Moorman D. and Looye G.:
The Modelica Flight Dynamics Library
2nd International Modelica Conference, Proceedings, pp.275-284

Paper presented at the 2nd International Modelica Conference, March 18-19, 2002,
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Oberpfaffenhofen, Germany.

All papers of this workshop can be downloaded from
http://www.Modelica.org/Conference2002/papers.shtml

The Modelica Flight Dynamics Library

D. Moormann and G. Looye
German Aerospace Center (DLR)
Institute of Robotics and Mechatronics
Oberpfaffenhofen, D-82234 Wessling, Germany
phone: +49 8153 28 1068 / fax: +49 8153 28 1441
E-Mail: Dieter.Moormann@dlr.de, Gertjan.Looye@dlr.de

Abstract

The Modelica Flight Dynamics Library has been developed to model 6-degrees-of-freedom, nonlinear flight dynamics and flight systems. Using this library the multidisciplinary interaction between flight dynamics and systems can easily be understood and analyzed. In this contribution the main benefits of the Flight Dynamics Library, concerning model building and efficient code generation – in particular for nonlinear parametric simulations and trim computations – are discussed. The library has been successfully applied to the development of aircraft models for several flight control system design projects.

1 Introduction

The design of aircraft requires contributions from different disciplines that are usually represented by different specialized groups within the aircraft development process. In design and evaluation of controlled flight system dynamics this is obvious.
In particular, the basic **flight dynamics** model consists of a description of aircraft geometry and mass together with equations of motion and of environmental influences such as gravity, atmosphere, and wind/gust. Basic flight dynamics are affected by aerodynamics and propulsion, two other distinct disciplines involved. The flight dynamics interact with the onboard **systems**, which can be grouped into motivators, sensors, and controls. Note that motivators consist of control surfaces such as elevators, and actuators which drive them.
Optimizing the interaction between flight dynamics and systems is an important area of investigation to improve efficiency of operation. For example, control surfaces can be designed to be 'just-right' in size and dynamic performance in order to minimize mass and drag of the aircraft, while still guaranteeing the required overall aircraft flight characteristics in case of failures.

Traditional aircraft models are built using domain specific software packages that best solve their specific task with respect to the different disciplines involved, e.g., flight mechanics, propulsion, controls and hydraulics. As a drawback, those packages usually have very limited capabilities with respect to other domains and thereby it is quite cumbersome to link the different model components together. Hence, to develop a comprehensive aircraft dynamics model with low engineering effort, it is necessary to apply a model description form that is well suited for all domains involved and meets the requirements for multidisciplinary aircraft model integration. This description form has to be equally expressive for flight dynamics and for systems, which includes mechanical, electrical, hydraulical, and discrete digital control elements.

For this purpose we propose an object-oriented modeling approach, developed as a general tool for a wide variety of systems described by differential and algebraic equations. The advantage of such an approach is that it is easy to understand and that it can be used to visualize the hierarchical decomposition of a complete system. For each discipline, reusable domain specific model libraries can be built to encapsulate pertinent knowledge. The ability to work with submodels of different granularity is helpful for the design of flight control systems, where it is necessary to work with system models **and** flight dynamics models concurrently. During the design iteration process it should be possible to adjust both the refinement of the system model and the complexity of the flight dynamics model.

An important feature of object diagrams is that they are not limited to block diagrams with signal-directed input/output behavior. In an object diagram, for ex-

ample, the constituents of flight dynamics can be connected naturally according to their physical energy flow interaction and it is not required to transform all objects into a mathematical block diagram form as it has to be done for block oriented control modeling environments such as MATLAB-SIMULINK.

This paper describes how nonlinear aircraft dynamics models can be composed using the MODELICA-Flight Dynamics Library. Its main benefits concerning model composition using object-oriented structuring principles are presented in section 2. Its benefits resulting from an efficient mathematical code generation are discussed in section 3, where special emphasis is put on code generation for efficient parametric simulation and on highly accurate and efficient trim procedures.

2 Interactive multi-point model composition via hierarchical object-diagrams

An aircraft consists of a variety of different systems, which represent the interacting disciplines involved in aircraft engineering (e.g. flight mechanics, aerodynamics, engine dynamics).

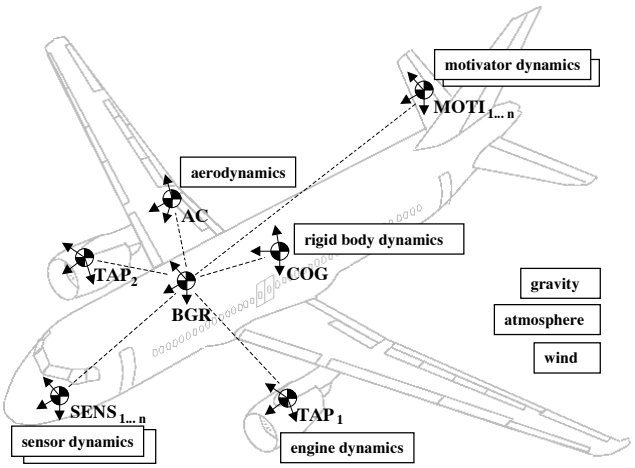

Figure 1: Domain specific reference points of flight dynamics and system models

Models of aircraft dynamics should be described in a notation close to the aircraft physics. The most natural way of modeling physical systems is as physical objects and phenomena, which are connected according to their physical energy flow interaction and kinematic constraints. This is different from modeling via signal flows or input-output block diagrams as traditionally used for controller modeling.

The 'local' description of each aircraft component (see Fig. 1) with respect to its intrinsic reference points (e.g., Center of Gravity COG, Body Geometric Reference BGR, Thrust Application Point TAP, Aerodynamic Center AC) in its domain specific coordinate system supports 'multi-point' modeling. The multi-point modeling approach allows, e.g., the proper handling of center of gravity variations and sensor positioning without any additional modeling effort, which is usually a very time-consuming and error-prone process.

A multi-point model also becomes necessary, when, e.g., the coupling effects between 'aircraft' and 'air flow' need to be modeled with higher accuracy than can be obtained by using a ordinary one-point model, where all the force, moment and velocity vectors are referred to the aircraft's center of gravity only [2].

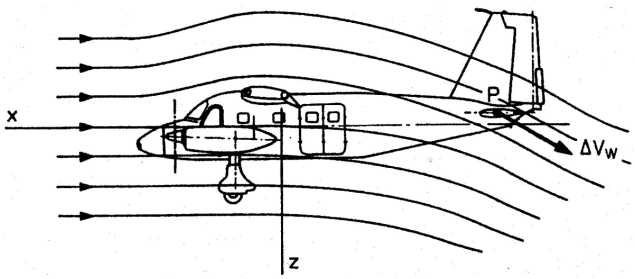

Figure 2: Local differential air velocity due to wing downwash and engine flow [2]

It is obvious from Fig. 2, that the local airflow is different for different points at the airframe due to aircraft rotation, changing wind fields, wing downwash and engine flow. The complete local airspeed $\vec{V}_a$ for each point P can be calculated from the local inertial speed $\vec{V}(\mathrm{P})$ and the local speed of the airflow $\vec{V}_w(\mathrm{P})$:

$$\vec{V}_a(\mathrm{P}) = \vec{V}(\mathrm{P}) - \vec{V}_w(\mathrm{P}) \,.$$

Above equation can be expanded to identify all partial velocity vectors to give the complete airspeed at the point considered:

$$\vec{V}_a(\mathrm{P}) = \vec{V}(\mathrm{COG}) - \vec{V}_w(\mathrm{COG}) + \Delta\vec{V}(\mathrm{P},\vec{\omega}) - \Delta\vec{V}_w(\mathrm{P}) - \Delta\vec{V}_{w_{\mathrm{dw}}}(\mathrm{P}) \,.$$

The total local airspeed is summed up by the inertial velocity of the center of gravity $\vec{V}(\mathrm{COG})$, the speed of wind at COG $\vec{V}_w(\mathrm{COG})$, the additional airspeed due to aircraft rotation ω at P about COG $\Delta\vec{V}(\mathrm{P})$, the effect of wind gradients due to its offset from COG $\Delta\vec{V}_w(\mathrm{P})$ and the effect of wing downwash and engine flow at P $\Delta\vec{V}_{w_{\mathrm{dw}}}(\mathrm{P})$.

Using a 2-point-aerodynamics approach it is quite convenient to properly model the influence of aircraft rotation, wing downwash and wind gradients. According

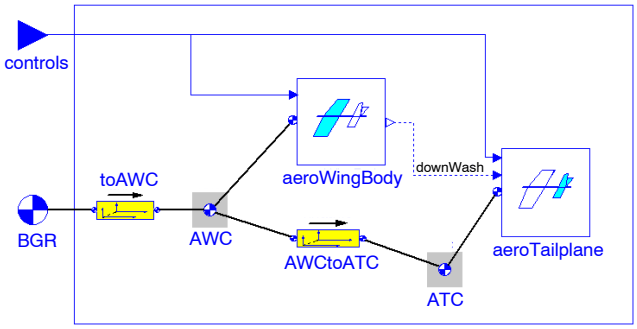

Figure 3: 2-point-aerodynamics object diagram with respect to the aircraft geometric reference BGR

to the object model of Fig. 3 this is done by separately describing the aerodynamics of wing/body (with respect to the Aerodynamic Wing Center AWC) and of the tailplane (with respect to the Aerodynamic Tail Center ATC). Between the aircraft body geometric reference BGR and these two aerodynamic reference points there are geometric offsets, which are explicitly made visible by the instances 'toAWC' and 'AWCtoATC' of a validated coordinate transformation class. The advantage of this approach is that the influence of aircraft rotation $\Delta\vec{V}(P)$ and the effect of wind gradients $\Delta\vec{V}_w(P)$ of above equation are automatically correctly handled by generic transformation objects.

In the same way, using the Flight Dynamics Library, all interactions between components of Fig. 1 can be properly formulated. In order to make the understanding of all submodels easy, each component of the library is described in its own coordinate system. Gravity, wind, and atmosphere are conveniently described in an earth related coordinate system, aerodynamics in a wind coordinate system, and engines in a system which is related to the body-fixed coordinate system. Therefore, in addition to the basic aircraft components, coordinate transformations are also detailed and handled as objects in the Flight Dynamics Library (see upper part of Fig. 4). Except for aerodynamics and engine objects all other objects are independent of a specific aircraft type.

The objects that constitute the rigid-body flight dynamics are interconnected according the object diagram of the bottom part of Fig. 4. Center point of the flight dynamics object model are the body geometric reference BGR and the center of gravity COG together with the body-object, which describes the mass properties and equations of motion of an aircraft. The

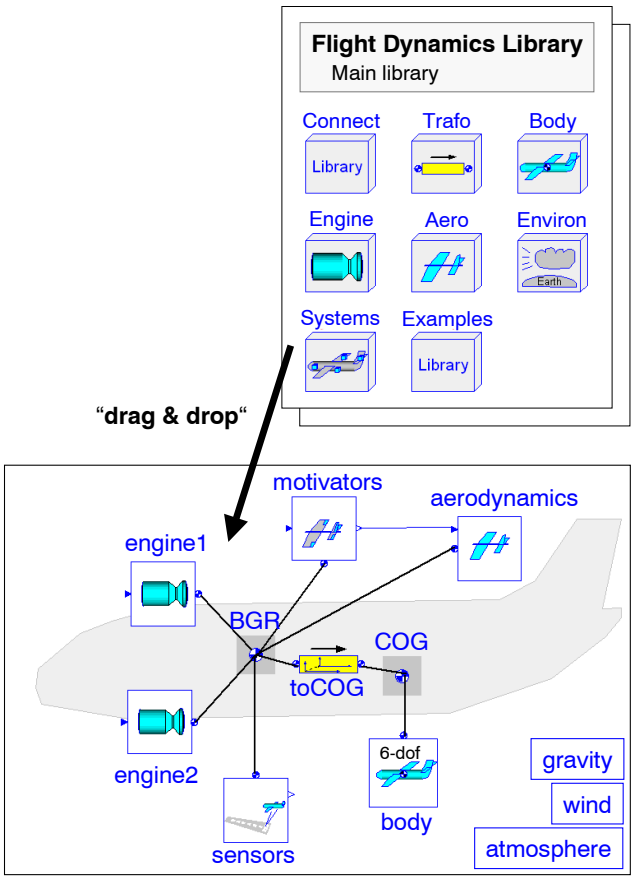

Figure 4: Interactive "drag & drop"-model building, top: flight dynamics class library, bottom: flight dynamics object-diagram

connections between objects represent their interaction. The complete aircraft consists of a body (fuselage and wing), which is powered by one or more engines. The aerodynamics describes the effects of the airflow over the aircraft. The aircraft is influenced by gravity and the surrounding atmosphere and winds. Additional dynamics is resulting from models of motivators and of sensors.

The connectors used to describe the interaction between flight dynamics objects, as specified by bold solid lines in the object diagram of Fig. 4, are the same as those used within the MODELICA-Multibody Library[1]. The connector contains all variables which specifies the orientation, position and the corresponding speeds and accelerations with respect to some inertia. For aircraft usually some point at the earth's surface together with a 'north-east-down' coordinate system is defined as inertial reference. Additional connector variables are the force and moment vector, acting

[1] URL: http://www.modelica.org/library/ModelicaAdditions/docu/ModelicaAdditions_MultiBody.html

at the origin of the point defined by the connector and solved in the coordinate system of the connector.

Specific for aircraft are the models of gravity, atmosphere and wind/gust. For multi-point models it is essential to properly formulate these models as fields, which usually vary with inertial position. For this purpose MODELICA offers the concept of 'dynamic scoping' [8]. Using this concept gravity fields, wind/gust fields and atmospheric data depending on the inertial position of individual aircraft components can be specified. Without any user effort the 2-point-aerodynamics of Fig. 3 is automatically handled correctly, because e.g. the wind field (and its gradients) are inherited position dependent to the aerodynamics models of wing/body and of the tailplane.

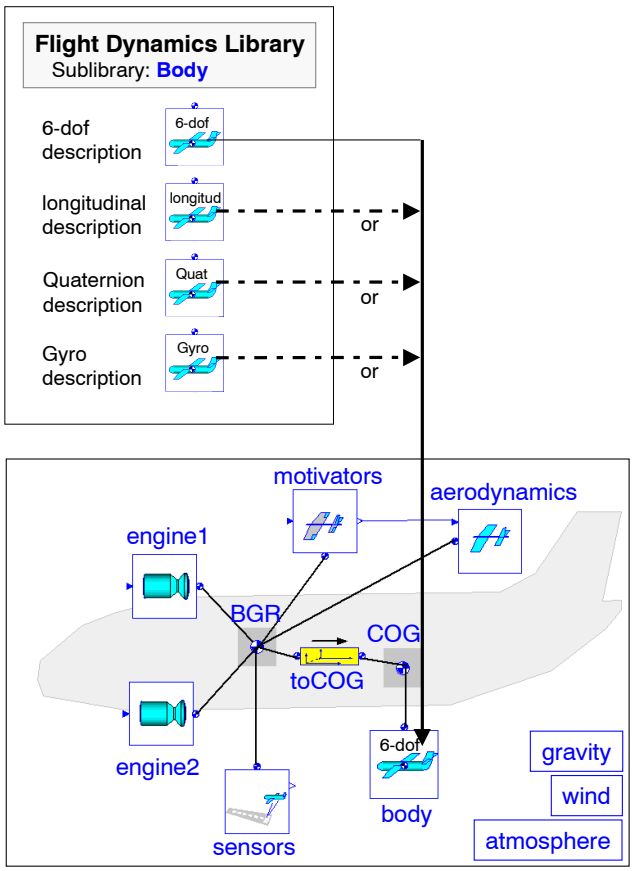

Figure 5: Local exchange of sub-components

In the Flight Dynamics Library different representations of one specific component can be found (see Fig. 5). There is a class with six degrees of freedom body, and another class with three degrees of freedom bodyLong that can be used to generate a nonlinear simulation model for the longitudinal motion only. The ordinary formulation of the equations of motion can be exchanged by a representation using quaternions. There are also wind, atmosphere and gravity models of different complexity. Existing codes for aerodynamics and engine models can easily be reused by using templates.

An important aspect of object-oriented modeling is the hierarchical structure and local encapsulation of objects. With the graphical user interface of DYMOLA, it can be zoomed into objects to display their internal structure. Zooming into the engine-object in the lower left part of Fig. 6 results in the engine model as detailed in the top left of this figure. It shows that the engine dynamics are described with respect to the thrust application point TAP. This is the most natural point at the airframe to formulate propulsion effects. The thrust application point is connected to the aircraft reference BGR via a transformation object toTAP, which details the offset in position and orientation from TAP with respect to BGR. Depending on this offset the generic transformation object is instantiated with the particular parameter values (top right of Fig. 6). As a result, the local engine forces and moments with respect to BGR are automatically computed.

Zooming into the body-object in the lower left part of Fig. 6 shows the equation layer of this component as displayed in the lower right part of this figure. Objects, which form the physical model, contain declarative mathematical equations, not assignments as is common in simulation languages. This makes the understanding and engineering reuse much easier as opposed to simulation code put in a form mainly for computational execution. A generic object with declarative equations can fulfill different application tasks. For example, the object toCOG which does the transformations between COG and BGR is used for transforming velocities with respect to COG to velocities with respect to BGR, as required as an interim step for aerodynamics and thrust calculations. The same object is used for the transformation of forces and moments from the BGR reference to the COG-reference, as required for solving the equations of motion within the body-object.

When connecting objects, only the relation between them is defined, and not the order in which the object equations are finally solved. Here the computer is used to sort the object equations automatically by a symbolic equation handler rather than performing this process manually. This aspect is handled in the following section.

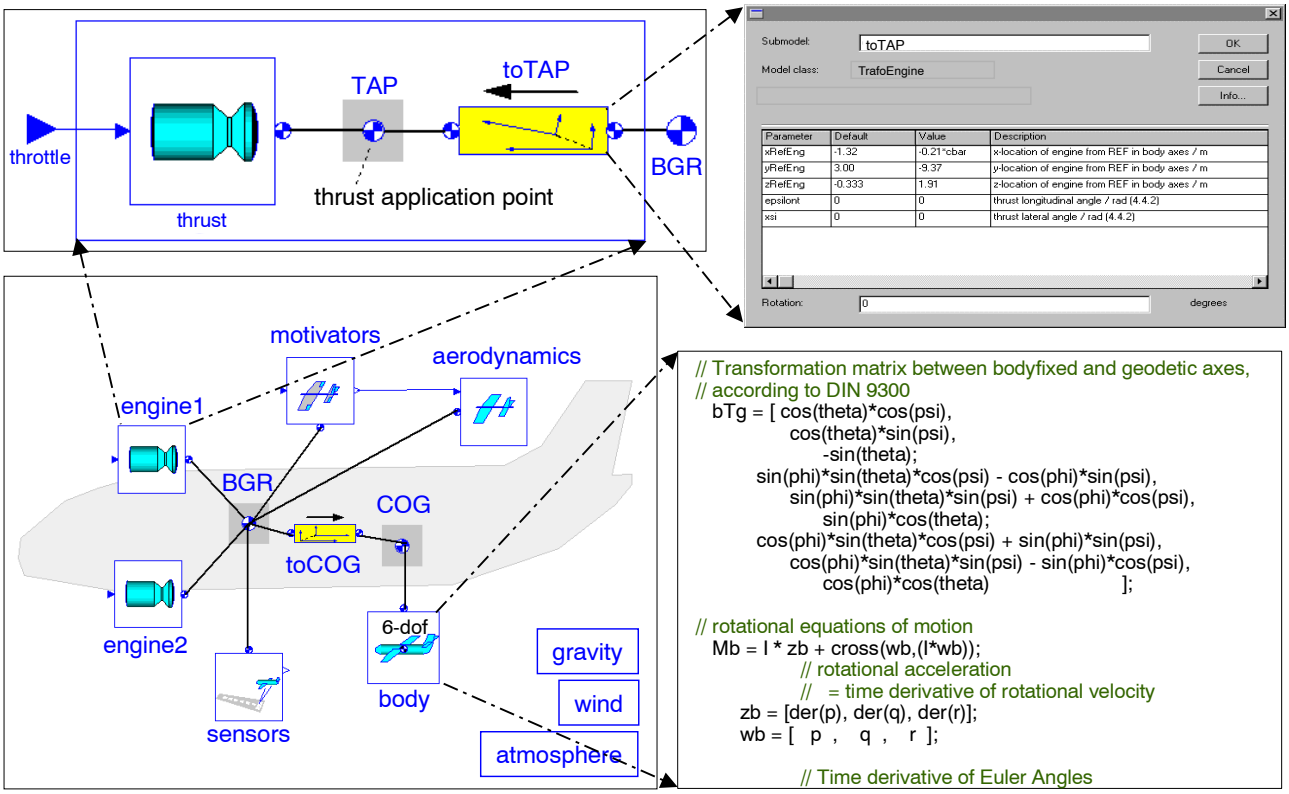

Figure 6: Zoom into an hierarchical structured object-diagram

3 Equation-based model building and efficient simulation code generation

From the model, that is graphically specified by a set of object diagrams, simulation models and documentation of the flight dynamics and systems can be generated automatically.

In the modeling process the object model is composed using different libraries and aircraft specific parameter data (see Fig. 7). The equation handler of DYMOLA solves the equations according to inputs and outputs of the complete aircraft model for a particular task. Equations, that are formulated in an object, but not needed for the specified configuration, are automatically removed in the following model building process. The result is a nonlinear symbolic state space description with a minimum number of equations for this task. Models for efficient parametric nonlinear simulations (section 3.1) can be automatically generated from object models of the Flight Dynamics Library, using the left branch of Fig. 7.

Due to MODELICAs equation-based approach it is possible to invert the interacting flight dynamics and flight systems model symbolically to the highest possible extend. This allows to generate so called 'inverse models' which can be used for trim computations or which become nonlinear Dynamic Inversion (DI) control code [5] within a flight computer. The inversion according to the middle and right branch of Fig. 7 is mainly done by exchanging the external inputs and outputs while still using the same object model which has already been used to generate code for the simulation model. The equation handler of DYMOLA is used to solve all equations according to one of the above specified tasks. The derivation of an highly accurate and very efficient trim code is discussed in section 3.2, the DI-code generation is detailed in [5] and not presented here.

All generated models can be simulated with DYMOLA or with SIMULINK, using DYMOLAS S-function model generator. Additionally, automatic code generation is possible for the real-time engineering flight simulator AVDS (section 3.3).

3.1 Efficient parameterized simulation models

Generally a simulator requires that system models are transferred to a state space description:

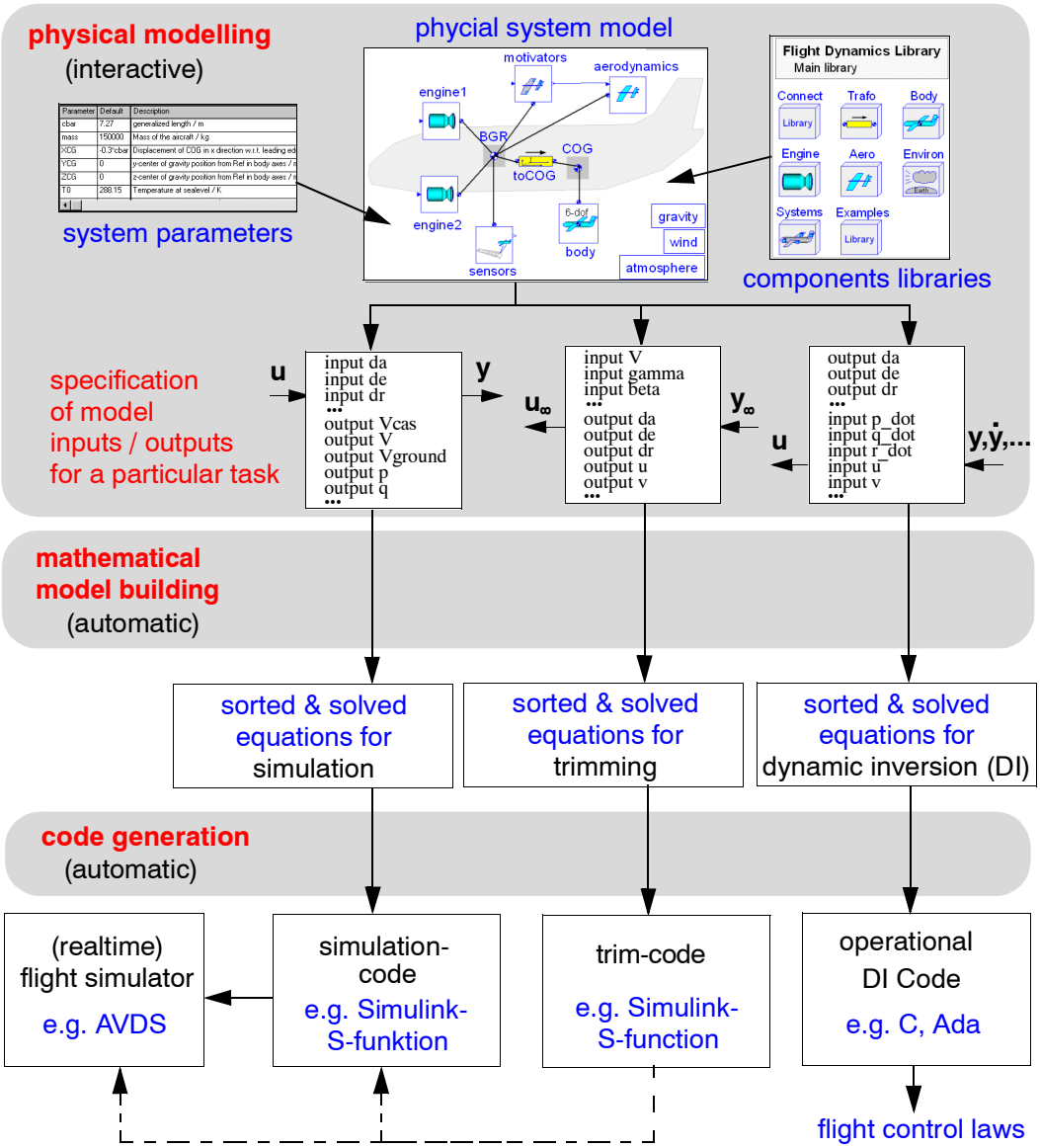

Figure 7: Equation based model building process

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}_s, \mathbf{par}, t)$$
$$\mathbf{y}_s = \mathbf{g}(\mathbf{x}, \mathbf{u}_s, \mathbf{par}, t)\,.$$

To achieve such a standard description from an object model it is only necessary to assign the aerodynamic and engine controls as simulation inputs $\mathbf{u}_s$ and the measurement and evaluation signals as simulation outputs $\mathbf{y}_s$. For the generation of simulation models the state vector $\mathbf{x}$, which consists of the 12 basic flight dynamics states and of additional states of engine dynamics, actuator- and sensor dynamics, is automatically considered as known. Additional inputs are the simulation time t and the parameter vector **par** (e.g. mass, center of gravity position, wing area). Depending on all inputs time derivative of the state vector $\dot{\mathbf{x}}$ and the output vector $\mathbf{y}_s$ is computed. Fig. 8 defines a typical set of inputs, outputs, and states of a flight dynamics model. A typical set of flight dynamics states consist of the velocity (V), the angle of attack (α), the angle of sideslip (β), the angular velocities (p, q, r), the attitude (ϕ, θ, ψ) and the inertial position (x, y, z). Simulation model inputs are the aerodynamic control surface deflections of tailplane (dt), elevator (de), aileron (da) and rudder (dr), the engine controls ($throttle1$, $throttle2$) and, for example, additional gust inputs (u_gust, v_gust, w_gust). Typical simulation model outputs are the measurements signals such as $height$, V, α, β and the roll angle ϕ and the evaluation signal such as the flight path angle γ and the load

states	der(states)	control inputs	outputs
V	der(V) = 0	dt	height
α	der(α) = 0	de = const	V
β	der(β) = 0	da	$\gamma = 0$
p	der(p) = 0	dr	α
q	der(q) = 0	throttle1 = const	nz
r	der(r) = 0	throttle2 = const	β
ϕ	der(ϕ) = 0	u_gust = const	ϕ
θ	der(θ)	v_gust = const	
$\psi = 0$	der(ψ)	w_gust = const	
x = 0	der(x)		
y = 0	der(y)		
z	der(z)		

Figure 8: Inputs, outputs, and states of a flight dynamics model (actuator, sensor models are omitted here)

factor *nz*. For a detailed definition of these variables see, e.g. [1].
Automatic code generation, for example for SIMULINK, is possible separately for subcomponents as specified in Fig. 9 as well as for the complete aircraft model. The latter approach has the advantage that the transformation equations, which are in particular necessary for multi-point models, can be sorted (and eliminated) according the specified task. Algebraic loops, which occur if, e.g., the aerodynamic forces depend on accelerations, can be solved automatically using *tearing* [3].
Before starting a simulation, the initial, stationary inputs and states for a desired point of the flight envelope have to be calculated by a trim procedure. This aspect is dealt with in the following section.

3.2 Accurate trim computation

Trim calculations of complex flight system dynamics models are a very challenging computational task, involving the numerical solution of a system of nonlinear equations to calculate the stationary values of state and control variables. The difficulties mainly arise because of the lack of differentiability in aerodynamic and engine models due to the presence of various look-up tables used for linear interpolations. These severe nonlinearities as well as the presence of, e.g., control surface deflection limiters make the numerical solution of this high order system of equations very challenging.

Simulation environments, such as MATALB/SIMULINK, offer trim routines for this task that use the simulation model to perform trim calculations, which are driven by a numerical optimization algorithm. For this purpose the state derivatives $\dot{\mathbf{x}}$ and outputs of the of the simulation model $\mathbf{y}_s$ are set equal to their desired trim values $\dot{\mathbf{x}}_{tr}$ and $\mathbf{y}_{tr}$:

$$\begin{aligned} \dot{\mathbf{x}} &= \dot{\mathbf{x}}_{tr} \\ \mathbf{y}_s &= \mathbf{y}_{tr} \end{aligned}$$

The trim values of states $\mathbf{x}$ and simulation inputs $\mathbf{u}_s$ are calculated using the following constraint equation:

$$\begin{aligned} \dot{\mathbf{x}}_{tr} - \mathbf{f}\left(\mathbf{x}, \mathbf{u}_s, \mathbf{par}, t\right) &= 0 \\ \mathbf{y}_{tr} - \mathbf{g}\left(\mathbf{x}, \mathbf{u}_s, \mathbf{par}, t\right) &= 0 \end{aligned}$$

The advantage of this numerical approach, using the complete simulation model, is that consistency between simulation model and trim computation is automatically guaranteed by using the same model. The disadvantage of this approach, which is achieved by a very high number of model evaluation of the simulation model, is its rather high calculation time and its comparatively low accuracy ('miss-trim'). The inaccuracy, which increases with the complexity and nonlinearity of the model, results from this procedure neglecting the fact that some of the states directly depend on each other. For example, actuator and sensor states are treated as independent from the flight dynamics states and inputs even though they are directly related to them.
An alternative is to trim the subcomponents of the aircraft model (see Fig. 9) separately. First the flight dynamics submodel is trimmed. The trim values of this submodel are taken to trim in three additional steps the actuator, engine and sensor dynamics models, again using the same numerical trim approach. The trim computation is more accurate compared to the above one-step-approach, but usually more time consuming.
Computation time and accuracy can be improved, if actuator and sensor model variables are not trimmed by an optimizer but directly set by the user. For example, actuator states and inputs can often be directly related to flight dynamics inputs and sensor states and outputs to flight dynamics outputs. The disadvantage of this approach is, that the procedure takes more engineer interaction and therefore increases the likeliness of errors, if submodels change during design.
A third option for trim computations, which is base on model inversion, is proposed here. The symbolic

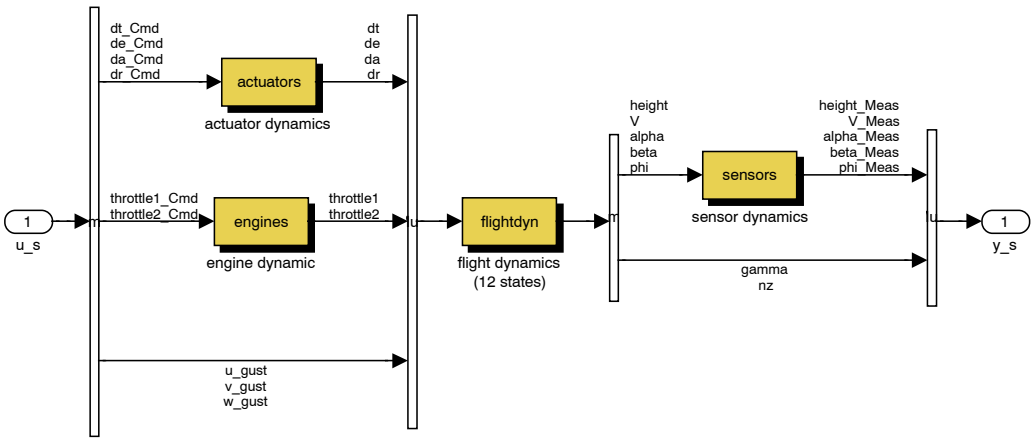

Figure 9: SIMULINK-block diagram of the flight system dynamics

engine of DYMOLA allows to generate C-code for an 'inverse model' to serve for trimming. To serve this purpose the inputs and outputs of the simulation model are inverted. The states derivatives $\dot{\mathbf{x}}$ become known, also the outputs of the simulation model $\mathbf{y}_s$, which are now the inputs of the trim model $\mathbf{u}_t$. The unknown variables are the the inputs of the simulation model $\mathbf{u}_s$ = $\mathbf{y}_t$ and the states $\mathbf{x}$:

$$\begin{aligned} \mathbf{y}_t = \mathbf{u}_s &= \mathbf{h}\left(\mathbf{u}_t, \dot{\mathbf{x}}, \mathbf{par}, t\right) \\ \mathbf{x} &= \mathbf{j}\left(\mathbf{u}_t, \dot{\mathbf{x}}, \mathbf{par}, t\right). \end{aligned}$$

One trim condition is specified in Fig. 8. In contrast to the simulation model code generation of section 3.1, where the variables are column-wise known or unknown, the known variables of the trim model (trim inputs) are shaded grey, whereas the unknown variables (trim outputs) are not. For each variable of the simulation model changed from known to unknown, one other variable is changed from unknown to known. The balance of known to unknown variables is kept equal. The inputs of the trim model are the desired trim conditions (such as velocity V and angle of attack α) and the outputs are the corresponding equilibrium values of trimmed state and aerodynamic and engine control vectors. DYMOLA generates essentially explicit equations for the inverse model by solving the high order nonlinear equation symbolically to the highest possible extend. Even if it is not possible to determine a symbolic solution, DYMOLA is still able to reduce the burden of solving numerically a high order system of nonlinear equations to the solution of a small core system of nonlinear equations which ultimately must be solved numerically.

The proposed trim approach based on model inversion was compared to the traditional approach in the HIRM benchmark flight dynamics model of the GARTEUR Flight Mechanics Action Group on "'New Analysis Techniques for Clearance of Flight Control Laws"' [4]. An optimization based clearance process for flight control systems requires highly precise computations of trim values, because these values are the base for the following nonlinear or linear analysis. Even very small inaccuracies in trim values can corrupt the optimization progress. Here the trim computation based on an inverse model has proven its advantage compared to a standard optimization based trim approach. Another advantage of the inverse model trim approach is its computational time efficiency. Trimming the same, highly nonlinear flight dynamics model took, depending on the trim point, between 15 and 65 seconds using conventional trimming, just 50 to 70 milliseconds using the inverse model. Both trim computations were done within MATLAB/SIMULINK on a 400MHz personal computer [6].

3.3 Interactive Real-time Simulation using the Engineering Flight Simulator AVDS

For the evaluation of critical flight conditions and for the validation of flight control systems an aircraft animation tool can help the design engineer to analyze the aircraft performance. The Aircraft Visual Design Simulator (AVDS [7]) is such a tool to fill the gap between batch and motion-based simulation by allowing the flight control design engineer to quickly test and re-test response in a real-time environment on a low-cost PC (Fig. 10).

Fig. 11 illustrates the data flow within the interactive mode of AVDS, which allows the design engineer to virtually 'fly' the aircraft. Using, e.g. the cockpit-view together with the head-up display (HUD) of Fig. 10,

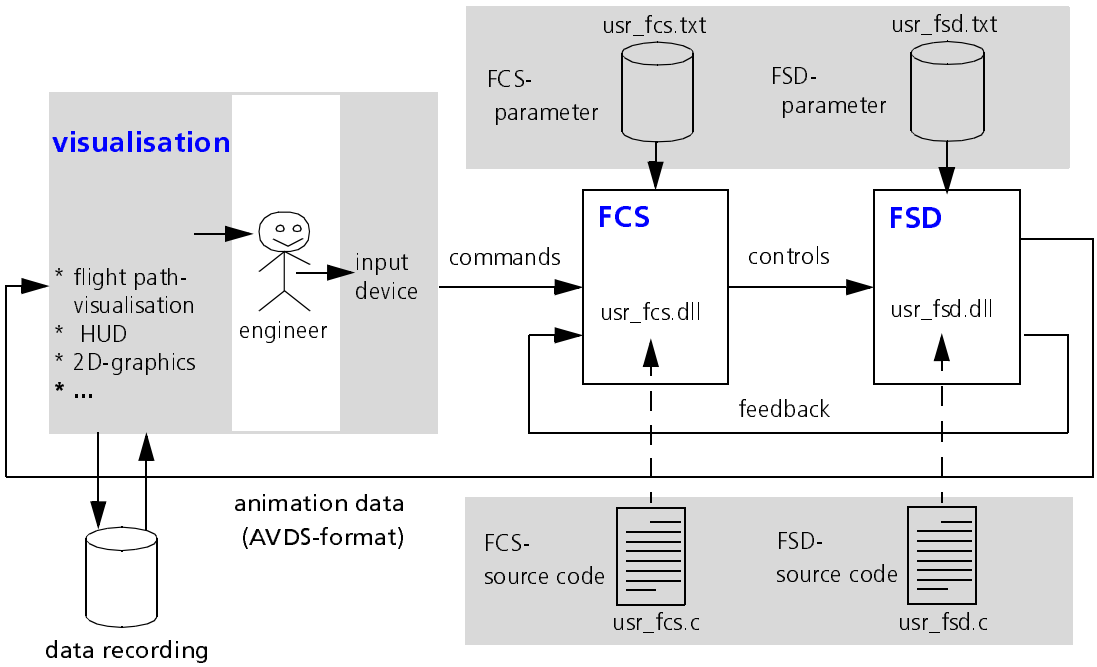

Figure 11: AVDS data flow in interactive simulation

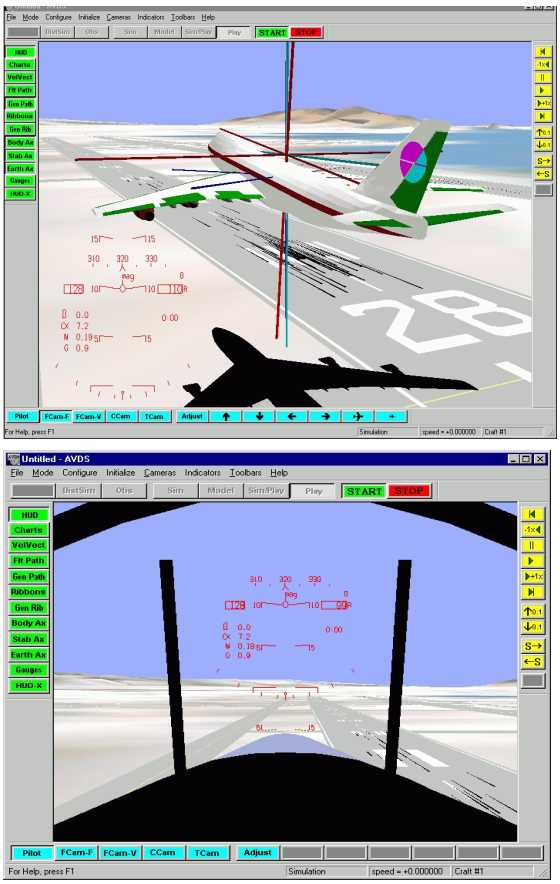

Figure 10: AVDS interactive real-time flight simulation

the engineer controls the aircraft via input devices like mouse, keyboards and other control instruments. These commands are transferred to the flight control system (FCS) and result in the controls of the flight system dynamics model (FSD). For the implementation of flight dynamics models AVDS offers an interface which consists of C-subroutines for controller and flight dynamics with systems with its corresponding set of parameters.

To avoid a manual re-implementation of these model codes, we propose to automatically generate the AVDS-codes and their parameter sets starting from the same object model as used for the parametric simulation (section 3.1). The big advantage using the MODELICA-AVDS interface is the complete automation of the code generation. The trim computation (as detailed in section 3.2) is already included in the code. This means that neither initial states of flight dynamics or systems have to be directly assigned by the user nor that an external, separate trim tool has to be used to specify flight conditions in AVDS.

The strategy of interfacing MODELICA flight dynamics models to AVDS can be transferred to any other flight simulator. The symbolic equation handler of DYMOLA guarantees a highly efficient model code. Different levels of detail do not have any influence on the interface structure. The only limit is the computational power of the platform which is used to run the flight simulator.

4 Conclusion

Complex aircraft models including actuator and sensor dynamics in addition to electronic flight control systems, are aggregated from contributions of many different disciplines involved. This paper shows that complex models are best comprehended if each disciplinary contribution is described in its own specific domain. For flight dynamics, the MODELICA Flight Dynamics Library serves this purpose.

For systematic and transparent modeling, it turned out to be important to describe all aircraft components and physical phenomena locally with respect to their intrinsic reference points, which usually have an offset in position and orientation from the aircraft's body geometric reference BGR.

The computer aided model building technique allows the modeling of engineering systems such as flight dynamics on a physical level in the form of declarative mathematical equations specifying energy exchange and kinematic constraints.

The equation-based modeling language of MODELICA allows the generation of codes for an *inverse model* to serve for trimming. Such a model has as inputs the desired trim conditions and as outputs the corresponding equilibrium values of trimmed state and controls vector. The equation handler of DYMOLA generates essentially explicit equations for the inverse model by solving the high order nonlinear equation to the highest possible extend symbolically. Thus, the trimming procedure based on such an inverse model has proven to be very accurate and fast compared to conventional optimization based trim procedure.

The code generation facility of DYMOLA allows the use of different simulators, (e.g., MATALB/SIMULINK, DYMOLA's own simulation environment, the flight simulator AVDS) as a run-time environment for model execution. Using the Flight Dynamics Library offers the opportunity that trim code is automatically included into the simulation model and executed at simulation start. This means that no separate trim tool has to be used.

References

[1] R. Brockhaus. *Flugregelung*. Springer Verlag, Berlin, 1994.

[2] R. Brockhaus. A Mathematical Multi-Point Model for Aircraft Motion in Moving Air. *Zeitschrift für Flugwissenschaften und Weltraumforschung*, pages. 187-184, 1987.

[3] H. Elmqvist und M. Otter. Methods for Tearing Systems of Equations in Object-Oriented Modeling. In *Proceedings ESM'94 European Simulation Multiconference*, pp. 326-332, Barcelona, Spain, 1994.

[4] GARTEUR FM(AG11). Scope of a new GARTEUR Flight Mechanics Action Group on "'New Analysis Techniques for Clearance of Flight Control Laws"'. Group for Aeronautical Research and Technology in Europe (GARTEUR), Technical Report GARTEUR TP-119-1, 1999.

[5] G. Looye. Design of Autopilot Control Laws with Nonlinear Dynamic Inversion *at Automatisierungstechnik*, pp. 523-531, No. 12, 2001.

[6] D. Moormann. *Automatisierte Modellbildung der Flugsystemdynamik*. Dissertation, RWTH Aachen. VDI Fortschrittsberichte, Mess-, Steuerungs- und Regelungstechnik, Reihe 8, No. 931, 2002.

[7] S.J. Rasmussen and S.G. Breslin. AVDS: A Flight Systems Design Tool for Visualization and Engineer-in-the-Loop Simulation. *AIAA Modeling and Simulation Technologies Conference*, No. AIAA-3467-97, 1997.

[8] M. Tiller. *Introduction to Physical Modeling with Modelica*. The Kluwer International Series in Engineering and Computer Sciences. ISBN 0792373677, 2001.