Series 03 Control and Simulation 03

DASMAT-Delft University Aircraft Simulation Model and Analysis Tool

A Matlab/Simulink Environment for Flight Dynamics and Control Analysis

C.A.A.M. van der Linden

Delft University Press

DASMAT-Delft University Aircraft Simulation Model and Analysis Tool

A Matlab/Simulink Environment for Flight Dynamics and Control Analysis

Series 03: Control and Simulation 03

DASMAT-Delft University Aircraft Simulation Model and Analysis Tool

A Matlab/Simulink Environment for Flight Dynamics and Control Analysis

C.A.A.M. van der Linden

Delft University Press / 1998

Published and distributed by:

Delft University Press
Mekelweg 4
2628 CD Delft
The Netherlands
Telephone +31 (0)15 278 32 54
Fax +31 (0)15 278 16 61
e-mail: DUP@DUP.TUDelft.NL

by order of:

Faculty of Aerospace Engineering
Delft University of Technology
Kluyverweg 1
P.O. Box 5058
2600 GB Delft
The Netherlands
Telephone +31 (0)15 278 14 55
Fax +31 (0)15 278 18 22
e-mail: Secretariaat@LR.TUDelft.NL
website: http://www.lr.tudelft.nl

Cover: Aerospace Design Studio, 66.5 x 45.5 cm, by:
Fer Hakkaart, Dullenbakkersteeg 3, 2312 HP Leiden, The Netherlands
Tel. +31 (0)71 512 67 25

90-407-1582-3

Printed in The Netherlands

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This manual documents the implementation of the *DASMAT* software package in MATLAB/SIMULINK. The acronym *DASMAT* stands for **D**elft University **A**ircraft **S**imulation **M**odel and **A**nalysis **T**ool. The package is designed around a generic nonlinear aircraft model which is available as a SIMULINK S-function. In addition to this model, *DASMAT* provides the following analysis tools, available as MATLAB M-files:

- simulate aircraft model
- simulate engine model
- trim aircraft model
- linearize aircraft model
- fit aerodynamic model
- plot time-responses of aircraft simulation
- show animation of aircraft simulation

The *DASMAT* package offers the user a powerful and flexible Computer Assisted Design (CAD) environment for use in various disciplines of flight mechanics research, i.e. design of control systems, simulation purposes, design of flight tests. The base of the package is a generic nonlinear aircraft model with well-defined and generalized interfaces to aerodynamic, propulsion and engine models, as well as to models of external conditions as atmospheric wind and turbulence. The generic aircraft model can thus be made applicable to any aircraft and condition by including user-supplied models or models from a library for that aircraft or condition. Furthermore, the model architecture gives the user the ability to define control inputs and select observation outputs without destroying the model integrity, and thus to adapt the model to the analysis problem.

DASMAT contains all required tools for both off-line and on-line system analysis. There are trimming and linearization routines for the aircraft model as well as for the aerodynamic model. The aircraft and engine models may be analyzed through nonlinear simulations where the dynamical behaviour may be visualized as a real-time 3-dimensional animation or as an activity of any external or internal signal. Both the model and model inputs may be manipulated on-line during a simulation.

Aside from the tools available from *DASMAT*, the user has the possibility to create self-defined applications and evaluation functions using the results from any *DASMAT* tool. The results are directly accessible to the user and MATLAB/SIMULINK offers a great variety of standard functions and an easy-to-use program facility.
The *DASMAT* model structure and architecture allows a good starting point for further development. The modular structure of the aircraft model has the ability for easy implementation of more complex sub-systems, i.e. an airframe flexibility model, an equation of motion model using quaternions or an even more comprehensive observation model. The generic structure of the various models decouple the *DASMAT* tools from any specific aircraft. Although *DASMAT* was developed using the available model data of the Cessna Citation 500, it will in future primarily be used for applications regarding the Cessna Citation II. The model architecture allows an easy integration of the aircraft model with control systems for different applications. The large set of observation outputs is likely sufficient for a broad class of control design problems. The stand-alone usability of the aerodynamic model is essential for parameter identification problems.

The *DASMAT* package operates in the MATLAB/SIMULINK environment from The MathWorks Inc., [9, 10]. MATLAB is a technical computing environment for high-performance numeric computation and visualisation. It is an interactive system which integrates numerical analysis, matrix computation, signal processing and graphics in an easy-to-use environment. Typical uses include general purpose numeric computation, algorithm prototyping and special purpose problem solving with matrix formulations that arise in disciplines such as automatic control and digital signal processing (time-series analysis).
MATLAB also features a family of application-specific solutions called toolboxes. These are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment in order to solve particular classes of problems. Relevant areas in which toolboxes are available from The MathWorks Inc. include signal processing, control systems design, dynamic systems simulation and system identification. The *DASMAT* package is also designed as a toolbox. Generated results from *DASMAT* are thus suitable for further processing in many areas.
SIMULINK is an extension to MATLAB for specific applications to dynamic systems. It can be used for both model definition and model analysis. For model definition, SIMULINK models can be created and edited in graphical block diagram windows which give a direct insight in the model structure. Model analysis can either be performed by choosing options from SIMULINK menus or by commands in the MATLAB command window for instance through M-files. A number of analysis options are available in this way via the analysis tools of *DASMAT* in the above list.

The history of *DASMAT* started with the need for a standard Flight CAD package for control and design purposes within the Disciplinary Group for Stability and Control at the Faculty of Aerospace Engineering of Delft University of Technology. This need arose with the purchase of the Cessna Citation II aircraft which is to be configured as a fly-by-wire testbed using new flight control methodologies as H_∞/μ-synthesis. Before implementing these control techniques in real flight, an extensive development and testing facility on laboratory computers should be provided first.
A major initial contribution towards a Flight CAD environment was done by Rauw [14], using model data of the former laboratory owned DHC-2 'Beaver' aircraft. This led to the basic

framework of the aircraft model structure in the MATLAB/SIMULINK environment. This framework was later on further decoupled from a specific aircraft, so as to make it more generic, and a standard input/output format between the various sub-models was established. The set of observation outputs has also been extensively elaborated. Aside from the applied models, the analysis tools in MATLAB were further developed and automated. With the further development of MATLAB/SIMULINK itself, the package was extended with graphical user-interfaces for on-line control and visualisation features. Execution speed was improved by introducing compiled C-code for standard sub-models. This finally evolved to the present day *DASMAT* package.
However, the package may keep on evolving, like most software, by generating a larger library of sub-system models and application models. A next step is integrating *DASMAT* with a simulation and identification program for flight test data analysis in the MATLAB environment. As for experimental applications, the integration with DSPACE is considered.

The choice for MATLAB/SIMULINK as the computational environment for the *DASMAT* package is based on the following considerations. The combination of high-performance numeric computation with technical application-specific capabilities through a wide range of available tools makes MATLAB/SIMULINK a good environment for analysing complex systems as aircraft models. The analysis is enhanced by its visualisation capabilities. Its easy-to-use program facilities provide a great flexibility and extensibility, allowing the user to contribute and adapt applications to user-specific needs. For actual applications, the integration with DSPACE provides a rapid implementation of the MATLAB/SIMULINK software with hardware systems for real-time simulation, system analysis and controller design.

This manual discusses the structure of the models and operational aspects of the *DASMAT* package. The user is assumed to be familiar with MATLAB/SIMULINK.
The manual uses the following typographical conventions:

- `Monospace` : Commands, function names and screen displays (quotes mean substitution of an appropriate keyword); for example `dasmat`.
- **Bold** : Key names, SIMULINK window names, menu names and items that may be selected from menus; for example, the **Enter** key.
- *Italics* : MATLAB toolbox names, SIMULINK block names, names of sections in this manual, mathematical notation and for introduction of new terms; for example Chapter 1 *Introduction.*

The organization of this manual is as follows. Chapter two gives an overview of the *DASMAT* package with a discussion of the structure of the models and variables present. In chapter three, the operational aspects for the simulation and analysis tools is discussed. Chapter four focuses on the application of the *DASMAT* package for control design purposes. The appendices include a complete list of files from the *DASMAT* package, the formats of state, input and output signals of the aircraft model, a complete list of significant variables with their accessibility in the MATLAB workspace and the outlines of the SIMULINK models.

Chapter 2

General Overview

2.1 Installation

Before installing the *DASMAT* package, the user should create a directory in which the *DASMAT* package is to be copied, e.g. `matlab/contrib/dasmat`. It is recommended to include this directory in the `matlabpath`. The *DASMAT* package is then treated as a toolbox and the user may run the package from an arbitrary directory.
It is further recommended to create a work-directory for each specific aircraft, e.g. `citation` for the Cessna Citation, and an allied data-directory, e.g. `citation/data`. The work-directory should be used as operational directory. It contains the files with aircraft specific sub-models and model data. Moreover, if the user wishes to modify existing operating shells by redefining input blocks or making a selection of observation variables, these new SIMULINK models may then be stored in the work-directory and will not affect the original set of *DASMAT* files. The data-directory is used for locating the data-files with the results generated by the *DASMAT* tools. When running the package, the user is asked to specify this data-directory.

2.2 *DASMAT* files

The files of the *DASMAT* package may be distinguished in the following categories:

- SIMULINK models (S-functions)
- MATLAB routines (M-functions/M-scripts)
- data-files (MAT-files)

A list of the files is presented in Appendix A *Files used by DASMAT*. The files containing the generic SIMULINK models, MATLAB routines and non-aircraft dependent data-files are maintained in the *DASMAT* directory, e.g. `matlab/contrib/dasmat`. User-defined SIMULINK models, i.e. aircraft specific sub-models or modified operating shells, together with the associated model data-files are kept in the operational work-directory. All data-files with user-generated data from executing the *DASMAT* tools are saved in the data-directory designated by the variable `datadir`.

All files from the *DASMAT* package may be listed via the commands `help dasmat` or `what dasmat`. The generated data-files may be listed via the command `dir(datadir)`.

The SIMULINK models comprise models for the aircraft, aerodynamics, propulsion, engine, wind and turbulence models. There are moreover SIMULINK models which serve as operating shells from which a simulation is executed. The SIMULINK models appear as S-functions and may have three types of representations, i.e. a graphical block diagram, a text M-file or a compiled MEX-file. The operating shells have the block diagram representation and may thus be displayed on the screen in a SIMULINK window. The MEX-files are platform dependent and should therefore be compiled from the supplied C-code files.
The MATLAB routines comprise the routines for the execution of the simulation and the analysis tools. There are script M-files and function M-files. The script M-files contain the sequence of statements for the analysis tools. The function M-files are subroutines which perform specific tasks within the script M-files. Each M-file has a heading which includes a short description of the function, the syntax of the function call, a reference to the routine it was called from and the required subroutines and/or SIMULINK models. This heading may be displayed using the command `help 'filename'`.

The data-files comprise data which is required for running *DASMAT*, which is specific to the analyzed aircraft and data which is generated from running an analysis tool. All data-files have the structure of MAT-files, although they do not have all the default `.mat` file extension. The file extension is used for distinguishing the data-files. In this way the variables in the various data-files are automatically managed by *DASMAT*.

2.3 Simulink models

2.3.1 Architecture of Simulink models

The architecture of the SIMULINK models as presented in Figure 2.1 provides flexibility and creates a generic structure. It also allows adjustment of simulation and analysis according to the wishes of the user while preserving the integrity of the aircraft model itself.
The various diagrams in Figure 2.1 indicate SIMULINK S-functions. The texts in monospace designate the variable names or file names through which the models are accessible by the *DASMAT* package. If a diagram is concentric within another diagram, then the SIMULINK model of the inner diagram is imbedded in the SIMULINK model of the outer diagram via an *S-function* block.

The heart of the model architecture is formed by generic models. Generic models exist for the aircraft system, engine system and the aerodynamics system, see Figure 2.1. A generic model represents a framework which is applicable to any system and condition. It is configured to a specific system and condition by substituting modules which model the object specific characteristics. In SIMULINK, this substitution is performed by implementing the modules as *S-function* blocks and modeling the object specific characteristics as independent S-functions. These S-functions may be taken from a library of models or they may be created by the user. Details will be discussed below in this paragraph.
The generic model structure provides both uniqueness, standardization, flexibility and as a result better applicability. All these aspects are closely related. The uniqueness manifests by

the shared use of general applicable equations and models, as for example the equations of motion and the atmospheric model. Besides, these shared functionalities constitute a standard. Standardization is further introduced for the transfer of data. A standard input/output format is required to fit the object specific sub-models in the generic model. Using the generic model as interface with the outside, as for example with a controller, a standard format data structure is provided. The flexibility evolves from the standardization. The object specific sub-models may be used in different generic models and are easily exchangeable. For example, an aerodynamic model may be analyzed separately via the aerodynamic generic model or it may be included in the aircraft generic model, either as a set of polynomials or as a collection of lookup tables.
The aircraft generic model determines a kinematic system model using a set of 12 nonlinear equations of motion for a rigid aircraft over a flat non-rotating earth. Furthermore it includes the equations for defining the observation outputs. This SIMULINK model needs to be configured for a specific aircraft by defining the aerodynamic model and the propulsion model. The model is further extended by adding a wind and turbulence model.
The generic models for the engines and the aerodynamics only extend the user-specified models with the atmospheric model and thus facilitate the use of a common atmosphere. Furthermore, they provide a standardized input format which is compatible with the data structure used at the aircraft generic model.

The outer diagrams in Figure 2.1 represent operating shells. There are operating shells for the aircraft and the engine model. These SIMULINK models provide the control inputs to the aircraft or engine system and give a flexible access to the observation outputs from the aircraft or engine system. The user performs the simulations from these operating shells and has the option to adjust them (after copying the default files to the work-directory). For example the user can redefine the generation of control inputs and select the format and/or visualization of observation outputs.
There are tree types of operating shells depending on the type of simulation, see Section 2.3.11 *Operating shells for aircraft simulation*: the control inputs are either defined by *Inport* blocks (for open-loop simulation from the command window), by other than *Inport* blocks (for open-loop simulation in a SIMULINK window), or they are generated via a controller (for closed-loop simulation in a SIMULINK window).

The innermost diagrams in Figure 2.1 represent the aircraft specific sub-models, i.e. the aerodynamic and propulsion model, and the wind and turbulence model. These SIMULINK models should be supplied by the user or selected from the *DASMAT* package. They have a standard format for data transfer to make them applicable in different generic models.
The aerodynamic model calculates the aerodynamic force and moment coefficients and the force and moment coefficients due to gust. The coefficients may either be defined in the air-path, the stability or the body axes. The moment coefficients are with respect to a reference location defined in the datum reference frame. A template SIMULINK model for the aerodynamic model is included in the *DASMAT* directory.
The propulsion model calculates the forces and moments generated by all engines and passes a number of general engine parameters. The forces and moments are defined in the body axes and are derived from the thrust, position and orientation of individual engines. Each engine is again a SIMULINK model. If no engine model is to be included then the thrust is directly obtained from the thrust control inputs in the operating shell. The *DASMAT*

directory includes a SIMULINK model for a two-engine propulsion model.
The wind model specifies the wind velocity along the earth axes. The wind may be defined as a function of the aircraft position so that the aircraft may fly through a varying wind field or may experience wind shear. The SIMULINK model for a zero wind condition is available in the *DASMAT* directory.
The turbulence model calculates the gust velocities and their time derivatives along the body axes. These quantities become zero if no turbulence is considered. Otherwise they may be calculated from white noise using Dryden spectra. The SIMULINK models are available in the *DASMAT* directory.

The various SIMULINK models are designated by variables in the MATLAB workspace. This gives a generic code in the M-files where the models are indirectly referred via the contents of standard named variables. The user may evaluate the model configuration from either the model in a SIMULINK window or from the variables in the workspace.
The variables which name the aircraft specific sub-models are assigned during the initialization phase of *DASMAT*. This is done via a user-supplied M-script. The condition specific sub-models are named at the start of an analysis or simulation tool. For some tools, only default sub-models are included. Other tools allow the selection of user-supplied sub-models.

The transfer of data between the various models is done through standardized formats of SIMULINK *Inport* and *Outport* blocks. This allows the application of a sub-model in various other models, i.e. the engine model may be used in both the aircraft generic model and the engine generic model. It may thus be simulated as part of the aircraft as well as independently. The formats of the *Inport/Outport* blocks of the aerodynamic, propulsion, engine, wind and turbulence models are presented in Table 2.1. They are further elaborated in the next section at the discussion of the various SIMULINK models.

2.3.2 Structure of Simulink models

This section describes the above mentioned SIMULINK models in more detail. The formats of the model state, input and output vectors are discussed, as well as the formats of the input/output vectors for data-transfer between the object specific sub-models and the generic models.
The complete outlines of the SIMULINK models through the graphical block diagram representation are given in Appendix D *Definition of reference frames and outlines of* SIMULINK *models.*

The SIMULINK models will be described in the following order:

- generic models
  - generic aircraft models `ac_mod.m` and `ac_modpc.m`
  - generic engine model `eng_mod.m`
  - generic aerodynamic model `aero_mod.m`
- object specific sub-models
  - aerodynamic model `ac_aeromodel`

- propulsion model `ac_powermodel`
- engine model `eng_dynmodel`, `eng_statmodel`, `eng_none.m`
- wind model `wnd_none.m`
- turbulence model `tur_none.m`, `tur_dryd.m`

• operation shells

- default operating shells for a/c open-loop simulation in SIMULINK window `ac_sim.m`, `ac_simpc.m`
- default operating shells for a/c open-loop simulation in command window `ac_fun.m`, `ac_funpc.m`
- default operating shell for a/c closed-loop simulation in SIMULINK window `cl_sim.m`, `cl_simpc.m`
- default operating shell for engine simulation in SIMULINK window `eng_sim.m`
- default operating shell for engine simulation in command window `eng_fun.m`

2.3.3 Generic aircraft model

The generic aircraft model is provided in the SIMULINK S-functions `ac_mod.m` and `ac_modpc.m`. The models use the 6-DOF nonlinear equations of motion for a rigid body aircraft with constant mass over a flat non-rotating earth. The force equations are defined in the air-path reference frame to allow explicit solution of the state equations in the presence of angle of attack rate and angle of sideslip rate dependency for the aerodynamic coefficients. The moment equations are defined in the body reference frame to obtain constant inertia terms. Within the equations of motions, the aircraft and condition specific terms are provided by the user-defined aerodynamic, propulsion, engine, mass, wind and turbulence models.
The model `ac_modpc.m` is a stripped version of `ac_mod.m`. It uses the same equations of motion but it generates only the most significant observation variables. As the complete set of observation variables is rather extensive, `ac_mod.m` requires much computation power and consumes large amounts of memory, making it only applicable to workstations. The model `ac_modpc.m` is more suitable for PC-platforms.
Each generic aircraft model has two realizations, one including the engine models for providing the thrust, the other having the thrust as a direct control input. This dual design results in different formats of the states and control inputs.
The SIMULINK S-functions of `ac_mod.m` and `ac_modpc.m` with their highest sub-system levels are shown in Figure 2.2 and Figure 2.3 respectively.

The generic aircraft model requires the inclusion of aircraft and condition specific sub-models. They configure the general applicable equations of motion to the analyzed aircraft and condition. The sub-models are either SIMULINK S-functions and included in the generic model via *S-function* blocks, or MATLAB M-functions which run independently and generate data which is then included via variable names in mainly *Constant* blocks. The sub-models are designated by variables. The variables which name the aircraft specific sub-models are assigned during the initialization phase of *DASMAT* by executing the aircraft specification routine designated by `ac_info`, see Section 2.4.1 *Aircraft specification routine.* This M-script actually

configures the generic aircraft model to a specific aircraft. The variables for the condition specific sub-models are assigned at the start of an analysis tool from user-supplied entries.
The aerodynamic, propulsion, wind and turbulence models are S-functions. The aerodynamic and propulsion models are called by the file names designated by the variables `ac_aeromodel` and `ac_powermodel`. The wind and turbulence model are called via the variables `ac_windmodel` and `ac_turbmodel` which refer to files in the *DASMAT* directory or user-supplied models in the work-directory. The transfer of data between the sub-models and the generic model is done through the formats presented in Table 2.1.
The engine model is also an S-function. It is however included in the S-function of the propulsion model designated by `ac_powermodel`. Which S-function is called depends however on the selected realization for modeling the thrust during the execution of an analysis tool. If the thrust should be provided via the engine model, then an S-function whose file name is designated by the variable `eng_dynmodel` is called. If the thrust is modelled as a direct control input, then the S-function `eng_none.m` is called from the *DASMAT* directory.
The mass model is an M-function which calculates the mass properties. Its file name is designated by the variable `ac_massmodel`.

The data entered in the generic aircraft model are provided by external signals via a series of *Inport* blocks and by constant values, appearing as variables in the MATLAB workspace, via mainly *Constant* blocks. The external signals are the control inputs which drive the aircraft system. They are supplied via the SIMULINK operating shells during simulation and via the MATLAB functions during the execution of the analysis tools.
The variables in the workspace adapt the generic aircraft model to a specific aircraft and condition and they supply reference locations for a number of observation variables. The aircraft specific variables mainly appear in the aircraft specific sub-models and are read from the aircraft specific data-files. The variables not referring to a specific aircraft appear in the generic aircraft model itself and are supplied by the general *DASMAT* data-files as default values.

The results from the generic aircraft model are both supplied as external signals via a series of *Outport* blocks and as variables in the MATLAB workspace via *To Workspace* blocks. The external signals provide the connections of the generic aircraft model with SIMULINK operating shells for simulation purposes with on-line and off-line evaluation and to MATLAB functions for analysis purposes. The format of these signals is standardized to guarantee a correct data transfer.
The output of variables to the workspace is applied for results which may have an arbitrary format. They are generated in the aircraft specific sub-models. Although they may consist of interesting parameters and should therefore be made accessible, they cannot be transfered to the generic model by way of nested SIMULINK S-functions because their format is not fixed. The format of these variables is only standardized for those first elements which are required in the higher-level model.

The state vector x is determined through the 6-DOF equations of motion and the choice for including of the engine model. The 6-DOF equations result in 12 state variables, indicated as aircraft states x_a. If the engine model is included, then x_a is extended with the engine states x_t times the number of engines. The representation of the state vector in the MATLAB workspace therefore becomes:

$$\texttt{xtot} = x = \begin{cases} x_a & \text{(engine model not included)} \\ [x_a \;\; x_{t_1} \;\; x_{t_2}] & \text{(engine model included)} \end{cases} \tag{2.1}$$

where:

$$\begin{aligned} \texttt{x} \;\; &= \;\; x_a \;\; = \;\; [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \\ \texttt{xt} \;\; &= \;\; x_t \end{aligned} \tag{2.2}$$

A detailed specification of the aircraft states with their units and description is given in Table B.1. The format of the engine states depends on the included engine model, see also Section 2.3.8 *Aircraft specific engine model.* If this is a static version, x_t has no elements and `xt` is an empty variable.

The control inputs u are subdivided into aerodynamic and thrust controls. The aerodynamic controls u_a consist of the primary and secondary flight controls. The thrust controls u_{t_i} are either the throttle (power lever angle PLA_i) settings when the engine model is included or the thrust T_{N_i} itself for each engine. The representation of the control inputs in the MATLAB workspace therefore becomes:

$$\texttt{utot} = u = [u_a \;\; u_{t_1} \;\; u_{t_2}] \tag{2.3}$$

where:

$$\begin{aligned} \texttt{u} \;\; &= \;\; u_a \;\; = \;\; [\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}] \\ \left.\begin{matrix} \texttt{ut(i)} \\ \texttt{Tn(i)} \end{matrix}\right\} \;\; &= \;\; u_{t_i} \;\; = \;\; \begin{cases} PLA_i & \text{(engine model included)} \\ T_{N_i} & \text{(engine model not included)} \end{cases} \end{aligned} \tag{2.4}$$

By default, the throttle settings PLA_i are assigned in to the control variable `ut`. The thrusts T_{N_i} are stored in a separate thrust variable `Tn`. A detailed specification of the control inputs is given in Table B.2.

The observation outputs y may contain up to 240 variables, including the aircraft states, their time derivatives and a variety of other parameters of interest as for example accelerations and force and moment components from the aerodynamic and propulsion sub-models. The observation outputs generated via *Outport* blocks as external signals are distinguished in 18 groups to provide easier accessibility within the SIMULINK models:

y1	x	x_a		y10	ydl	y_{dl}	
y2	xdot	$\dot{x}_a$		y11	yabh	$y_{\alpha\beta h}$	
y3	yair	y_{air}		y12	yCaero	$y_{C_{aero}}$	
y4	yacc	y_{acc}		y13	yFMaero	$y_{FM_{aero}}$	
y5	yfp	y_{fp}		y14	yCgust	y_{C_g}	(2.5)
y6	ys	y_s		y15	yFMgust	y_{FM_g}	
y7	ypqr	y_{pqr}		y16	yCt	y_{C_t}	
y8	yuvw	y_{uvw}		y17	yFMt	y_{FM_t}	
y9	yuvwdot	$y_{\dot{u}\dot{v}\dot{w}}$		y18	yFgrav	$y_{F_{gr}}$	

All above observation outputs are generated by the general aircraft model `ac_mod.m`. The representation of the complete vector of external outputs in the MATLAB workspace therefore becomes:

$$\begin{aligned} \texttt{y} = y = [&x_a \;\; \dot{x}_a \;\; y_{air} \;\; y_{acc} \;\; y_{fp} \;\; y_s \;\; y_{pqr} \;\; y_{uvw} \;\; y_{\dot{u}\dot{v}\dot{w}} \;\; y_{dl} \;\; y_{\alpha\beta h} \\ &y_{C_{aero}} \;\; y_{FM_{aero}} \;\; y_{C_g} \;\; y_{FM_g} \;\; y_{C_t} \;\; y_{FM_t} \;\; y_{F_{gr}}] \end{aligned} \tag{2.6}$$

The compact aircraft model `ac_modpc.m` generates only a reduced number of these observation outputs. Its vector of external outputs is given by:

$$\texttt{y} = y_{red} = [x_a \;\; \dot{x}_a \;\; y_{air} \;\; y_{acc} \;\; y_{fp} \;\; y_s \;\; y_{FM_{aero}} \;\; y_{FM_g} \;\; y_{FM_t}] \tag{2.7}$$

The above lists represent the internal formulation of observation outputs and provide the user with a broad application area for analysis and control design problems. For actual applications, the user must select the specific parameters desired in the operating shell, see Section 2.3.11 *Operating shells for aircraft simulation.* A further specification to the individual observations with their units and description is given in Table B.3 and Table B.4.

The observation outputs written to the MATLAB workspace as variables via *To Workspace* blocks consist for both `ac_mod.m` and `ac_modpc.m` of data generated in the SIMULINK models of the aircraft specific sub-models. This data is not ported to the generic aircraft model because its format may be variable. In fact, this only applies to y_{pow} which is the concatenation of the general engine parameters of all engines and which is generated by the aircraft specific propulsion model. The format of y_{pow} depends on the number of engines and the output format of the individual engine models which is only fixed for the first two elements, being thrust and fuel flow. Hence, its representation in the MATLAB workspace is given by:

$$\texttt{ypow} = y_{pow} = [T_{N_1} \;\; FF_1 \;\; \ldots \;\; T_{N_2} \;\; FF_2 \;\; \ldots] \tag{2.8}$$

The format of y_{pow} is discussed in more detail in Section 2.3.8 *Aircraft specific engine model.*

A number of variables from the MATLAB workspace are entered into the generic aircraft model. These variables can be subdivided into data specifying aircraft specific quantities

and locations for a number of observation parameters. The aircraft specific data consists of geometric parameters as wing area S, wing span b, mean aerodynamic chord $\bar{c}$, the reference location for the aerodynamic coefficients $(x,y,z)_{cg_{ref}}$ in the datum reference frame and the variables `axes_FMaero` and `axes_FMgust` which indicate the applied reference frame for the aerodynamic and gust coefficients:

$$\begin{array}{llllll} \texttt{S} & = S & \texttt{xcgref} & = x_{cg_{ref}} & \texttt{axes\_FMaero} & \\ \texttt{span} & = b & \texttt{ycgref} & = y_{cg_{ref}} & \texttt{axes\_FMgust} & (2.9) \\ \texttt{chord} & = \bar{c} & \texttt{zcgref} & = z_{cg_{ref}} & & \end{array}$$

The variables `axes_FM*` are assigned to [0/1/2] for coefficients defined in the [air-path/ stability/body] reference frame. Moreover, the mass properties are included via the variable `massinit` which has the format:

$$\texttt{massinit} = [m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}] \tag{2.10}$$

The mass m is in kg, the location of the center of gravity $(x,y,z)_{cg}$ is in the datum reference frame and the moments and products of inertia I_{xx}, I_{yy}, I_{zz} and I_{xz} are with respect to the body reference frame and the center of gravity.
The locations required for the observations parameters consist of locations of measurement sensors and navigation ground stations. The measurement locations are defined in the datum reference frame and refer to the accelerometers $(x,y,z)_{i_{acc}}$ and the sensors for angle of attack, angle of sideslip and altitude $(x,y,z)_{i_{\alpha\beta h}}$:

$$\begin{array}{llll} \texttt{xiacc} & = x_{i_{acc}} & \texttt{xiabh} & = x_{i_{\alpha\beta h}} \\ \texttt{yiacc} & = y_{i_{acc}} & \texttt{yiabh} & = y_{i_{\alpha\beta h}} \\ \texttt{ziacc} & = z_{i_{acc}} & \texttt{ziabh} & = z_{i_{\alpha\beta h}} \end{array} \tag{2.11}$$

The location of the navigation ground stations are defined in the earth fixed reference frame and refer to the locations of the DME beacon $(x,y,z)_{DME}$, from which the range is calculated, and the glide-slope and localizer antenna $(x,y,z)_{GS}$ and $(x,y,\psi)_{LOC}$:

$$\begin{array}{llllll} \texttt{xDME} & = x_{DME} & \texttt{xGS} & = x_{GS} & \texttt{xLOC} & = x_{LOC} \\ \texttt{yDME} & = y_{DME} & \texttt{yGS} & = y_{GS} & \texttt{yLOC} & = y_{LOC} \\ \texttt{hDME} & = h_{DME} & \texttt{hGS} & = h_{GS} & \texttt{psiLOC} & = \psi_{LOC} \end{array} \tag{2.12}$$

Moreover, a number of variables are indirectly entered from the workspace into the generic aircraft model via the object specific sub-models. These variables are discussed in the sections devoted to these sub-models.
The geometric parameters and the reference location and reference frame of the aerodynamic and gust coefficients are loaded from the data-file designated by `ac_data`. The mass properties are calculated from the mass model designated by `ac_massmodel`. The locations for the observation parameters may be specified by the user, otherwise they are zero by default from the data-file `ac_genrl.mat` in the *DASMAT* directory.

2.3.4 Generic engine model

The generic engine model is provided in the SIMULINK S-function `eng_mod.m`. The model extends the aircraft specific engine model with the atmospheric model. It thus provides equal conditions as in the generic aircraft model.
The SIMULINK S-function `eng_mod.m` and its highest sub-system level is shown in Figure 2.4.

The engine model is an S-function which is included via an *S-function* block. It is possible to include a static model whose S-function is designated by the variable `eng_statmodel` and a dynamic model designated by the variable `eng_dynmodel`. These variables are specified via the aircraft specification routine designated by `ac_info` (Figure 2.16) and which is executed during the initialization of *DASMAT*. The selection of the static or dynamic engine model is done during the execution of an analysis tool.

The data entered into the generic engine model is provided via external signals via a series of *Inport* blocks and a constant for the engine drag via a *Constant* block. The external signals form the inputs which drive the engine system. They are supplied by the SIMULINK operating shell during simulation and by the MATLAB function for the trimming tool.
The constant engine drag entry is introduced for compatibility of the generic engine model with the inputs of the engine model itself. It is used for engine failure modeling. The assigned value is directly ported to the engine model.

The results from the generic engine model are again both supplied as external signals via a series of *Outport* blocks and as variables in the MATLAB workspace via *To Workspace* blocks. The external signals provide the connections of the generic engine model with SIMULINK operating shells for simulation purposes with on-line evaluation and with the MATLAB routine for trimming.

The engine state vector x_t is completely determined by the engine model which is called by the generic engine model. If the static engine model is included, i.e. the S-function named in the variable `eng_statmodel`, then there are no engine states. If the dynamic engine model is included then x_t is equal to the state vector of the S-function designated by the variable `eng_dynmodel`. The engine states usually comprise the rpm's of the fan and gas turbine.

The inputs of the generic engine model consist of the engine control input $u_t = PLA$, i.e. the throttle (power lever angle) setting, and the aircraft states x_a. The inputs x_a define the operating condition of the engine, basically the air speed and altitude. The engine inputs are thus specified as:

$$[\mathtt{x}\ \ \mathtt{ut}] = [x_a\ \ u_t] \tag{2.13}$$

where:

$$\begin{aligned} \mathtt{x} &= x_a = [p_b\ q_b\ r_b\ V_{TAS}\ \alpha\ \beta\ \phi\ \theta\ \psi\ h_e\ x_e\ y_e] \\ \mathtt{ut} &= u_t = PLA \end{aligned} \tag{2.14}$$

The outputs of the generic engine model are equal to the outputs of the included engine model. All outputs are written to the MATLAB workspace via a *To Workspace* block. The first 8 outputs are furthermore ported to *Outport* blocks for transfer to the SIMULINK operating shell. The format of the engine outputs is only specified for the first two elements which are thrust and fuel flow. The succeeding elements depend on the modelled outputs of the included engine and may consist of rpm's and other interesting engine parameters. This leads to:

$$\texttt{ypow} \;=\; y_{pow} \;=\; [T_N \;\; FF \;\; \ldots] \tag{2.15}$$

The above engine outputs are also used in the concatenation of the engine parameters of all engines in the aircraft specific propulsion model. As such, they are part of the observation outputs of the aircraft model, see (2.8).

The data read from the MATLAB workspace by the generic engine model only consists of the engine drag D_{eng}:

$$\texttt{Deng} \;=\; D_{eng} \tag{2.16}$$

In general, a zero value corresponds to normal engine operation. A non-zero value is used to model engine failure. Its effect depends on the treatment within the engine model itself, see Section 2.3.8 *Aircraft specific engine model.* It may be treated as a reduction on the normally supplied engine thrust or as a switch to select engine failure operation mode.
Additional variables in the workspace may be required by the included engine model. These variables consist of engine parameters and constants which are specific to the engine model. The value of engine drag is retrieved from a trim-file `.tri` which is loaded prior to the use of the generic engine model. The engine specific variables are loaded from the data-file designated by the variable `eng_data`.

2.3.5 Generic aerodynamic model

The generic aerodynamic model is provided in the SIMULINK S-function `aero_mod.m`. The model again extends the aircraft specific aerodynamic model with the atmospheric model to obtain conditions equal to the generic aircraft model. It also corrects the baseline components of the aerodynamic coefficients with contributions from the angle of attack and angle of sideslip rates, in which format the aerodynamic coefficients are supplied by the aircraft specific aerodynamic model. It further contains an axis transformation model to provide flexibility, i.e. the supplied aerodynamic coefficients from the included aerodynamic model may be defined in an arbitrary reference frame and transformed to any desired reference frame.
It should be noted that the aerodynamic model also models the coefficients due to atmospheric turbulence. These coefficients are also with respect to a freely defined reference frame.
The SIMULINK S-function `aero_mod.m` and its highest sub-system level is given in Figure 2.5.

The aerodynamic model is included via an *S-function* block by calling the S-function named in the variable `ac_aeromodel`. This variable is defined during the initialization phase of *DASMAT* by executing the aircraft specification routine designated by `ac_info`, see Figure 2.16.

The data entered in the generic aerodynamic model are provided by external signals via a series of *Inport* blocks and by variables in the MATLAB workspace with constant values via *Constant* blocks. The external signals are the inputs which drive the aerodynamic system. They are supplied by the MATLAB function of the analysis tool and have a fixed format.

The results from the generic aerodynamic model are only supplied as external signals via a series of *Outport* blocks.

The aerodynamic model has no states. It usually consist of series of polynomials or table look-ups. The generic aerodynamic model therefore has no states either.

The inputs of the generic aerodynamic model consist of those variables which primarily influence the aerodynamic forces and moments plus the turbulence velocities and gust velocity rates. The relevant terms for the aerodynamics in still air are the aircraft states x_a, the aerodynamic controls u_a, the angle of attack rate $\dot{\alpha}$ and the angle of sideslip rate $\dot{\beta}$. The gust velocity and velocity rates are included in the variable u_g and have components specified along the negative body axes. The representation of the generic aerodynamic model inputs is thus:

$$[\texttt{x abdot u gust}] = \left[x_a \;\; \dot{\alpha} \;\; \dot{\beta} \;\; u_a \;\; u_g\right] \tag{2.17}$$

where:

$$\begin{aligned}
\texttt{x} \quad &= \quad x_a \quad = \quad [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \\
\texttt{u} \quad &= \quad u_a \quad = \quad [\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}] \\
\texttt{abdot} \quad &= \quad \left[\dot{\alpha} \;\; \dot{\beta}\right] \\
\texttt{gust} \quad &= \quad u_g \quad = \quad \left[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g \bar{c}/V \;\; \dot{\alpha}_g \bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{g_{asym}} \;\; \alpha_{g_{asym}}\right]
\end{aligned} \tag{2.18}$$

The outputs of the generic aerodynamic model consists of the aerodynamic force and moment coefficients plus the force and moment coefficients due to turbulence. They are with respect to the reference location $(x, y, z)_{cg_{ref}}$. Both type of coefficients are available in both the air-path, stability and body reference frame. This gives the following representation of the generic aerodynamic model outputs:

$$[\texttt{yCaero yCgust}] = [y_{C_{aero}} \;\; y_{C_{aero}}] \tag{2.19}$$

where:

$$
\begin{aligned}
\texttt{yCaero} = y_{C_{aero}} &= [C_D \;\; C_Y \;\; C_L \;\; C_\ell \;\; C_m \;\; C_n]_a \\
&\quad [C_D \;\; C_Y \;\; C_L \;\; C_\ell \;\; C_m \;\; C_n]_s \\
&\quad [C_T \;\; C_Y \;\; C_N \;\; C_\ell \;\; C_m \;\; C_n]_b \\
\texttt{yCgust} = y_{C_{gust}} &= [C_{D_g} \;\; C_{Y_g} \;\; C_{L_g} \;\; C_{\ell_g} \;\; C_{m_g} \;\; C_{n_g}]_a \\
&\quad [C_{D_g} \;\; C_{Y_g} \;\; C_{L_g} \;\; C_{\ell_g} \;\; C_{m_g} \;\; C_{n_g}]_s \\
&\quad [C_{T_g} \;\; C_{Y_g} \;\; C_{N_g} \;\; C_{\ell_g} \;\; C_{m_g} \;\; C_{n_g}]_b
\end{aligned}
\tag{2.20}
$$

The aerodynamic outputs are again part of the observation outputs of the aircraft model through $\mathtt{y}_{12} = y_{C_{aero}}$ and $\mathtt{y}_{14} = y_{C_g}$.

The variables in the MATLAB workspace specify geometric parameters as wing area S, wing span b and mean aerodynamic chord $\bar{c}$, the reference location for the aerodynamic coefficients $(x, y, z)_{cg_{ref}}$ and the variables `axes_FMaero` and `axes_FMgust` which indicate the applied reference frame for the aerodynamic and gust coefficients. The reference location is defined in the datum reference frame and entered via the signal for the mass properties. Hence, the variables read from the workspace are:

`S`	$= S$	`xcgref`	$= x_{cg_{ref}}$	`axes_FMaero`	
`span`	$= b$	`ycgref`	$= y_{cg_{ref}}$	`axes_FMgust`	(2.21)
`chord`	$= \bar{c}$	`zcgref`	$= z_{cg_{ref}}$		

The variables `axes_FM*` are assigned to [0/1/2] for coefficients defined in the [air-path/ stability/body] reference frame. All above variables are loaded from the data-file designated by `ac_data`.

2.3.6 Aircraft specific aerodynamic model

The aircraft specific aerodynamic model is an independent SIMULINK S-function which calculates the aerodynamic force and moment coefficients and the force and moment coefficients due to turbulence, the latter sometimes being called gust coefficients. The model is included in the generic aircraft model and the generic aerodynamic model via an *S-function* block and it is designated by the variable `ac_aeromodel`.
A template SIMULINK model for the aerodynamic model is provided in the S-function `ac_aero.m` in the *DASMAT* directory. It is shown in Figure 2.6.

The aerodynamic coefficients are calculated from the aircraft states, the primary and secondary flight controls, the aircraft center of gravity and airdata parameters. The coefficients C are expressed as baseline components C_{base} and linear contributions from the angle of attack rate $C_{\dot{\alpha}}$ and angle of sideslip rate and $C_{\dot{\beta}}$:

$$C = C_{base} + C_{\dot{\alpha}} \frac{\dot{\alpha}\bar{c}}{V_{TAS}} + C_{\dot{\beta}} \frac{\dot{\beta}b}{V_{TAS}} \tag{2.22}$$

The forces and moments due to turbulence are also modelled within the aircraft specific aerodynamic model. The effects of turbulence, which is in fact a disturbance on the air speed, may be modelled as extra aerodynamic forces and moments, or they may be modelled as corrections to the velocity terms in the equations of motion. In the first case, the coefficients due to turbulence may be expressed in terms of the aerodynamic derivatives and thus made dependent of the gust velocities and gust velocity rates. The gust penetration effect is then also easily modelled. In the second case, the coefficients due to turbulence are deduced from the extra forces and moments which arise from using the relative air velocity instead of inertial velocity in the equations of motion. Now, the gust penetration is not modelled and the effect of turbulence is in fact independent of the aerodynamic model. Nevertheless, by including the turbulence effects within the aerodynamic model, the user has the flexibility of selecting either modeling option.

The aerodynamic model has no states. It usually consist of series of polynomials or table look-ups.

The inputs of the aircraft specific aerodynamic model are the aircraft states x_a, the aerodynamic control inputs u_a, the location of the aircraft center of gravity $(x, y, z)_{cg}$, the airdata parameters collected in y_{ad_1} and y_{ad_2} and the gust velocity terms u_g. The center of gravity is defined in the datum reference frame and the gust velocity terms are taken positive along the negative body axes. The aerodynamic model inputs thus become:

$$[\texttt{x u cg yad1 yad2 gust}] = [x_a \;\; u_a \;\; (x,y,z)_{cg} \;\; y_{ad_1} \;\; y_{ad_2} \;\; u_g] \tag{2.23}$$

where:

$$\begin{aligned}
\texttt{x} &= x_a &&= [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \\
\texttt{u} &= u_a &&= [\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}] \\
\texttt{cg} &= (x,y,z)_{cg} &&= [x_{cg} \;\; y_{cg} \;\; z_{cg}] \\
\texttt{yad1} &= y_{ad_1} &&= [M \;\; \bar{q}] \\
\texttt{yad2} &= y_{ad_2} &&= [Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}] \\
\texttt{gust} &= u_g &&= \left[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}\right]
\end{aligned} \tag{2.24}$$

It is advised to put a saturation block after multiplexing the aerodynamic control inputs, see Figure 2.6. This garantees that the inported control inputs cannot attain unrealistic values, for example deflections beyond any hard stops on the controls, and thus lead to an unreliable

calculation of the aerodynamic coefficients. The limits may be assigned in the variables `umin` and `umax`.

The outputs of the aircraft specific aerodynamic model are the aerodynamic force and moment coefficients, subdivided in the baseline component C_{base} and the $\dot{\alpha}$- and $\dot{\beta}$-contributions according to (2.22), and the force and moment coefficients due to turbulence. All coefficients are defined along the positive axes and may be supplied in either the air-path, stability or body reference frame. The aerodynamic model outputs are therefore:

$$[\texttt{Ca Cadot Cbdot Cag}] = \left[C_{base} \;\; C_{\dot{\alpha}} \;\; C_{\dot{\beta}} \;\; C_g\right] \tag{2.25}$$

where:

$$\begin{aligned}
\texttt{Ca} &= C_{base} = [C_X \;\; C_Y \;\; C_Z \;\; C_\ell \;\; C_m \;\; C_n] \\
\texttt{Cadot} &= C_{\dot{\alpha}} = [C_{X_{\dot{\alpha}}} \;\; C_{Y_{\dot{\alpha}}} \;\; C_{Z_{\dot{\alpha}}} \;\; C_{\ell_{\dot{\alpha}}} \;\; C_{m_{\dot{\alpha}}} \;\; C_{n_{\dot{\alpha}}}] \\
\texttt{Cbdot} &= C_{\dot{\beta}} = \left[C_{X_{\dot{\beta}}} \;\; C_{Y_{\dot{\beta}}} \;\; C_{Z_{\dot{\beta}}} \;\; C_{\ell_{\dot{\beta}}} \;\; C_{m_{\dot{\beta}}} \;\; C_{n_{\dot{\beta}}}\right] \\
\texttt{Cag} &= C_g = [C_{X_g} \;\; C_{Y_g} \;\; C_{Z_g} \;\; C_{\ell_g} \;\; C_{m_g} \;\; C_{n_g}]
\end{aligned} \tag{2.26}$$

If no $\dot{\alpha}$- or $\dot{\beta}$-contributions are modelled, then the corresponding *Outport* blocks are fed by zero constants.

The aircraft specific aerodynamic model generally reads numerous variables from the MATLAB workspace. Although these variables will be very specific to the model, they will incorporate look-up tables for the various aerodynamic coefficients, constants and geometric parameters. All these variables are loaded from the data-file designated by the variable `ac_data`.

2.3.7 Aircraft specific propulsion model

The aircraft specific propulsion model is an independent SIMULINK S-function which calculates the total force and moment generated by the propulsion system. That is, the force and moment contributions of all engines and possible engine-airframe interference and gyroscopic effects. The force and moment contributions are the baseline terms and are calculated from the generated thrust of each engine, its position and the direction of its thrust line. The interference and gyroscopic effects may be modelled as extra contributions to the forces and moments. The model is included in the generic aircraft model via an *S-function* block and it is designated by the variable `ac_powermodel`.
An example SIMULINK model for a propulsion model of a two-engine aircraft is provided in the S-function `ac_pow.m` in the *DASMAT* directory. This model is shown in Figure 2.7.

The thrust of each engine is obtained from an independent SIMULINK S-function. Two kinds of S-functions may be included, depending on whether the thrust control is selected as throttle (power lever angle) setting or as thrust itself. In the first case the S-function which models the engine is included. This model is designated by the variable `eng_dynmodel`. In the second

case, no real engine model is included, but instead the feed-through model provided by the S-function `eng_none.m` from the *DASMAT* package. This model just directs its input, i.e. the thrust, to the output in the prescribed output format, see also Section 2.3.8 *Aircraft specific engine model.*

The results from the propulsion model consist of forces and moments and a vector with the concatenated outputs of all included engine models. The forces and moments are ported to the generic aircraft model via *Outport* blocks. The vector with concatenated engine model outputs is written to the MATLAB workspace via a *To Workspace* block. The reason is that the forces and moments are required in the generic aircraft model and have a fixed format which does not depend on the number of engines. On the other hand, the vector size of the concatenated engine model outputs is determined by the number of engines and can therefore not be related to an invariable data structure in the generic aircraft model. The engine model outputs, aside from the thrust, are moreover not relevant in the aircraft model. However, they may still be interesting for the user and therefore made accessible in the workspace.

The inputs of the aircraft specific propulsion model are the aircraft states x_a, the engine control inputs u_{t_i}, the atmospheric parameters y_{atm} and airdata parameters y_{ad_1} and y_{ad_2}. The propulsion model inputs thus become:

$$[\texttt{x ut yatm yad1 yad2}] = [x_a \;\; u_{t_1} \;\; u_{t_2} \;\; y_{atm} \;\; y_{ad_1} \;\; y_{ad_2}] \tag{2.27}$$

where:

$$\begin{aligned}
\texttt{x} &= x_a = [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \\
\texttt{ut(i)} &= u_{t_i} = \begin{cases} PLA_i & \text{(engine model included)} \\ T_{N_i} & \text{(engine model not included)} \end{cases} \\
\texttt{yatm} &= y_{atm} = [p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}] \\
\texttt{yad1} &= y_{ad_1} = [M \;\; \bar{q}] \\
\texttt{yad2} &= y_{ad_2} = [Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]
\end{aligned} \tag{2.28}$$

The outputs of the aircraft specific propulsion model via *Outport* blocks are the total force and moment of the engines. These forces and moments are defined along the positive axes of the body reference frame. The moments are with respect to the origin of the datum reference frame. The propulsion model outputs are:

$$\texttt{FMt} = y_{FM_t} = [X_t \;\; Y_t \;\; Z_t \;\; \bar{L}_t \;\; M_t \;\; \bar{N}_t] \tag{2.29}$$

The engine parameters of all engines are also written to the MATLAB workspace via *To Workspace* blocks. They consist of the outputs of the engine models and are concatenated in the vector y_{pow}. The format of y_{pow} is:

$$\texttt{ypow} \;=\; y_{pow} \;=\; [T_{N_1} \;\; FF_1 \;\; \ldots \;\; T_{N_2} \;\; FF_2 \;\; \ldots] \tag{2.30}$$

This output is equal to (2.8) generated in the generic aircraft model. The format of y_{pow} is discussed in more detail in Section 2.3.8 *Aircraft specific engine model.*

The variables which are entered into the aircraft propulsion model from the MATLAB workspace consist of the location of the engines, the direction of the thrust lines and the drag of the engines. The variables are row vectors with lengths corresponding to the number of engines. The engine locations $(x, y, z)_{eng}$ are specified in the datum reference frame. The directions of the thrust lines are specified by the tow angle t_o and the tilt angle t_i. The tow angle is defined as the angle between the projection of the positive thrust vector on the datum XY-plane and the negative datum X-axis. The tilt angle is defined as the angle between the projection of the positive thrust vector on the datum XZ-plane and the negative datum X-axis. The engine drag represents a variable which is used for engine failure modeling. Its effect depends on the treatment within the engine model itself, see Section 2.3.8. In general, a zero value corresponds to normal engine operation. This now gives the following entries from the workspace:

$$\begin{array}{lllllll}
\texttt{xeng(i)} & = & x_{eng_i} & \texttt{toweng(i)} & = & t_{o_{eng_i}} & \texttt{Deng(i)} \;=\; D_{eng_i} \\
\texttt{yeng(i)} & = & y_{eng_i} & \texttt{tilteng(i)} & = & t_{i_{eng_i}} & \\
\texttt{zeng(i)} & = & z_{eng_i} & & & &
\end{array} \tag{2.31}$$

These variables are loaded from the data-file designated by `ac_data`, except for D_{eng} which is retrieved from a trim-file `.tri`.

2.3.8 Aircraft specific engine model

The aircraft specific engine model is an independent SIMULINK S-function which calculates the thrust and a number of interesting engine parameters as for example rpm's, fuel flow, various temperatures, etc. The model is included in the generic aircraft model via the aircraft specific propulsion model and directly in the generic engine model, in both cases via an *S-function* block.

The engine model may be either a dynamic or a static model. With the option of using thrust as control input for the aircraft model, *DASMAT* also supplies a feed-through model which just ports the injected thrust in the standardized output format.
The S-function of the engine dynamic model is designated by the variable `eng_dynmodel`. This model has state variables which are determined by the way the engine is modelled. This may be a simple first-order lag model where x_t is the actual power level. The model may also be nonlinear using thermodynamic relationships for describing the physical processes inside the engine. For turbofan engines, x_t may then consist of the rpm's of the gas generator and the fan. The dynamic engine model is used via the aircraft specific propulsion model in the generic aircraft model for executing simulations with the throttle (power lever angle) setting as thrust controls.

The S-function of the engine static model is designated by the variable `eng_statmodel`. This model has no state variables and may be derived from the dynamic version by removing feedback loops of the actual and the commanded power levels or rpm's. The static engine is used in the trimming routine for trimming the engine, i.e. calculating the required power lever angle necessary for the required thrust.
The feed-through model is provided in the S-function `eng_none.m`, see Figure 2.8. This model has the standardized input/output format, but only the thrust control input is transfered from the appropriate *Inport* block to the *Outport* block. The other *Outport* blocks are fed with Not-a-Number constants.

The state vector x_t depends on the selection of the static or dynamic version of the engine model, or the *DASMAT* supplied feed-through model `eng_none.m`. For the static version and `eng_none.m`, x_t is an empty vector. For the dynamic version, x_t is determined by the modeling of the engine. The engine model states are represented by:

$$\texttt{xt} = x_t \tag{2.32}$$

The inputs of the aircraft specific engine model specify the operating conditions and the thrust control. The operating conditions are inported via the aircraft states x_a, the atmospheric parameters y_{atm}, the airdata parameters collected in y_{ad_1} and y_{ad_2} and the engine drag D_{eng}. The thrust control input u_t depends on the selection of the engine model itself or the *DASMAT* supplied feed-through model `eng_none.m`. For the engine models u_t is the throttle (power lever angle) setting. For `eng_none.m` u_t consists of the thrust itself. The engine model inputs are therefore:

$$[\texttt{x ut yatm yad1 yad2 Deng}] = [x_a \;\; u_t \;\; y_{atm} \;\; y_{ad_1} \;\; y_{ad_2} \;\; D_{eng}] \tag{2.33}$$

where:

$$\begin{aligned}
\texttt{x} &= x_a &&= [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \\
\texttt{ut} &= u_t &&= \begin{cases} PLA & \texttt{(eng_dynmodel / eng_statmodel)} \\ T_N & \texttt{(eng_none.m)} \end{cases} \\
\texttt{yatm} &= y_{atm} &&= [p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}] \\
\texttt{yad1} &= y_{ad_1} &&= [M \;\; \bar{q}] \\
\texttt{yad2} &= y_{ad_2} &&= [Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}] \\
\texttt{Deng} &= &&= D_{eng}
\end{aligned} \tag{2.34}$$

It is advised to put a saturation block after inporting the thrust control. This garantees that the inported control input cannot attain an unrealistic value for which the engine model is not valid. For engine models, the limits for the throttle (power lever angle) setting may be assigned in the variables `utmin` and `utmax`. In the feed-through model `eng_none.m`, the thrust

is not saturated by default. However, the thrust limits may be assigned in `Tnmin` and `Tnmax`.

The outputs of the aircraft specific engine model consist the thrust, the fuel flow and any additional engine parameters which are not really required but which may be of interest to the user. For the static engine model, the additional parameters should anyway include the engine states for the dynamic engine model. The format of the outputs is not unique, but the first two outputs should be thrust and fuel flow. The subsequent outputs are the additionally modelled engine parameters. The output format below gives an example of engine outputs that satisfies the data structure:

$$\begin{aligned} \texttt{ypow} \;=\; y_{pow} \;=\; & [T_N \;\; FF \;\; \ldots] \\ & [T_N \;\; FF \;\; NLP \;\; NHP \;\; NLP_c \;\; NHP_c \\ & \;\; WT \;\; ITT \;\; EGT \;\; OILT \;\; OILP] \end{aligned} \tag{2.35}$$

The aircraft specific engine model generally reads numerous variables from the MATLAB workspace. Although these variables will be very specific to the model, they will incorporate look-up tables for the various engine parameters and constants. All these variables are loaded from the data-file designated by the variable `eng_data`.

2.3.9 Condition specific wind model

The condition specific wind model is an independent SIMULINK S-function which specifies the wind velocities. The wind is defined as the velocity with respect to the earth of the air which surrounds the aircraft but which is not disturbed by the aircraft. However, it does not include turbulence which is a stochastic velocity signal with zero mean and which is specified in the turbulence model in Section 2.3.10 *Condition specific turbulence model.* The model is included in the generic aircraft model via an *S-function* block and is designated by the variable `ac_windmodel`.

The wind velocity may be modelled as a function of geographical position and thus lets the aircraft fly through a varying wind field. In this way the effects of windshear may be investigated by modeling abrupt or more gradual changes in wind speed and direction. The *DASMAT* package provides only a model for zero wind condition via the S-function `wnd_none.m`, see Figure 2.9, but the user may supply a wind model himself.

The inputs of the wind model are the aircraft states x_a. Only the elements of x_a which specify the aircraft position are generally used in the function for the wind velocities. For a constant wind field, no elements of x_a are used at all, as in `wnd_none.m`. The wind model inputs thus become:

$$\texttt{x} \;=\; x_a \;=\; [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \tag{2.36}$$

The outputs of the wind model are the wind velocities along the positive axes of the earth reference frame. The outputs thus become:

$$\texttt{wind} = V_{w_e} = [u_{w_e} \; v_{w_e} \; w_{w_e}] \tag{2.37}$$

The wind model may read data from the MATLAB workspace. It is however advisable to have any wind parameters explicitly declared in the wind model.

2.3.10 Condition specific turbulence model

The condition specific turbulence model is an independent SIMULINK S-function which generates the gust velocities and gust velocity rates. Gust is a random process which describes the chaotic motion of the air. Here, it is regarded as the zero mean stochastic contribution to the wind velocity which is modelled in the wind model, described in the previous section. The model is included in the generic aircraft model via an *S-function* block and is designated by the variable `ac_turbmodel`.

Turbulence models are available for calm air without turbulence and for turbulence which is stationary, isotropic, homogeneous with a Gaussian distribution. The turbulence is derived from white noise using analytical functions for the turbulence spectra, i.e. Dryden spectra. The model for zero turbulence condition and the turbulence model using the Dryden spectra are provided by the *DASMAT* package via the S-functions `tur_none.m` and `tur_dryd.m` respectively.

The S-function `tur_none.m` for no turbulence just supplies zero gust velocities and rates in the correct output format, see Figure 2.10.
The S-function `tur_dryd.m` models the gust velocities and rates using the Dryden spectra. The spectra define two-dimensional fields of flow as function of spatial frequencies along the earth X-axis and Y-axis. However, for modeling the forces and moments due to turbulence, the spectra are reduced to one dimensional spectra depending on the circular frequency along the earth X-axis. The symmetrical gust velocities will give rise to symmetrical forces and moments and the asymmetrical gust velocities will cause asymmetric forces and moments. The power spectral density functions are converted to transfer functions of stable non-minimum phase filters. The filters are modelled in state-space form in the SIMULINK model because the polynomial coefficients in the transfer functions depend on the air speed and geometric altitude. The equations for the transfer functions are taken from [12]. The S-function of `tur_dryd.m` is shown in Figure 2.11.
For the symmetrical forces and moments, the following transfer functions are applied:

$$\begin{aligned} H_{\hat{u}_g w}(s) &= \frac{\sigma_{u_g}}{V}\sqrt{\frac{2L_g}{V}}\,\frac{1}{1+\frac{L_g}{V}s} \\ H_{\alpha_g w}(s) &= \frac{\sigma_{w_g}}{V}\sqrt{\frac{L_g}{V}}\,\frac{1+\sqrt{3}\frac{L_g}{V}s}{\left(1+\frac{L_g}{V}s\right)^2} \end{aligned} \tag{2.38}$$

For the asymmetrical forces and moments, the next transfer functions are applied:

$$
\begin{aligned}
H_{\beta_g w}(s) &= \frac{\sigma_{v_g}}{V}\sqrt{\frac{L_g}{V}}\,\frac{1+\sqrt{3}\frac{L_g}{V}s}{\left(1+\frac{L_g}{V}s\right)^2} \\
H_{\hat{u}_{gasym}w}(s) &= \sqrt{I_{\hat{u}_g}(0,B)\,\frac{L_g}{V}}\,\frac{1+\tau_3\frac{L_g}{V}s}{\left(1+\tau_1\frac{L_g}{V}s\right)\left(1+\tau_2\frac{L_g}{V}s\right)} \\
H_{\alpha_{gasym}w}(s) &= \sqrt{I_{\alpha_g}(0,B)\,\frac{L_g}{V}}\,\frac{1+\tau_6\frac{L_g}{V}s}{\left(1+\tau_4\frac{L_g}{V}s\right)\left(1+\tau_5\frac{L_g}{V}s\right)}
\end{aligned}
\tag{2.39}
$$

In these transfer functions, the intensity σ_g and scale length L_g are constants which completely describe the turbulence. The parameters $I_{\hat{u}_g}(0,B)$ and $I_{\alpha_g}(0,B)$ and $\tau_1 \ldots \tau_6$ are respectively variances and time constants in the approximated power spectral density functions for α_{gasym} and β_{gasym}. They depend on $B = \frac{b}{2L_g}$, the fraction of the wing span to the scale length of the turbulence, and as such interpolated from tables in [12].
The transfer functions of the gust velocity rates simply follow from multiplying the transfer functions of the gust velocities with the Laplace variables s.

The turbulence models do have states, but these are irrelevant for the aircraft model.

The inputs of the turbulence models consist of the aircraft states x_a. The air speed and geometric altitude are used for defining the intensity and scale length of the turbulence. If desired, the horizontal position may be used to define an area where the turbulence occurs. The turbulence model inputs are:

$$
\mathtt{x} = x_a = [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \tag{2.40}
$$

The outputs of the turbulence models are the gust velocities and gust velocity rates. All terms are dimensionless and defined along the negative body axes:

$$
\mathtt{gust} = u_g = \left[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}\right] \tag{2.41}
$$

The terms $\hat{u}_g$, α_g, $\frac{\dot{\hat{u}}_g\bar{c}}{V}$ and $\frac{\dot{\alpha}_g\bar{c}}{V}$ should be used for calculating the symmetrical forces and moments due to turbulence. The terms β_g, $\frac{\dot{\beta}_g b}{V}$, α_{gasym} and β_{gasym} should be used for calculating the asymmetrical forces and moments due to turbulence.

The variables read from the MATLAB workspace are the constants in the transfer functions for the gust velocities (2.38) and (2.39). The values σ_g and L_g apply to all components of the gust because the turbulence is considered to be isotropic. The parameters $I_{\hat{u}_g}(0,B)$ and $I_{\alpha_g}(0,B)$ and $\tau_1 \ldots \tau_6$ are calculated during the initialization phase of *DASMAT* from look-up tables. Hence:

`sigmag`	$=$ σ_g	$=$ 1 m^2/s^2		`Iug`	$=$ $I_{\hat{u}_g}(0,B)$	
`Lg`	$=$ L_g	$=$ 300 m		`Iag`	$=$ $I_{\alpha_g}(0,B)$	(2.42)
				`taug(i)`	$=$ τ_i	

The variables and look-up tables are defined in the data-file `ac_turb.mat` in the *DASMAT* directory.

2.3.11 Operating shells for aircraft simulation

Default operating shells are available for open-loop and closed-loop aircraft simulation in the SIMULINK window and from the command window. Open-loop simulation may be performed using either the general aircraft model or its compact version. Closed-loop simulation is only available for the general model in the SIMULINK window. Using the general aircraft model `ac_mod.m`, the operating shells are provided in the SIMULINK S-functions `ac_sim.m` and `ac_fun.m` for performing open-loop simulation in the SIMULINK window and the command window respectively, and in the SIMULINK S-function `cl_sim.m` for closed-loop simulation. Using the compact aircraft model `ac_modpc`, the SIMULINK S-functions are `ac_simpc.m`, `ac_funpc.m` and `cl_simpc.m`. These operating shells may be adapted by the user to serve personal wishes, see the remarks at the end of this section.
The SIMULINK S-functions of the operating shells are shown in Figure 2.12 and Figure 2.13 for open-loop simulation and in Figure 2.14 for closed- loop simulation. All default operating shells for open-loop simulation are basically the same.

The operating shell `ac_sim.m` for open-loop simulating the general aircraft model in the SIMULINK window is most extensive. It offers the greatest simulation flexibility and the widest choice of selectable observation outputs. The generic aircraft model `ac_mod.m` is included via an *S-function* block.
The control inputs u are entered from the MATLAB workspace into the model via *Constant* blocks and again written to the workspace via *To Workspace* blocks as variables `u`, `ut` and `Tn`. In this way the control inputs become available at the integration time points which are also used for the state variables and observation outputs. Via the variable `do_PLA` in a *Gain* block, the applied thrust control is distinguished as throttle (power lever angle) setting, i.e. including the engine model, or as thrust itself. When using `ac_sim.m`, the control inputs may be changed on-line during simulation using sliders in an optionally available **On-line Control** window, see Section 3.3.1 *Open-loop simulation in* SIMULINK *window.*
The observation outputs y may become available in three ways. The blocks with labels *Response* and *Animation* contain S-functions which on-line visualize the dynamic responses as a 3D-animation or as time-history plots. They are activated when the corresponding option was selected during the dialog for performing the simulation from the SIMULINK window, see Section 3.3.1. The observation outputs are also demultiplexed and reduced in two steps to scalar signals. The first demultiplexing step separates y into 18 groups according to (2.6). Any second step separates the selected groups further into the scalar observations in Table B.3. Scalar signals connected to an *Outport* block are written to the MATLAB workspace and included in the variable `y`, which is defined as return variable in the Control Panel dialog box and which is displayed by selecting **Parameters** from the **Simulation** menu.

Moreover, the time vector t and the complete state vector x, i.e. aircraft states x_a plus the states x_t of all engines, are defined as return variables. As such, they are written to the workspace as variables `t` and `xtot`.
The SIMULINK window finally contains a *Button* block, a *Clock* block and a *Floating Scope* block. The red *Button* block configures, when activated it by double clicking, the generic aircraft model to a specific aircraft and condition by including the appropriate sub-models. These sub-models are designated by variables assigned in the M-script of the aircraft specification routine designated by `ac_info`, see Section 2.4.1 *Aircraft specification routine.* After successful completion the block turns green. The *Clock* block provides a window that continuously displays the elapsed time as the simulation progresses. The *Floating Scope* block may be used to display on-line the activity of any block-connection.

The operating shell `ac_simpc.m` for open-loop simulating the compact aircraft model in the SIMULINK window is almost identical to the just described `ac_sim.m`. In order to reduce computations and required memory, the blocks for the on-line visualization are removed. The observation outputs thus only become off-line available in the MATLAB workspace. Furthermore, the use of the `ac_modpc.m` involves a reduced number of available observation outputs, see (2.7) and Table B.4.

The operating shell `ac_fun.m` for open-loop simulating the general aircraft model in the command window is derived from `ac_sim.m`. The control inputs are now entered into the model via *Inport* blocks which allows them to be controlled from a MATLAB M-function. All blocks dealing with any visualization are removed from the model. However, the *Clock* block is connected to a block labelled *Display* to print the elapsed time in the command window as the simulation progresses.

The operating shell `ac_funpc.m` for open-loop simulating the compact aircraft model in the command window is identical to `ac_fun.m`, except for the demultiplexing and reduction of the observation outputs. This is now based on the available observation outputs from the compact aircraft model `ac_modpc.m`.

The operating shells `cl_sim.m` and `cl_simpc.m` for closed-loop simulating the general and compact aircraft model in the SIMULINK window essentially differ from the operating shells for open-loop simulation, see Figure 2.14. Now, the control inputs u are provided via an actuator model by the controller. The controller gets its inputs from a reference block and a sensor model. Moreover, a transport delay block is included to prevent the occurrence of an algebraic loop.
All models are included via *S-function* blocks. The aircraft model is related to the operating shell, i.e. the general model `ac_mod.m` is included in `cl_sim.m` and the compact model `ac_modpc.m` is included in `cl_simpc.m`. The models for the controller, actuators, sensors and reference signals generator are designated by variables assigned in the M-script of the closed-loop specification routine designated by `cl_info`, see Section 4.3 MATLAB *routine for closed-loop specification.*
The time vector t, control inputs u and observation outputs y become available in the same way as with the operating shell `ac_sim.m` for open-loop simulating. The reference signals r are also available. They are written to the MATLAB workspace via *To Workspace* blocks as the variables `u`, `ut` and `r`, and via *Outport* blocks as return variables `t` and `y`. The state vector

x_{cl} of the closed-loop system, which is returned to the workspace as variable `xcl`, consists of the aircraft states x_a, the states x_t of all engines, the states of the controller and the states of the actuators.
The SIMULINK window further contains two *Button* blocks, a *Clock* block, a *Floating Scope* block and a *Control Mode Panel* block-array. The left *Button* block configures the operating shell by including the closed-loop models specified in the closed-loop specification routine `cl_info`, while the right *Button* block again configures the generic aircraft model by including the aircraft and condition specific sub-models specified in the aircraft specification routine `ac_info`. The blocks in the *Control Mode Panel* engage standard autopilot modes or disconnect a control channel. When a block is activated, values are assigned to standard named variables in the MATLAB workspace which may be used within the controller model to operate switches for controlling the applicable equations.
A more extensive description of the closed-loop applications is presented in Chapter 4 *Control Design Applications.*

The user may modify the above default operating shells. A very useful modification is a different selection of observation outputs. The following remarks should be observed.
A (default) operating shell to be modified is best copied to the current work-directory. When executing the simulation tool, a non-default operating shell may be selected from this directory, see Section 3.3.1 and Section 3.3.2.
The operating shells for performing simulations in the SIMULINK window are not allowed to have *Inport* blocks. Such blocks assume external inputs which are not present. Instead blocks from the *simulink/Sources* library may be used, e.g. the blocks *Constant*, *Signal Generator*, *From Workspace*, *From File*, etc. To guarantee on-line control for open-loop simulation, the control inputs should be entered via *Constant* blocks which have identical labels as in the default operating shells, for example `ac_sim.m`. The operating shells may have *Outport* blocks in which case the corresponding signals are written to the workspace as variable `y` if it is specified as return variable in the Control Panel dialog box which is displayed by selecting **Parameters** from the **Simulation** menu. Blocks from the *simulink/Sinks* library may also be used, especially the blocks *Scope*, *Graph* and *To Workspace*. The *Scope* and *Graph* blocks display signals during simulation while the *To Workspace* block lets the user manipulate the signals off-line after the simulation is finished.
The operating shells for performing simulations in the command window are only allowed to be modified as far as the selection of observation outputs via the *Outport* blocks and any method for making signals available. However, including any block which displays signals on-line, for example a *Scope* block, has no effect. The user may consider to write a complete group of observation outputs to the workspace via a *To Workspace* block.

2.3.12 Operating shells for engine simulation

Default operating shell are also available for engine simulation in the SIMULINK window and the command window. The operating shells are provided in the SIMULINK S-functions `eng_sim.m` and `eng_fun.m`. These S-function are shown in Figure 2.15.

The operating shells `eng_sim.m` and `eng_fun.m` for simulating an engine model are basically the same as the operating shells `ac_sim.m` and `ac_fun.m` for simulating the aircraft model. Now, the generic engine model `eng_mod.m` is included via an *S-function* block.

The operating shell `eng_sim.m` for simulating in the SIMULINK window is again most extensive. The inputs of the generic engine model, i.e. the engine control input $u_t = PLA$ and the aircraft states x_a, are all entered from the MATLAB workspace via *Constant* blocks. They are again written to the workspace via *To Workspace* blocks as variables `ut` and `x` to make them available at the integration time points. The engine control input PLA may be changed on-line during simulation using a slider in an optionally available **On-line Control** window, see Section 3.4 *Simulation of Engine.*
The outputs of the generic engine model become available in the MATLAB workspace as variable `ypow` via a *To Workspace* block included the S-function `eng_mod.m`, see Section 2.3.4 *Generic engine model.* Moreover, the first 8 outports are ported to the operating shell. Although connected to *Outport* blocks, these signals are not further processed this way. Their activity may however thus be displayed in the opened **Floating Scope** window.
The time vector t and the engine state vector x_t are defined as return variables in the Control Panel dialog box which is displayed by selecting **Parameters** from the **Simulation** menu. As such, they are written to the workspace as variables `t` and `xt`.
The SIMULINK window finally contains a *Button* block, a *Clock* block and a *Floating Scope* block. The red *Button* block configures, when activated it by double clicking, the generic engine model by including the appropriate engine model. After successful completion the block turns green. The *Clock* block provides a window that continuously displays the elapsed time as the simulation progresses. The *Floating Scope* block may be used to display on-line the activity of any block-connection.

The operating shell `eng_fun.m` for simulating the engine model in the command window only differs slightly from the just described `eng_sim.m`. The engine control input PLA is now entered into the model via an *Inport* block and may thus be controlled from a MATLAB M-function. Moreover, the *Floating Scope* block is removed and the *Clock* block is connected to a block labelled *Display* for printing the elapsed simulation time in the command window.

2.4 MATLAB routines

This section describes the MATLAB routines. These routines consist of M-files which are partly included in the *DASMAT* package and which should be partly supplied by the user. The routines can be distinguished by the following subdivision:

- routines which operate *DASMAT* tools
- aircraft and closed-loop specification routine
- aircraft specific mass model

The routines for the *DASMAT* tools are provided in the *DASMAT* package and included in the *DASMAT* directory. The other M-files should be user-supplied and be placed in the operational work-directory.

The routines for the *DASMAT* tools will not be discussed here. Instead, they will be clarified via the operational aspects of the *DASMAT* tools in Chapter 3 *Operation.* The closed-loop specification routine will be discussed in Chapter 4 *Control Design Applications* where the

configuration and operation of closed-loop systems is discussed. This section is devoted to the required commands in the aircraft specification routine and the aircraft specific mass model.

2.4.1 Aircraft specification routine

The aircraft specification routine is a MATLAB M-script which adapts the generic models to a specific aircraft. The routine designates the names of aircraft specific sub-models and data-files to standardized variables. These variables are then used for calling the SIMULINK sub-models in *S-function* blocks in the generic SIMULINK models, for executing the MATLAB routines and for loading the data-files.
The basic lay-out of MATLAB M-script is given in Figure 2.16. The statements should be completed with strings for the file names.

The routine is designated by the variable `ac_info`. This variable is defined during the initialization phase of *DASMAT* via input from the command window, see Section 3.2 *Starting and initializing DASMAT*. It is then executed immediately.

2.4.2 Aircraft specific mass model

The aircraft specific mass model is a MATLAB M-function which calculates the mass properties as a function of the aircraft mass. The mass properties consist of the mass itself, the location of the center of gravity and the moments and products of inertia. These values determine a fixed operational condition in the aircraft model. The routine for the mass model is therefore executed during the specification phase of the aircraft configuration in the trimming tool. The M-function is designated by the variable `ac_massmodel`.

The mass properties are calculated from the mass distribution of the aircraft. This distribution can be subdivided into contributions from the aircraft basic empty weight, the amount of fuel and the number of passengers. In general, the amount of fuel and the number of passengers may be derived from the user-specified mass. Using the locations of the fuel tanks and the passenger seats, the mass distribution becomes known and subsequently also the mass properties.

The input argument of the aircraft specific mass model consists of the aircraft mass m.

The output argument of the aircraft specific mass model consists of a single variable containing the mass properties. This variable is available in the MATLAB workspace as `massinit` and has the following representation:

$$\texttt{massinit} = [m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}] \tag{2.43}$$

The mass m is in kg, the location of the center of gravity $(x, y, z)_{cg}$ is in the datum reference frame and the moments and products of inertia I_{xx}, I_{yy}, I_{zz} and I_{xz} are with respect to the body axis frame and the center of gravity.

2.5 Data-files

This section describes the data-files and the variables stored in these data-files. The data-files contain necessary data for running *DASMAT*, aircraft specific data which are required in the aircraft specific sub-models and the results which are generated from the various *DASMAT* tools.
The data-files which contain results have dual application. First, they serve as an interface between the various *DASMAT* tools, i.e. the results generated in one tool are used as condition in another analysis tool. This is primarily the case with the data for the trimmed flight condition. Secondly, the data-files may also serve as depository for further off-line analysis via user-defined commands or applications. For the variables in the data-files generally completely specify the operating conditions.
All data-files have the structure of MAT-files, i.e. a double-precision binary MATLAB format, although they do not have all the default `.mat` file extension.

The various types of data-files are distinguished via their file extensions. In this way *DASMAT* recognizes the data-file and is assured that the correct variables are loaded at a certain point during the execution of an analysis tool. Furthermore, related data-files may share the same name and are still held apart.
The following types of data-files are distinguished:

- `.mat` : data of *DASMAT* package and aircraft specific sub-models
- `.inp` : data of input signals
- `.tri` : data of trimmed flight-condition
- `.lin` : data of linearized model
- `.sim` : data of simulated flight
- `.aer` : data of aerodynamic model parameters

The data-files which contain data for running the *DASMAT* package are maintained in the *DASMAT* directory. The data-files with aircraft specific data are kept in the work-directory. The data-files with generated results are both loaded and saved in the data-directory, i.e. the directory which is designated by the variable `datadir`.

The variables to be stored in the various data-files are automatically managed by *DASMAT*. This is achieved by strings which define the names of the variables. These strings with the corresponding types of data-files are presented in Table 2.2. A description of the variables themselves is given in the next section and Appendix C *Variables used by DASMAT*.

2.6 Variables

The transparent structure of MATLAB/SIMULINK allows the user to take a look at the contents of variables used by *DASMAT*. User-defined commands may be issued on these variables for a more specific analysis of results beyond the scope of the *DASMAT* tools. For this purpose,

a short description of the significant variables is presented in this section.

The generic framework of the *DASMAT* package introduces a number of variables that are used for naming the applied SIMULINK models for the analysis tools and the components of the signals. Furthermore, there are variables which are needed by *DASMAT* to operate, which contain data of the analyzed aircraft and conditions and of course the variables which contain the simulation and analysis results. The variables can therefore be distinguished by the following groups:

- variables which name model-files and model-blocks
- variables which name signal components
- variables which name data-files and stored variables
- variables which configure simulation models and control simulations
- variables which specify controller modes for closed-loop simulation
- variables which contain results from *DASMAT* tools
- variables which specify aircraft geometric parameters and control input limitations
- variables which specify navigation ground station and measurement locations
- variables which specify atmospheric and turbulence constants
- variables which specify aircraft geometry for animation

The most significant variables are listed in Appendix C *Variables used by DASMAT* with their size and the file where they are stored or created. Most of these variables get their values during the initialization phase of *DASMAT* when they are read from a data-file. The names of the variables are as much in correspondence with ISO-1151 [2]

The variables which name the the model-files and model-blocks are used to adapt the generic models to a specific aircraft and condition. The names of the aircraft specific model-files, e.g. the aerodynamic model in `ac_aeromodel`, are specified in the aircraft specification routine designated in `mod_data` and which is supplied by the user.
The variables which name the signal components are used to label the components of the control inputs, state variables and observation outputs of the generic aircraft model. The variables for the observation outputs are subdivided into the groups given in (2.5), where the output labels in `y?lab` of the general aircraft model `ac_mod.m` are distinguished from the output labels in `yred?lab` of the compact aircraft model `ac_modpc.m`. The contents of these variables agree with the format of the signals described in Section 2.3.3 *Generic aircraft model* and are given in Table B.1 to Table B.4. The variables are read from the data-file `ac_genrl.mat` in the *DASMAT* package.
The variables which name the data-files and the stored variables provide exchangeability of data between the *DASMAT* tools. They take care that the right data is stored in the right type of data-file. In this way for example, the results from the trimming tool are correctly saved in a trim-file `.tri`, which may then automatically be read in another tool. Table 2.2 presents these variables and also lists which variables are stored in the various types of data-files.

The variables which configure the simulation models and control the simulations are used as switches. They are set during the execution of the *DASMAT* tools after the user has selected a certain option, e.g. simulation from the SIMULINK window, and thus control which commands are executed. For memory purposes they are read from the data-file `ac_genrl.mat` in the *DASMAT* package and are initially zero by default.

The variables which specify controller modes for closed-loop simulation are used as switches in the controller. They are therefore closely associated to the implemented controller and may even become irrelevant. Nevertheless, by introducing standardized names the controller mode becomes easily detectable and mode switching may be controlled from the default operating shell for closed-loop simulation `cl_sim.m`. For memory purposes the variables are read from the data-file `cl_ctrl.mat` in the *DASMAT* package and are initially -1 by default.

The variables which contain results from the *DASMAT* tools are all user-generated data stored in the various data-files in the data-directory designated by the variable `datadir`. Depending on the *DASMAT* tool and the type of data-file, they consist of signals from trimmed flight-conditions and simulations, system matrices of the linearized aircraft model and polynomial coefficients of the aerodynamic model. The variables in each data-file are presented in Table 2.2. The trimming results are stored in a trim-file `.tri` and are used to define the starting point of a simulation or the analysis point of the linearization and aerodynamic model fitting tool. The simulation results are stored in a simulation-file `.sim` and are used for off-line analysis of the performed simulation. They may be used in the *DASMAT* tools for off-line animation and time-responses plotting tools, or the user may apply them in direct commands from the MATLAB prompt. The linearization results in the linearization-file `.lin`, and aerodynamic fitting results in the aerofit-file `.aer` can only be analyzed and applied via user-defined commands.

The variables which specify aircraft geometric parameters and control input limitations consist of required aircraft and engine specific data for using *DASMAT*. These data should be included using the specified variable names in the aircraft and engine specific data-files designated by the variables `ac_data` and `eng_data`, which in their turn are specified in the aircraft specification routine designated in `mod_data`. Aside from these data, much more data will be required in the aircraft specific sub-models, but these data do not have standardized names and only need be in correspondence with their designation in the sub-models.

The variables which specify navigation ground station and measurement locations are used for calculating observation outputs with respect to these locations. For memory purposes, they are read from the data-file `ac_genrl.mat` in the *DASMAT* package and are zero by default. However, the user may set them to appropriate values in the MATLAB workspace before starting a simulation or linearization.

The variables which specify atmospheric constants contain the values of primary constants for the International Standard Atmosphere. They are taken from [1] and used to calculate atmospheric and airdata parameters. The variables which specify turbulence constants are used for calculating Dryden turbulence spectra. The atmospheric constants are stored in `ac_genrl.mat`, while the turbulence constants are stored in `ac_turb.mat`.

The variables which specify the aircraft geometry are matrices which are used to draw aircraft contours during the animation of an aircraft simulation. They are stored in the data-file `ac_geom.mat` in the *DASMAT* package.

Table 2.1: Format for *Inport/Outport* blocks in SIMULINK models.

Aerodynamic model `ac_aeromodel`

inport			
`x`	x_a	=	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
`u`	u_a	=	$[\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}]$
`cg`	$(x,y,z)_{cg}$	=	$[x_{cg} \;\; y_{cg} \;\; z_{cg}]$
`yad1`	y_{ad_1}	=	$[M \;\; \bar{q}]$
`yad2`	y_{ad_2}	=	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
`gust`	u_g	=	$\left[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \hat{u}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}\right]$
outport			
`Ca`	C_{base}	=	$[C_X \;\; C_Y \;\; C_Z \;\; C_\ell \;\; C_m \;\; C_n]$
`Cadot`	$C_{\dot{\alpha}}$	=	$[C_{X_{\dot{\alpha}}} \;\; C_{Y_{\dot{\alpha}}} \;\; C_{Z_{\dot{\alpha}}} \;\; C_{\ell_{\dot{\alpha}}} \;\; C_{m_{\dot{\alpha}}} \;\; C_{n_{\dot{\alpha}}}]$
`Cbdot`	$C_{\dot{\beta}}$	=	$\left[C_{X_{\dot{\beta}}} \;\; C_{Y_{\dot{\beta}}} \;\; C_{Z_{\dot{\beta}}} \;\; C_{\ell_{\dot{\beta}}} \;\; C_{m_{\dot{\beta}}} \;\; C_{n_{\dot{\beta}}}\right]$
`Cag`	C_g	=	$[C_{X_g} \;\; C_{Y_g} \;\; C_{Z_g} \;\; C_{\ell_g} \;\; C_{m_g} \;\; C_{n_g}]$

Propulsion model `ac_powermodel`

inport			
`x`	x_a	=	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
`ut`	$[u_{t_1} \;\; u_{t_2}]$	=	$\begin{cases} [PLA_1 \;\; PLA_2] & \text{(engine model included)} \\ [T_{N_1} \;\; T_{N_2}] & \text{(engine model not included)} \end{cases}$
`yatm`	y_{atm}	=	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
`yad1`	y_{ad_1}	=	$[M \;\; \bar{q}]$
`yad2`	y_{ad_2}	=	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
outport			
`FMt`	y_{FM_t}	=	$[X_t \;\; Y_t \;\; Z_t \;\; \bar{L}_t \;\; M_t \;\; \bar{N}_t]$

Engine model `eng_statmodel` / `eng_dynmodel` / `eng_none.m`

inport			
`x`	x_a	=	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
`ut`	u_t	=	PLA
`yatm`	y_{atm}	=	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
`yad1`	y_{ad_1}	=	$[M \;\; \bar{q}]$
`yad2`	y_{ad_2}	=	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
outport			
`ypow`	y_{pow}	=	$[T_N \;\; FF \;\; N_{LP} \;\; N_{HP} \;\; N_{LPc} \;\; N_{HPc}$ $WT \;\; ITT \;\; EGT \;\; OILT \;\; OILP]$

Table 2.1: Continue.

Wind model `ac_windmodel` = `wnd_none.m`

inport			
`x`	x_a	=	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
outport			
`wind`	u_w	=	$[u_{w_e} \;\; v_{w_e} \;\; w_{w_e}]$

Turbulence model `ac_turbmodel` = `tur_none.m` / `tur_dryd.m`

inport			
`x`	x_a	=	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
outport			
`gust`	u_g	=	$\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}$

Table 2.2: Strings and specified variables for storing generated results in data-files.

data-file	file ext	string data-file	string variables	variables
trim-file	`.tri`	`trimdata`	`trimvar`	`massinit x0 xt0 u0 ut0 Tn0 Deng x0dot flightcond constraint`
input-file	`.inp`	`inpdata`	`inpvar`	`tsig usig utsig Tnsig`
linearization-file	`.lin`	`lindata`	`linvar`	`massinit x0 u0 Tn0 Deng flightcond constraint Alin Blin Clin Dlin yndx do_PLA ac_model`
simulation-file	`.sim`	`simdata`	`simvar`	`massinit x0 xt0 u0 ut0 Tn0 Deng t r u ut Tn x xt y ypow do_PLA do_simulink do_clsim ac_model ac_simmodel ac_funmodel cl_simmodel`
aerofit-file	`.aer`	`fitdata`	`fitvar`	`sourcedata polycoef aerovar Upoly0 Upoly1 Uaero Yaero fitstat do_fitmodel do_fitgust do_fitaxes`

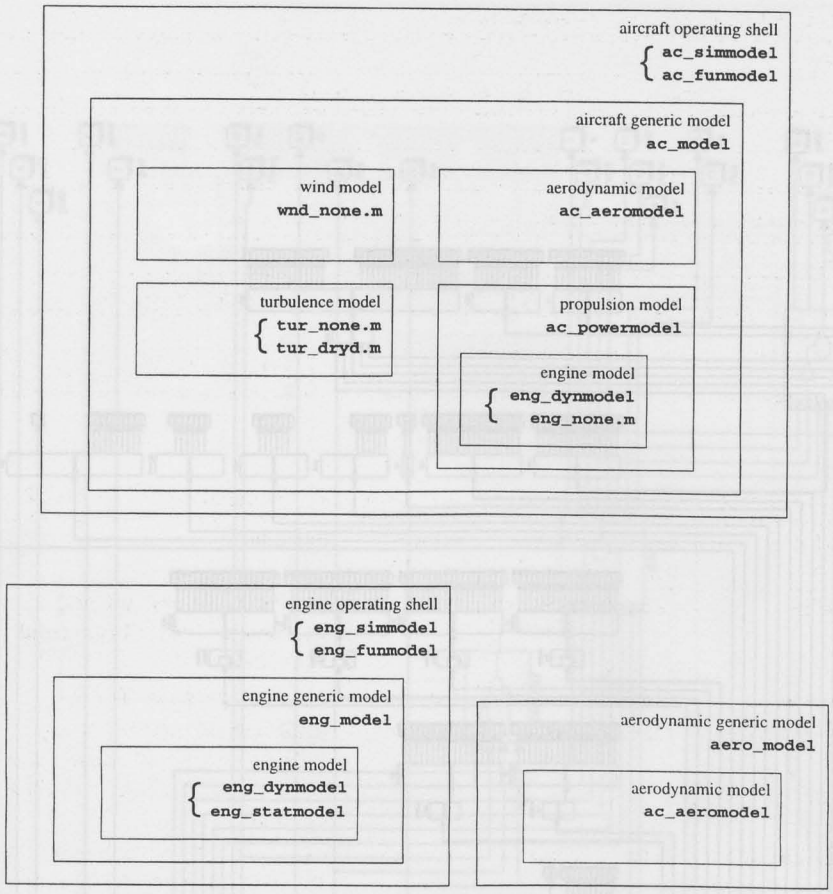


Figure 2.1: Architecture of SIMULINK models and variables designating the SIMULINK models.

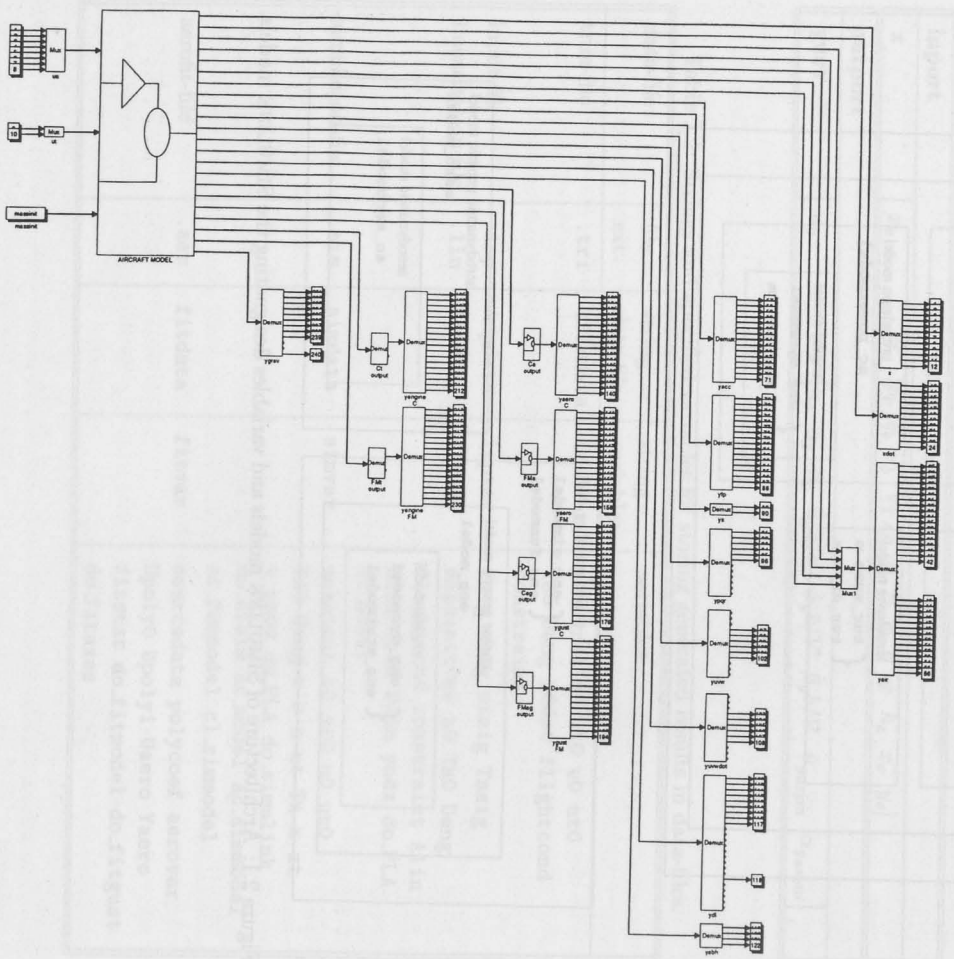


Figure 2.2: SIMULINK S-function `ac_mod.m` and its highest sub-system level of general aircraft model.

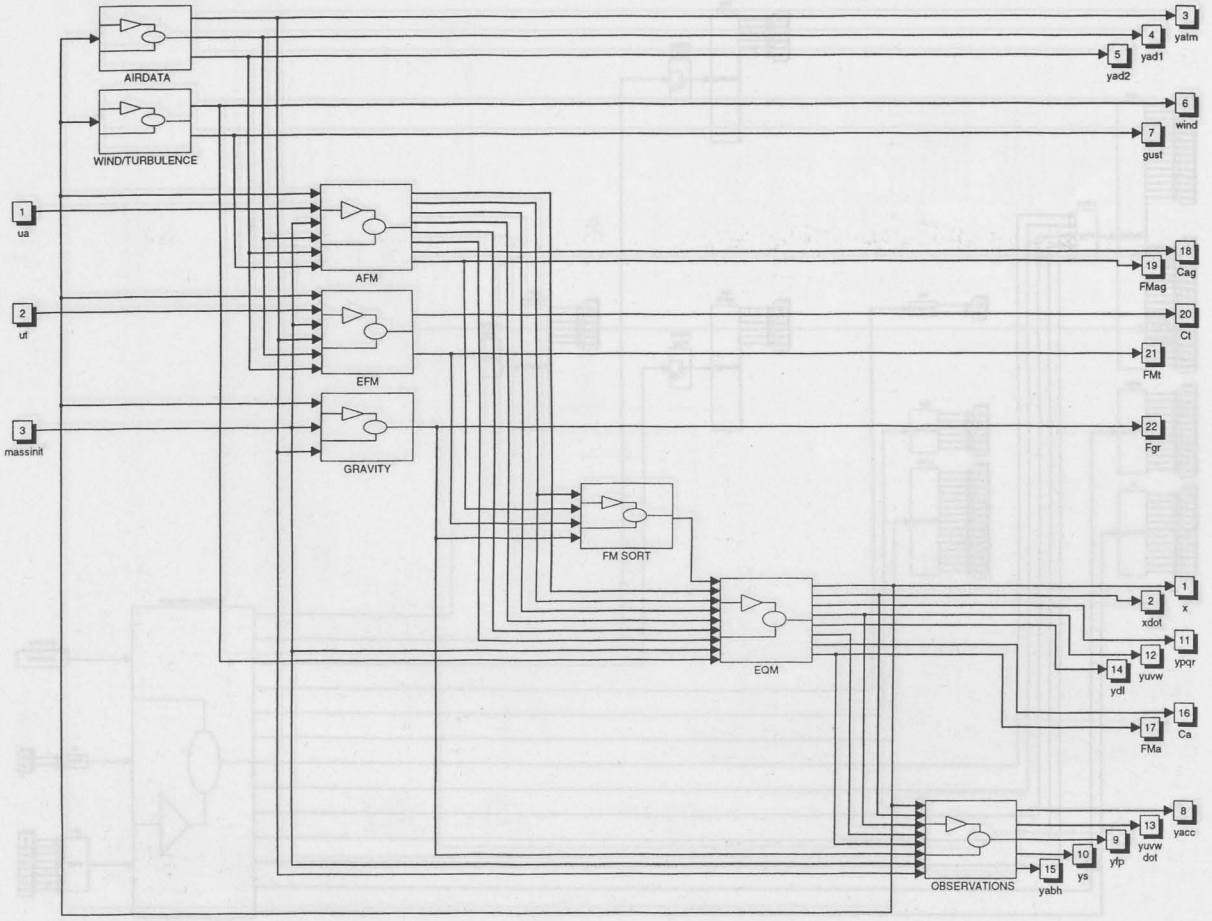


Figure 2.2: Continue.

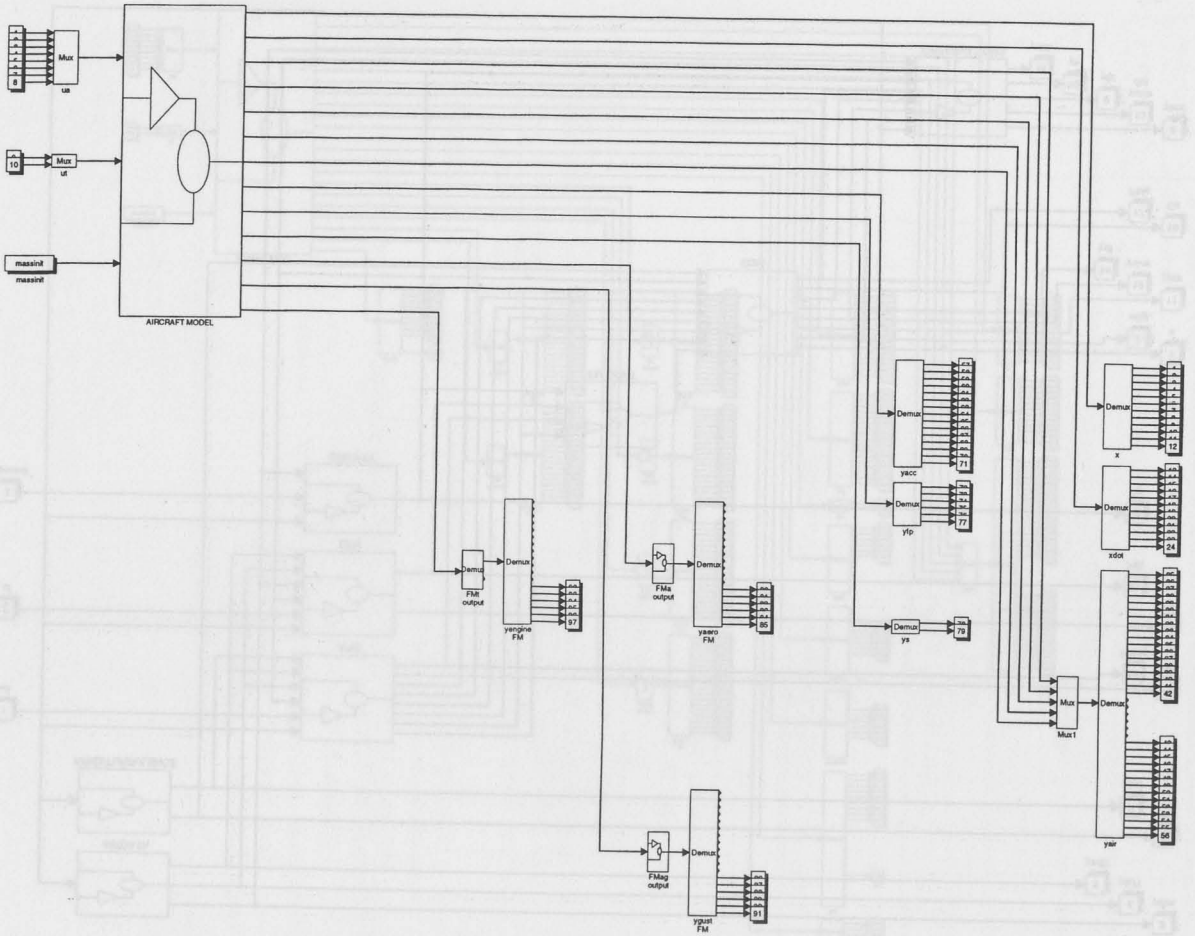


Figure 2.3: SIMULINK S-function `ac_modpc.m` and its highest sub-system level of compact aircraft model.

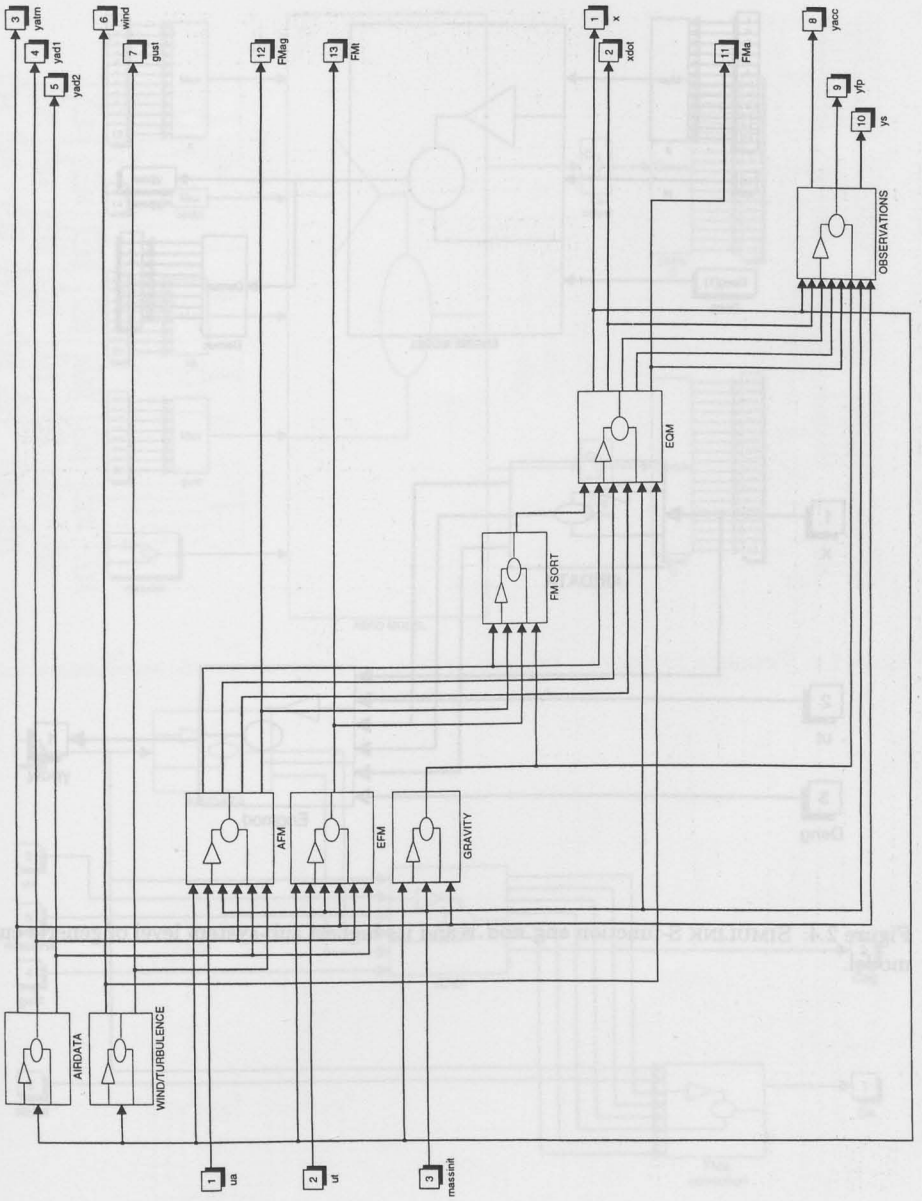


Figure 2.3: Continue.

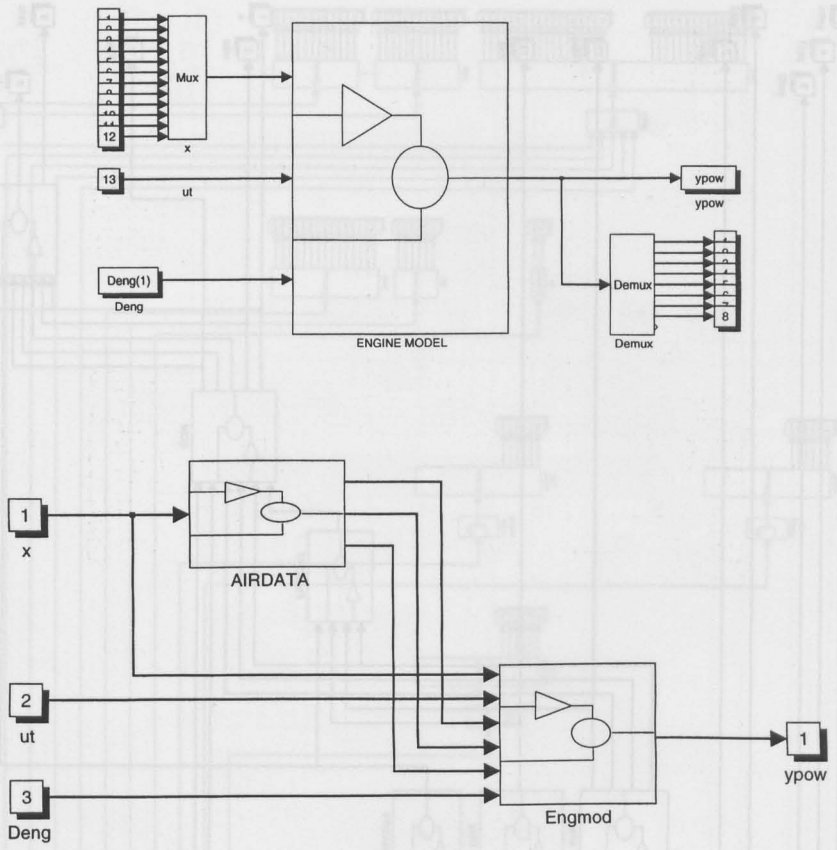


Figure 2.4: SIMULINK S-function `eng_mod.m` and its highest sub-system level of generic engine model.

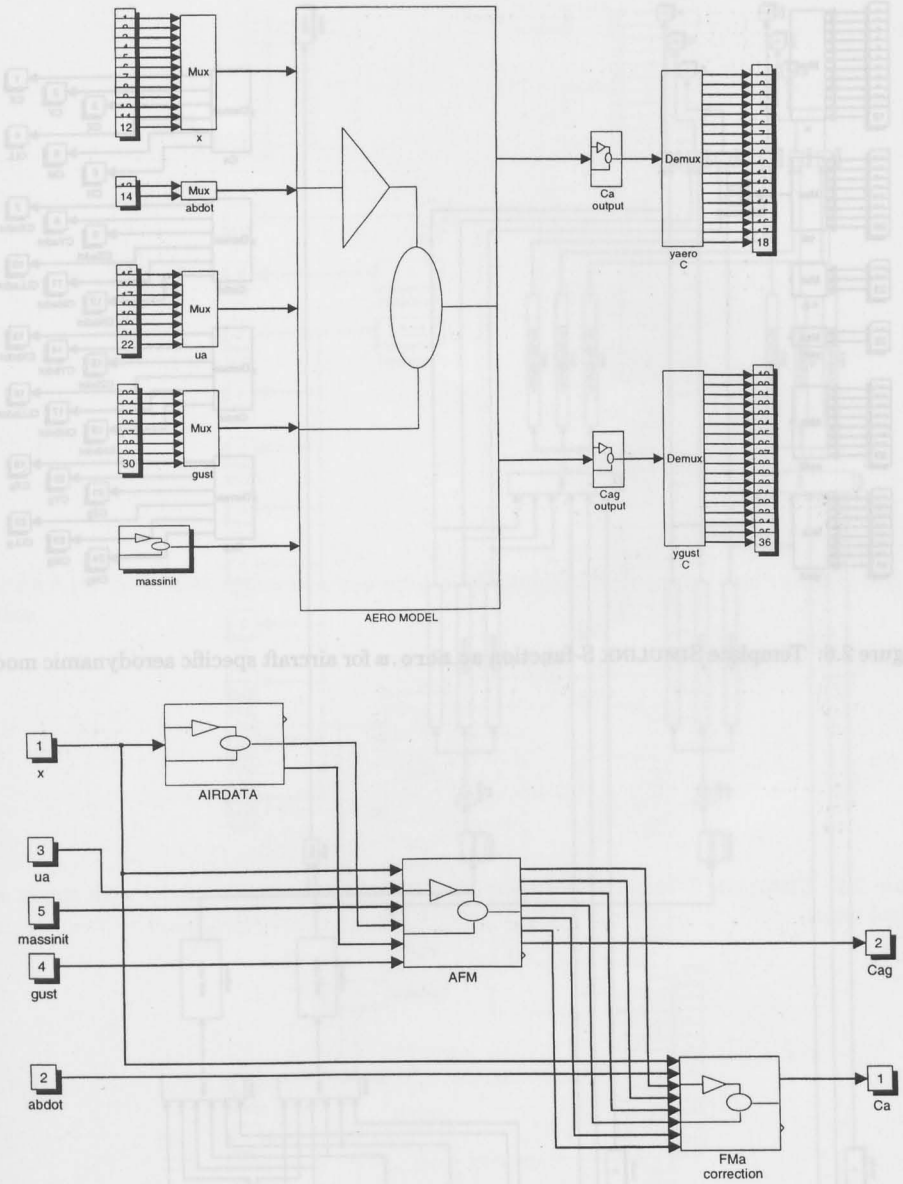


Figure 2.5: SIMULINK S-function `aero_mod.m` and its highest sub-system level of generic aerodynamic model.

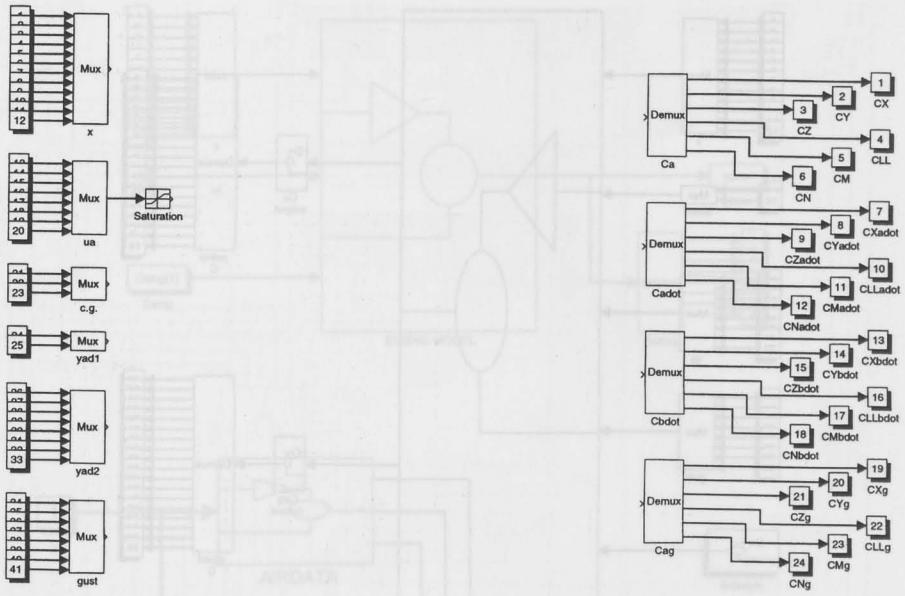


Figure 2.6: Template SIMULINK S-function `ac_aero.m` for aircraft specific aerodynamic model.

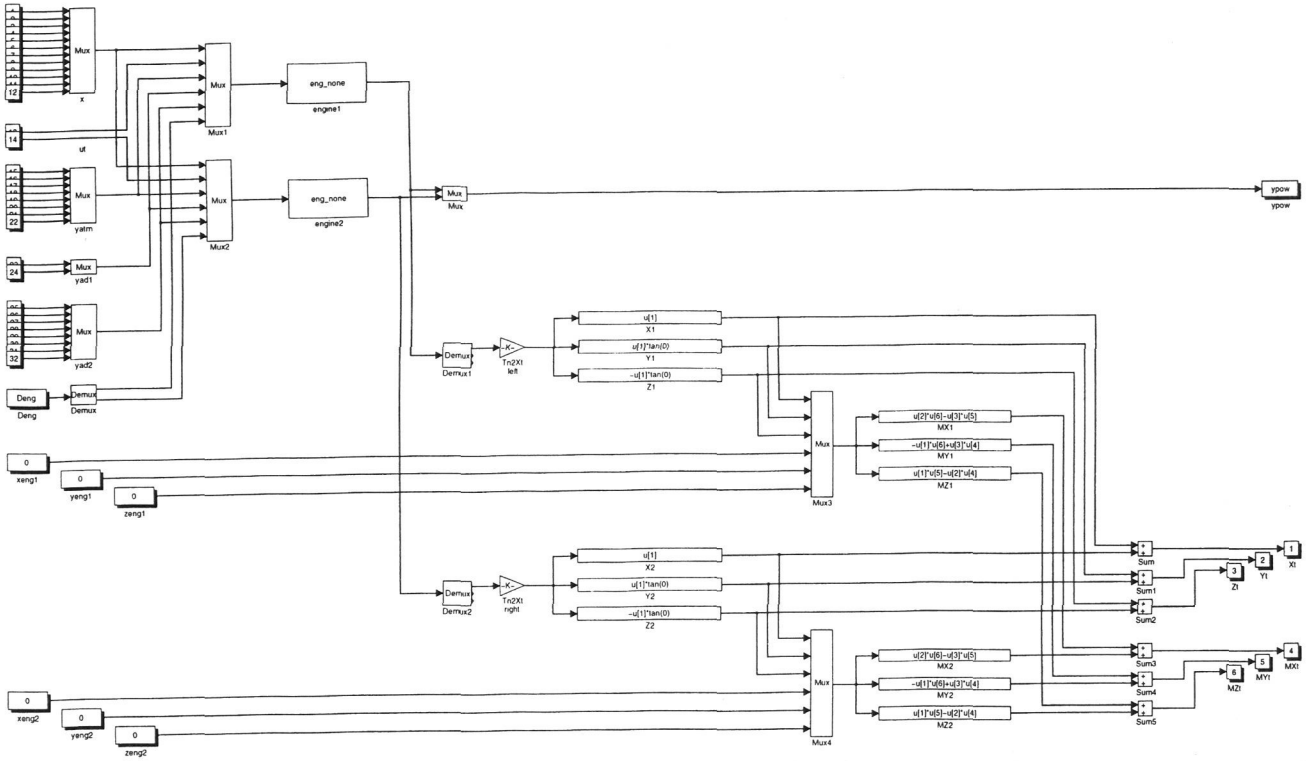


Figure 2.7: Example SIMULINK S-function `ac_pow.m` for propulsion model of a two-engine aircraft.

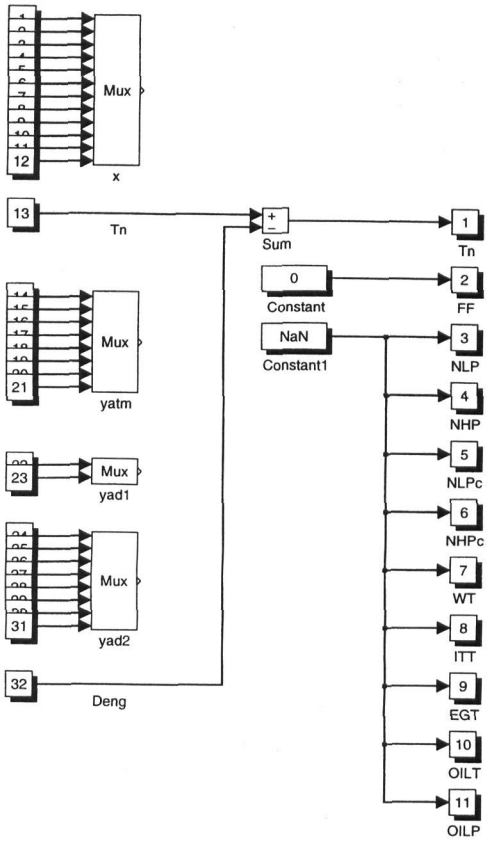


Figure 2.8: SIMULINK S-function `eng_none.m` of feed-through engine model with thrust as control input.

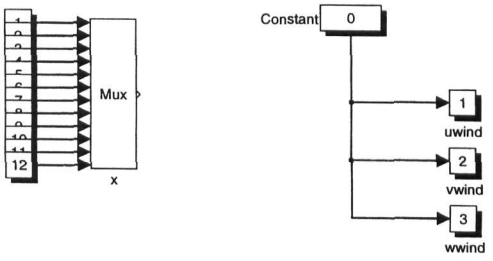


Figure 2.9: SIMULINK S-function `wnd_none.m` of wind model for zero wind condition.

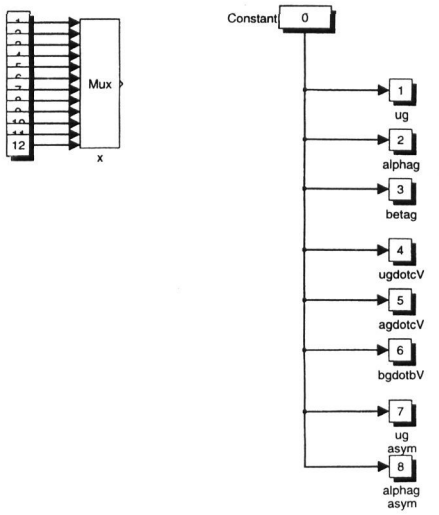


Figure 2.10: SIMULINK S-function `tur_none.m` of turbulence model for zero turbulence condition.

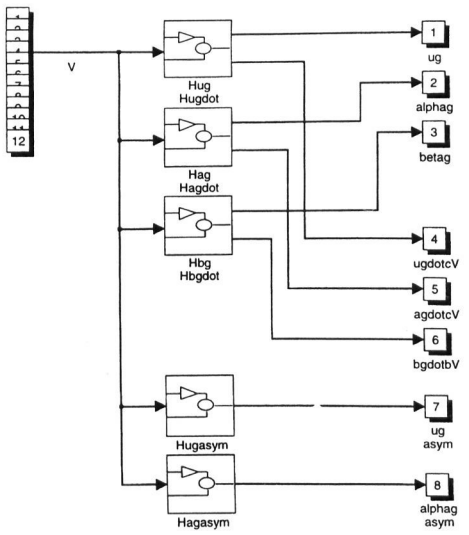
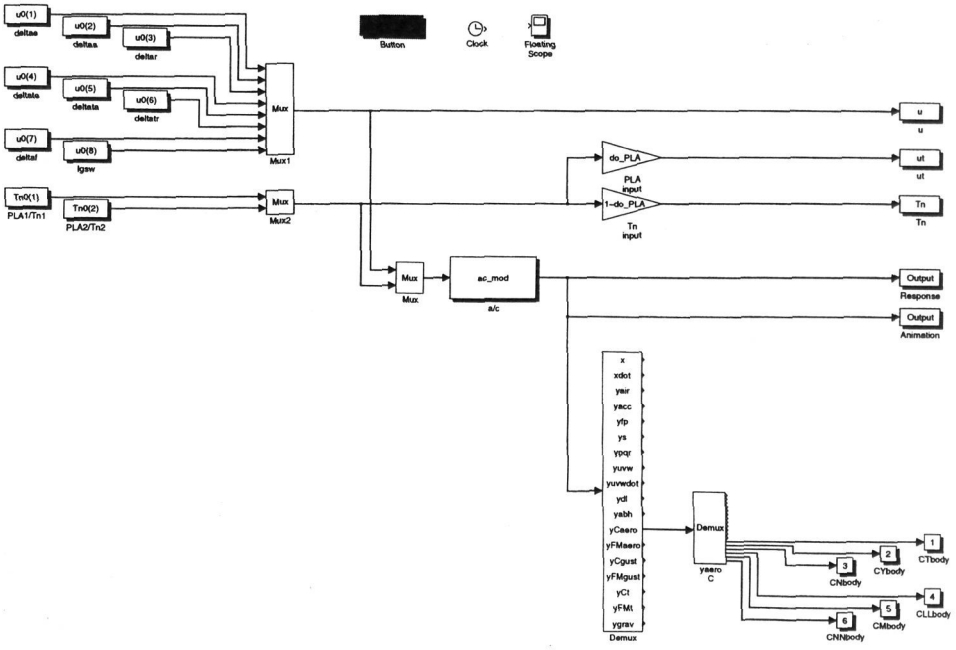
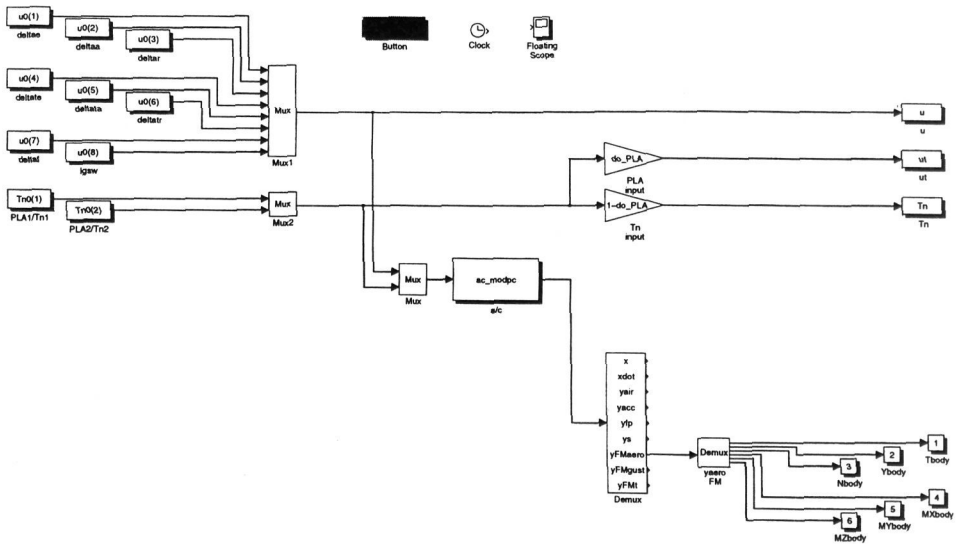


Figure 2.11: SIMULINK S-function `tur_dryd.m` of turbulence model using Dryden spectra.

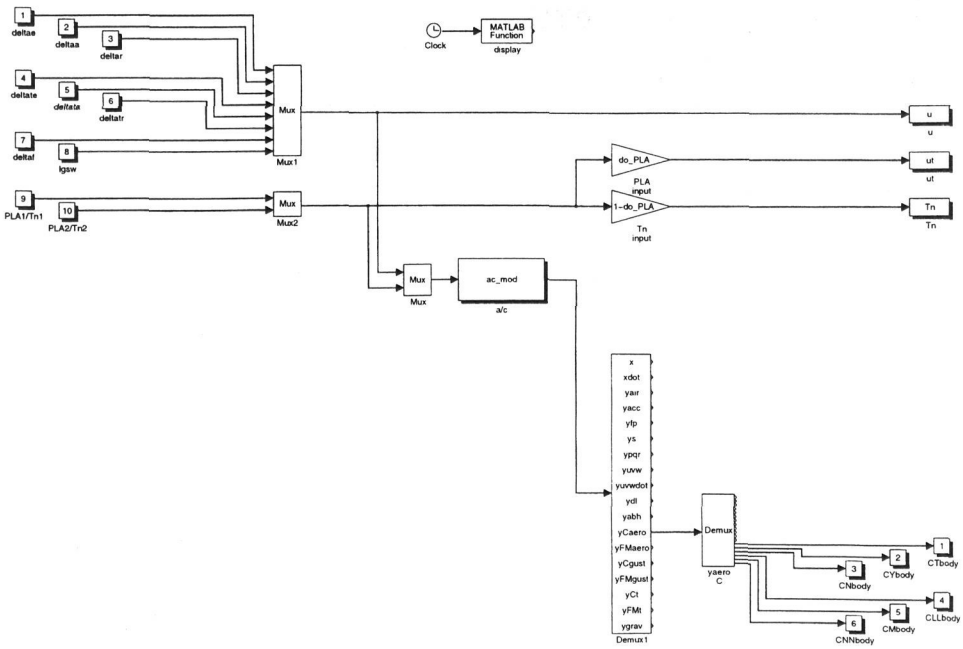


`ac_sim.m` for general aircraft model `ac_mod.m`

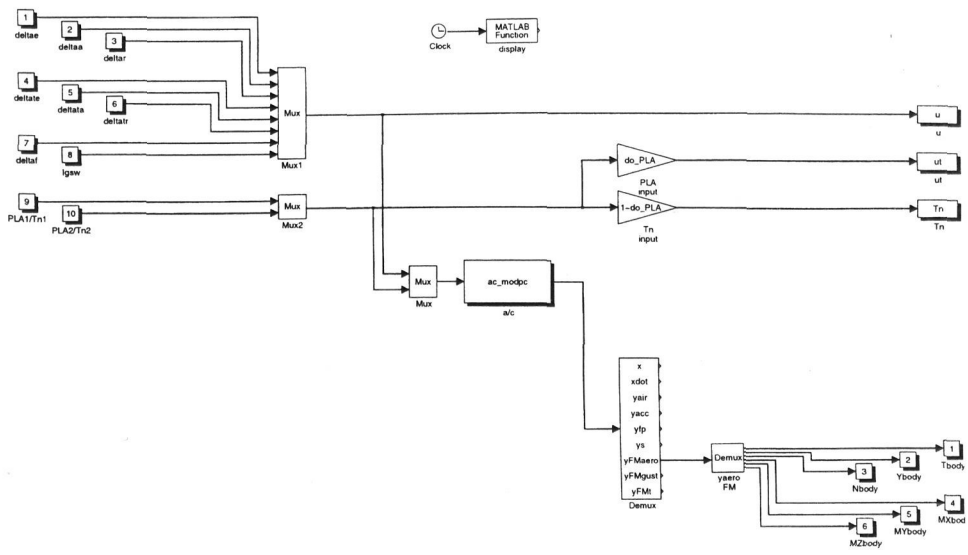


`ac_simpc.m` for compact aircraft model `ac_modpc.m`

Figure 2.12: Default operating shells for open-loop aircraft simulation in SIMULINK window.

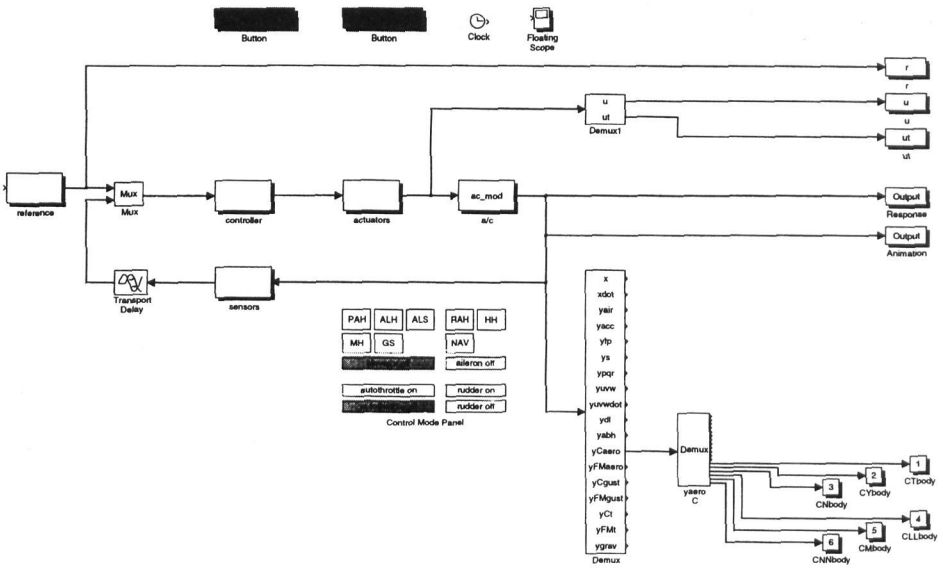


`ac_fun.m` for general aircraft model `ac_mod.m`

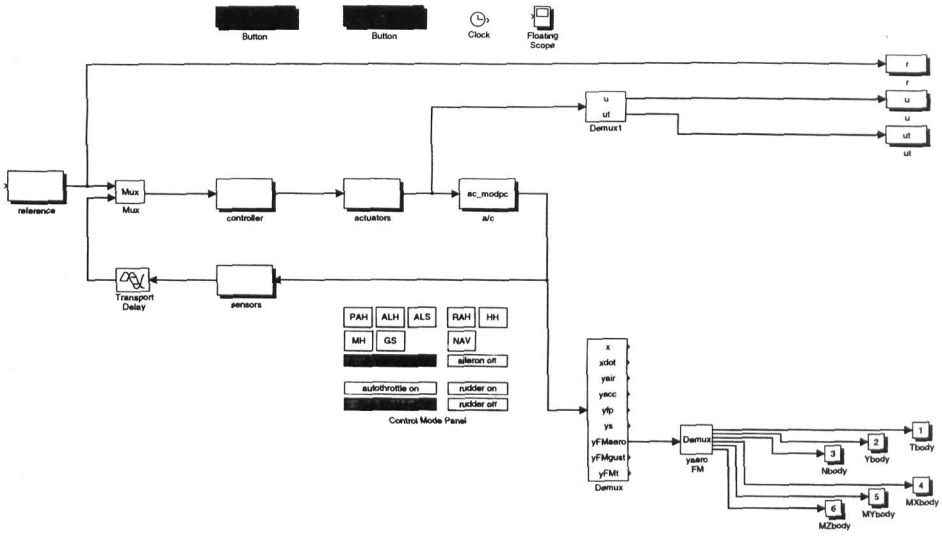


`ac_funpc.m` for compact aircraft model `ac_modpc.m`

Figure 2.13: Default operating shells for open-loop aircraft simulation in command window.

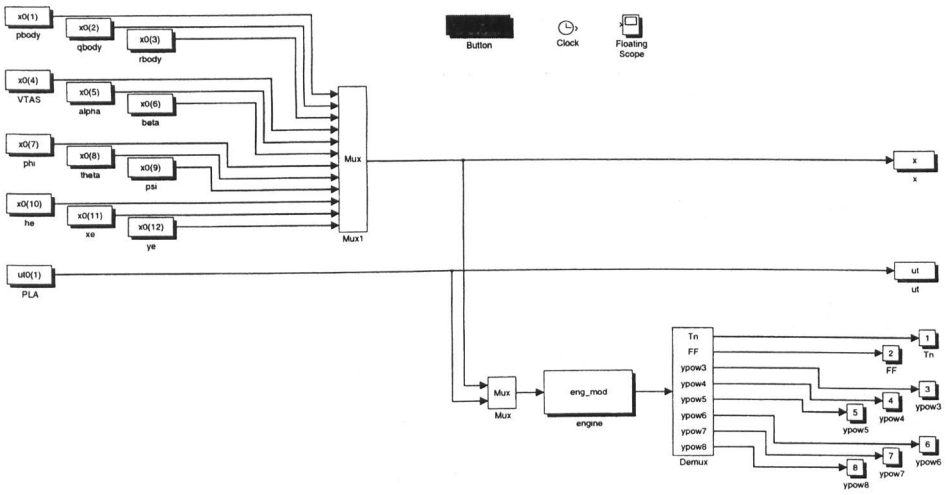


`cl_sim.m` for general aircraft model `ac_mod.m`

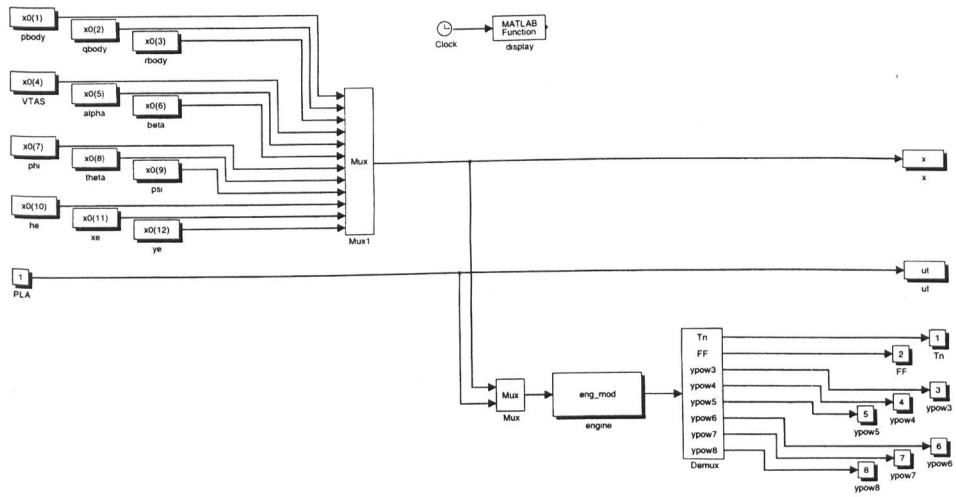


`cl_simpc.m` for compact aircraft model `ac_modpc.m`

Figure 2.14: Default operating shells for closed-loop aircraft simulation in SIMULINK window.



`eng_sim.m` for simulation in SIMULINK window



`eng_fun.m` for simulation in command window

Figure 2.15: Default operating shells for engine simulation in SIMULINK and command window.

```
% 'filename'
% 'date'
% 'author'
% program for defining names of models and data-files for ...
%
% main prog : ac_init.m


% define names
ac_name        ='';
ac_enginename ='';


% aircraft specific data
ac_data          ='';
ac_geometry      ='';
ac_massmodel     ='';
ac_aeromodel     ='';
ac_powermodel    ='';


% engine specific data
eng_data         ='';
eng_statmodel    ='';
eng_dynmodel     ='';
```

Figure 2.16: Basic lay-out of MATLAB M-script for aircraft specification routine designated by `ac_info`.

Chapter 3

Operation

3.1 General operation

The operation of the *DASMAT* package follows a specified general scheme. This chapter discusses general aspects for running a simulation or an analysis tool. The specific operations for each tool will be discussed in individual sections.

The simulation and analysis tools of the *DASMAT* package can be executed in two ways. The general way is via the command `dasmat` from the MATLAB prompt. In this way, a menu window **DASMAT - MAIN MENU** is created which lists all available options, see Figure 3.1. The simulation or the desired tool may be started by clicking the appropriate button with the mouse.
The other way of executing a simulation or a tool is by issuing the appropriate command directly from the MATLAB prompt. The commands are listed in Figure 3.2.

The *DASMAT* package works interactively. When a routine is started, the user is asked for input. The routine stays in a loop until a correct value is entered. *DASMAT* specifies a list of options, boundary values or a default value. For entering the default value, the user may directly press the **return** key. When a data-file needs to be loaded, the available files of the correct type are automatically listed. This also applies when generated results are saved in a data-file, with the addition that the user is asked for confirmation when a file gets overridden.

The simulation and all tools use a flight-condition which is read from a trim-file `.tri`. It is therefore necessary to have these files created from the trim-routine `trim_ac` beforehand.

3.2 Starting and initializing *DASMAT*

To start the *DASMAT* package, MATLAB has to be invoked first. The user should then change directory to the work-directory from which *DASMAT* will be executed, i.e. the directory which contains all aircraft specific sub-models and model data. *DASMAT* may be started by issuing the command `dasmat` from the MATLAB prompt. This displays the menu window **DASMAT - MAIN MENU** in Figure 3.1 from which all simulation and analysis tools are available. The *DASMAT* tools may also be started by issuing their own commands from the MATLAB prompt, as specified in Figure 3.2.

After starting *DASMAT* the user has to specify some initialization information, see Figure 3.3. First, a directory needs to be specified where all data-files are accessible. If the user returns an empty input then the current directory is applicable. The returned directory is stored in the variable `datadir`.
Next, the user has to choose from a general or a compact aircraft-model. The general model refers to the generic aircraft model in the S-function `ac_mod.m`. The compact model refers to the S-function `ac_modpc.m` which is a stripped version of `ac_mod.m`, see Section 2.3.3 *Generic aircraft model.* The model `ac_mod.m` generates the complete list of observation parameters in (2.6) and Table B.3 and is really only suited for workstations. The model `ac_modpc.m` generates with the same formulae only the most significant observation parameters in (2.7) and is more suitable for PC platforms. The name of the applied aircraft-model is stored in the variable `ac_model`. Together with the selection of the aircraft model, the default operating shells for aircraft simulation are specified. For open-loop simulation, the names of the appropriate S-functions are assigned to the variable `ac_simmodel`, i.e. `ac_sim.m` and `ac_simpc.m`, and to the variable `ac_funmodel`, i.e. `ac_fun.m` and `ac_funpc.m`. For closed-loop simulation, the S-function name `cl_sim.m` is assigned to the variable `cl_simmodel`.
Next, the user has to specify the aircraft to be analyzed. An M-file has to be selected which serves as the aircraft specification routine, see Section 2.4.1 *Aircraft specification routine* and Figure 2.16. This file defines the variables which name the S-functions for the aerodynamic model, the propulsion model and engine model, the M-function for the mass model and the data-files which contain the variables used in these models.

The initialization is controlled by the existence of the variable `datadir` in the MATLAB workspace. It is executed at the start of every simulation and analysis tool when `datadir` is not found in the workspace, otherwise it is skipped. Therefore, if the user wants to reconfigure the application of models or aircraft, it is sufficient to clear only this variable from the workspace via the command `clear datadir`.

3.3 Simulation of Aircraft

The aircraft simulation tool provides an open-loop and closed-loop simulation for a specified aircraft. The model takes aerodynamic and thrust controls as inputs and generates a variety of observation outputs using the aircraft model described in Section 2.3.3 *Generic aircraft model.*
The simulation tool offers the user the possibility of configuring the simulation model, the running procedure of the simulation and the display of observation outputs. Various (user-defined) wind and turbulence models may be selected. The engine model may or may not be included in the model, thus using either throttle setting or thrust as control input. For open-loop simulation, the control inputs may be changed on-line during simulation or they may be prespecified off-line as time-histories having user-defined forms. For closed-loop simulation, the systems in the closing loop may be easily replaced with user-specified models of the controller, actuators, sensors and reference signals generator. The dynamic responses of the aircraft may be visualized on-line through a 3D-animation or as time-history plots. They may further be saved in workspace or in data-files for off-line analysis. Finally, the user is offered a great flexibility in defining the control inputs and in selecting the observation outputs.

This section describes the available options for starting the aircraft simulation and running the open-loop aircraft simulation. The follow-up procedure after selecting closed-loop simulation is described in Section 4.4 *Closed-loop simulation in* SIMULINK *window.* A step by step description is given of the user-supplied data, the executed routines and the screen displays. The simulation tool is provided in the MATLAB M-scripts `sim_ac.m`, `sim_ac1.m`, `sim_ac2.m` and `sim_ac3.m`. The last three scripts are used for respectively open-loop simulation in the SIMULINK window, open-loop simulation in the command window and closed-loop simulation in the SIMULINK window. When simulating in the SIMULINK window, the M-functions `ac_ctrl*.m` and `ac_anim*.m` provide the commands for on-line control and on-line animation, and the S-function `ac_resp.m` pops up an array of scope windows for displaying the aircraft states. When simulating in the command window, the M-function `ac_sig.m` provides the generation of standard control input trajectories. The default operating shells can handle both the general and compact aircraft model in the S-functions `ac_mod.m` and `ac_modpc.m` respectively. They are provided by the S-functions `ac_sim.m` (`ac_simpc.m`) and `ac_fun.m` (`ac_funpc.m`) for open-loop simulation in respectively the SIMULINK window and the command window, and by the S-function `cl_sim.m` (`cl_simpc.m`) for closed-loop simulation in the SIMULINK window. The flow diagram of the simulation tool is given in Figure 3.4.

The simulation of the aircraft may either be started by clicking **Simulate 'aircraft'** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package from `dasmat`, or directly by the command `sim_ac` from the MATLAB prompt.
The user may also directly open an operating shell for running the aircraft simulation in the SIMULINK window by entering their names from the MATLAB prompt, i.e. `ac_sim`, `ac_simpc`, `cl_sim` or `cl_simpc`. The aircraft model should then be configured by double clicking the *Button* block *Set Aircraft Model* in the SIMULINK window. However, just the models specified by standardized variable names, see Table C.1, will be implemented in the generic aircraft model. Furthermore, a trimmed flight-condition should already have been loaded in the MATLAB workspace. For the closed-loop operating shells, the models for the controller, actuator, sensors and reference signals should additionally be implemented by double clicking the *Button* block *Set Closed-Loop model* in the SIMULINK window. Again, the models designated by the appropriate variables in Table C.1 will be used. After successful configuration of the aircraft model and the closed-loop model, the clicked *Button* block turns from red into green. The simulation may then be started via the procedure described in Section 3.3.1 *Open-loop simulation in* SIMULINK *window.*

After the simulation tool is started the user is returned to the command window where the display of Figure 3.5 is generated step by step. As a first step the starting point must be set from a flight-condition which is saved in a trim-file `.tri`. If no trim-file is entered, the simulation routine stops and one returns to the MATLAB prompt.
The next input asks for running the open-loop or the closed-loop simulation. When open-loop simulation is selected, the control inputs are provided by the user, either on-line in the SIMULINK window or off-line in the command window. When closed-loop simulation is selected, the aircraft model is included in a feedback loop and the control inputs are generated via a controller.
The next two inputs select the wind and turbulence conditions to be used. The S-functions for these models are described in Section 2.3.9 *Condition specific wind model* and Section 2.3.10

Condition specific turbulence model. If a non-zero wind condition is selected, the user may include an S-function for the wind model which should be created beforehand and kept in the current work-directory. If a non-zero turbulence condition is selected, the S-function `tur_dryd.m` from the *DASMAT* package is included. This model generates gust velocities from white noise using Dryden spectra. When no wind and no turbulence are selected, the S-functions `wnd_none.m` and `tur_none.m` from the *DASMAT* package are included and all wind and gust velocities become zero.

From this point, the dialogs for open-loop and closed loop simulation go separate ways, see Figure 3.4. For open-loop simulation, the user should further specify the control inputs and the running procedure. This will be treated in the remaining of this section and the following sections. For closed-loop simulation, the dialog directly continues with the visualization of the simulation. This will be further treated in Section 3.3.4 *Specification of simulation visualization* and Section 4.4 *Closed-loop simulation in* SIMULINK *window.*

For the open-loop simulation, the user should now enter how the thrust is entered into the aircraft model via the thrust control input. The thrust may be generated via the engine model through a throttle (power lever angle) setting as input, or the thrust may be directly submitted to the aircraft model as input itself. In the first case, the engine model is included in the aircraft model, i.e. the S-function designated by `eng_dynmodel`. In the second case, the engine model is replaced by a feed-through model in the S-function `eng_none.m`. It should be noted that the selection of the type of thrust control influences the order of the aircraft system. If a power lever input is selected, then the aircraft states are augmented with the engine states times the number of engines. With a thrust input, the aircraft model is a 12th order system, i.e. the number of aircraft states only.
Subsequently, the running procedure of the simulation needs to be entered as either from the SIMULINK window or from the command window. Simulating from the SIMULINK window is the most interactive way with the simulation performed on the screen. The command window option performs a more automated simulation. Both options will be discussed in more detail in the next sections.

3.3.1 Open-loop simulation in Simulink window

The simulation in the SIMULINK window lets the user analyze the behaviour of the aircraft system during the simulation. The simulation model and control inputs may be adjusted during the simulation, internal signals may be viewed, simulations may be repeated until the desired result is obtained for future usage. This procedure is most useful for quick look and initial set-up of the simulation model.

The dialog in the command window which started with the display in Figure 3.5 is continued with the display in Figure 3.6. First, the user may select the option for on-line simulation. In that case, an **On-line Control** window is opened with sliders for controlling the main flight-controls: elevator, aileron, rudder and power lever/thrust, see Figure 3.7. The slider bars may be moved between the minimum and maximum limits via pressing the mouse button over the bar, or they may be set at their values for trimmed condition via clicking **set trim condition**. Via the sliders, the user has direct access to the corresponding blocks with control inputs in the operating shell. Setting the slider bars directly affects the input signals to the

aircraft model and thus provides an on-line control.
Next, the user should select how the responses should be visualized during the simulation. The responses may be viewed as animation, time-responses or as model-activity. The option for animation provides a 3-dimensional picture of the aircraft motion along the flight-path in an opened **Animation** window. The options for time-responses and model-activity display the time-traces of all aircraft states or any block-connection in the SIMULINK window respectively by means of scope windows. More details are given in Section 3.3.4 *Specification of simulation visualization.*
The following question lets the user specify the operating shell for the simulation. The default operating shell is either the S-function `ac_sim.m` or `ac_simpc.m`, see Section 2.3.11 *Operating shells for aircraft simulation* and Figure 2.12, depending on whether the general or compact aircraft model was selecting during initialization. If desired, the default operating shell may be adjusted just before a simulation is started, see below. Instead, a user-defined operating shell may be selected from the current work-directory. However, the user should be sure that its data format agrees with that of the included aircraft model, see also the remarks in Section 2.3.11. The operational operating shell is designated in the variable `ac_simmodel`.
The simulation parameters are specified next. These are the stop time, which is actually the duration of the simulation, and the minimum and maximum step sizes of the integration. The step size itself may vary per integration step and is controlled by the relative error of the integration at each step (default 10^{-3}) which on its turn depends on the operational integration method (default fifth order Runga-Kutta `rk45`). To obtain a fixed step size the maximum step size should be set equal to the minimum step size.

The aircraft simulation model is now configured, loaded in memory and appropriate windows are opened. This may take some time. Depending on the selected visualization method, the SIMULINK window with the name of the operating shell (default **ac_sim** or **ac_simpc** from Figure 2.12) is displayed on the whole screen or reduced to the upper part of the screen. Furthermore, a clock window and scope windows are opened. The window **Clock** continuously displays the elapsed time as the simulation progresses. The scope windows, either one scope for each state variable or a window **Floating Scope** for an arbitrary block-connection, display the activity of those signals or connection. If the on-line simulation is selected, then the **On-line Control** window from Figure 3.7 is also opened. If the simulation responses are shown as animation, then the **Animation** is opened too. Representative screen lay-outs are shown in Figure 3.8.

From this moment, the MATLAB routine is halted and the simulation may be started. The routine only continues after activating the command window and pressing any key. This allows the user to run several simulations, using different control inputs and/or adjusting the operating shell.
Before starting the simulation the operating shell in the SIMULINK window of Figure 2.12 may be adjusted first. The parameters in the various blocks may be given new values. The sources of the control inputs may be changed by replacing their blocks with blocks from the simulink/Sources library, e.g. *From File* or *From Workspace*. (Remember that on-line simulation may then not be possible anymore.) The selection of observation outputs may be modified by extracting other signals from the *Demux* block. The observation outputs may also made accessible in a different way by using blocks from the simulink/Sinks library as for example *To File* and *To Workspace*.

The simulation is started by selecting **Start** from the **Simulation** menu on top of the SIMULINK window. If desired, the simulation parameters may be adjusted first. They are accessible in the Control Panel dialog box which is displayed by selecting **Parameters** from the **Simulation** menu.
During the simulation, the user is allowed to interactively perform the following operations:

- change the parameters of blocks with control inputs; this may be done directly in the SIMULINK window or via the slider bars in the **On-line Control** window.
- change any of the simulation parameters or the simulation algorithm in the Control Panel dialog box.
- change the view condition in the **Animation** window when the simulation responses are shown as animation, see Section 3.3.4.
- suspend and restart the simulation via the items **Pause** and **Restart** in the **Simulation** menu; during suspension, lines or blocks in the operating shell may be added or deleted.
- stop the simulation via the item **Stop** in the **Simulation** menu.

The simulation results include the vector of the integration time points t and the time trajectories of the input controls, state variables and observations outputs:

$$
\begin{array}{lll}
\text{input controls} & \left\{ \begin{array}{lcll} u & = & \texttt{u} & \text{(aerodynamic controls)} \\ PLA & = & \texttt{ut} & \text{(thrust controls, nonzero if engine model included)} \\ T_N & = & \texttt{Tn} & \text{(thrust controls, nonzero if engine model not included)} \end{array} \right. \\
\text{state variables} & \left\{ \begin{array}{lcll} x & = & \texttt{xtot} & \text{(all model states)} \\ x_a & = & \texttt{x} & \text{(aircraft states)} \\ x_t & = & \texttt{xt} & \text{(engine states, empty if engine model not included)} \end{array} \right. \\
\text{observation outputs} & \left\{ \begin{array}{lcll} y & = & \texttt{y} & \text{(selected observations in operating shell)} \\ y_{pow} & = & \texttt{ypow} & \text{(engine parameters of all engines in propulsion model)} \end{array} \right.
\end{array}
\tag{3.1}
$$

These results are overwritten each time a simulation is finished. They further only become available in the workspace when the simulation is terminated, if desired via the item **Stop** in the **Simulation** menu, but not when a simulation is suspended.

When the MATLAB routine is resumed, the dialog in the command window continues with the request for saving the simulation results in a simulation-file `.sim`, see Figure 3.6. The simulation results refer to the signals in (3.1) and the operating conditions. It should be noted however that these results should be available from the workspace. The saved variables are specified in the variable `simvar`, see also Section 2.5 *Data-files* and Table 2.2:

```
simvar = massinit x0 xt0 u0 ut0 Tn0 Deng t r u ut Tn x xt
         y ypow do_PLA do_simulink do_clsim
         ac_model ac_simmodel ac_funmodel cl_simmodel
```
(3.2)

The variable list includes the mass properties, all initial conditions, the time vector, the trajectories of all state variables, any reference signals, control inputs and selected observation outputs, the switches for the inclusion of the engine model, the selections of simulation running procedure, the applied aircraft model and operating shell. The simulation results can be used for off-line analysis, manually from the command window or via the tools for plotting time-responses `plot_ac` or showing animation `show_ac`.

3.3.2 Open-loop simulation in command window

If the simulation is executed from the command window, the simulation runs in the background. The simulation model and the control inputs are generated before starting the simulation and are not adjustable anymore. The user cannot interfere during the simulation and the responses only become available after the simulation has stopped.

The dialog in the command window which started with the display in Figure 3.5 is now continued with the display in Figure 3.9. The first question lets the user specify the operating shell for the simulation. The default operating shell is either the S-function `ac_fun.m` or `ac_funpc.m` in Figure 2.13. The operating shell does not appear on the screen and is therefore not adjustable anymore before starting the simulation. Instead, the user may create an operating shell before starting this simulation routine, store it in the current work-directory and load it here. Remarks for creating an operating shell are given in Section 2.3.11 *Operating shells for aircraft simulation.* The operational operating shell is designated in the variable `ac_funmodel`.

Next, the time traces of the control inputs are specified. They may be loaded from an input-file `.inp` or they may created interactively, see Section 3.3.3 *Specification of control inputs.* The generated control inputs are treated as perturbations on the trimmed values for the flight-condition in the trim-file `.tri`. There is no check on whether the absolute amplitudes of the signals satisfy the constraints set in the aircraft model. This may lead to saturation of the control inputs during the simulation.

The simulation parameters are now specified. The stop time or duration of the the simulation is derived from the time-length of the control inputs, but it may be adjusted by the user. The minimum and maximum step sizes of the integration are used to constrain the step size that is taken for the error control. The relative error of the integration at each step is set at 10^{-3} and the applied integration method is the fifth order Runga-Kutta `rk45`.

The aircraft simulation model is now configured and loaded in memory. When this is finished the simulation may be started by pressing any key. During the simulation the elapsed time is displayed via a counter as the simulation progresses. The simulation cannot not be prematurely stopped without loosing the simulation results.

The simulation results are written to the MATLAB workspace. They include the vector of the integration time points t and the time trajectories of the input controls, state variables and observations outputs:

$$
\begin{array}{lll}
\text{input controls} & \left\{ \begin{array}{lcll} u & = & \texttt{u} & \text{(aerodynamic controls)} \\ PLA & = & \texttt{ut} & \text{(thrust controls, nonzero if engine model included)} \\ T_N & = & \texttt{Tn} & \text{(thrust controls, nonzero if engine model not included)} \end{array} \right. \\
\text{state variables} & \left\{ \begin{array}{lcll} x & = & \texttt{x} & \text{(aircraft states)} \\ x_t & = & \texttt{xt} & \text{(engine states, empty if engine model not included)} \end{array} \right. \\
\text{observation outputs} & \left\{ \begin{array}{lcll} y & = & \texttt{y} & \text{(selected observations in operating shell)} \\ y_{pow} & = & \texttt{ypow} & \text{(engine parameters of all engines in propulsion model)} \end{array} \right.
\end{array}
\tag{3.3}
$$

The trajectories of the control inputs are linearly interpolated from their specified time traces before starting the simulation. During the simulation, additional values are obtained with respect to the step size at each integration step.

When the simulation is terminated, the dialog in the command window continues with the requests for saving the control inputs and simulation results in data-files, see Figure 3.9. The control inputs refer to the specified time traces before the simulation and are saved in an input-file `.inp`. The saved variables are specified in the variable `inpvar`, see also Section 2.5 *Data-files* and Table 2.2:

$$
\texttt{inpvar} \;=\; \texttt{tsig usig utsig Tnsig} \tag{3.4}
$$

The variable list includes the vector with time points at which the control inputs get changed and the matrices with the amplitudes of the control inputs. These data may best be used for running an identical simulation at a different flight-condition.
The simulation results refer to the signals in (3.3) and the operating conditions, just as for the simulation in the SIMULINK window in the previous section. The variable list is the same, see (3.2), and the variables are also saved in a simulation-file `.sim`.

3.3.3 Specification of control inputs

When the simulation is executed from the command window, the user may specify the control inputs interactively. It is also possible to independently create the control inputs and store them in an input-file `.inp` via the command `inp_ac` from the MATLAB prompt.
The control inputs may be automatically generated from five standard types of signals or they may be specified by the user as a function of a time vector. All generated signals are used as perturbations from the trimmed values obtained from a trim-file `.tri` and may thus be applied for different flight-conditions. However, as a consequence it is not possible to check whether the input signals do not exceed the physical limitations of the controls. They may therefore be saturated when submitting them into the aircraft model.

The control inputs are generated through a dialog in the command window as in Figure 3.10. The first screen is only displayed when the command `inp_ac` is executed. It asks the user, whether the thrust control should be provided as throttle (power lever angle) setting or as

the thrust itself. When the inputs are generated during the execution of the simulation tool, this selection has already been made.
Next, the nonzero signals which are to be specified are selected. Although all control inputs are displayed, only the primary flight controls δ_e = `delta_e`, δ_a = `delta_a`, δ_r = `delta_r` and the thrust control can be selected. The thrust control is either PLA = `PLA` or T_N = `Tn` and applies to each engine. If no signal is selected, one returns to the question for loading control inputs, or to the MATLAB prompt when running `inp_ac`.
The time-length of the signals T has to be entered next. All signals are extended to this time-length which is also used as the default duration of the simulation.

Now, each selected nonzero signal has to be separately specified in detail. The following types of signals are available, see Figure 3.10:

- doublet
- 3211
- step
- pulse
- sine
- zero signal
- user-defined

For each signal the start time t_0 ($0 \leq t_0 < T$), the end time t_1 ($t_0 < t_1 \leq T$) and the maximum amplitude should be successively specified. However, a zero signal is automatically set to a sequence of zeros and a step signal automatically gets the end time $t_1 = T$. For a sine signal, the frequency has to be entered additionally. The end time t_1 is then reduced such that $t_1 - t_0$ spans a complete number of periods. For a user-defined signal, both a sequence of time points between t_0 and t_1 and the normalized amplitude at these time points needs to be specified. As illustration, the signals created in Figure 3.10 are shown in Figure 3.11.

The generated control inputs become available in the MATLAB workspace via the variables in (3.4). When running the simulation tool `sim_ac`, they may saved in an input-file `.inp` after a simulation. When running the command `inp_ac`, they may be saved directly after they are generated.

3.3.4 Specification of simulation visualization

When the simulation is executed in the SIMULINK window, the responses may be on-line visualized as animation, time-responses or as model-activity. The selection of the visualization mode is controlled via the dialogs during the execution of the simulation tool. After the simulation has been started from the **Simulation** menu on top of the SIMULINK window, a further specification may be required. The details of the three visualization modes and any additional dialogs will be explained in this section.

When the responses are visualized via animation, a 3-dimensional picture of the aircraft is displayed along the flight-path from a chosen viewpoint, together with some important flight-data. The view condition may be selected from three modes, i.e. attitude, vertical position and flight-path. Depending on the selected mode, the view parameters should be entered through a dialog in the command window as in Figure 3.12. This dialog also applies to the off-line animation which is executed via the animation tool `show_ac`, see Section 3.9 *Animation of aircraft simulation.*

The attitude mode follows the origin of the datum reference frame during the flight, see Figure 3.13. The aircraft figure remains in the center of the plot while the scales of the axes are adjusted. The obtained view may be imagined as if the user is in a second aircraft which has a constant position with reference to the viewed aircraft and which thus flies the same manoeuvre. This mode is best used to analyze the aircrafts attitudes, e.g. the pitch angle or roll angle.

The vertical position mode also follows the origin of the datum reference frame, but the viewpoint now remains at a constant altitude. The aircraft figure continuously stays in the horizontal center of the plot but it can translate vertically. The axes for the horizontal aircraft position are continuously adjusted while the altitude axis is fixed between user-specified values. The obtained view may be imagined as if the user follows the viewed aircraft from a second aircraft which flies the same manoeuvres in the horizontal plane. This mode is best used to analyze the aircrafts symmetric responses, for example the responses of the phugoid.

The flight-path mode views the aircraft from a fixed position in the fixed earth reference frame, see Figure 3.13. The aircraft follows its flight-path and moves through the plot. The flight-path is represented by a line in the 3D-space and by its projection on the horizontal base plane. All axes are fixed and set by user-specified minimum and maximum values. A useful application of this mode is the visualization of the tracking error during an ILS-approach or the effects of wind on the flight-path.

When the flight-path mode is selected, the user has the option to retain the aircraft figures at previous time-points, see Figure 3.13. This allows the user a good analysis in the progress aircraft position and attitude. This option is actually only useful with the animation tool `show_ac` where subsequent frames may have enough time-spacing for preventing any disturbing overlap of the aircraft figures.

After any input of axes limits, the user needs to enter a scale factor and viewing angles. The scale factor determines the size of the aircraft figure in the plots. The figure is scaled via the span and the limits of the earth Y-axis, where the default figure size is scaled to a span which is half the length of the Y-axis. The view angles specify the location of the observer with respect to the plot via the azimuth angle (rotation around the negative earth Z-axis starting from the earth Y-axis) and elevation angle (rotation around a line in the earth XY-plane and positive upwards). The default viewing angles are 40° in azimuth and 20° in elevation, which shows the aircraft from the front-right-top.

After the view parameters have been entered, the simulation actually begins and the aircraft figure moves through the plot. During the simulation, both the **Animation** window may be resized and the view conditions may be altered. The window is resized by dragging a corner of the window. The axes get automatically adjusted to the new window size. The view conditions are altered by clicking **set view condition** in the **Animation** window. The simulation then suspends and the dialog in Figure 3.12 appears in the command window. After the new view parameters have entered, the simulation resumes using the new view condition. These both adjustments are not available when executing the animation tool `show_ac`.

When the responses are shown as time-responses, the time-traces of each component of the 12-dimensional aircraft state vector are displayed in an array of scope windows, see Figure 3.14. The scope windows cover the whole screen and show the behaviour of the aircraft states simultaneously by means of oscilloscopes. The horizontal and vertical ranges may be adjusted on-line.

When the model-activity is selected, the graphical block diagram of the SIMULINK model is displayed together with a scope window. The activity of any block-connection may now be displayed in the **Floating Scope** window by clicking that line with the mouse. Figure 3.8 shows a representative screen lay-out for this visualization mode.

3.4 Simulation of Engine

The engine simulation tool provides an open-loop simulation for a specified engine. The model takes the throttle (power lever) setting as input and generates observation outputs of at least the thrust and fuel flow, using the engine model described in Section 2.3.4 *Generic engine model.*
The simulation tool offers the user a quick analysis facility of the engine model alone. Both a static and a dynamic version of the engine model may be simulated. As with the aircraft simulation, the throttle control input may be changed on-line during simulation or it may be prespecified off-line as time-historie having a user-defined form. The operating condition, specified via the aircraft state variables, however remains constant. The performance of the engine may be visualized on-line through a time-history plots.

This section describes the available options for running the engine simulation. The procedure merely follows the same steps as with the open-loop aircraft simulation. A step by step description is given of the user-supplied data, the executed routines and the screen displays. The simulation tool is provided in the MATLAB M-scripts `sim_eng.m`, `sim_eng1.m` and `sim_eng2.m`. The last two scripts are used for respectively simulation in the SIMULINK window and the command window. When simulating in the SIMULINK window, the M-functions `ac_ctrl*.m` provide the commands for on-line control. When simulating in the command window, the M-function `ac_sig.m` provides the generation of standard control input trajectories. In both cases, only the thrust control input will of course be effective. The default operating shells are provided by the S-functions `eng_sim.m` and `eng_fun.m` for simulation in respectively the SIMULINK window and the command window. They both call the generic engine model `eng_mod.m` which can handle both the dynamic and static engine model. These models are user-supplied S-functies designated by the variables `eng_dynmodel` and `eng_statmodel`. The flow diagram of the simulation tool is given in Figure 3.15.

The simulation of the engine may either be started by clicking **Simulate 'engine'** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package from `dasmat`, or directly by the command `sim_eng` from the MATLAB prompt.
The user may also directly open the operating shell for running the engine simulation in the SIMULINK window by entering `eng_sim` from the MATLAB prompt. The engine model should then be included by double clicking the *Button* block *Set Engine Model* in the SIMULINK

window. However, depending on the value of the variable `do_engdyn` either the dynamic or static engine model, i.e. the S-function designated by `eng_dynmodel` or `eng_statmodel`, will be implemented in the generic engine model. Furthermore, a trimmed flight-condition should already have been loaded in the MATLAB workspace. After successful implementation of the engine model, the *Button* block turns from red into green. The simulation may then be started via the procedure described below.

After the simulation tool is started the user is returned to the command window where the display of Figure 3.16 is generated step by step. As a first step the starting point must be set from a flight-condition which is saved in a trim-file `.tri`. If no trim-file is entered, the simulation routine stops and one returns to the MATLAB prompt.
The next input asks for running the simulation of the static or dynamic engine model. When the static engine is simulated, the S-function designated by `eng_statmodel` is included. Otherwise, the S-function named by `eng_dynmodel` will be included. Both engine models have in general the same sets of outputs, but the static model has no engine states, see also Section 2.3.8 *Aircraft specific engine model.*
Now, the running procedure of the simulation needs to be entered as either from the SIMULINK window or from the command window. Again, simulating from the SIMULINK window is again the most interactive way with the simulation performed on the screen, while the command window option performs a more automated simulation. Both options will be discussed in the next sections.

3.4.1 Simulation in Simulink window

When simulating in the SIMULINK window, the user may select the option of on-line simulation. The **On-line Control** window which will then open is exactly the same as for the aircraft simulation, see Figure 3.7. Although all sliders may be manipulated, only the slider for the power lever angle is effective for controlling the simulation. Moving the sliders for the main flight-controls has no effect. The trimmed condition may again be restored by clicking **set trim condition**.
The simulation parameters are specified next. These are the stop time, which is actually the duration of the simulation, and the minimum and maximum step sizes of the integration. The step size itself may vary per integration step and is controlled by the relative error of the integration at each step (default 10^{-3}) which on its turn depends on the operational integration method (default fifth order Runga-Kutta `rk45`). To obtain a fixed step size the maximum step size should be set equal to the minimum step size.

The selected static or dynamic engine model is now loaded in memory, the SIMULINK window **eng_sim** is diplayed on the whole screen and and appropriate windows are opened. This may take some time. The window **Clock** continuously displays the elapsed time as the simulation progresses. The window **Floating Scope** displays the activity of any clicked connection in the SIMULINK window. If the on-line simulation is selected, then the **On-line Control** window from Figure 3.7 is also opened.

From this moment, the MATLAB routine is halted and the simulation may be started. The routine only continues after activating the command window and pressing any key. This allows the user to run several simulations, using different control inputs and/or adjusting the

operating shell.
Before starting the simulation the operating shell in the SIMULINK window of Figure 2.15 may be adjusted first. The parameters in the various blocks may be given new values. The sources of the control inputs may be changed by replacing their blocks with blocks from the simulink/Sources library, e.g. *From File* or *From Workspace*. (Remember that on-line simulation may then not be possible anymore.) The selection of observation outputs may be modified by extracting other signals from the *Demux* block. The observation outputs may also made accessible in a different way by using blocks from the simulink/Sinks library as for example *To File* and *To Workspace*.
The simulation is started by selecting **Start** from the **Simulation** menu on top of the SIMULINK window. If desired, the simulation parameters may be adjusted first. They are accessible in the Control Panel dialog box which is displayed by selecting **Parameters** from the **Simulation** menu.
During the simulation, the user is allowed to interactively perform the following operations:

- change the parameters of blocks with engine control input and aircraft state variables; this may be done directly in the SIMULINK window, but the engine control input may also changed via its slider bars in the **On-line Control** window.
- change any of the simulation parameters or the simulation algorithm in the Control Panel dialog box.
- suspend and restart the simulation via the items **Pause** and **Restart** in the **Simulation** menu; during suspension, lines or blocks in the operating shell may be added or deleted.
- stop the simulation via the item **Stop** in the **Simulation** menu.

The simulation results include the vector of the integration time points t and the time trajectories of the input controls, state variables and observations outputs:

$$\begin{array}{lll} \text{input controls} & \begin{cases} PLA = \texttt{ut} & \text{(thrust controls)} \\ x_a(0) = \texttt{x0} & \text{(aircraft states)} \end{cases} & \\ \text{state variables} & x_t = \texttt{xt} & \text{(engine states, empty if static engine model included)} \\ \text{observation outputs} & y_{pow} = \texttt{ypow} & \text{(engine parameters)} \end{array} \tag{3.5}$$

These results are overwritten each time a simulation is finished. They further only become available in the workspace when the simulation is terminated, if desired via the item **Stop** in the **Simulation** menu, but not when a simulation is suspended.

This ends the engine simulation. In contrast to the aircraft simulation, the simulation results cannot be saved in a standardized data-file. However, the results are available in the MATLAB workspace and may manually be further processed and/or saved by the user.

3.4.2 Simulation in command window

If the simulation is executed from the command window, the simulation runs in the background. The simulation model and the control inputs are generated before starting the simulation and are not adjustable anymore. The user cannot interfere during the simulation and the responses only become available after the simulation has stopped.

The dialog in the command window which started with the display in Figure 3.16 is now continued with the display in Figure 3.17 where the time traces of the control inputs need to be specified. They may be loaded from an input-file `.inp` or they may created interactively, see Section 3.3.3 *Specification of control inputs.* Although the inputs of the main flight-controls may also be specified only the thrust control is effective during the engine simulation. This provides compatibility with application for aircraft simulation. The same conditions therfore apply, i.e. the signals are treated as perturbations on the trimmed values for the flight-condition in the trim-file `.tri` and the thrust control input may get saturated during the engine simulation.
The simulation parameters are now specified. The stop time or duration of the the simulation is derived from the time-length of the control inputs, but it may be adjusted by the user. The minimum and maximum step sizes of the integration are used to constrain the step size that is taken for the error control. The relative error of the integration at each step is set at 10^{-3} and the applied integration method is the fifth order Runga-Kutta `rk45`.

The engine simulation model is now loaded in memory. When this is finished the simulation may be started by pressing any key. During the simulation the elapsed time is displayed via a counter as the simulation progresses. The simulation cannot not be prematurely stopped without loosing the simulation results.
The simulation results are written to the MATLAB workspace. They include the vector of the integration time points t and the time trajectories of the input controls, state variables and observations outputs:

$$
\begin{array}{lll}
\text{input controls} & \begin{cases} PLA = \texttt{ut} & \text{(thrust controls)} \\ x_a(0) = \texttt{x0} & \text{(aircraft states)} \end{cases} & \\
\text{state variables} & x_t = \texttt{xt} & \text{(engine states, empty if static engine model included)} \\
\text{observation outputs} & y_{pow} = \texttt{ypow} & \text{(engine parameters)}
\end{array}
\tag{3.6}
$$

The trajectories of the control inputs are linearly interpolated from their specified time traces before starting the simulation. During the simulation, additional values are obtained with respect to the step size at each integration step.

When the simulation is terminated, the dialog in the command window continues with the requests for saving the control inputs and simulation results in data-files, see Figure 3.17. The control inputs refer to the specified time traces before the simulation and are saved in an input-file `.inp`. The saved variables are specified in the variable `inpvar` in (3.4), see also

Section 2.5 *Data-files* and Table 2.2. These data may therefore also be used for running an aircraft simulation.
The simulation results itself are again not further processed via the simulation tool.

3.5 Trimming of Aircraft

The aircraft trimming tool provides a facility for generating a steady-state flight-condition for a specified aircraft configuration at a specified operating point. This trimmed flight-condition serves as the condition for the other tools of the *DASMAT* package. It is the starting point for the aircraft and engine simulation tools and the analysis point for the linearisation tool and aerodynamic model fitting tool.
The trimming tool returns the control inputs and aircraft and engine states which correspond to a flight-condition at which the linear and angular accelerations are zero. The flight-condition can be selected from the same six options which are also available in the NASA-program *LINEAR* [6].
Of all control inputs, only the primary flight controls and the thrust are regarded as independent for trimming the aircraft model. So these are calculated by the trim routine. The remaining control inputs, i.e. the secondary flight controls, as trim-tab settings, flap deflection and landing-gear position are regarded as invariable parameters of the aircraft configuration prespecified by the user. Moreover, the aircraft trimming is performed for a prespecified aircraft mass and corresponding mass distribution with zero wind and zero turbulence conditions. The trimmed thrust is subsequently used for trimming the engine model in the course of which the throttle (power lever angle) setting and the engine states are calculated.

This section first gives a short description of the trimming procedure. It then continues with the documentation of the available options for running the trimming tool, giving a step by step description of the user-supplied data and screen displays.
The trimming tool is provided in the MATLAB M-script `trim_ac.m`. The actual trimming procedure is performed via a cost criterion which is supplied in the M-function `trimcost.m`. The trimming procedure for the engine is performed via the M-function `trim_eng.m`. The flow diagram of the trimming tool is given in Figure 3.18.

The trimmed flight-condition represents a condition in which all of the motion variables are constant or zero, i.e. a condition at which the rotational and translational accelerations are zero. For the aircraft model, only the following time derivatives of the aircraft state vector must be zero to satisfy a trimmed condition:

$$\begin{aligned} \dot{p}_b &= \dot{q}_b = \dot{r}_b = 0 \\ \dot{V}_{TAS} &= \dot{\alpha} = \dot{\beta} = 0 \end{aligned} \tag{3.7}$$

In addition to the above requirements which trim the aircraft, the engine should be trimmed as well. This additionally requires the engine state derivatives to become zero. However, in the trimming process a static version version of the engine is used. The engine states of any dynamic engine model should be modelled as outputs of the static engine model.
The trimmed flight-condition is obtained from the solution of simultaneous nonlinear differential equations of motion with the state variables and control inputs as driving variables.

These equations are expanded with constraint equations which specify a fixed relationship between state variables in a certain flight-condition. The calculation is performed with a numerical optimisation algorithm which iteratively adjusts independent variables until a cost criterion is met.
The time derivatives of the aircraft states in (3.7) are functions of the state variables and the control inputs, which in turn are either specified by the aircraft configuration and operating point, or calculated during the trimming process. If calculated, they are either independent or constrained. Independent variables are iteratively adjusted by the numerical optimisation algorithm. Constrained variables are calculated via the constraint equations from the independent variables, the operating point and any additionally specified parameters for a selected flight-condition.
Considering the aircraft states, the position and direction of the aircraft in the horizontal plane have no influence on the forces and moments acting on the aircraft in an isentropic atmosphere. Therefore, the state variables x_e, y_e and ψ are irrelevant for trimming and set to zero. The geometric altitude h_e is relevant via the air density. Together with the air speed V_{TAS}, it is regarded as the specification of the operating point. The remaining aircraft state variables, i.e. the body axis rates and the angles of attack, sideslip, roll and pitch, are either constrained or independent variables according to the selected flight-condition. If constrained, they may follow from a possible further specification. The angle of attack and angle of sideslip remain as independent variables for most trimmed flight-conditions.

The trimming tool of the aircraft may either be started by clicking **Trim 'aircraft'** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package with `dasmat`. It may also directly be started by the command `trim_ac` from the MATLAB prompt.

After the trimming tool is started, the user is returned to the command-window where the display of Figure 3.19 is generated step by step. The first screen lets the user specify the aircraft configuration. This entails the aircraft weight, the trim-tab settings, flap deflection and landing-gear position. This data may be copied from a previously generated trim-file `.tri`. After a trim-file is selected, the applicable configuration is displayed, see Figure 3.20. If the configuration is not to be read from file, the configuration is successively entered by the user where the boundary values are displayed. The initial mass is used for calculating the center of gravity and moments of inertia from the mass model designated in the variable `ac_massmodel`. If nothing is entered at the question for reading the configuration from file, the trim routine stops and one returns to the MATLAB prompt.
Next, the operating point is specified via the desired altitude, air speed and initial guess of the thrust per engine and any non-working engines. If the configuration has been read from a trim-file, then the operating point may also be read from this file. This option is useful for evaluating operations at different flight-conditions only. The operating point will now also be displayed, see Figure 3.20. In all other cases, the operating point is entered from the keyboard, see Figure 3.19. The air speed may also be entered as Mach number, in which case nothing should be entered at the question for the desired air speed. Any non-working engines should be entered by their number, added by the drag of a non-working. For all engines working nothing should be entered at the question for non-working engine(s).

Now the aircraft configuration and operating point are defined, the flight-condition is specified. The following flight-conditions are available to the user, see also Figure 3.21:

- straight-and-level trim
- pushover-pullup
- level turn
- thrust-stabilised turn
- beta trim
- specific power turn

If desired, the trimming routine may stopped here. After a flight-condition has been selected, the trim routine continues requesting for additionally required specifications for the flight-condition. Moreover, constraints for the selected flight-condition are set by the trim routine. The details for each flight-condition are discussed next.

straight-and-level trim:
The straight-and-level trim determines a wings-level, constant flight-path angle trajectory. This flight-condition entails zero angular velocities and a zero angle of roll. The flight-condition is further specified via a flight-path angle γ or rate of climb ROC, see Figure 3.21. The routine automatically toggles between both options when nothing is entered. The flight-condition is thus determined within the following constraints:

$$\begin{array}{ll} \text{direct constraints:} & \left\{ \begin{array}{rcl} p_b = q_b = r_b & = & 0 \\ \phi & = & 0 \end{array} \right. \\ \text{entered constraints:} & \gamma \ / \ ROC \\ \text{operating point constraints:} & V_{TAS} \ / \ M, \ h_e \end{array} \tag{3.8}$$

The angle of pitch θ is calculated from the kinematic relationship between the air-path axes and the body axes:

$$\begin{array}{rcl} \theta & = & \arctan \dfrac{ab + c\sqrt{a^2 + b^2 - c^2}}{a^2 - c^2} \\ a & = & \cos\alpha \cos\beta \\ b & = & \sin\alpha \cos\beta \\ c & = & \sin\gamma \end{array} \tag{3.9}$$

All relevant state variables are now determined and the state derivatives have become functions only of the independent variables: primary flight controls, summed thrust of all engines, angle of attack and angle of sideslip.

pushover-pullup:
The pushover-pullup condition specifies a wings-level flight with load factor $n \neq 1$ at the moment the flight-path is horizontal. For $n > 1$, the operating point is the minimum altitude

of a pullup. For $n < 1$, the operating point is the maximum altitude of a pushover. The flight-condition is further specified via the load factor n, see Figure 3.21, and is subject to the following constraints:

$$\begin{array}{ll} \text{direct constraints:} & \left\{ \begin{array}{lcl} p_b & = & r_b = 0 \\ \phi & = & 0 \\ \gamma & = & 0 \end{array} \right. \\ \text{entered constraints:} & n \\ \text{operating point constraints:} & V_{TAS}/M,\ h_e \end{array} \tag{3.10}$$

The pitch rate q_b is calculated from the load factor via the equation for zero acceleration along the air-path Z-axis, while the angle of pitch θ is equal to the angle of attack as follows from (3.9) after substituting $\gamma = 0$:

$$\begin{aligned} q_b &= \frac{mg\,(n-1) + X_t\,\sin\alpha - Z_t\,\cos\alpha}{m\,V_{TAS}\,\cos\beta} \\ \theta &= \alpha \end{aligned} \tag{3.11}$$

The independent variables are again the primary flight controls, summed thrust of all engines, angle of attack and angle of sideslip.

level turn:
The level turn condition is a non-wings-level, constant-turn-rate flight at load factor $n > 1$. This load factor is to be specified by the user. Moreover, the user may also specify a flight-path angle γ or rate of climb ROC leading to ascending or descending spirals, see Figure 3.21. The routine automatically toggles between γ and ROC when nothing is entered. The constraints for this flight-condition are given by:

$$\begin{array}{ll} \text{direct constraints:} & - \\ \text{entered constraints:} & n,\ \gamma/ROC \\ \text{operating point constraints:} & V_{TAS}/M,\ h_e \end{array} \tag{3.12}$$

All relevant state variables are calculated from the above constraints. Using the requested load factor, the aerodynamic angle of roll μ and the turn rate $\dot{\psi}$ are calculated first from the equations for zero acceleration along the air-path Z-axis and along the projection of the air-path Y-axis in the horizontal plane:

$$\begin{aligned} \mu &= \pm \arctan \frac{\sqrt{\left(n + \frac{X_t\,\sin\alpha - Z_t\,\cos\alpha}{W}\right)^2 - \cos^2\gamma}}{\cos\gamma} \\ \dot{\psi} &= \frac{g\,\tan\mu}{V_{TAS}} \end{aligned} \tag{3.13}$$

where the positive (negative) sign for μ is used for a right (left) turn. Using these two definitions, the state variables are determined from the equation for zero acceleration along the body Y-axis, the kinematic relationship between the air-path axes and the body axes and the decomposition of $\dot{\psi}$ along the body axes:

$$
\begin{aligned}
q_b &= \dot{\psi}\,\sin^2\mu\left\{-\sin\gamma\,\sin\beta \pm \sqrt{\sin^2\gamma\,\sin^2\beta - \frac{\sin^2\gamma-\cos^2\beta}{\sin^2\mu}}\right\}\quad,\ (q_b>0)\\
p_b &= -\frac{1}{\cos\beta}\left\{q_b\left(\frac{\sin\alpha}{\tan\mu}+\cos\alpha\,\sin\beta\right)+\dot{\psi}\,\cos\alpha\,\sin\gamma\right\}\\
r_b &= \frac{1}{\cos\beta}\left\{q_b\left(\frac{\cos\alpha}{\tan\mu}-\sin\alpha\,\sin\beta\right)-\dot{\psi}\,\sin\alpha\,\sin\gamma\right\}\\
\theta &= -\arctan\frac{p_b}{\dot{\psi}}\\
\phi &= \arctan\frac{q_b}{r_b}
\end{aligned}
\tag{3.14}
$$

The independent variables are again the primary flight controls, summed thrust of all engines, angle of attack and angle of sideslip.

thrust-stabilised turn:
The thrust-stabilised turn has a constant-throttle, non-wings-level turn with a nonzero flight-path angle. In contrast with the level turn, the flight-path angle cannot be chosen now but is determined via a user-specified thrust. This thrust is taken from the initial guess during the specification of the operating point. The load factor n is again user-defined, see Figure 3.21. The flight-condition is therefore subject to the following constraints:

direct constraints:	–	
entered constraints:	$n,\ \sum_i T_{N_i}$	(3.15)
operating point constraints:	$V_{TAS}/M,\ h_e$	

The constraint equations are the same as those for the level turn in (3.13) and (3.14). However, the flight-path angle γ instead of the thrust control is now an independent variable during the minimisation process of the cost criterion. The other independent variables are again the primary flight controls, angle of attack and angle of sideslip.

beta trim:
The beta trim results in a non-wings-level, horizontal flight with constant heading, i.e. $\dot{\psi}=0$, at a user-defined angle of sideslip β, see Figure 3.21. For an aerodynamic symmetric aircraft, this trim condition for $\beta=0$ results in the same condition as the straight-and-level trim for $\gamma=0$. The constraints used with this flight-condition are:

$$
\begin{array}{lll}
\text{direct constraints:} & \left\{ \begin{array}{llllll} p_b & = & q_b & = & r_b & = & 0 \\ \gamma & = & 0 & & & & \end{array} \right. & \\
\text{entered constraints:} & \beta & & & & & (3.16) \\
\text{operating point constraints:} & V_{TAS}/M,\ h_e & & & & &
\end{array}
$$

The angle of pitch θ is calculated with the same constraint equations leading to (3.9) but now for the special case $\gamma = 0$:

$$
\theta = \arctan \frac{\sin\phi \, \sin\beta + \cos\phi \, \sin\alpha \, \cos\beta}{\cos\alpha \, \cos\beta} \tag{3.17}
$$

Moreover, the angle of roll ϕ is used as independent variable instead of the prespecified angle of sideslip. The other independent variables are again the primary flight controls, summed thrust of all engines and angle of sideslip.

specific power turn:
The specific power turn results in a condition for a horizontal level turn at a user-specified specific power P_s, see Figure 3.21. Unlike the other flight-conditions, the specific power flight-condition does not achieve a constant air speed. The other acceleration-like terms are however zero. The constraints for this flight-condition now become:

$$
\begin{array}{ll}
\text{direct constraints:} & \gamma = 0 \\
\text{entered constraints:} & P_s \\
\text{operating point constraints:} & V_{TAS}/M,\ h_e
\end{array} \tag{3.18}
$$

The constraint equations are the same as those for the level turn in (3.13) and (3.14), but simplified because $\gamma = 0$. This now gives:

$$
\begin{array}{rcl}
\mu & = & \pm \arctan \sqrt{\left(n + \frac{X_t \, \sin\alpha - Z_t \, \cos\alpha}{W}\right)^2 - 1} \\
\dot{\psi} & = & \dfrac{g \, \tan\mu}{V_{TAS}}
\end{array} \tag{3.19}
$$

with a positive (negative) sign for μ for a right (left) turn. The constrained state variables follow from:

$$
\begin{aligned}
q_b &= \dot{\psi}\ \sin\mu\ \cos\beta \\
p_b &= -\dot{\psi}\ \sin\mu\ \left(\frac{\sin\alpha}{\tan\mu} + \cos\alpha\ \sin\beta\right) \\
r_b &= \dot{\psi}\ \sin\mu\ \left(\frac{\cos\alpha}{\tan\mu} - \sin\alpha\ \sin\beta\right) \\
\theta &= -\arctan\ \frac{p_b}{\dot{\psi}} \\
\phi &= \arctan\ \frac{q_b}{r_b}
\end{aligned}
\tag{3.20}
$$

For the minimisation of the cost function, the load factor n has become an independent variable. The other independent variables are the primary flight controls, summed thrust of all engines and the angle of attack. Moreover, the cost function is altered to achieve a minimal difference from the air speed rate which results from the specified specific power:

$$
\dot{V}_{TAS_{spec}} = \frac{g\ P_s}{V_{TAS}} \tag{3.21}
$$

In all other flight-conditions $\dot{V}_{TAS_{spec}} = 0$ as depicted in (3.7).

This ends the discussion of all available flight-conditions. The trim routine now continues configuring the aircraft model and loading it in memory. This may take some time. The applied aircraft model is designated in the variable `ac_model`. It is either the S-function `ac_mod.m` or `ac_modpc.m`, depending on whether the general or compact aircraft model was selected during initialisation of *DASMAT*.

The trim routine then directly starts the minimisation algorithm for the cost criterion. This cost criterion J is a least squares type and includes weight factors w_i for the various acceleration terms:

$$
\begin{aligned}
J &= w_1\dot{p}_b^2 + w_2\dot{q}_b^2 + w_3\dot{r}_b^3 + w_4(\dot{V}_{TAS} - \dot{V}_{TAS_{spec}})^2 + w_5\dot{\alpha}^2 + w_6\dot{\beta}^2 \\
w &= [5\ \ 5\ \ 5\ \ 1\ \ 2\ \ 10]
\end{aligned}
\tag{3.22}
$$

The weights, which are invariably set in the code, account for the differences in absolute values of the various acceleration terms. The term for the air speed rate includes only a non-zero reference value for the specific power flight-condition. The cost criterion is minimised with the standard MATLAB routine `fmins.m` for multidimensional minimisation [9]. It is a direct search method which operates according to the Nelder-Meade simplex method [4].
The minimisation halts when the termination criterion is met for the cost criterion (default 10^{-10}) or the independent variables (default 10^{-10}), or when the maximum number of steps (default 5000) has been executed. The obtained results for the control inputs, aircraft state

variables and aircraft state derivatives may then be displayed after pressing any key, see Figure 3.22. The value of the cost function is also displayed. The relative cost indicates the factor of the obtained cost over the cost criterion. If the user is not yet satisfied with the results, the minimisation routine may be continued where the obtained results are now used as the starting values of the trim variables.

After the aircraft is satisfactory trimmed, the trim routine continues with trimming the engine. The engine trim determines the throttle (power lever angle) setting to obtain the required thrust for the aircraft trimmed condition. Moreover, the engine states are determined for this operating condition.
The engine trim is performed using a static version of the engine, designated in the variable `eng_statmodel`. This version has a direct feed-through of its submitted thrust control to the provided thrust. Furthermore, it provides the parameters which may serve as engine states for the dynamic engine model. The static engine model may be a derivation from the dynamic engine model by removing feedback loops between demanding and actual conditions.
The throttle setting and engine states are calculated with a false position (regula falsi) algorithm [13]. This algorithm is an interpolation method for finding the zero in a specified interval of a one-dimensional function. In the present case, the function is defined as the deviation of the engine model thrust from the required trimmed thrust and the interval is set by the minimum and maximum throttle settings. If the required thrust is beyond the interval of the realisable thrust, it will be displayed on the screen and throttle and engine states will be set to zero. The trim results may now only be applied for analysis with the thrust as control input, i.e. without including the engine model in the generic aircraft model.

At the end of the trimming routine the trim results may be saved in a trim-file `.tri`, see Figure 3.22. The saved variables are specified in the variable `trimvar`, see also Section 2.5 and Table 2.2:

```
trimvar  =  massinit x0 xt0 u0 ut0 Tn0 x0dot
            flightcond constraint                    (3.23)
```

The variable list includes the mass properties, the trimmed state variables and control inputs for both the aircraft and the engine, the trimmed thrust and the flight-condition with its constrained variables. These variables are used in the other *DASMAT* tools, either for specifying an analysis point or as starting point for a simulation.

3.6 Linearizing of Aircraft

The aircraft linearization tool provides a facility for linearizing the nonlinear model of a specified aircraft about an operating point. This operating point includes a flight-condition generated via the trimming tool.
The linearization tool returns the linear state-space model for the aircraft model without engine model and with zero wind and no turbulence. The control inputs consist of the primary flight controls and thrust. The state vector only contains the aircraft state variables. The secondary flight controls, flap deflection and landing-gear position are set as invariables through the operating point specification. The observation outputs may be iteratively selected by the

user from the list corresponding to the generic aircraft model. Furthermore, the user has the option to linearize the complete nonlinear model, or to extract the symmetric and asymmetric model equations only.

This section first gives a short description of the applied linearization technique and then continues with the documentation of the available options for running the linearization tool. It gives a step by step description of the user-supplied data and the screen displays.
The linearization tool is provided in the MATLAB M-script `lin_ac.m`. The actual linearization procedure is performed in the M-function `lin_ac1.m` which is an adapted version of the standard MATLAB routine `linmod.m` [10]. The flow diagram of the linearization tool is given in Figure 3.23.

The linear model of the aircraft consist of time-invariant system matrices which are the first-order terms of a Taylor series expansion about the analysis point:

$$\begin{array}{rcl} \dot{x} & = & Ax + Bu \\ y & = & Cx + Du \end{array} \tag{3.24}$$

The technique which is employed to obtain these matrices numerically is a simple approximation to the partial derivative. Small positive and negative perturbations are imposed on each component of the aircraft states and control inputs in the analysis point. The average of the resulting perturbations for the aircraft states and observation outputs are then used to construct the system matrices:

$$\begin{array}{rcl} A_{ij} & = & \dfrac{f_i(x_0 + \delta x_j, u_0) - f_i(x_0 - \delta x_j, u_0)}{2\ \delta x_j} \\ B_{ij} & = & \dfrac{f_i(x_0, u_0 + \delta u_j) - f_i(x_0, u_0 - \delta u_j)}{2\ \delta u_j} \\ C_{ij} & = & \dfrac{g_i(x_0 + \delta x_j, u_0) - g_i(x_0 - \delta x_j, u_0)}{2\ \delta x_j} \\ D_{ij} & = & \dfrac{g_i(x_0, u_0 + \delta u_j) - g_i(x_0, u_0 - \delta u_j)}{2\ \delta u_j} \end{array} \tag{3.25}$$

where the operating point is specified by x_0 and u_0 and where the nonlinear equations are given by:

$$\begin{array}{rcl} \dot{x} & = & f(x, u) \\ y & = & g(x, u) \end{array} \tag{3.26}$$

This method is generally known as the central difference method [6].

The linearization tool of the aircraft may either be started by clicking **Linearize 'aircraft'** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package with

`dasmat`. It may also directly started by the command `lin_ac` from the MATLAB prompt.

After the linearization tool is started, the user is returned to the command window where the display of Figure 3.24 is generated step by step. As a first step the operating point must be set from a flight-condition which is saved in a trim-file `.tri`. If no file is entered, the routine stops and one returns to the MATLAB prompt.
The next input selects which aircraft states and control inputs should be applied for the linearization. If the total model is selected, the complete nonlinear model is linearized, i.e. perturbing all 12 aircraft states, the primary flight controls and the thrust for all engines. For the symmetric model, only the aircraft states for the symmetric degrees of freedom plus the elevator and thrust control are perturbed. For the asymmetric model, the aircraft states for the asymmetric degrees of freedom plus aileron, rudder and thrust control are perturbed. The states and inputs in the linearized models therefore become:

$$\begin{array}{ll} \text{total model:} & \left\{ \begin{array}{lcl} x & = & [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \\ u & = & [\delta_e \;\; \delta_a \;\; \delta_r \;\; T_{N_1} \;\; T_{N_2}] \end{array} \right. \\ \text{symmetric model:} & \left\{ \begin{array}{lcl} x & = & [q_b \;\; V_{TAS} \;\; \alpha \;\; \theta \;\; h_e \;\; x_e] \\ u & = & [\delta_e \;\; T_{N_1} \;\; T_{N_2}] \end{array} \right. \\ \text{asymmetric model:} & \left\{ \begin{array}{lcl} x & = & [p_b \;\; r_b \;\; \beta \;\; \phi \;\; \psi \;\; y_e] \\ u & = & [\delta_a \;\; \delta_r \;\; T_{N_1} \;\; T_{N_2}] \end{array} \right. \end{array} \tag{3.27}$$

The thrust control is used in all models because both symmetric and asymmetric thrust may be generated through engine selection.
Now the observation outputs for the linear model are selected. The user should first select from which group the outputs are included. In Figure 3.24 the groups are shown for the general aircraft model `ac_mod.m`. The next screen(s) subsequently request for the scalar observation outputs per selected group. When the compact aircraft model `ac_modpc.m` would have been selected during the initialization of *DASMAT*, only a reduced number of groups and observation outputs are selectable, see Section 2.3.3 and Table B.3.

The aircraft model is now configured and loaded in memory. This may take some time. The applied aircraft model, either the S-function `ac_mod.m` or `ac_modpc.m`, is designated in the variable `ac_model`.
The linearization algorithm is directly started afterwards. The linearization results become available in the MATLAB workspace as the variables `Alin`, `Blin`, `Clin` and `Dlin`.

At the end of the linearization tool, the linearization results may be saved in a linearization-file `.lin`, see Figure 3.24. The saved variables are specified in the variable `linvar`, see also Section 2.5 and Table 2.2.

```
linvar = massinit x0 u0 Tn0 Deng flightcond constraint
         Alin Blin Clin Dlin yndx do_PLA ac_model
```
(3.28)

The variable list includes specifications for the aircraft configuration and operating point, i.e. the trim results from the trim-file `.tri`, the system matrices, the indices of the selected observation outputs and the name of the applied aircraft model.

3.7 Fitting of aerodynamic model

The aerodynamic model fitting tool provides a facility for fitting a nonlinear aerodynamic model of a specified aircraft into a polynomial structure using data at one or more operating points. The data is either generated by submitting perturbations around trimmed flight-conditions to the aerodynamic model, or the data consists of time histories obtained from simulations or actual flight tests.
The fitting tool returns the coefficients in the polynomials which describe the aerodynamic force and moment coefficients or the force and moment and moment coefficients due to turbulence. The independent variables in the aerodynamic polynomials are selected from the nondimensional aircraft states, angle of attack rate, angle of sideslip rate and aerodynamic control inputs. Since the force and moment coefficients do not depend on all independent variables, the user may interactively select the variables to be used in the fitting procedure for each aerodynamic polynomial. Moreover, additional variables may be created by raising variables to a power or multiplying variables. If the polynomials are restricted to have linear dependency only, then the polynomial coefficients become the stability and control derivatives.
The independent variables in the gust polynomials consist of the nondimensional gust velocities and velocity rates. The applied polynomials assume only linear dependency and the polynomial coefficients are generally known as gust derivatives.

This section first gives a short description of the applied polynomial structure and fitting technique. It then continues with the documentation of the available options for running the fitting tool, giving a step by step description of the user-supplied data and screen displays.
The fitting tool is provided in the MATLAB M-script `fit_aero.m`. The actual fitting procedure is performed in the M-script `fit_aero1.m` for both the aerodynamic and gust polynomials. The flow diagram of the fitting tool is given in Figure 3.25.

The nonlinear aerodynamic model is expressed into a polynomial structure which consists of a baseline part and an extended part. The baseline part only includes the linear dependency of the force and moment coefficients on the independent variables. The following equations are assumed for the aerodynamic and gust polynomials of each force and moment coefficient:

$$\begin{aligned} C &= C_0 + C_\alpha \alpha + C_{\dot{\alpha}} \frac{\dot{\alpha}\bar{c}}{V} + C_\beta \beta + C_{\dot{\beta}} \frac{\dot{\beta} b}{V} + C_p \frac{pb}{V} + C_q \frac{q\bar{c}}{V} + C_r \frac{rb}{V} + C_M M + \\ &\quad C_{\delta_e}\delta_e + C_{\delta_a}\delta_a + C_{\delta_r}\delta_r + C_{\delta_{te}}\delta_{te} + C_{\delta_{ta}}\delta_{ta} + C_{\delta_{tr}}\delta_{tr} + C_{\delta_f}\delta_f \\ C_g &= C_{0_g} + C_{u_g}\hat{u}_g + C_{\alpha_g}\alpha_g + C_{\beta_g}\beta_g + C_{\dot{u}_g}\frac{\dot{\hat{u}}_g\bar{c}}{V} + C_{\dot{\alpha}_g}\frac{\dot{\alpha}_g\bar{c}}{V} + C_{\dot{\beta}_g}\frac{\dot{\beta}_g b}{V} + \\ &\quad C_{u_g}\hat{u}_{g_{asym}} + C_{\alpha_g}\alpha_{g_{asym}} \end{aligned} \tag{3.29}$$

Any extended part of the polynomials consist of terms which are functions generated from the independent variables. These terms should be manually created by the user, which is best done before the fitting procedure is started. The details will be described below.
The polynomials are fitted using the least squares technique. Each polynomial may be vectorized into the following equation:

$$
\begin{aligned}
y_m(i) &= \sum_k x_k(i)\,\theta_k + \epsilon(i) \\
Y_m &= X\,\theta + e
\end{aligned}
\tag{3.30}
$$

where θ denotes the polynomial coefficients and Y_m and X have per row the dependent and independent variables at each data point i. The vector e denotes the residuals between the 'measured' values and the results from the polynomial. The least squares estimate of θ is now given by:

$$
\hat{\theta} = \left[X^T X\right]^{-1} X^T Y_m \tag{3.31}
$$

The significance of the fitting results may be examined by several quantities. The following fit analysis statistics are computed by the fitting tool:

- residual sum of squares RSS
- goodness of fit or squared multiple correlation coefficient R^2
- statistical significance of all parameters F-value
- estimated mean of residuals $\hat{E}(\epsilon)$
- estimated variance of residuals $\hat{V}(\epsilon)$
- max. relative residue $\left|\frac{\epsilon}{y_m}\right|_{\max}$

The residual sum of squares RSS is defined as:

$$
\begin{aligned}
\mathrm{RSS} &= \sum_{i=1}^{N} \hat{\epsilon}^2(i) = \sum_{i=1}^{N} \left[y_m(i) - \hat{y}(i)\right]^2 \\
&= \hat{e}^T \hat{e}
\end{aligned}
\tag{3.32}
$$

where $\hat{y}(i) = \sum_k x_k(i)\,\hat{\theta}_k$ is the estimate of the polynomial result and N is the total number of data points.
The goodness of fit or squared multiple correlation coefficient R^2 expresses the correlation between $y_m(i)$ and $\hat{y}(i)$. It is defined as:

$$
\begin{aligned}
R^2 &= \frac{\left[\sum_i \left(y_m(i) - \bar{y}_m\right)\left(\hat{y}(i) - \bar{\hat{y}}\right)\right]^2}{\sum_i \left(y_m(i) - \bar{y}_m\right)^2 \sum_i \left(\hat{y}(i) - \bar{\hat{y}}\right)^2} \\
&= 1 - \frac{\hat{e}^T \hat{e}}{\left[Y_m - \bar{Y}_m\right]^T \left[Y_m - \bar{Y}_m\right]}
\end{aligned}
\tag{3.33}
$$

where the overlined variables denote their average values. The value of R^2 varies from 0 to 1 where the goodness of fit becomes one for a perfect fit.
The statistical value F is given as the ratio of the regression mean square to the residual mean square:

$$\begin{aligned} F &= \frac{\frac{1}{r-1}\sum_i \left(\hat{y}(i)-\bar{\hat{y}}\right)^2}{\frac{1}{N-r}\sum_i \hat{\epsilon}^2(i)} \\ &= \frac{N-r}{r-1}\frac{R^2}{1-R^2} \end{aligned} \tag{3.34}$$

where r is the number of applied parameters in the polynomial. When evaluating different models, the polynomial structure with the maximum F-value is recommended as the 'best' one for a given set of data.
The residuals are examined via their mean and variance. The estimates of these statistical moments are calculated via:

$$\begin{aligned} \hat{E}(\epsilon) &= \frac{1}{N}\sum_{i=1}^{N}\hat{\epsilon}(i) \\ \hat{V}(\epsilon) &= \frac{1}{N-r}\sum_{i=1}^{N}\hat{\epsilon}^2(i) = \frac{\hat{e}^T\hat{e}}{N-r} \end{aligned} \tag{3.35}$$

For an adequate fit, the mean should become zero and the variance should go to the value or estimate of the implemented ϵ. The time history of the residuals itself should be close to a random sequence which is uncorrelated.
A fit may be improved by developing the polynomial via residual analysis, where in successive steps variables are removed and appended. A very well known procedure is stepwise regression, see [5, 7].

The fitting tool of the aerodynamic model may either be started by clicking **Fit Aerodynamic Model** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package with `dasmat`. It may also directly started by the command `fit_aero` from the MATLAB prompt.

After the fitting tool is started, the user is returned to the command window where the display of Figure 3.26 is generated step by step. As a first step the user has to specify whether the polynomial should be fitted from data applied to an aerodynamic model or from (simulated) flight-test data. If no input is given, the routine stops and one returns to the MATLAB prompt. If an aerodynamic model is used, the operating point(s) must next be set from a flight-condition(s) which is saved in a trim-file `.tri`, see Figure 3.26. If flight-test data are to be used, a simulation-file `.sim` must be specified, see Figure 3.27. This data-file may be created after the simulation of the aircraft at the end of the aircraft simulation tool `sim_ac` or it may be compiled from recorded data of an actual flight. The MAT-file with extension `.sim` should then at least contain the aerodynamic control inputs u_a and aircraft states x_a

in the correct format, and the observation outputs y should include the force and moment coefficients. Moreover, y may include angle of attack rate and angle of sideslip rate for use in the aerodynamic polynomials. For fitting gust polynomials, y should always include the dimensionless gust variables. Hence (see also Appendix B *Signal formats of generic aircraft models* and Appendix C *Variables used by DASMAT*):

$$
\begin{aligned}
\mathbf{u} &= u_a = [\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}] \\
\mathbf{x} &= x_a = [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \\
\mathbf{y} &= y \supset \begin{cases} [C_D \;\; C_Y \;\; C_L \;\; C_\ell \;\; C_m \;\; C_n] & \\ [\dot{\alpha} \;\; \dot{\beta}] & \text{(aerod. polynomials)} \\ [\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g \bar{c}/V \;\; \dot{\alpha}_g \bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}] & \text{(gust polynomials)} \end{cases}
\end{aligned}
\tag{3.36}
$$

After the first trim-file or simulation-file is selected, the user is requested whether the fit should be based on more than one operating point by concatenating signals for each flight-condition. The polynomial coefficients will then be calculated from a polynomial which best fits the scattered data around each of the selected operating points. This will result in polynomials which may expand the complete flight envelope, instead of being applicable to only a restricted condition. If additional operating points are selected, a loop is initiated where a list of already selected trim-files or simulation-files is displayed and an additional file may be entered. This loop continues until no file is entered.
For fitting data applied to an aerodynamic model, the user must subsequently enter the number of data points. The number is regarded as number of 'measurements' and entails the number of perturbations to be created around the operating point(s). If multiple operating points are used for fitting the polynomials, the specified number of data points is equally divided over the various operating points.
The next input selects the reference frame in which the force and moment coefficients are defined. The user may selected from the air-path, stability and body reference frame.
Now, the user has to specify whether the aerodynamic or the gust model has to be fitted, i.e. whether the coefficients in the first or second polynomial in (3.29) have to be calculated.
The routine subsequently requests information about the structure of the aerodynamic polynomials if the aerodynamic model was previously selected. If the gust model was selected, the structure of the polynomials is fixed to the second equation in (3.29) and the routine skips the dialog for the polynomial structure determination.

The dialog in the command window continues with the display in Figure 3.28 when the aerodynamic model is fitted. First, the user may select the option for fitting the complete nonlinear model, or the polynomials for the symmetric or asymmetric force and moment coefficients only. This selection only determines the number of fits to be executed, i.e. six polynomials for the complete model and three for both other models.
The next request determines the structure of the aerodynamic model, i.e. the choice of the independent variables in the baseline polynomials according to the first equation in (3.29). If the user wants to select the variables himself, a list of all independent variables is shown for each aerodynamic force and moment coefficient, see Figure 3.28. The user may then select which variables are to be included in each polynomial. If the standard variables are used, the aerodynamic force and moment coefficients have the following arguments:

$$
\begin{aligned}
C_D &= f(1, \alpha, \beta, \tfrac{q\bar{c}}{V}, M, \delta_e, \delta_a, \delta_r, \delta_f) \\
C_Y &= f(1, \alpha, \beta, \tfrac{rb}{V}, M, \delta_a, \delta_r, \delta_f) \\
C_L &= f(1, \alpha, \tfrac{\dot{\alpha}\bar{c}}{V}, \tfrac{q\bar{c}}{V}, M, \delta_e, \delta_f) \\
C_\ell &= f(1, \alpha, \beta, \tfrac{pb}{V}, \tfrac{rb}{V}, M, \delta_a, \delta_r, \delta_{t_a}, \delta_f) \\
C_m &= f(1, \alpha, \tfrac{\dot{\alpha}\bar{c}}{V}, \beta, \tfrac{q\bar{c}}{V}, M, \delta_e, \delta_r, \delta_{t_e}, \delta_f) \\
C_n &= f(1, \alpha, \beta, \tfrac{pb}{V}, \tfrac{rb}{V}, M, \delta_a, \delta_r, \delta_{t_r}, \delta_f)
\end{aligned}
\tag{3.37}
$$

The selected variables are recorded in variable `aerovar` by means of nonzero elements. The rows in `aerovar` correspond to the polynomials, i.e. $C_D \ldots C_n$. The columns refer to the polynomial structure in the first equation in (3.29).

The independent and dependent variables to be used in the fitting procedure are now created, see Figure 3.29. These variables become available in the MATLAB workspace as part of `Uaero` respectively `Yaero`. Each row corresponds to a data point, while the columns are arranged according to the signal formats of the generic aerodynamic model `aero_mod.m` in (2.17) and (2.19), see also Section 2.3.5 *Generic aerodynamic model.*
For fitting data applied to an aerodynamic model, the independent variables are created first. The selected variables are defined as sequences of uniformly distributed random numbers by imposing random perturbations on their values in the operating point(s). The other variables as well as any other input to the aerodynamic model remain constant at the trimmed values. MATLAB's random number generator is used with seed reset to 0 and maximum perturbations set to 0.01 for all selected variables, with the single exception of air speed which has a maximum perturbation of 1. The aerodynamic model is subsequently loaded in memory, i.e. the S-function `aero_mod.m` which is designated in the variable `aero_model`. Within this generic aerodynamic model, the S-function designated by the variable `ac_aeromodel` of the aircraft specific aerodynamic model is included. Directly afterwards, the aerodynamic force and moment coefficients or the force and moment coefficients due to turbulence, being the dependent variables, are calculated for each data point.
When using (simulated) flight-test data, the independent variables are obtained from the aerodynamic control inputs and aircraft states. For fitting the aerodynamic polynomials, these signals may be completed with the angle of attack rate and angle of sideslip rate, when available as observation output. For fitting the gust polynomials the signals need to be completed with the dimensionless gust signals. The additional signals are specified by the user via their indices in the observation outputs, see Figure 3.29. Subsequently, the indices of the observation outputs referring to the force and moment coefficients also have to be specified.

The necessary data to be used in the fitting procedure is now available. If the aerodynamic model is fitted, the user has the option to extend the baseline aerodynamic polynomials with additional terms, see Figure 3.30. These terms need to be functions of the independent variables and are usually of the form $x_1^{j_1} x_2^{j_2} \ldots x_n^{j_n}$. A frequently applied extension is the insertion of α^2 as independent variable in the polynomial for the aerodynamic drag and lift coefficients. The additional terms should be defined in the variable `Upoly1` where each column refers to a newly inserted variable.

The user has the possibility to iteratively modify the structure of the polynomials and fit them. After the data for generating the dependent and independent variables (`Uaero` and `Yaero`) exist in the MATLAB workspace, the fitting procedure may be restarted more than once by issuing the command `fit_aero1` from the MATLAB prompt. The additional terms of the polynomials may be redefined every time, before starting the fit procedure.

Now, the fitting procedure is started. Each polynomial for a force and moment equation is fitted separately. The independent variables are defined in the variables `Upoly0` for the baseline terms according to (3.29) and `Upoly1` contains any additional terms (Note that for the gust polynomials `Upoly1` is always empty.) Within `Upoly0` a selection is made of the baseline terms to be used for the each polynomial fit via the variable `aerovar`, see above.
After the fitting procedure is completed, the calculated coefficients in the polynomials are displayed, see Figure 3.31. They are available in the MATLAB workspace as `polycoef`. The rows correspond to a polynomial and the columns refer to the terms in either the aerodynamic polynomial or the gust polynomial in (3.29). The values `NaN` indicate that a particular term is not included in the polynomial.
The next screen shows the analysis statistics of the fitting procedure. For each polynomial, the residual sum of squares RSS, goodness of fit R^2, F-value and the mean, variance and max. relative value of the residues are shown. The equations for these quantities are given in (3.32) to (3.35). Depending on these values, the user may start a new fit procedure with a revised polynomial structure. This restart can be made via the command `fit_aero1` and then redefining `Upoly1`.

At the end of the fitting tool, the results may be saved in an aerofit-file `.aer`. The saved variables are specified in the variable `fitvar`, see also Section 2.5 and Table 2.2:

```
fitvar = sourcedata polycoef aerovar Upoly0 Upoly1 Uaero Yaero
         fitstat do_fitmodel do_fitgust do_fitaxes
```
(3.38)

The variable list includes the names of the applied trim-file(s) `.tri` or simulation-file(s) `.sim`, the polynomial coefficients, the selected independent variables in the baseline polynomials, the matrices with all independent variables, the matrices used for generating the independent and dependent variables and the fit analysis statistics. The switch `do_fitmodel` indicates whether the fit is based on flight-test data or data applied to an aerodynamic model ([flight/model] = [0/1]). The switch `do_fitgust` indicates the type of polynomials ([aerodynamic/gust] = [0/1]). The switch `do_fitaxes` indicates the applied reference frame for the force and moment coefficients ([air-path/stability/body] = [0/1/2]).

3.8 Plotting of aircraft simulation time-responses

The plotting tool of aircraft simulation time-responses provides a facility for off-line plotting the trajectories of the control inputs and aircraft state variables from a simulated or actual executed flight. The simulated flight includes the results generated via the aircraft simulation tool. When plotting actual flight data, the recorded data should be first organized in the format used for simulated flights.

This section gives the documentation of the available options for running the plotting tool. It gives a step by step description of the user-supplied data and the screen displays.
The plotting tool is provided in the MATLAB M-script `plot_ac.m`. The actual plotting procedure is performed in the M-function `plot_ac1.m`. The flow diagram of the plotting tool is given in Figure 3.32.

The plotting tool of the time-responses may either be started by clicking **Plot Time-Responses 'aircraft'** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package with `dasmat`. It may also directly started by the command `plot_ac` from the MATLAB prompt.

After the plotting tool is started, the user is returned to the command window where the display of Figure 3.33 is generated. First, the data of the flight to be plotted must be loaded from a simulation-file `.sim`. This data-file may be created after the simulation of the aircraft at the end of the aircraft simulation tool `sim_ac`. The data-file may also be compiled from recorded data of an actual flight, where at least the control inputs $u = [u_a \; u_t]$ and aircraft states x_a should be put in the correct format (see Appendix B *Signal formats of generic aircraft models* and Appendix C *Variables used by DASMAT*). The MAT-file with extension `.sim` should therefore include:

$$
\begin{aligned}
&\texttt{u} && = u_a && = [\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}] \\
&\left.\begin{matrix}\texttt{ut(i)} \\ \texttt{Tn(i)}\end{matrix}\right\} && = u_{t_i} && = \begin{cases} PLA_i & (\texttt{do\_PLA = 1; engine model included}) \\ T_{N_i} & (\texttt{do\_PLA = 0; engine model not included}) \end{cases} \\
&\texttt{x} && = x_a && = [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]
\end{aligned}
\tag{3.39}
$$

If no file is entered, the routine stops and one returns to the MATLAB prompt.
The next input selects the time-length of the trajectories which should be displayed. This value is used for defining the limit of the time axis in the graphs. The default time-length is taken from the stop time of the simulated flight. The user may however truncate the length of the displayed trajectories by entering a smaller value.

Now, the graphs with the trajectories of the primary flight controls, the thrust control and the aircraft states are generated. The thrust controls consist of either the throttle (power lever angle) settings when the engine model was included in the aircraft model, or the thrust itself. The trajectories are displayed in an array of 4×4 graphs. The graphs in the left column show the 4 control inputs, where the graph for the thrust controls shows the inputs for all engines:

$$
[\texttt{u(1:3)} \;\; \texttt{ut/Tn}] \;\; = \;\; [\delta_e \;\; \delta_a \;\; \delta_r \;\; PLA/T_N] \tag{3.40}
$$

The remaining graphs show the 12 aircraft states:

$$
\texttt{x} \;\; = \;\; [p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e] \tag{3.41}
$$

Figure 3.34 shows an example of the display with the graphs of the trajectories generated via the plotting tool.

The plotting tool is terminated by clicking the mouse in the window which displays the graphs.

3.9 Animation of aircraft simulation

The animation tool of an aircraft simulation provides a facility for off-line playing a movie of a simulated or actually executed flight with an aircraft figure in 3D. Next to the animation of the aircraft motions, some important flight data are also displayed. The simulated flight includes the results generated via the aircraft simulation tool. When playing an animation of an actual flight, the recorded data should be first organized in the format used for simulated flights. The animation tool may also be used on-line during a simulation in the SIMULINK window, i.e. using the default operating shell in the S-function `ac_sim.m`.

This section gives the documentation of the available options for running the animation tool. It gives a step by step description of the user-supplied data and the screen displays. When the animation tool is used during on-line simulation, the requests start with the selection of the visualization mode and the option for movie recording drops.
The animation tool is provided in the MATLAB M-script `show_ac.m`. The actual plotting procedure is performed in the M-functions `ac_anim*.m`. The flow diagram of the animation tool is given in Figure 3.35.

The animation tool of the aircraft simulation may either be started by clicking **Show Animation 'aircraft'** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package with `dasmat`. It may also directly started by the command `show_ac` from the MATLAB prompt.

After the animation tool is started, the user is returned to the command window where the display of Figure 3.36 is generated. First, the data of the animated flight must be loaded from a simulation-file `.sim`. This data-file may be created after the simulation of the aircraft at the end of the aircraft simulation tool `sim_ac.m`. If the data-file is compiled from recorded flight data, at least the control inputs $u = [u_a\ u_t]$ and aircraft states x_a should be put in the correct format (see Appendix B *Signal formats of generic aircraft models* and Appendix C *Variables used by DASMAT*). The MAT-file with extension `.sim` should therefore include:

$$
\begin{aligned}
\texttt{u} \quad &= \quad u_a \quad = \quad [\delta_e\ \ \delta_a\ \ \delta_r\ \ \delta_{t_e}\ \ \delta_{t_a}\ \ \delta_{t_r}\ \ \delta_f\ \ \ell g_{sw}] \\
\left.\begin{matrix}\texttt{ut(i)} \\ \texttt{Tn(i)}\end{matrix}\right\} \quad &= \quad u_{t_i} \quad = \quad \begin{cases} PLA_i & (\texttt{do\_PLA = 1; engine model included}) \\ T_{N_i} & (\texttt{do\_PLA = 0; engine model not included}) \end{cases} \\
\texttt{x} \quad &= \quad x_a \quad = \quad [p_b\ \ q_b\ \ r_b\ \ V_{TAS}\ \ \alpha\ \ \beta\ \ \phi\ \ \theta\ \ \psi\ \ h_e\ \ x_e\ \ y_e]
\end{aligned}
\tag{3.42}
$$

If no file is entered, the routine stops and one returns to the MATLAB prompt.

The next two inputs select the time-length of the animation and the number of frames which should be displayed. The default time-length is taken from the stop time of the simulated flight. The number of frames specifies how many animation plots should be rendered. In principle, a plot may be rendered at each time-point within the specified animation period as displayed via the selectable maximum number. A smaller number renders plots at equally spaced time-periods within the animation period, regardless any unequal time-spacing of the flight data. The user should be aware that more frames and larger windows may cause memory problems when the windows will be recorded for replay. Moreover, a high number of frames may cause overlap of aircraft figures when later on figures at previous time-points are selected to retain in the plots.
The following question lets the user specify whether the animation should be recorded for replay. If yes, a movie is built by storing a snapshot (pixmap) of each generated plot in memory. The animation may then be replayed as a movie several times and at variable speeds. However, building the movie causes a slow update of the plots and may require excessive memory. If no movie recording is selected, the user needs to specify the speed at which the animation has to be played in frames per second. A very fast animation may cause skipping of some frames.

Now, the user should specify the position of the viewpoint during the animation of the flight. Three modes may be selected, i.e. attitude, vertical position and flight path, followed by further specification of the view condition. The dialogs for the selection and specification modes are identical to those for animation during the aircraft simulation in the SIMULINK window. The dialogs are explained in Section 3.3.4 *Specification of simulation visualization.* For completeness this dialog is repeated in Figure 3.37 and examples of the displays are given in Figure 3.38.

The animation is now started. The **Animation** window is displayed on the screen and the recorded data from the simulation-file `.sim` are converted to aircraft motions. At the same time, the progress of some aircraft states and the elapsed time are displayed at the bottom of the window. The animation continues until the end of the selected animation-period without any interrupt.
As distinct from running an animation during on-line simulation, the view condition cannot be modified during the animation via `show_ac`. The button **set view condition** is therefore disabled. Moreover, the **Animation** window cannot be resized during the building and replaying of a movie because memory is preallocated by an amount according to the initial size of the window. The window should may however be resized before the animation is started.

The animation may be replayed if recording for replay was selected earlier. The dialog in the command window from Figure 3.37 then continues with the display in Figure 3.39. The number and the speed of the replays have to be entered after each other. The speed of the replays should be given in frames per second.

The animation tool is terminated by clicking the mouse in the animation window when no movie was recorded or by setting the number of replays to zero in the case of a movie recording.

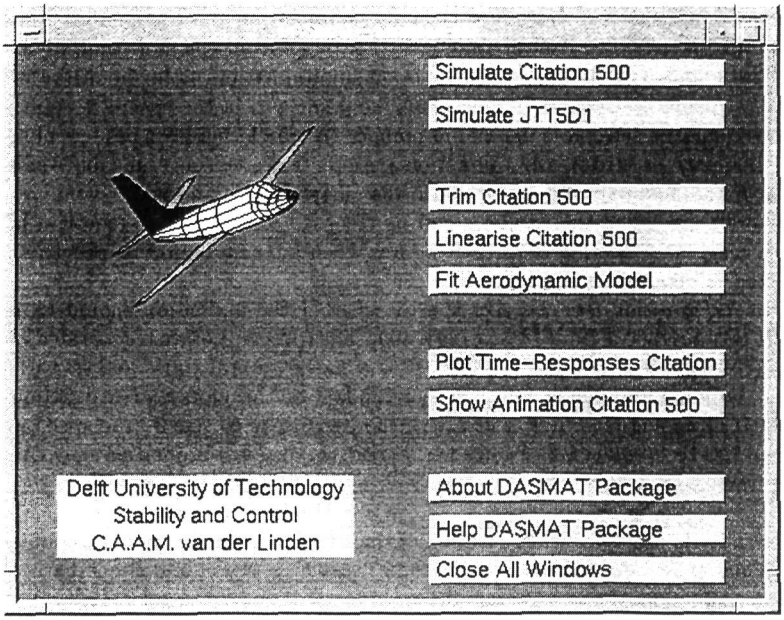


Figure 3.1: Menu window for controlling *DASMAT* package.

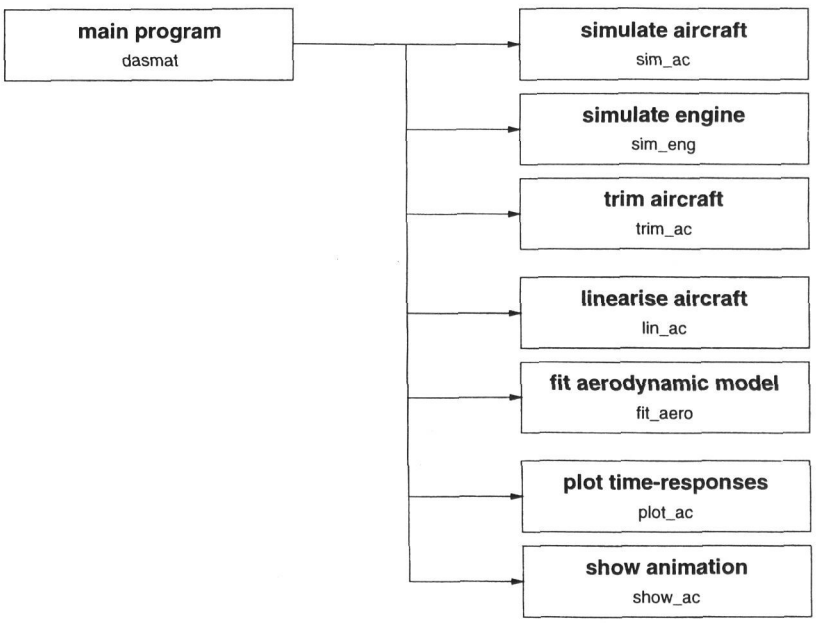


Figure 3.2: Commands for executing *DASMAT* routines from MATLAB prompt.

```
    ************************************************
    **                                            **
    **         DASMAT Initialisation-routine      **
    **                                            **
    ************************************************


directory for data files                  : data

use general/compact aircraft-model   [g/c] : g


specify aircraft to be analyzed


available files : *.m


cit_act.m   cit_afcs.m  cit_mass.m  cit_ref.m
cit_aero.m  cit_ctrl.m  cit_pow.m   citation.m


select file : citation
```

Figure 3.3: Display for *DASMAT* initialisation routine.

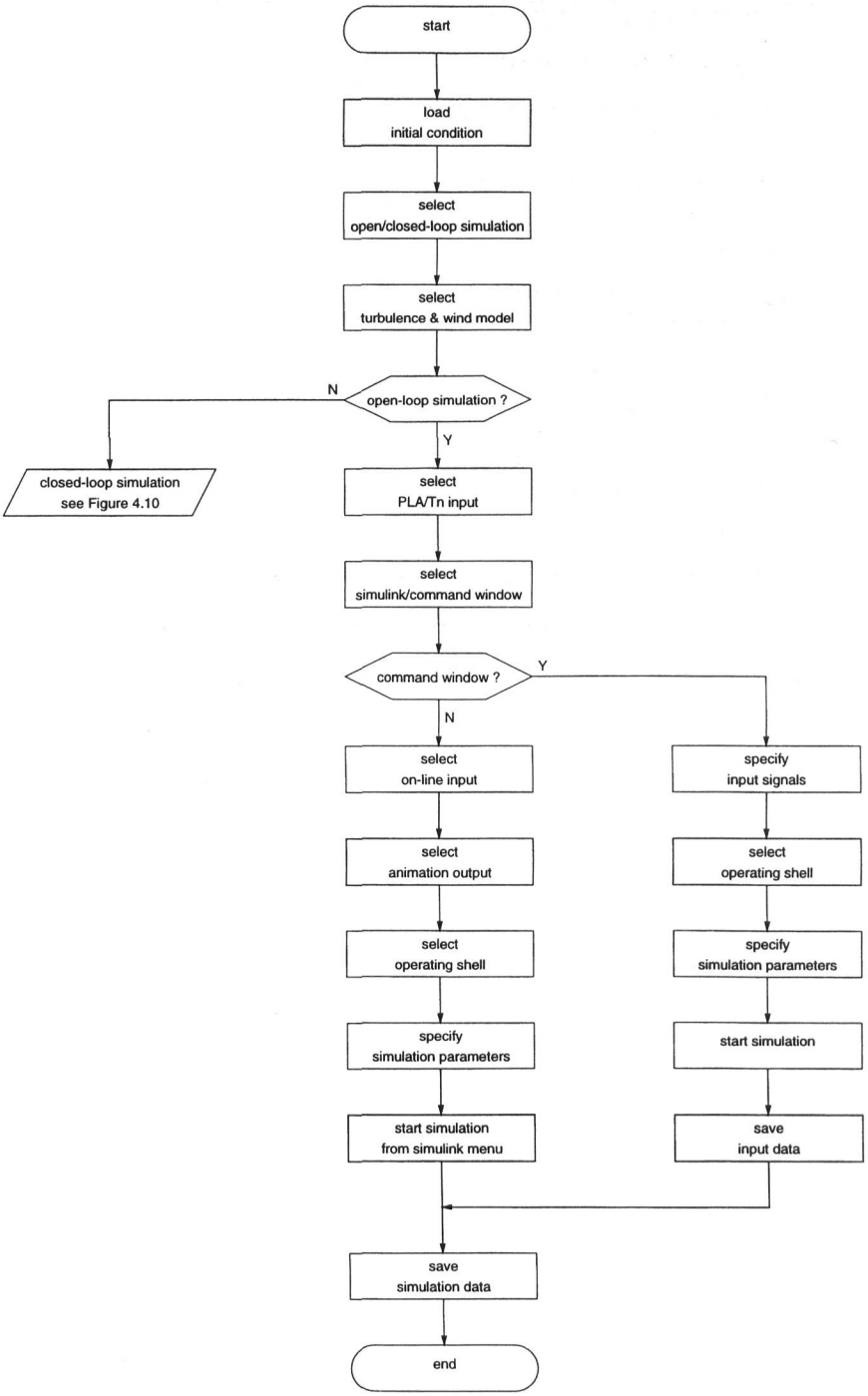


Figure 3.4: Flow-diagram of aircraft simulation tool `sim_ac.m`.

```
    ************************************************
    **                                            **
    **          Aircraft Simulation-routine       **
    **                                            **
    ************************************************


set starting point from trimmed flight-condition


available files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file : lr50_1

run open-loop or closed-loop simulation         [o/c] : o

include non-zero wind condition                 [y/n] : n

include non-zero turbulence condition           [y/n] : n

use power lever or thrust for engine input      [p/t] : p

run simulation from simulink or command window  [s/c] : s
```

Figure 3.5: Display for aircraft simulation tool.

```
do on-line simulation                              [y/n] : y

show animation, time-responses or model          [a/r/m] : m

do simulation with default operating shell         [y/n] : y

================================== NEW SCREEN ==================================

specify simulation parameters

stop time          (in sec) :  10

min. step size     (in sec) :   0.001

max. step size     (in sec) :   0.010

loading simulation-model 'ac_sim', please wait ...

start simulation from menu in simulink window 'ac_sim'

save simulated flight in file                      [y/n] : y

current files : *.sim

data/example.sim

select file :
```

Figure 3.6: Follow-up display for aircraft simulation in SIMULINK window.

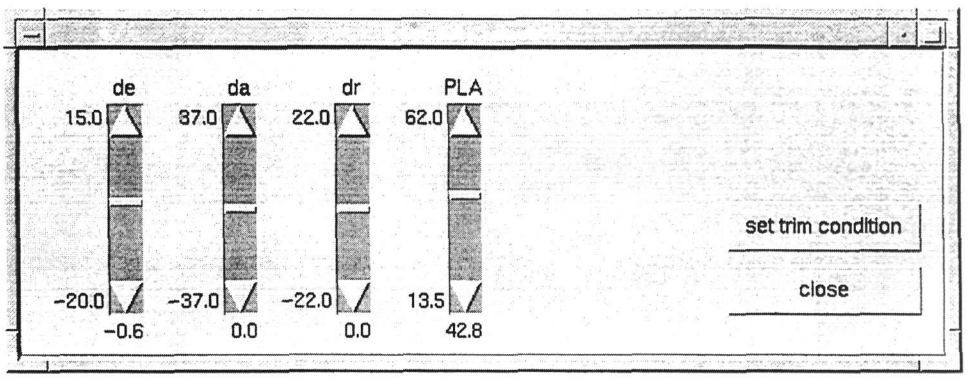


Figure 3.7: Window for on-line controlling main flight-controls at simulation in SIMULINK window.

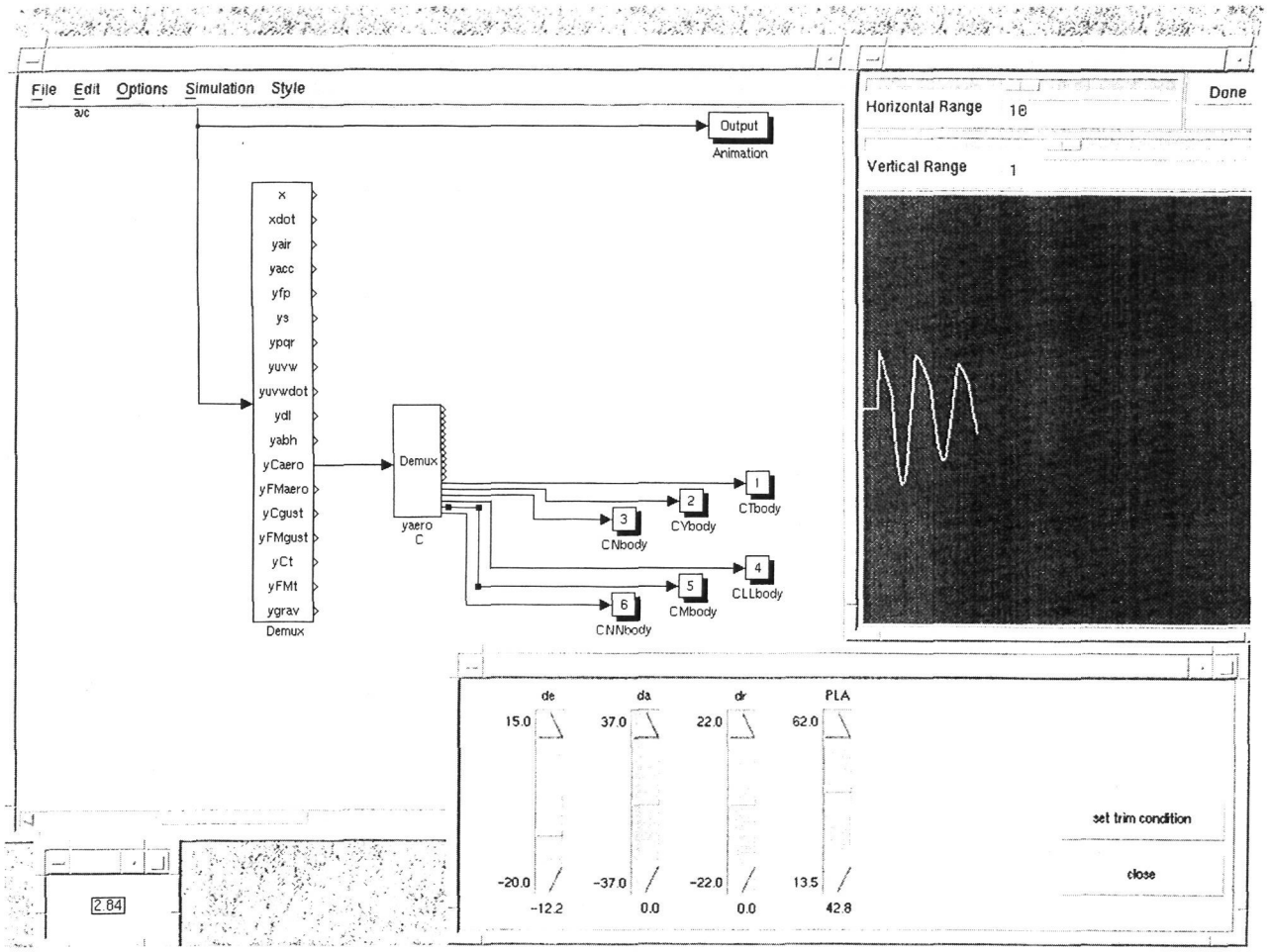


Figure 3.8: Screen lay-out for aircraft simulation with on-line control and model display.

```
do simulation with default operating shell      [y/n] : y

load input signals from file                    [y/n] : y


available files : *.inp

data/example.inp


select file : example

================================== NEW SCREEN ==================================

specify simulation parameters

stop time          (in sec) :  10

min. step size     (in sec) :   0.001

max. step size     (in sec) :   0.010


loading simulation-model 'ac_fun', please wait ...


press any key to start simulation

t :   0.00


save input signals in file                      [y/n] : y


current files : *.inp

data/example.inp


select file :

save simulated flight in file                   [y/n] : y


current files : *.sim

data/example.sim


select file :
```

Figure 3.9: Follow-up display for aircraft simulation in command window.

```
    ************************************************
    **                                            **
    **        Aircraft Control Input-routine      **
    **                                            **
    ************************************************


use power lever or thrust for engine input      [p/t] : p

================================== NEW SCREEN ==================================

specify inputs with non-zero signals

  1. delta_e          4. delta_te         7. delta_f
  2. delta_a          5. delta_ta         8. landgear
  3. delta_r          6. delta_tr         9. PLA

  0. none            10. all

Available option(s) :

  [ 0 1 2 3 9 10 ]

Select option(s) in vector [..] : 1


time-length signal     (in sec) : 10

================================== NEW SCREEN ==================================

specify signal for input : delta_e

   1. doublet
   2. 3211
   3. step
   4. puls
   5. sinus
   6. user defined

   0. zero signal

   signal type : 1

start time signal  [ 0.0 -  10.0] (in sec) : 2

end   time signal  [ 2.0 -  10.0] (in sec) : 6

max. amplitude signal                      : 5/180*pi
```

Figure 3.10: Display for creating control inputs.

```
================================= OTHER SIGNAL ================================

   signal type : 2

start time signal  [ 0.0 -  10.0] (in sec) : 2

end   time signal  [ 2.0 -  10.0] (in sec) : 9

max. amplitude signal                      : 5/180*pi

================================= OTHER SIGNAL ================================

   signal type : 3

start time signal  [ 0.0 -  10.0] (in sec) : 2

max. amplitude signal                      : 5/180*pi

================================= OTHER SIGNAL ================================

   signal type : 4

start time signal  [ 0.0 -  10.0] (in sec) : 2

end   time signal  [ 2.0 -  10.0] (in sec) : 2.5

max. amplitude signal                      : 5/180*pi

================================= OTHER SIGNAL ================================

   signal type : 5

start time signal  [ 0.0 -  10.0] (in sec) : 2

end   time signal  [ 2.0 -  10.0] (in sec) : 8

max. amplitude signal                      : 5/180*pi

frequency                    (in rad/sec) : pi

================================= OTHER SIGNAL ================================

   signal type : 6

start time signal  [ 0.0 -  10.0] (in sec) : 2

end   time signal  [ 2.0 -  10.0] (in sec) : 8

max. amplitude signal                      : 5/180*pi

signal in normalized units for t=[2,8] sec

t    = [2,3,5,6:0.01:8]

u(t) = [0,2,2,sin(2*pi*t(t>=6))]
```

Figure 3.10: Continue.

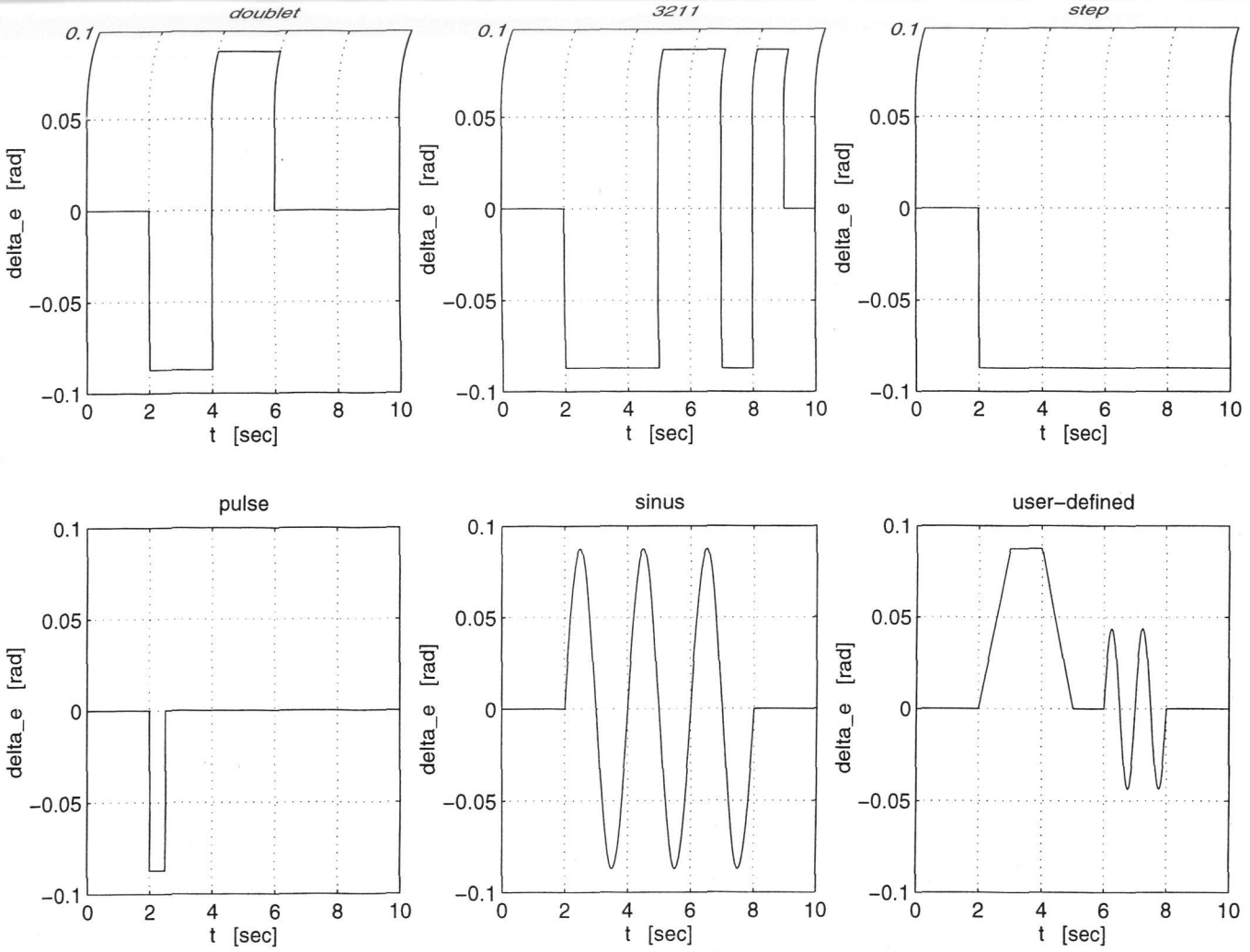
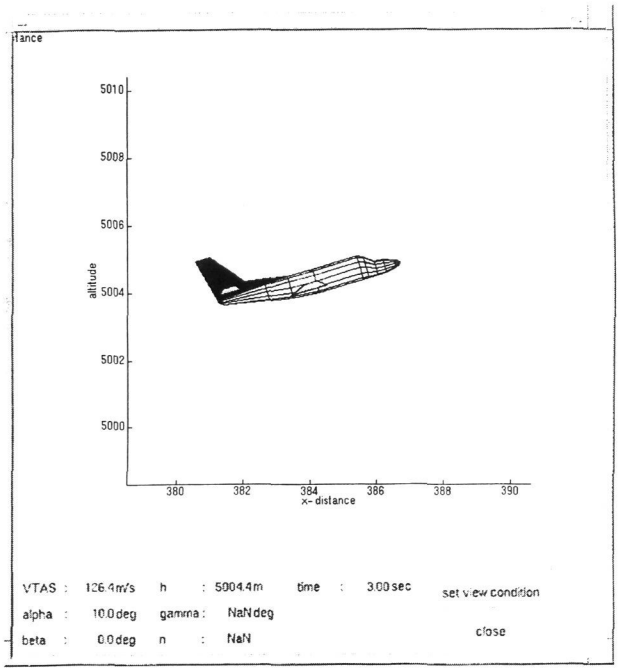


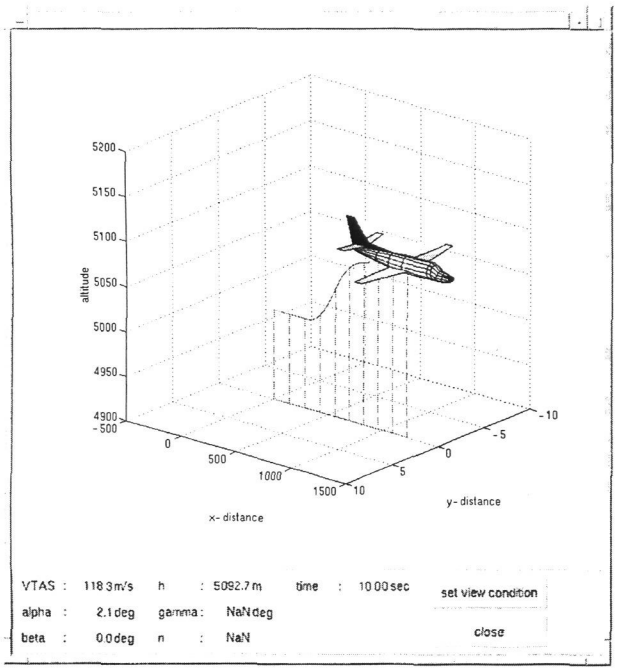
Figure 3.11: Control inputs generated from the entries in Figure 3.10.

```
    ************************************************
    **                                            **
    **            Aircraft Visualisation          **
    **                                            **
    ************************************************


view attitude, vert. position or flightpath   [a/v/f] : a


scale factor                                        : 0.454887
view azimuth                                (in deg) : 40
view elevation                              (in deg) : 20
================================== OTHER VIEW ==================================
view attitude, vert. position or flightpath   [a/v/f] : v

min. altitude                                (in m) :   4900
max. altitude                                (in m) :   5200

scale factor                                        : 0.454887
view azimuth                                (in deg) : 40
view elevation                              (in deg) : 20
================================== OTHER VIEW ==================================
view attitude, vert. position or flightpath   [a/v/f] : f
Retain a/c graphs in plot                      [y/n] : n

min. altitude                                (in m) :   4900
max. altitude                                (in m) :   5200
min. x-distance                              (in m) :   -500
max. x-distance                              (in m) :   1500
min. y-distance                              (in m) :    -10
max. y-distance                              (in m) :     10

scale factor                                        :  0.75188
view azimuth                                (in deg) : 40
view elevation                              (in deg) : 20
```

Figure 3.12: Display for specifying view parameters with animation of aircraft simulation.

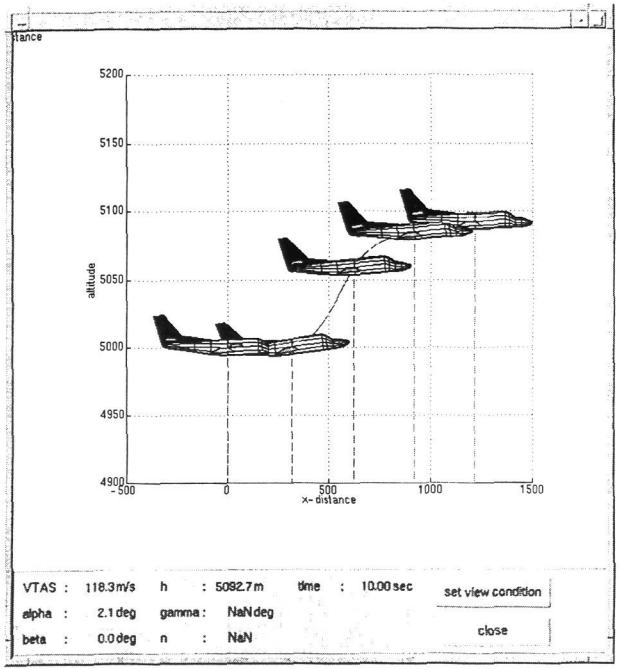


Example display of simulated response using option for attitude view.



Example display of simulated response using options for flight-path view and no retaining a/c graphs.

Figure 3.13: Screen lay-out for aircraft simulation displayed as animation.



Example display of simulated response using options for flight-path view and retaining a/c graphs.

Figure 3.13: Continue.

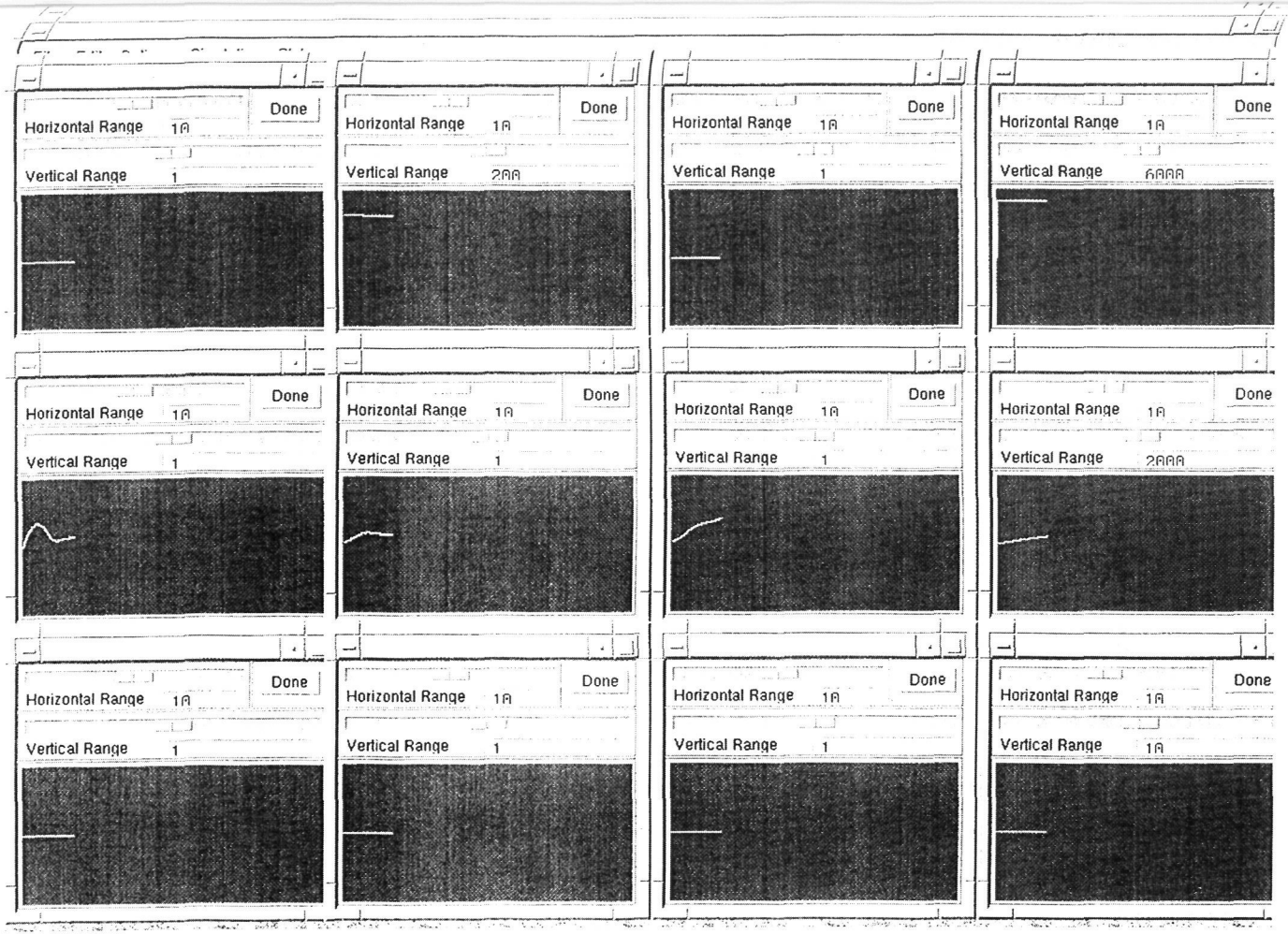


Figure 3.14: Screen lay-out for aircraft simulation displayed as time-responses.

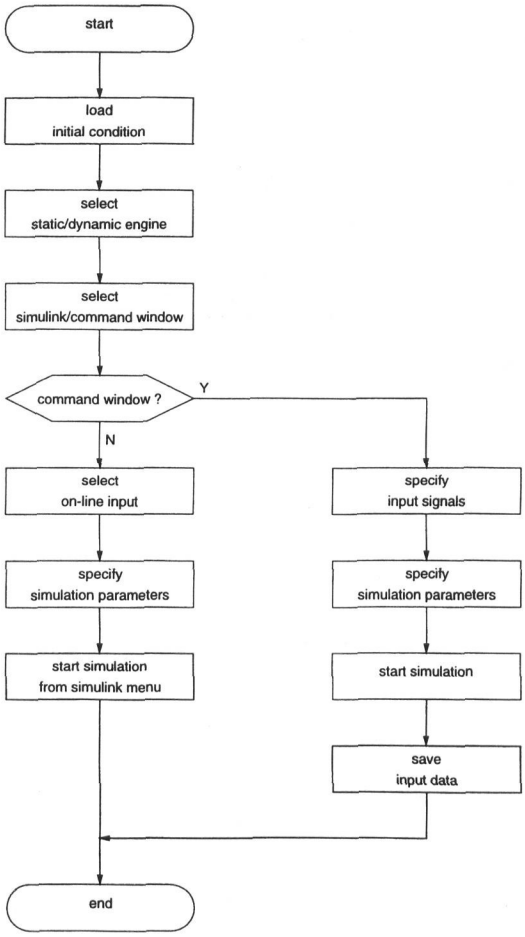


Figure 3.15: Flow-diagram of engine simulation tool `sim_eng.m`.

```
    ************************************************
    **                                            **
    **          Engine Simulation-routine         **
    **                                            **
    ************************************************


set starting point from trimmed flight-condition

available files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file : lr50_1

use static or dynamic engine                    [s/d] : d

run simulation from simulink or command window  [s/c] : s

do on-line simulation                           [y/n] : y

================================= NEW SCREEN ==================================

specify simulation parameters

stop time          (in sec) :  10

min. step size     (in sec) :   0.001

max. step size     (in sec) :   0.010


loading simulation-model 'eng_sim', please wait ...


start simulation from menu in simulink window 'eng_sim'


save simulated flight in file                   [y/n] : y


current files : *.sim

data/example.sim


select file :
```

Figure 3.16: Display for engine simulation tool in SIMULINK window.

```
load input signals from file                    [y/n] : y

available files : *.inp

data/example.inp

select file : example

================================== NEW SCREEN ==================================

specify simulation parameters

stop time          (in sec) :  10

min. step size     (in sec) :   0.001

max. step size     (in sec) :   0.010

loading simulation-model 'eng_fun', please wait ...

press any key to start simulation

t :   0.00

save input signals in file                      [y/n] : y

current files : *.inp

data/example.inp

select file :
```

Figure 3.17: Follow-up display for engine simulation tool in command window.

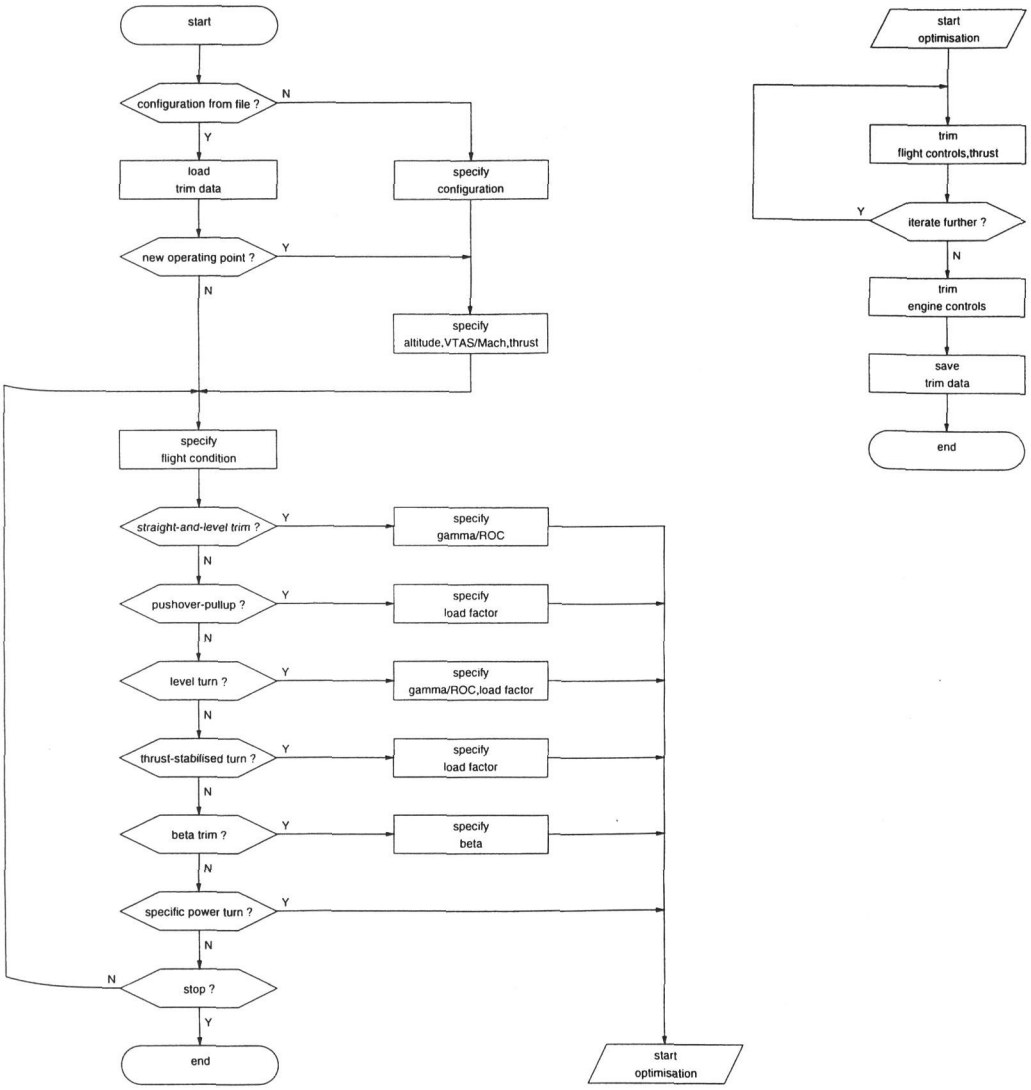


Figure 3.18: Flow-diagram of aircraft trimming tool `trim_ac.m`.

```
    ************************************************
    **                                            **
    **             Aircraft Trim-routine          **
    **                                            **
    ************************************************


read configuration from file            [y/n] : n

initial mass           [3338 - 5375]  (in kg) : 5375

delta_te           [-0.262 -  0.262] (in rad) : 0.000

delta_ta           [-0.349 -  0.349] (in rad) : 0.000

delta_tr           [-0.175 -  0.175] (in rad) : 0.000

delta_f            [ 0.000 -  0.698] (in rad) : 0.000

landing gear position     [0/1] (is [in/out]) : 0

the model to trim is now defined     press any key to continue

================================= NEW SCREEN ================================

desired altitude                    (S.A. in m) : 5000

desired airspeed                       (in m/s) :

desired Mach number                             : .3

initial guess thrust per engine          (in N) : 3000

non-working engine(s)   (default is all working) :
```

Figure 3.19: Display for aircraft trimming tool.

```
     ************************************************
     **                                            **
     **             Aircraft Trim-routine          **
     **                                            **
     ************************************************


read configuration from file            [y/n] : y


available files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file : lr50_1

initial mass                            (in kg) :  5207
delta_te                               (in rad) :  0.000
delta_ta                               (in rad) :  0.000
delta_tr                               (in rad) :  0.000
delta_f                                (in rad) :  0.000
landing gear position     [0/1] (is [in/out]) :  0

the model to trim is now defined     press any key to continue

================================= NEW SCREEN ==================================

trim old operating point                   [y/n] : y

desired altitude                     (S.A. in m) :  5000.000
desired airspeed                        (in m/s) :   128.211
desired Mach number                              :     0.400
average thrust per engine                 (in N) :  2410.907
```

Figure 3.20: Display for aircraft trimming tool after reading aircraft configuration and operating point from file.

```
specify flight-condition

   1. straight-and-level trim
   2. pushover-pullup
   3. level turn
   4. thrust-stabilized turn
   5. beta trim
   6. specific power turn

   0. stop

   flight-condition : 1

flightpath angle gamma                    (in rad) :

rate of climb                             (in m/s) : 0

============================ OTHER FLIGHT-CONDITION ============================

   flight-condition : 2

load factor                                        : 1.5

============================ OTHER FLIGHT-CONDITION ============================

   flight-condition : 3

flightpath angle gamma                    (in rad) :

rate of climb                             (in m/s) : 0

load factor                                        : 1.5

============================ OTHER FLIGHT-CONDITION ============================

   flight-condition : 4

load factor                                        : 1.5

============================ OTHER FLIGHT-CONDITION ============================

   flight-condition : 5

angle of sideslip                         (in rad) : 0

============================ OTHER FLIGHT-CONDITION ============================

   flight-condition : 6

specific power                            (in m/s) : 0
```

Figure 3.21: Follow-up display for aircraft trimming tool.

```
loading aircraft-model 'ac_mod', please wait ...


start of optimization

iteration stopped     press any key to get results

================================= NEW SCREEN =================================

                  states    derivatives
pbody       :      0.000   -2.50361e-10
qbody       :      0.000    3.36121e-16
rbody       :      0.000   -4.02139e-10
VTAS        :    128.211    1.35373e-15
alpha       :      0.036   -1.53679e-15
beta        :      0.000    9.36372e-12
phi         :      0.000    0.00000e+00
theta       :      0.036    0.00000e+00
psi         :      0.000    0.00000e+00
he          :   5000.000    8.88178e-16
xe          :      0.000    1.28211e+02
ye          :      0.000   -8.21737e-09

                  inputs
delta_e     :     -0.011
delta_a     :      0.000
delta_r     :      0.000
delta_te    :      0.000
delta_ta    :      0.000
delta_tr    :      0.000
delta_f     :      0.000
landgear    :      0.000
Tn no. 1    :   2410.907
Tn no. 2    :   2410.907

                relative       absolute
cost        :      0.000    1.12286e-18

iterate further   [y/n] : n


end of optimization


save trimmed flight-condition in file   [y/n] : y


current files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file :
```

Figure 3.22: Follow-up display for aircraft trimming tool.

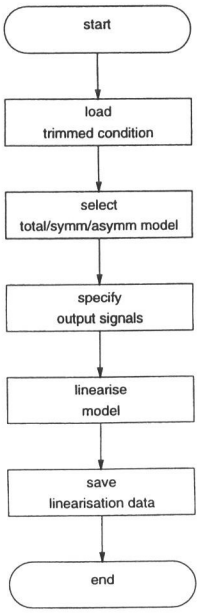


Figure 3.23: Flow-diagram of aircraft linearisation tool `lin_ac.m`.

```
    ************************************************
    **                                            **
    **        Aircraft Linearisation-routine      **
    **                                            **
    ************************************************


set operating point from trimmed flight-condition

available files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file : lr50_1

linearize total/symm/asymm model            [t/s/a] : t

================================== NEW SCREEN ==================================

specify groups with observation outputs

  1. x              7. ypqr           13. yFMaero
  2. xdot           8. yuvw           14. yCgust
  3. yair           9. yuvwdot        15. yFMgust
  4. yacc          10. ydl            16. yCt
  5. yfp           11. yabh           17. yFMt
  6. ys            12. yCaero         18. ygrav

  0. none          19. all

Available option(s) :

  [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 ]

Select option(s) in vector [..] : 12

================================== NEW SCREEN ==================================

specify observation outputs for group : yCaero

  1. CDwind         7. CDstab         13. CTbody
  2. CYwind         8. CYstab         14. CYbody
  3. CLwind         9. CLstab         15. CNbody
  4. CLLwind       10. CLLstab        16. CLLbody
  5. CMwind        11. CMstab         17. CMbody
  6. CNNwind       12. CNNstab        18. CNNbody

  0. none          19. all

Available option(s) :

  [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 ]

Select option(s) in vector [..] : 13:18
```

Figure 3.24: Display for aircraft linearisation tool.

```
loading aircraft-model 'ac_mod', please wait ...

start of linearisation


end of linearisation

save linearization results in file            [y/n] : y

current files : *.lin


select file :
```

Figure 3.24: Continue.

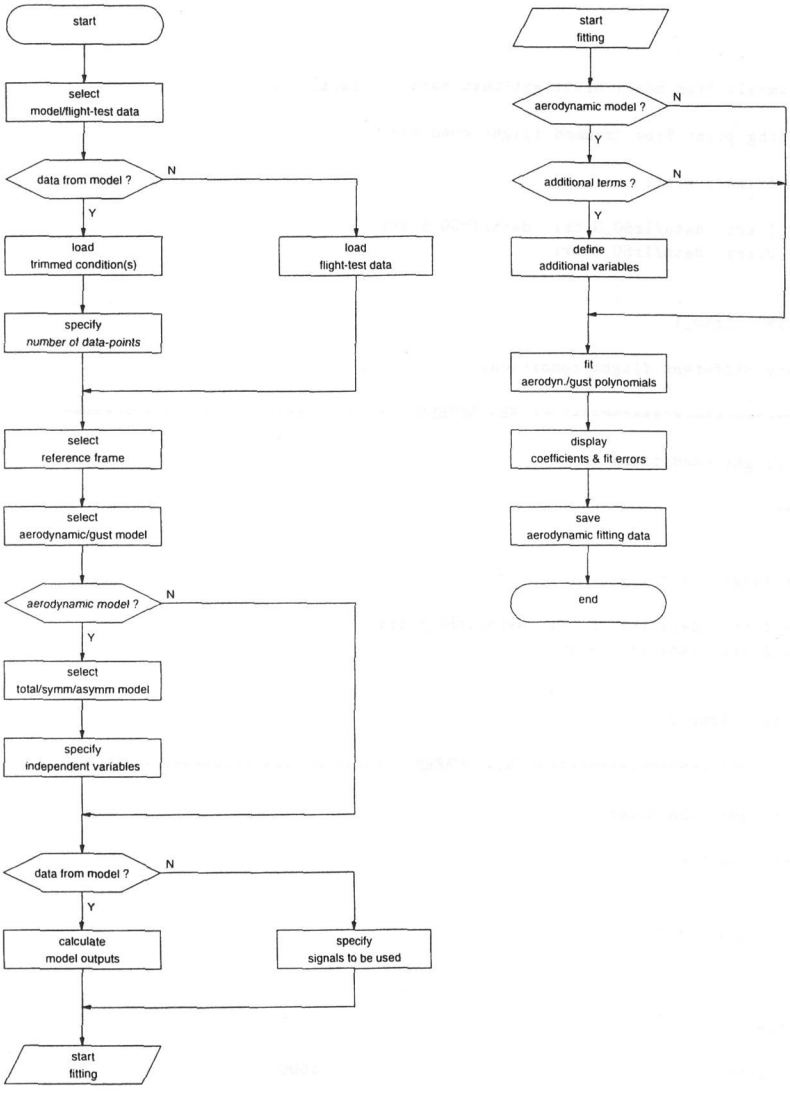


Figure 3.25: Flow-diagram of aerodyanmic fitting tool `fit_aero.m`.

```
      ****************************************************
      **                                                **
      **          Aerodynamic Model Fit-routine         **
      **                                                **
      ****************************************************


fit polynomials from model or flight-test data     [m/f] : m

set operating point from trimmed flight-condition

available files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file : lr50_1

concatenate different flight-conditions       [y/n] : y

================================== NEW SCREEN ==================================

selected flight-conditions :

lr50_1.tri


available files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file : lr50_2

================================== NEW SCREEN ==================================

selected flight-conditions :

lr50_1.tri lr50_2.tri


available files : *.tri



select file :

number of data points                              : 1000

coef. in airpath/stab./body ref. frame      [a/s/b] : b

fit aerodynamic/gust model                    [a/g] : a
```

Figure 3.26: Display for aerodynamic fitting tool using data generated from an aerodynamic model.

```
      ******************************************************
      **                                                  **
      **          Aerodynamic Model Fit-routine           **
      **                                                  **
      ******************************************************


fit polynomials from model or flight-test data      [m/f] : f

load data from simulated flight

available files : *.sim

data/example.sim


select file : example

concatenate different flight-conditions         [y/n] : n

coef. in airpath/stab./body ref. frame       [a/s/b] : b

fit aerodynamic/gust model                     [a/g] : a
```

Figure 3.27: Display for aerodynamic fitting tool using (simulated) flight-test data.

```
fit non-lin./lin.symm/lin.asymm model       [n/s/a] : s

use standard/user-defined variables          [s/u] : u

specify independent variables in polynomial for coefficient : CTbody

  1. constant        7. qc/V          13. delta_te
  2. alpha           8. rb/2V         14. delta_ta
  3. alphadot        9. Mach          15. delta_tr
  4. beta           10. delta_e       16. delta_f
  5. betadot        11. delta_a
  6. pb/2V          12. delta_r

  0. none           17. all

Available option(s) :

  [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ]

Select option(s) in vector [..] : [1 2 3 7 9 10]

================================== NEW SCREEN ==================================

specify independent variables in polynomial for coefficient : CNbody

  1. constant        7. qc/V          13. delta_te
  2. alpha           8. rb/2V         14. delta_ta
  3. alphadot        9. Mach          15. delta_tr
  4. beta           10. delta_e       16. delta_f
  5. betadot        11. delta_a
  6. pb/2V          12. delta_r

  0. none           17. all

Available option(s) :

  [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ]

Select option(s) in vector [..] : [1 2 3 7 9 10]

================================== NEW SCREEN ==================================

specify independent variables in polynomial for coefficient : CMbody

  1. constant        7. qc/V          13. delta_te
  2. alpha           8. rb/2V         14. delta_ta
  3. alphadot        9. Mach          15. delta_tr
  4. beta           10. delta_e       16. delta_f
  5. betadot        11. delta_a
  6. pb/2V          12. delta_r

  0. none           17. all

Available option(s) :

  [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ]

Select option(s) in vector [..] : [1 2 3 7 9 10]
```

Figure 3.28: Follow-up display for aerodynamic fitting tool for defining structure of aerodynamic polynomials.

```
loading aerodynamic-model 'aero_mod', please wait ...

calculating aerodynamic-model responses

nr. of data points to be processed : 1000
processing data point              :    1

================================= OTHER OPTION ================================

specify adot,bdot from observation outputs

flight-condition : example.sim

  1. y1        3. y3        5. y5
  2. y2        4. y4        6. y6

  0. none      7. all

Available option(s) :

  [ 0 1 2 3 4 5 6 7 ]

Select option(s) in vector [..] : 0

================================= NEW SCREEN ==================================

specify force,moment coefficients from observation outputs

flight-condition : example.sim

  1. y1        3. y3        5. y5
  2. y2        4. y4        6. y6

  0. none      7. all

Available option(s) :

  [ 0 1 2 3 4 5 6 7 ]

Select option(s) in vector [..] : 7
```

Figure 3.29: Follow-up display for aerodynamic fitting tool for calculating independent and dependent variables from aerodynamic model or specifying independent and dependent variables in observation outputs.

```
create additional terms in polynomials        [y/n] : y

================================= NEW SCREEN ==================================

define additional variables in 'Upoly1' using columns from 'Uaero'
Press 'return' when finished

K>> Upoly1=Uaero(:,2).^2;
K>> return
```

Figure 3.30: Follow-up display for aerodynamic fitting tool for defining additional terms in aerodynamic polynomials (is also initial display after command `fit_aero1`).

```
number of data points   : 1000

polynomial coefficients :

                constant         alpha       adotc/V          beta       bdotb/V
CTbody       1.1446e-02    3.9910e-02           NaN           NaN           NaN
CNbody       1.2070e-01    5.5491e+00           NaN           NaN           NaN
CMbody       2.4614e-02   -5.3808e-01           NaN           NaN           NaN

                  pb/2V          qc/V         rb/2V          Mach       delta_e
CTbody              NaN    3.7030e-01           NaN    3.3255e-02    6.1082e-03
CNbody              NaN    5.7789e+00           NaN    5.2025e-02    5.7978e-01
CMbody              NaN   -1.0158e+01           NaN    1.8617e-02   -1.2810e+00

                delta_a       delta_r      delta_te      delta_ta      delta_tr
CTbody              NaN           NaN           NaN           NaN           NaN
CNbody              NaN           NaN           NaN           NaN           NaN
CMbody              NaN           NaN           NaN           NaN           NaN

                delta_f
CTbody              NaN   -4.9164e+00
CNbody              NaN    2.7741e+00
CMbody              NaN    1.8816e+00


press any key to list fit analysis statistics

================================== NEW SCREEN ==================================

number of data points   : 1000

fit analysis statistics :

                    RSS           R^2       F-value
CTbody       8.8983e-06    9.9949e-01    3.9180e+05
CNbody       3.6614e-05    9.9999e-01    2.5486e+07
CMbody       1.1891e-04    9.9847e-01    1.2984e+05

                Var(res.)     max(res.)   max(res./y)
CTbody       8.9520e-09    3.4859e-04    1.2227e-02
CNbody       3.6835e-08    6.7255e-04    4.1877e-03
CMbody       1.1963e-07    1.0253e-03    8.3275e-02


save polynomial fits in file                    [y/n] : y


current files : *.aer


select file :
```

Figure 3.31: Follow-up display for aerodynamic fitting tool with fitting results.

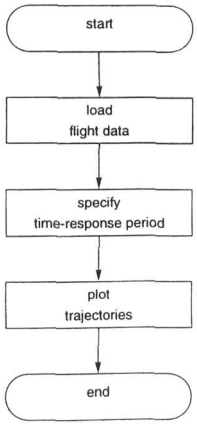


Figure 3.32: Flow-diagram of plotting tool of aircraft simulation time-responses `plot_ac.m`.

```
    ************************************************
    **                                            **
    **        Aircraft Time-Response-routine      **
    **                                            **
    ************************************************


load flightpath from simulated flight

available files : *.sim

data/example.sim


select file : example

time-response period              [<  10.00] (in sec) :  10.00
```

Figure 3.33: Display for plotting tool of aircraft simulation time-responses.

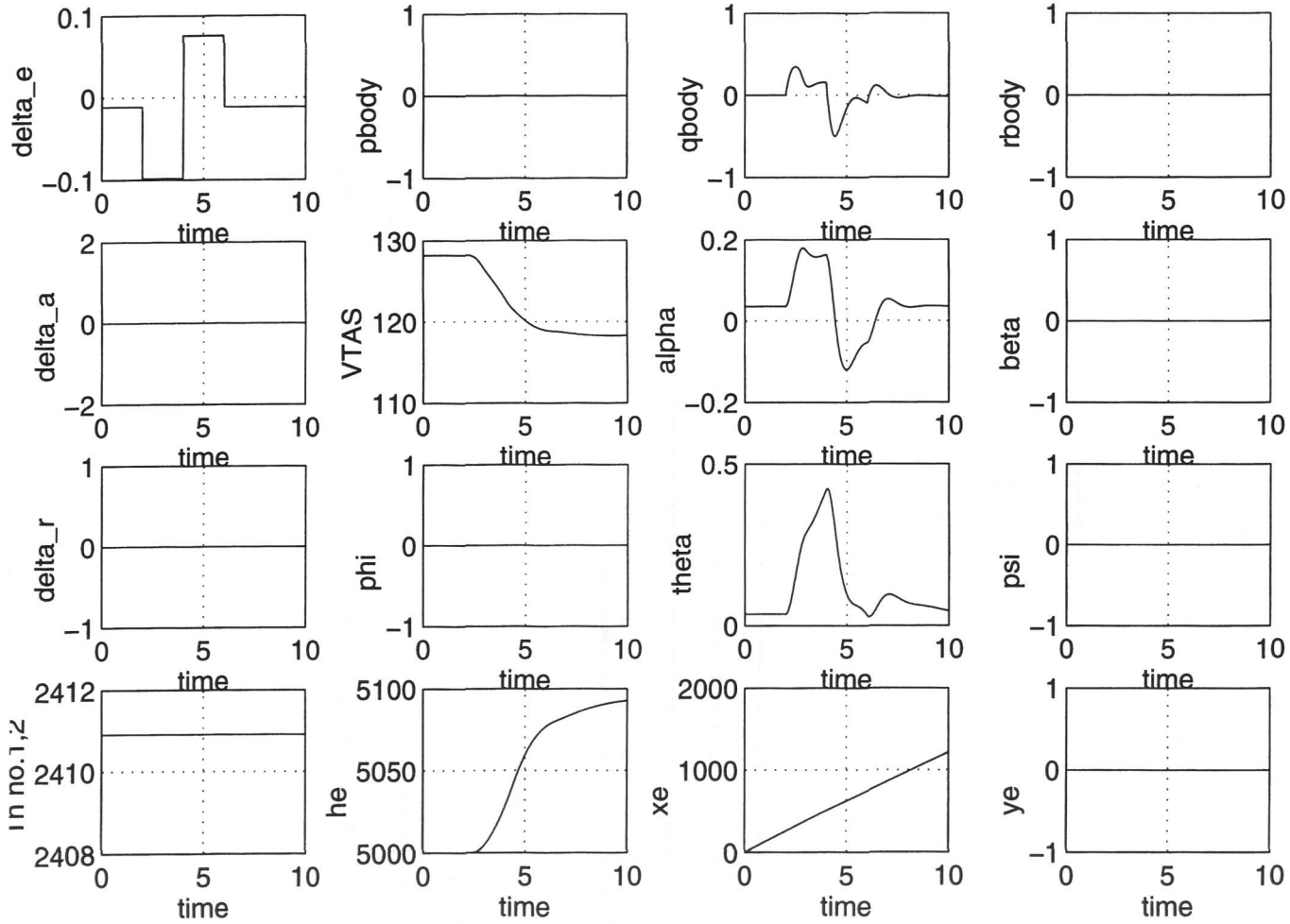


Figure 3.34: Example display of simulated trajectories generated via plotting tool of time-responses.

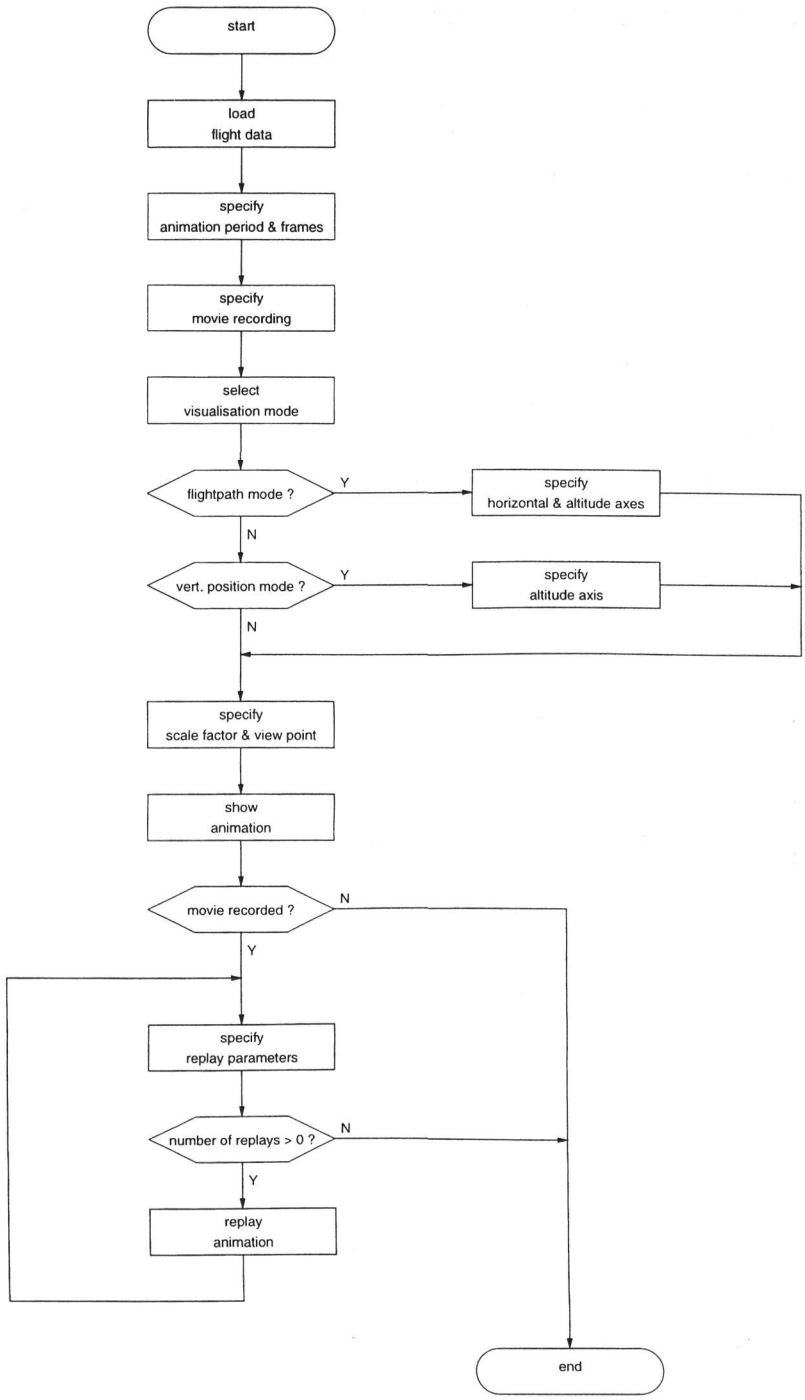


Figure 3.35: Flow-diagram of animation tool of aircraft simulation `show_ac.m`.

Figure 3.34: Example display of simulated trajectories generated via plotting tool of time-responses.

```
    ************************************************
    **                                            **
    **          Aircraft Animation-routine        **
    **                                            **
    ************************************************


load flightpath from simulated flight

available files : *.sim


data/example.sim

select file : example

animation period                [<  10.00] (in sec) :  10.00

number of frames                        [<   1002] : 10

record data for movie                       [y/n] : n

speed of plays                            (in f/s) : inf
```

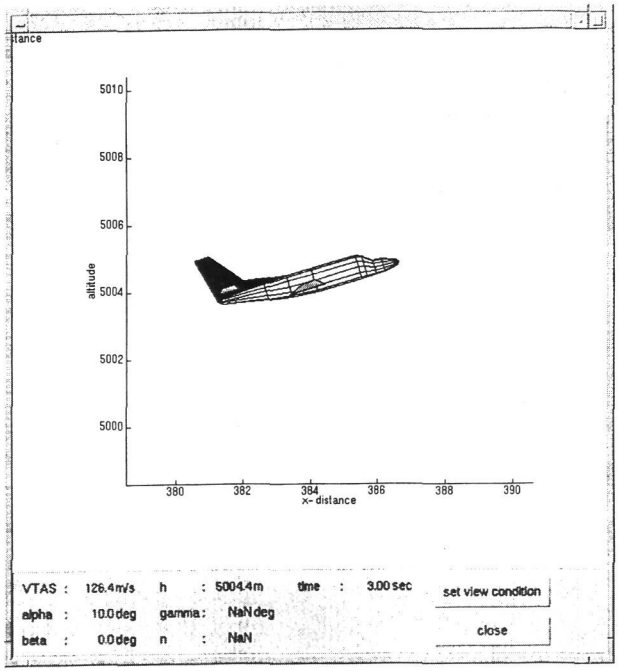
Figure 3.36: Display for animation tool of aircraft simulation.

```
number of replays                                  : 1

speed  of replays                         (in f/s) : 10

number of replays                                  : 0
```

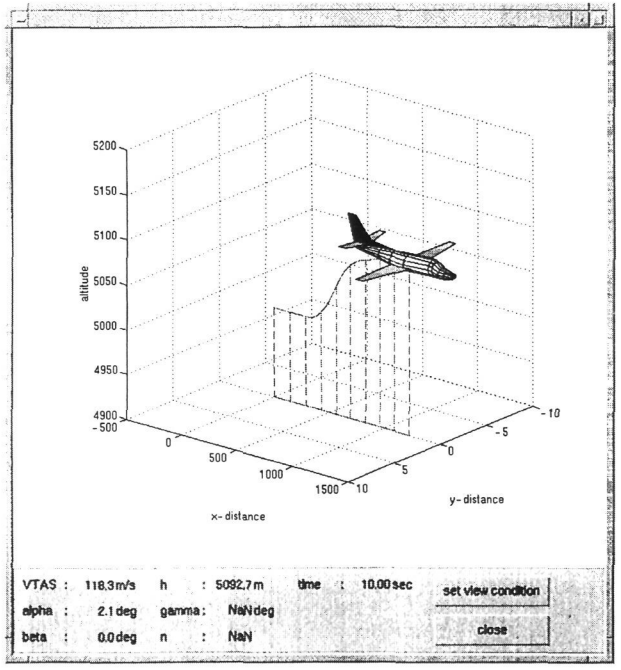
Figure 3.39: Follow-up display for animation tool of aircraft simulation.

```
    ************************************************
    **                                            **
    **            Aircraft Visualisation          **
    **                                            **
    ************************************************


view attitude, vert. position or flightpath   [a/v/f] : a


scale factor                                        : 0.454887

view azimuth                                (in deg) : 40

view elevation                              (in deg) : 20

================================= OTHER VIEW ================================

view attitude, vert. position or flightpath   [a/v/f] : v

min. altitude                                (in m) :   4900

max. altitude                                (in m) :   5200

scale factor                                        : 0.454887

view azimuth                                (in deg) : 40

view elevation                              (in deg) : 20

================================= OTHER VIEW ================================

view attitude, vert. position or flightpath   [a/v/f] : f

Retain a/c graphs in plot                      [y/n] : n

min. altitude                                (in m) :   4900

max. altitude                                (in m) :   5200

min. x-distance                              (in m) :   -500

max. x-distance                              (in m) :   1500

min. y-distance                              (in m) :    -10

max. y-distance                              (in m) :     10

scale factor                                        :  0.75188

view azimuth                                (in deg) : 40

view elevation                              (in deg) : 20
```

Figure 3.37: Follow-up display for animation tool of aircraft simulation (identical to Figure 3.12).

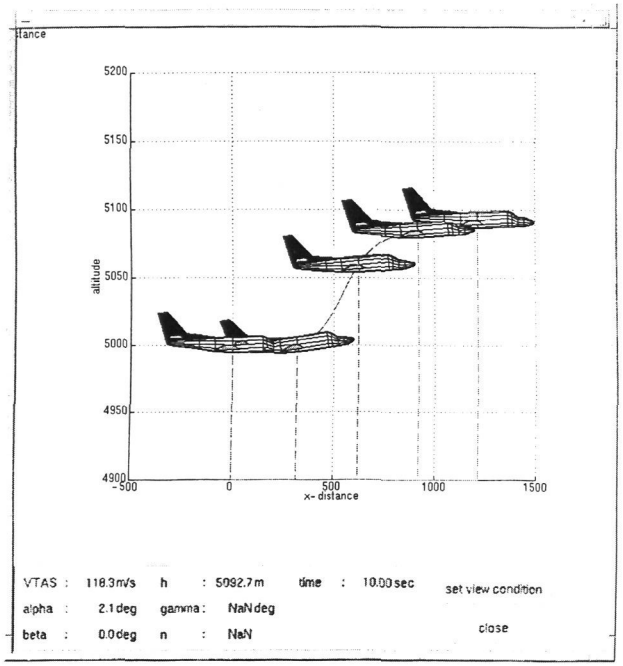


Example display of simulated response using option for attitude view.



Example display of simulated response using options for flight-path view and no retaining a/c graphs.

Figure 3.38: Screen lay-out for aircraft simulation displayed as animation. (identical to Figure 3.13).



Example display of simulated response using options for flight-path view and retaining a/c graphs.

Figure 3.38: Continue.

Chapter 4

Control Design Applications

4.1 Introduction

A main application of the *DASMAT* tools discussed in the previous chapters is to design and evaluate flight control systems. This chapter now focuses on the framework within *DASMAT* for feedback control design.

The design process of flight control systems can be subdivided into several phases, which should be passed along iteratively. Two major phases consist of the synthesis of the controller and the simulation of the controller on the aircraft dynamics. Flight control systems are generally synthesized for linear systems at various operating points in the flight envelope. These linear systems may be obtained from the trimming and linearizing tools in the *DASMAT* package. The MATLAB environment subsequently provides a number of toolboxes which may be applied for designing the controller using a user-desired design procedure. Finally, the controller can be used again with *DASMAT* tools to perform nonlinear simulation and analysis.

Various design procedures exist for controller design and may therefore also be applied for flight control system design. First, there are classical design techniques such as root locus and Bode analysis, which select feedback gains to place closed-loop poles at locations which guarantee desirable time responses and robustness qualities. These techniques are directly applicable for SISO systems, whereas MIMO systems require successive closures of individual loops. The design procedure is straightforward and may be executed in both the MATLAB and the SIMULINK environment by building closed-loop systems with adjustable feedback gains, calculating the pole locations and performing time simulations.

Modern control techniques close the multiple feedback loops in one step and simultaneously compute all the feedback gains. The control gains are now solved from matrix equations which arise from expressing performance specifications in terms of a performance criterion. An important approach is model following, where the criterion penalizes the response deviations of the closed-loop system from an ideal aircraft model with desirable flying qualities. Another important approach is the application of robust analysis in the frequency domain approach. Widely applied techniques in aircraft control are Linear Quadratic Regulator/Loop-Transfer Recovery (LQR/LTR) techniques. Within MATLAB, the *Control* and *Robust Control* toolboxes provide M-functions for designing controllers with modern control techniques.

A very recent approach to feedback design is given by modern robust design methodologies

as H_∞/μ-synthesis. Controllers designed this way can deal with system uncertainties, i.e. they still provide stability and perform well in conditions which are off from the design operating point. The uncertainties may arise from parameter variations or disturbances and are expressed in a frequency-domain robustness measures by using the notion of the singular value. The controllers may be designed from the M-functions in the *μ-Analysis and Synthesis* toolbox in MATLAB.

Once a controller has been synthesized, simulation of the control system on the nonlinear aircraft dynamics becomes an essential part of the design process. Nonlinear control elements such as rate and deflection limiters on control surface actuators or time-delays in sensor measurements may now be added in the closed-loop model before running a simulation. Depending on the evaluated performance, any adjustments can then be made to the controller design. The actuator limiters and sensor delays may be modelled independently in the actuator and sensor model respectively and then added in a closed-loop with the nonlinear aircraft model. The closed-loop model may then be used for the simulation in SIMULINK, where a reference signal may be generated by means of an independently added signal generator.

This chapter first describes the SIMULINK models which are used in the closed-loop model structure. The operation of a closed loop simulation is discussed next. For application of the MATLAB toolboxes to design the controller, the user is referred to the manuals for these toolboxes [9].

4.2 Simulink models in closed-loop system

For flight control systems, the open-loop aircraft model is included in a feedback system and its control inputs are commanded by a flight controller. Besides, the general feedback structure also includes sensor and actuator models and the closed-loop system itself is commanded by a reference input. The models in the closed-loop flight control system can therefore be subdivided into the following elements:

- aircraft model
- controller model
- sensor model
- actuator model
- reference signal generator

The closed-loop system is provided by a SIMULINK S-function which serves as an operating shell. The models in the closed-loop system are independent SIMULINK S-functions which are imbedded in the SIMULINK model of the operating shell.

All models in the closed-loop system are user-defined, except for the aircraft model. The aircraft model is either `ac_mod.m` or `ac_modpc.m`. The other models in the closed-loop system are specified in the closed-loop specification routine designated by the variable `cl_info`, the same way as the configuration of the generic aircraft model is dealt with in the aircraft specification routine designated by the variable `ac_info`.

4.2.1 Operating shell for closed-loop aircraft simulation

Default operating shells for closed-loop simulation in the SIMULINK window are available via the S-functions `cl_sim.m` and `cl_simpc.m`. They have already been shortly discussed in Section 2.3.11 *Operating shells for aircraft simulation* where remarks are also given for adapting operating shells to serve user-defined wishes.
The SIMULINK S-functions of the operating shells are shown in Figure 4.1.

The operating shells provide a framework for including the aircraft and a controller model plus additional models in a feedback structure. They also serve as interface for simulating and analyzing the closed-loop system.
All models in the operating shell are all included via *S-function* blocks. The models are user-defined except for the aircraft model, which is related to the operating shell because of the format of the observation outputs. The general aircraft model `ac_mod.m` is included in `cl_sim.m`, while `ac_modpc.m` is used in `cl_simpc.m`. It is assumed that the aircraft model always includes the engine models for providing the thrust, i.e. has the throttle (power lever angle) settings as thrust control.
The other models in the closed-loop, i.e. the models for the controller, actuators, sensors and reference signals generator, are designated by a standard named variables which refer to SIMULINK S-functions. These variables are assigned at the start a closed-loop simulation by executing the closed-loop specification routine designated by `cl_info`, see Section 4.3 MATLAB *routine for closed-loop specification.*
The closed-loop also contains a *Transport Delay* block. This block is included to prevent the presence of an algebraic loop, which would arise because all models in the feedback structure have direct feed-through signals. An algebraic loop considerably reduces the speed of a simulation because it requires an iterative solution during each integration step. Besides, processing measurement signals always takes an amount of time and transport delays are therefore always present in physical closed-loop systems. The transport delay is set via the variable `delay` which has default value 10^{-2}.

The control inputs u and reference signals r are written to the MATLAB workspace via *To Workspace* blocks. They become available as variables `u`, `ut` and `r`, having the same integration time points as the state variables and observation outputs. It should be noted that u always contains throttle (power lever angle) settings $u_{t_i} = PLA_i$ as thrust controls because the engine model is included in the aircraft model.
The observation outputs y may become available in three ways. The blocks with labels *Response* and *Animation* (only available for the general aircraft simulation via `cl_sim.m`) contain S-functions which on-line visualize the dynamic responses as a 3D-animation or as time-history plots. They are activated when the corresponding option was selected during the dialog for performing the closed-loop simulation, see Section 4.4 *Closed-loop simulation in* SIMULINK *window.* The observation outputs are also demultiplexed and reduced in two steps to scalar signals. The first demultiplexing step separates y into 18 groups for the general aircraft model `ac_mod.m` according to (2.6), and into 9 groups for its compact version `ac_modpc.m` according to (2.7). Any second step separates the selected groups further into the scalar observations in Table B.3 and Table B.4. Scalar signals connected to an *Outport* block are written to the MATLAB workspace and included in the variable `y`, which is defined as return variable in the Control Panel dialog box and which is displayed by selecting **Parameters** from

the **Simulation** menu.
Moreover, the time vector t and the complete state vector x_{cl} of the closed-loop system, i.e. aircraft states x_a, the states x_t of all engines and the states of the controller, actuators and sensors, are defined as return variables. As such, they are written to the workspace as variables `t` and `xcl`.

The SIMULINK window further contains two *Button* blocks, a *Clock* block, a *Floating Scope* block and a *Control Mode Panel* block-array. The red *Button* blocks are activated by double clicking. The left *Button* block then configures the closed-loop system by including the closed-loop models designated by variables assigned in the M-script of the closed-loop specification routine designated by `cl_info`, see Section 4.3 MATLAB *routine for closed-loop specification.* The right *Button* block configures the generic aircraft model by including the aircraft and condition specific sub-models specified in the M-script of the aircraft specification routine designated by `ac_info`, see Section 2.4.1 *Aircraft specification routine.* After successful completion each block turns green. The *Clock* block provides a window that continuously displays the elapsed time as the simulation progresses. The *Floating Scope* block may be used to display on-line the activity of any block-connection.
The blocks in the *Control Mode Panel* engage standard autopilot modes or disconnect a control channel, see also Section 4.2.3 *Controller model.* When a block is activated, values are assigned to standard named variables in the MATLAB workspace which may be used within the controller model to operate switches for controlling the applicable equations.

The default operating shell only includes the basic elements of the closed-loop system of a flight control system. For actual use, the SIMULINK model of the operating shell may be enhanced with additional signals and models.
A very common extension to the default SIMULINK model is the insertion of additional disturbance and measurement error signals. These signals may for example be generated as white noises in *Random Number* blocks from the simulink/Sources library and then summed to the signals connecting the models. If desired, any additional signal may be passed through a filter before the summation to make it frequency dependent. The additional signals may of course also be included in the imbedded models themselves. This is for example the case for the wind and turbulence, which are included in the generic aircraft model via the wind and turbulence models, see Section 2.3.9 *Condition specific wind model* and Section 2.3.10 *Condition specific turbulence model.* Further, disturbances may also be modelled in the reference generator model, see Section 4.2.6 *Reference signal generator.*
For analyzing robust performance of the controlled aircraft system, the models in the closed-loop system are not fixed to one model but to a set of models which cover the uncertainty of such a model. Additionally, the actual inputs and outputs for the various models are weighted to make the so created new inputs and outputs suitable for the desired performance specifications. These extra perturbation models and weighting functions may of course be integrated with the actual models and leave the operating shell unchanged. However, for the integrity of the models and accessibility of the new elements by the user, it is advisable to have the original model included in the operating shell, treating it as a nominal model, and to add the perturbation models and weighting functions to the operating shell. Figure 4.2 gives an example of how this may be accomplished for the actuator model. The newly added signals and blocks allow easy and direct adjustments of the closed-loop performance specifications by tuning the weighting functions and control over the actuator model uncertainties by modifying

its perturbations.

4.2.2 Aircraft model

The aircraft model consists of the generic aircraft model in the SIMULINK S-functions `ac_mod.m` and `ac_modpc.m`. These models are adapted for a specific aircraft and condition by including aircraft and condition specific sub-models, designated by variables which are assigned in the aircraft specification routine designated by `ac_info`. A detailed description of the aircraft models is given in Section 2.3.3 *Generic aircraft model.* The aircraft specification routine is described in Section 2.4.1.

4.2.3 Controller model

The controller is an independent SIMULINK S-function which determines the control inputs to the aircraft model (via the actuator model), based on the commanded reference signals and the controlled signals of the aircraft responses (via the sensor model). The controller model is designated by the variable `cl_controlmodel`.

The tasks of a flight controller can be subdivided into three categories, depending on the responsiveness of the aircraft to manoeuvring commands and required pilot activity. For very fast modes where the pilot would find it difficult or impossible to control the aircraft if they were lightly damped or unstable, the control system is known as stability augmentation system (SAS). This control system is simply a feedback control system designed to increase the relative damping of a particular mode. It is generally imbedded in the next mentioned control systems. If the augmentation system is intended to control a mode and to provide the pilot with a particular type of response to the control inputs, it is known as control augmentation system (CAS). Finally, there is an integrated automatic control system which takes over any manual pilot control in order to provide pilot relief. It is generally referred to as the autopilot. This control system offers a collection of control modes from which the pilot may select to suit the task required for a particular phase of flight.
The above mentioned control modes are generally integrated in one controller. For each mode, the appropriate control law is then made operational via logical switches. Common autopilot modes are:

- Pitch-Attitude Hold (PAH)
- Altitude Hold (ALH)
- Altitude Select (ALS)
- Mach Hold (MH)
- Glide-slope (GS)
- Autothrottle
- Roll Attitude Hold (RAH)
- Heading Hold (HH)

- Navigation (NAV)

The PAH mode is normally used to maintain the pitch angle, leading to a climbing flight when thrust is increased or aircraft weight decreases (smaller angle of attack required for vertical force equilibrium). The ALH mode allows the aircraft to be held at a fixed altitude, while the ALS mode controls the aircraft in such way that the selected altitude is arrived at a prescribed climbing speed. Generally, the ALS and ALH modes automatically follow each other when the aircraft is commanded to change altitude. The MH mode holds the aircraft at a specified Mach number. The GS mode controls the aircraft along the reference glide-path during automatic landing. All these modes are similar in that they mainly use elevator control for controlling the aircraft dynamics. The longitudinal control modes further include the autothrottle mode which controls the engine throttle settings to maintain a reference air speed. This mode may be simultaneously selected with another mode, where its combination with the GS mode is common practise.
As far as the lateral control modes, the RAH mode maintains a selected roll angle. It is also often applied as inner loop for other autopilot modes to allow the aircraft to fly on a fixed compass heading (HH mode) or home on an omnidirectional radio-navigational beacon (NAV mode). The latter may either belong to a VOR or ILS localizer system. The lateral modes generally effect the aileron and rudder controls simultaneously.
A possibility for selecting the autopilot mode(s) is by setting switches inside the controller. When the controller is represented as a SIMULINK model, this may be done by including *Switch* blocks from the simulink/Nonlinear library and controlling the switches through variables in the MATLAB workspace. The variable list of *DASMAT* contains standard named variables which may be assigned by clicking buttons in a *Control Mode Panel* in the closed-loop operating shell, see Figure 4.1. The variables assigned by the buttons are listed in Table 4.1. Various modes may be active simultaneously, for example GS and Autothrottle on, or a control input may be completely disengaged from the controller, for example elevator off.

The controller may be synthesized with any method, e.g. using classical or modern (robust) control theory. The structure within the controller is free as long as the controller inputs and outputs correspond with the connected models. Further, the controller needs not to be linear and may incorporate nonlinear elements such as limiters and saturation functions.
The imbedding of the controller in the closed-loop system allows great flexibility for choosing the internal structure of the controller. A two degrees of freedom controller may be synthesized, having a separate feedback filter and a pre-filter on the reference signal. Figure 4.3 gives some examples. The pre-filter may included as a feed-forward compensator to cancel measured disturbances or it may represent a model for ideal closed-loop system response. The feedback filter is generally used for stability augmentation and regulation. Feedback filters may appear both in the feedback path and the forward path. In classical control, the feedback path generally contains gains, scheduled by additionally feedback signals, and the forward path may have a dynamic compensator, for example a proportional-plus-integral (PI) compensator. In modern control, e.g. LQG, the forward path may contain a compensator consisting of a series connection of a Kalman filter with a state-feedback matrix.
Information on the synthesis of flight controllers can be found in various textbooks and papers. Classical flight control design methods can be found in [15, 11]. Modern flight control design techniques are also described in [15], while [8] treats the methods for multivariable control systems in general. The control design techniques using H_∞/μ-synthesis may best be

reviewed from MATLAB's *μ-Analysis and Synthesis* toolbox [3].

For implementation on an aircraft, the control gains in the flight controller should be gain scheduled. The aircraft dynamics vary over the flight envelope, whereas the controller performance should remain nearly invariant. Because the gains are derived from the aerodynamic characteristics, gains and forms of control laws which give a satisfactory response in one flight condition may lead to inadequately damped, or even unstable, dynamics in another flight condition. The gains and if necessary the control laws therefore need to be adapted according to a representative schedule depending on the flight control function. The scheduling variable will normally be measured dynamic pressure but may involve other variables in more complicated cases, e.g. angle of attack, Mach number, air density, etc.

The inputs to the controller can be subdivided into two groups. The first group consists of the reference inputs which are generated by the reference signal generator. The second group consists of feedback signals supplied by the sensor model. The feedback signals may be applied for following the reference inputs, i.e. for constructing an error signal to be submitted to a control gain, or they may be applied for scheduling the gains. An example of how a scheduled gain may be implemented in SIMULINK is shown in Figure 4.4.
The outputs of the controller consist of signals which drive the actuators. Generally, the outputs involve elevator, aileron and rudder control plus thrust control for each engine. Thrust control is by default provided through engine throttle (power lever angle) setting. All control signals are regarded as additive signals to the nominal values in the operating point.

An example of a controller model is shown in Figure 4.5. In this model, the controller inputs are first split into reference signals, feedback signals for tracking the reference signals and feedback signals for scheduling the gains. All gains are then scheduled in a separate subsystem. The error signals are then processed with the gains in separate longitudinal and lateral controller sub-systems. The longitudinal controller determines the elevator and engine throttle (power lever angle) settings, while the lateral controller provides the aileron and rudder controls.

4.2.4 Actuator model

The actuator model is an independent SIMULINK S-function which contains the dynamics of all actuators or servomechanisms that move the aerodynamic control surfaces and adjust the engine throttle settings. If desired, the mechanical, hydraulic and electrical components of the control system may also be modelled here. The actuator model is designated by the variable `cl_actuatormodel`.

Actuator systems have their own dynamic characteristics which affect the performance of the closed-loop system. In comparison with the aircraft dynamics, the dynamic response of the actuator is generally very rapid. However, for large aircraft and/or high speeds the actuator must provide large hinge moments for control and the response is not instantaneous. It is therefore customary to represent the actuator dynamics via a first order lag filter, where the gain and time constant depend upon the actuator's characterization. It should however be noted that the actual actuator dynamics contains higher order terms and nonlinearities. These effects should always be studied during final simulation tests.

Some information on the nature of the dynamic characteristics of electric and electrohydraulic actuators can be found in [11].

The inputs of the actuator model consist of the commanded signals supplied by the flight controller. The control channels usually consist of the primary flight controls and the engine throttle settings. The format of the channels may be user-defined, as long as it is in conformity with the output format of the applied controller model. The actuator inputs are further regarded as additive signals on the nominal values in the operating point.
The outputs of the actuator contain all control inputs of the aircraft model. Next to the controlled channels, the outputs therefore also include the secondary flight controls. Further, the actuator outputs always have a fixed format corresponding to the representation of the control inputs of `ac_mod.m` and `ac_modpc.m`. With the engine model included in the aircraft model this yields for the actuator outputs $y_{act} = u$, see also Section 2.3.3 *Generic aircraft model* and Table B.2:

$$y_{act} = u = [u_a \;\; u_{t_1} \;\; u_{t_2}] \tag{4.1}$$

where:

$$\begin{aligned} \texttt{u} \quad &= \quad u_a \quad = \quad [\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}] \\ \texttt{ut(i)} \quad &= \quad u_{t_i} \quad = \quad PLA_i \end{aligned} \tag{4.2}$$

An example of an actuator model in SIMULINK is given in Figure 4.6, which links up with the controller model in Figure 4.5. Within the model, the actuators for the primary flight control surfaces are independently modelled by three first order lag filters and the engine throttle settings are provided as two direct feed-through signals. The trim conditions are further included by means of constant values, read from the MATLAB workspace.

4.2.5 Sensor model

The sensor model is an independent SIMULINK S-function which feedbacks the aircraft responses to the flight controller. It models the dynamics of each measurement instrument, usually a transducer. Most common sensors are gyroscopes and accelerometers. The sensor model is designated by the variable `cl_sensormodel`.

A sensor transfers a motion variable into a signal applicable for the controller. Some electronic sensors process the information so quickly in comparison with the aircraft's response that it is customary to regard their transducing action as instantaneous. However, sensors often have built-in filters to improve their noise characteristics and the time constants of such filters are then considered as representing that of the sensor. Aside from the sensor dynamics, another important aspect of the sensor performance is its location in the aircraft, especially when an elastic aircraft is considered.
A brief treatment of linear mathematical models of the principal sensors is presented in [11].

Because a large number of observation outputs are calculated in the aircraft model, the sensor model merely only needs to model the measurement dynamics. Any measured signals which

are not yet modelled in the aircraft model should first be constructed from the supplied observation outputs.
If no sensor dynamics is included, the SIMULINK S-function of the sensor model is best supplied by C-code. The direct feed-through relationship between the sensor inputs and outputs may simply be accomplished by re-indexing the input channels to the desired output channels. A pre-requisite is that such C-code should be seperately compiled to a MEX-file for each different computer platform.

The inputs of the sensor model consist of all modelled observation outputs of the aircraft model. The format of these inputs depends on the applied aircraft model, either the general model `ac_mod.m` or its compact version `ac_modpc.m`. The representation of the sensor inputs $u_{sens} = y \;/\; y_{red}$ thus becomes:

$$\mathtt{y} = \begin{cases} y = [x_a \;\; \dot{x}_a \;\; y_{air} \;\; y_{acc} \;\; y_{fp} \;\; y_s \;\; y_{pqr} \;\; y_{uvw} \;\; y_{\dot{u}\dot{v}\dot{w}} \;\; y_{dl} \;\; y_{\alpha\beta h} \\ \qquad y_{Caero} \;\; y_{FMaero} \;\; y_{Cg} \;\; y_{FMg} \;\; y_{Ct} \;\; y_{FMt} \;\; y_{Fgr}] & \texttt{(ac\_mod.m)} \\ y_{red} = [x_a \;\; \dot{x}_a \;\; y_{air} \;\; y_{acc} \;\; y_{fp} \;\; y_s \;\; y_{FMaero} \;\; y_{FMg} \;\; y_{FMt}] & \texttt{(ac\_modpc.m)} \end{cases} \tag{4.3}$$

The outputs of the sensor model consist of the controlled feedback signals plus any required signals for scheduling gains in the controller. The format of the signals is user-defined, as long as it corresponds with the inputs of the applied controller.

Figure 4.7 gives an example of a sensor model which links up with the controller model in Figure 4.5. The model does not contain any sensor dynamics and its SIMULINK S-function is therefore written as C-code. The aircraft states and some accelerations are instantaneously fed back to the controller.

4.2.6 Reference signal generator

The reference signal generator is an independent SIMULINK S-function which supplies the desired aircraft responses the controller should try to induce. When the controller has a regulator task, the reference signal generator provides either zero signals or no signals at all. Aside from the reference signals, any disturbances may also be modelled here. The reference signal generator is designated by the variable `cl_refmodel`.

In the real world, the reference signals may come from a guidance system, may be commanded by the pilot, or they may be zero. These signals are closely related to the various tasks of the controller, see Section 4.2.3 *Controller model.* Zero signals are related to the SAS, while commanded signals are generally related to the CAS but may also be used with some autopilot modes. Signals from guidance systems generally always control autopilot modes.

Various types of reference signals may be distinguished. The most simple reference signals have constant values, for example zero. Examples of these signals are glide-slope deviations or speed and altitude reference values. These types of reference signals are applied with the various autopilot hold modes itemized in Section 4.2.3.

Very common applied reference signals consist of step functions. A main application is using them as test signals for analyzing the controller performance via time-domain nonlinear simulations. By comparing the reference step input with the step response of the controlled aircraft, various standard performance quantities may be derived such as rise time, peak time, settling time and overshoot.
Most complete reference signals may consist of a complete flight trajectory the aircraft has to follow. Such a trajectory may consist of a flight-path to be flown, complete with disturbances the aircraft encounters at pre-defined points along the trajectory. The reference signals are then usually modelled as functions of time or position, where the latter requires special attention because SIMULINK uses time as independent variable during a simulation. The reference trajectory may include signals as aircraft's position, commanded air speed, commanded heading, switches to (dis)engage a controller mode or generate disturbances.

The applied blocks in the reference signal generator usually consist of *Constant* or *Step Input* blocks from the simulink/Sources library. Disturbances may be generated via a *Random Number* block. More complicated reference signals may first be generated off-line and then applied for closed-loop simulation via the blocks *From Workspace* or *From File*.
It is advised to make any reference signal generator model as complete as possible. For example, many *Step Input* blocks may be included where they will never be applied simultaneously. If no step signal is desired for any reference signal, the final value in the *Step Input* block is simply set to the initial value. Further, any reference signal is best modelled as a deviation from the trimmed value. These trimmed values may be read from the MATLAB workspace.

The reference generator generally has no inputs.
The outputs of the reference generator have no prescribed rule. Their format may be user-defined. The only requirement is that their format should agree with the applied controller model.

An example of a reference signal generator is presented in Figure 4.8. The generated reference signals correspond with the required inputs of the controller model in Figure 4.5. A number of reference signals consist of constant values, some referring to trimmed values. Further, a number of *Step Input* blocks are present to generate step signals from trimmed values. The available step signals allow testing the various autopilot modes itemized in Section 4.2.3.

4.3 Matlab routine for closed-loop specification

The closed-loop specification routine is a MATLAB M-script which configures the closed-loop system. Via this routine specified controller, actuator, sensor and reference generator models are put in a feedback structure supplied by a closed-loop operating shell. The routine designates the names of the models in the closed-loop system to standardized variables. These variables are then used for calling the SIMULINK models in *S-function* blocks in the SIMULINK model of the operating shell.
Additionally, the routine may provide initialization commands to be executed before a simulation should be started. Such commands may load data-files with variables to be used in the various models, calculating any reference values, etc.

The basic lay-out of MATLAB M-script is given in Figure 4.9. The statements should be completed with strings for the file names. The M-file should be user-supplied and be placed in the operational work-directory.

The routine is designated by the variable `cl_info`. This variable is defined after double clicking the left *Button* block in the operating shell, see Figure 4.1, or during the execution of simulation tool before the closed-loop simulation is started. The desired closed-loop specification routine may then be selected from the work-directory, after which it is immediately executed, see also the next section.

4.4 Closed-loop simulation in Simulink window

The aircraft simulation tool provides, next to the open-loop simulation, also closed-loop simulation for a specified aircraft in a feedback structure with user-defined models for the controller, sensors, actuators and reference signals. The aircraft model takes aerodynamic and thrust controls as inputs from the actuator model and generates a variety of observation outputs using the aircraft model described in Section 2.3.3 *Generic aircraft model.* These outputs are both directly accessible to the user and ported to a sensor model. The closed-loop model may be configured with any combination of models described in Section 4.2 SIMULINK *models in closed-loop system*, as long as the input/output format of these models agree with each other.
The closed-loop simulation tool generally offers the same features as with open-loop simulation described in Section 3.3 *Simulation of Aircraft.* The user has the possibility of configuring both the aircraft and closed-loop simulation model and the display of observation outputs. Various (user-defined) wind and turbulence models may be selected. The engine model is however assumed to be always included in the aircraft model, thus using the throttle setting as control input only. All control inputs are further determined through feedback control without user-interference during simulation. The reference inputs to the closed-loop system are however always accessible in the reference generator model via the standard SIMULINK interface. The dynamic responses of the aircraft may be visualized on-line through a 3D-animation or as time-history plots. They may further be saved in workspace or in data-files for off-line analysis. Finally, the user is offered a great flexibility in defining the models and in selecting the observation outputs.

This section describes the available options for starting the aircraft simulation and running the closed-loop aircraft simulation. The procedure for open-loop simulation is described in Section 3.3 *Simulation of Aircraft.* A step by step description is given of the user-supplied data, the executed routines and the screen displays.
The simulation tool for closed-loop is provided in the MATLAB M-scripts `sim_ac.m` and `sim_ac3.m`. The first script starts up the simulation and gives access to all simulation ,i.e. both open-loop and closed-loop simulation. The second then contains the commands for running the closed-loop simulation in the SIMULINK window. The M-function `ac_anim*.m` provides the commands for on-line animation, and the S-function `ac_resp.m` pops up an array of scope windows for displaying the aircraft states. The default operating shells can handle both the general and compact aircraft model in the S-functions `ac_mod.m` and `ac_modpc.m` respectively. They are provided by the S-functions `cl_sim.m` and `cl_simpc.m`. The flow diagram of the

simulation tool is given in Figure 4.10.

The simulation of the aircraft may either be started by clicking **Simulate 'aircraft'** in the window **DASMAT - MAIN MENU** from Figure 3.1 after starting the package from `dasmat`, or directly by the command `sim_ac` from the MATLAB prompt.
The user may also directly open an operating shell for running the closed-loop aircraft simulation in the SIMULINK window by entering their names from the MATLAB prompt, i.e. `cl_sim` or `cl_simpc`. The aircraft model and the closed-loop system should then be configured by double clicking the *Button* blocks *Set Aircraft Model* and *Set Closed-Loop model* in the SIMULINK window. However, just the models specified by standardized variable names, see Table C.1, will be implemented in the generic aircraft model and the closed-loop operating shell. Furthermore, a trimmed flight-condition should already have been loaded in the MATLAB workspace. After successful configuration of the aircraft model and the closed-loop model, the clicked *Button* block turns from red into green. The simulation may then be started via the procedure described below.

After the simulation tool is started the user is returned to the command window where the display of Figure 4.11 is generated step by step. The initial dialog is identical for open-loop and closed-loop simulation. As a first step the starting point must be set from a flight-condition which is saved in a trim-file `.tri`. If no trim-file is entered, the simulation routine stops and one returns to the MATLAB prompt.
The next input asks for running the open-loop or the closed-loop simulation. The selection of open-loop simulation lets the control inputs be provided by the user, see further Section 3.3. The selection of the closed-loop simulation, which follow-up procedure will be treated here, includes the aircraft model in a feedback loop and the control inputs are now generated via a controller.
The next two inputs select the wind and turbulence conditions to be used. The S-functions for these models are described in Section 2.3.9 *Condition specific wind model* and Section 2.3.10 *Condition specific turbulence model.* If a non-zero wind condition is selected, the user may include an S-function for the wind model which should be created beforehand and kept in the current work-directory. If a non-zero turbulence condition is selected, the S-function `tur_dryd.m` from the *DASMAT* package is included. This model generates gust velocities from white noise using Dryden spectra. When no wind and no turbulence are selected, the S-functions `wnd_none.m` and `tur_none.m` from the *DASMAT* package are included and all wind and gust velocities become zero.

From this point, the dialogs for open-loop and closed loop simulation go separate ways, see Figure 4.10. For open-loop simulation, the dialog continues with the specification of the control inputs and the running procedure, see Section 3.3. For closed-loop simulation, the follow-up procedure is described below.

The user should next select how the responses should be visualized during the simulation. The responses may be viewed as animation, time-responses or as model-activity. The option for animation provides a 3-dimensional picture of the aircraft motion along the flight-path in an opened **Animation** window. The options for time-responses and model-activity display the time-traces of all aircraft states or any block-connection in the SIMULINK window respectively by means of scope windows. More details are given in Section 3.3.4 *Specification of simulation*

visualization.
The following question lets the user specify the operating shell for the simulation. The default operating shell for closed-loop simulation is either the S-function `cl_sim.m` or `cl_simpc.m`, see Section 4.2.1 *Operating shell for closed-loop aircraft simulation* and Figure 4.1, depending on whether the general or compact aircraft model was selecting during initialization. If desired, the default operating shell may be adjusted just before a simulation is started, see below. Instead, a user-defined operating shell may be selected from the current work-directory. However, the user should be sure that its data format agrees with that of the included aircraft model, see also the remarks in Section 2.3.11 *Operating shells for aircraft simulation.* The operational operating shell is designated in the variable `cl_simmodel`.
The simulation parameters are specified next. These are the stop time, which is actually the duration of the simulation, and the minimum and maximum step sizes of the integration. The step size itself may vary per integration step and is controlled by the relative error of the integration at each step (default 10^{-3}) which on its turn depends on the operational integration method (default fifth order Runga-Kutta `rk45`). To obtain a fixed step size the maximum step size should be set equal to the minimum step size.

The aircraft simulation model is now configured, loaded in memory and appropriate windows are opened. This may take some time. Depending on the selected visualization method, the SIMULINK window with the name of the operating shell (default **cl_sim** or **cl_simpc** from Figure 4.1) is displayed on the whole screen or reduced to the upper part of the screen. Furthermore, a clock window and scope windows are opened. The window **Clock** continuously displays the elapsed time as the simulation progresses. The scope windows, either one scope for each state variable or a window **Floating Scope** for an arbitrary block-connection, display the activity of those signals or connection. If the simulation responses are shown as animation, then the **Animation** is opened too.
Next, the closed-loop model has to be configured. The user has to specify which models are to be imbedded in the closed-loop system, see Figure 4.12. An M-file has to be selected which serves as the closed-loop specification routine, see Section 4.3 MATLAB *routine for closed-loop specification* and Figure 4.9. This file defines the variables which name the S-functions for the controller, sensor, actuator and reference generator models and any data-files which contain the variables used in these models. Further, any necessary initialization commands for closed-loop simulation are executed.

From this moment, the MATLAB routine is halted and the simulation may be started. The routine only continues after activating the command window and pressing any key. This allows the user to run several simulations, using different reference inputs and/or adjusting the operating shell.
Before starting the simulation the operating shell in the SIMULINK window of Figure 2.12 may be adjusted first. A specific autopilot may be selected from the *Control Mode Panel.* Additionally, any additional error signals and or uncertainty models may be inserted in the operating shell, see Section 4.2.1 *Operating shell for closed-loop aircraft simulation*, or any reference signal in the reference generator model may be modified, see Section 4.2.6 *Reference signal generator.* The selection of observation outputs may be modified by extracting other signals from the *Demux* block. The observation outputs may also made accessible in a different way by using blocks from the simulink/Sinks library as for example *To File* and *To Workspace.*
The simulation is started by selecting **Start** from the **Simulation** menu on top of the

SIMULINK window. If desired, the simulation parameters may be adjusted first. They are accessible in the Control Panel dialog box which is displayed by selecting **Parameters** from the **Simulation** menu.
During the simulation, the user is allowed to interactively perform the following operations:

- change the autopilot mode by double clicking a button in the *Control Mode Panel.*
- change any of the simulation parameters or the simulation algorithm in the Control Panel dialog box.
- change the view condition in the **Animation** window when the simulation responses are shown as animation, see Section 3.3.4.
- suspend and restart the simulation via the items **Pause** and **Restart** in the **Simulation** menu; during suspension, lines or blocks in the operating shell may be added or deleted.
- stop the simulation via the item **Stop** in the **Simulation** menu.

The simulation results include the vector of the integration time points t and the time trajectories of the input controls, state variables and observations outputs:

$$
\begin{array}{lll}
\text{input controls} & \left\{ \begin{array}{lll} r & = \texttt{r} & \text{(reference signals)} \\ u & = \texttt{u} & \text{(aerodynamic controls)} \\ PLA & = \texttt{ut} & \text{(engine throttle settings)} \end{array} \right. \\
\text{state variables} & \left\{ \begin{array}{lll} x & = \texttt{xtot} & \text{(all model states)} \\ x_a & = \texttt{x} & \text{(aircraft states)} \\ x_t & = \texttt{xt} & \text{(engine states)} \end{array} \right. \\
\text{observation outputs} & \left\{ \begin{array}{lll} y & = \texttt{y} & \text{(selected observations in operating shell)} \\ y_{pow} & = \texttt{ypow} & \text{(engine parameters of all engines in propulsion model)} \end{array} \right.
\end{array}
\tag{4.4}
$$

These results are overwritten each time a simulation is finished. They further only become available in the workspace when the simulation is terminated, if desired via the item **Stop** in the **Simulation** menu, but not when a simulation is suspended.

When the MATLAB routine is resumed, the dialog in the command window continues with the request for saving the simulation results in a simulation-file `.sim`, see Figure 4.12. The simulation results refer to the signals in (4.4) and the operating conditions. It should be noted however that these results should be available from the workspace. The saved variables are specified in the variable `simvar`, see also Section 2.5 *Data-files* and Table 2.2:

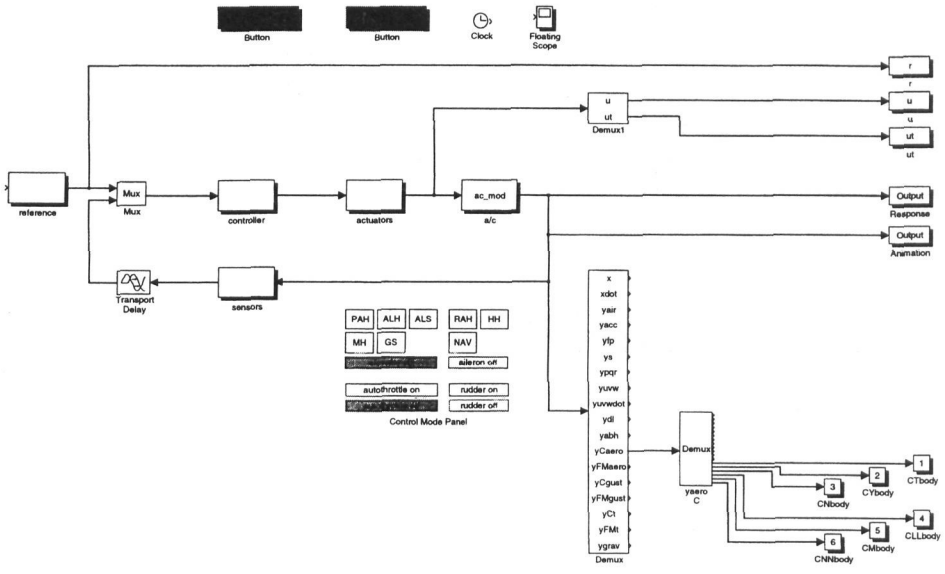
```
simvar = massinit x0 xt0 u0 ut0 Tn0 Deng t r u ut Tn x xt
         y ypow do_PLA do_simulink do_clsim                    (4.5)
         ac_model ac_simmodel ac_funmodel cl_simmodel
```

The variable list includes the mass properties, all initial conditions, the time vector, the trajectories of all state variables, any reference signals, control inputs and selected observation

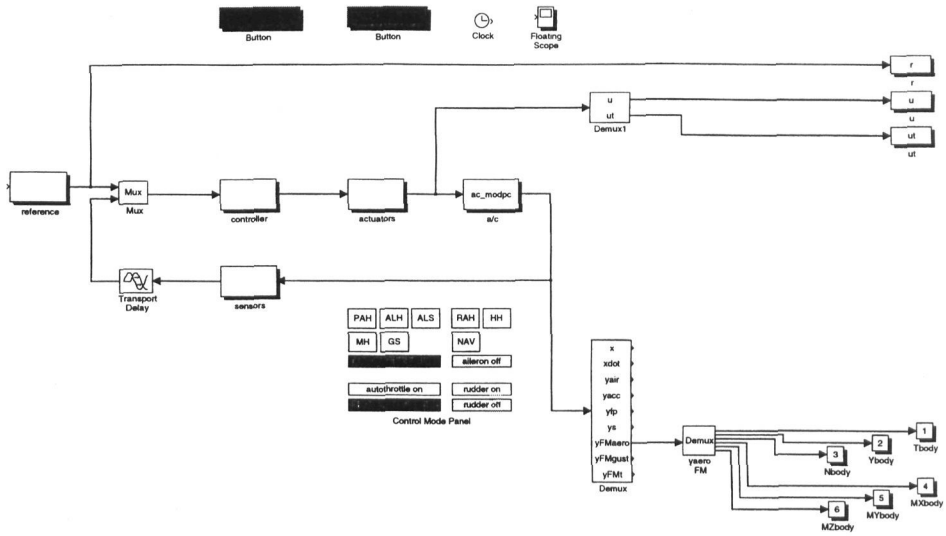
outputs, the switches for the inclusion of the engine model, the selections of simulation running procedure, the applied aircraft model and operating shell. The simulation results can be used for off-line analysis, manually from the command window or via the tools for plotting time-responses `plot_ac` or showing animation `show_ac`.

Table 4.1: Control modes and variables for closed-loop simulation.

control mode	block	variables				
		`mdpah`	`mdalh`	`mdals`	`mdmh`	`mdgs`
pitch attitude hold	PAH	1	−1	−1	−1	−1
altitude hold	ALH	−1	1	−1	−1	−1
altitude select	ALS	−1	−1	1	−1	−1
mach hold	MH	−1	−1	−1	1	−1
glide-slope	GS	−1	−1	−1	−1	1
elevator off		−1	−1	−1	−1	−1
		`mdautothr`				
autothrottle on		1				
autothrottle off		−1				
		`mdrah`	`mdhh`	`mdloc`		
roll attitude hold	RAH	1	−1	−1		
heading hold	HH	−1	1	−1		
navigation	NAV	−1	−1	1		
aileron off		−1	−1	−1		
		`mdrud`				
rudder on		1				
rudder off		−1				



`cl_sim.m` for general aircraft model `ac_mod.m`



`cl_simpc.m` for compact aircraft model `ac_modpc.m`

Figure 4.1: Default operating shells for closed-loop aircraft simulation in SIMULINK window (identical to Figure 2.14).

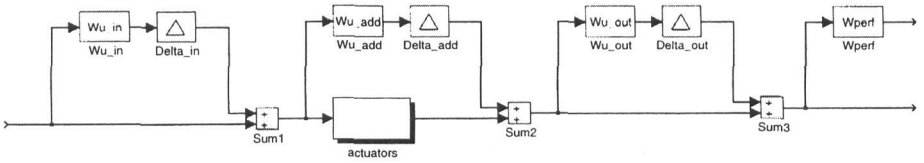


Figure 4.2: Example of implementing peturbation models and performance weights for the actuator model in the SIMULINK operating shell (Δ_{in} normalized input multiplicative perturbations, Δ_{add} normalized additive perturbations, Δ_{out} normalized output muliplicative perturbations, W_u perturbation weights, W_{perf} performance weight).

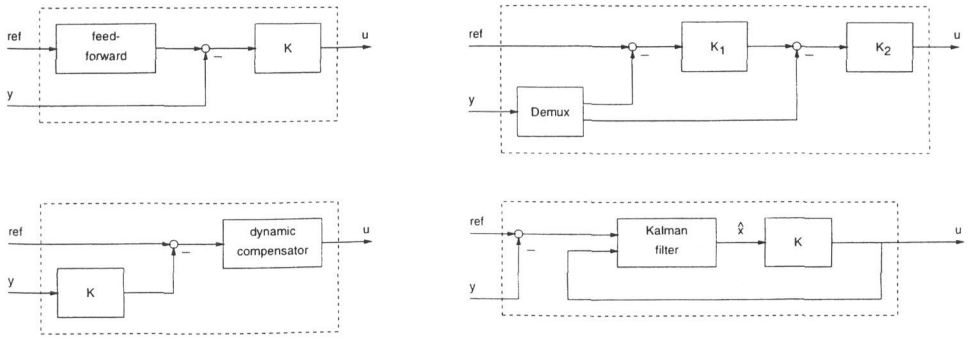


Figure 4.3: Examples of interal structures of the controller.

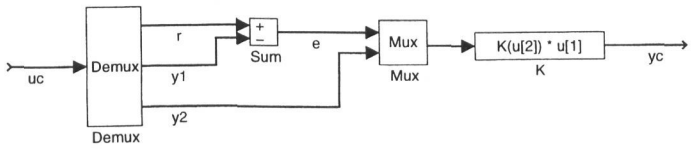


Figure 4.4: Example of implementing a scheduled control gain in SIMULINK controller model.

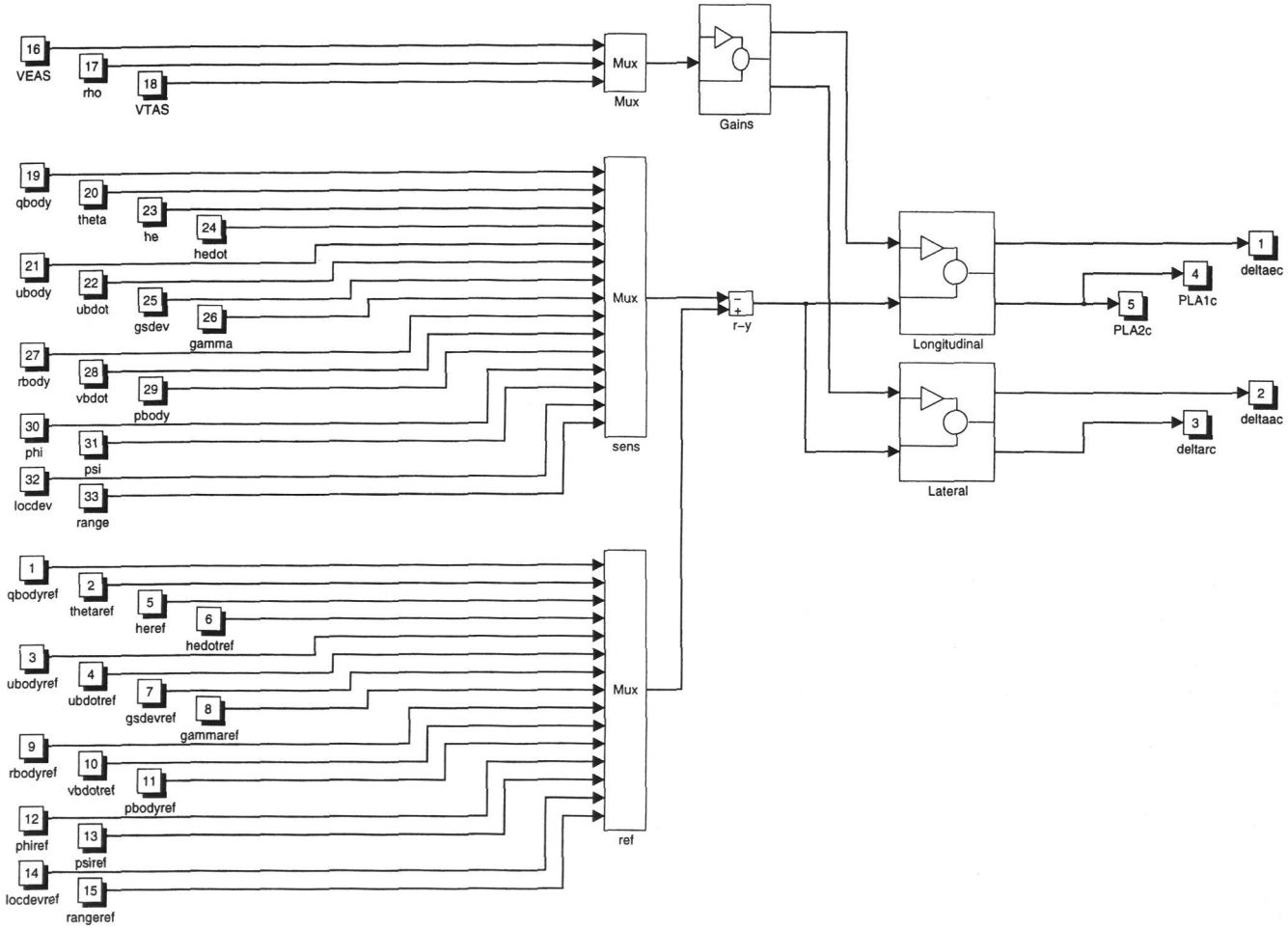


Figure 4.5: Example SIMULINK S-function for a flight controller model.

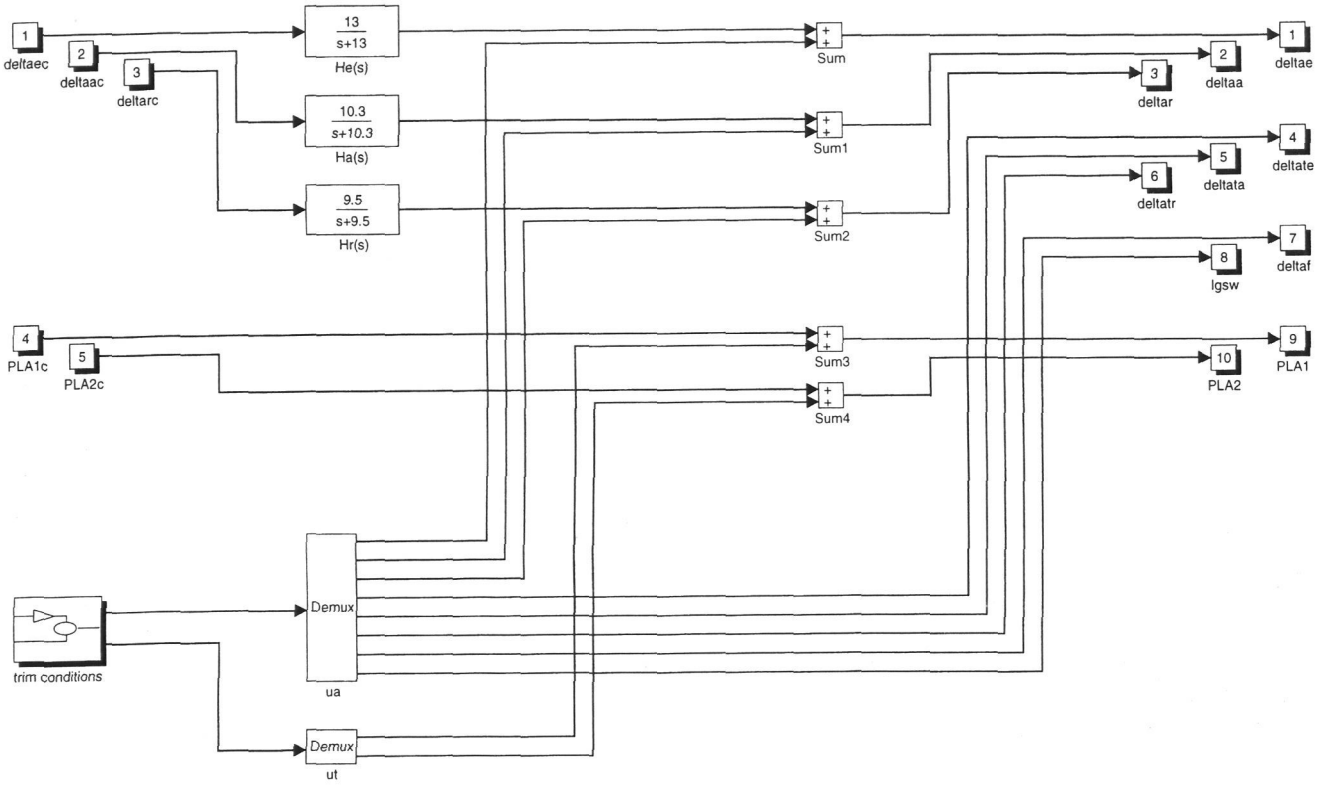


Figure 4.6: Example SIMULINK S-function for an actuator model.

```
/*
 * CIT_SENS.C
 * 21-05-96
 * C.A.A.M. van der Linden
 *
 * syntax     : [ret,x0]=cit_sens(t,x,u,flag)
 * parameters: ret : ...
 * simulink   : cl_sim.m
 */

/*
 * The following #define is used to specify the name of your S-Function.
 */

#define S_FUNCTION_NAME cit_sens

/*
 * Need to include simstruc.h for the definition of the SimStruct and
 * its associated macro definitions.
 */

#include "simstruc.h"

/*
 * Need to include libraries for the definition of mathematical functions.
 */

/*
 * mdlInitializeSizes - initialize the sizes array
 *
 * The sizes array is used by SIMULINK to determine the S-function block's
 * characteristics (number of inputs, outputs, states, etc.).
 */

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumContStates(    S, 0);      /* number of continuous states */
    ssSetNumDiscStates(    S, 0);      /* number of discrete states */
    ssSetNumInputs(        S,-1);      /* number of inputs */
    ssSetNumOutputs(       S,18);      /* number of outputs */
    ssSetDirectFeedThrough(S, 1);      /* direct feedthrough flag */
    ssSetNumSampleTimes(   S, 1);      /* number of sample times */
    ssSetNumInputArgs(     S, 0);      /* number of input arguments */
    ssSetNumRWork(         S, 0);      /* number of real work vector elements */
    ssSetNumIWork(         S, 0);      /* number of integer work vector elements */
    ssSetNumPWork(         S, 0);      /* number of pointer work vector elements */
}

/*
 * mdlInitializeSampleTimes - initialize the sample times array
 *
 * This function is used to specify the sample time(s) for your S-function.
 * If your S-function is continuous, you must specify a sample time of 0.0.
 * Sample times must be registered in ascending order.  If your S-function
 * is to acquire the sample time of the block that is driving it, you must
 * specify the sample time to be INHERITED_SAMPLE_TIME.
 */

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTimeEvent(S, 0, 0.0);
    ssSetOffsetTimeEvent(S, 0, 0.0);
}

/*
 * mdlInitializeConditions - initialize the states
 *
 * In this function, you should initialize the continuous and discrete
 * states for your S-function block.  The initial states are placed
 * in the x0 variable.  You can also perform any other initialization
 * activities that your S-function may require.
 */

static void mdlInitializeConditions(double *x0, SimStruct *S)
{
}

/*
 * mdlOutputs - compute the outputs
 *
 * In this function, you compute the outputs of your S-function
 * block.  The outputs are placed in the y variable.
 */

static void mdlOutputs(double *y, double *x, double *u, SimStruct *S, int tid
{
        y[0]  = u[39];          /* VEAS */
        y[1]  = u[25];          /* rho */
        y[2]  = u[3];           /* VTAS */
        y[3]  = u[1];           /* qbody */
        y[4]  = u[7];           /* theta */
        y[5]  = u[3];           /* ubody (= VTAS) */
        y[6]  = u[105];         /* ubdot */
        y[7]  = u[9];           /* he */
        y[8]  = u[21];          /* hedot */
        y[9]  = u[82];          /* gsdev */
        y[10] = u[71];          /* gamma */
        y[11] = u[2];           /* rbody */
        y[12] = u[106];         /* vbdot */
        y[13] = u[0];           /* pbody */
        y[14] = u[6];           /* phi */
        y[15] = u[8];           /* psi */
        y[16] = u[83];          /* locdev */
        y[17] = u[84];          /* range */
}

/*
 * mdlUpdate - perform action at major integration time step
 *
 * This function is called once for every major integration time step.
 * Discrete states are typically updated here, but this function is useful
 * for performing any tasks that should only take place once per integration
 * step.
 */

static void mdlUpdate(double *x, double *u, SimStruct *S, int tid)
{
}

/*
 * mdlDerivatives - compute the derivatives
 *
 * In this function, you compute the S-function block's derivatives.
 * The derivatives are placed in the dx variable.
 */

static void mdlDerivatives(double *dx, double *x, double *u, SimStruct *S, in
{
}

/*
 * mdlTerminate - called when the simulation is terminated.
 *
 * In this function, you should perform any actions that are necessary
 * at the termination of a simulation.  For example, if memory was allocated
 * in mdlInitializeConditions, this is the place to free it.
 */

static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Figure 4.7: Example SIMULINK S-function for a sensor model.

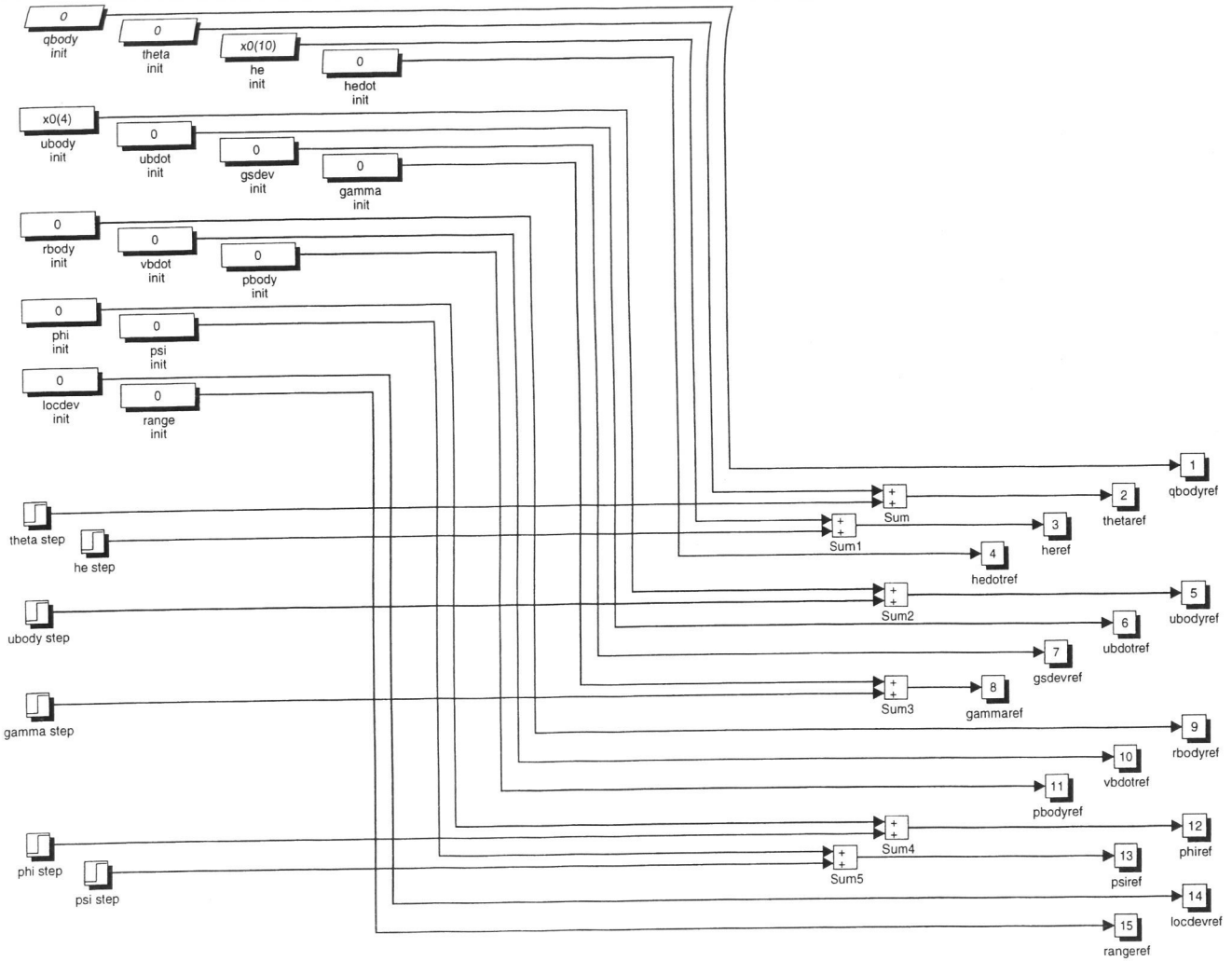


Figure 4.8: Example SIMULINK S-function for a reference signal generator.

```
% 'filename'
% 'date'
% 'author'
% program for defining names of models and data-files and executing
% initialisation commands for ...
%
% main prog : cl_set.m


% closed-loop specific data
cl_data         ='';
cl_refmodel     ='';
cl_controlmodel ='';
cl_actuatormodel='';
cl_sensormodel  ='';


% initialisation commands
yair =ac_atmos(0,[],x0,3)';
range=x0(10)/sin(3/180*pi);
```

Figure 4.9: Basic lay-out of MATLAB M-script for closed-loop specification routine designated by `cl_info`.

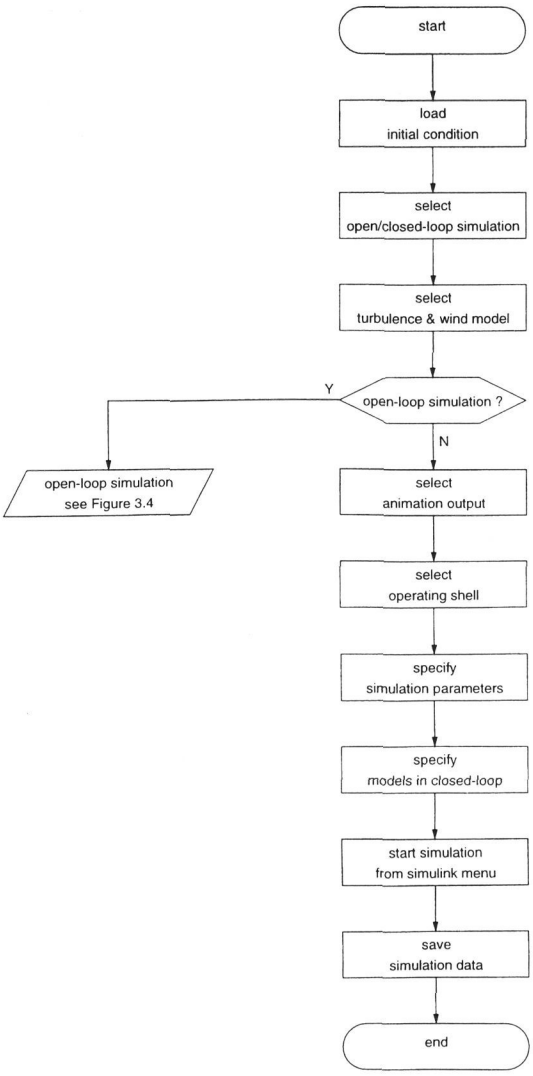


Figure 4.10: Flow-diagram of aircraft simulation tool `sim_ac.m` with closed-loop simulation.

```
    ************************************************
    **                                            **
    **          Aircraft Simulation-routine       **
    **                                            **
    ************************************************


set starting point from trimmed flight-condition


available files : *.tri

data/lr50_1.tri  data/lr50_3.tri  data/lr50_5.tri
data/lr50_2.tri  data/lr50_4.tri


select file : lr50_1

run open-loop or closed-loop simulation         [o/c] : c

include non-zero wind condition                 [y/n] : n

include non-zero turbulence condition           [y/n] : n

show animation, time-responses or model       [a/r/m] : m

do simulation with default operating shell      [y/n] : y

================================= NEW SCREEN ================================

specify simulation parameters

stop time          (in sec) :  10

min. step size     (in sec) :   0.001

max. step size     (in sec) :   0.010


loading simulation-model 'cl_sim', please wait ...
```

Figure 4.11: Display for aircraft simulation tool with closed-loop simulation.

```
specify models in close-loop system to be analyzed

available files : *.m


cit_act.m    cit_afcs.m  cit_mass.m  cit_ref.m
cit_aero.m   cit_ctrl.m  cit_pow.m   citation.m


select file : cit_afcs


start simulation from menu in simulink window 'cl_sim'


save simulated flight in file                    [y/n] : y


current files : *.sim

data/example.sim


select file :
```

Figure 4.12: Follow-up display for closed-loop aircraft simulation.

Bibliography

[1] Anon. International Standard Atmosphere (ISA). Technical Report 2533, ISO, 1975.

[2] Anon. Flight Dynamics; Concepts, Quantities and Symbols. Technical Report 1151, ISO, 1988.

[3] G.J. Balas, J.C. Doyle, K. Glover, A. Packard, and R. Smith. *μ-Analysis and Synthesis Toolbox (μ-Tools).* MUSYN Inc. and The MathWorks Inc., 1994.

[4] J.E. Dennis, Jr. and D.J. Woods. Optimization on Microcomputers: The Nelder-Mead Simplex Algorithm. In A. Wouk, editor, *New Computing Environments: Microcomputers in Large-Scale Computing*, pages 116–122. SIAM, 1987.

[5] N.R. Draper and H. Smith. *Applied Regression Analysis.* John Wiley & Sons Inc., New York, 1981.

[6] E.L. Duke, B.P. Patterson, and R.F. Antoniewicz. User's Manual for LINEAR, a FORTRAN Program to Derive Linear Aircraft Models. Technical Report TP-2768, NASA, Ames Research Center, Dryden Flight Research Facility, Edwards, California, 1987.

[7] V. Klein. Estimation of Aircraft Aerodynamic Parameters from Flight Data. *Progress in Aerospace Sciences*, 26:1–77, 1989.

[8] J.M. Maciejowski. *Multivariable Feedback Design.* Addison-Wesley Publishers Company Ltd., Wokingham, UK, 1989.

[9] The MathWorks Inc. *MATLAB, High-Performance Numeric Computation and Visualization Software*, version 4.2c edition, November 1994.

[10] The MathWorks Inc. *SIMULINK, Dynamic System Simulation Software*, version 1.3c edition, August 1994.

[11] D. McLean. *Automatic Flight Control Systems.* Prentice Hall International Ltd., Hemel Hempstead, UK, 1990.

[12] J.A. Mulder and J.C. van der Vaart. Aircraft Responses to Atmospheric Turbulence. Technical Report D-47, Delft University of Technology, Faculty of Aerospace Engineering, Delft, The Netherlands, 1993.

[13] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Receipes in C, The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1992.

[14] M.O. Rauw. A SIMULINK Environment for Flight Dynamics and Control Analysis - Application to the DHC-2 'Beaver'. Master's thesis, Delft University of Technology, Faculty of Aerospace Engineering, Delft, The Netherlands, September 1993.

[15] B.L. Stevens and F.L. Lewis. *Aircraft Control and Simulation.* John Wiley & Sons Inc., New York, 1992.

Appendix A

Files used by *DASMAT*

This appendix lists all files supplied by the *DASMAT* package and. The files can be distinguished in the following categories:

- *DASMAT* -supplied files
  - General MATLAB functions (M-functions)
  - Generic SIMULINK models (S-functions)
  - Visualization, Animation and Supplementary functions (M-functions)
  - Data-files (MAT-files)
- User-generated data-files
- User-defined aircraft specific files

The files in the first item are maintained in the *DASMAT* directory. The user-generated data-files are saved in a user-specified data-directory. The user-defined aircraft specific files should be in the operational work-directory.

```
DASMAT Package files

General matlab functions (m-functions)
  dasmat        - Main program for DASMAT package
  sim_ac        - Program for startup a/c simulation
  sim_eng       - Program for startup engine simulation
  trim_ac       - Program for trimming a/c for different flight conditions
  lin_ac        - Program for linearizing a/c around trimmed flight condition
  fit_aero      - Program for fitting aerodynamic model in polynomials
  show_ac       - Program for off-line showing animation of a/c simulation
  plot_ac       - Program for off-line plotting time-responses of a/c simulation

Generic simulink models (s-functions)
  ac_mod        - General a/c model
  ac_modpc      - Compact a/c model
  ac_sim        - Operating shell for open-loop simulation 'ac_mod' in simulink-window
  ac_simpc      - Operating shell for open-loop simulation 'ac_modpc' in simulink-window
  ac_fun        - Operating shell for open-loop simulation 'ac_mod' in command-window
  ac_funpc      - Operating shell for open-loop simulation 'ac_modpc' in command-window
  ac_anim       - Flight animation function
  ac_resp       - Flight responses function
  aero_mod      - Aerodynamic model
  cl_sim        - Operating shell for closed-loop simulation 'ac_mod' in simulink-window
  cl_simpc      - Operating shell for closed-loop simulation 'ac_modpc' in simulink-window
  eng_mod       - Engine model
  eng_none      - Engine model with thrust input (direct feed through)
  eng_sim       - Operating shell for engine simulation in simulink-window
  eng_fun       - Operating shell for engine simulation in command-window
  tur_none      - Turbulence model (zero gust)
  tur_dryd      - Turbulence model (dryden spectrum)
  wnd_none      - Wind model (zero wind)
  ...
  ac_aero       - Template for user-supplied aerodynamic model
  ac_pow        - Example for user-supplied propulsion model

Visualization, Animation and Supplementary functions (m-functions)
  ac_menu       - Initializing DASMAT menu-window
  ac_help       - Displaying help-windows
  ac_init       - Initializing DASMAT
  ac_ctrl0      - Initializing on-line control-window
  ac_ctrl1      - Reading and setting on-line controls
  ac_ctrl2      - Setting on-line controls to trim values
  ac_anim0      - Initializing animation-window
  ac_anim1      - Setting a/c responses
  ac_anim2      - Setting view and scale for animation
  ac_anim3      - Setting size animation-window
  ac_draw       - Plotting a/c figure
  ...
  sim_ac1       - Simulating a/c in open-loop via simulink-window
  sim_ac2       - Simulating a/c in open-loop via command-window
  sim_ac3       - Simulating a/c in closed-loop via simulink-window
  sim_eng1      - Simulating engine via simulink-window
  sim_eng2      - Simulating engine via command-window
  ac_set        - Setting sub-models in generic a/c model
  cl_mode       - Setting controller modes for closed-loop simulation
  cl_set        - Setting models for closed-loop simulation
  eng_set       - Setting engine in generic engine model
  inp_ac        - Defining time-traces of input signals
  ...
```

Figure A.1: List of files for the *DASMAT* package.

```
  ...
  trimcost      - Calculating trim cost function
  trim_eng      - Trimming engine for specified thrust
  lin_ac1       - Linearizing a/c model
  plot_ac1      - Plotting a/c time-responses
  fit_aero1     - Least squares fitting aerodynamic/gust polynomials
  ...
  ac_axes       - Routine for axes-transformation (m/c/mex-file)
  ac_atmos      - Routine for atmospheric model (c/mex-file)
  ac_sig        - Routine for defining input signals
  ac_slct       - Routine for selecting options from displayed table
  iofile        - Routine for file input/output operations
  isdir         - Routine for checking directory existence
  fig_chk       - Routine for checking screen-windows
  sftable3      - Routine for three dimensional table lookup

Data-files (mat-files)
  ac_windw.mat - Window-interface data
  ac_genrl.mat - General operating data
  ac_geom.mat  - Model-geometry data
  ac_turb.mat  - Turbulence psd data
  cl_ctrl.mat  - General closed-loop data
```

Figure A.1: Continue.

```
User-generated data-files (mat-files in directory 'datadir')

  *       .inp - Data of input signals
  *       .tri - Data of trimmed flight condition
  *       .lin - Data of linearized model
  *       .sim - Data of simulated flight
  *       .aer - Data of aerodynamic model parameters
```

Figure A.2: List of user-generated data-files from *DASMAT* tools.

```
Aircraft specific simulink models/matlab functions (s/m-functions)

  cit_aero      - Aerodynamic model
  cit_pow       - Propulsion model
  jt15dn1       - Engine model with throttle input (dynamic version)
  jt15st1       - Engine model with throttle input (static version)
  ...
  citation      - Program for defining a/c models and data-files
  cit_mass      - Function for calculating mass-properties
  ...
  cit_ctrl      - Controller model
  cit_act       - Actuator model
  cit_ref       - Reference signal generator
  cit_sens      - Sensor model (c/mex-file)
  ...
  cit_afcs      - Program for defining closed-loop models

Aircraft specific data-files (mat-files)

  citdata .mat - Specific a/c data
  jt15data.mat - Specific engine data
```

Figure A.3: List of aircraft specific files for Citation 500 aircraft.

Appendix B

Signal formats of generic aircraft models

This appendix lists the formats of the main signals of the generic aircraft models `ac_mod.m` and `ac_modpc.m`. These signals are related to *Integrator* blocks and to the system level *Inport* and *Outport* blocks. Tables for the following signals are presented:

- Aircraft state variables `x`
- Aircraft control inputs `u` and `ut/Tn`
- Aircraft observation outputs `y` of S-function `ac_mod.m`
- Aircraft observation outputs `y` of S-function `ac_modpc.m`
- Engine observation outputs `ypow` of S-function `eng_none.m`

For all above signals, the tables present the number and name of the *Inport/Outport* block, the symbol and dimension of the signal and a short description.

Table B.1: Format of aircraft state variables in aircraft generic models `ac_mod.m` and `ac_modpc.m`.

Aircraft state variables: $\mathtt{x} = x_a$

abs	rel	name	symbol	dim	description
1	1	`pbody`	p_b	[rad/s]	roll rate about body X-axis
2	2	`qbody`	q_b	[rad/s]	pitch rate about body Y-axis
3	3	`rbody`	r_b	[rad/s]	yaw rate about body Z-axis
4	4	`VTAS`	V_{TAS}	[m/s]	true airspeed
5	5	`alpha`	α	[rad]	angle of attack
6	6	`beta`	β	[rad]	angle of sideslip
7	7	`phi`	ϕ	[rad]	roll angle
8	8	`theta`	θ	[rad]	pitch angle
9	9	`psi`	ψ	[rad]	yaw angle
10	10	`he`	h_e	[m]	geometric altitude
11	11	`xe`	x_e	[m]	horizontal position along earth X-axis
12	12	`ye`	y_e	[m]	horizontal position along earth Y-axis

Table B.2: Format of *Inport* blocks in aircraft generic models `ac_mod.m` and `ac_modpc.m`.

Aerodynamic control inputs: $\mathtt{u} = u_a$

abs	rel	name	symbol	dim	description
1	1	`deltae`	δ_e	[rad]	elevator deflection
2	2	`deltaa`	δ_a	[rad]	aileron deflection
3	3	`deltar`	δ_r	[rad]	rudder deflection
4	4	`deltate`	δ_{t_e}	[rad]	elevator trimtab deflection
5	5	`deltata`	δ_{t_a}	[rad]	aileron trimtab deflection
6	6	`deltatr`	δ_{t_r}	[rad]	rudder trimtab deflection
7	7	`deltaf`	δ_f	[rad]	flap deflection
8	8	`lgsw`	ℓg_{sw}	[1/0]	switch which specifies undercarriage retraction / extension

Thrust control inputs: $\left.\begin{matrix}\mathtt{ut}\\ \mathtt{Tn}\end{matrix}\right\} = [u_{t_1} \quad u_{t_2}]$

abs	rel	name	symbol	dim	description
1	1	`ut(1)`	PLA_1	[rad]	power lever angle engine 1
2	2	`ut(2)`	PLA_2	[rad]	power lever angle engine 2
1	1	`Tn(1)`	T_{N_1}	[N]	thrust engine 1
2	2	`Tn(2)`	T_{N_2}	[N]	thrust engine 2

Table B.3: Format of *Outport* blocks in aircraft generic model `ac_mod.m`.

Aircraft state variables: `y1` = `x` = x_a

abs	rel	name	symbol	dim	description
1	1	`pbody`	p_b	[rad/s]	roll rate about body X-axis
2	2	`qbody`	q_b	[rad/s]	pitch rate about body Y-axis
3	3	`rbody`	r_b	[rad/s]	yaw rate about body Z-axis
4	4	`VTAS`	V_{TAS}	[m/s]	true airspeed
5	5	`alpha`	α	[rad]	angle of attack
6	6	`beta`	β	[rad]	angle of sideslip
7	7	`phi`	ϕ	[rad]	roll angle
8	8	`theta`	θ	[rad]	pitch angle
9	9	`psi`	ψ	[rad]	yaw angle
10	10	`he`	h_e	[m]	geometric altitude
11	11	`xe`	x_e	[m]	horizontal position along earth X-axis
12	12	`ye`	y_e	[m]	horizontal position along earth Y-axis

Aircraft state derivatives: `y2` = `xdot` = $\dot{x}_a$

abs	rel	name	symbol	dim	description
13	1	`pbdot`	$\dot{p}_b$	[rad/s^2]	roll acceleration about body X-axis
14	2	`qbdot`	$\dot{q}_b$	[rad/s^2]	pitch acceleration about body Y-axis
15	3	`rbdot`	$\dot{r}_b$	[rad/s^2]	yaw acceleration about body Z-axis
16	4	`VTASdot`	$\dot{V}_{TAS}$	[m/s^2]	time derivative of true airspeed
17	5	`alphadot`	$\dot{\alpha}$	[rad/s]	angle of attack rate
18	6	`betadot`	$\dot{\beta}$	[rad/s]	angle of sideslip rate
19	7	`phidot`	$\dot{\phi}$	[rad/s]	roll attitude rate
20	8	`thetadot`	$\dot{\theta}$	[rad/s]	pitch attitude rate
21	9	`psidot`	$\dot{\psi}$	[rad/s]	heading rate
22	10	`hedot`	$\dot{h}_e$	[m/s]	geometric altitude rate
23	11	`xedot`	$\dot{x}_e$	[m/s]	horizontal ground speed along earth X-axis
24	12	`yedot`	$\dot{y}_e$	[m/s]	horizontal ground speed along earth Y-axis

Table B.3: Continue.

Airdata parameters: y3 = yair = y_{air}

abs	rel	name	symbol	dim	description
25	1	pstat	p_a	[N/m^2]	ambient pressure
26	2	rho	ρ	[kg/m^3]	air density
27	3	temp	T	[K]	ambient temperature
28	4	grav	g	[m/s^2]	acceleration of gravity
29	5	hpress	h_p	[m]	pressure altitude
30	6	hradio	h_R	[m]	radio altitude
31	7	Hgeopot	H	[m]	geopotential altitude
32	8	Vsound	V_{sound}	[m/s]	speed of sound
33	9	Mach	M	[-]	Mach number
34	10	qdyn	$\bar{q}$	[N/m^2]	dynamic pressure
35	11	Reynl	Re'	[-]	Reynolds number per unit length
36	12	qc	q_c	[N/m^2]	impact pressure
37	13	qrel	q_{rel}	[-]	relative impact pressure
38	14	ptot	p_t	[N/m^2]	total pressure
39	15	temptot	T_t	[K]	total temperature
40	16	VEAS	V_{EAS}	[m/s]	equivalent airspeed
41	17	VCAS	V_{CAS}	[m/s]	calibrated airspeed
42	18	VIAS	V_{IAS}	[m/s]	indicated airspeed
43	19	uwindb	u_{w_b}	[m/s]	wind velocity along body X-axis
44	20	vwindb	v_{w_b}	[m/s]	wind velocity along body Y-axis
45	21	wwindb	w_{w_b}	[m/s]	wind velocity along body Z-axis
46	22	uwinde	u_{w_e}	[m/s]	wind velocity along earth X-axis
47	23	vwinde	v_{w_e}	[m/s]	wind velocity along earth Y-axis
48	24	wwinde	w_{w_e}	[m/s]	wind velocity along earth Z-axis
49	25	ug	$\hat{u}_g$	[-]	dimensionless gust velocity along negative body X-axis
50	26	alphag	α_g	[rad]	gust angle of attack
51	27	betag	β_g	[rad]	gust angle of sideslip
52	28	ugdotcV	$\dot{\hat{u}}_g\bar{c}/V$	[-]	dimensionless gust velocity derivative along negative body X-axis
53	29	agdotcV	$\dot{\alpha}_g\bar{c}/V$	[rad]	dimensionless gust angle of attack rate
54	30	bgdotbV	$\dot{\beta}_g b/V$	[rad]	dimensionless gust angle of sideslip rate
55	31	ugasym	$\hat{u}_{gasym}$	[-]	dimensionless gust velocity along negative body X-axis, varying along wingspan
56	32	alphagasym	α_{gasym}	[rad]	gust angle of attack, varying along wingspan

Table B.3: Continue.

Acceleration parameters: y4 = yacc = y_{acc}

abs	rel	name	symbol	dim	description
57	1	axb	a_{x_b}	[g]	acceleration at c.g. along body X-axis
58	2	ayb	a_{y_b}	[g]	acceleration at c.g. along body Y-axis
59	3	azb	a_{z_b}	[g]	acceleration at c.g. along body Z-axis
60	4	anxb	a_{nx_b}	[g]	accelerometer output at c.g. along body X-axis
61	5	anyb	a_{ny_b}	[g]	accelerometer output at c.g. along body Y-axis
62	6	anzb	a_{nz_b}	[g]	accelerometer output at c.g. along body Z-axis
63	7	anxa	a_{nx_a}	[g]	accelerometer output at c.g. along airpath X-axis
64	8	anya	a_{ny_a}	[g]	accelerometer output at c.g. along airpath Y-axis
65	9	anza	a_{nz_a}	[g]	accelerometer output at c.g. along airpath Z-axis
66	10	anxib	a_{nx,i_b}	[g]	accelerometer output at $(x,y,z)_{i_{acc}}$ along body X-axis
67	11	anyib	a_{ny,i_b}	[g]	accelerometer output at $(x,y,z)_{i_{acc}}$ along body Y-axis
68	12	anzib	a_{nz,i_b}	[g]	accelerometer output at $(x,y,z)_{i_{acc}}$ along body Z-axis
69	13	anb	a_{n_b}	[g]	normal acceleration at c.g.
70	14	anib	a_{n,i_b}	[g]	normal acceleration at $(x,y,z)_{i_{acc}}$
71	15	n	n	[g]	load factor

Flight-path-related parameters: y5 = yfp = y_{fp}

abs	rel	name	symbol	dim	description
72	1	gamma	γ	[rad]	flight-path angle
73	2	chi	χ	[rad]	azimuth angle
74	3	gammadot	$\dot{\gamma}$	[rad/s]	flight-path angle rate
75	4	chidot	$\dot{\chi}$	[rad/s]	azimuth angle rate
76	5	gammatrue	γ_T	[rad]	true flight-path angle
77	6	chitrue	χ_T	[rad]	true azimuth angle
78	7	gammatdot	$\dot{\gamma}_T$	[rad/s]	true flight-path angle rate
79	8	chitdot	$\dot{\chi}_T$	[rad/s]	true azimuth angle rate
80	9	chimagn	χ_M	[rad]	magnetic azimuth angle
81	10	psimagn	ψ_M	[rad]	magnetic heading
82	11	bank	Φ	[rad]	bank angle
83	12	gsdev	Γ	[rad]	glide-slope deviation
84	13	locdev	λ	[rad]	localizer deviation
85	14	range	R	[m]	range to ground beacon
86	15	Vground	V_G	[m/s]	ground speed
87	16	heacc	$\ddot{h}_e$	[m/s^2]	vertical acceleration
88	17	fpacc	fpa	[m/s^2]	flight-path acceleration

Energy-related terms: y6 = ys = y_s

abs	rel	name	symbol	dim	description
89	1	Espec	E_s	[m]	specific energy
90	2	Pspec	P_s	[m/s]	specific power

Table B.3: Continue.

Angular axis velocities: y7 = ypqr = y_{pqr}

abs	rel	name	symbol	dim	description
91	1	pair	p_a	[rad/s]	roll rate about airpath X-axis
92	2	qair	q_a	[rad/s]	pitch rate about airpath Y-axis
93	3	rair	r_a	[rad/s]	yaw rate about airpath Z-axis
94	4	pstab	p_s	[rad/s]	roll rate about stability X-axis
95	5	qstab	q_s	[rad/s]	pitch rate about stability Y-axis
96	6	rstab	r_s	[rad/s]	yaw rate about stability Z-axis

Linear axis velocities: y8 = yuvw = y_{uvw}

abs	rel	name	symbol	dim	description
97	1	ustab	u_s	[m/s]	velocity along stability X-axis
98	2	vstab	v_s	[m/s]	velocity along stability Y-axis
99	3	wstab	w_s	[m/s]	velocity along stability Z-axis
100	4	ubody	u_b	[m/s]	velocity along body X-axis
101	5	vbody	v_b	[m/s]	velocity along body Y-axis
102	6	wbody	w_b	[m/s]	velocity along body Z-axis

Linear axis velocity derivatives: y9 = yuvwdot = $y_{\dot{u}\dot{v}\dot{w}}$

abs	rel	name	symbol	dim	description
103	1	usdot	$\dot{u}_s$	[m/s^2]	velocity derivative along stability X-axis
104	2	vsdot	$\dot{v}_s$	[m/s^2]	velocity derivative along stability Y-axis
105	3	wsdot	$\dot{w}_s$	[m/s^2]	velocity derivative along stability Z-axis
106	4	ubdot	$\dot{u}_b$	[m/s^2]	velocity derivative along body X-axis
107	5	vbdot	$\dot{v}_b$	[m/s^2]	velocity derivative along body Y-axis
108	6	wbdot	$\dot{w}_b$	[m/s^2]	velocity derivative along body Z-axis

Dimensionless axis velocities: y10 = ydl = y_{dl}

abs	rel	name	symbol	dim	description
109	1	pab2V	$p_a b/2V$	[-]	dimensionless roll rate about airpath X-axis
110	2	qacV	$q_a \bar{c}/V$	[-]	dimensionless pitch rate about airpath Y-axis
111	3	rab2V	$r_a b/2V$	[-]	dimensionless yaw rate about airpath Z-axis
112	4	psb2V	$p_s b/2V$	[-]	dimensionless roll rate about stability X-axis
113	5	qscV	$q_s \bar{c}/V$	[-]	dimensionless pitch rate about stability Y-axis
114	6	rsb2V	$r_s b/2V$	[-]	dimensionless yaw rate about stability Z-axis
115	7	pbb2V	$p_b b/2V$	[-]	dimensionless roll rate about body X-axis
116	8	qbcV	$q_b \bar{c}/V$	[-]	dimensionless pitch rate about body Y-axis
117	9	rbb2V	$r_b b/2V$	[-]	dimensionless yaw rate about body Z-axis
118	10	ubV	u_b/V	[-]	dimensionless velocity along body X-axis

Sensor measurements: y11 = yabh = $y_{\alpha\beta h}$

abs	rel	name	symbol	dim	description
119	1	alphai	α_i	[rad]	angle of attack at $(x, y, z)_{i_{\alpha\beta h}}$
120	2	betai	β_i	[rad]	angle of sideslip at $(x, y, z)_{i_{\alpha\beta h}}$
121	3	hei	$h_{e,i}$	[m]	geometric altitude at $(x, y, z)_{i_{\alpha\beta h}}$
122	4	heidot	$\dot{h}_{e,i}$	[m/s]	geometric altitude rate at $(x, y, z)_{i_{\alpha\beta h}}$

Table B.3: Continue.

Aerodynamic force and moment coefficients: y12 = yCaero = $y_{C_{aero}}$

abs	rel	name	symbol	dim	description
123	1	CDair	C_{D_a}	[-]	aerodynamic drag coefficient in airpath reference frame
124	2	CYair	C_{Y_a}	[-]	aerodynamic sideforce coefficient in airpath reference frame
125	3	CLair	C_{L_a}	[-]	aerodynamic lift coefficient in airpath reference frame
126	4	CLLair	C_{ℓ_a}	[-]	aerodynamic rolling moment coefficient in airpath reference frame
127	5	CMair	C_{m_a}	[-]	aerodynamic pitching moment coefficient in airpath reference frame
128	6	CNNair	C_{n_a}	[-]	aerodynamic yawing moment coefficient in airpath reference frame
129	7	CDstab	C_{D_s}	[-]	aerodynamic drag coefficient in stability reference frame
130	8	CYstab	C_{Y_s}	[-]	aerodynamic sideforce coefficient in stability reference frame
131	9	CLstab	C_{L_s}	[-]	aerodynamic lift coefficient in stability reference frame
132	10	CLLstab	C_{ℓ_s}	[-]	aerodynamic rolling moment coefficient in stability reference frame
133	11	CMstab	C_{m_s}	[-]	aerodynamic pitching moment coefficient in stability reference frame
134	12	CNNstab	C_{n_s}	[-]	aerodynamic yawing moment coefficient in stability reference frame
135	13	CTbody	C_{T_b}	[-]	aerodynamic tangential force coefficient in body reference frame
136	14	CYbody	C_{Y_b}	[-]	aerodynamic sideforce coefficient in body reference frame
137	15	CNbody	C_{N_b}	[-]	aerodynamic normal force coefficient in body reference frame
138	16	CLLbody	C_{ℓ_b}	[-]	aerodynamic rolling moment coefficient in body reference frame
139	17	CMbody	C_{m_b}	[-]	aerodynamic pitching moment coefficient in body reference frame
140	18	CNNbody	C_{n_b}	[-]	aerodynamic yawing moment coefficient in body reference frame

Aerodynamic forces and moments: y13 = yFMaero = $y_{FM_{aero}}$

abs	rel	name	symbol	dim	description
141	1	Dair	D_a	[N]	aerodynamic drag in airpath reference frame
142	2	Yair	Y_a	[N]	aerodynamic sideforce in airpath reference frame
143	3	Lair	L_a	[N]	aerodynamic lift in airpath reference frame
144	4	MXair	$\bar{L}_a$	[Nm]	aerodynamic rolling moment in airpath reference frame
145	5	MYair	M_a	[Nm]	aerodynamic pitching moment in airpath reference frame
146	6	MZair	$\bar{N}_a$	[Nm]	aerodynamic yawing moment in airpath reference frame
147	7	Dstab	D_s	[N]	aerodynamic drag in stability reference frame
148	8	Ystab	Y_s	[N]	aerodynamic sideforce in stability reference frame
149	9	Lstab	L_s	[N]	aerodynamic lift in stability reference frame
150	10	MXstab	$\bar{L}_s$	[Nn]	aerodynamic rolling moment in stability reference frame
151	11	MYstab	M_s	[Nm]	aerodynamic pitching moment in stability reference frame
152	12	MZstab	$\bar{N}_s$	[Nm]	aerodynamic yawing moment in stability reference frame
153	13	Tbody	T_b	[N]	aerodynamic tangential force in body reference frame
154	14	Ybody	Y_b	[N]	aerodynamic sideforce coefficient in body reference frame
155	15	Nbody	N_b	[N]	aerodynamic normal force in body reference frame
156	16	MXbody	$\bar{L}_b$	[Nm]	aerodynamic rolling moment in body reference frame
157	17	MYbody	M_b	[Nm]	aerodynamic pitching moment in body reference frame
158	18	MZbody	$\bar{N}_b$	[Nm]	aerodynamic yawing moment in body reference frame

Table B.3: Continue.

Force and moment coefficients due to turbulence: `y14` = `yCgust` = y_{C_g}

abs	rel	name	symbol	dim	description
159	1	`CDgair`	$C_{D_{g_a}}$	[-]	drag coefficient due to turbulence in airpath reference frame
160	2	`CYgair`	$C_{Y_{g_a}}$	[-]	sideforce coefficient due to turbulence in airpath reference frame
161	3	`CLgair`	$C_{L_{g_a}}$	[-]	lift coefficient due to turbulence in airpath reference frame
162	4	`CLLgair`	$C_{\ell_{g_a}}$	[-]	rolling moment coefficient due to turbulence in airpath reference frame
163	5	`CMgair`	$C_{m_{g_a}}$	[-]	pitching moment coefficient due to turbulence in airpath reference frame
164	6	`CNNgair`	$C_{n_{g_a}}$	[-]	yawing moment coefficient due to turbulence in airpath reference frame
165	7	`CDgstab`	$C_{D_{g_s}}$	[-]	drag coefficient due to turbulence in stability reference frame
166	8	`CYgstab`	$C_{Y_{g_s}}$	[-]	sideforce coefficient due to turbulence in stability reference frame
167	9	`CLgstab`	$C_{L_{g_s}}$	[-]	lift coefficient due to turbulence in stability reference frame
168	10	`CLLgstab`	$C_{\ell_{g_s}}$	[-]	rolling moment coefficient due to turbulence in stability reference frame
169	11	`CMgstab`	$C_{m_{g_s}}$	[-]	pitching moment coefficient due to turbulence in stability reference frame
170	12	`CNNgstab`	$C_{n_{g_s}}$	[-]	yawing moment coefficient due to turbulence in stability reference frame
171	13	`CTgbody`	$C_{T_{g_b}}$	[-]	tangential force coefficient due to turbulence in body reference frame
172	14	`CYgbody`	$C_{Y_{g_b}}$	[-]	sideforce coefficient due to turbulence in body reference frame
173	15	`CNgbody`	$C_{N_{g_b}}$	[-]	normal force coefficient due to turbulence in body reference frame
174	16	`CLLgbody`	$C_{\ell_{g_b}}$	[-]	rolling moment coefficient due to turbulence in body reference frame
175	17	`CMgbody`	$C_{m_{g_b}}$	[-]	pitching moment coefficient due to turbulence in body reference frame
176	18	`CNNgbody`	$C_{n_{g_b}}$	[-]	yawing moment coefficient due to turbulence in body reference frame

Table B.3: Continue.

Forces and moments due to turbulence: `y15` = `yFMgust` = y_{FM_g}

abs	rel	name	symbol	dim	description
177	1	`Dgair`	D_{g_a}	[N]	drag due to turbulence in airpath reference frame
178	2	`Ygair`	Y_{g_a}	[N]	sideforce due to turbulence in airpath reference frame
179	3	`Lgair`	L_{g_a}	[N]	lift due to turbulence in airpath reference frame
180	4	`MXgair`	$\bar{L}_{g_a}$	[Nm]	rolling moment due to turbulence in airpath reference frame
181	5	`MYgair`	M_{g_a}	[Nm]	pitching moment due to turbulence in airpath reference frame
182	6	`MZgair`	$\bar{N}_{g_a}$	[Nm]	yawing moment due to turbulence in airpath reference frame
183	7	`Dgstab`	D_{g_s}	[N]	drag due to turbulence in stability reference frame
184	8	`Ygstab`	Y_{g_s}	[N]	sideforce due to turbulence in stability reference frame
185	9	`Lgstab`	L_{g_s}	[N]	lift due to turbulence in stability reference frame
186	10	`MXgstab`	$\bar{L}_{g_s}$	[Nn]	rolling moment due to turbulence in stability reference frame
187	11	`MYgstab`	M_{g_s}	[Nm]	pitching moment due to turbulence in stability reference frame
188	12	`MZgstab`	$\bar{N}_{g_s}$	[Nm]	yawing moment due to turbulence in stability reference frame
189	13	`Tgbody`	T_{g_b}	[N]	tangential force due to turbulence in body reference frame
190	14	`Ygbody`	Y_{g_b}	[N]	sideforce coefficient due to turbulence in body reference frame
191	15	`Ngbody`	N_{g_b}	[N]	normal force due to turbulence in body reference frame
192	16	`MXgbody`	$\bar{L}_{g_b}$	[Nm]	rolling moment due to turbulence in body reference frame
193	17	`MYgbody`	M_{g_b}	[Nm]	pitching moment due to turbulence in body reference frame
194	18	`MZgbody`	$\bar{N}_{g_b}$	[Nm]	yawing moment due to turbulence in body reference frame

Table B.3: Continue.

Propulsion force and moment coefficients: y16 = yCt = y_{C_t}

abs	rel	name	symbol	dim	description
195	1	CXtair	$C_{X_{t_a}}$	[-]	propulsion drag coefficient in airpath reference frame
196	2	CYtair	$C_{Y_{t_a}}$	[-]	propulsion sideforce coefficient in airpath reference frame
197	3	CZtair	$C_{Z_{t_a}}$	[-]	propulsion lift coefficient in airpath reference frame
198	4	CLLtair	$C_{\ell_{t_a}}$	[-]	propulsion rolling moment coefficient in airpath reference frame
199	5	CMtair	$C_{m_{t_a}}$	[-]	propulsion pitching moment coefficient in airpath reference frame
200	6	CNNtair	$C_{n_{t_a}}$	[-]	propulsion yawing moment coefficient in airpath reference frame
201	7	CXtstab	$C_{X_{t_s}}$	[-]	propulsion drag coefficient in stability reference frame
202	8	CYtstab	$C_{Y_{t_s}}$	[-]	propulsion sideforce coefficient in stability reference frame
203	9	CZtstab	$C_{Z_{t_s}}$	[-]	propulsion lift coefficient in stability reference frame
204	10	CLLtstab	$C_{\ell_{t_s}}$	[-]	propulsion rolling moment coefficient in stability reference frame
205	11	CMtstab	$C_{m_{t_s}}$	[-]	propulsion pitching moment coefficient in stability reference frame
206	12	CNNtstab	$C_{n_{t_s}}$	[-]	propulsion yawing moment coefficient in stability reference frame
207	13	CXtbody	$C_{X_{t_b}}$	[-]	propulsion tangential force coefficient in body reference frame
208	14	CYtbody	$C_{Y_{t_b}}$	[-]	propulsion sideforce coefficient in body reference frame
209	15	CZtbody	$C_{Z_{t_b}}$	[-]	propulsion normal force coefficient in body reference frame
210	16	CLLtbody	$C_{\ell_{t_b}}$	[-]	propulsion rolling moment coefficient in body reference frame
211	17	CMtbody	$C_{m_{t_b}}$	[-]	propulsion pitching moment coefficient in body reference frame
212	18	CNNtbody	$C_{n_{t_b}}$	[-]	propulsion yawing moment coefficient in body reference frame

Propulsion forces and moments: y17 = yFMt = y_{FM_t}

abs	rel	name	symbol	dim	description
213	1	Xtair	X_{t_a}	[N]	propulsion drag in airpath reference frame
214	2	Ytair	Y_{t_a}	[N]	propulsion sideforce in airpath reference frame
215	3	Ztair	Z_{t_a}	[N]	propulsion lift in airpath reference frame
216	4	MXtair	$\bar{L}_{t_a}$	[Nm]	propulsion rolling moment in airpath reference frame
217	5	MYtair	M_{t_a}	[Nm]	propulsion pitching moment in airpath reference frame
218	6	MZtair	$\bar{N}_{t_a}$	[Nm]	propulsion yawing moment in airpath reference frame
219	7	Xtstab	X_{t_s}	[N]	propulsion drag in stability reference frame
220	8	Ytstab	Y_{t_s}	[N]	propulsion sideforce in stability reference frame
221	9	Ztstab	Z_{t_s}	[N]	propulsion lift in stability reference frame
222	10	MXtstab	$\bar{L}_{t_s}$	[Nn]	propulsion rolling moment in stability reference frame
223	11	MYtstab	M_{t_s}	[Nm]	propulsion pitching moment in stability reference frame
224	12	MZtstab	$\bar{N}_{t_s}$	[Nm]	propulsion yawing moment in stability reference frame
225	13	Xtbody	X_{t_b}	[N]	propulsion tangential force in body reference frame
226	14	Ytbody	Y_{t_b}	[N]	propulsion sideforce coefficient in body reference frame
227	15	Ztbody	Z_{t_b}	[N]	propulsion normal force in body reference frame
228	16	MXtbody	$\bar{L}_{t_b}$	[Nm]	propulsion rolling moment in body reference frame
229	17	MYtbody	M_{t_b}	[Nm]	propulsion pitching moment in body reference frame
230	18	MZtbody	$\bar{N}_{t_b}$	[Nm]	propulsion yawing moment in body reference frame

Table B.3: Continue.

Gravity forces: `y18` = `ygrav` = y_{gr}

abs	rel	name	symbol	dim	description
231	1	`Xgrair`	X_{gr_a}	[N]	component of aircraft weight along airpath X-axis
232	2	`Ygrair`	Y_{gr_a}	[N]	component of aircraft weight along airpath Y-axis
233	3	`Zgrair`	Z_{gr_a}	[N]	component of aircraft weight along airpath Z-axis
234	4	`Xgrstab`	X_{gr_s}	[N]	component of aircraft weight along stability X-axis
235	5	`Ygrstab`	Y_{gr_s}	[N]	component of aircraft weight along stability Y-axis
236	6	`Zgrstab`	Z_{gr_s}	[N]	component of aircraft weight along stability Z-axis
237	7	`Xgrbody`	X_{gr_b}	[N]	component of aircraft weight along body X-axis
238	8	`Ygrbody`	Y_{gr_b}	[N]	component of aircraft weight along body Y-axis
239	9	`Zgrbody`	Z_{gr_b}	[N]	component of aircraft weight along body Z-axis
240	10	`Weight`	W	[N]	aircraft weight

Table B.4: Format of *Outport* blocks in aircraft generic model `ac_modpc.m`.

Aircraft states: `y1` = `x` = x_a

abs	rel	name	symbol	dim	description
1	1	`pbody`	p_b	[rad/s]	roll rate about body X-axis
2	2	`qbody`	q_b	[rad/s]	pitch rate about body Y-axis
3	3	`rbody`	r_b	[rad/s]	yaw rate about body Z-axis
4	4	`VTAS`	V_{TAS}	[m/s]	true airspeed
5	5	`alpha`	α	[rad]	angle of attack
6	6	`beta`	β	[rad]	angle of sideslip
7	7	`phi`	ϕ	[rad]	roll angle
8	8	`theta`	θ	[rad]	pitch angle
9	9	`psi`	ψ	[rad]	yaw angle
10	10	`he`	h_e	[m]	geometric altitude
11	11	`xe`	x_e	[m]	horizontal position along earth X-axis
12	12	`ye`	y_e	[m]	horizontal position along earth Y-axis

Aircraft state derivatives: `y2` = `xdot` = $\dot{x}_a$

abs	rel	name	symbol	dim	description
13	1	`pbdot`	$\dot{p}_b$	[rad/s^2]	roll acceleration about body X-axis
14	2	`qbdot`	$\dot{q}_b$	[rad/s^2]	pitch acceleration about body Y-axis
15	3	`rbdot`	$\dot{r}_b$	[rad/s^2]	yaw acceleration about body Z-axis
16	4	`VTASdot`	$\dot{V}_{TAS}$	[m/s^2]	time derivative of true airspeed
17	5	`alphadot`	$\dot{\alpha}$	[rad/s]	angle of attack rate
18	6	`betadot`	$\dot{\beta}$	[rad/s]	angle of sideslip rate
19	7	`phidot`	$\dot{\phi}$	[rad/s]	roll attitude rate
20	8	`thetadot`	$\dot{\theta}$	[rad/s]	pitch attitude rate
21	9	`psidot`	$\dot{\psi}$	[rad/s]	heading rate
22	10	`hedot`	$\dot{h}_e$	[m/s]	geometric altitude rate
23	11	`xedot`	$\dot{x}_e$	[m/s]	horizontal ground speed along earth X-axis
24	12	`yedot`	$\dot{y}_e$	[m/s]	horizontal ground speed along earth Y-axis

Table B.4: Continue.

Airdata parameters: y3 = yair = y_{air}

abs	rel	name	symbol	dim	description
25	1	pstat	p_a	[N/m^2]	ambient pressure
26	2	rho	ρ	[kg/m^3]	air density
27	3	temp	T	[K]	ambient temperature
28	4	grav	g	[m/s^2]	acceleration of gravity
29	5	hpress	h_p	[m]	pressure altitude
30	6	hradio	h_R	[m]	radio altitude
31	7	Hgeopot	H	[m]	geopotential altitude
32	8	Vsound	V_{sound}	[m/s]	speed of sound
33	9	Mach	M	[-]	Mach number
34	10	qdyn	$\bar{q}$	[N/m^2]	dynamic pressure
35	11	Reynl	Re'	[-]	Reynolds number per unit length
36	12	qc	q_c	[N/m^2]	impact pressure
37	13	qrel	q_{rel}	[-]	relative impact pressure
38	14	ptot	p_t	[N/m^2]	total pressure
39	15	temptot	T_t	[K]	total temperature
40	16	VEAS	V_{EAS}	[m/s]	equivalent airspeed
41	17	VCAS	V_{CAS}	[m/s]	calibrated airspeed
42	18	VIAS	V_{IAS}	[m/s]	indicated airspeed
43	19	uwindb	u_{w_b}	[m/s]	wind velocity along body X-axis
44	20	vwindb	v_{w_b}	[m/s]	wind velocity along body Y-axis
45	21	wwindb	w_{w_b}	[m/s]	wind velocity along body Z-axis
46	22	uwinde	u_{w_e}	[m/s]	wind velocity along earth X-axis
47	23	vwinde	v_{w_e}	[m/s]	wind velocity along earth Y-axis
48	24	wwinde	w_{w_e}	[m/s]	wind velocity along earth Z-axis
49	25	ug	$\hat{u}_g$	[-]	dimensionless gust velocity along negative body X-axis
50	26	alphag	α_g	[rad]	gust angle of attack
51	27	betag	β_g	[rad]	gust angle of sideslip
52	28	ugdot	$\dot{\hat{u}}_g$	[1/s]	dimensionless gust velocity derivative along negative body X-axis
53	29	alphagdot	$\dot{\alpha}_g$	[rad/s]	gust angle of attack rate
54	30	betagdot	$\dot{\beta}_g$	[rad/s]	gust angle of sideslip rate
55	31	ugasym	$\hat{u}_{gasym}$	[-]	dimensionless gust velocity along negative body X-axis, varying along wingspan
56	32	alphagasym	α_{gasym}	[rad]	gust angle of attack, varying along wingspan

Table B.4: Continue.

Acceleration parameters: y4 = yacc = y_{acc}

abs	rel	name	symbol	dim	description
57	1	axb	a_{x_b}	[g]	acceleration at c.g. along body X-axis
58	2	ayb	a_{y_b}	[g]	acceleration at c.g. along body Y-axis
59	3	azb	a_{z_b}	[g]	acceleration at c.g. along body Z-axis
60	4	anxb	a_{nx_b}	[g]	accelerometer output at c.g. along body X-axis
61	5	anyb	a_{ny_b}	[g]	accelerometer output at c.g. along body Y-axis
62	6	anzb	a_{nz_b}	[g]	accelerometer output at c.g. along body Z-axis
63	7	anxa	a_{nx_a}	[g]	accelerometer output at c.g. along airpath X-axis
64	8	anya	a_{ny_a}	[g]	accelerometer output at c.g. along airpath Y-axis
65	9	anza	a_{nz_a}	[g]	accelerometer output at c.g. along airpath Z-axis
66	10	anxib	a_{nx,i_b}	[g]	accelerometer output at $(x,y,z)_{i_{acc}}$ along body X-axis
67	11	anyib	a_{ny,i_b}	[g]	accelerometer output at $(x,y,z)_{i_{acc}}$ along body Y-axis
68	12	anzib	a_{nz,i_b}	[g]	accelerometer output at $(x,y,z)_{i_{acc}}$ along body Z-axis
69	13	anb	a_{n_b}	[g]	normal acceleration at c.g.
70	14	anib	a_{n,i_b}	[g]	normal acceleration at $(x,y,z)_{i_{acc}}$
71	15	n	n	[g]	load factor

Flight-path-related parameters: y5 = yfp = y_{fp}

abs	rel	name	symbol	dim	description
72	1	gamma	γ	[rad]	flight-path angle
73	2	chi	χ	[rad]	azimuth angle
74	3	gammadot	$\dot{\gamma}$	[rad/s]	flight-path angle rate
75	4	chidot	$\dot{\chi}$	[rad/s]	azimuth angle rate
76	5	heacc	$\ddot{h}_e$	[m/s^2]	vertical acceleration
77	6	fpacc	fpa	[m/s^2]	flight-pathq acceleration

Energy-related terms: y6 = ys = y_s

abs	rel	name	symbol	dim	description
78	1	Espec	E_s	[m]	specific energy
79	2	Pspec	P_s	[m/s]	specific power

Table B.4: Continue.

Aerodynamic forces and moments: `y13` = `yFMaero` = $y_{FM_{aero}}$

abs	rel	name	symbol	dim	description
80	1	`Tbody`	T_b	[N]	aerodynamic tangential force in body reference frame
81	2	`Ybody`	Y_b	[N]	aerodynamic sideforce coefficient in body reference frame
82	3	`Nbody`	N_b	[N]	aerodynamic normal force in body reference frame
83	4	`MXbody`	$\bar{L}_b$	[Nm]	aerodynamic rolling moment in body reference frame
84	5	`MYbody`	M_b	[Nm]	aerodynamic pitching moment in body reference frame
85	6	`MZbody`	$\bar{N}_b$	[Nm]	aerodynamic yawing moment in body reference frame

Forces and moments due to turbulence: `y15` = `yFMgust` = y_{FM_g}

abs	rel	name	symbol	dim	description
86	1	`Tgbody`	T_{g_b}	[N]	tangential force due to turbulence in body reference frame
87	2	`Ygbody`	Y_{g_b}	[N]	sideforce coefficient due to turbulence in body reference frame
88	3	`Ngbody`	N_{g_b}	[N]	normal force due to turbulence in body reference frame
89	4	`MXgbody`	$\bar{L}_{g_b}$	[Nm]	rolling moment due to turbulence in body reference frame
90	5	`MYgbody`	M_{g_b}	[Nm]	pitching moment due to turbulence in body reference frame
91	6	`MZgbody`	$\bar{N}_{g_b}$	[Nm]	yawing moment due to turbulence in body reference frame

Propulsion forces and moments: `y17` = `yFMt` = y_{FM_t}

abs	rel	name	symbol	dim	description
92	1	`Ttbody`	T_{t_b}	[N]	propulsion tangential force in body reference frame
93	2	`Ytbody`	Y_{t_b}	[N]	propulsion sideforce coefficient in body reference frame
94	3	`Ntbody`	N_{t_b}	[N]	propulsion normal force in body reference frame
95	4	`MXtbody`	$\bar{L}_{t_b}$	[Nm]	propulsion rolling moment in body reference frame
96	5	`MYtbody`	M_{t_b}	[Nm]	propulsion pitching moment in body reference frame
97	6	`MZtbody`	$\bar{N}_{t_b}$	[Nm]	propulsion yawing moment in body reference frame

Table B.5: Format of *Outport* blocks in aircraft specific engine model `eng_none.m`.

Engine parameters: `ypow` = y_{pow}

abs	rel	name	symbol	dim	description
1	1	`Tn`	T_N	[N]	net thrust
2	2	`FF`	FF	[kg/s]	fuel flow
3	3	`NLP`	N_{LP}	[%]	fan speed (low pressure rpm)
4	4	`NHP`	N_{HP}	[%]	gasgenerator speed (high pressure rpm)
5	5	`NLPc`	N_{LPc}	[%]	corrected fan speed (low pressure rpm)
6	6	`NHPc`	N_{HPc}	[%]	corrected gasgenerator speed (high pressure rpm)
7	7	`WT`	WT	[kg/s]	total mass flow (core + fan air flow)
8	8	`ITT`	ITT	[K]	interstage turbine temperature
9	9	`EGT`	EGT	[K]	exhaust gas temperature
10	10	`OILT`	$OILT$	[K]	oil temperature
11	11	`OILP`	$OILP$	[N/m^2]	oil pressure

Appendix C

Variables used by *DASMAT*

This appendix lists all significant variables used by *DASMAT* and available in the MATLAB workspace. The variables may be distinguished in the following groups:

- variables which name model-files and model-blocks
- variables which name signal components
- variables which name data-files and stored variables
- variables which configure simulation models and control simulations
- variables which specify controller modes for closed-loop simulation
- variables which contain results from *DASMAT* tools
- variables which specify aircraft geometric parameters and control input limitations
- variables which specify navigation ground station and measurement locations
- variables which specify atmospheric and turbulence constants
- variables which specify aircraft geometry for animation

The tables also present the size of the variables (a ? indicates a variable size), the file where they are stored or created and a short description. If applicable the dimension is also given.

Table C.1: Variables which name model-files and model-blocks.

name	size	file	description
ac_info	[1,?]	ac_init.m	name of M-script file with names of a/c specific sub-models and data-files
ac_aeromodel	[1,?]	'ac_info'	name of S-function file with a/c specific aerodynamic model
ac_data	[1,?]	'ac_info'	name of MAT-data file with a/c specific data
ac_enginename	[1,?]	'ac_info'	name of engine
ac_massmodel	[1,?]	'ac_info'	name of M-function file with a/c specific mass model
ac_name	[1,?]	'ac_info'	name of aircraft
ac_powermodel	[1,?]	'ac_info'	name of S-function file with a/c specific propulsion model
cl_info	[1,?]	cl_set.m	name of M-script file with names of closed-loop models and data-files
cl_actuatormodel	[1,?]	'cl_info'	name of S-function file with a/c specific actuators model
cl_controlmodel	[1,?]	'cl_info'	name of S-function file with a/c specific controller model
cl_data	[1,?]	'cl_info'	name of MAT-data file with a/c specific closed-loop data
cl_refmodel	[1,?]	'cl_info'	name of S-function file with a/c specific reference signal model
cl_sensormodel	[1,?]	'cl_info'	name of S-function file with a/c specific sensors model
eng_data	[1,?]	'ac_info'	name of MAT-data file with specific engine data
eng_dynmodel	[1,?]	'ac_info'	name of S-function file with specific engine dynamic model
eng_statmodel	[1,?]	'ac_info'	name of S-function file with specific engine static model
ac_aeroblock	[1,?]	ac_init.m	name of *S-function* block for aerodynamic model in generic a/c model
ac_engblock	[1,?]	ac_init.m	name of *S-function* block for engine model in propulsion model
ac_funmodel	[1,?]	ac_init.m	name of S-function file for open-loop a/c simulation
ac_model	[1,?]	ac_init.m	name of S-function file of generic a/c model
ac_powerblock	[1,?]	ac_init.m	name of *S-function* block for propulsion model in generic a/c model
ac_simmodel	[1,?]	ac_init.m	name of S-function for open-loop a/c simulation from **Simulation** menu
ac_turbblock	[1,?]	ac_init.m	name of *S-function* block for turbulence model in generic a/c model
ac_turbmodel	[1,?]	sim_ac.m	name of S-function file of turbulence model
ac_windblock	[1,?]	ac_init.m	name of *S-function* block for wind model in generic a/c model
ac_windmodel	[1,?]	sim_ac.m	name of S-function file of wind model
aero_aeroblock	[1,?]	ac_init.m	name of *S-function* block for aerodynamic model in generic aerodynamic model
aero_funmodel	[1,?]	ac_init.m	name of S-function file for aerodynamic simulation
aero_model	[1,?]	ac_init.m	name of S-function file of generic aerodynamic model
aero_simmodel	[1,?]	ac_init.m	name of S-function file for aerodynamic simulation from **Simulation** menu
cl_actuatorblock	[1,?]	cl_set.m	name of *S-function* block for actuators model in closed-loop operating shell
cl_controlblock	[1,?]	cl_set.m	name of *S-function* block for controller model in closed-loop operating shell
cl_refblock	[1,?]	cl_set.m	name of *S-function* block for reference signals in closed-loop operating shell
cl_sensorblock	[1,?]	cl_set.m	name of *S-function* block for sensors model in closed-loop operating shell
cl_simmodel	[1,?]	ac_init.m	name of S-function for closed-loop a/c simulation from **Simulation** menu

Table C.1: Continue.

name	size	file	description
eng_engblock	[1,?]	ac_init.m	name of *S-function* block for engine model in generic engine model
eng_funmodel	[1,?]	ac_init.m	name of S-function file for engine simulation
eng_model	[1,?]	ac_init.m	name of S-function file of generic engine model
eng_simmodel	[1,?]	ac_init.m	name of S-function file for engine simulation from **Simulation** menu
clock_block	[1,?]	sim_ac?.m	name of *Clock* block in simulation model
func_block	[1,?]	sim_ac?.m	name of *S-function* block for generic model in simulation model
scope_block	[1,?]	sim_ac?.m	name of *Scope* block in simulation model
u_block	[8,?]	sim_ac1.m	names of aerodynamic control blocks in simulation model
ut_block	[?,?]	sim_ac?.m	names of thrust control blocks in simulation model

Table C.2: Variables which name signal components.

name	*size*	*file*	*description*
aerolab	[16,10]	ac_genrl.mat	names of independent variables in aerodynamic polynomials
ulab	[8,10]	ac_genrl.mat	names of aerodynamic control inputs in generic a/c model
utlab	[1,10]	ac_genrl.mat	names of engine throttle control input in generic a/c model
xlab	[12,10]	ac_genrl.mat	names of state variables in generic a/c model
ylab	[18,10]	ac_genrl.mat	names of groups of observation outputs in generic a/c model
ypowlab	[22,10]	ac_genrl.mat	names of observation outputs in a/c specific propulsion model
yred1lab	[12,10]	ac_genrl.mat	names of group 1 observation outputs in generic a/c model (aircraft state variables x_a)
yred2lab	[12,10]	ac_genrl.mat	names of group 2 observation outputs in generic a/c model (aircraft state derivatives $\dot{x}_a$)
yred3lab	[32,10]	ac_genrl.mat	names of group 3 observation outputs in generic a/c model (airdata parameters y_{air})
yred4lab	[15,10]	ac_genrl.mat	names of group 4 observation outputs in generic a/c model (acceleration parameters y_{acc})
yred5lab	[6,10]	ac_genrl.mat	names of group 5 observation outputs in generic a/c model (flight-path-related parameters y_{fp})
yred6lab	[2,10]	ac_genrl.mat	names of group 6 observation outputs in generic a/c model (energy-related parameters y_s)
yred13lab	[6,10]	ac_genrl.mat	names of group 13 observation outputs in generic a/c model (aerodynamic forces and moments $y_{FM_{aero}}$)
yred15lab	[6,10]	ac_genrl.mat	names of group 15 observation outputs in generic a/c model (forces and moments due to turbulence y_{FM_g})
yred17lab	[6,10]	ac_genrl.mat	names of group 17 observation outputs in generic a/c model (propulsion forces and moments y_{FM_t})

Table C.2: Continue.

name	size	file	description
y1lab	[12,10]	ac_genrl.mat	names of group 1 observation outputs in generic a/c model (aircraft state variables x_a)
y2lab	[12,10]	ac_genrl.mat	names of group 2 observation outputs in generic a/c model (aircraft state derivatives $\dot{x}_a$)
y3lab	[32,10]	ac_genrl.mat	names of group 3 observation outputs in generic a/c model (airdata parameters y_{air})
y4lab	[15,10]	ac_genrl.mat	names of group 4 observation outputs in generic a/c model (acceleration parameters y_{acc})
y5lab	[17,10]	ac_genrl.mat	names of group 5 observation outputs in generic a/c model (flight-path-related parameters y_{fp})
y6lab	[2,10]	ac_genrl.mat	names of group 6 observation outputs in generic a/c model (energy-related parameters y_s)
y7lab	[6,10]	ac_genrl.mat	names of group 7 observation outputs in generic a/c model (angular axis velocities y_{pqr})
y8lab	[6,10]	ac_genrl.mat	names of group 8 observation outputs in generic a/c model (linear axis velocities y_{uvw})
y9lab	[6,10]	ac_genrl.mat	names of group 9 observation outputs in generic a/c model (linear axis velocity derivatives $y_{\dot{u}\dot{v}\dot{w}}$)
y10lab	[10,10]	ac_genrl.mat	names of group 10 observation outputs in generic a/c model (dimensionless axis velocities y_{dl})
y11lab	[4,10]	ac_genrl.mat	names of group 11 observation outputs in generic a/c model (sensor measurements $y_{\alpha\beta h}$)
y12lab	[18,10]	ac_genrl.mat	names of group 12 observation outputs in generic a/c model (aerodynamic force and moment coefficients y_{Caero})
y13lab	[18,10]	ac_genrl.mat	names of group 13 observation outputs in generic a/c model (aerodynamic forces and moments y_{FMaero})
y14lab	[18,10]	ac_genrl.mat	names of group 14 observation outputs in generic a/c model (force and moment coefficients due to turbulence y_{Cg})
y15lab	[18,10]	ac_genrl.mat	names of group 15 observation outputs in generic a/c model (forces and moments due to turbulence y_{FMg})
y16lab	[18,10]	ac_genrl.mat	names of group 16 observation outputs in generic a/c model (propulsion force and moment coefficients y_{C_t})
y17lab	[18,10]	ac_genrl.mat	names of group 17 observation outputs in generic a/c model (propulsion forces and moments y_{FM_t})
y18lab	[10,10]	ac_genrl.mat	names of group 18 observation outputs in generic a/c model (gravity forces y_{gr})

Table C.3: Variables which name data-files and stored variables.

name	size	file	description
inpdata	[1,?]	sim_ac2.m	name of .inp data-file with input trajectories
inpvar	[1,21]	ac_genrl.mat	names of variables in .inp data-files
lindata	[1,?]	lin_ac.m	name of .lin data-file with linearisation results
linvar	[1,86]	ac_genrl.mat	names of variables in .lin data-files
simdata	[1,?]	sim_ac.m	name of .sim data-file with simulation results
simvar	[1,128]	ac_genrl.mat	names of variables in .sim data-files
trimdata	[1,?]	trim_ac.m	name of .tri data-file with trimming results
trimvar	[1,59]	ac_genrl.mat	names of variables in .tri data-files
fitdata	[1,?]	fit_aero.m	name of .aer data-file with aerodynamic fitting results
fitvar	[1,95]	ac_genrl.mat	names of variables in .aer data-files
datadir	[1,?]	ac_init.m	name of directory where user-generated data-files are maintained

Table C.4: Variables which configure simulation models and control simulations.

name	size	file	description
do_PLA	[1,1]	ac_genrl.mat	switch which specifies to use engine throttle settings or thrust as input for a/c generic model
do_anim	[1,1]	ac_genrl.mat	switch which specifies to show a/c simulation as animation
do_clsim	[1,1]	ac_genrl.mat	switch which specifies to run open-loop or closed-loop a/c simulation
do_engdyn	[1,1]	ac_genrl.mat	switch which specifies to run engine simulation with static or dynamic engine model
do_fitgust	[1,1]	ac_genrl.mat	switch which specifies to fit aerodynamic or gust model in polynomials
do_fitmodel	[1,1]	ac_genrl.mat	switch which specifies to fit polynomials from model-data or flight-test data
do_fitstand	[1,1]	ac_genrl.mat	switch which specifies to fit aerodynamic polynomials with standard or user-selected variables
do_genmodel	[1,1]	ac_genrl.mat	switch which specifies to use general or compact version of a/c generic model
do_movie	[1,1]	ac_genrl.mat	switch which specifies to record a/c simulation animation for replay as movie
do_online	[1,1]	ac_genrl.mat	switch which specifies to control simulation from **On-line Control** window
do_path	[1,1]	ac_genrl.mat	switch which specifies to show flight-path during a/c simulation animation
do_resp	[1,1]	ac_genrl.mat	switch which specifies to show a/c simulation as time-responses of state variables
do_simulink	[1,1]	ac_genrl.mat	switch which specifies to run a/c simulation from **Simulation** menu or in background
do_turb	[1,1]	ac_genrl.mat	switch which specifies to include non-zero turbulence model in generic a/c model
do_wind	[1,1]	ac_genrl.mat	switch which specifies to include non-zero wind model in generic a/c model
Nsteps	[1,1]	ac_init.m	max. number of integration steps for simulation
delay	[1,1]	ac_genrl.mat	time delay for preventing algebraic loop closed-loop simulation
dtmax	[1,1]	sim_ac?.m	max. integration step size during simulation
dtmin	[1,1]	sim_ac?.m	min. integration step size during simulation
tfinal	[1,1]	sim_ac?.m	stop time of simulation

Table C.5: Variables which specify controller modes for closed-loop simulation.

name	size	file	description
mdalh	[1,1]	cl_mode.m	switch which specifies altitude hold mode
mdals	[1,1]	cl_mode.m	switch which specifies altitude select mode
mdautothr	[1,1]	cl_mode.m	switch which specifies autothrottle mode
mdgs	[1,1]	cl_mode.m	switch which specifies glide-slope mode
mdhh	[1,1]	cl_mode.m	switch which specifies heading hold mode
mdloc	[1,1]	cl_mode.m	switch which specifies localizer mode
mdmh	[1,1]	cl_mode.m	switch which specifies mach hold mode
mdpah	[1,1]	cl_mode.m	switch which specifies pitch attitude hold mode
mdrah	[1,1]	cl_mode.m	switch which specifies roll attitude hold mode
mdrud	[1,1]	cl_mode.m	switch which specifies rudder control mode
mode	[1,?]	cl_sim.m	name of selected mode

Table C.6: Variables which contain results from *DASMAT* tools.

name	symbol	size	dim	file	description
Deng	D_{eng}	[1,?]		.tri	row vector with engine drag or failure switch
Tn0	T_{N_0}	[1,?]	[N]	.tri	row vector with initial engine thrust control inputs
constraint		[1,?]		.tri	row vector with initial state variables and input controls and constraints imposed by flight-condition
flightcond		[1,1]	[0-6]	.tri	specifier of trimmed flight-condition
massinit		[1,8]		.tri	row vector with aircraft mass, position of center of gravity and products of inertia
u0	u_{a_0}	[1,8]		.tri	row vector with initial aerodynamic control inputs
ut0	u_{t_0}	[1,?]	[rad]	.tri	row vector with initial engine throttle control inputs
x0	x_{a_0}	[1,12]		.tri	row vector with initial aircraft state variables
x0dot	$\dot{x}_{a_0}$	[1,12]		.tri	row vector with initial aircraft state derivatives
xt0	x_{t_0}	[1,?]		.tri	row vector with initial engine state variables
Tn	T_N	[?,?]	[N]	.sim	matrix with engine thrust control input trajectories from simulation
r	r	[?,?]		.sim	matrix with reference signal trajectories from closed-loop simulation
t	t	[?,1]	[sec]	.sim	column vector with integration time points from simulation
u	u_a	[?,8]		.sim	matrix with aerodynamic control input trajectories from simulation
ut	u_t	[?,?]	[rad]	.sim	matrix with engine throttle control input trajectories from simulation
x	x_a	[?,12]		.sim	matrix with aircraft state trajectories from simulation
xt	x_t	[?,?]		.sim	matrix with all engines state trajectories from simulation
xtot	x_{tot}	[?,?]			matrix with all state trajectories from simulation
y	y	[?,?]		.sim	matrix with aircraft observation output trajectories from simulation
ypow	y_{pow}	[?,?]		.sim	matrix with engine observation output trajectories from simulation
Alin	A	[?,?]		.lin	system matrix for linearized aircraft model
Blin	B	[?,?]		.lin	input matrix for linearized aircraft model
Clin	C	[?,?]		.lin	output matrix for linearized aircraft model
Dlin	D	[?,?]		.lin	feed-through matrix for linearized aircraft model
yndx		[1,?]		.lin	row vector with indices of observation outputs used during linearization
Upoly0		[?,16]		.aer	matrix with all independent variables in baseline polynomials
Upoly1		[?,?]		.aer	matrix with all independent variables in extended polynomials
Uaero		[?,30]		.aer	matrix with measured data in input signal format of generic aerodynamic model
Yaero		[?,6]		.aer	matrix with measured data in output signal format of generic aerodynamic model
aerovar		[6,16]		.aer	matrix which specifies selected variables in baseline polynomials
fitstat		[6,6]		.aer	matrix with fit analysis statistics
polycoef		[6,?]		.aer	matrix with fitted polynomial coefficients
Tnsig	T_N	[?,?]	[N]	.inp	matrix with specified engine thrust control input trajectories
tsig	t	[?,1]	[sec]	.inp	column vector with time points for specified input trajectories
usig	u_a	[?,8]		.inp	matrix with specified aerodynamic control input trajectories
utsig	u_t	[?,?]	[rad]	.inp	matrix with specified engine throttle control input trajectories

Table C.7: Variables which specify aircraft geometric parameters and control input limitations.

name	symbol	size	dim	file	description
S	S	[1,1]	[m^2]	'ac_data'	wing planform area
chord	$\bar{c}$	[1,1]	[m]	'ac_data'	mean aerodynamic chord
lh	ℓ_h	[1,1]	[m]	'ac_data'	tail length
massmin	m_{min}	[1,1]	[kg]	'ac_data'	min. aircraft mass (basic empty weight)
massmin	m_{max}	[1,1]	[kg]	'ac_data'	max. aircraft weight (max. take-off weight)
neng	n_{eng}	[1,1]	[-]	'ac_data'	number of engines
span	b	[1,1]	[m]	'ac_data'	wing span
tilteng	t_i	[1,?]	[rad]	'ac_data'	row vector with tilt angles of thrustline
toweng	t_o	[1,?]	[rad]	'ac_data'	row vector with tow angles of thrustlines
umax	$u_{a_{max}}$	[1,8]		'ac_data'	row vector with max. values of aerodynamic control inputs
umin	$u_{a_{min}}$	[1,8]		'ac_data'	row vector with min. values of aerodynamic control inputs
xcgref	$x_{cg_{ref}}$	[1,1]	[m]	'ac_data'	position of reference center of gravity along datum X-axis
ycgref	$y_{cg_{ref}}$	[1,1]	[m]	'ac_data'	position of reference center of gravity along datum Y-axis
zcgref	$z_{cg_{ref}}$	[1,1]	[m]	'ac_data'	position of reference center of gravity along datum Z-axis
xeng	x_{eng}	[1,?]	[m]	'ac_data'	row vector with positions of engines along datum X-axis
yeng	y_{eng}	[1,?]	[m]	'ac_data'	row vector with positions of engines along datum Y-axis
zeng	z_{eng}	[1,?]	[m]	'ac_data'	row vector with positions of engines along datum Z-axis
axes_CFMaero		[1,1]		'ac_data'	specifier of reference frame for aerodynamic forces/moments from aerodynamic model [0/1/2/3] is [air-path/stability/body/earth]-axes
axes_CFMgust		[1,1]		'ac_data'	specifier of reference frame for forces/moments due to turbulence from aerodynamic model [0/1/2/3] is [air-path/stability/body/earth]-axes
Tnmax	$T_{N_{max}}$	[1,1]	[N]	'eng_data'	max. value of engine thrust control input
Tnmin	$T_{N_{min}}$	[1,1]	[N]	'eng_data'	min. value of engine thrust control input
utmax	$u_{t_{max}}$	[1,1]	[rad]	'eng_data'	max. value of engine throttle control input
utmin	$u_{t_{min}}$	[1,1]	[rad]	'eng_data'	min. value of engine throttle ontrol input
xtndx		[1,?]		'eng_data'	row vector with indices of engine states in engine output vector

Table C.8: Variables which specify navigation ground station and measurement locations.

name	symbol	size	dim	file	description
xDME	x_{DME}	[1,1]	[m]	ac_genrl.mat	location of DME beacon along earth X-axis
yDME	x_{DME}	[1,1]	[m]	ac_genrl.mat	location of DME beacon along earth Y-axis
hDME	h_{DME}	[1,1]	[m]	ac_genrl.mat	geometric altitude of DME beacon
xGS	x_{GS}	[1,1]	[m]	ac_genrl.mat	location of glide-slope antenna along earth X-axis
yGS	x_{GS}	[1,1]	[m]	ac_genrl.mat	location of glide-slope antenna along earth Y-axis
hGS	h_{GS}	[1,1]	[m]	ac_genrl.mat	geometric altitude of glide-slope antenna
xLOC	x_{LOC}	[1,1]	[m]	ac_genrl.mat	location of localizer antenna along earth X-axis
yLOC	x_{LOC}	[1,1]	[m]	ac_genrl.mat	location of localizer antenna along earth Y-axis
psiLOC	ψ_{LOC}	[1,1]	[m]	ac_genrl.mat	horizontal direction of localizer antenna
xiabh	$x_{i_{\alpha\beta h}}$	[1,1]	[m]	ac_genrl.mat	location of sensor for α,β or h_e along datum X-axis
yiabh	$y_{i_{\alpha\beta h}}$	[1,1]	[m]	ac_genrl.mat	location of sensor for α,β or h_e along datum Y-axis
ziabh	$z_{i_{\alpha\beta h}}$	[1,1]	[m]	ac_genrl.mat	location of sensor for α,β or h_e along datum Z-axis
xiacc	$x_{i_{acc}}$	[1,1]	[m]	ac_genrl.mat	location of accelerometer along datum X-axis
yiacc	$y_{i_{acc}}$	[1,1]	[m]	ac_genrl.mat	location of accelerometer along datum Y-axis
ziacc	$z_{i_{acc}}$	[1,1]	[m]	ac_genrl.mat	location of accelerometer along datum Z-axis

Table C.9: Variables which specify atmospheric and turbulence constants.

name	symbol	size	dim	file	description
GASCON	R	[1,1]	[m^2/s^2K]	ac_genrl.mat	specific gas constant of air
Hs	H_s	[1,1]	[m]	ac_genrl.mat	geopotential altitude of tropopause
RADIUS	R_e	[1,1]	[m]	ac_genrl.mat	radius of earth
T0	T_0	[1,1]	[K]	ac_genrl.mat	ambient temperature at sea-level
g0	g_0	[1,1]	[kg/m^2]	ac_genrl.mat	acceleration of gravity at sea-level
gammah	γ	[1,1]	[-]	ac_genrl.mat	ratio of specific heats of air
lambda	λ	[1,1]	[K/m]	ac_genrl.mat	temperature gradient with geopotential altitude in troposphere
pstat0	p_0	[1,1]	[N/m^2]	ac_genrl.mat	ambient pressure at sea-level
rho0	ρ_0	[1,1]	[kg/m^3]	ac_genrl.mat	air density at sea-level
Iag	I_{α_g}	[1,1]	[rad^2]	ac_init.m	power spectral density function of α_g (asymmetric motions)
Iug	$I_{\hat{u}_g}$	[1,1]	[-]	ac_init.m	power spectral density function of $\hat{u}_g$ (symmetric motions)
Lg	L_g	[1,1]	[m]	ac_turb.mat	scale of turbulence
sigmag	σ_g	[1,1]	[m/s]	ac_turb.mat	standard deviation of gust velocities
taug	$\tau_{1...6}$	[1,6]	[-]	ac_init.m	constants in power spectral density of $\hat{u}_g$ and α_g

Table C.10: Variables which specify aircraft geometry for animation.

name	symbol	size	dim.	file	description
FUSEX	x_{fus}	[?,?]	[m]	ac_geom.mat	x-coordinates along body axis of fuselage surface
FUSEY	y_{fus}	[?,?]	[m]	ac_geom.mat	y-coordinates along body axis of fuselage surface
FUSEZ	z_{fus}	[?,?]	[m]	ac_geom.mat	z-coordinates along body axis of fuselage surface
LW		[3,?]	[m]	ac_geom.mat	x, y, z-coordinates along body axes of left wing planform vertices
RW		[3,?]	[m]	ac_geom.mat	x, y, z-coordinates along body axes of right wing planform vertices
LTP		[3,?]	[m]	ac_geom.mat	x, y, z-coordinates along body axes of left horizontal tailplane planform vertices
RTP		[3,?]	[m]	ac_geom.mat	x, y, z-coordinates along body axes of left horizontal tailplane planform vertices
TP		[3,?]	[m]	ac_geom.mat	x, y, z-coordinates along body-axes of vert tailplane planform vertices

Appendix D

Definition of reference frames and outlines of Simulink models

This appendix discusses the outlines of the SIMULINK models of the generic aircraft model, the standard sub-models and the user-supplied aircraft specific sub-models. Further, the applied reference frames are defined first.
The SIMULINK models are hierarchically subdivided in several levels, especially the generic aircraft model. These levels will be discussed in a top-down way, giving the format of the *Inport/Outport* blocks, any included S-functions and required variables in workspace, a brief discussion of the tasks and the applied formulas.

D.1 Definition of reference frames

For describing the motion of the aircraft as well as the forces and moments on the aircraft, five reference frames are used. Furthermore, a seperate reference frame is used for specifying the locations of the aircraft components:

- fixed earth reference frame ($X_eY_eZ_e$)
- moving earth reference frame ($X_eY_eZ_e$)
- body reference frame ($X_bY_bZ_b$)
- stability reference frame ($X_sY_sZ_s$)
- air-path reference frame ($X_aY_aZ_a$)
- datum reference frame ($X_dY_dZ_d$)

D.1.1 Fixed earth reference frame

The fixed earth reference frame is a right-handed, rectangular Cartesian reference frame which has its origin at an arbitrary but fixed position on the surface of the earth. By default, the origin is located at the projection of the aircraft's center of gravity on the earth's surface at the start of a simulation. The Z_e-axis points downwards parallel to the local direction of the gravitation, so that the X_eY_e-plane is tangent to the earth's surface. The X_e-axis points to the North, the Y_e-axis points to the East.
The fixed earth reference frame is used as an inertial reference frame. The aircraft's location and orientation are expressed in this reference frame. It is refered to by the subscript $_e$.

D.1.2 Moving earth reference frame

The moving earth reference frame has identical orientation to the fixed earth reference frame, but its origin retains at the aircraft's center of gravity during the simulation. The moving earth reference frame therefore translates with the aircraft in the fixed earth reference frame. The moving earth reference frame is primarily used to express the orientation of the aircraft and the direction of the gravity force. It is also denoted by the subscript $_e$.

D.1.3 Body reference frame

The body reference frame is a right-handed, rectangular Cartesian reference frame which is centered at the aircraft's center of gravity and has a fixed alignment relative to the aircraft body. The X_bY_b-plane coincides with the aircraft's plane of symmetry. The X_b-axis points forward, the Y_b-axis points to starboard and the Z_b-axis points downward in normal flight. All body axes are anti-parallel to the axes of the datum reference frame.
The body reference frame is used for the derivation of the moment equations of motion. It is denoted by the subscript $_b$.

D.1.4 Stability reference frame

The stability reference frame is not a body fixed reference frame. The origin is located at the aircraft's center of gravity. The Y_s-axis coincides with the Y_b-axis. The X_s- and Z_s-axis are defined by performing a plane rotation about the negative Y_b-axis over the angle of attack. The projection of the velocity vector relative to the atmosphere (true airspeed) on the aircraft's plane of symmetry is therefore continuously parallel to the stability X_s-axis.
The stability reference frame is often used for expressing the aerodynamic force and moment coefficients. It is denoted by the subscript $_s$.

D.1.5 Air-path reference frame

The air-path reference frame has its origin fixed to the aircraft's center of gravity and its X_a-axis is directed along the velocity vector relative to the atmosphere (true airspeed). The Z_a lies in the aircraft's plane of symmetry and coincides with the Z_s-axis. The X_a- and Y_a-axis are related to the X_s- and Y_s-axis by performing a plane rotation about the positive Z_s-axis over the angle of sideslip.
The air-path reference frame is used for the derivation of the translational equations of motion. It is denoted by the subscript $_a$.

D.1.6 Datum reference frame

The datum refence frame is a left-handed, rectangular Cartesian reference frame. The origin is usually located in the aircraft's plane of symmetry but outside the physical dimensions of the aircraft. The X_d-axis is in the plane of symmetry, the Y_d-axis is perpendicular to this plane of symmetry and points to port. The direction of the Z_d-axis is upwards in normal flight. The datum axes are therefore anti-parallel to the body axes.
The datum reference frame is used for defining the locations of characteristic points relative to the aircraft, for example the aircraft's center of gravity and sensors.

D.2 Generic aircraft model ac_mod.m

AC_MOD

inports	ua	Table B.2
	ut	Table B.2
outports	y	Table B.3
	ypow	Table B.5
blocks	AIRCRAFT MODEL	
variables	massinit	

AIRCRAFT MODEL

inports	ua	$[\delta_e \ \delta_a \ \delta_r \ \delta_{t_e} \ \delta_{t_a} \ \delta_{t_r} \ \delta_f \ \ell g_{sw}]$
	ut	$\begin{cases} [PLA_1 \ PLA_2] & \text{(engine model included)} \\ [T_{N_1} \ T_{N_2}] & \text{(engine model not included)} \end{cases}$
	massinit	$[m \ x_{cg} \ y_{cg} \ z_{cg} \ I_{xx} \ I_{yy} \ I_{zz} \ I_{xz}]$
outports	x, xdot, yatm, yad1, yad2, wind, gust, yacc, yfp, ys, ypqr, yuvw, yuvwdot, ydl, yabh, yCaero, yFMaero, yCgust, yFMgust, yCt, yFMt, yFgrav	
blocks	AIRDATA, WIND/TURBULENCE, AFM, EFM, FM SORT, EQM, OBSERVATIONS	

The generic aircraft model calculates a number of observation outputs from aerodynamic and thrust control inputs, using the 6-DOF nonlinear equations of motion for a rigid body aircraft with constant mass over a flat nonrotating earth. The representation of the state variables and the equations of motion are adopted from [15]. The set of observation outputs is an extensively elaborated set initially taken from the NASA-program *LINEAR* [6].

This section discusses only the general aircraft model ac_mod.m. The outline of the compact aircraft model ac_modpc.m is similar, but with the omission of less significant observation signals and blocks. The equations are however the same.

The block AIRCRAFT MODEL is the highest sub-system level of of the generic aircraft model. Its blocks will be discussed in the sections below.

D.2.1 AIRDATA

inports	x	$[p_b \ q_b \ r_b \ V_{TAS} \ \alpha \ \beta \ \phi \ \theta \ \psi \ h_e \ x_e \ y_e]$
outports	yatm	$[p_a \ \rho \ T \ g \ h_p \ h_R \ H \ V_{sound}]$
	yad1	$[M \ \bar{q}]$
	yad2	$[Re' \ q_c \ q_{rel} \ p_t \ T_t \ V_{EAS} \ V_{CAS} \ V_{IAS}]$
S-functions	ac_atmos.mex*	

In this block the atmospheric and airdata parameters are calculated. The equations are compiled in the MEX-type S-function ac_atmos.mex*.
The applied equations are given in Section D.3.1 *Atmosphere and airdata model* ac_atmos.mex*.

D.2.2 WIND/TURBULENCE

inports	x	$[p_b \ q_b \ r_b \ V_{TAS} \ \alpha \ \beta \ \phi \ \theta \ \psi \ h_e \ x_e \ y_e]$
outports	wind	$[\vec{V}_{w_a} \ \vec{V}_{w_s} \ \vec{V}_{w_b} \ \vec{V}_{w_e}]$
	gust	$[\hat{u}_g \ \alpha_g \ \beta_g \ \hat{u}_g\bar{c}/V \ \dot{\alpha}_g\bar{c}/V \ \dot{\beta}_g b/V \ \hat{u}_{gasym} \ \alpha_{gasym}]$
S-functions	ac_axes.mex*, (wnd_none.m), (tur_none.m), (tur_dryd.m)	

In this block the wind and gust velocities are calculated from user-supplied S-functions for the wind and turbulence models. By default, the S-functions `wnd_none.m` and `tur_none.m` for zero wind and zero turbulence are included. Moreover, the S-function `tur_dryd.m` with a turbulence model based on Dryden spectra is supplied in the *DASMAT* directory.
The applied equations for the wind and turbulence models are given in Section D.3.3 *Wind model (`wnd_none.m`)* and Section D.3.4 *Turbulence models (`tur_none.m`), (`tur_dryd.m`)* respectively.

D.2.3 AFM

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`ua`	$[\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}]$
	`massinit`	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
	`yad1`	$[M \;\; \bar{q}]$
	`yad2`	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
	`gust`	$[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}]$
outports	`Ca`	$[C_{\vec{F}_{base_a}} \;\; C_{\vec{M}_{base_a}} \;\; C_{\vec{F}_{base_s}} \;\; C_{\vec{M}_{base_s}} \;\; C_{\vec{F}_{base_b}} \;\; C_{\vec{M}_{base_b}} \;\; C_{\vec{F}_{base_e}} \;\; C_{\vec{M}_{base_e}}]$
	`FMa`	$[\vec{F}_{base_a} \;\; \vec{M}_{base_a} \;\; \vec{F}_{base_s} \;\; \vec{M}_{base_s} \;\; \vec{F}_{base_b} \;\; \vec{M}_{base_b} \;\; \vec{F}_{base_e} \;\; \vec{M}_{base_e}]$
	`Cadot`	$[C_{\vec{F}_{\dot{\alpha}_a}} \;\; C_{\vec{M}_{\dot{\alpha}_a}} \;\; C_{\vec{F}_{\dot{\alpha}_s}} \;\; C_{\vec{M}_{\dot{\alpha}_s}} \;\; C_{\vec{F}_{\dot{\alpha}_b}} \;\; C_{\vec{M}_{\dot{\alpha}_b}} \;\; C_{\vec{F}_{\dot{\alpha}_e}} \;\; C_{\vec{M}_{\dot{\alpha}_e}}]$
	`FMadot`	$[\vec{F}_{\dot{\alpha}_a} \;\; \vec{M}_{\dot{\alpha}_a} \;\; \vec{F}_{\dot{\alpha}_s} \;\; \vec{M}_{\dot{\alpha}_s} \;\; \vec{F}_{\dot{\alpha}_b} \;\; \vec{M}_{\dot{\alpha}_b} \;\; \vec{F}_{\dot{\alpha}_e} \;\; \vec{M}_{\dot{\alpha}_e}]$
	`Cbdot`	$[C_{\vec{F}_{\dot{\beta}_a}} \;\; C_{\vec{M}_{\dot{\beta}_a}} \;\; C_{\vec{F}_{\dot{\beta}_s}} \;\; C_{\vec{M}_{\dot{\beta}_s}} \;\; C_{\vec{F}_{\dot{\beta}_b}} \;\; C_{\vec{M}_{\dot{\beta}_b}} \;\; C_{\vec{F}_{\dot{\beta}_e}} \;\; C_{\vec{M}_{\dot{\beta}_e}}]$
	`FMbdot`	$[\vec{F}_{\dot{\beta}_a} \;\; \vec{M}_{\dot{\beta}_a} \;\; \vec{F}_{\dot{\beta}_s} \;\; \vec{M}_{\dot{\beta}_s} \;\; \vec{F}_{\dot{\beta}_b} \;\; \vec{M}_{\dot{\beta}_b} \;\; \vec{F}_{\dot{\beta}_e} \;\; \vec{M}_{\dot{\beta}_e}]$
	`Cg`	$[C_{\vec{F}_{g_a}} \;\; C_{\vec{M}_{g_a}} \;\; C_{\vec{F}_{g_s}} \;\; C_{\vec{M}_{g_s}} \;\; C_{\vec{F}_{g_b}} \;\; C_{\vec{M}_{g_b}} \;\; C_{\vec{F}_{g_e}} \;\; C_{\vec{M}_{g_e}}]$
	`FMg`	$[\vec{F}_{g_a} \;\; \vec{M}_{g_a} \;\; \vec{F}_{g_s} \;\; \vec{M}_{g_s} \;\; \vec{F}_{g_b} \;\; \vec{M}_{g_b} \;\; \vec{F}_{g_e} \;\; \vec{M}_{g_e}]$
blocks	`Aeromod, Aero axes-cg transformation, FMaerodims, Gust axes-cg transformation, FMgustdims`	

In this block the aerodynamic forces and moments and the forces and moments due to turbulence are calculated. These signals are derived via a user-supplied aerodynamic model. The block is consists of sub-blocks which are described next.

Aeromod

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`ua`	$[\delta_e \;\; \delta_a \;\; \delta_r \;\; \delta_{t_e} \;\; \delta_{t_a} \;\; \delta_{t_r} \;\; \delta_f \;\; \ell g_{sw}]$
	`massinit`	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
	`yad1`	$[M \;\; \bar{q}]$
	`yad2`	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
	`gust`	$[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}]$
outports	`Ca`	$[C_{\vec{F}_{base_m}} \;\; C_{\vec{M}_{base_m}}]$
	`Cadot`	$[C_{\vec{F}_{\dot{\alpha}_m}} \;\; C_{\vec{M}_{\dot{\alpha}_m}}]$
	`Cbdot`	$[C_{\vec{F}_{\dot{\beta}_m}} \;\; C_{\vec{M}_{\dot{\beta}_m}}]$
	`Cg`	$[C_{\vec{F}_{g_m}} \;\; C_{\vec{M}_{g_m}}]$
S-functions	`ac_aero.m`	

In this block the aerodynamic force and moment coefficients and the force and moment coefficients due to turbulence are calculated. This is accomplished by including a user-defined S-function of the aerodynamic model in an *S-function* block. The default S-function `ac_aero.m` is a template, see also Section D.4.1 *Aircraft specific aerodynamic model* `ac_aeromodel`.

Aero axes-cg transformation

inports	`Ca`	$[C_{\vec{F}_{base_m}} \quad C_{\vec{M}_{base_m}}]$
	`Cadot`	$[C_{\vec{F}_{\dot{\alpha}_m}} \quad C_{\vec{M}_{\dot{\alpha}_m}}]$
	`Cbdot`	$[C_{\vec{F}_{\dot{\beta}_m}} \quad C_{\vec{M}_{\dot{\beta}_m}}]$
	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`massinit`	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
outports	`Ca`	$[C_{\vec{F}_{base_a}} \quad C_{\vec{M}_{base_a}} \quad C_{\vec{F}_{base_s}} \quad C_{\vec{M}_{base_s}} \quad C_{\vec{F}_{base_b}} \quad C_{\vec{M}_{base_b}} \quad C_{\vec{F}_{base_e}} \quad C_{\vec{M}_{base_e}}]$
	`Cadot`	$[C_{\vec{F}_{\dot{\alpha}_a}} \quad C_{\vec{M}_{\dot{\alpha}_a}} \quad C_{\vec{F}_{\dot{\alpha}_s}} \quad C_{\vec{M}_{\dot{\alpha}_s}} \quad C_{\vec{F}_{\dot{\alpha}_b}} \quad C_{\vec{M}_{\dot{\alpha}_b}} \quad C_{\vec{F}_{\dot{\alpha}_e}} \quad C_{\vec{M}_{\dot{\alpha}_e}}]$
	`Cbdot`	$[C_{\vec{F}_{\dot{\beta}_a}} \quad C_{\vec{M}_{\dot{\beta}_a}} \quad C_{\vec{F}_{\dot{\beta}_s}} \quad C_{\vec{M}_{\dot{\beta}_s}} \quad C_{\vec{F}_{\dot{\beta}_b}} \quad C_{\vec{M}_{\dot{\beta}_b}} \quad C_{\vec{F}_{\dot{\beta}_e}} \quad C_{\vec{M}_{\dot{\beta}_e}}]$
S-functions	`ac_axes.mex*`	
variables	`xcgref, ycgref, zcgref, chord, span, axes_CFMaero`	

This block calculates the aerodynamic force and moment coefficients with respect to the aircraft center of gravity and the air-path, stability, body and moving earth reference frames.

The aerodynamic coefficients are expressed as baseline component C_{base} and components $C_{\dot{\alpha}}$ and $C_{\dot{\beta}}$ due to linear dependency on $\frac{\dot{\alpha}\bar{c}}{V}$ and $\frac{\dot{\beta}b}{V}$. The presented equations apply to each of these components and the subscripts $_{base}$, $_{\dot{\alpha}}$ and $_{\dot{\beta}}$ are therefore omitted. Moreover, the subscript $_m$ refers to reference frame in which the coefficients are supplied by the included user-defined aerodynamic sub-model.
The aerodynamic force coefficients are directly transformed to all reference frames according to:

$$\begin{pmatrix} C_{\vec{F}_a} \\ C_{\vec{F}_s} \\ C_{\vec{F}_b} \\ C_{\vec{F}_e} \end{pmatrix} = \begin{bmatrix} R_{am} \\ R_{sm} \\ R_{bm} \\ R_{em} \end{bmatrix} C_{\vec{F}_m} \tag{D.1}$$

The aerodynamic moment coefficients are corrected in the body reference frame for the offset between the reference location for the aerodynamic forces and the center of gravity:

$$\begin{aligned} \Delta C_{\ell_b} &= \frac{C_{Y_b}(z_{cg_{ref}} - z_{cg}) - C_{Z_b}(y_{cg_{ref}} - y_{cg})}{b} \\ \Delta C_{m_b} &= \frac{-C_{X_b}(z_{cg_{ref}} - z_{cg}) + C_{Z_b}(x_{cg_{ref}} - x_{cg})}{\bar{c}} \\ \Delta C_{n_b} &= \frac{C_{X_b}(y_{cg_{ref}} - y_{cg}) - C_{Y_b}(x_{cg_{ref}} - x_{cg})}{b} \end{aligned} \tag{D.2}$$

The aerodyamic moment coefficients and their corrections are next transformed to all reference frames and subsequently added:

$$\begin{pmatrix} C_{\vec{M}_a} \\ C_{\vec{M}_s} \\ C_{\vec{M}_b} \\ C_{\vec{M}_e} \end{pmatrix} = \begin{bmatrix} R_{am} \\ R_{sm} \\ R_{bm} \\ R_{eb} \end{bmatrix} C_{\vec{M}_m} + \begin{bmatrix} R_{ab} \\ R_{sb} \\ I \\ R_{eb} \end{bmatrix} \begin{pmatrix} \Delta C_{\ell_b} \\ \Delta C_{m_b} \\ \Delta C_{n_b} \end{pmatrix} \tag{D.3}$$

FMaerodims

inports	`Ca`	$[C_{\vec{F}_{base_a}} \; C_{\vec{M}_{base_a}} \; C_{\vec{F}_{base_s}} \; C_{\vec{M}_{base_s}} \; C_{\vec{F}_{base_b}} \; C_{\vec{M}_{base_b}} \; C_{\vec{F}_{base_e}} \; C_{\vec{M}_{base_e}}]$
	`Cadot`	$[C_{\vec{F}_{\dot{\alpha}_a}} \; C_{\vec{M}_{\dot{\alpha}_a}} \; C_{\vec{F}_{\dot{\alpha}_s}} \; C_{\vec{M}_{\dot{\alpha}_s}} \; C_{\vec{F}_{\dot{\alpha}_b}} \; C_{\vec{M}_{\dot{\alpha}_b}} \; C_{\vec{F}_{\dot{\alpha}_e}} \; C_{\vec{M}_{\dot{\alpha}_e}}]$
	`Cbdot`	$[C_{\vec{F}_{\dot{\beta}_a}} \; C_{\vec{M}_{\dot{\beta}_a}} \; C_{\vec{F}_{\dot{\beta}_s}} \; C_{\vec{M}_{\dot{\beta}_s}} \; C_{\vec{F}_{\dot{\beta}_b}} \; C_{\vec{M}_{\dot{\beta}_b}} \; C_{\vec{F}_{\dot{\beta}_e}} \; C_{\vec{M}_{\dot{\beta}_e}}]$
	`x`	$[p_b \; q_b \; r_b \; V_{TAS} \; \alpha \; \beta \; \phi \; \theta \; \psi \; h_e \; x_e \; y_e]$
	`yad1`	$[M \; \bar{q}]$
outports	`FMa`	$[\vec{F}_{base_a} \; \vec{M}_{base_a} \; \vec{F}_{base_s} \; \vec{M}_{base_s} \; \vec{F}_{base_b} \; \vec{M}_{base_b} \; \vec{F}_{base_e} \; \vec{M}_{base_e}]$
	`FMadot`	$[\vec{F}_{\dot{\alpha}_a} \; \vec{M}_{\dot{\alpha}_a} \; \vec{F}_{\dot{\alpha}_s} \; \vec{M}_{\dot{\alpha}_s} \; \vec{F}_{\dot{\alpha}_b} \; \vec{M}_{\dot{\alpha}_b} \; \vec{F}_{\dot{\alpha}_e} \; \vec{M}_{\dot{\alpha}_e}]$
	`FMbdot`	$[\vec{F}_{\dot{\beta}_a} \; \vec{M}_{\dot{\beta}_a} \; \vec{F}_{\dot{\beta}_s} \; \vec{M}_{\dot{\beta}_s} \; \vec{F}_{\dot{\beta}_b} \; \vec{M}_{\dot{\beta}_b} \; \vec{F}_{\dot{\beta}_e} \; \vec{M}_{\dot{\beta}_e}]$
variables	`S, chord, span`	

This block calculates the aerodynamic forces and moments from their dimensionless coefficients.

The presented equations apply to forces and moments in any reference frame. The subscripts which refer to the reference frames are therefore omitted.
The baseline aerodynamic forces and moments directly follow from multiplying their coefficients with the dynamic pressure and wing area. For the moments an additional multiplication with a characteristic length is performed:

$$\begin{aligned} X_{base} &= C_{X_{base}} \, \bar{q} S & \bar{L}_{base} &= C_{\ell_{base}} \, \bar{q} S \, b \\ Y_{base} &= C_{Y_{base}} \, \bar{q} S & M_{base} &= C_{m_{base}} \, \bar{q} S \, \bar{c} \\ Z_{base} &= C_{Z_{base}} \, \bar{q} S & \bar{N}_{base} &= C_{n_{base}} \, \bar{q} S \, b \end{aligned} \tag{D.4}$$

The $\dot{\alpha}$- and $\dot{\beta}$-contributions in the aerodynamic forces and moments require yet an additional multiplication factor. The forces and moments are linear in $\dot{\alpha}$ and $\dot{\beta}$, whereas the coefficients are linear in $\frac{\dot{\alpha}\bar{c}}{V}$ and $\frac{\dot{\beta}b}{V}$. Hence:

$$\begin{aligned} X_{\dot{\alpha}} &= C_{X_{\dot{\alpha}}} \, \bar{q} S \, \frac{\bar{c}}{V_{TAS}} & \bar{L}_{\dot{\alpha}} &= C_{\ell_{\dot{\alpha}}} \, \bar{q} S \, b \, \frac{\bar{c}}{V_{TAS}} \\ Y_{\dot{\alpha}} &= C_{Y_{\dot{\alpha}}} \, \bar{q} S \, \frac{\bar{c}}{V_{TAS}} & M_{\dot{\alpha}} &= C_{m_{\dot{\alpha}}} \, \bar{q} S \, \bar{c} \, \frac{\bar{c}}{V_{TAS}} \\ Z_{\dot{\alpha}} &= C_{Z_{\dot{\alpha}}} \, \bar{q} S \, \frac{\bar{c}}{V_{TAS}} & \bar{N}_{\dot{\alpha}} &= C_{n_{\dot{\alpha}}} \, \bar{q} S \, b \, \frac{\bar{c}}{V_{TAS}} \end{aligned} \tag{D.5}$$

and:

$$
\begin{aligned}
X_{\dot{\beta}} &= C_{X_{\dot{\beta}}}\, \bar{q}S\, \frac{b}{V_{TAS}} & \bar{L}_{\dot{\beta}} &= C_{\ell_{\dot{\beta}}}\, \bar{q}S\, b\, \frac{b}{V_{TAS}} \\
Y_{\dot{\beta}} &= C_{Y_{\dot{\beta}}}\, \bar{q}S\, \frac{b}{V_{TAS}} & M_{\dot{\beta}} &= C_{m_{\dot{\beta}}}\, \bar{q}S\, \bar{c}\, \frac{b}{V_{TAS}} \\
Z_{\dot{\beta}} &= C_{Z_{\dot{\beta}}}\, \bar{q}S\, \frac{b}{V_{TAS}} & \bar{N}_{\dot{\beta}} &= C_{n_{\dot{\beta}}}\, \bar{q}S\, b\, \frac{b}{V_{TAS}}
\end{aligned}
\tag{D.6}
$$

Gust axes-cg transformation

inports	`Cg`	$[C_{\vec{F}_{gm}} \;\; C_{\vec{M}_{gm}}]$
	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`massinit`	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
outports	`Cg`	$[C_{\vec{F}_{ga}} \;\; C_{\vec{M}_{ga}} \;\; C_{\vec{F}_{gs}} \;\; C_{\vec{M}_{gs}} \;\; C_{\vec{F}_{gb}} \;\; C_{\vec{M}_{gb}} \;\; C_{\vec{F}_{ge}} \;\; C_{\vec{M}_{ge}}]$
S-functions	`ac_axes.mex*`	
variables	`xcgref, ycgref, zcgref, chord, span, axes_CFMgust`	

This block calculates the force and moment coefficients due to turbulence with respect to the aircraft center of gravity and the air-path, stability, body and moving earth reference frames.

The block has a simular structure as the block `Aero axes-cg tranformation` in Section D.2.3 for calculating the aerodynamic coefficients. The equations are identical and just repeated for completeness. Again, the subscript $_m$ refers to reference frame in which the coefficients are supplied by the included user-defined aerodynamic sub-model.

The force coefficients due to turbulence are directly transformed to all reference frames according to:

$$
\begin{pmatrix} C_{\vec{F}_{ga}} \\ C_{\vec{F}_{gs}} \\ C_{\vec{F}_{gb}} \\ C_{\vec{F}_{ge}} \end{pmatrix} = \begin{bmatrix} R_{am} \\ R_{sm} \\ R_{bm} \\ R_{em} \end{bmatrix} C_{\vec{F}_{gm}}
\tag{D.7}
$$

The moment coefficients due to turbulence are corrected in the body reference frame for the offset between the reference location for the forces due to turbulence and the center of gravity. It is assumed that this reference location is equal to the reference location for the aerodynamic forces. Hence:

$$
\begin{aligned}
\Delta C_{\ell_{g_b}} &= \frac{C_{Y_{g_b}}(z_{cg_{ref}} - z_{cg}) - C_{Z_{g_b}}(y_{cg_{ref}} - y_{cg})}{b} \\
\Delta C_{m_{g_b}} &= \frac{-C_{X_{g_b}}(z_{cg_{ref}} - z_{cg}) + C_{Z_{g_b}}(x_{cg_{ref}} - x_{cg})}{\bar{c}} \\
\Delta C_{n_{g_b}} &= \frac{C_{X_{g_b}}(y_{cg_{ref}} - y_{cg}) - C_{Y_{g_b}}(x_{cg_{ref}} - x_{cg})}{b}
\end{aligned}
\tag{D.8}
$$

The moment coefficients due to turbulence and their corrections are next transformed to all reference frames and subsequently added:

$$
\begin{pmatrix} C_{\vec{M}_{g_a}} \\ C_{\vec{M}_{g_s}} \\ C_{\vec{M}_{g_b}} \\ C_{\vec{M}_{g_e}} \end{pmatrix} = \begin{bmatrix} R_{am} \\ R_{sm} \\ R_{bm} \\ R_{eb} \end{bmatrix} C_{\vec{M}_{g_m}} + \begin{bmatrix} R_{ab} \\ R_{sb} \\ I \\ R_{eb} \end{bmatrix} \begin{pmatrix} \Delta C_{\ell_{g_b}} \\ \Delta C_{m_{g_b}} \\ \Delta C_{n_{g_b}} \end{pmatrix} \tag{D.9}
$$

FMgustdims

inports	`Cg`	$[C_{\vec{F}_{g_a}} \; C_{\vec{M}_{g_a}} \; C_{\vec{F}_{g_s}} \; C_{\vec{M}_{g_s}} \; C_{\vec{F}_{g_b}} \; C_{\vec{M}_{g_b}} \; C_{\vec{F}_{g_e}} \; C_{\vec{M}_{g_e}}]$
	`yad1`	$[M \;\; \bar{q}]$
outports	`FMg`	$[\vec{F}_{g_a} \; \vec{M}_{g_a} \; \vec{F}_{g_s} \; \vec{M}_{g_s} \; \vec{F}_{g_b} \; \vec{M}_{g_b} \; \vec{F}_{g_e} \; \vec{M}_{g_e}]$
variables	`S, chord, span`	

This block calculates the forces and moments due to turbulence from their dimensionless coefficients.

The block structure is a reduced copy of `FMaerodims` in Section D.2.3, using only the part for calculating the baseline aerodynamic forces and moments. The equations are similar to (D.4) and apply again to any reference frame. Omitting subscripts for the reference frames, this gives:

$$
\begin{aligned}
X_g &= C_{X_g}\, \bar{q}S & \bar{L}_g &= C_{\ell_g}\, \bar{q}S\, b \\
Y_g &= C_{Y_g}\, \bar{q}S & M_g &= C_{m_g}\, \bar{q}S\, \bar{c} \\
Z_g &= C_{Z_g}\, \bar{q}S & \bar{N}_g &= C_{n_g}\, \bar{q}S\, b
\end{aligned}
\tag{D.10}
$$

D.2.4 EFM

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	ut	$\begin{cases} [PLA_1 \;\; PLA_2] & \text{(engine model included)} \\ [T_{N_1} \;\; T_{N_2}] & \text{(engine model not included)} \end{cases}$
	massinit	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
	yatm	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
	yad1	$[M \;\; \bar{q}]$
	yad2	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
	gust	$[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}]$
outports	Ct	$[C_{\vec{F}_{t_a}} \;\; C_{\vec{M}_{t_a}} \;\; C_{\vec{F}_{t_s}} \;\; C_{\vec{M}_{t_s}} \;\; C_{\vec{F}_{t_b}} \;\; C_{\vec{M}_{t_b}} \;\; C_{\vec{F}_{t_e}} \;\; C_{\vec{M}_{t_e}}]$
	FMt	$[\vec{F}_{t_a} \;\; \vec{M}_{t_a} \;\; \vec{F}_{t_s} \;\; \vec{M}_{t_s} \;\; \vec{F}_{t_b} \;\; \vec{M}_{t_b} \;\; \vec{F}_{t_e} \;\; \vec{M}_{t_e}]$
blocks	`Engmod, Engine axes-cg transformation, FMengnondims`	

In this block the propulsion forces and moments are calculated. These signals are derived via a user-supplied propulsion model. The block is consists of sub-blocks which are described next.

Engmod

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	ut	$\begin{cases} [PLA_1 \;\; PLA_2] & \text{(engine model included)} \\ [T_{N_1} \;\; T_{N_2}] & \text{(engine model not included)} \end{cases}$
	massinit	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
	yatm	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
	yad1	$[M \;\; \bar{q}]$
	yad2	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
outports	FMt	$[\vec{F}_{t_b} \;\; \vec{M}_{t_b}]$
S-functions	`ac_pow.m`	

In this block the propulsion forces and moments are obtained from the user-defined S-function of the propulsion model in an *S-function* block. The default S-function `ac_pow.m` is a template, see also Section D.4.2 *Aircraft specific propulsion model* ***ac_powermodel***. The forces and moments should be supplied in the body reference frame.

Engine axes-cg transformation

inports	FMt	$[\vec{F}_{t_b} \;\; \vec{M}_{t_b}]$
	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	massinit	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
outports	FMt	$[\vec{F}_{t_a} \;\; \vec{M}_{t_a} \;\; \vec{F}_{t_s} \;\; \vec{M}_{t_s} \;\; \vec{F}_{t_b} \;\; \vec{M}_{t_b} \;\; \vec{F}_{t_e} \;\; \vec{M}_{t_e}]$
S-functions	`ac_axes.mex*`	

This block calculates the propulsion forces and moments with respect to the aircraft center of gravity and the air-path, stability, body and moving earth reference frames.

The propulsion forces are directly transformed from the body reference frame to all reference frames according to:

$$\begin{pmatrix} \vec{F}_{t_a} \\ \vec{F}_{t_s} \\ \vec{F}_{t_b} \\ \vec{F}_{t_e} \end{pmatrix} = \begin{bmatrix} R_{ab} \\ R_{sb} \\ I \\ R_{eb} \end{bmatrix} \vec{F}_{t_b} \tag{D.11}$$

The propulsion moments are corrected in the body reference frame for the offset between the origin of the datum reference frame and the center of gravity:

$$\begin{aligned} \Delta\bar{L}_{t_b} &= -Y_{t_b} z_{cg} + Z_{t_b} y_{cg} \\ \Delta M_{t_b} &= X_{t_b} z_{cg} - Z_{t_b} x_{cg} \\ \Delta\bar{N}_{t_b} &= -X_{t_b} y_{cg} + Y_{t_b} x_{cg} \end{aligned} \tag{D.12}$$

The moments and their corrections are next transformed to all reference frames and subsequently added:

$$\begin{pmatrix} \vec{M}_{t_a} \\ \vec{M}_{t_s} \\ \vec{M}_{t_b} \\ \vec{M}_{t_e} \end{pmatrix} = \begin{bmatrix} R_{ab} \\ R_{sb} \\ I \\ R_{eb} \end{bmatrix} \vec{M}_{t_b} + \begin{bmatrix} R_{ab} \\ R_{sb} \\ I \\ R_{eb} \end{bmatrix} \begin{pmatrix} \Delta\bar{L}_{t_b} \\ \Delta M_{t_b} \\ \Delta\bar{N}_{t_b} \end{pmatrix} \tag{D.13}$$

FMengnondims

inports	`FMt`	$[\vec{F}_{t_a}\ \vec{M}_{t_a}\ \vec{F}_{t_s}\ \vec{M}_{t_s}\ \vec{F}_{t_b}\ \vec{M}_{t_b}\ \vec{F}_{t_e}\ \vec{M}_{t_e}]$
	`yad1`	$[M\ \bar{q}]$
outports	`Ct`	$[C_{\vec{F}_{t_a}}\ C_{\vec{M}_{t_a}}\ C_{\vec{F}_{t_s}}\ C_{\vec{M}_{t_s}}\ C_{\vec{F}_{t_b}}\ C_{\vec{M}_{t_b}}\ C_{\vec{F}_{t_e}}\ C_{\vec{M}_{t_e}}]$
variables	`S, chord, span`	

This block calculates the propulsion force and moment coefficients from the forces and moments.

The presented equations apply to any reference frame. The subscripts which refer to the reference frames are therefore omitted.

$$\begin{aligned} C_{X_t} &= \frac{X_t}{\bar{q}S} & C_{\ell_t} &= \frac{\bar{L}_t}{\bar{q}S\,b} \\ C_{Y_t} &= \frac{Y_t}{\bar{q}S} & C_{m_t} &= \frac{M_t}{\bar{q}S\,\bar{c}} \\ C_{Z_t} &= \frac{Z_t}{\bar{q}S} & C_{n_t} &= \frac{\bar{N}_t}{\bar{q}S\,b} \end{aligned} \tag{D.14}$$

D.2.5 GRAVITIY

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	massinit	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
	yatm	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
outports	Fgr	$[\vec{F}_{gr_a} \;\; \vec{F}_{gr_s} \;\; \vec{F}_{gr_b} \;\; \vec{F}_{gr_e}]$
S-functions	ac_axes.mex*	

This block calculates the components of the gravity force in the air-path, stability, body and moving earth reference frames.

The gravity force is calculated in the moving earth reference frame from the aircraft mass and the altitude varying acceleration of gravity. It is subsequently transformed to all reference frames:

$$\begin{pmatrix} \vec{F}_{gr_a} \\ \vec{F}_{gr_s} \\ \vec{F}_{gr_b} \\ \vec{F}_{gr_e} \end{pmatrix} = \begin{bmatrix} R_{ae} \\ R_{se} \\ R_{be} \\ I \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} \tag{D.15}$$

D.2.6 FM SORT

inports	FMa	$[\vec{F}_{base_a} \;\; \vec{M}_{base_a} \;\; \vec{F}_{base_s} \;\; \vec{M}_{base_s} \;\; \vec{F}_{base_b} \;\; \vec{M}_{base_b} \;\; \vec{F}_{base_e} \;\; \vec{M}_{base_e}]$
	FMg	$[\vec{F}_{g_a} \;\; \vec{M}_{g_a} \;\; \vec{F}_{g_s} \;\; \vec{M}_{g_s} \;\; \vec{F}_{g_b} \;\; \vec{M}_{g_b} \;\; \vec{F}_{g_e} \;\; \vec{M}_{g_e}]$
	FMt	$[\vec{F}_{t_a} \;\; \vec{M}_{t_a} \;\; \vec{F}_{t_s} \;\; \vec{M}_{t_s} \;\; \vec{F}_{t_b} \;\; \vec{M}_{t_b} \;\; \vec{F}_{t_e} \;\; \vec{M}_{t_e}]$
	Fgr	$[\vec{F}_{gr_a} \;\; \vec{F}_{gr_s} \;\; \vec{F}_{gr_b} \;\; \vec{F}_{gr_e}]$
outports	FMtot	$[\vec{F}^*_{tot_a} \;\; \vec{M}^*_{tot_a} \;\; \vec{F}^*_{tot_s} \;\; \vec{M}^*_{tot_s} \;\; \vec{F}^*_{tot_b} \;\; \vec{M}^*_{tot_b} \;\; \vec{F}^*_{tot_e} \;\; \vec{M}^*_{tot_e}]$

In this block all external generated forces and moments are added.

The equations in this block are just additions. As for the aerodynamic forces, only the baseline terms are used. (The $\dot{\alpha}$- and $\dot{\beta}$-contributions are only used in the equations of motions.) The presented equations apply to all reference frames and the subscripts refering to reference frames are therefore again omitted:

$$\begin{aligned} \vec{F}^*_{tot} &= \vec{F}_{base} + \vec{F}_g + \vec{F}_t + \vec{F}_{gr} \\ \vec{M}^*_{tot} &= \vec{M}_{base} + \vec{M}_g + \vec{M}_t \end{aligned} \tag{D.16}$$

D.2.7 EQM

inports	FMtot	$[\vec{F}^*_{tot_a}\ \vec{M}^*_{tot_a}\ \vec{F}^*_{tot_s}\ \vec{M}^*_{tot_s}\ \vec{F}^*_{tot_b}\ \vec{M}^*_{tot_b}\ \vec{F}^*_{tot_e}\ \vec{M}^*_{tot_e}]$
	Ca	$[C_{\vec{F}_{base_a}}\ C_{\vec{M}_{base_a}}\ C_{\vec{F}_{base_s}}\ C_{\vec{M}_{base_s}}\ C_{\vec{F}_{base_b}}\ C_{\vec{M}_{base_b}}\ C_{\vec{F}_{base_e}}\ C_{\vec{M}_{base_e}}]$
	FMa	$[\vec{F}_{base_a}\ \vec{M}_{base_a}\ \vec{F}_{base_s}\ \vec{M}_{base_s}\ \vec{F}_{base_b}\ \vec{M}_{base_b}\ \vec{F}_{base_e}\ \vec{M}_{base_e}]$
	Cadot	$[C_{\vec{F}_{\dot{\alpha}_a}}\ C_{\vec{M}_{\dot{\alpha}_a}}\ C_{\vec{F}_{\dot{\alpha}_s}}\ C_{\vec{M}_{\dot{\alpha}_s}}\ C_{\vec{F}_{\dot{\alpha}_b}}\ C_{\vec{M}_{\dot{\alpha}_b}}\ C_{\vec{F}_{\dot{\alpha}_e}}\ C_{\vec{M}_{\dot{\alpha}_e}}]$
	FMadot	$[\vec{F}_{\dot{\alpha}_a}\ \vec{M}_{\dot{\alpha}_a}\ \vec{F}_{\dot{\alpha}_s}\ \vec{M}_{\dot{\alpha}_s}\ \vec{F}_{\dot{\alpha}_b}\ \vec{M}_{\dot{\alpha}_b}\ \vec{F}_{\dot{\alpha}_e}\ \vec{M}_{\dot{\alpha}_e}]$
	Cbdot	$[C_{\vec{F}_{\dot{\beta}_a}}\ C_{\vec{M}_{\dot{\beta}_a}}\ C_{\vec{F}_{\dot{\beta}_s}}\ C_{\vec{M}_{\dot{\beta}_s}}\ C_{\vec{F}_{\dot{\beta}_b}}\ C_{\vec{M}_{\dot{\beta}_b}}\ C_{\vec{F}_{\dot{\beta}_e}}\ C_{\vec{M}_{\dot{\beta}_e}}]$
	FMbdot	$[\vec{F}_{\dot{\beta}_a}\ \vec{M}_{\dot{\beta}_a}\ \vec{F}_{\dot{\beta}_s}\ \vec{M}_{\dot{\beta}_s}\ \vec{F}_{\dot{\beta}_b}\ \vec{M}_{\dot{\beta}_b}\ \vec{F}_{\dot{\beta}_e}\ \vec{M}_{\dot{\beta}_e}]$
	massinit	$[m\ x_{cg}\ y_{cg}\ z_{cg}\ I_{xx}\ I_{yy}\ I_{zz}\ I_{xz}]$
	wind	$[\vec{V}_{w_a}\ \vec{V}_{w_s}\ \vec{V}_{w_b}\ \vec{V}_{w_e}]$
outports	x	$[p_b\ q_b\ r_b\ V_{TAS}\ \alpha\ \beta\ \phi\ \theta\ \psi\ h_e\ x_e\ y_e]$
	xdot	$[\dot{p}_b\ \dot{q}_b\ \dot{r}_b\ \dot{V}_{TAS}\ \dot{\alpha}\ \dot{\beta}\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\ \dot{h}_e\ \dot{x}_e\ \dot{y}_e]$
	ypqr	$[\vec{\Omega}_a\ \vec{\Omega}_s\ \vec{\Omega}_b\ \vec{\Omega}_e]$
	yuvw	$[\vec{V}_a\ \vec{V}_s\ \vec{V}_b\ \vec{V}_e]$
	ydl	$[\vec{\Omega}_{dl_a}\ \vec{\Omega}_{dl_s}\ \vec{\Omega}_{dl_b}\ \vec{\Omega}_{dl_e}\ \vec{V}_{dl_a}\ \vec{V}_{dl_s}\ \vec{V}_{dl_b}\ \vec{V}_{dl_e}]$
	FMtot_c	$[\vec{F}_{tot_a}\ \vec{M}_{tot_a}\ \vec{F}_{tot_s}\ \vec{M}_{tot_s}\ \vec{F}_{tot_b}\ \vec{M}_{tot_b}\ \vec{F}_{tot_e}\ \vec{M}_{tot_e}]$
	Ca_c	$[C_{\vec{F}_a}\ C_{\vec{M}_a}\ C_{\vec{F}_s}\ C_{\vec{M}_s}\ C_{\vec{F}_b}\ C_{\vec{M}_b}\ C_{\vec{F}_e}\ C_{\vec{M}_e}]$
	FMa_c	$[\vec{F}_a\ \vec{M}_a\ \vec{F}_s\ \vec{M}_s\ \vec{F}_b\ \vec{M}_b\ \vec{F}_e\ \vec{M}_e]$
blocks	State Derivatives, CFMa Correction	
variables	x0	

In this block the equations of motion are solved which results in the aircraft states and its derivatives. Moreover, the aerodynamic and total forces and moments and their coefficients are corrected for the $\dot{\alpha}$- and $\dot{\beta}$-contributions. The block consists of two sub-blocks, described below, and an integration block.

The aircraft states are calculated from integrating the aircraft state derivatives, derived in the equations of motion. The initial condition x_0 should be supplied by a trim-file `.tri`.

$$x_a = \int_0 \dot{x}_a\, dt \qquad , x_a(0) = x_0 \tag{D.17}$$

State Derivatives

inports	x	$[p_b\ q_b\ r_b\ V_{TAS}\ \alpha\ \beta\ \phi\ \theta\ \psi\ h_e\ x_e\ y_e]$
	FMtot	$[\vec{F}^*_{tot_a}\ \vec{M}^*_{tot_a}\ \vec{F}^*_{tot_s}\ \vec{M}^*_{tot_s}\ \vec{F}^*_{tot_b}\ \vec{M}^*_{tot_b}\ \vec{F}^*_{tot_e}\ \vec{M}^*_{tot_e}]$
	FMadot	$[\vec{F}_{\dot{\alpha}_a}\ \vec{M}_{\dot{\alpha}_a}\ \vec{F}_{\dot{\alpha}_s}\ \vec{M}_{\dot{\alpha}_s}\ \vec{F}_{\dot{\alpha}_b}\ \vec{M}_{\dot{\alpha}_b}\ \vec{F}_{\dot{\alpha}_e}\ \vec{M}_{\dot{\alpha}_e}]$
	FMbdot	$[\vec{F}_{\dot{\beta}_a}\ \vec{M}_{\dot{\beta}_a}\ \vec{F}_{\dot{\beta}_s}\ \vec{M}_{\dot{\beta}_s}\ \vec{F}_{\dot{\beta}_b}\ \vec{M}_{\dot{\beta}_b}\ \vec{F}_{\dot{\beta}_e}\ \vec{M}_{\dot{\beta}_e}]$
	massinit	$[m\ x_{cg}\ y_{cg}\ z_{cg}\ I_{xx}\ I_{yy}\ I_{zz}\ I_{xz}]$
	wind	$[\vec{V}_{w_a}\ \vec{V}_{w_s}\ \vec{V}_{w_b}\ \vec{V}_{w_e}]$
outports	xdot	$[\dot{p}_b\ \dot{q}_b\ \dot{r}_b\ \dot{V}_{TAS}\ \dot{\alpha}\ \dot{\beta}\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\ \dot{h}_e\ \dot{x}_e\ \dot{y}_e]$
	ypqr	$[\vec{\Omega}_a\ \vec{\Omega}_s\ \vec{\Omega}_b\ \vec{\Omega}_e]$
	yuvw	$[\vec{V}_a\ \vec{V}_s\ \vec{V}_b\ \vec{V}_e]$
	ydl	$[\vec{\Omega}_{dl_a}\ \vec{\Omega}_{dl_s}\ \vec{\Omega}_{dl_b}\ \vec{\Omega}_{dl_e}\ \vec{V}_{dl_a}\ \vec{V}_{dl_s}\ \vec{V}_{dl_b}\ \vec{V}_{dl_e}]$
	FMtot_c	$[\vec{F}_{tot_a}\ \vec{M}_{tot_a}\ \vec{F}_{tot_s}\ \vec{M}_{tot_s}\ \vec{F}_{tot_b}\ \vec{M}_{tot_b}\ \vec{F}_{tot_e}\ \vec{M}_{tot_e}]$
blocks	states transformation, mass properties, Vabdot, pqrdot, eulerdot, hxydot	

This block derives the aircraft state derivatives from the equations of motion. Before these equations can be formed, the velocity components in the states should be transformed to appropriate reference frames and the mass properties should be rewritten.

states transformation

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
outports	`ypqr`	$[\vec{\Omega}_a \;\; \vec{\Omega}_s \;\; \vec{\Omega}_b \;\; \vec{\Omega}_e]$
	`yuvw`	$[\vec{V}_a \;\; \vec{V}_s \;\; \vec{V}_b \;\; \vec{V}_e]$
	`ydl`	$[\vec{\Omega}_{dl_a} \;\; \vec{\Omega}_{dl_s} \;\; \vec{\Omega}_{dl_b} \;\; \vec{\Omega}_{dl_e} \;\; \vec{V}_{dl_a} \;\; \vec{V}_{dl_s} \;\; \vec{V}_{dl_b} \;\; \vec{V}_{dl_e}]$
S-functions	`ac_axes.mex*`	

This block calculates the linear and angular velocity components in the aircraft states with respect to the air-path, stability, body and moving earth reference frame. Moreover, the velocity components are expressed in dimensionless terms.

The angular velocity components in the aircraft states are with respect to the body reference frame. The transformation to all reference frames therefore becomes:

$$\begin{pmatrix} \vec{\Omega}_a \\ \vec{\Omega}_s \\ \vec{\Omega}_b \\ \vec{\Omega}_e \end{pmatrix} = \begin{bmatrix} R_{ab} \\ R_{sb} \\ I \\ R_{eb} \end{bmatrix} \begin{pmatrix} p_b \\ q_b \\ r_b \end{pmatrix} \tag{D.18}$$

The linear velocity components are included in the aircraft states via the true airspeed and the aerodynamic angles. The true airspeed is aligned with the X_a-axis of the air-path reference frame and the transformation to all reference frames is therefore:

$$\begin{pmatrix} \vec{V}_a \\ \vec{V}_s \\ \vec{V}_b \\ \vec{V}_e \end{pmatrix} = \begin{bmatrix} I \\ R_{sa} \\ R_{ba} \\ R_{ea} \end{bmatrix} \begin{pmatrix} V_{TAS} \\ 0 \\ 0 \end{pmatrix} \tag{D.19}$$

The dimensionless terms of the angular and linear velocities are derived in all reference frames. They take the following standard forms:

$$\vec{\Omega}_{dl} = \begin{pmatrix} \dfrac{pb}{2V_{TAS}} \\ \dfrac{q\bar{c}}{V_{TAS}} \\ \dfrac{rb}{2V_{TAS}} \end{pmatrix} \qquad \vec{V}_{dl} = \begin{pmatrix} \dfrac{u}{V_{TAS}} \\ \dfrac{v}{V_{TAS}} \\ \dfrac{w}{V_{TAS}} \end{pmatrix} \tag{D.20}$$

mass properties

inports	massinit	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
outports	mass	$[m]$
	inertia	$[c_1 \;\; \ldots \;\; c_9]$

This block extracts the mass from the mass properties vector and processes the moments and products of inertia for use in the moment equations.

The moment and products of inertia are rewritten to the following constants:

$$
\begin{aligned}
c_1 &= \frac{(I_{yy} - I_{zz})I_{zz} - I_{xz}^2}{I_{xx}I_{zz} - I_{xz}^2} & \qquad c_5 &= \frac{I_{zz} - I_{xx}}{I_{yy}} \\
c_2 &= \frac{(I_{xx} - I_{yy} + I_{zz})I_{xz}}{I_{xx}I_{zz} - I_{xz}^2} & \qquad c_6 &= \frac{I_{xz}}{I_{yy}} \\
c_3 &= \frac{I_{zz}}{I_{xx}I_{zz} - I_{xz}^2} & \qquad c_7 &= \frac{1}{I_{yy}} \\
c_4 &= \frac{I_{xz}}{I_{xx}I_{zz} - I_{xz}^2} & \qquad c_8 &= \frac{I_{xx}(I_{xx} - I_{yy}) + I_{xz}^2}{I_{xx}I_{zz} - I_{xz}^2} \\
& & \qquad c_9 &= \frac{I_{xx}}{I_{xx}I_{zz} - I_{xz}^2}
\end{aligned}
\tag{D.21}
$$

Vabdot

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	ypqr	$[\vec{\Omega}_a \;\; \vec{\Omega}_s \;\; \vec{\Omega}_b \;\; \vec{\Omega}_e]$
	Ftot	$[\vec{F}^*_{tot_a} \;\; \vec{F}^*_{tot_s} \;\; \vec{F}^*_{tot_b} \;\; \vec{F}^*_{tot_e}]$
	Fadot	$[\vec{F}_{\dot{\alpha}_a} \;\; \vec{F}_{\dot{\alpha}_s} \;\; \vec{F}_{\dot{\alpha}_b} \;\; \vec{F}_{\dot{\alpha}_e}]$
	Fbdot	$[\vec{F}_{\dot{\beta}_a} \;\; \vec{F}_{\dot{\beta}_s} \;\; \vec{F}_{\dot{\beta}_b} \;\; \vec{F}_{\dot{\beta}_e} \;]$
	mass	$[m]$
	wind	$[\vec{V}_{w_a} \;\; \vec{V}_{w_s} \;\; \vec{V}_{w_b} \;\; \vec{V}_{w_e}]$
outports	Vabdot	$[\dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta}]$
	Ftot_c	$[\vec{F}_{tot_a} \;\; \vec{F}_{tot_s} \;\; \vec{F}_{tot_b} \;\; \vec{F}_{tot_e}]$

This block calculates the derivatives of the linear velocity components in the aircraft state vector and includes the $\dot{\alpha}$- and $\dot{\beta}$-contributions in the total force. The derivatives follow from the force equations in the air-path reference frame, using an explicit formulation for the force equations along the air-path axes to include the $\dot{\alpha}$- and $\dot{\beta}$-contributions in the force:

$$
\begin{aligned}
\vec{F}_{tot_a} &= \vec{F}^*_{tot_a} + \vec{F}_{\dot{\alpha}_a}\dot{\alpha} + \vec{F}_{\dot{\beta}_a}\dot{\beta} \\
&= m\left\{ \frac{\partial}{\partial t}(\vec{V}_a + \vec{V}_{w_a}) + \vec{\Omega}_a \times (\vec{V}_a + \vec{V}_{w_a}) \right\}
\end{aligned}
\tag{D.22}
$$

The time derivative of the wind velocity is approximated from the wind velocity itself by evaluating its change in value since the previous integration step.
The derivatives $\dot{\alpha}$ and $\dot{\beta}$ are derived first. The force equations along the air-path Z_a- and Y_a-axes are made explicit in $\dot{\alpha}$ and $\dot{\beta}$ by collecting all $\dot{\alpha}$- and $\dot{\beta}$-terms, i.e. the contributions in the Z- and Y-forces, on the left-hand side. Applying a matrix inversion of a 2×2 matrix, $\dot{\alpha}$ and $\dot{\beta}$ then follow from:

$$\begin{pmatrix} \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{bmatrix} m(V_{TAS} + u_{w_a})\cos\beta - mv_{w_a}\sin\beta - Z_{\dot{\alpha}_a} & -Z_{\dot{\beta}_a} \\ mw_{w_a}\sin\beta - Y_{\dot{\alpha}_a} & m(V_{TAS} + u_{w_a}) - Y_{\dot{\beta}_a} \end{bmatrix}^{-1} \times$$

$$\begin{pmatrix} Z^*_{tot_a} - m\dot{w}_{w_a} - mp_a v_{w_a} + mq_a(V_{TAS} + u_{w_a}) \\ Y^*_{tot_a} - m\dot{v}_{w_a} + mp_a w_{w_a} - mr_a(V_{TAS} + u_{w_a}) \end{pmatrix} \tag{D.23}$$

The derivative $\dot{V}_{TAS}$ follows from the force equation along the air-path X_a-axis after correcting the X-force for the $\dot{\alpha}$- and $\dot{\beta}$-contributions with the above derived $\dot{\alpha}$ and $\dot{\beta}$ values:

$$\dot{V}_{TAS} = \frac{X^*_{tot_a} + X_{\dot{\alpha}_a}\dot{\alpha} + X_{\dot{\beta}_a}\dot{\beta}}{m} - \dot{u}_{w_a} - (q_a - \dot{\alpha}\cos\beta)w_{w_a} + (r_a - \dot{\beta})v_{w_a} \tag{D.24}$$

Finally, the total force is corrected for the $\dot{\alpha}$- and $\dot{\beta}$-contributions in all reference frames:

$$\vec{F}_{tot} = \vec{F}^*_{tot} + \vec{F}_{\dot{\alpha}}\dot{\alpha} + \vec{F}_{\dot{\beta}}\dot{\beta} \tag{D.25}$$

pqrdot

inports	`ypqr`	$[\vec{\Omega}_a \;\; \vec{\Omega}_s \;\; \vec{\Omega}_b \;\; \vec{\Omega}_e]$
	`Mtot`	$[\vec{M}^*_{tot_a} \;\; \vec{M}^*_{tot_s} \;\; \vec{M}^*_{tot_b} \;\; \vec{M}^*_{tot_e}]$
	`Madot`	$[\vec{M}_{\dot{\alpha}_a} \;\; \vec{M}_{\dot{\alpha}_s} \;\; \vec{M}_{\dot{\alpha}_b} \;\; \vec{M}_{\dot{\alpha}_e}]$
	`Mbdot`	$[\vec{M}_{\dot{\beta}_a} \;\; \vec{M}_{\dot{\beta}_s} \;\; \vec{M}_{\dot{\beta}_b} \;\; \vec{M}_{\dot{\beta}_e}\;]$
	`inertia`	$[c_1 \;\; \ldots \;\; c_9]$
	`Vabdot`	$[\dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta}]$
outports	`pqrdot`	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b]$
	`Mtot_c`	$[\vec{M}_{tot_a} \;\; \vec{M}_{tot_s} \;\; \vec{M}_{tot_b} \;\; \vec{M}_{tot_e}]$

This block calculates the derivatives of the angular velocity components in the aircraft state vector and includes the $\dot{\alpha}$- and $\dot{\beta}$-contributions in the total moment. The derivatives follow from the moment equations in the body reference frame, using the total moment which is corrected for the $\dot{\alpha}$- and $\dot{\beta}$-contributions.

The total moment is corrected first by including the $\dot{\alpha}$- and $\dot{\beta}$-contributions:

$$\vec{M}_{tot} = \vec{M}^*_{tot} + \vec{M}_{\dot{\alpha}}\dot{\alpha} + \vec{M}_{\dot{\beta}}\dot{\beta} \tag{D.26}$$

The derivatives $\dot{p}_b$, $\dot{q}_b$ and $\dot{r}_b$ follow from the moment equations in the body reference frame. Because the moments and products of inertia have been expressed as constants c_i, the derivatives only appear on the left-hand side:

$$\begin{aligned} \dot{p}_b &= (c_1 r_b + c_2 p_b) q_b + c_3 \bar{L}_{tot} + c_4 \bar{N}_{tot} \\ \dot{q}_b &= c_5 p_b r_b - c_6 (p_b^2 - r_b^2) + c_7 M_{tot} \\ \dot{r}_b &= (c_8 p_b - c_2 r_b) q_b + c_4 \bar{L}_{tot} + c_9 \bar{N}_{tot} \end{aligned} \tag{D.27}$$

eulerdot

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
outports	`eulerdot`	$[\dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi}]$

This block calculates the derivatives of the Euler angles in the aircraft state vector.

The derivatives $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ directly follow from the rotation of the body reference frame with respect to the moving earth reference frame. This kinematic relationship is given by:

$$\begin{aligned} \dot{\phi} &= p_b + \tan\theta \, (q_b \sin\phi + r_b \cos\phi) \\ \dot{\theta} &= q_b \cos\phi - r_b \sin\phi \\ \dot{\psi} &= \frac{q_b \sin\phi + r_b \cos\phi}{\cos\theta} \end{aligned} \tag{D.28}$$

hxydot

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`wind`	$[\vec{V}_{w_a} \;\; \vec{V}_{w_s} \;\; \vec{V}_{w_b} \;\; \vec{V}_{w_e}]$
outports	`hxydot`	$[\dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$

This block calculates the derivatives of the position vector relative to the fixed earth reference frame.

The derivatives $\dot{h}_e$, $\dot{x}_e$ and $\dot{y}_e$ can directly be obtained in the moving earth reference frame from the aircraft velocity relative to the surrounding air plus the wind velocity. The relative aircraft velocity is deduced from the true airspeed, while the wind velocity is supplied by the wind model. Hence, the navigation equations become:

$$\begin{aligned} \dot{h}_e &= -(w_e + w_{w_e}) \\ \dot{x}_e &= u_e + u_{w_e} \\ \dot{y}_e &= v_e + v_{w_e} \end{aligned} \tag{D.29}$$

CFMa Correction

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	xdot	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b \;\; \dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta} \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi} \;\; \dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$
	Ca	$[C_{\vec{F}_{base_a}} \;\; C_{\vec{M}_{base_a}} \;\; C_{\vec{F}_{base_s}} \;\; C_{\vec{M}_{base_s}} \;\; C_{\vec{F}_{base_b}} \;\; C_{\vec{M}_{base_b}} \;\; C_{\vec{F}_{base_e}} \;\; C_{\vec{M}_{base_e}}]$
	FMa	$[\vec{F}_{base_a} \;\; \vec{M}_{base_a} \;\; \vec{F}_{base_s} \;\; \vec{M}_{base_s} \;\; \vec{F}_{base_b} \;\; \vec{M}_{base_b} \;\; \vec{F}_{base_e} \;\; \vec{M}_{base_e}]$
	Cadot	$[C_{\vec{F}_{\dot{\alpha}_a}} \;\; C_{\vec{M}_{\dot{\alpha}_a}} \;\; C_{\vec{F}_{\dot{\alpha}_s}} \;\; C_{\vec{M}_{\dot{\alpha}_s}} \;\; C_{\vec{F}_{\dot{\alpha}_b}} \;\; C_{\vec{M}_{\dot{\alpha}_b}} \;\; C_{\vec{F}_{\dot{\alpha}_e}} \;\; C_{\vec{M}_{\dot{\alpha}_e}}]$
	FMadot	$[\vec{F}_{\dot{\alpha}_a} \;\; \vec{M}_{\dot{\alpha}_a} \;\; \vec{F}_{\dot{\alpha}_s} \;\; \vec{M}_{\dot{\alpha}_s} \;\; \vec{F}_{\dot{\alpha}_b} \;\; \vec{M}_{\dot{\alpha}_b} \;\; \vec{F}_{\dot{\alpha}_e} \;\; \vec{M}_{\dot{\alpha}_e}]$
	Cbdot	$[C_{\vec{F}_{\dot{\beta}_a}} \;\; C_{\vec{M}_{\dot{\beta}_a}} \;\; C_{\vec{F}_{\dot{\beta}_s}} \;\; C_{\vec{M}_{\dot{\beta}_s}} \;\; C_{\vec{F}_{\dot{\beta}_b}} \;\; C_{\vec{M}_{\dot{\beta}_b}} \;\; C_{\vec{F}_{\dot{\beta}_e}} \;\; C_{\vec{M}_{\dot{\beta}_e}}]$
	FMbdot	$[\vec{F}_{\dot{\beta}_a} \;\; \vec{M}_{\dot{\beta}_a} \;\; \vec{F}_{\dot{\beta}_s} \;\; \vec{M}_{\dot{\beta}_s} \;\; \vec{F}_{\dot{\beta}_b} \;\; \vec{M}_{\dot{\beta}_b} \;\; \vec{F}_{\dot{\beta}_e} \;\; \vec{M}_{\dot{\beta}_e}]$
outports	Ca_c	$[C_{\vec{F}_a} \;\; C_{\vec{M}_a} \;\; C_{\vec{F}_s} \;\; C_{\vec{M}_s} \;\; C_{\vec{F}_b} \;\; C_{\vec{M}_b} \;\; C_{\vec{F}_e} \;\; C_{\vec{M}_e}]$
	FMa_c	$[\vec{F}_a \;\; \vec{M}_a \;\; \vec{F}_s \;\; \vec{M}_s \;\; \vec{F}_b \;\; \vec{M}_b \;\; \vec{F}_e \;\; \vec{M}_e]$
variables	chord, span	

In this block the aerodynamic coefficients, forces and moments are corrected for the $\dot{\alpha}$- and $\dot{\beta}$-contributions.

The corrected aerodynamic coefficients follow by adding the terms with the linear dependencies on $\frac{\dot{\alpha}\bar{c}}{V}$ and $\frac{\dot{\beta}b}{V}$ to the baseline component. These additions apply to all reference frames, so:

$$\begin{pmatrix} C_{\vec{F}} \\ C_{\vec{M}} \end{pmatrix} = \begin{pmatrix} C_{\vec{F}_{base}} \\ C_{\vec{M}_{base}} \end{pmatrix} + \begin{pmatrix} C_{\vec{F}_{\dot{\alpha}}} \\ C_{\vec{M}_{\dot{\alpha}}} \end{pmatrix} \frac{\dot{\alpha}\bar{c}}{V_{TAS}} + \begin{pmatrix} C_{\vec{F}_{\dot{\beta}}} \\ C_{\vec{M}_{\dot{\beta}}} \end{pmatrix} \frac{\dot{\beta}b}{V_{TAS}} \tag{D.30}$$

The corrected aerodynamic forces and moments are similarly obtained from their baseline components and their linear dependencies on $\dot{\alpha}$ and $\dot{\beta}$. In all reference frames, the equations are:

$$\begin{pmatrix} \vec{F} \\ \vec{M} \end{pmatrix} = \begin{pmatrix} \vec{F}_{base} \\ \vec{M}_{base} \end{pmatrix} + \begin{pmatrix} \vec{F}_{\dot{\alpha}} \\ \vec{M}_{\dot{\alpha}} \end{pmatrix} \dot{\alpha} + \begin{pmatrix} \vec{F}_{\dot{\beta}} \\ \vec{M}_{\dot{\beta}} \end{pmatrix} \dot{\beta} \tag{D.31}$$

D.2.8 OBSERVATIONS

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	xdot	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b \;\; \dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta} \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi} \;\; \dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$
	yuvw	$[\vec{V}_a \;\; \vec{V}_s \;\; \vec{V}_b \;\; \vec{V}_e]$
	FMtot_c	$[\vec{F}_{tot_a} \;\; \vec{M}_{tot_a} \;\; \vec{F}_{tot_s} \;\; \vec{M}_{tot_s} \;\; \vec{F}_{tot_b} \;\; \vec{M}_{tot_b} \;\; \vec{F}_{tot_e} \;\; \vec{M}_{tot_e}]$
	FMa_c	$[\vec{F}_a \;\; \vec{M}_a \;\; \vec{F}_s \;\; \vec{M}_s \;\; \vec{F}_b \;\; \vec{M}_b \;\; \vec{F}_e \;\; \vec{M}_e]$
	Fgr	$[\vec{F}_{gr_a} \;\; \vec{F}_{gr_s} \;\; \vec{F}_{gr_b} \;\; \vec{F}_{gr_e}]$
	massinit	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
	yatm	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
outports	yacc	$[\vec{a}_b \;\; \vec{a}_{n_b} \;\; \vec{a}_{n_a} \;\; \vec{a}_{n,i_b} \;\; a_{n_b} \;\; a_{n,i_b} \;\; n]$
	yuvwdot	$[\dot{\vec{V}}_a \;\; \dot{\vec{V}}_s \;\; \dot{\vec{V}}_b \;\; \dot{\vec{V}}_e]$
	yfp	$[\gamma \;\; \chi \;\; \dot{\gamma} \;\; \dot{\chi} \;\; \gamma_T \;\; \chi_T \;\; \dot{\gamma}_T \;\; \dot{\chi}_T \;\; \chi_M \;\; \Psi_M \;\; \Phi \;\; \Gamma \;\; \lambda \;\; R \;\; V_G \;\; \ddot{h}_e \;\; fpa]$
	ys	$[E_s \;\; P_s]$
	yhab	$[\alpha_i \;\; \beta_i \;\; h_{e,i} \;\; \dot{h}_{e,i}]$
blocks	Acceleration, uvwdot, Flight-Path, Energy, abh measured	

In this block a wide range of observation parameters are calculated. The parameters are distinguished in accelerations, linear velocity time derivatives, flight-path related parameters and measurements outside the center of gravity. These sets are calculated in seperate sub-blocks described below.

Acceleration

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`xdot`	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b \;\; \dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta} \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi} \;\; \dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$
	`FMtot_c`	$[\vec{F}_{tot_a} \;\; \vec{M}_{tot_a} \;\; \vec{F}_{tot_s} \;\; \vec{M}_{tot_s} \;\; \vec{F}_{tot_b} \;\; \vec{M}_{tot_b} \;\; \vec{F}_{tot_e} \;\; \vec{M}_{tot_e}]$
	`FMa_c`	$[\vec{F}_a \;\; \vec{M}_a \;\; \vec{F}_s \;\; \vec{M}_s \;\; \vec{F}_b \;\; \vec{M}_b \;\; \vec{F}_e \;\; \vec{M}_e]$
	`Fgr`	$[\vec{F}_{gr_a} \;\; \vec{F}_{gr_s} \;\; \vec{F}_{gr_b} \;\; \vec{F}_{gr_e}]$
	`massinit`	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
outports	`yacc`	$[\vec{a}_b \;\; \vec{a}_{n_b} \;\; \vec{a}_{n_a} \;\; \vec{a}_{n,i_b} \;\; a_{n_b} \;\; a_{n,i_b} \;\; n]$
	`acc_e`	$[\vec{a}_e]$
variables	`xiacc, yiacc, ziacc, g0`	

This block calculates the aircraft accelerations. All accelerations are available in g-units, using the sea-level condition g_0.

The accelerations along the body axes are directly obtained from the body axes forces:

$$\vec{a}_b = \frac{\vec{F}_{tot_b}}{mg_0} \tag{D.32}$$

The outputs of the accelerometers, denoted by subscript $_n$, that are at the center of gravity follow from the total forces minus the components of the gravity force. Both along the body and air-path reference frame, this gives:

$$\vec{a}_n = \frac{\vec{F}_{tot} - \vec{F}_{gr}}{mg_0} \tag{D.33}$$

The outputs of the accelerometers that are not at the center of gravity, denoted by subscript $_{,i}$, are corrected in the body reference frame for the offset between the locations of the accelerometers and the center of gravity:

$$\begin{aligned}
\Delta a_{nx,i_b} &= -\frac{(q_b^2 + r_b^2)(x_{cg} - x_{i_{acc}}) - (p_b q_b - \dot{r}_b)(y_{cg} - y_{i_{acc}}) - (p_b r_b + \dot{q}_b)(z_{cg} - z_{i_{acc}})}{g_0} \\
\Delta a_{ny,i_b} &= \frac{(p_b q_b + \dot{r}_b)(x_{cg} - x_{i_{acc}}) - (p_b^2 + r_b^2)(y_{cg} - y_{i_{acc}}) + (q_b r_b - \dot{p}_b)(z_{cg} - z_{i_{acc}})}{g_0} \\
\Delta a_{nz,i_b} &= \frac{(p_b r_b - \dot{q}_b)(x_{cg} - x_{i_{acc}}) + (q_b r_b + \dot{p}_b)(y_{cg} - y_{i_{acc}}) - (p_b^2 + q_b^2)(z_{cg} - z_{i_{acc}})}{g_0}
\end{aligned} \tag{D.34}$$

The accelerometer outputs along the body axis and at any location therefore become:

$$\vec{a}_{n,i_b} = \vec{a}_{n_b} + \Delta\vec{a}_{n,i_b} \tag{D.35}$$

The accelerometer outputs along the negative body Z_b-axis are also made available:

$$\begin{aligned} a_{n_b} &= -a_{nz_b} \\ a_{n,i_b} &= -a_{nz,i_b} \end{aligned} \tag{D.36}$$

Finally, the load factor, i.e. total aerodynamic lift over vehicle weight, is included in the set of accelerations. Because the lift is by definition along the negative air-path Z_a-axis, this becomes:

$$n = \frac{-Z_a}{mg_0} \tag{D.37}$$

uvwdot

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`xdot`	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b \;\; \dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta} \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi} \;\; \dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$
	`acc_e`	$[\vec{a}_e]$
outports	`yuvwdot`	$[\dot{\vec{V}}_a \;\; \dot{\vec{V}}_s \;\; \dot{\vec{V}}_b \;\; \dot{\vec{V}}_e]$

This block calculates the time derivatives of the linear velocities along the axes of the air-path, stability, body and moving earth reference frames. These quantities follow from component-wise differentation.

The time derivatives of the linear velocities along the air-path, stability and body axes are calculated by transformation of the true airspeed from the air-path axes to the stability and body axes, followed by differentation with respect to time:

$$\dot{\vec{V}}_a = \frac{d}{dt}\begin{pmatrix} u_a \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \dot{V}_{TAS} \\ 0 \\ 0 \end{pmatrix} \tag{D.38}$$

$$\dot{\vec{V}}_s = \frac{d}{dt}\begin{pmatrix} u_a \cos\beta \\ u_a \sin\beta \\ 0 \end{pmatrix} = \begin{pmatrix} \dot{u}_a \cos\beta - u_a \sin\beta\, \dot{\beta} \\ \dot{u}_a \sin\beta + u_a \cos\beta\, \dot{\beta} \\ 0 \end{pmatrix} \tag{D.39}$$

$$\dot{\vec{V}}_b = \frac{d}{dt}\begin{pmatrix} u_s \cos\alpha \\ v_s \\ u_s \sin\alpha \end{pmatrix} = \begin{pmatrix} \dot{u}_s \cos\alpha - u_s \sin\alpha\, \dot{\alpha} \\ \dot{v}_s \\ \dot{u}_s \sin\alpha + u_s \cos\alpha\, \dot{\alpha} \end{pmatrix} \tag{D.40}$$

The time derivatives of the linear velocities along the moving earth axes is directly obtained from the acceleration vector because this reference frame does not rotate. Hence:

$$\dot{\vec{V}}_e = g_0\, \vec{a}_e \tag{D.41}$$

Flight-path

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	xdot	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b \;\; \dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta} \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi} \;\; \dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$
	yuvw	$[\vec{V}_a \;\; \vec{V}_s \;\; \vec{V}_b \;\; \vec{V}_e]$
	acc_e	$[\vec{a}_e]$
outports	yfp	$[\gamma \;\; \chi \;\; \dot{\gamma} \;\; \dot{\chi} \;\; \gamma_T \;\; \chi_T \;\; \dot{\gamma}_T \;\; \dot{\chi}_T \;\; \chi_M \;\; \Psi_M \;\; \Phi \;\; \Gamma \;\; \lambda \;\; R \;\; V_G \;\; \ddot{h}_e \;\; fpa]$

This block calculates various flight-path related parameters, including navigation parameters such as angles for an ILS-landing.

The orientation and motion of the air-path reference frame are both calculated relative to the surrounding air, i.e. the moving earth reference frame, and to the ground, i.e. the fixed earth reference frame. The equations for the first case yield:

$$\begin{aligned} \gamma &= \arcsin\frac{-w_e}{V_{TAS}} & \dot{\gamma} &= \frac{-V_{TAS}\,\dot{w}_e + \dot{V}_{TAS}\,w_e}{V_{TAS}\,\sqrt{V_{TAS}^2 - w_e^2}} \\ \chi &= \arctan\frac{v_e}{u_e} & \dot{\chi} &= \frac{u_e\,\dot{v}_e - \dot{u}_e\,v_e}{V_{TAS}^2 - w_e^2} \end{aligned} \tag{D.42}$$

In the second case, the above angles and angle rates are corrected for the presence of wind. They are denoted with the subscript $_T$ and calculated from:

$$\begin{aligned} \gamma_T &= \arcsin\frac{\dot{h}_e}{V_G} & \dot{\gamma}_T &= \frac{V_G\,\ddot{h}_e - \dot{V}_G\,\dot{h}_e}{V_G\,\sqrt{V_G^2 - \dot{h}_e^2}} = \frac{-V_G\,\dot{w}_e + \dot{V}_G\,w_e}{V_G\,\sqrt{V_G^2 - \dot{h}_e^2}} \\ \chi_T &= \arctan\frac{\dot{y}_e}{\dot{x}_e} & \dot{\chi}_T &= \frac{\dot{x}_e\,\ddot{y}_e - \ddot{x}_e\,\dot{y}_e}{V_G^2 - \dot{h}_e^2} = \frac{\dot{x}_e\,\dot{v}_e - \dot{u}_e\,\dot{y}_e}{V_G^2 - \dot{h}_e^2} \end{aligned} \tag{D.43}$$

where the ground speed is calculated from:

$$V_G = \left|\vec{V} + \vec{V}_w\right| = \sqrt{\dot{x}_e^2 + \dot{y}_e^2 + \dot{h}_e^2} \tag{D.44}$$

The magnetic navigation angles are not calculated for the present, but space is only reserved in the output vector. Their values are set to the constant `NaN`.
The bank angle, defined as the angle between the body Y_b-axis and the horizontal plane, is calculated from:

$$\Phi = \arcsin(\sin\phi\,\sin\theta) \tag{D.45}$$

The angles of the ILS landing system are made available as angular deviations above and to the east of the ILS track which is directed 3° upward from the ground along the centerline of the runway:

$$\begin{aligned} \epsilon_{GS} &= \arctan \frac{h_e - h_{GS}}{\sqrt{(x_e - x_{GS})^2 + (y_e - y_{GS})^2}} - 3\,\frac{\pi}{180} \\ \epsilon_{LOC} &= \psi_{LOC} - \arctan \frac{y_{LOC} - y_e}{x_{LOC} - x_e} \end{aligned} \tag{D.46}$$

The range is defined as the direct distance ('slant range') from the aircraft to the beacon:

$$R = \sqrt{(x_e - x_{DME})^2 + (y_e - y_{DME})^2 + (h_e - h_{DME})^2} \tag{D.47}$$

Finally, the accelerations along the vertical and along the flight-path are calculated in g-units:

$$\begin{aligned} \ddot{h}_e &= -a_{z_e} \\ fpa &= \frac{\dot{V}_{TAS}}{g_0} \end{aligned} \tag{D.48}$$

Energy

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`xdot`	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b \;\; \dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta} \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi} \;\; \dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$
	`yatm`	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
outports	`ys`	$[E_s \;\; P_s]$

This block calculates two energy related parameters, i.e. specific energy and specific power.

The specific energy and specific power are defined as:

$$\begin{aligned} E_s &= h_e + \frac{V_{TAS}^2}{2g} \\ P_s &= \dot{h}_e + \frac{V_{TAS}\dot{V}_{TAS}}{g} \end{aligned} \tag{D.49}$$

abh measurement

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`xdot`	$[\dot{p}_b \;\; \dot{q}_b \;\; \dot{r}_b \;\; \dot{V}_{TAS} \;\; \dot{\alpha} \;\; \dot{\beta} \;\; \dot{\phi} \;\; \dot{\theta} \;\; \dot{\psi} \;\; \dot{h}_e \;\; \dot{x}_e \;\; \dot{y}_e]$
	`massinit`	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
outports	`yhab`	$[\alpha_i \;\; \beta_i \;\; h_{e,i} \;\; \dot{h}_{e,i}]$
variables	`xiabh, yiabh, ziabh`	

This block calculates measurements from sensors not located at the center of gravity, i.e. the aerodynamic angles, altitude and altitude rate.

The measured aerodynamic angles have extra velocity terms due to the rotation of the body reference frame. If the vane axes of rotation are parallel to the body axes, the equations can be approximated by:

$$\begin{aligned}
\alpha_i &= \arctan\frac{w_{i_b}}{u_b} \approx \alpha - \frac{q_b(x_{cg} - x_{i_{\alpha\beta h}}) - p_b(y_{cg} - y_{i_{\alpha\beta h}})}{V_{TAS}} \\
\beta_i &= \arctan\frac{v_{i_b}}{u_b} \approx \beta + \frac{r_b(x_{cg} - x_{i_{\alpha\beta h}}) - p_b(z_{cg} - z_{i_{\alpha\beta h}})}{V_{TAS}}
\end{aligned} \tag{D.50}$$

The measured altitude and altitude rate have extra terms due to the attitude and rotation of the body reference frame. The following equations apply:

$$\begin{aligned}
h_{e,i} &= h_e + (x_{cg} - x_{i_{\alpha\beta h}})\sin\theta - (y_{cg} - y_{i_{\alpha\beta h}})\sin\phi\ \cos\theta - (z_{cg} - z_{i_{\alpha\beta h}})\cos\phi\ \cos\theta \\
\dot{h}_{e,i} &= \dot{h}_e + \dot{\theta}\left((x_{cg} - x_{i_{\alpha\beta h}})\cos\theta + (y_{cg} - y_{i_{\alpha\beta h}})\sin\phi\ \sin\theta + (z_{cg} - z_{i_{\alpha\beta h}})\cos\phi\ \sin\theta\right) \\
&\quad - \dot{\phi}\left((y_{cg} - y_{i_{\alpha\beta h}})\cos\phi\ \cos\theta - (z_{cg} - z_{i_{\alpha\beta h}})\sin\phi\ \cos\theta\right)
\end{aligned} \tag{D.51}$$

D.3 Standard Simulink sub-models

D.3.1 Atmosphere and airdata model ac_atmos.mex*

inports	x	$[p_b \ q_b \ r_b \ V_{TAS} \ \alpha \ \beta \ \phi \ \theta \ \psi \ h_e \ x_e \ y_e]$
outports	yatm	$[p_a \ \rho \ T \ g \ h_p \ h_R \ H \ V_{sound}]$
	yad1	$[M \ \bar{q}]$
	yad2	$[Re' \ q_c \ q_{rel} \ p_t \ T_t \ V_{EAS} \ V_{CAS} \ V_{IAS}]$
variables	g0, T0, rho0, p0, lambda, GASCON, gammah, RADIUS, Hs	

The atmosphere and airdata model calculates the atmosphere and airdata parameters. These quantities are calculated from the geometric altitude and air speed, using the equations and primary constants applying to the International Standard Atmosphere (ISA) [1].

The atmosphere parameters in yatm are based on the geopotential altitude which is derived from the altitude varying acceleration of gravity and the geometric altitude:

$$H = \frac{1}{g_0} \int_0^{h_e} g \, dh_e = \frac{R_e h_e}{R_e + h_e} \tag{D.52}$$

$$g = g_0 \frac{R_e^2}{(R_e + h_e)^2} \tag{D.53}$$

The ambient temperature, pressure and air density are now calculated from:

$$T = T_0 + \lambda H \tag{D.54}$$

$$p_a = \begin{cases} p_0 \left(\frac{T}{T_0} \right)^{-\frac{g_0}{R\lambda}} & , \quad H \leq 11000 \text{ m} \\ p_s \, \mathrm{e}^{-\frac{g_0}{RT_s}(H - H_s)} & , \quad 11000 \text{ m} < H \leq 20000 \text{ m} \end{cases} \tag{D.55}$$

$$\rho = \begin{cases} \rho_0 \left(\frac{T}{T_0} \right)^{-\left(\frac{g_0}{R\lambda}+1\right)} & , \quad H \leq 11000 \text{ m} \\ \rho_s \, \mathrm{e}^{-\frac{g_0}{RT_s}(H - H_s)} & , \quad 11000 \text{ m} < H \leq 20000 \text{ m} \end{cases} \tag{D.56}$$

where T_s, p_s and ρ_s at the tropopause are calculated from the first equations with $H_s = 11000$ m.
The pressure and radio altitude are defined equal to the geometric altitude:

$$h_p = h_e \tag{D.57}$$

$$h_R = h_e - h_G = h_e \tag{D.58}$$

The speed of sound is calculated from the temperature by:

$$V_{sound} = \sqrt{\gamma RT} \tag{D.59}$$

The primary airdata parameters in `yad1`, Mach number and dynamic pressure, are calculated from the air speed:

$$M = \frac{V_{TAS}}{V_{sound}} \tag{D.60}$$

$$\bar{q} = \frac{1}{2}\rho V_{TAS}^2 \tag{D.61}$$

The additional airdata parameters in `yad2` consist of the Reynolds number and quantities used for the airdata instruments. The Reynols number per unit length is defined as:

$$\begin{aligned} Re' &= \frac{\rho V_{TAS}}{\mu} \\ \mu &= \frac{\beta T^{3/2}}{T+S} = \frac{1.458\ 10^{-6}\ T^{3/2}}{T+110.4} \end{aligned} \tag{D.62}$$

The total pressure is calculated from:

$$p_t = p_a \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}} \tag{D.63}$$

The impact pressure and relative impact pressure are:

$$q_c = p_t - p_a = p_a \left(\left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}} - 1\right) \tag{D.64}$$

$$q_{rel} = \frac{q_c}{p_a} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}} - 1 \tag{D.65}$$

The total temperature follows from:

$$T_t = T \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}} \tag{D.66}$$

The equivalent, calibrated and indicated airspeed are calculated from:

$$V_{EAS} = \sqrt{\frac{2\bar{q}}{\rho_0}} \tag{D.67}$$

$$V_{CAS} = \sqrt{\frac{2\gamma}{\gamma - 1}\frac{p_0}{\rho_0}\left(\left(1 + \frac{q_c}{p_0}\right)^{\frac{\gamma-1}{\gamma}} - 1\right)} \tag{D.68}$$

$$V_{IAS} = V_{CAS} \tag{D.69}$$

D.3.2 Transformation model `ac_axes.mex*`

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$ $[\vec{r}_m]$
outports		$[\vec{r}_a \;\; \vec{r}_s \;\; \vec{r}_b \;\; \vec{r}_e]$
parameters	`axis`	

The transformation model transforms a vector in any reference frame to all applied reference frames, i.e. air-path, stability, body and moving earth reference frame. This transformation is applied at various places in the generic aircraft model, for instance at the aerodynamic coefficients and the velocity components.

The relationships between the reference frames are described by rotational transformations. The equations therefore consist of matrix multiplications with orthogonal rotation matrices. The sequence of required transformations depends on the reference frame which applies to the inported vector $\vec{r}_m$ and which is designated by the parameter `axis`. The values `axis` = [0/1/2/3], refer to the [air-path/stability/body/earth]-axes.

The transformation from the moving earth to the body reference frame is described by a sequence of three plane rotations, i.e. about the positive earth Z_e-axis over the yaw angle ψ, followed by a rotation about the positive new Y-axis over the pitch angle θ and finally about the positive new X-axis over the roll angle ϕ:

$$\vec{r}_b = R_{be}\,\vec{r}_e$$
$$\vec{r}_e = R_{eb}\,\vec{r}_b$$

$$R_{be} = R_{eb}^T = \begin{bmatrix} \cos\theta\,\cos\psi & \cos\theta\,\sin\psi & -\sin\theta \\ \sin\phi\,\sin\theta\,\cos\psi - \cos\phi\,\sin\psi & \sin\phi\,\sin\theta\,\sin\psi + \cos\phi\,\cos\psi & \sin\phi\,\cos\theta \\ \cos\phi\,\sin\theta\,\cos\psi + \sin\phi\,\sin\psi & \cos\phi\,\sin\theta\,\sin\psi - \sin\phi\,\cos\psi & \cos\phi\,\cos\theta \end{bmatrix} \tag{D.70}$$

The tranformation of the body reference frame to the stability reference frame is described by a rotation about the negative body Y_a-axis over the angle of attack:

$$\vec{r}_s = R_{sb}\,\vec{r}_b$$
$$\vec{r}_b = R_{bs}\,\vec{r}_s$$
$$R_{sb} = R_{bs}^T = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{D.71}$$

The tranformation of the stability reference frame to the air-path reference frame is described by a rotation about the positive stability Z_s-axis over the angle of sideslip:

$$\vec{r}_a = R_{as}\,\vec{r}_s$$
$$\vec{r}_s = R_{sa}\,\vec{r}_a$$
$$R_{as} = R_{sa}^T = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{D.72}$$

D.3.3 Wind model (wnd_none.m)

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
outports	wind	$[\vec{V}_{w_e}]$

The wind model supplies the wind velocities in the fixed earth reference frame. The *DASMAT* package only includes the wind model for zero wind conditions, given by the S-function `wnd_none.m`.
If the user wants to model a non-zero wind condition, the presented model may be used as a template for the *Inport/Outport* blocks. The signals to the *Outport* blocks may then be connected with the block of the aircraft states, using
The equations for zero wind conditions are:

$$\vec{V}_{w_e} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{D.73}$$

For non-zero wind conditions, the wind velocity may be modelled as any function of the aircraft states and/or variables in the MATLAB workspace.

D.3.4 Turbulence models (tur_none.m), (tur_dryd.m)

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
outports	gust	$[\hat{u}_g \;\; \alpha_g \;\; \beta_g \;\; \dot{\hat{u}}_g\bar{c}/V \;\; \dot{\alpha}_g\bar{c}/V \;\; \dot{\beta}_g b/V \;\; \hat{u}_{gasym} \;\; \alpha_{gasym}]$

The turbulence model supplies the gust velocities and gust velocity rates in the body reference frame. These quantities are derived from white noise and taken positive along the negative body axes. The *DASMAT* package includes a turbulence model for no turbulence (`tur_none.m`) and for turbulence derived from Dryden spectra (`tur_dryd.m`).

The equations for no turbulence in the S-function `tur_none.m` are given as:

$$\begin{aligned} \hat{u}_g &= 0 & \frac{\dot{\hat{u}}_g \bar{c}}{V_{TAS}} &= 0 & \hat{u}_{gasym} &= 0 \\ \alpha_g &= 0 & \frac{\dot{\alpha}_g \bar{c}}{V_{TAS}} &= 0 & \alpha_{gasym} &= 0 \\ \beta_g &= 0 & \frac{\dot{\beta}_g b}{V_{TAS}} &= 0 \end{aligned} \tag{D.74}$$

The equations for turbulence using Dryden spectra in the S-function `tur_dryd.m` are derived from the transfer functions given by (2.38) and (2.39). These transfer functions are converted to the state space form because the polynomial coefficients depend on the air speed which is not a constant variables. The conversion results in the following cannonical form:

$$H(s) = K\,\frac{b_0}{s+a_0} \quad \Longleftrightarrow \quad \begin{cases} \dot{x}_1 &= -a_0\, x_1 + w \\ y &= K\, b_0\, x_1 \end{cases}$$

$$H(s) = K\,\frac{b_1 s + b_0}{s^2 + a_1 s + a_0} \quad \Longleftrightarrow \quad \begin{cases} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} &= \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w \\ y &= K \begin{bmatrix} b_0 & b_1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \end{cases} \tag{D.75}$$

The gust velocities and gust velocity rates for the symmetrical forces and moments are therefore given by:

$$\begin{aligned} \dot{x}_{u_g} &= -\frac{V}{L_g}\, x_1 + w_1 \\ \hat{u}_g &= \frac{\sigma_g}{V} \sqrt{\frac{2V}{L_g}}\, x_{u_g} \\ \frac{\dot{\hat{u}}_g \bar{c}}{V_{TAS}} &= \frac{\sigma_g \bar{c}}{V^2} \sqrt{\frac{2V}{L_g}}\, \dot{x}_{u_g} \end{aligned} \tag{D.76}$$

$$
\begin{aligned}
\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}_{\alpha_g} &= \begin{bmatrix} 0 & 1 \\ -\dfrac{V^2}{L_g^2} & -2\dfrac{V}{L_g} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\alpha_g} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_2 \\
\alpha_g &= \frac{\sigma_g}{V}\sqrt{\frac{V}{L_g}} \begin{bmatrix} \dfrac{V}{L_g} & \sqrt{3} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\alpha_g} \\
\frac{\dot{\alpha}_g \bar{c}}{V_{TAS}} &= \frac{\sigma_g \bar{c}}{V^2}\sqrt{\frac{V}{L_g}} \begin{bmatrix} \dfrac{V}{L_g} & \sqrt{3} \end{bmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}_{\alpha_g}
\end{aligned}
\tag{D.77}
$$

For the asymmetrical forces and moments, the gust velocities and velocity rates are given by:

$$
\begin{aligned}
\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}_{\beta_g} &= \begin{bmatrix} 0 & 1 \\ -\dfrac{V^2}{L_g^2} & -2\dfrac{V}{L_g} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\beta_g} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_3 \\
\beta_g &= \frac{\sigma_g}{V}\sqrt{\frac{V}{L_g}} \begin{bmatrix} \dfrac{V}{L_g} & \sqrt{3} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\beta_g} \\
\frac{\dot{\beta}_g b}{V_{TAS}} &= \frac{\sigma_g \bar{c}}{V^2}\sqrt{\frac{V}{L_g}} \begin{bmatrix} \dfrac{V}{L_g} & \sqrt{3} \end{bmatrix} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}_{\beta_g}
\end{aligned}
\tag{D.78}
$$

$$
\begin{aligned}
\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}_{\hat{u}_{g_{asym}}} &= \begin{bmatrix} 0 & 1 \\ \dfrac{1}{\tau_1\tau_2}\dfrac{V^2}{L_g^2} & \dfrac{\tau_1+\tau_2}{\tau_1\tau_2}\dfrac{V}{L_g} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\hat{u}_{g_{asym}}} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_4 \\
\hat{u}_{g_{asym}} &= \frac{\sigma_g}{V}\frac{1}{\tau_1\tau_2}\sqrt{I_{u_g}(0,B)\frac{V}{L_g}} \begin{bmatrix} \dfrac{V}{L_g} & \tau_3 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\hat{u}_{g_{asym}}}
\end{aligned}
\tag{D.79}
$$

$$
\begin{aligned}
\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}_{\alpha_{g_{asym}}} &= \begin{bmatrix} 0 & 1 \\ \dfrac{1}{\tau_1\tau_2}\dfrac{V^2}{L_g^2} & \dfrac{\tau_1+\tau_2}{\tau_1\tau_2}\dfrac{V}{L_g} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\alpha_{g_{asym}}} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_5 \\
\alpha_{g_{asym}} &= \frac{\sigma_g}{V}\frac{1}{\tau_1\tau_2}\sqrt{I_{u_g}(0,B)\frac{V}{L_g}} \begin{bmatrix} \dfrac{V}{L_g} & \tau_3 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{\alpha_{g_{asym}}}
\end{aligned}
\tag{D.80}
$$

In the above equations, $w_1 \ldots w_5$ are independent white noise signals with zero mean and intensity one.

D.3.5 Feed-through engine model eng_none.m

inports	x	$[p_b \ q_b \ r_b \ V_{TAS} \ \alpha \ \beta \ \phi \ \theta \ \psi \ h_e \ x_e \ y_e]$
	ut	$[T_N]$
	yatm	$[p_a \ \rho \ T \ g \ h_p \ h_R \ H \ V_{sound}]$
	yad1	$[M \ \bar{q}]$
	yad2	$[Re' \ q_c \ q_{rel} \ p_t \ T_t \ V_{EAS} \ V_{CAS} \ V_{IAS}]$
	Deng	$[D_{eng}]$
outports	ypow	$[T_N \ FF \ \ldots]$

The feed-through engine model is applied when thrust is taken as aircraft control input. The thrust controls are then directly ported to the propulsion model.

The outputs of the feed-through engine model only have rational values for the first two elements, being thrust and fuel flow. The thrust is directly taken from the control input and diminished with engine drag for modeling a potential nonworking engine operation. The fuel flow is set to zero. Hence:

$$\begin{aligned} T_N &= u_t - D_{eng} \\ FF &= 0 \end{aligned} \tag{D.81}$$

Additional outputs are provided for compatibility of the model with the standardized input/ouput format of engine models. These outputs are irrelevant and have therefore Not-a-Number constants.

D.4 User-supplied Simulink models

D.4.1 Aircraft specific aerodynamic model `ac_aeromodel`

inports	`x`	$[p_b \ q_b \ r_b \ V_{TAS} \ \alpha \ \beta \ \phi \ \theta \ \psi \ h_e \ x_e \ y_e]$
	`u`	$[\delta_e \ \delta_a \ \delta_r \ \delta_{t_e} \ \delta_{t_a} \ \delta_{t_r} \ \delta_f \ \ell g_{sw}]$
	`cg`	$[x_{cg} \ y_{cg} \ z_{cg}]$
	`yad1`	$[M \ \bar{q}]$
	`yad2`	$[Re' \ q_c \ q_{rel} \ p_t \ T_t \ V_{EAS} \ V_{CAS} \ V_{IAS}]$
	`gust`	$[\hat{u}_g \ \alpha_g \ \beta_g \ \dot{\hat{u}}_g\bar{c}/V \ \dot{\alpha}_g\bar{c}/V \ \dot{\beta}_g b/V \ \hat{u}_{gasym} \ \alpha_{gasym}]$
outports	`Ca`	$[C_X \ C_Y \ C_Z \ C_\ell \ C_m \ C_n]$
	`Cadot`	$[C_{X_{\dot{\alpha}}} \ C_{Y_{\dot{\alpha}}} \ C_{Z_{\dot{\alpha}}} \ C_{\ell_{\dot{\alpha}}} \ C_{m_{\dot{\alpha}}} \ C_{n_{\dot{\alpha}}}]$
	`Cbdot`	$[C_{X_{\dot{\beta}}} \ C_{Y_{\dot{\beta}}} \ C_{Z_{\dot{\beta}}} \ C_{\ell_{\dot{\beta}}} \ C_{m_{\dot{\beta}}} \ C_{n_{\dot{\beta}}}]$
	`Cag`	$[C_{X_g} \ C_{Y_g} \ C_{Z_g} \ C_{\ell_g} \ C_{m_g} \ C_{n_g}]$
variables	user supplied	

This model calculates the aerodynamic force and moment coefficients and the force and the force and moment coefficients due to turbulence. The coefficients may be supplied in any reference frame. The moment coefficients are with respect to a reference location specified by the variables $(x, y, z)_{cg_{ref}}$ = `[xcgref,ycgref,zcgref]`.

The equations for the aerodynamic coefficients depend on the aerodynamic model which is derived for a specific aircraft. They generally incorporate results from look-up tables describing the aerodynamic coefficients as functions of the flight condition, for example airspeed, aerodynamic angles, aerodynamic control deflections, etc. The coefficients due to turbulence may be modelled as extra aerodynamic force and moment coefficients, or they may be modelled as corrections to the velocity terms in the equations of motion.

The equations for the aerodynamic coefficients are very specific to the aircraft and will therefore not further discussed. The equations for the coefficients due to turbulence are more standard and will therefore described.

If the coefficients due to turbulence are modelled as extra aerodynamic force and moment coefficients, then the velocity used in the equations of motion differs from the actual air speed encountered by the aircraft and which should therefore be used in calculating the aerodynamic forces and moments. The force and moment differences caused by using the actual air speed and the aircraft velocity in still air are now the forces and moments due to turbulence. Their symmetric and asymmetric coefficients are modelled according to the following equations:

$$
\begin{aligned}
C_{X_g} &= C_{X_{u_g}}\hat{u}_g + C_{X_{\dot{u}_g}}\frac{\dot{\hat{u}}_g\bar{c}}{V_{TAS}} + C_{X_{\alpha_g}}\alpha_g + C_{X_{\dot{\alpha}_g}}\frac{\dot{\alpha}_g\bar{c}}{V_{TAS}} \\
C_{Z_g} &= C_{Z_{u_g}}\hat{u}_g + C_{Z_{\dot{u}_g}}\frac{\dot{\hat{u}}_g\bar{c}}{V_{TAS}} + C_{Z_{\alpha_g}}\alpha_g + C_{Z_{\dot{\alpha}_g}}\frac{\dot{\alpha}_g\bar{c}}{V_{TAS}} \\
C_{m_g} &= C_{m_{u_g}}\hat{u}_g + C_{m_{\dot{u}_g}}\frac{\dot{\hat{u}}_g\bar{c}}{V_{TAS}} + C_{m_{\alpha_g}}\alpha_g + C_{m_{\dot{\alpha}_g}}\frac{\dot{\alpha}_g\bar{c}}{V_{TAS}}
\end{aligned}
\tag{D.82}
$$

$$
\begin{aligned}
C_{Y_g} &= C_{Y_{\beta_g}}\beta_g + C_{Y_{\dot{\beta}_g}}\frac{\dot{\beta}_g b}{V_{TAS}} + C_{Y_{u_g}}\hat{u}_{gasym} + C_{Y_{\alpha_g}}\alpha_{gasym} \\
C_{\ell_g} &= C_{\ell_{\beta_g}}\beta_g + C_{\ell_{\dot{\beta}_g}}\frac{\dot{\beta}_g b}{V_{TAS}} + C_{\ell_{u_g}}\hat{u}_{gasym} + C_{\ell_{\alpha_g}}\alpha_{gasym} \\
C_{n_g} &= C_{n_{\beta_g}}\beta_g + C_{n_{\dot{\beta}_g}}\frac{\dot{\beta}_g b}{V_{TAS}} + C_{n_{u_g}}\hat{u}_{gasnm} + C_{n_{\alpha_g}}\alpha_{gasnm}
\end{aligned}
\tag{D.83}
$$

The gust derivatives in these equations are now expressed in terms of the aerodynamic derivatives which are extracted from the aerodynamic model in still air. In [12], approximations of the gust derivatives are given when the equations are applied in the body reference frame. The symmetric gust derivatives are then approximated with:

$$
\begin{aligned}
C_{X_{u_{g_b}}} &= C_{X_{u_b}} \approx 2C_{X_b} = \ 2\left(C_L\ \sin\alpha - C_D\ \cos\alpha\right) \\
C_{Z_{u_{g_b}}} &= C_{Z_{u_b}} \approx 2C_{Z_b} = -2\left(C_L\ \cos\alpha + C_D\ \sin\alpha\right) \\
C_{m_{u_{g_b}}} &= C_{m_{u_b}} \approx 0 \\
\\
C_{X_{\dot{u}_{g_b}}} &= 0 \\
C_{Z_{\dot{u}_{g_b}}} &= 2\left(C_{Z_w}\,\frac{x_{cg}-x_w}{\bar{c}} + C_{Z_h}\,\frac{x_{cg}-x_h}{\bar{c}}\right) \approx 0 \\
C_{m_{\dot{u}_{g_b}}} &= 2\left(C_{m_w}\,\frac{x_{cg}-x_w}{\bar{c}} + C_{m_h}\,\frac{x_{cg}-x_h}{\bar{c}}\right) \approx -2C_{m_h}\,\frac{\ell_h}{\bar{c}} \\
\\
C_{X_{\alpha_{g_b}}} &= C_{X_{\alpha_b}} \approx C_L - C_{D_\alpha} \\
C_{Z_{\alpha_{g_b}}} &= C_{Z_{\alpha_b}} \approx -C_{L_\alpha} \\
C_{m_{\alpha_{g_b}}} &= C_{m_{\alpha_b}} \\
\\
C_{X_{\dot{\alpha}_{g_b}}} &= 0 \\
C_{Z_{\dot{\alpha}_{g_b}}} &= C_{Z_{\dot{\alpha}}} - C_{Z_q} \approx 0 \\
C_{m_{\dot{\alpha}_{g_b}}} &= C_{m_{\dot{\alpha}}} - C_{m_q} \approx -C_{m_q}
\end{aligned}
\tag{D.84}
$$

and the asymmetric gust derivatives are calculated according:

$$
\begin{aligned}
C_{Y_{u_{g_b}}} &= 0 \\
C_{\ell_{u_{g_b}}} &= -C_{\ell_{r_w}} \approx -C_{\ell_{r_b}} \\
C_{n_{u_{g_b}}} &= -C_{n_{r_w}} \approx -C_{n_{r_b}} \\
\\
C_{Y_{\alpha_{g_b}}} &= 0 \\
C_{\ell_{\alpha_{g_b}}} &= C_{\ell_{p_w}} \approx C_{\ell_{p_b}} \\
C_{n_{\alpha_{g_b}}} &= C_{n_{p_w}} \approx C_{n_{p_b}} \\
\\
C_{Y_{\beta_{g_b}}} &= C_{Y_{\beta_b}} \\
C_{\ell_{\beta_{g_b}}} &= C_{\ell_{\beta_b}} \\
C_{n_{\beta_{g_b}}} &= C_{n_{\beta_b}} \\
\\
C_{Y_{\dot{\beta}_{g_b}}} &= C_{Y_{\dot{\beta}_b}} + \tfrac{1}{2} C_{Y_r} \\
C_{\ell_{\dot{\beta}_{g_b}}} &= C_{\ell_{\dot{\beta}_b}} + \tfrac{1}{2} C_{\ell_r} \\
C_{n_{\dot{\beta}_{g_b}}} &= C_{n_{\dot{\beta}_b}} + \tfrac{1}{2} C_{n_r}
\end{aligned}
\tag{D.85}
$$

If the coefficients due to turbulence are modelled as corrections to the air velocity, then the velocity terms in the equations of motion should be corrected to inertial velocity. These corrections may be modelled as extra forces according to:

$$
\vec{F}_g = -m\left(\dot{\vec{V}}_g + \vec{\Omega} \times \vec{V}_g\right) \tag{D.86}
$$

The extra forces are now converted to non-dimensionless coefficients which will then be expressed through the gust derivatives. Because the gust velocities are taken positive along the negative body axes, the following expressions result:

$$
\begin{aligned}
C_{X_{g_b}} &= 2\mu_c \left(\frac{\dot{\hat{u}}_g \bar{c}}{V_{TAS}} + \alpha_g \frac{q_b \bar{c}}{V_{TAS}} - \beta_g \frac{r_b \bar{c}}{V_{TAS}} \right) \\
C_{Y_{g_b}} &= 2\mu_b \left(\frac{\dot{\beta}_g b}{V_{TAS}} + \hat{u}_g \frac{r_b b}{V_{TAS}} - \alpha_g \frac{p_b b}{V_{TAS}} \right) \\
C_{Z_{g_b}} &= 2\mu_c \left(\frac{\dot{\alpha}_g \bar{c}}{V_{TAS}} + \beta_g \frac{p_b \bar{c}}{V_{TAS}} - \hat{u}_g \frac{q_b \bar{c}}{V_{TAS}} \right)
\end{aligned}
\tag{D.87}
$$

Although this turbulence modeling does not generate a moment, there will be moment with respect to the reference location for the aerodynamic model. The moment coefficients in the body reference frame are given by:

$$
\begin{aligned}
C_{\ell_{g_b}} &= \frac{-C_{Y_b}(z_{cg_{ref}} - z_{cg}) + C_{Z_b}(y_{cg_{ref}} - y_{cg})}{b} \\
C_{m_{g_b}} &= \frac{C_{X_b}(z_{cg_{ref}} - z_{cg}) - C_{Z_b}(x_{cg_{ref}} - x_{cg})}{\bar{c}} \\
C_{n_{g_b}} &= \frac{-C_{X_b}(y_{cg_{ref}} - y_{cg}) + C_{Y_b}(x_{cg_{ref}} - x_{cg})}{b}
\end{aligned}
\tag{D.88}
$$

The gust derivatives thus become:

$$
\begin{aligned}
C_{X_{u_{g_b}}} &= 0 & C_{\ell_{u_{g_b}}} &= -2\mu_b \frac{r_b(z_{cg_{ref}} - z_{cg}) + q_b(y_{cg_{ref}} - y_{cg})}{V_{TAS}} \\
C_{Y_{u_{g_b}}} &= 2\mu_b \frac{r_b b}{V_{TAS}} & C_{m_{u_{g_b}}} &= 2\mu_c \frac{q_b(x_{cg_{ref}} - x_{cg})}{V_{TAS}} \\
C_{Z_{u_{g_b}}} &= -2\mu_c \frac{q_b \bar{c}}{V_{TAS}} & C_{n_{u_{g_b}}} &= 2\mu_b \frac{r_b(x_{cg_{ref}} - x_{cg})}{V_{TAS}} \\
\\
C_{X_{\alpha_{g_b}}} &= 2\mu_c \frac{q_b \bar{c}}{V_{TAS}} & C_{\ell_{\alpha_{g_b}}} &= 2\mu_b \frac{p_b(z_{cg_{ref}} - z_{cg})}{V_{TAS}} \\
C_{Y_{\alpha_{g_b}}} &= -2\mu_b \frac{p_b b}{V_{TAS}} & C_{m_{\alpha_{g_b}}} &= 2\mu_c \frac{q_b(z_{cg_{ref}} - z_{cg})}{V_{TAS}} \\
C_{Z_{\alpha_{g_b}}} &= 0 & C_{n_{\alpha_{g_b}}} &= -2\mu_b \frac{q_b(y_{cg_{ref}} - y_{cg}) + p_b(x_{cg_{ref}} - x_{cg})}{V_{TAS}} \\
\\
C_{X_{\beta_{g_b}}} &= -2\mu_c \frac{r_b \bar{c}}{V_{TAS}} & C_{\ell_{\beta_{g_b}}} &= 2\mu_b \frac{p_b(y_{cg_{ref}} - y_{cg})}{V_{TAS}} \\
C_{Y_{\beta_{g_b}}} &= 0 & C_{m_{\beta_{g_b}}} &= -2\mu_c \frac{r_b(z_{cg_{ref}} - z_{cg}) + p_b(x_{cg_{ref}} - x_{cg})}{V_{TAS}} \\
C_{Z_{\beta_{g_b}}} &= 2\mu_c \frac{p_b \bar{c}}{V_{TAS}} & C_{n_{\beta_{g_b}}} &= 2\mu_b \frac{r_b(y_{cg_{ref}} - y_{cg})}{V_{TAS}} \\
\\
C_{X_{\dot{u}_{g_b}}} &= 2\mu_c & C_{\ell_{\dot{u}_{g_b}}} &= 0 \\
C_{Y_{\dot{u}_{g_b}}} &= 0 & C_{m_{\dot{u}_{g_b}}} &= 2\mu_c \frac{z_{cg_{ref}} - z_{cg}}{\bar{c}} \\
C_{Z_{\dot{u}_{g_b}}} &= 0 & C_{n_{\dot{u}_{g_b}}} &= -2\mu_c \frac{y_{cg_{ref}} - y_{cg}}{b} \\
\\
C_{X_{\dot{\alpha}_{g_b}}} &= 0 & C_{\ell_{\dot{\alpha}_{g_b}}} &= 2\mu_c \frac{y_{cg_{ref}} - y_{cg}}{b} \\
C_{Y_{\dot{\alpha}_{g_b}}} &= 0 & C_{m_{\dot{\alpha}_{g_b}}} &= -2\mu_c \frac{x_{cg_{ref}} - x_{cg}}{\bar{c}} \\
C_{Z_{\dot{\alpha}_{g_b}}} &= 2\mu_c & C_{n_{\dot{\alpha}_{g_b}}} &= 0 \\
\\
C_{X_{\dot{\beta}_{g_b}}} &= 0 & C_{\ell_{\dot{\beta}_{g_b}}} &= -2\mu_b \frac{z_{cg_{ref}} - z_{cg}}{b} \\
C_{Y_{\dot{\beta}_{g_b}}} &= 2\mu_b & C_{m_{\dot{\beta}_{g_b}}} &= 0 \\
C_{Z_{\dot{\beta}_{g_b}}} &= 0 & C_{n_{\dot{\beta}_{g_b}}} &= 2\mu_b \frac{x_{cg_{ref}} - x_{cg}}{b}
\end{aligned}
\tag{D.89}
$$

D.4.2 Aircraft specific propulsion model `ac_powermodel`

inports	`x`	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	`ut`	$\begin{cases} [PLA_1 \;\; PLA_2] & \text{(engine model included)} \\ [T_{N_1} \;\; T_{N_2}] & \text{(engine model not included)} \end{cases}$
	`massinit`	$[m \;\; x_{cg} \;\; y_{cg} \;\; z_{cg} \;\; I_{xx} \;\; I_{yy} \;\; I_{zz} \;\; I_{xz}]$
	`yatm`	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
	`yad1`	$[M \;\; \bar{q}]$
	`yad2`	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
outports	`FMt`	$[\vec{F}_{t_b} \;\; \vec{M}_{t_b}]$
	`ypow`	$[T_{N_1} \;\; FF_1 \;\; \ldots T_{N_2} \;\; FF_2 \;\; \ldots]$
S-functions	`eng_none.m`	
variables	`Deng, xeng, yeng, zeng, toweng, tilteng`	

This model calculates the total force and moment in the body reference frame of all engines and ports a number of engine parameters to the MATLAB workspace. The moments are with respect to the origin of the body reference frame. The engine data are supplied by engine models which are included via *S-function* blocks, see also Section D.3.5 *Feed-through engine model* ***`eng_none.m`*** and Section D.4.3 *Aircraft specific engine model* ***`(eng_statmodel), (eng_dynmodel)`***.
An example propulsion model is provided in the S-function `ac_pow.m` for a two-engine aircraft.

The total force and moment of the propulsion system are calculated from the thrusts of all engines. First, the thrusts of all engines are transformed to the body axes via the tow and tilt angles and summed according to the following equations:

$$\begin{aligned} X_{t_b} &= \sum_{i=1} \frac{1}{\sqrt{1+\tan^2 t_{i_i}+\tan^2 t_{o_i}}} T_{N_i} \\ Y_{t_b} &= \sum_{i=1} \frac{\tan t_{o_i}}{\sqrt{1+\tan^2 t_{i_i}+\tan^2 t_{o_i}}} T_{N_i} \\ Z_{t_b} &= \sum_{i=1} \frac{\tan t_{i_i}}{\sqrt{1+\tan^2 t_{i_i}+\tan^2 t_{o_i}}} T_{N_i} \end{aligned} \tag{D.90}$$

Next, the moment with respect to the origin of the body reference frame is calculated from the thrust components along the body axes of each engine:

$$\begin{aligned} \bar{L}_{t_b} &= \sum_{i=1} Y_{t_{b_i}} z_{eng_i} - Z_{t_{b_i}} y_{eng_i} \\ M_{t_b} &= \sum_{i=1} -X_{t_{b_i}} z_{eng_i} + Z_{t_{b_i}} x_{eng_i} \\ \bar{N}_{t_b} &= \sum_{i=1} X_{t_{b_i}} y_{eng_i} - Y_{t_{b_i}} x_{eng_i} \end{aligned} \tag{D.91}$$

Moreover, the outputs from all engine models are concatenated in one signal y_{pow} and written to the MATLAB workspace as `ypow` via a *To Workspace* block.

D.4.3 Aircraft specific engine model (eng_statmodel), (eng_dynmodel)

inports	x	$[p_b \;\; q_b \;\; r_b \;\; V_{TAS} \;\; \alpha \;\; \beta \;\; \phi \;\; \theta \;\; \psi \;\; h_e \;\; x_e \;\; y_e]$
	ut	$[PLA]$
	yatm	$[p_a \;\; \rho \;\; T \;\; g \;\; h_p \;\; h_R \;\; H \;\; V_{sound}]$
	yad1	$[M \;\; \bar{q}]$
	yad2	$[Re' \;\; q_c \;\; q_{rel} \;\; p_t \;\; T_t \;\; V_{EAS} \;\; V_{CAS} \;\; V_{IAS}]$
	Deng	$[D_{eng}]$
outports	ypow	$[T_N \;\; FF \;\; \ldots]$
variables	user supplied	

This model calculates the thrust and fuel flow of the engine and any additional engine parameters.

The equations for the engine model are very specific and will therefore not further discussed. However, *some remarks should be notified for all engine models.*
The first two elements of the outputs should be thrust and fuel flow respectively.
The outputs of the static engine model should at least include the engine states used in the dynamic engine model. The trim routine uses the static engine model for calculating the engine operating conditions to obtain a required thrust. The output components indexed by the variable `xtndx` will be regarded as the trimmed engine states of the dynamic model.
The total number of outputs should be at least 8 to allow the model included in the generic engine model `eng_mod.m`. The contents of the elements three and above in the output signal may be user-defined since they are not used by the propulsion and aircraft model.
The effect of a nonworking engine may be modelled via the input D_{eng}, where a zero signal corresponds to normal engine operation. This input may be used as engine drag and subtracted from the generated engine thrust to account for the loss of thrust. The input may also be used as switch to distinguish from a working and a nonworking operating condition within the model.

Series 01: Aerodynamics

01. F. Motallebi, 'Prediction of Mean Flow Data for Adiabatic 2-D Compressible Turbulent Boundary Layers'
1997 / VI + 90 pages / ISBN 90-407-1564-5

02. P.E. Skåre, 'Flow Measurements for an Afterbody in a Vertical Wind Tunnel'
1997 / XIV + 98 pages / ISBN 90-407-1565-3

03. B.W. van Oudheusden, 'Investigation of Large-Amplitude 1-DOF Rotational Galloping'
1998 / IV + 100 pages / ISBN 90-407-1566-1

04. E.M. Houtman / W.J. Bannink / B.H. Timmerman, 'Experimental and Computational Study of a Blunt Cylinder-Flare Model in High Supersonic Flow'
1998 / VIII + 40 pages / ISBN 90-407-1567-X

05. G.J.D. Zondervan, 'A Review of Propeller Modelling Techniques Based on Euler Methods'
1998 / IV + 84 pages / ISBN 90-407-1568-8

06. M.J. Tummers / D.M. Passchier, 'Spectral Analysis of Individual Realization LDA Data'
1998 / VIII + 36 pages / ISBN 90-407-1569-6

07. P.J.J. Moeleker, 'Linear Temporal Stability Analysis'
1998 / VI + 74 pages / ISBN 90-407-1570-X

08. B.W. van Oudheusden, 'Galloping Behaviour of an Aeroelastic Oscillator with Two Degrees of Freedom'
1998 / IV + 128 pages / ISBN 90-407-1571-8

09. R. Mayer, 'Orientation on Quantitative IR-thermografy in Wall-shear Stress Measurements'
1998 / XII + 108 pages / ISBN 90-407-1572-6

10. K.J.A. Westin / R.A.W.M. Henkes, 'Prediction of Bypass Transition with Differential Reynolds Stress Models'
1998 / VI + 78 pages / ISBN 90-407-1573-4

11. J.L.M. Nijholt, 'Design of a Michelson Interferometer for Quantitative Refraction Index Profile Measurements'
1998 / 60 pages / ISBN 90-407-1574-2

12. R.A.W.M. Henkes / J.L. van Ingen, 'Overview of Stability and Transition in External Aerodynamics'
1998 / IV + 48 pages / ISBN 90-407-1575-0

13. R.A.W.M. Henkes, 'Overview of Turbulence Models for External Aerodynamics'
1998 / IV + 40 pages / ISBN 90-407-1576-9

Series 02: Flight Mechanics

01. E. Obert, 'A Method for the Determination of the Effect of Propeller Slipstream on a Static Longitudinal Stability and Control of Multi-engined Aircraft'
    1997 / IV + 276 pages / ISBN 90-407-1577-7
02. C. Bill / F. van Dalen / A. Rothwell, 'Aircraft Design and Analysis System (ADAS)'
    1997 / X + 222 pages / ISBN 90-407-1578-5
03. E. Torenbeek, 'Optimum Cruise Performance of Subsonic Transport Aircraft'
    1998 / X + 66 pages / ISBN 90-407-1579-3

Series 03: Control and Simulation

01. J.C. Gibson, 'The Definition, Understanding and Design of Aircraft Handling Qualities'
    1997 / X + 162 pages / ISBN 90-407-1580-7
02. E.A. Lomonova, 'A System Look at Electromechanical Actuation for Primary Flight Control'
    1997 / XIV + 110 pages / ISBN 90-407-1581-5
03. C.A.A.M. van der Linden, 'DASMAT-Delft University Aircraft Simulation Model and Analysis Tool. A Matlab/Simulink Environment for Flight Dynamics and Control Analysis'
    1998 / XII + 220 pages / ISBN 90-407-1582-3

Series 05: Aerospace Structures and Computional Mechanics

01. A.J. van Eekelen, 'Review and Selection of Methods for Structural Reliability Analysis'
    1997 / XIV + 50 pages / ISBN 90-407-1583-1
02. M.E. Heerschap, 'User's Manual for the Computer Program Cufus. Quick Design Procedure for a CUt-out in a FUSelage version 1.0'
    1997 / VIII + 144 pages / ISBN 90-407-1584-X
03. C. Wohlever, 'A Preliminary Evaluation of the B2000 Nonlinear Shell Element Q8N.SM'
    1998 / IV + 44 pages / ISBN 90-407-1585-8
04. L. Gunawan, 'Imperfections Measurements of a Perfect Shell with Specially Designed Equipment (UNIVIMP)
    1998 / VIII + 52 pages / ISBN 90-407-1586-6

Series 07: Aerospace Materials

01. A. Vašek / J. Schijve, 'Residual Strenght of Cracked 7075 T6 Al-alloy Sheets under High Loading Rates'
1997 / VI + 70 pages / ISBN 90-407-1587-4
02. I. Kunes, 'FEM Modelling of Elastoplastic Stress and Strain Field in Centre-cracked Plate'
1997 / IV + 32 pages / ISBN 90-407-1588-2
03. K. Verolme, 'The Initial Buckling Behavior of Flat and Curved Fiber Metal Laminate Panels'
1998 / VIII + 60 pages / ISBN 90-407-1589-0
04. P.W.C. Provó Kluit, 'A New Method of Impregnating PEI Sheets for the *In-Situ* Foaming of Sandwiches'
1998 / IV + 28 pages / ISBN 90-407-1590-4
05. A. Vlot / T. Soerjanto / I. Yeri / J.A. Schelling, 'Residual Thermal Stresses around Bonded Fibre Metal Laminate Repair Patches on an Aircraft Fuselage'
1998 / IV + 24 pages / ISBN 90-407-1591-2
06. A. Vlot, 'High Strain Rate Tests on Fibre Metal Laminates'
1998 / IV + 44 pages / ISBN 90-407-1592-0
07. S. Fawaz, 'Application of the Virtual Crack Closure Technique to Calculate Stress Intensity Factors for Through Cracks with an Oblique Elliptical Crack Front'
1998 / VIII + 56 pages / ISBN 90-407-1593-9
08. J. Schijve, 'Fatigue Specimens for Sheet and Plate Material'
1998 / VI + 18 pages / ISBN 90-407-1594-7

Series 08: Astrodynamics and Satellite Systems

01. E. Mooij, 'The Motion of a Vehicle in a Planetary Atmosphere'
1997 / XVI + 156 pages / ISBN 90-407-1595-5
02. G.A. Bartels, 'GPS-Antenna Phase Center Measurements Performed in an Anechoic Chamber'
1997 / X + 70 pages / ISBN 90-407-1596-3
03. E. Mooij, 'Linear Quadratic Regulator Design for an Unpowered, Winged Re-entry Vehicle'
1998 / X + 154 pages / ISBN 90-407-1597-1

Computer Assisted Design (CAD) environments have become important devices for the design and evaluation of flight control systems. For general use, different aircraft and operational conditions should be easily implemented in such a CAD environment and it should be equipped with a set of simulation and analysis tools. To extend its functionalities, it is best integrated in a high-performance computing environment with an extensive library of control design routines. This report documents the CAD environment DASMAT, which stands for Delft University Aircraft Simulation Model and Analysis Tool. It operates in the computing environment MATLAB/SIMULINK, having highperformance numeric computation and visualization functionalities. The essential element in the DASMAT package is a generic nonlinear simulation model conceived with well-defined and generalized interfaces. For linear flight control design, the package contains special tools for trimming and linearizing the aircraft at user-defined operating points. A finished design may be evaluated by visualizing the time behaviour in nonlinear simulations. Both on-line and off-line analysis functions are available with the possibility of 3D flight-path and attitude visualization trough animation. For analyzing flight test data, the package also includes identification routines of which results are easily implemented in the simulation model. After a short introduction of DASMAT, this book focuses on the models, signals and variables present in the DASMAT package. The operational aspects for the simulation and analysis tools are discussed next, followed by the application of the DASMAT package for control design purposes. The appendices include a complete list of files of the DASMAT package, the complete lists of signal formats and all essential variables used in the generic models, and the lay-out plus equations of the SIMULINK models.

ISBN 90-407-1582-3

DELFT AEROSPACE
ENGINEERING & TECHNOLOGY