perturbations versions of the input, state, and output vectors of the system *BEAVER*. As illustrated in chapter 6, this type of linear aircraft model can easily be implemented in systems which originally called the nonlinear aircraft model *BEAVER*. The state vector still includes the coordinates x_e and y_e , even though those results are not accurate because the deviations from the nominal velocity components are integrated instead of the actual airspeed. In this way it is possible to replace the nonlinear aircraft model by the linear model in a straightforward manner.

The 'full-blown' linearization program *ACLIN* gives additional linearization options, although the linearization process itself is started with the same command as used by the 'tiny' linearization. First, the routine *FIXSTATE* is called, which makes it possible to fix state variables to their initial value, for instance, if the user wants to eliminate symmetrical or asymmetrical aircraft dynamics from the model. Then the system *BEAVER* will be linearized, which yields the state-space model {*Abeav*, *Bbeav*, *Cbeav*, *Dbeav*}.

It is possible to extract symmetrical and asymmetrical models from that linear state-space model, yielding the state-space systems {*Asymm*, *Bsymm*} and {*Aasym*, *Basym*} respectively. Note: no output equations are added to the symmetrical and asymmetrical models! The user must specify output equations manually, or change the M-file *ACLIN.M* according to his/her own wishes. Both symmetrical and asymmetrical models will be created, even if the user specified to fix the symmetrical or asymmetrical state variables before. It is possible to include engine and wind inputs to both the symmetrical and asymmetrical models, and the user can specify if H and ψ must be included to the state equations. The definitions of the input and state vectors will always be displayed when *ACLIN* has been used!

For both linearization blocks, it is necessary to specify the initial condition first; the linear model will be obtained by perturbing the input and state variables around their initial conditions!

List of variables, used in *aclin.m*:

<i>Aasym</i>	A-matrix of asymmetrical linear 'Beaver' model
<i>Abeav</i>	A-matrix of general linear 'Beaver' model
<i>a_eng</i>	<i>a_eng</i> = 'y' if the user wants to include engine effects in the asymmetrical state equations, else <i>a_eng</i> = 'n'
<i>ainput</i>	string variable, which contains the definition of the input vector to the asymmetrical linear state-space model
<i>answ</i>	<i>answ</i> = 'y' if the user wants to extract symmetrical and asymmetrical state-space models, else <i>answ</i> = 'n'
<i>answ1</i>	<i>answ1</i> = 'y' if the user wants to save the results, else <i>answ1</i> = 'n'
<i>answ2</i>	<i>answ2</i> = 'y' if the user wants to save the initial condition to the same file as the linearized aircraft model, else <i>answ2</i> = 'n'
<i>astate</i>	string variable, which contains the definition of the state vector to the asymmetrical linear state-space model
<i>Asymm</i>	A-matrix of symmetrical linear 'Beaver' model



a_wind	a_wind = 'y' if the user wants to include wind inputs to the asymmetrical state equations, else a_wind = 'n'
Ba_eng	contribution to B-matrix of asymmetrical linear 'Beaver' model, which brings into account the engine inputs
Basym	B-matrix of asymmetrical linear 'Beaver' model
Ba_wind	contribution to B-matrix of asymmetrical linear 'Beaver' model, which brings into account the wind inputs
Bbeav	B-matrix of general linear 'Beaver' model
Bs_eng	contribution to B-matrix of symmetrical linear 'Beaver' model, which brings into account the engine inputs
Bsymm	B-matrix of symmetrical linear 'Beaver' model
Bs_wind	contribution to B-matrix of symmetrical linear 'Beaver' model, which brings into account the wind inputs
Cbeav	C-matrix of general linear 'Beaver' model
Dbeav	D-matrix of general linear 'Beaver' model
flighttype	string variable which contains some user-specified information about the steady-state flight condition (this variable will be present in the MATLAB workspace, along with <i>xinco</i> , <i>ua0</i> , and <i>ut0</i> if the trimmed flight condition has been obtained with the program <i>ACTRIM</i>)
Htoo	Htoo = 'y' if the user wants to add the \dot{H} -equation to the symmetrical linear 'Beaver' model, else Htoo = 'n'
name	name of MAT-file (including path) to which models must be saved
psitoo	psitoo = 'y' if the user wants to add the ψ -equation to the asymmetrical linear 'Beaver' model, else psitoo = 'n'
savecmmnd	string variable, containing the specification of the save command, used to store the linearized aircraft models in a MAT-file
s_eng	s_eng = 'y' if the user wants to include engine effects in the symmetrical state model of the 'Beaver', else s_eng = 'n'
sinput	string variable, which contains the definition of the input vector, used in the symmetrical linear 'Beaver' model
sstate	string variable, which contains the definition of the state vector, used in the symmetrical linear 'Beaver' model
s_wind	s_wind = 'y' if the user wants to include wind effects in the symmetrical state model of the 'Beaver', else s_wind = 'n'
ua0	$\mathbf{u}_a(0)$ = inputvector to aerodynamic model at $t = 0$ $(\mathbf{u}_a = [\delta_e \ \delta_a \ \delta_r \ \delta_f]^T)$
uinco	$uinco = [ua0; ut0; 0; 0; 0; 0; 0; 0; 0]$
ut0	$\mathbf{u}_t(0)$ = inputvector to engine model at $t = 0$ ($\mathbf{u}_t = [n \ p_z]^T$)
xfix	vector of length 12, with elements that equal 0 if the corresponding state variable should be fixed, or 1 if it shouldn't (see <i>FIXSTATE</i>)
xinco	$\mathbf{x}(0)$ = state vector of BEAVER at $t = 0$

Listing of ACLIN.M:

```

%
% ACLIN linearizes the system BEAVER for x = xinco, u = [ua0; ut0; 0; 0; 0; 0; 0].
% The initial state vector xinco, the initial inputvector to the aerodynamic
% model ua0, and the initial inputvector to the engine model ut0 can all be
% computed for trimmed flight with ACTRIM and saved to a file, which can be
% retrieved with INCOLOAD. So either compute a new trim condition before running
% ACLIN, or load a trim condition into the Matlab workspace. It is possible to
% neglect symmetrical or asymmetrical motions, or to fix arbitrary states.
% Furthermore, the symmetrical and asymmetrical states can be used as in the
% nonlinear model, or be separated from eachother in separate linear state-space
% models.
%
```

```

% Check if aircraft model parameters are present in Matlab workspace.
% -----
if (exist('AM')~=1 | exist('EM')~=1 | exist('GM1')~=1 | exist('GM2')~=1)
    loader
end

% Check if initial condition has been specified.
% -----
if (exist('xinco')~=1 | exist('ua0')~=1 | exist('ut0')~=1)
    disp('Define xinco (= initial value of state vector), ua0 (= initial');
    disp('value of input vector to aerodynamic model), and ut0 (= initial');
    disp('value of input vector to engine model) before linearizing!');
    disp('');
    disp('x = [V alpha beta p q r psi theta phi xe ye H]''');
    disp('ua = [deltae deltaaa deltar deltarf]''');
    disp('ut = [n pz]''');
    error
end

disp('Linearize aircraft model.');
disp('=====');
disp('First, select which states need to be fixed in case you want to');
disp('simplify the aircraft dynamics further.');
disp('');

% If part of aircraft dynamics are neglected, xfix should be set. Type
% HELP FIXSTATE or TYPE FIXSTATE at the Matlab command line for more details.
% -----
fixstate

% Linearize the aircraft model in point, defined by xinco, ua0, and ut0.
% Standard SIMULINK linearization is used, change perturbation levels, or
% linearization algorithm if that gives better results.
% -----
disp('Now linearizing...');

uinco = [ua0; ut0; 0; 0; 0; 0; 0; 0];
[Abeav, Bbeav, Cbeav, Dbeav] = linmod('beaver', xinco, uinco);
clear uinco
disp('');
disp('Linear aircraft model is now contained in state-space system');
disp('Abeav, Bbeav, Cbeav, Dbeav');

if length(xfix) == 12
    if xfix(1) == 0, disp('V has been fixed'); end
    if xfix(2) == 0, disp('alpha has been fixed'); end
    if xfix(3) == 0, disp('beta has been fixed'); end
    if xfix(4) == 0, disp('p has been fixed'); end
    if xfix(5) == 0, disp('q has been fixed'); end
    if xfix(6) == 0, disp('r has been fixed'); end
    if xfix(7) == 0, disp('psi has been fixed'); end
    if xfix(8) == 0, disp('theta has been fixed'); end
    if xfix(9) == 0, disp('phi has been fixed'); end
    if xfix(10) == 0, disp('xe has been fixed'); end
    if xfix(11) == 0, disp('ye has been fixed'); end
    if xfix(12) == 0, disp('ze has been fixed'); end
end

answ = input('Extract separate symm. & asymm. models (y/n)? ','s');
disp('');
if answ == 'y'
    % Extract model with symmetrical aircraft dynamics.
    % -----
    disp('Symmetrical aircraft model');
    disp('=====');
    Asymm = [Abeav(1,1) Abeav(1,2) Abeav(1,5) Abeav(1,8);
              Abeav(2,1) Abeav(2,2) Abeav(2,5) Abeav(2,8);
              Abeav(5,1) Abeav(5,2) Abeav(5,5) Abeav(5,8);
              Abeav(8,1) Abeav(8,2) Abeav(8,5) Abeav(8,8)];
    Bsimm = [Bbeav(1,1) Bbeav(1,4);
              Bbeav(2,1) Bbeav(2,4);
              Bbeav(5,1) Bbeav(5,4);
              Bbeav(8,1) Bbeav(8,4)];
    sinput = ['symmetrical inputvector: [deltae deltarf'];
end

```



```

% Enhance B-matrix if engine speed and manifold pressure are used as inputs
% -----
s_eng = input('Include engine effects in symmm. model (y/n)? ','s');
if s_eng == 'y'
    Bs_eng = [Bbeav(1,5) Bbeav(1,6);
               Bbeav(2,5) Bbeav(2,6);
               Bbeav(5,5) Bbeav(5,6);
               Bbeav(8,5) Bbeav(8,6)];
    Bsymm = [Bsymm Bs_eng];
    sinput = [sinput ' n pz'];
end

% Enhance B-matrix if wind velocities and accelerations are used as inputs.
% -----
s_wind = input('Include wind inputs in symmm. model (y/n)? ','s');
if s_wind == 'y'
    Bs_wind = [Bbeav(1,7:12);
                Bbeav(2,7:12);
                Bbeav(5,7:12);
                Bbeav(8,7:12)];
    Bsymm = [Bsymm Bs_wind];
    sinput = [sinput ' uw vw ww uwdot vwdot wwdot'];
end
sinput = [sinput '''];

% Enhance A and B matrices if H is added to symmetrical state vector
% (dimensions of B depend upon symmm. input vector which is currently used).
% -----
Htoo = input('Do you want to add H-equation to model (y/n)? ','s');
if Htoo == 'y'
    Asymm=[Asymm [Abeav(1,12);Abeav(2,12);Abeav(5,12);Abeav(8,12)]];
    Asymm=[Asymm; [Abeav(12,1) Abeav(12,2) Abeav(12,5) Abeav(12,8),
                    Abeav(12,12)]];
    if s_eng == 'y'
        if s_wind == 'y'
            Bsomm=[Bsomm; [Bbeav(12,1) Bbeav(12,4) Bbeav(12,5) Bbeav(12,6:12)]];
        else
            Bsomm=[Bsomm; [Bbeav(12,1) Bbeav(12,4) Bbeav(12,5) Bbeav(12,6)]];
        end
    else
        if s_wind == 'y'
            Bsomm=[Bsomm; [Bbeav(12,1) Bbeav(12,4) Bbeav(12,7:12)]];
        else
            Bsomm=[Bsomm; [Bbeav(12,1) Bbeav(12,4)]];
        end
    end
    disp('H-equation added');
    disp('');
    sstate = ['symmetrical state vector: [V alpha q theta H]'''];
else
    sstate = ['symmetrical state vector: [V alpha q theta]'''];
end

% Extract model with asymmetrical aircraft dynamics.
% -----
disp('Asymmetrical aircraft model');
disp('');
Aasym = [Abeav(3,3) Abeav(3,4) Abeav(3,6) Abeav(3,9);
          Abeav(4,3) Abeav(4,4) Abeav(4,6) Abeav(4,9);
          Abeav(6,3) Abeav(6,4) Abeav(6,6) Abeav(6,9);
          Abeav(9,3) Abeav(9,4) Abeav(9,6) Abeav(9,9)];

Basym = [Bbeav(3,2) Bbeav(3,3);
          Bbeav(4,2) Bbeav(4,3);
          Bbeav(6,2) Bbeav(6,3);
          Bbeav(9,2) Bbeav(9,3)];

ainput = ['asymmetrical inputvector: [deltaaa deltar']];

% Enhance B-matrix if engine speed and manifold pressure are used as inputs
% -----
a_eng = input('Include engine effects in asymmm. model (y/n)? ','s');

```

```

% Include symmetrical/asymmetrical state-space matrices if these exist.
% -----
if answ == 'y'
    savecmmnd = [savecmmnd 'Asymm Bsymb Aasym Basym '];
end

% Include initial condition (working point) if required.
% -----
answ2 = input('Include initial condition to this file (y/n)? ','s');
if answ2 == 'y'
    savecmmnd = [savecmmnd 'xinco ua0 ut0 '];

% Include description of flight condition if this is present in
% Matlab workspace.
% -----
if exist('flighttype') == 1
    savecmmnd = [savecmmnd 'flighttype '];
end
eval(savecmmnd);
clear name savecmmnd
end

clear sinput ainput sstate astate s_eng a_eng s_wind a_wind Bs_eng Ba_eng
clear Bs_wind Ba_wind Htoo psitoo answ answ1 answ2

disp('Check current definition of outputvector of the system BEAVER');
disp('(i.e., the meaning of Cbeav and Dbeav). The state-space model');
disp('{Abeav, Bbeav, Cbeav, Dbeav} can replace the nonlinear model in');
disp('systems which call BEAVER via an S-function block. But remember');
disp('that this is a small-perturbation model, which uses deviations');
disp('from nominal inputs, states, and outputs, whereas the nonlinear');
disp('model uses the actual values of these variables.');
disp('');
disp('READY.');
%-----
% Program written by Marc Rauw. Version: August 1, 1993.

```

- 3 - *INCOLOAD*. This M-file is a simple load routine, which can be used to retrieve MAT-files from file into the MATLAB workspace. Although this routine has been included for loading initial conditions, saved with *ACTRIM*, or linearized aircraft models, saved with *ACLIN*, it can be used to retrieve arbitrary MAT-files. *INCOLOAD* can be called by typing *inco-load* at the MATLAB command line, or by double-clicking the *Load initial conditions or models* block from the library *ACTRIM*.

The listing of *INCOLOAD* is given below:

```

%-----
% INCOLOAD loads MAT-files into the Matlab workspace.
% -----
disp('Enter path\filename.ext (filename max. 8 characters)');
disp('(without extension, a .MAT file will be loaded):');
disp('');
name = input('> ','s');
loadcmmnd = ['load ',name];
eval(loadcmmnd);
clear name loadcmmnd
who
%-----
% Program written by: Marc Rauw. Version: july 2, 1993.

```

name is the filename (+path) and *loadcmmnd* contains the actual load command.

F.3 Models of wind and atmospheric turbulence.

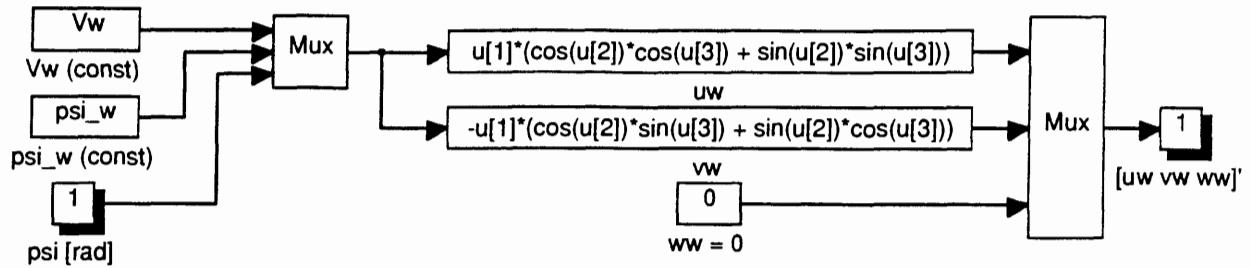
F.3.1 Introduction.

In appendix B, some expressions for wind and atmospheric turbulence were given. These models have been implemented as separate blocks in SIMULINK. Via the *Inport* blocks for the wind velocities and their time-derivatives, shown in figure F-1, it becomes possible to connect the wind and turbulence models to the system *BEAVER*. The time derivatives of the wind can be calculated by applying a *Derivative* block of SIMULINK (see appendix E). The blocks for atmospheric turbulence already calculate the time derivatives of the turbulence velocities themselves.

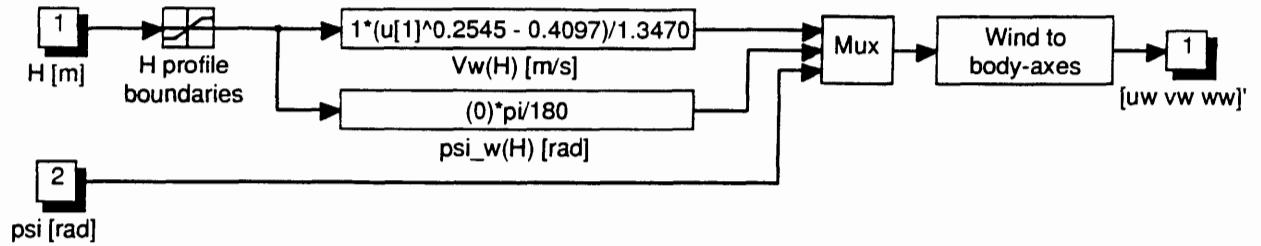
F.3.2 List of variables, used for the wind and turbulence models.

a0, a1, ...	a_i	= denominator coefficients of second (or higher) order filter:
		$\frac{b_n s^{n-1} + b_{n-1} s^{n-2} + \dots + b_1 s + b_0}{s^n + a_n s^{n-1} + \dots + a_1 s + a_0}$
H	H	= altitude; [m]
Lug	L_{u_g}	= scale length of longitudinal turbulence; [m]
Lvg	L_{v_g}	= scale length of lateral turbulence; [m]
Lwg	L_{w_g}	= scale length of vertical turbulence; [m]
K	K	= coefficient of first order filter $\frac{K}{1 + \tau \cdot s}$
psi	ψ	= yaw angle; [rad]
psi_w	ψ_w	= wind direction (note: $\psi_w = \pi$ rad if the wind is blowing to the north); [rad]
sigma_ug	σ_{u_g}	= standard deviation of longitudinal turbulence; [m/s]
sigma_vg	σ_{v_g}	= standard deviation of lateral turbulence; [m/s]
sigma_wg	σ_{w_g}	= standard deviation of vertical turbulence; [m/s]
tau		= coefficient of first order filter $\frac{K}{1 + \tau \cdot s}$
ug	u_g	= contribution of atmospheric turbulence to u_w ; [m/s]
ugdot	\dot{u}_g	= time derivative of u_g ; [m/s ²]
uw	u_w	= component of wind velocity along X _B -axis; [m/s]
uwdot	\dot{u}_w	= time derivative of u_w ; [m/s ²]
V	V	= true airspeed; [m/s]
vg	v_g	= contribution of atmospheric turbulence to v_w ; [m/s]
vgdot	\dot{v}_g	= time derivative of v_g ; [m/s ²]
Vw	V_w	= wind velocity ($V_w = \sqrt{u_w^2 + v_w^2 + w_w^2}$); [m/s ²]
vw	v_w	= component of wind velocity along Y _B -axis; [m/s]
vwdot	\dot{v}_w	= time derivative of v_w ; [m/s ²]
wg	w_g	= contribution of atmospheric turbulence to w_w ; [m/s]
wgdot	\dot{w}_g	= time derivative of w_g ; [m/s ²]
ww	w_w	= component of wind velocity along Z _B -axis; [m/s]
wwdot	\dot{w}_w	= time derivative of w_w ; [m/s ²]

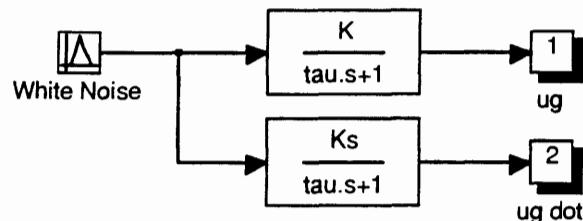




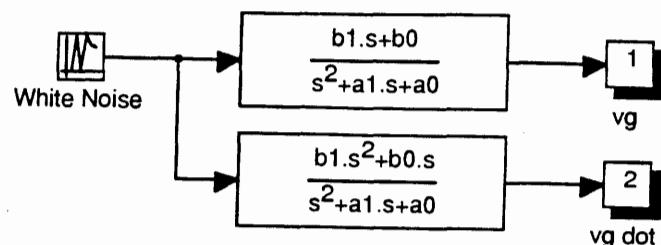
**Figure F-30. Internal structure of the block *CWIND*.
WINDLIB**



**Figure F-31. Internal structure of the block *BLWIND*.
WINDLIB**



**Figure F-32. Internal structure of the block *UDRYD1*.
WINDLIB**



**Figure F-33. Internal structure of the block *VDRYD1*.
WINDLIB**

F.3.3 Description of the wind and turbulence models and their helpfiles.

Models of wind and atmospheric turbulence have been included in the library **WINDLIB**. There are two different blocks which can be used to implement steady wind and a wind-profile in the earth's boundary layer respectively:

- 1 - **CWIND**. This block is used to implement wind with constant speed and direction ('C' = constant). With the *Mask* utility, a dialog-box was created, in which the user must enter the windspeed V_w and direction ψ_w (note: $\psi_w = \pi$ if the wind is blowing to the north). The block determines the components of the wind along the aircraft's body-axes. The vertical component w_w is assumed to be zero!

On-line help is available by typing *help cwind* at the MATLAB command line. See figure F-30 for the internal structure of **CWIND**.

Inputsignal: ψ
 Outputvector: $[u_w \ v_w \ w_w]^T \ (w_w \equiv 0)$

- 2 - **BLWIND**. This block calculates the components of the windspeed along the aircraft's body-axes in the boundary layer of the earth ('BL' = boundary layer). Both the wind velocity and its direction can be defined as functions of altitude. In figure F-31, the wind profile according to equation (B-1) of appendix B is implemented. The direction of the wind is here assumed to be constant.

Type *help blwind* at the MATLAB command line for on-line help.

Inputs: H, ψ
 Outputvector: $[u_w \ v_w \ w_w]^T \ (w_w \equiv 0)$

Time-derivatives of the wind velocity components can be obtained by using a SIMULINK *Derivative* block from the library **LINEAR**, which is accurate enough for slowly changing wind velocities¹⁾.

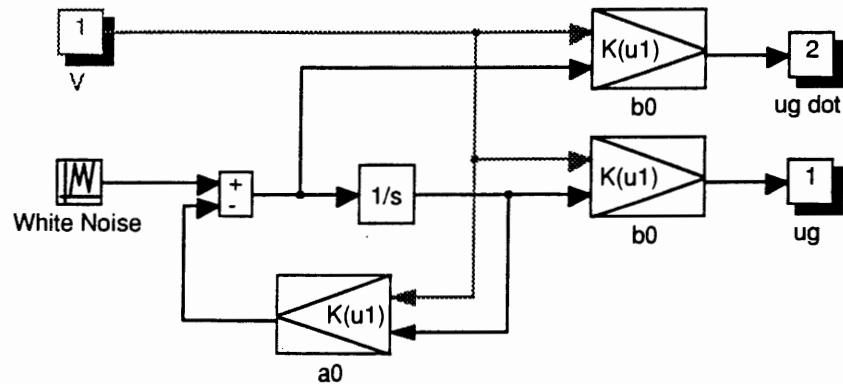
The library **WINDLIB** also contains six blocks for the calculation of turbulence velocities along the three body-axes of the aircraft. The following blocks are available:

- 1 - **UDRYD1**. Longitudinal Dryden filter with constant coefficients. The user must specify the scale length L_u , the standard deviation σ_u , and the velocity V . On-line help is available from the MATLAB command line (type *help udryd1*). See figure F-32 for the internal structure of **UDRYD1**.

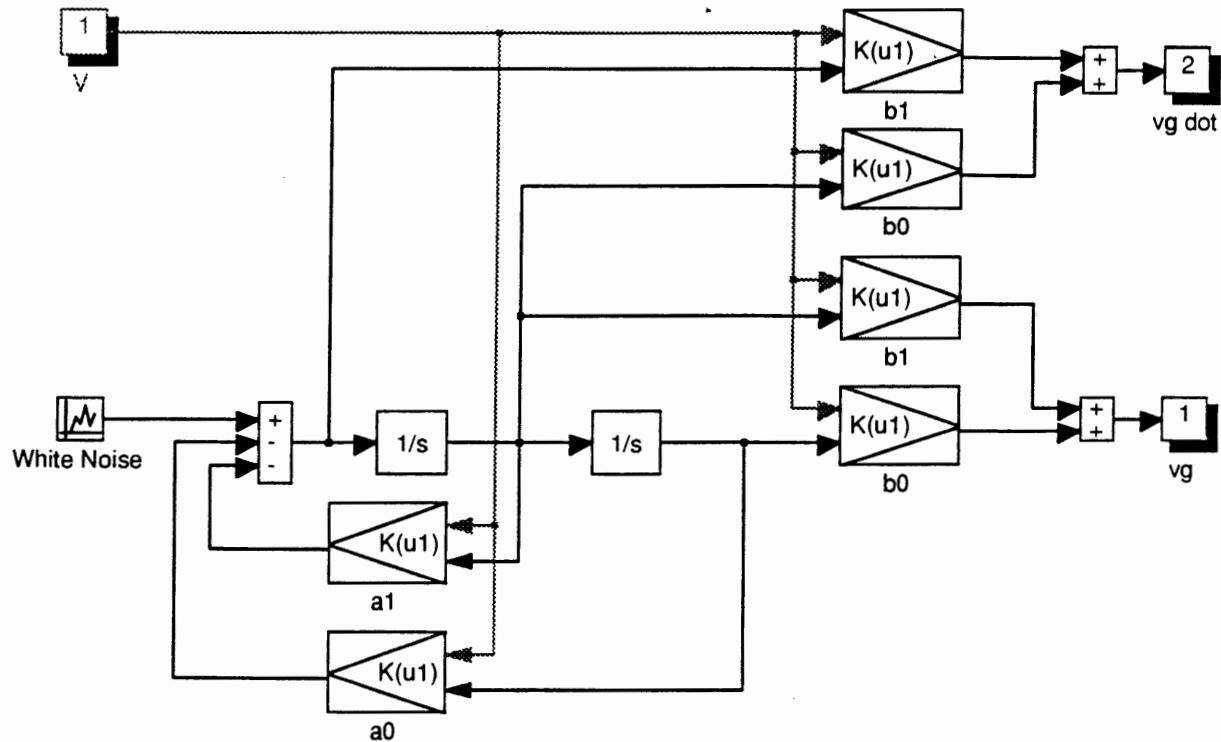
Outputsignals: u_g and \dot{u}_g .

¹⁾ The *Derivative* block approximates the derivative of its input with $\Delta u / \Delta t$, where Δu is the change in input value since the previous simulation step, and Δt is the change in time since the previous simulation timestep. The magnitude of the timesteps must be set to a value small enough for good approximation of the derivative.





**Figure F-34. Internal structure of the block *UDRYD2*.
WINDLIB**



**Figure F-35. Internal structure of the block *VDRYD2*.
WINDLIB**

- 2 - VDRYD1. Lateral Dryden filter with constant coefficients. The user must specify the scale length L_{v_g} , the standard deviation σ_{v_g} , and the velocity V. Type `help vdryd1` at the MATLAB command line for on-line help. Figure F-33 shows the internal structure of VDRYD1.

Outputsignals: v_g and \dot{v}_g .

- 3 - WDRYD1. Vertical Dryden filter with constant coefficients. The user must specify the scale length L_{w_g} , the standard deviation σ_{w_g} , and the velocity V. On-line help is available from the MATLAB command line (type `help wdryd1`). The internal structure of WDRYD1 looks similar to the structure of VDRYD1 if the subscripts v_g are exchanged for w_g . See figure F-33.

Outputsignals: w_g and \dot{w}_g

- 4 - UDRYD2. Longitudinal Dryden filter with coefficients which vary with V. Only L_u and σ_u have to be defined here; the velocity V is used as input signal. Type `help udryd2` at the MATLAB command line for on-line help. Figure F-34 shows the internal structure of UDRYD2.

Inputsignal: V

Outputsignals: u_g and \dot{u}_g

- 5 - VDRYD2. Lateral Dryden filter with coefficients which vary with V. The user must specify L_{v_g} and σ_{v_g} , but the airspeed V is used as input signal. Type `help vdryd2` at the MATLAB command line for on-line help. The internal structure of VDRYD2 is shown in figure F-35.

Inputsignal: V

Outputsignals: v_g and \dot{v}_g

- 6 - WDRYD2. Vertical Dryden filter with coefficients which vary with V. The user must specify L_{w_g} and σ_{w_g} , but the airspeed V is now used as input variable. Type `help wdryd2` at the MATLAB command line for on-line help. The internal structure of WDRYD2 looks similar to the structure of VDRYD2, if the subscripts v_g are exchanged for w_g . See figure F-35.

Inputsignal: V

Outputsignals: w_g and \dot{w}_g

For most purposes, the blocks UDRYD1, VDRYD1, and WDRYD1 will suffice. Only when the changes in airspeed are *very* large, it is useful to use UDRYD2, VDRYD2, or WDRYD2 instead. But remember that the Dryden turbulence filters are approximative anyway! For the 'Beaver', the filters with constant coefficients are good enough, because this aircraft has a very limited speed range. Note that the 'random' sequences of the *White Noise* blocks are actually reproducible. To make sure that the three turbulence velocities are independent, different values of the initial seeds were used for the different turbulence generators.



F.4 ILS approach and VOR navigation models.

F.4.1 Introduction.

It is possible to generate navigation signals from ILS and VOR systems within the simulation models, to simulate (automatic) approaches and short-range navigation. A detailed description of the equations which are used to calculate the ILS and VOR signals is included in appendix C. In this section, the ILS and VOR block-library will be presented. A list of all variables, available blocks, and helpfiles for the ILS and VOR models will now be given. The internal structure of each block will be shown and explained.

F.4.2 List of variables, used for the ILS and VOR models.

The following variables appear within programs, helpfiles, block-diagrams, and 'Mask' initialization lines of the ILS and VOR models:

cat	$cat =$ ILS performance category ($cat = 1, 2,$ or 3 for performance category I, II, or III respectively)
CD	$CD =$ Course Datum = reference VOR bearing (see figure C-11); [rad]
cos_gamgs	$\cos(\gamma_{gs})$
cos_psiRW	$\cos(\psi_{RW})$
dgs	$d_{gs} =$ distance from aircraft to glideslope reference line, see figure C-9; in [m]
D_igs	$\Delta i_{gs} =$ steady-state offset error in i_{gs} , due to a deviation of the nominal glideslope elevation angle (table C-2 and equation (C-7)); in [μA]
D_iloc	$\Delta i_{loc} =$ steady-state offset error in i_{loc} , if the localizer reference plane is not aligned along the runway centerline (table C-1 and equation (C-1)); in [μA]
epsilon_gs	ϵ_{gs} see figure C-9; in [rad]
gamgs	γ_{gs} see figure C-9; in [rad]
Gamma_gs	Γ_{gs} see figure C-9; in [rad]
Gamma_loc	Γ_{loc} see figure C-8; in [rad]
Gamma_VOR	$\Gamma_{VOR} = CD - QDR$ (see figure C-11); in [rad]
GSflag	glideslope flag. $GSflag = 0$ if the aircraft flies in area where the glideslope signal can be received with appropriate reliability; else, $GSflag = 1$ (see figure C-5)
H	$H =$ aircraft altitude above sea level; in [m]
Hf	$H_f =$ aircraft altitude above runway; in [m]
HRW	$H_{RW} =$ altitude of runway above sea level; in [m]
HVOR	$H_{VOR} =$ altitude of VOR transmitter above sea level; in [m]
igs	$i_{gs} =$ nominal value of current through glideslope indicator; in [μA]
igs_true	$i_{gs\ true} =$ actual value of current through glideslope indicator, including steady-state errors and/or noise; in [μA]
iloc	$i_{loc} =$ nominal value of current through localizer indicator; in [μA]
iloc_true	$i_{loc\ true} =$ actual value of current through localizer indicator, including steady-state errors and/or noise; in [μA]
K	$K =$ gain factor of 1 st -order transfer function $\frac{K}{1 + \tau s}$
KSgs	$KS_{gs} =$ correction factor to implement steady-state error in glideslope sensitivity S_{gs} ($S_{gs\ true} = KS_{gs} \cdot S_{gs}$)
KSloc	$KS_{loc} =$ correction factor to implement steady-state error in localizer sensitivity S_{loc} ($S_{loc\ true} = KS_{loc} \cdot S_{loc}$)

KVORerr	K_{VORerr} = correction factor to implement steady-state error in VOR signal $(\Gamma_{VOR\ true} = K_{VORerr} \cdot \Gamma_{VOR})$
Lgs	L_{gs} = scale length for glideslope noise, see equation (C-16); in [m]
Lloc	L_{loc} = scale length for localizer noise, see equation (C-15); in [m]
LOCflag	localizer flag. $LOCflag = 0$ if the aircraft flies in area where the localizer signal can be received with appropriate reliability; else, $LOCflag = 1$ (see figure C-3)
psi	ψ = yaw angle; in [rad]
psiRW	Ψ_{RW} = heading of the runway (take-off and landing direction); in [rad]
QDR	QDR = VOR bearing where aircraft currently flies (see figure C-11); [rad]
Rgs	R_{gs} = ground distance from aircraft to glideslope antenna (see figure C-9); in [m]
Rloc	R_{loc} = ground distance from aircraft to localizer antenna (see figure C-9); in [m]
R_VOR	R_{vor} = ground distance from aircraft to VOR station
Sgs	S_{gs} = glideslope sensitivity, see equation (C-8); in [$\mu\text{A}/\text{rad}$]
sigma_gs	σ_{gs} = standard deviation of glideslope noise; in [μA]
sigma_loc	σ_{loc} = standard deviation of localizer noise; in [μA]
sin_psiRW	$\sin(\psi_{RW})$
Sloc	S_{loc} = localizer sensitivity, see equation (C-2); in [$\mu\text{A}/\text{rad}$]
tan_gamgs	$\tan(\gamma_{gs})$
tau	τ = numerator coefficient of 1 st -order transfer function $\frac{K}{1 + \tau s}$
ToFrom	To/From indicator. $ToFrom = 1$ if aircraft flies TO the VOR station, and $ToFrom = 0$ if it flies FROM the VOR station.
V	V = true airspeed (here: approach speed); in [m/s]
xe	x_e = aircraft X-coordinate relative to earth-fixed reference frame (see figure C-7 for the definition of F_E that is used during ILS approach simulation); in [m]
xf	x_f = aircraft X-coordinate relative to runway-fixed reference frame (see figure C-7); in [m]
xgs	x_{gs} = X-coordinate of glideslope transmitter relative to runway-fixed reference frame (see figures C-1 and C-7); in [m]
xloc	x_{loc} = X-coordinate of localizer transmitter relative to runway-fixed reference frame (see figures C-1 and C-7); in [m]
xRW	x_{RW} = X-coordinate of origin of F_F , relative to F_E
xVOR	x_{VOR} = X-coordinate of VOR station, relative to F_E
ye	y_e = aircraft Y-coordinate relative to earth-fixed reference frame (see figure C-7 for the definition of F_E that is used during ILS approach simulation); in [m]
yf	y_f = aircraft Y-coordinate relative to runway-fixed reference frame (see figure C-7); in [m]
ygs	y_{gs} = Y-coordinate of glideslope transmitter, relative to runway-fixed reference frame (see figures C-1 and C-7); in [m]
yRW	y_{RW} = Y-coordinate of origin of F_F , relative to F_E
yVOR	y_{VOR} = Y-coordinate of VOR station, relative to F_E
zf	z_f = Z-coordinate of aircraft relative to runway-fixed reference frame ($z_f \equiv -H_f$); in [m]

Note: in dialog-boxes where the user must enter the values of some angles, *degrees* will be used instead of *radians*! Radians are used within the equations themselves. To avoid confusion, the unity which has to be used will always be displayed within the dialog-boxes themselves.



The following input and output vectors are used for the ILS and VOR blocks:

uils	\mathbf{u}_{ils}	$= [x_e \ y_e \ H]^T$
yils1	$\mathbf{y}_{ils\ 1}$	$= [i_{gs} \ i_{loc}]^T$
yils2	$\mathbf{y}_{ils\ 2}$	$= [\epsilon_{gs} \ \Gamma_{loc}]^T$
yils3	$\mathbf{y}_{ils\ 3}$	$= [x_f \ y_f \ H_f \ d_{gs} \ R_{gs} \ R_{loc}]^T$
yils4	$\mathbf{y}_{ils\ 4}$	$= [LOCflag \ GSflag]^T$
uvor1	$\mathbf{u}_{vor\ 1}$	$= [x_e \ y_e \ H]^T$
uvor2	$u_{vor\ 2}$	$= \psi$
yvor1	$y_{vor\ 1}$	$= \Gamma_{VOR}$
yvor2	$y_{vor\ 2}$	$= R_{VOR}$
yvor3	$\mathbf{y}_{vor\ 3}$	$= [Cone-of-silence-flag \ Range-flag]^T$
yvor4	$y_{vor\ 4}$	$= ToFrom (=1 if TO, 0 if FROM)$

Note: d_{loc} is not included in \mathbf{y}_{ils3} , because $d_{loc} \equiv y_f$.

F.4.3 Description of the ILS and VOR blocks and helpfiles.

The ILS and VOR blocks are stored in the library *NAVLIB*, which includes examples to demonstrate the practical use of these blocks. The following ILS blocks are available:

- 1 - ILS. In this block, the nominal signals are calculated. Also, localizer and glideslope flags will be set if the aircraft flies outside the area where the ILS signals can be received with appropriate accuracy. See section C.2.1 for details. The subsystem ILSTEST is used to calculate the localizer and glideslope flag settings. On-line help is available from the MATLAB command line (type *help ils* or *help ilstest*, respectively). The internal structure of *ILS* is shown in figure F-36. *ILSTEST* is shown in figure F-37.

Inputvector: \mathbf{u}_{ils}

Outputvectors: $\mathbf{y}_{ils\ 1}$, $\mathbf{y}_{ils\ 2}$, $\mathbf{y}_{ils\ 3}$, and $\mathbf{y}_{ils\ 4}$

The user must specify the position of the origin of the runway-fixed reference frame F_F , which is given by the coordinates x_{RW} , y_{RW} , and the altitude above sea-level H_{RW} . Other parameters which must be set are the heading of the runway, ψ_{RW} , the nominal glide path angle γ_{gs} , the distance from the runway threshold to the localizer transmitter, x_{loc} , the X-distance from the runway threshold to the glideslope transmitter, x_{gs} , and the Y-distance from the runway threshold to the glideslope transmitter, y_{gs} .

- 2 - LOCERR and GSEERR. These blocks are used for the calculation of steady-state errors in the localizer and glideslope signals, respectively. The maximum permissible steady-state errors according to ICAO guidelines will be used (see refs.[1], [2], or [16] and section C.2.2 of appendix C). The user must specify the percentage of the maximum permissible noise that is really needed for the simulation.

Inputsignal: i_{loc} or i_{gs} in [μA] (nominal values)

Outputsignal: i_{loc} or i_{gs} in [μA] (values with steady-state offset errors)

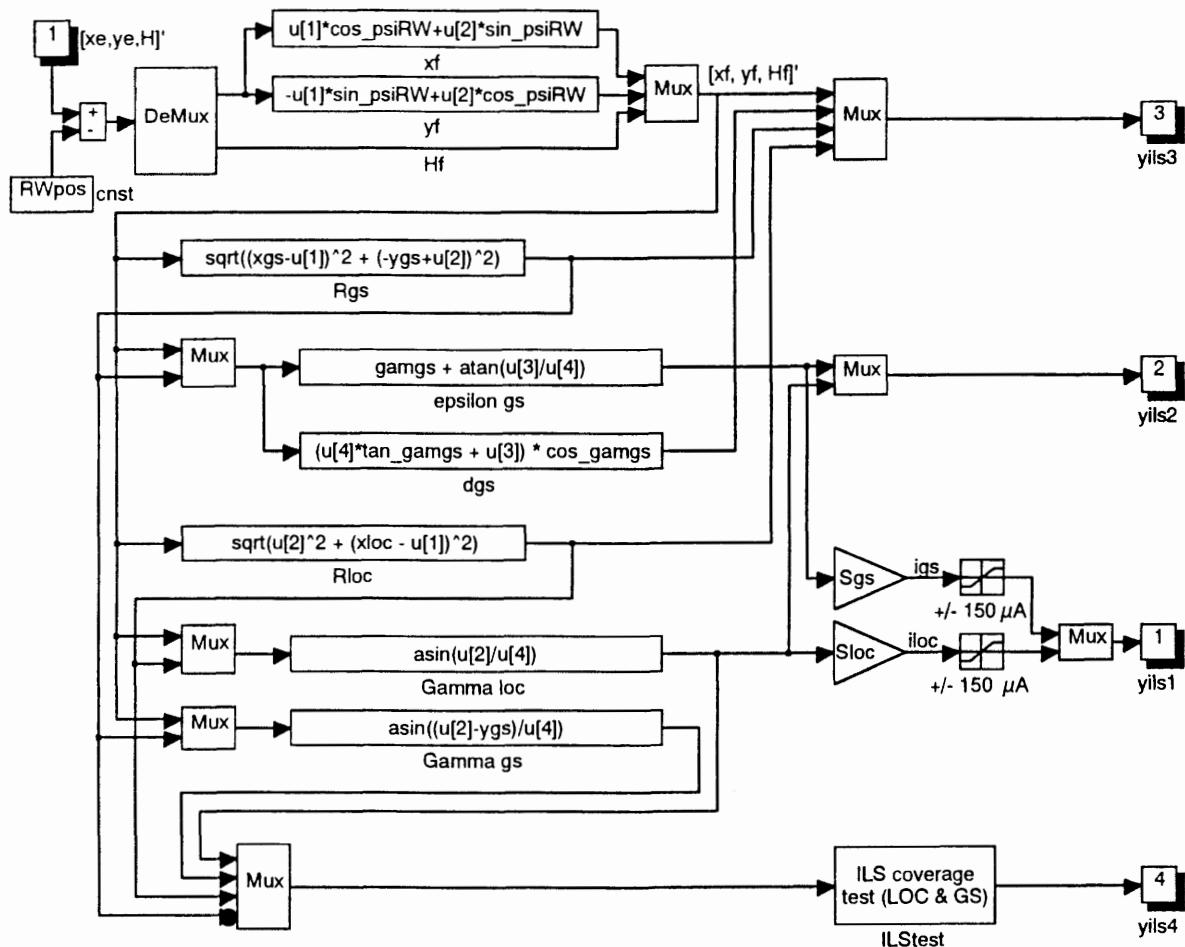


Figure F-36. Internal structure of the block ILS.
NAVLIB

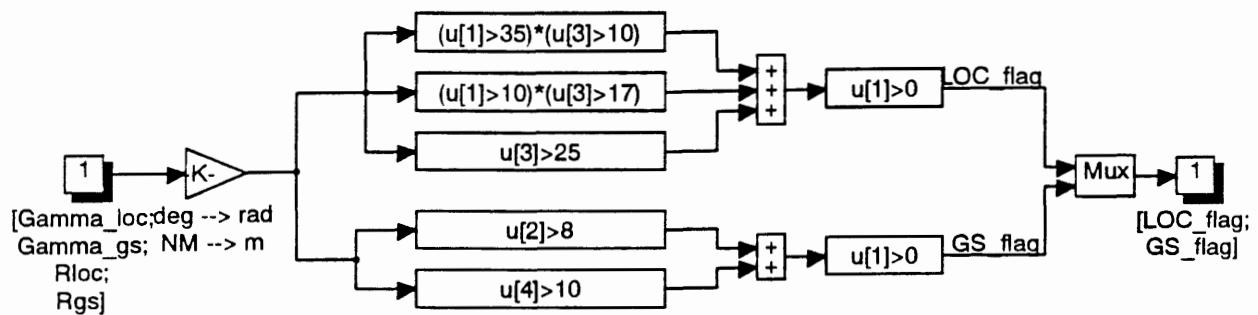


Figure F-37. Internal structure of the block ILSTEST.
NAVLIB: ILS \ ILSTEST



See figures F-38 and F-39. Type *help locerr* or *help gserr* for on-line help at the MATLAB command line.

Both blocks have been created with the *Mask* utility of SIMULINK. Every time a simulation is started, these blocks will be initiated. The following *Mask* initialization commands have been included:

```
LOCERR: xloc = @4; Sloc = 1.4*xloc; cat = @1;
if cat == 1
    D_iloc = Sloc*atan(10.5/xloc);
    KSloc = 1 + (@2/100)*0.17;
elseif cat == 2
    D_iloc = Sloc*atan(7.5/xloc);
    KSloc = 1 + (@2/100)*0.17;
else
    D_iloc = Sloc*atan(3/xloc);
    KSloc = 1 + (@2/100)*0.10;
end
D_iloc = D_iloc * @3/100;
```

@1 is the performance category (1, 2, or 3), @2 is the percentage of the maximum allowable error in localizer sensitivity, @3 is the percentage of the maximum allowable localizer misalignment, and @4 is the distance from the runway threshold to the localizer transmitter.

See table C-1 from appendix C (section C.2.2) for the ICAO guidelines concerning localizer steady-state errors. The arctan commands are used to transfer the maximum deviation of the localizer reference plane from the runway centerline at runway threshold into an error-angle $\Delta\Gamma_{loc}$.

```
GSERR: gamgs = -abs(@4*pi/180); Sgs = 625/abs(gamgs); cat = @1;
if cat == 1
    D_igs = Sgs*0.075*gamgs;
    KSgs = 1 + (@2/100)*0.25;
elseif cat == 2
    D_igs = Sgs*0.075*gamgs;
    KSgs = 1 + (@2/100)*0.20;
else
    D_igs = Sgs*0.040*gamgs;
    KSgs = 1 + (@2/100)*0.15;
end
D_igs = D_igs * @3/100
```

@1 is the performance category, @2 is the percentage of the maximum allowable error in the glideslope sensitivity, @3 is the percentage of the maximum glideslope misalignment, and @4 is the nominal glideslope elevation angle. The maximum permissible glideslope steady-state errors are listed in table C-2 of appendix C (section C.2.2).

3 - *LOCNOISE* and *GSNOISE*. These blocks are used to compute ILS noise. There are two types of ILS noise blocks, see appendix C, section C.2.3.

LOCNOISE1 and *GSNOISE1* calculate the localizer and glideslope noise according to AGARD R-632 (ref.[1]), using a fixed (user-specified) value for the approach speed. The scale length and standard deviation of the localizer and glideslope noise must be specified.

Inputsignal: none

Outputsignal: localizer noise or glideslope noise in [μ A]

See figures F-40 and F-41.

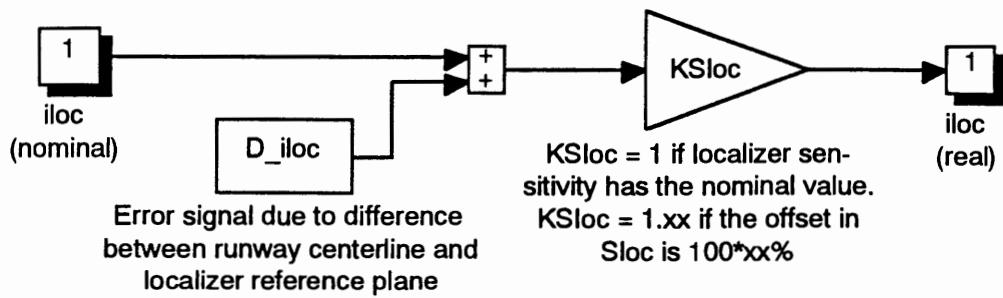


Figure F-38. Internal structure of the block *LOCERR*.
NAVLIB

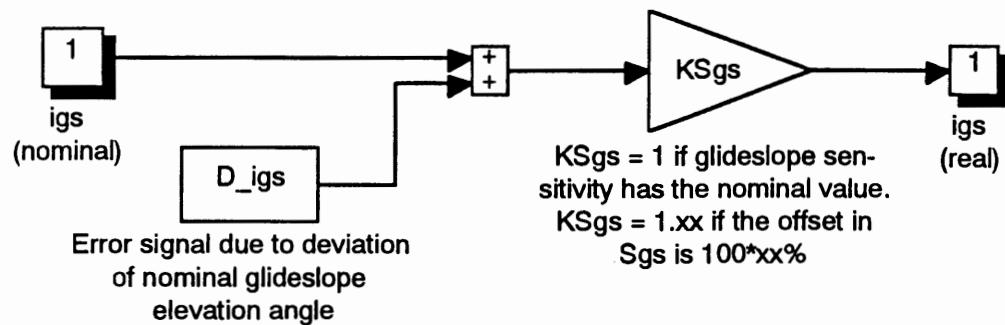


Figure F-39. Internal structure of the block *GSERR*.
NAVLIB

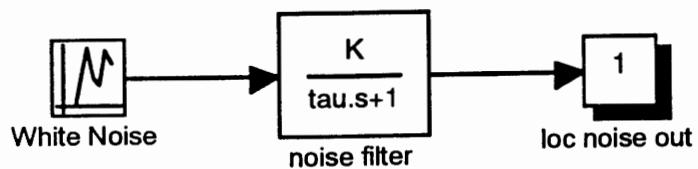


Figure F-40. Internal structure of the block *LOCNOISE1*.
NAVLIB

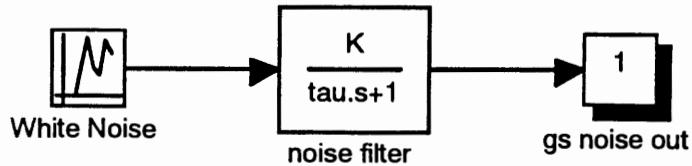
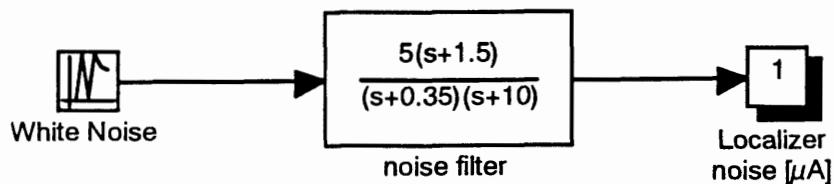
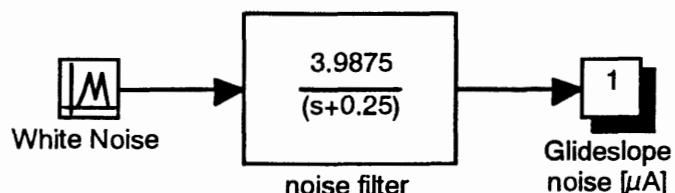


Figure F-41. Internal structure of the block *GSNOISE1*.
NAVLIB

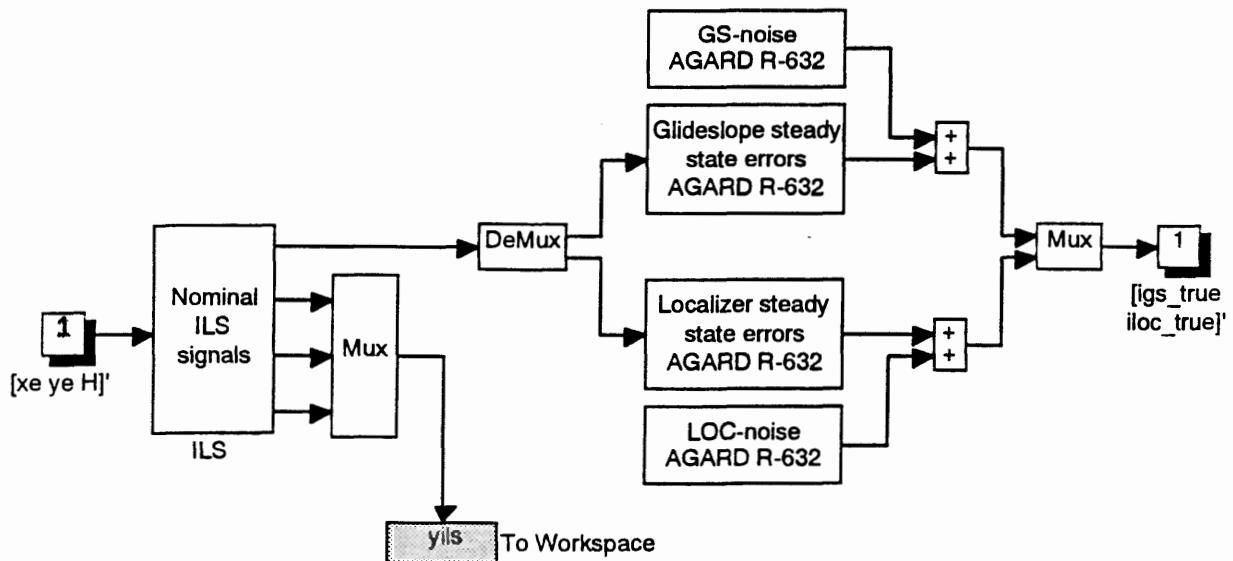




**Figure F-42. Internal structure of the block *LOCNOISE2*.
NAVLIB**



**Figure F-43. Internal structure of the block *GSNOISE2*.
NAVLIB**



**Figure F-44. Internal structure of the block *ILSXMPPL*.
NAVLIB (also contained as separate system in directory EXAMPLES)**

LOCNOISE2 and *GSNOISE2* create localizer and glideslope noise according to NASA CR-2022 (ref.[20]).

Inputsignal: none

Outputsignal: localizer noise or glideslope noise in [μA]

See figures F-42 and F-43.

On-line help for the localizer and glideslope noise blocks is available from the MATLAB command line (type *help locnoise* or *help gsnoise*, respectively).

- 4 - *ILSXMPL*. This block demonstrates the practical use of the ILS blocks. It uses the blocks *ILS*, *LOCERR*, *GSEERR*, *LOCNOISE1*, and *GSNOISE1*. Only the localizer and glideslope currents, including noise and steady-state errors, are returned as block-outputs, to which other systems can be connected. But all output signals from *ILSXMPL* are sent to the vector y_{ils} in the MATLAB workspace.

Inputs: \mathbf{u}_{ils}

Outputs: y_{ils} 1 (including errors)

See figure F-44.

- 5 - *VOR*. This block contains the equations, necessary to compute the nominal VOR signal. It also contains a To/From test, a Cone of silence flag, and a Range flag. The output from the To/From indicator equals 1 if the aircraft flies TO the VOR transmitter; it equals 0 if it flies FROM the VOR transmitter. The Cone of silence flag is set to 1 if the aircraft has entered the Cone of silence; the Range flag is set to 1 if the distance from the aircraft to the VOR station is too large (see section C.3.2).

Inputs: \mathbf{u}_{vor} 1 and u_{vor} 2

Outputs: y_{vor} 1, y_{vor} 2, y_{vor} 3, and y_{vor} 4

The user must specify the coordinates of the VOR-station, x_{VOR} and y_{VOR} , its altitude above sea level H_{VOR} , and the course datum CD .

Type *help vor* at the MATLAB command line for on-line help. The internal structure of *VOR* is shown in figure F-45.

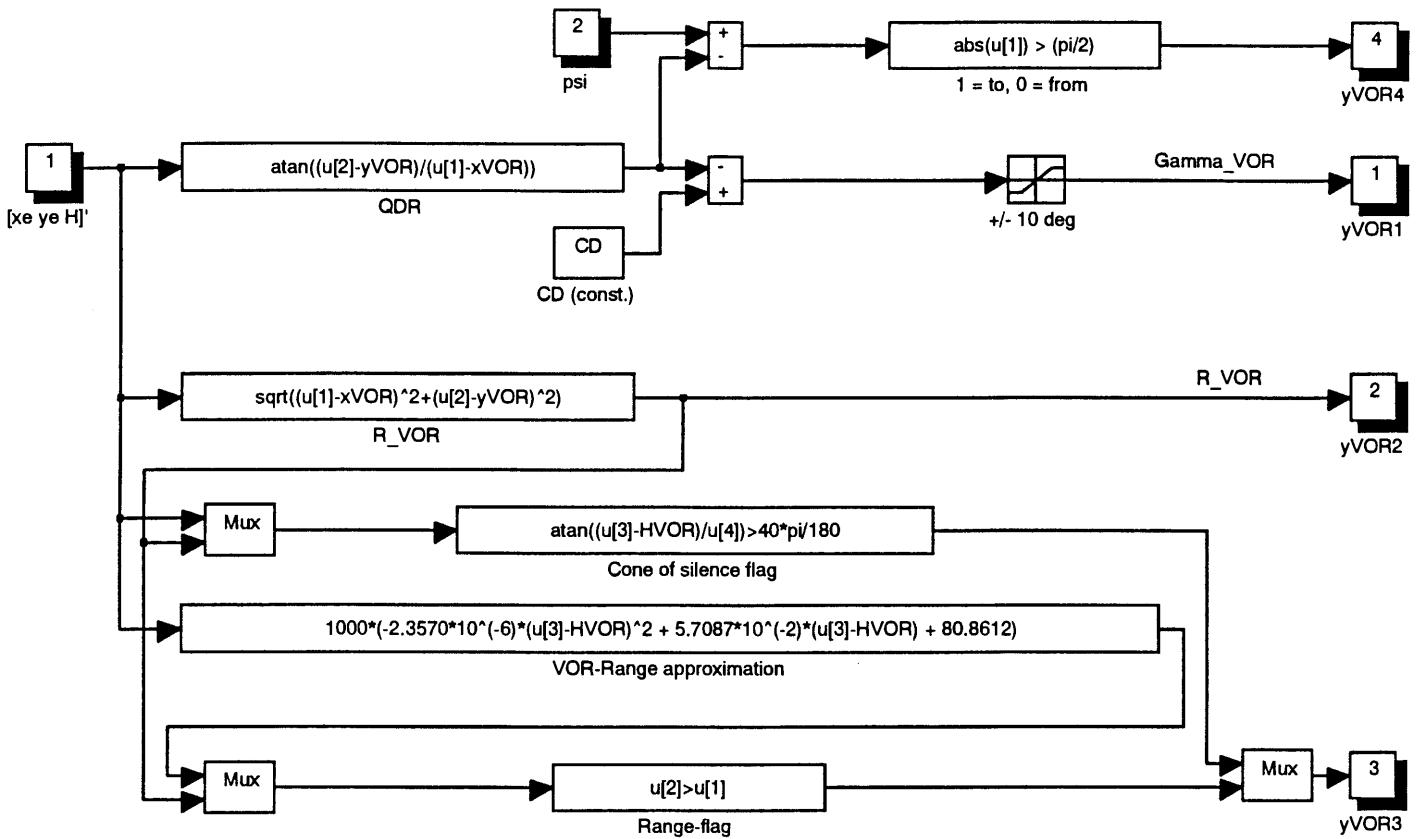
- 6 - *VORERR*. This block can be used to implement a steady-state error in the VOR signal. The user must specify the error percentage. Type *help vorerr* at the MATLAB command line for on-line help.

Inputsignal: Γ_{VOR} (nominal value)

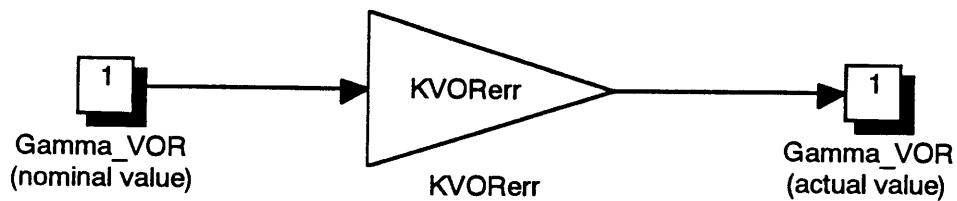
Outputsignal: Γ_{VOR} (nominal value + steady-state error)

See figure F-46 for the internal structure of this block.





**Figure F-45. Internal structure of the block VOR.
NAVLIB**



**Figure F-46. Internal structure of the block VORERR.
NAVLIB**

F.5 Conclusions.

This appendix contains all details of the nonlinear 'Beaver' model, the VOR and ILS models, and wind and turbulence models. The equations have all been implemented in block-diagrams, using the *Function* blocks of SIMULINK (see also appendix G). Flexibility has been obtained by grouping small sets of equations into 'basic' SIMULINK blocks. With the *Mask* utility, the internal structure of the blocks has been hidden from the user, and sometimes dialog boxes have been included (*black-box approach*). The user is encouraged to use the models for his/her own experiment. Don't be impressed too much by the detailed description in this appendix! Chapters 4 and 6 will often give sufficient information for using the current models if it is not necessary to change the models, but if more details are needed, this appendix gives all information.



$AM = [$	C_{X_0}	C_{Y_0}	C_{Z_0}	C_{l_0}	C_{m_0}	C_{n_0}	1
	C_{X_α}	0	C_{Z_α}	0	C_{m_α}	0	2
	$C_{X_{\alpha^2}}$	0	0	0	$C_{m_{\alpha^2}}$	0	3
	$C_{X_{\alpha^3}}$	0	$C_{Z_{\alpha^3}}$	0	0	0	4
	0	C_{Y_β}	0	C_{l_β}	0	C_{n_β}	5
	0	0	0	0	$C_{m_{\beta^2}}$	0	6
	0	0	0	0	0	$C_{n_{\beta^3}}$	7
	0	C_{Y_p}	0	C_{l_p}	0	C_{n_p}	8
	C_{X_q}	0	C_{Z_q}	0	C_{m_q}	C_{n_q}	9
	0	C_{Y_r}	0	C_{l_r}	C_{m_r}	C_{n_r}	10
	0	0	$C_{Z_{\delta_e}}$	0	$C_{m_{\delta_e}}$	0	11
	$C_{X_{\delta_f}}$	0	$C_{Z_{\delta_f}}$	0	$C_{m_{\delta_f}}$	0	12
	0	$C_{Y_{\delta_a}}$	0	$C_{l_{\delta_a}}$	0	$C_{n_{\delta_a}}$	13
	$C_{X_{\delta_r}}$	$C_{Y_{\delta_r}}$	0	$C_{l_{\delta_r}}$	0	$C_{n_{\delta_r}}$	14
	$C_{X_{\alpha\delta_f}}$	0	$C_{Z_{\alpha\delta_f}}$	0	0	0	15
	0	$C_{Y_{\delta_r\alpha}}$	0	0	0	0	16
	0	0	0	$C_{l_{\delta_a\alpha}}$	0	0	17
	0	0	$C_{Z_{\delta_e\beta^2}}$	0	0	0	18
	0	C_{Y_β}	0	0	0	0	19
	$]^T$						
	1	2	3	4	5	6	

Table F-1. Parameters of the aerodynamic model of the 'Beaver' in the format used by the SIMULINK simulation model.

$\text{EM} = [$	$C_{X_{\Delta p_t}}$	0	$C_{Z_{\Delta p_t}}$	0	$C_{m_{\Delta p_t}}$	0	$C_{n_{\Delta p_t^3}}$	$]^T$	1
	0	0	0	0	0	0	$C_{n_{\Delta p_t^3}}$		2
	$C_{X_{\alpha \Delta p_t^2}}$	0	0	0	0	0	0		3
	0	0	0	$C_{l_{\alpha^2 \Delta p_t}}$	0	0			4

1	2	3	4	5	6
---	---	---	---	---	---

Table F-2. Parameters of the engine model of the 'Beaver' in the format used by the SIMULINK simulation model.

$\text{GM1} = [$	\bar{c}	b	S	I_x	I_y	I_z	J_{xy}	J_{xz}	J_{yz}	m	$]^T$
	1	2	3	4	5	6	7	8	9	10	

$\text{GM2} = [$	P_l	P_m	P_n	P_{pp}	P_{pq}	P_{pr}	P_{qq}	P_{qr}	P_{rr}	$]^T$	1
	Q_l	Q_m	Q_n	Q_{pp}	Q_{pq}	Q_{pr}	Q_{qq}	Q_{qr}	Q_{rr}		2
	R_l	R_m	R_n	R_{pp}	R_{pq}	R_{pr}	R_{qq}	R_{qr}	R_{rr}		3

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

Table F-3. Aircraft geometry and mass distribution data in the format used by the SIMULINK simulation model.



\mathbf{C}_a	$= [C_{X_a} \ C_{Y_a} \ C_{Z_a} \ C_{I_a} \ C_{m_a} \ C_{n_a}]^T$
\mathbf{C}_t	$= [C_{X_t} \ C_{Y_t} \ C_{Z_t} \ C_{I_t} \ C_{m_t} \ C_{n_t}]^T$
\mathbf{F}_a	$= [X_a \ Y_a \ Z_a \ L_a \ M_a \ N_a]^T$
\mathbf{F}_{gr}	$= [X_{gr} \ Y_{gr} \ Z_{gr}]^T$
\mathbf{F}_t	$= [X_t \ Y_t \ Z_t \ L_t \ M_t \ N_t]^T$
\mathbf{F}_{tot}	$= [F_x \ F_y \ F_z]^T$
\mathbf{F}_w	$= [X_w \ Y_w \ Z_w]^T$
\mathbf{M}_{tot}	$= [L \ M \ N]^T$
\mathbf{u}_a	$= [\delta_e \ \delta_a \ \delta_r \ \delta_f]^T$ (inputs to the aerodynamic model)
\mathbf{u}_t	$= [n \ p_z]^T$ (inputs to the engine model)
\mathbf{u}_{wind}	$= [u_w \ v_w \ w_w \ \dot{u}_w \ \dot{v}_w \ \dot{w}_w]^T$ (wind and turbulence inputs)
\mathbf{x}	$= [V \ \alpha \ \beta \ p \ q \ r \ \psi \ \theta \ \phi \ x_e \ y_e \ z_e]^T$ (state vector)
$\dot{\mathbf{x}}$	$= [\dot{V} \ \dot{\alpha} \ \dot{\beta} \ \dot{p} \ \dot{q} \ \dot{r} \ \dot{\psi} \ \dot{\theta} \ \dot{\phi} \ \dot{x}_e \ \dot{y}_e \ \dot{H}]^T$
\mathbf{y}_{acc}	$= [A_x \ A_y \ A_z \ a_{x,k} \ a_{y,k} \ a_{z,k}]^T$
\mathbf{y}_{ad1}	$= [a \ M \ q_{dyn}]^T$
\mathbf{y}_{ad2}	$= [q_c \ V_e \ V_c]^T$
\mathbf{y}_{ad3}	$= [T_t \ R_e \ R_c]^T$
\mathbf{y}_{atm}	$= [p \ p_s \ T \ \mu \ g]^T$
\mathbf{y}_{bvel}	$= [u \ v \ w]^T$
\mathbf{y}_{dl}	$= [\frac{pb}{2V} \ \frac{qc}{V} \ \frac{rb}{2V}]^T$
\mathbf{y}_{eul}	$= [\dot{\psi} \ \dot{\theta} \ \dot{\phi}]^T$
\mathbf{y}_{fp}	$= [\gamma \ fpa \ \chi \ \Phi]^T$
\mathbf{y}_{hlp}	$= [\cos(\alpha) \ \sin(\alpha) \ \cos(\beta) \ \sin(\beta) \ \tan(\beta) \ \sin(\psi) \ \cos(\psi) \ \sin(\theta) \ \cos(\theta) \ \sin(\phi) \ \cos(\phi)]^T$
\mathbf{y}_{pow}	$= [dpt \ P]^T$
\mathbf{y}_{pqr}	$= [\dot{p} \ \dot{q} \ \dot{r}]^T$
\mathbf{y}_{uvw}	$= [\dot{u} \ \dot{v} \ \dot{w}]^T$
\mathbf{y}_{vab}	$= [\dot{V} \ \dot{\alpha} \ \dot{\beta}]^T$
\mathbf{y}_{xyh}	$= [\dot{x}_e \ \dot{y}_e \ \dot{H}]^T$

	<u>Inputs:</u>	<u>Outputs:</u>
Atmosph , element of: <i>Airdata Group</i>	\mathbf{x}	\mathbf{y}_{atm}
Airdata1 , element of: <i>Airdata Group</i>	$\mathbf{x}, \mathbf{y}_{atm}$	\mathbf{y}_{ad1}
Airdata2 , element of: <i>Airdata Group</i>	$\mathbf{y}_{atm}, \mathbf{y}_{ad1}$	\mathbf{y}_{ad2}
Airdata3 , element of: <i>Airdata Group</i>	$\mathbf{x}, \mathbf{y}_{atm}, \mathbf{y}_{ad1}$	\mathbf{y}_{ad3}
Dimless , element of: <i>Aerodynamics Group</i>	\mathbf{x}	\mathbf{y}_{dl}
Aeromod (Beaver) , element of: <i>Aerodynamics Group</i>	$\mathbf{x}, \mathbf{u}_a, \mathbf{y}_{dl}$	\mathbf{C}_a
FMdims , element of: <i>Aerodynamics Group/Engine Group</i>	$\mathbf{y}_{ad1}, \mathbf{C}_a$ or \mathbf{C}_t	\mathbf{F}_a or \mathbf{F}_t
Power (Beaver) , element of: <i>Engine Group</i>	$\mathbf{x}, \mathbf{u}_t, \mathbf{y}_{atm}$	\mathbf{y}_{pow}
Engmod (Beaver) , element of: <i>Engine Group</i>	$\mathbf{x}, \mathbf{y}_{pow}$	\mathbf{C}_t
Gravity	$\mathbf{x}, \mathbf{y}_{atm}$	\mathbf{F}_{gr}
Fwind	\mathbf{x}, \mathbf{u}_w	\mathbf{F}_w
FMSort	$\mathbf{F}_a, \mathbf{F}_t, \mathbf{F}_{gr}, \mathbf{F}_w$	$\mathbf{F}_{tot}, \mathbf{M}_{tot}$
Hlpcn , element of: <i>Equations of motion\State derivatives</i>	\mathbf{x}	\mathbf{y}_{hlp}
Vabd , element of: <i>Equations of motion\State derivatives</i>	$[\mathbf{x}^T \ \mathbf{F}_{tot}^T \ \mathbf{M}_{tot}^T \ \mathbf{y}_{hlp}^T]^T$	\mathbf{y}_{vab}
pqrdot , element of: <i>Equations of motion\State derivatives</i>	$[\mathbf{x}^T \ \mathbf{F}_{tot}^T \ \mathbf{M}_{tot}^T \ \mathbf{y}_{hlp}^T]^T$	\mathbf{y}_{pqr}
xyHdot , element of: <i>Equations of motion\State derivatives</i>	$[\mathbf{x}^T \ \mathbf{F}_{tot}^T \ \mathbf{M}_{tot}^T \ \mathbf{y}_{hlp}^T]^T$	\mathbf{y}_{eul}
uvw , element of: <i>Equations of motion\State derivatives</i>	$[\mathbf{x}^T \ \mathbf{F}_{tot}^T \ \mathbf{M}_{tot}^T \ \mathbf{y}_{hlp}^T]^T$	\mathbf{y}_{bvel}
xdotcorr , element of: <i>Equations of motion</i>	$\mathbf{x}, \dot{\mathbf{x}}$ (uncorrected), \mathbf{y}_{atm}	\mathbf{x} (corrected)
Flpath	$\mathbf{x}, \dot{\mathbf{x}}$	\mathbf{y}_{fp}
Accel	$\mathbf{F}_{tot}, \mathbf{F}_{gr}$	\mathbf{y}_{acc}
uvwdot	$\mathbf{x}, \dot{\mathbf{x}}$	\mathbf{y}_{uvw}

Table F-4. List of basic blocks within the 'Beaver' model, including their inputs and outputs.

Sub-vector	Definition of the matrix <i>Out</i> (the numbers correspond with the columns which contain the time-trajectories of the specified variables)											
x	V 1	α 2	β 3	p 4	q 5	r 6	ψ 7	θ 8	φ 9	x_e 10	y_e 11	H 12
\dot{x}	\dot{V} 13	$\dot{\alpha}$ 14	$\dot{\beta}$ 15	\dot{p} 16	\dot{q} 17	\dot{r} 18	$\dot{\psi}$ 19	$\dot{\theta}$ 20	$\dot{\varphi}$ 21	\dot{x}_e 22	\dot{y}_e 23	\dot{H} 24
y_{bvel}	u 25	v 26	w 27									
y_{uvw}	\dot{u} 28	\dot{v} 29	\dot{w} 30									
y_{dl}	$\frac{pb}{2V}$ 31	$\frac{qc}{V}$ 32	$\frac{rb}{2V}$ 33									
y_{fp}	γ 34	fpa 35	χ 36	Φ 37								
y_{pow}	dpt 38	P 39										
y_{acc}	A_x 40	A_y 41	A_z 42	$a_{x,k}$ 43	$a_{y,k}$ 44	$a_{z,k}$ 45						
C_a	C_{X_a} 46	C_{Y_a} 47	C_{Z_a} 48	C_{l_a} 49	C_{m_a} 50	C_{n_a} 51						
C_t	C_{X_t} 52	C_{Y_t} 53	C_{Z_t} 54	C_{l_t} 55	C_{m_t} 56	C_{n_t} 57						
F_a	X_a 58	Y_a 59	Z_a 60	L_a 61	M_a 62	N_a 63						
F_t	X_t 64	Y_t 65	Z_t 66	L_t 67	M_t 68	N_t 69						
F_{gr}	X_{gr} 70	Y_{gr} 71	Z_{gr} 72									
F_w	X_w 73	Y_w 74	Z_w 75									
y_{atm}	ρ 76	p_s 77	T 78	μ 79	g 80							
y_{ad1}	a 81	M 82	q_{dyn} 83									
y_{ad2}	q_c 84	V_e 85	V_c 86									
y_{ad3}	T_t 87	R_e 88	R_c 89									

Table F-5. List of all outputs from the system *BEAVER* which are sent to the MATLAB workspace (continued on next page).



Definition of S-function output vector (= outputs which have been connected to an *Outport* block in the first level of *BEAVER*, see figure F-1)

V	α	β	p	q	r	ψ	θ	φ	x_e	y_e	H	\dot{H}	$\frac{pb}{2V}$	$\frac{qc}{V}$	$\frac{rb}{2V}$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Definition of S-function input vector (= inputs which have been connected to an *Inport* block in the first level of *BEAVER*, see figure F-1)

δ_e	δ_a	δ_r	δ_f	p_z	n	u_w	v_w	w_w	\dot{u}_w	\dot{v}_w	\dot{w}_w
1	2	3	4	5	6	7	8	9	10	11	12

Sub-vector	Definition of the matrix <i>In</i> (the numbers correspond with the columns which contain the time-trajectory of the specified variables)										
\mathbf{u}_a	$\delta_e \quad \delta_a \quad \delta_r \quad \delta_f$ 1 2 3 4										
\mathbf{u}_t	$n \quad p_z$ 5 6										
\mathbf{u}_{wind}	$u_w \quad v_w \quad w_w \quad \dot{u}_w \quad \dot{v}_w \quad \dot{w}_w$ 7 8 9 10 11 12										

Table F-5 (continued). List of S-function inputs and outputs and the inputs of the system *BEAVER* which are sent to the MATLAB workspace.



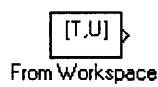
Appendix G. Standard SIMULINK blocks.

G.1 Introduction.

This appendix explains the functions of the most important SIMULINK blocks, which are used in the graphical block-diagrams, presented in this report. However, this list is not complete, since this appendix is not intended to be a full replacement for the SIMULINK manual (ref.[4]). Similarly to the main SIMULINK blocklibrary, we will distinguish six block categories: *Sources*, *Sinks*, *Discrete*, *Linear*, *Nonlinear*, and *Connections*. In section G.8, some basics about the SIMULINK 'group' and 'mask' functions will be described. See chapter 3 for more general information about SIMULINK.

G.2 Blocks from the Sources-library.

The *Sources* library contains all necessary input blocks. It is possible to generate standard input signals, such as block-shaped inputs, white noise, sine inputs, etc., or to load input data from the MATLAB workspace or from files.



The *From Workspace* block can be used to send arbitrary input-signals, defined in the MATLAB workspace, to the SIMULINK system. The inputvariable must be a matrix with at least two rows. The first row must contain monotonically increasing time points; the other row(s) define(s) the input sequence(s). If an output value is needed at a time between two values, the output is linearly interpolated between the two values bracketing the required time.



The *From File* block functions the same as the *From Workspace* block, but this time the data is read from a MAT-file.



The *Constant* block injects a constant value into the system. This may be a vector!



The *Step Function* block provides a step between two arbitrary values at a specified time. The user must specify the initial value, final value, and step time.



The *White Noise* block generates normally distributed random noise. The user must specify an initial seed value, which makes the sequence of random numbers reproducible.



The *Sine Wave* block inserts a sine wave into the system. The user must specify the amplitude, frequency, and phase of the sinusoid.



The *Clock* block sends the time to the system as an input signal, which can be used to implement time-dependent functions, or to create a time-axis for plotting purposes, if it is connected to a *To Workspace* or *To File* block from the *Sinks* library. Opening the block provides a window that continuously displays the system time as simulation progresses.



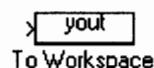
G.3 Blocks from the Sinks-library.

The *Sinks* library contains blocks which send signals from the SIMULINK system to the MATLAB workspace, to a file, or to a 'Scope'.



Scope

The *Scope* block is used to visualize signals during a simulation. It shows signals like an oscilloscope. The horizontal and vertical range can be adjusted by the user.



To Workspace

The *To Workspace* block sends signals to a user-defined variable in the MATLAB workspace. If a vector-line is connected to this block, a matrix with as many columns as vector elements will be built in the MATLAB workspace. It is possible to create a time-axis in the workspace by connecting a *Clock* element to a *To Workspace* block.

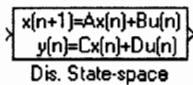


To File

The *To File* block sends the data to a matrix in a user-defined MAT-file. The first column of the matrix contains the time values used during simulation.

G.4 Blocks from the Discrete-library.

The *Discrete* library contains linear discrete-time elements, such as discrete state-space systems, transfer functions, and zero-pole blocks. The continuous equivalent of this library is the *Linear* library, see section E.5. If these blocks are used within a continuous system, SIMULINK will use a Zero Order Hold filter to determine output values between two sample intervals. Systems with a mixture of continuous and discrete dynamics are called 'hybrid' systems. In SIMULINK, it is possible to use different discrete blocks having different sampling rates in one system ('multirate systems').



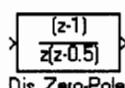
Dis. State-space

The *Discrete State-Space* block contains a linear state-space description of a discrete system. The matrices A, B, C, and D, the initial state vector \mathbf{x}_0 , and the sample time T_s can be defined by the user.



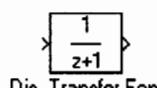
Unit Delay

The *Unit Delay* block delays the input by one sample interval T_s .



Dis. Zero-Pole

The *Discrete Zero-Pole* block implements discrete-time systems, characterized by their gain, zeros, and poles in the z-domain, and the sample time T_s , which can all be specified by the user.

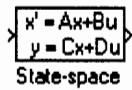


Dis. Transfer Fcn

The *Discrete Transfer Function* block defines a discrete-time transfer function in terms of its coefficients of the numerator and denominator polynomials in the z-domain and the sample time T_s , which can all be specified by the user.

G.5 Blocks from the Linear-library.

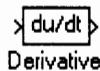
The *Linear* library contains a number of linear, continuous-time system dynamics blocks, such as transfer functions and linear state-space systems. This library is the continuous equivalent of the *Discrete* library, see section E.4.



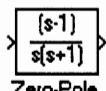
The *State-space* block can be used to implement a linear system in State-space format within a block-diagram. The matrices A, B, C, and D, and the initial state vector \mathbf{x}_0 must be defined by the user.



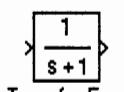
The *Integrator* block can be used to integrate a signal. It is possible to specify the initial value of the signal. If the initial value of the signal is specified by a column vector of length n , the input may also be a vector with length n , and all elements of the inputvector are integrated and combined in the outputvector of length n .



The *Derivative* block approximates the time derivative of a signal. It subdivides the difference between the value at time t and time $t-h$ (the previous integration step) by the step size h . Accuracy depends on the size of the timesteps taken in the simulation. Smoother and more accurate outputs are obtained if h is smaller.



The *Zero-Pole* block can be used to implement a continuous transfer function, characterized by its gain, poles, and zeros in the s -domain.



The *Transfer-Function* block also implements a continuous transfer function, but this time, the user must specify the coefficients of the numerator and denominator polynomials in the s -domain.



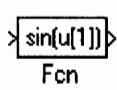
The *Gain* block is used to multiply a (scalar or vector) inputsignal by a constant value, which must be defined by the user. If the input is a vector, gain factors for each element can be defined by specifying a row vector with the same length as the input vector instead of the scalar gain. Each m^{th} element of the gain vector is multiplied with the m^{th} element of the inputvector.



The *Sum* block is used to add or subtract a number of signals. This block can be used for scalars and vectors, provided that the number of elements of all input signals are the same. The number of inputs is defined by the number of plusses or minusses in the block. For instance, the sequence + - + adds the first and fourth inputs to eachother and subtracts the second and third inputs.

G.6 Blocks from the Nonlinear-library.

The *Nonlinear* library contains all kinds of nonlinear signal processing blocks, such as time-delays, limiters, nonlinear functions and calls of other (nonlinear or linear) S-functions.



The *Function* block applies the specified function to its input. The function must take the form of a legal expression in C. The (scalar) output may be a function of multiple elements of the inputvector, where the elements of the inputvector are denoted by $u[i]$. For instance: the function definition $\sin(u[1])/\tan(u[2])$ divides the sinus



of the first element of the input vector by the tangent of the second element of the input vector. See ref.[4] for more details!



Product

The *Product* block can be used to multiply two inputs signals with each other. If the inputs are vectors, an element-by-element multiplication is performed.



Saturation

The *Saturation* block limits the signal passing through it. The upper and lower bounds must be specified by the user. The block accepts scalar input and output only.



Rate Limiter

The *Rate Limiter* limits the first derivative of the signal passing through it. The rising falling slew rates must be specified by the user. The *Rate Limiter* supports scalar input and output only.



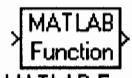
Transport Delay

The *Transport Delay* holds the input during a certain user-specified time interval.



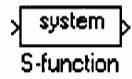
Switch

The *Switch* block switches between input 1 and input 3, depending on the value of input 2. If $u_2 \geq 0$, the output equals u_1 . Else, the output equals u_3 . Input 2 is the switch criterion.



MATLAB Fcn

The block *MATLAB Function* applies the specified function to the input(-vector) in the same way as the *Function* block, only this time it calls any valid M or MEX-file. Functions which are too complicated to implement with the *Function* block can now be programmed entirely in MATLAB, Fortran, or C, and all functions from the MATLAB toolboxes can thus be used within SIMULINK systems.



S-function

The *S-function* block is used to connect another SIMULINK system to the system by means of a separate block. S-functions are either SIMULINK block diagrams, or M or MEX-files which comply to certain I/O requirements. S-functions are discussed in more detail in chapter 3. See also ref.[4].

G.7 Blocks from the Connections-library.

The blocks from the *Connections* library combine multiple signals into one vector line, separate vectors in a number of smaller vectors, and it provide links between different (sub)systems.



Mux

The *Mux* block (multiplexer) groups several scalar or vector lines together into a single vector line. The number of elements or a vector with the number of elements of all input lines must be specified by the user. For instance, if the number of elements is specified as: [3 4 5 1 1], the *Mux* block will get five input ports, suited for vectors of lengths 3, 4, 5, 1, and 1, respectively.



Demux

The *DeMux* block (demultiplexer) separates a vector line into a number of smaller vector or scalar lines. The user must either specify the number of outputs, or the number of elements of each output. For instance, if the number of outputs is specified as: [4 1 3 11], the *DeMux* block will get four output ports, suited for vectors of lengths 4, 1, 3, and 11 respectively.



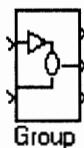
The *Import* block provides a link between the inputs of a subsystem block and the subsystem itself, and it can be used to represent external inputs for analysis tools, such as linearization or simulation from the MATLAB command line. Imports within a subsystem must be numbered sequentially, starting with 1.



The *Outport* block provides a link between the outputs of a subsystem block and the subsystem itself, and it can be used to represent outputs for analysis tools, such as linearization or simulation from the MATLAB command line. The outports within a subsystem must be numbered sequentially, starting with 1.

G.8 The *Group* and *Mask* functions.

Two useful options of SIMULINK are the *Group* and *Mask* functions, for building and manipulating subsystems. Since these options have been used within the systems from this report, a brief description of these SIMULINK functions is included here.



It is possible to convert any set of blocks and lines to a subsystem block with the *Group* command. The number of the input ports drawn on the subsystem block corresponds to the number of *inports* contained in the subsystem. The top input connects to *Import 1*, the second input connects to *Import 2*, and so on. The output ports drawn on the subsystem correspond in the same way to the number of *Outports* contained in the subsystem. Here we have two *Imports* and three *Outputs*.

The characteristics of any block (including subsystem and S-function blocks) can be changed by using the *Mask* function. Masking a block results in a block with a *black-box* character, which hides its internal structure from the user. It is possible to create an appropriate icon, define (local) variables within the block, and create a dialog box in which the user can set block-parameters. See ref.[4] for more details about this very useful feature. It is also recommended to examine examples of masked files, included in the SIMULINK software package, because those examples contain some useful tricks which aren't documented in the manual!

G.9 Conclusions.

In this appendix, the most important SIMULINK blocks have been reviewed briefly. With the information from this appendix, it should be possible to read all block-diagrams in this report. However, the reader should consult the SIMULINK User's Guide [4] for more detailed information. It is also very useful to examine examples of SIMULINK, which are included in the SIMULINK package. See chapter 3 for some general information about SIMULINK and S-functions.



Appendix H. Installation procedures.

This report should be accompanied 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 mode-controller;
- 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*.



