MULTI-DISCIPLINARY AIRCRAFT MODEL DEVELOPMENT USING OBJECT-ORIENTED MODELLING TECHNIQUES

Gertjan Looye, Simon Hecker, Thiemo Kier, Christian Reschke, Johann Bals

DLR – German Aerospace Center, Institnte of Robotics and Mechatronics
Oberpfaffenhofen
D-82234 Wessling, Germany
e-mail: Gertjan.Looye@DLR.de

ABSTRACT

In this paper a model component library for construction of mnlti-disciplinary aircraft flight dynamics models is presented. This library is based on the object-oriented modelling language Modelica, which has been designed for modelling of large scale multi-engineering systems. The Flight Dynamics Library, developed by DLR, allows for graphical construction of complex rigid as well as flexible aircraft dynamics models and is fully compatible with other available Modelica libraries for electronics, thermodynamics, control systems, etc. In the paper the structure of the Flight Dynamics Library and an example model of a flexible transport aircraft will be presented. The development of model components, construction of models, generation of simulation code, as well as the possibility of automatic generation of inverse models (e.g. for control laws or trim computation) will be demonstrated. The efficiency of the generated code allows for real-time simulation of complex flexible aircraft models.

1. INTRODUCTION

Dynamic simulation plays an important role in the aircraft design and certification process. Typical examples are development of flight control laws, flight loads analysis, specification and testing of on-board systems, etc. Involved engineering disciplines develop models for their specific types of analysis. However, since a long time it has been recognised that considerable performance may be gained by exploiting inter-disciplinary aspects. For this reason, discipline-specific models are more and more combined into integrated aircraft dynamics models, allowing interactions to be analysed and addressed. Furthermore, for cost reasons hardware rig testing and flight testing are more and more replaced with simulation. For this accurate dynamic models are needed in order to reliably predict the behaviour of system components in interaction with other systems, engines, as well as aircraft flight and structural dynamics.

Model integration may be achieved at a tool level by direct communication between engineering environments, or at a computer code level (C, FORTRAN). The latter is relatively straight-forward, since models are either already implemented this way, or the applied engineering tools offer export capability of such code. Most ideally, the model components are available in one and the same environment. This may require models from other disciplines, based on different modelling formalisms, to be translated into the formalism underlying the selected integration platform. In spite of this required effort, block-diagram modelling (especially Matlab/Simulink [21]) has found application in a wide variety of engineering domains. Consequently, the approach is more and more used as a common platform for multi-disciplinary model integration.

Tool- or code-based integration of simulation models are pragmatic approaches, bnt have a number of drawbacks. Coupling of tools is nnlikely to allow all disciplinary aspects to be included. It may further be slower than necessary and may be costly because of licensing. Code-based integration may require considerable software-engineering skills, but more importantly, it obscures the structure and contents of model components. Consequently, model components provided by the one discipline will remain a black-box to an engineer in the other discipline. Block diagram-oriented tools improve model visibility as well as construction due to their graphical possibilities. However, this methodology has an important aspect in common with code-based implementation: the user is forced to formnlate model components in the form of ordinary differential equations, with specific inputs and outputs. The required manipulation of underlying equations easily obscures the physics behind the model component, even when implemented graphically. The above is illustrated for a simple system in Figure 1.

It is for these reasons that in 1978 Elmqvist [6] proposed a dedicated modelling language, called Dymola (Dynamic Modelling Language). The principal feature of this language is that the user is not forced to specify solved model (differential) equations and is not limited to causal interconnections between model components. Physical equations may rather be entered in their textbook form, and interconnections may be bi-directional flows, or simply constraints. This has two important advantages:

1. physical objects and phenomena and their interactions may be implemented as model objects and model interactions respectively in a one-to-one fashion;
2. component libraries based on discipline-specific modelling paradigms may be developed, but based on a common language base.

In 1992 the first modelling and simulation environment

Example: dynamic structure in air flow
Consider the following generalised equation of motion of a structure:

$$M\ddot{\eta} + B\dot{\eta} + K\eta = Q \quad (1)$$

where M, B, K are generalised mass, damping, and stiffness matrices, η are generalised co-ordinates (mode shape multipliers), and Q are generalised forces acting on the structure. It is assumed that Q are aerodynamic forces, induced by deformation of the structure:

$$Q = \bar{q}\,[A_2\ddot{\eta} + A_1\dot{\eta} + A_0\eta] \quad (2)$$

where $\bar{q}$ is the dynamic pressure, and A_2, A_1, A_0 are aerodynamic mass, damping and stiffness respectively. In order to make the above equations suitable for simulation, these must be written in the form of an ordinary differential equation:

$$\ddot{\eta} = M^{-1}[Q - B\dot{\eta} - K\eta]$$

A block diagram of the combined equations is shown in Figure 2. The $\frac{1}{s}$ blocks are integrators. The diagram retains the separation between the equations of motion and the aerodynamic force equation, although formulation of the physical equations as a chain of input-output blocks is something most engineers have to get used to. The dependency of Q on $\ddot{\eta}$ introduces an algebraic loop. Although such simple loops are nowadays easily solved by most block-diagram simulation tools, in more complex cases a loop-breaking element may have to be added (usually a one-time step delay), or the equations need to be solved by hand:

$$\ddot{\eta} = -(M - \bar{q}A_2)^{-1}[(B - \bar{q}A_1)\dot{\eta} + (K - \bar{q}A_0)\eta]$$

Physical modelling
Besides manual derivation and coding (either via programming or by composing a block diagram), a third form of implementation is illustrated Figure 3.
The system is represented by two objects: one containing structural, the other containing aerodynamics equations. The kinematics of these objects (i.e. η and derivatives) are constrained to be equal and forces (Q) are in equilibrium. This is represented by the interconnection between the two objects. The variable η herein is handled as an "across" variable (i.e. set equal between the components) and Q is handled as a "through" or "flow" variable, indicated by the superscript "f". Flow variables are summed up to zero between connectors. Note that addition of any other components (e.g. a propulsion model) is very straightforward, since the very same interconnection principles apply.
The implementation in Figure 3 illustrates the basic principle of physical or object-oriented modelling using Modelica. The contents of the blocks consist of the physical equations, as well as declarations of internal variables and parameters. Figure 4 illustrates the implementation of the aerodynamic model equations using Modelica language. Its graphical representation is captured in annotations, which have been left out here.

Figure 1. Modelling example, comparing block diagram and object-oriented modelling

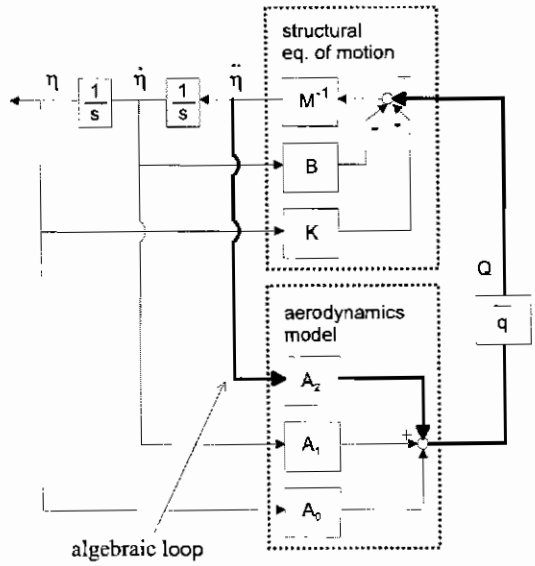

Figure 2. (with Figure 1) Implementation of the example system using block diagram modelling

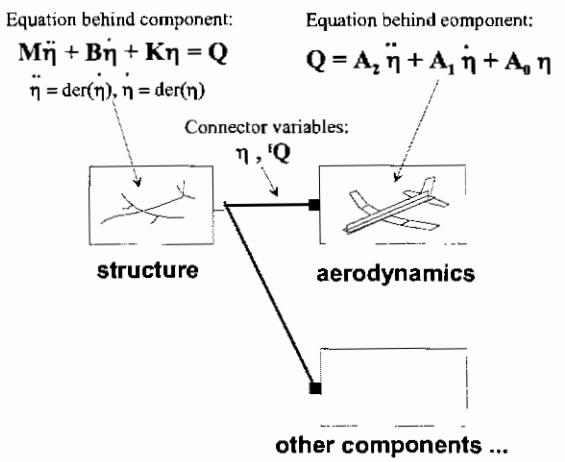

Figure 3. (with Figure 1) Implementation of the example system using physical modelling

based on the Dymola language, also called Dymola (Dynamic Modelling Laboratory) became available. In the mean time, several other research groups started developing dedicated physical modelling languages and tools. All languages are strongly supported by graphical tools (Figure 3), allowing model components to be constructed from discipline-specific libraries using "drag-and-drop". Since libraries may in turn be based on discipline-specific modelling paradigms, an electronic circuit (for example) also graphically looks as such, and a control law may still be implemented in the form of a block-diagram.

In 1996 several developers in the physical modelling field, as well as simulation and modelling experts joined in a non-profit organisation [23], with the objective to develop a common, free physical modelliug language standard, named **Modelica**. By now, the Modelica language has achieved a high degree of maturity and several modelling tools ("Modelica translators") have become available, featuring graphical model com-

```
model aerodynamics

  /* Declarations:  */

 constant SI.Length cref = 10; /* Reference length, declared
                                  as Real, with SI unit of
                                  length (= meter)          */

 parameter Integer nFlex = 20 "Number of flexible modes";
 parameter SI.Velocity V =200 "Equivalent airspeed";

 ... any other variables and parameters

 /* Declare generalised co-ordinate vector and derivatives: */
 Real   eta[nFlex] = MeanAxes.eta;
                           /* "MeanAxes" is the connector */
 Real  deta[nFlex] = der( eta);
                           /*  deta is derivative of  eta */
 Real ddeta[nFlex] = der(deta);
                           /* ddeta is derivative of deta */

protected /* Variables hidden for outer world: */

 Real A0[nFlex,nFlex]; /* Aerodynamic "stiffness" */
 Real A1[nFlex,nFlex]; /* Aerodynamic "damping"   */
 Real A2[nFlex,nFlex]; /* Aerodynamic "mass"      */

initial equation
           /* Equations evaluated before simulation start: */

 /* Load aerodynamic data from file */
 (A0,A1,A2) = GetAeroData(whichFile,whichCase,...);

equation /* The actual model equations: */

 /* Dynamic pressure */
 qdyn = 1.225*V^2 / 2;

 /* Model equation (in its "textbook" form): */
 MeanAxes.Q = -qdyn*(A2*(cref/V)^2*ddeta+A1*crefdV*deta+A0*eta);

end aerodynamics;
```

Figure 4. (with Figure 1) Modelica code behind example aerodynamics block in Figure 3

position and extensive simulation capabilities. A wide range of libraries has been developed (e.g. multi-body systems, electronics, thermodynamics, block diagrams, power trains, hydraulics, pneumatics, fuel cells, bond graphs) that on the one hand allow for composition of discipline-specific model components, while on the other hand these components may be used along side in a single multi-disciplinary model. This allows for development of large scale, intuitively structured hierarchical models. Papers on several complex examples may be downloaded from the Modelica home page [23].

Back in 1995 the DLR institute of Robotics and Mechatronics developed a first library (at the time, based on the Dymola language) for modelling of aircraft flight dynamics [26]. Objective was to build a solid basis for constructing integrated dynamic aircraft models, including flight dynamics, detailed on board system dynamics, structural dynamics, etc. First applications were a generic transport and a fighter aircraft [19] and a first flexible aircraft shortly thereafter [15]. Since then, the library has been expanded and applied to complex aircraft models that include e.g. system hydraulics and electronics [25, 27]. In the frame of the international projects REAL (Robust and Efficient Autopilot control Laws design, funded by the EU in the fifth frame work programme [30]) and the GARTEUR action group AG-11 on clearance of flight control laws [9], the automatic generation of inverse models for fast trim computation and nonlinear control laws was applied for the first time. Recent application examples are the thrust-vectored X-31A high-angle of attack experimental aircraft and a real-time capable integrated flight dynamics and aeroelastic transport aircraft model, including unsteady aerodynamics, structural dynamics, control system, etc.

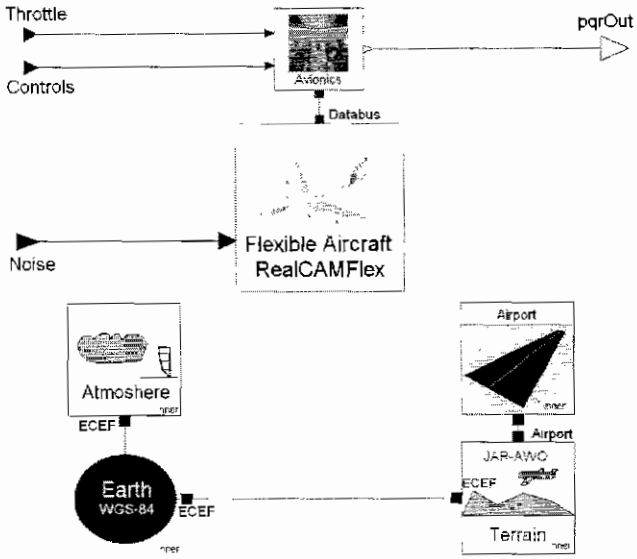

Figure 5. Top-level of model: aircraft and environment

This paper discusses the latest developments, features, and example applications of the Flight Dynamics Library, with emphasis on flexible aircraft. In the following section the selection of a generic aircraft model structure is discussed (Section 2), based on which the Flight Dynamics Library has been organised (Section 3). In Section 4 the automatic generation of model code for model simulation and analysis is discussed, followed by some example applications. Finally, a summary and outlook is given in Section 6.

2. THE AIRCRAFT MODEL STRUCTURE

The objective of this section is to introduce and motivate the basic structure of an aircraft model as may be composed from the Flight Dynamics Library. The structure of this library will be discussed thereafter. The full specification of the underlying Modelica language is given in [23, 24].

In constructing complex models the choice of hierarchy is crucial, since this largely determines how model components interact. For the Flight Dynamics Library a top-level model structure as shown in Figure 5 has been adopted. It consist of one or more aircraft, and environment objects (The "Avionics" component and input and triangular-shaped output connectors will be discussed in Section 4). The environment objects include an Earth, Atmosphere, Terrain, and Airport model. Note that the (in this case, single) aircraft model has no direct link with the environment models, which physically makes sense. Using the so-called inner-outer model feature of the Modelica language, these models represent field functions. For example, the aircraft may request its surrounding atmospheric conditions from the atmosphere model, based on its

local inertial position. Any other aircraft (or e.g. sensor) object in the model may do this as well. This is different from most block-oriented libraries, where an atmospheric model is directly linked to, and thus occupied by, the one aircraft. The ability to easily include multiple air vehicles is useful for applications involving mutnal interactions, like towed gliders, wake vortices, air-to-air refuelling, release of missiles, etc.

2.1. The Earth model

The earth model in Figure 5 has the following functions:

1. Provide the inertial reference. To this end, an Earth-Centred Inertial (ECI) frame is used. Its origin is attached to the Earth's centre of mass, its orientation is fixed with respect to "fixed" reference stars [12].

2. Provide the geodetic reference. As indicated in Figure 5, to this end the World Geodetic System 1984 [2] (WGS-84) is used. The object implements an Earth-Centred Earth-Fixed (ECEF) reference frame, which has the same origin as the ECI, but rotates with the earth. The attitude of the ECEF (w.r.t. ECI) is available in a connector. A set of functions transform ECI and ECEF referenced position vectors into geodetic longitude, latitude, and height co-ordinates (w.r.t. WGS-84 ellipsoid) and vice versa. For a given longitude and latitude, another function provides the local undulation of the so-called EGM96 (Earth Gravitational Model 1996) geoid with respect to the WGS-84 ellipsoid, providing the Mean Sea Level (MSL) reference.

3. Implement a model of the Earth's gravitation. The gravitational model to be used with WGS-84 is the Earth Gravitational Model 1996 (EGM96), provided in the form of tables describing equipotential surfaces a function of longitude and latitude. Cnrrently, a more simplified height and geocentric latitude-dependent (Ref. [32] - Eqn.(1.4-16)) and a constant gravity model are available.

Double-clicking on the Earth in Figure 5 object allows a number of parameters to be set, like whether the Earth is rotating or in rest, initial day time, and the type of gravity model (approximate EGM96, height independent, or constant). The features of the object may be overkill for many applications, but provide sufficient generality for use with for example high speed and high altitude flight vehicles. Furthermore, the applied WGS-84 ensures compatibility with most flight simulator vision systems, navigation system models, etc. Obviously, any parameter set in the Earth and other environment models applies to all aircraft.

2.2. The Atmosphere model

The second environmental object in Figure 5 is the atmosphere. Normally, the International Standard Atmosphere (ISA) as a function of the height above MSL is used. Alternatively, parameters for constant atmospheric conditions may be entered. The airmass is nominally assumed to be in rest with respect to the ECEF, explaining why a connection with the Earth ECEF-connector exists. However, the component also foresees implementation of wind fields. Currently, wind components in northern and eastern directions may be entered at a reference altitude of 100 ft above the earth surface. A simple earth boundary layer model logarithmically reduces the wind velocity to zero on the ground.

2.3. The Terrain model

To the right in Figure 5 a terrain model has been added. A component containing highly detailed, or simple parameterised models of the Earth's surface may be selected from the library. Depicted in Figure 5 is a terrain model as used for automatic landing control law design and certification, based on JAR-AWO specifications [11]. The location, elevation, direction, and slope of a runway may be specified, as well as slopes and steps in the terrain below the approach path. A simple function call from e.g. an aircraft sensor then returns the corresponding local terrain elevation above MSL, allowing for computation of for example the radio altitude.

2.4. Airport infrastructure model

The airport block implements earth-fixed navigational equipment (e.g. VOR, DME, ILS systems at specified locations). In the figure the ILS equipment of the one rnnway as positioned in the JAR-AWO terrain model is inclnded. Specific characteristics like glide slope angle and antenna transmitter positions may be specified via parameters. Any other model object may obtain its local glide slope and localiser deviation via a simple function call.

2.5. Rigid and flexible aircraft models

The core of the model structure is of course the component that represents the actual aircraft. The Flight Dynamics Library foresees the implementation of rigid just as well as flexible ones. A typical model structure for a flexible transport aircraft is shown in Figure 6. The specific example was originally used in the EU project REAL [30], as a benchmark model for automatic landing control laws design. The model is called RealCAM (REAL Civil Aircraft Model), and has been further extended in the frame of this work.

Kinematics

The backbone of the model depicted in Figure 6 are the Kinematics and Airframe blocks. The first defines a "North-East-Down" local vertical frame with its origin moving with a fixed position in the aircraft, preferably the centre of gravity. The object also defines a right-handed body-fixed reference frame with its origin at the same location, but with a fixed attitude w.r.t. the airframe (x-axis towards the nose, z-axis down). The attitudes and inertial positions of both reference systems are available in the two connectors (top: body,

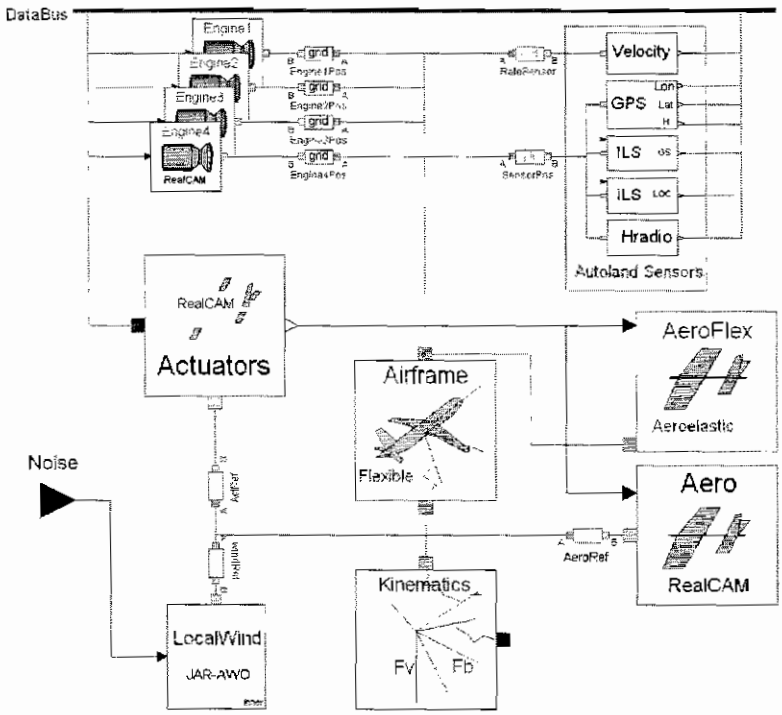

Figure 6. Structure of a Flexible Aircraft model in Figure 5

right: NED). For the kinematics block various versions are available or in preparation, like 6 Degrees of Freedom (DOF) with Euler angles and WGS-84 co-ordinates as states, as well as versions with 3 DOF (longitudinal or lateral), Quaternion states, or earth-fixed position states.

Airframe

The difference between a rigid and a flexible aircraft is, in fact, only in the Airframe object. In case of a rigid airframe, it contains the standard Newton-Euler force and moment equations with respect to a body reference system (attitude and position in lower connector). Although the origin of this reference system is preferably the centre of gravity (for compatibility with standard flight dynamics models), a fixed point w.r.t. the undeformed airframe shape may be more useful for referencing reasons. The local gravity acceleration is obtained by a call to the Earth object (Figure 5). Note that the computation of gravity depends on the method that is set in the Earth object. In case of a flexible airframe, linear elastic equations of motion in modal form augment the Newton-Euler equations [33, 5, 29]. The body axes system is hereby considered as a so-called mean axis system. The momentary shape of the airframe is characterised by states in the form of generalised co-ordinates. The underlying data (modal mass, damping, stiffness, and mode shape matrices) are automatically read from a specified file prior to simulation.

Connection of the Airframe block to the Kinematics block (see Figure 6) makes that the reference systems in both connectors merge, i.e. from then on the airframe is moving freely with respect to the inertial reference, with its kinematics described in the corresponding object.

The Airframe object has a second connector on top (see Figure 6). This connector may contain a different reference frame with a constant offset, or may simply be identical to the body frame (to be specified via an offset parameter). It is intended for interconnection of for example external force model components, sensor models, etc. In case of a flexible airframe also generalised co-ordinates are contained. Besides kinematic variables, each connector also describes (generalised) forces and moments along the local reference systems axes, declared as flow variables (see Figure 1).

External forces and moments

The airframe equations of motion are primarily driven by aerodynamic and propulsion forces and moments. These are computed in corresponding model components in Figure 6. These components often need to be prepared for each aircraft type individually, since application rules and data (sources) behind aerodynamics and propulsion models may strongly differ. For this reason, a base class is available that already defines interfaces as well as equations for computation of key variables like the angle of attack, side slip, true airspeed, etc. Local wind velocities are hereby requested from the Atmosphere model in Figure 5. The user may develop own model components, inheriting this base class.

Besides the airframe, each component may be developed around its own local reference frame. In case of aerodynamics, these may for example be the stability or wind axes. Interconnection with the airframe follows via a transformation object (e.g. AeroRef in Figure 6). This object has two connectors representing

two reference systems. The offset (position, orientation) in between may be specified via parameters that become visible and can be edited by double-clicking on the object. The object also relates the forces and moments that act along the connector reference systems. When connecting a model with the Airframe object, the transformation object makes sure that the kinematics between the local component and the airframe reference systems are correctly related, as well as forces and moments are applied correctly.

The aircraft model in Figure 6 has two aerodynamics models (right hand side). The lower one ("Aero") contains forces and moments as induced by the over-all motion of the aircraft ("rigid aerodynamics"), usually also corrected for quasi-steady deformation of the airframe. The underlying model may be based on complex application rules, table look-ups, etc. In case a data set is not available or incomplete, computational tools as described in [13] are available. The upper aerodynamics component ("AeroFlex") computes unsteady (generalised) forces and moments as induced by flexible deformation of the airframe. For this component extensive pre-processing tools have been developed, involving application of the Doublet-Lattice Method (as available in MSC-NASTRAN [22]), axis transformations, Rational Function Approximation and removal of quasi-steady effects (already accounted for in the rigid aerodynamics model), see [17] for more details. The unsteady aerodynamic data are read from a user-specified (Matlab) file at simulation start.

Note that the Aero component is connected to the lower Airframe connector via the AeroRef object, whereby the latter describes the offset between the airframe body axes and the aerodynamic reference system. The upper aerodynamics components is directly connected with the upper Airframe connector, making use of generalised co-ordinates declared therein. Both components are also connected to the "Actuators" model. In this case control surfaces deflections are passed on from the latter. However, the interconnection may also address balance and kinematics between aerodynamic and actuation forces.

The engine models (top left) are connected to the airframe via a slightly different type of transformation. Instead of an offset, the uumber of a structural grid point, where the object is to be attached, may be specified. At simulation start the transformation object requests the rows of the modal matrix that apply to the grid point from the Airframe object, allowing it to continuously compute the kinematic relation and force balance between its connectors as a function of the offset from the airfraune reference and the local deformation. This for example iuplies that directional thrust variations due to local deformation at the eugine attachment point are automatically taken into account.

Sensor models
The very same principle as used for interconnecting engine models with the airframe structure also applies to the sensor models, located in the top-right corner of Figure 6. A set of sensor types is available in the library. For example, accelerometers compute local accelerations at their point of attachment (specified via grid point number, or offset) as a function of the inertial motion of the airframe, its position in the airframe reference, as well as the local airframe deformation.

The ILS, GPS, and radio altimeter sensors obtain their values by making a function call to the Airport, Earth, and Terrain environment models (Figure 5), passing on their momentary inertial position as an argument. In this way, for example multiple GPS sensor objects may be included at various locations on the airframe. Each object can request its very local co-ordinates from the Earth object.

Local wind effects
As already discussed in Section 2.2, mean winds are computed in the atmosphere block at the top level of the model in Figure 5. However, turbulence models are usually described in aircraft body axes, whereby delays as gusts travel along the airframe, are taken into account. This is described in the LocalWind object (lower left, in this case based on JAR-AWO specifications for autoland assessment). Random turbulence velocities are obtained from dedicated filters (Dryden, Karman) that use white noise signals as inputs. This noise is provided via an external connector.

Systems
On-board systems are included in the Actuators component. This component may describe actuators and hydraulic / electric systems using simple transfer functions, as well as highly detailed physical models, constructed from hydraulics and electronics libraries. The library currently only provides the first variant, since detailed on-board system models are unique for each type or family of aircraft and are usually provided by systems specialists. A recent example of on-board system model implementation using Modelica can be found in [3].

Avionics bus
Finally, the thin bar at the top of Figure 6 represents a so-called data bus. This bus includes signals that one would typically find on avionics buses iu the aircraft, like the readings of all sensors, command signals to engine and control surface actuators, gear status, etc. For this reason, the sensor, actuator, and engine models have been attached to the bus object. The bus is also accessible from outside aud allows direct connection to elements from the Modelica block diagram library. This enables a coutrol system composed using this library to directly communicate with the aircraft data bus. When connecting a model component the user is asked which variable is to be inserted into, or retrieved from the data bus. A large amount of commonly used variables has been pre-defiued, but the user is of course free to add more.

3. THE FLIGHT DYNAMICS LIBRARY

The top-level structure of the Flight Dynamics Library is depicted in Figure 7. The "Modelica" branch in the depicted tree (top) contains the principal standard libraries delivered with Modelica (as listed in Section 1). The branches of the Flight Dynamics Library will be briefly described below:

- **Aerodynamics** contains example aerodynamics models for use in rigid and flexible aircraft models, as well as base classes that the user may extend (inherit) to develop his own aircraft-specific model components. Each aircraft type or family namely tends to use unique application rules.
- **Airframes** contains rigid and flexible airframe model objects. The rigid ones may have constant mass and inertia tensor (entered via parameters), or these may change at a given rate (e.g. as a function of fuel consumption). The flexible airframe components loads its mass, and modal data from an external file (usually a Matlab mat-file [20]).
- **Environment** contains all environment-related models as described at the beginning of Section 2. A WMM-2005 based magnetic field model is to be added.
- **Examples** contains actual implementations of aircraft models, as for example depicted in Figures 5 and 6.
- **Gear** currently contains a simplified landing gear model for which basic properties may be set and which may be attached to the Airframe object in Figure 6. A base class containing a standard interface for interconnection with the Airframe is provided for implementation of detailed landing gear models, e.g. composed with help of the multibody library by specialists in the field.
- **Interfaces** contains all library-specific connector types, as well as the data bus that was discussed Section 2.5.
- **Kinematics** contains the various types of models describiug the kinematics of the body reference system with respect to the local NED, as well as the inertial reference. Versions with e.g. coustraiued degrees of freedom are available as well.
- **Propulsion** contaius, as for the aerodynamics, example engine model implementations, as well as base classes that allow the user to implement his own.
- **Systems** mainly contains sensor models (accelerometers, ILS, GPS, etc.) with time constants and noise if desired, and simple transfer function-based actuator models.
- **Transformations** contains staudard transformations between refereuce systems.
- **Types** contains type definitions for internal variables, to which the user may add his own.
- **Utilities** contaius miscellaneous functions e.g. for reading external data files.

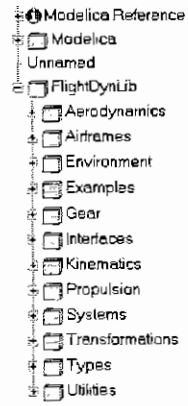

Figure 7. Top-level structure of the Flight Dynamics Library

Within a dedicated Modelica modelling and simulation environment, like Dymola (Dynamic Modelling Library, see `http://www.dynasim.se`) aircraft models may be composed from the library using drag and drop, see Figure 8. After copying a component into the model, its parameters may be edited by double-clicking on the object. For example, in case of the Flexible Airframe the number of modes to be considered may be altered, as well as the way of handling remaining modes (e.g. truncation or residualisation).

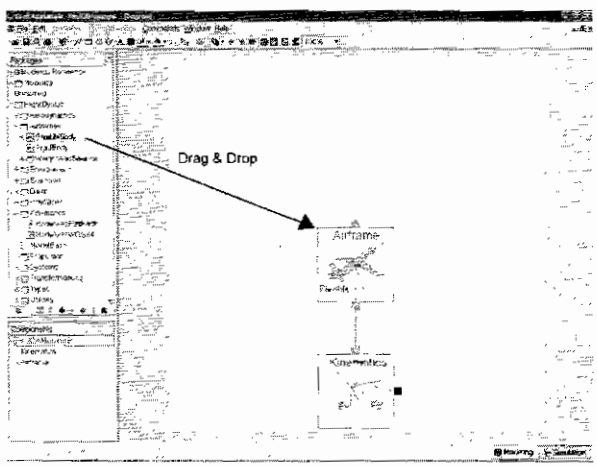

Figure 8. Graphical aircraft model composition using drag and drop from the library

4. AUTOMATIC CODE GENERATION

After model composition has been finished, a model translator sorts and solves all model equations according to specified inputs and outputs into Ordinary Differential Equations (ODE's) or Differential Algebraic Equations (DAE's), suitable for use in simulation. A modelling tool that is well capable of doing this is Dymola [7]. Besides a graphical modelling environment and advanced symbolic algorithms, the tool offers extensive simulation and data analysis capabilities. However, the model code may be used in other engineering environments aud simulation tools as well, like for example Matlab/Simulink [21]. For this environment an additional tool set has been developed that automatically generates trimming and linearisation scripts, allowing the user to easily specify and accurately compute initial conditions prior to simulation [15].

4.1. Specification of inputs and outputs

A simple way of specifying model inputs and outputs is shown at the top of Figure 5. Here a so-called Avion-

ics block has been connected to the bus connector of the aircraft. At this main model level, also input and output connectors have been defined. The Avionics block injects pilot throttle and control snrface input commands (from Throttle, Controls connectors) into the data bus. Output variables of interest, in this example case only body angular rates (p, q, r) are read from the bus and passed on to an output connector ("pqrOut").

4.2. Inverse model generation

Probably one of the most attractive features of Modelica, in combination with a model compiler like Dymola, is the possibility to generate inverse models just as easily as normal simulation models. Inverse models are extremely useful for fast and accurate trim computation, systems and control snrface sizing, as well as for automatic generation of control laws that are based on inverse model equations, like Nonlinear Dynamic Inversion (NDI [8]). Ref. [18] describes various types of inverse model-based control laws and their automatic generation from Modelica models. The basic principle is illustrated in Figure 9.

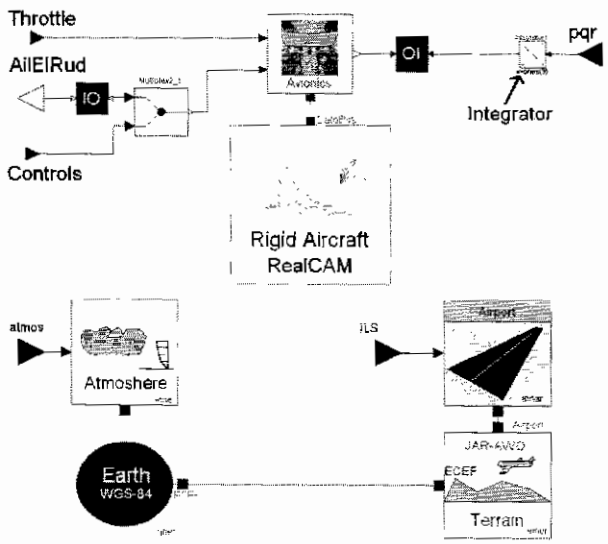

Figure 9. Reversal of inputs and outputs for inverse model generation

In the first place, the control inputs have been split into aileron, elevator and rudder (AilElRud), and other control surface deflections. The body angular rates are now defined as inputs (input connector top right), and aileron, elevator aud rudder are selected as outputs. The OI and IO objects (available in the "Utils" branch in Figure 7) overcome a protection in Modelica that forbids an external input resp. output connector to be connected to a component output resp. input connector.

When trying to generate simulation code, the model compiler will now issue an error, since the dynamic response between the control surfaces and the body angular rates (neglecting actuator dynamics) has a relative degree of 1, resulting in non-causal input-output relations (this information is provided with the error message). In order to generate a causal inverse model, the (in this case, first) time derivative of the inputs need to be available. This may be done by adding a first order filter, or as shown in Figure 9, an integrator (top right corner). This basically changes the type of the input into body angular accelerations, since the output of the integrator is connected with (i.e. set equal to) the aircraft body angular rates via the OI and Avionics components.

The model can now be translated. The inverse model may contain internal states and compute environment variables as before. However, in case the model is to be part of a control system, these states and environment variables should, as far as possible, be obtained from sensor measurements instead. The first can be realised using a simple compiler command (provided by Dymola), the latter is achieved by inserting environment model versions that, on request of other components, directly pass on measured signals. This explains why the environment models in Fignre 9 have additional inpnt connectors.

4.3. Desktop visualisation

Easy access to high-quality desktop visualisation tools becomes more and more important in the flight control law design process. This helps the engineer to better understand the (closed-loop) dynamics of the aircraft, and allows her or him to interactively "fly" the aircraft to qnalitatively assess control law performance and to find weaknesses before implementation in the full flight simulator [28]. End 2004 the engineering company AeroLabs AG [1] developed, in co-operation with the Technical University of Munich (Chair of Flight Mechanics and Flight Control), and based on specifications from the DLR Institute of Robotics and Mechatronics, a new visualisation tool for use with real-time desktop simulation of models (a.o.) constructed using the Flight Dynamics Library. As exclnsive features for DLR, a new high-quality on-line visualisation of airframe deformation was implemented, as well as the capability of visualising the aircraft and its environment using 3-D stereo projection. An application example is given at the end of the following section.

5. APPLICATION EXAMPLES

Since its first version in 1995, the Flight Dynamics Library has been applied in several projects at DLR, especially involving model development for design and evaluation of flight control laws. A number of these applications will be briefly discussed in this section.

GARTEUR Robust Flight Control
In the frame of the GARTEUR action group on Robust Flight Control, a generic military and civil aircraft model were implemented and nonlinear simulation code was generated for use in two flight control law design challenges [19]. At the end of the project, also so-called Linear Fractional Parametric Uncertainty Models (LF-PUM's) were automatically generated to allow for a symbolic worst-case analysis of developed control laws within the project, based on the structured singular value μ [14].

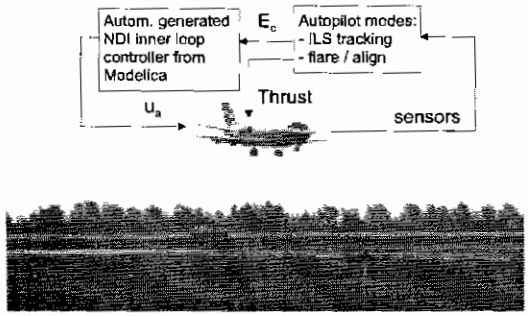

Figure 10. Automatic landing using automatically generated Nonlinear Dynamic Inversion (NDI) control laws

First Shot Approach in FCL design

In the frame of the German project "First Shot Approach in Flight Control Laws (FCL) design" (FSA) a complex model of a small transport aircraft was developed, including detailed engine and actuation system models [25].

AMANDA

During the HGF-project AMANDA (A multidisciplinary numerical definition and development system for aircraft, 1999–2001) a flexible aircraft model, based on public domain data, was implemented using the Flight Dynamics Library. The resulting model of a free-flying flexible aircraft could be flown interactively using a joystick. The model was coupled with the visualisation software AVDS (Aviator Visual Design Simulator [28]) in combination with a DLR-developed interface, showing the aircraft manoeuvering and deforming on-line during interactive flight.

REAL – Automatic Landing

In the frame of the EU-project REAL (Robust and Efficient Auto pilot control Laws design [30]) for the first time inverse model equations for a transport aircraft were automatically generated from the model implementation in Modelica. These inverse equations were used as the core of an automatic landing system that was developed iu the frame of this project [16]. The control laws were successfully flight tested on DLR's test bed ATTAS (Advanced Techuologies Testing Aircraft System [4]) during six automatic landings, see Figure 10.

X-31A with reduced vertical tail

The same procedure for automatic generatiou of Noulinear Dynamic Inversion control laws was applied to the thrust-vectored experimental fighter aircraft X-31A in the frame of the project VECTOR (Vectoring, Extremely short take-off and landing, Control, Tailless Operations Research [10]), in order to investigate reduced vertical tail coufigurations of this aircraft [31]. The control laws were exported from Dymola aud implemented in the ground-based flight simulator in Patuxent River, MD, USA, and successfully evaluated by five test pilots and one fleet pilot.

3D-Flexible aircraft flight simulator

From the example flexible aircraft model as presented in Section 2 simulation code was geuerated for use in interactive real-time simulation. The model was augmented with automatically generated Dynamic Inversion-based control laws, the automatic landing system as developed in the project REAL, as well as load alleviation control laws. The desktop flight simulator as mentioned in Section 4 hereby visualises aircraft flight dynamics and structural dynamics in real-time in very high quality, see Figure 11. The visualisation of structural deformation greatly helped to qualitatively assess performance of structural control laws.

Figure 11. 3D-stereo visualisation of aircraft flight and structural dynamics

6. SUMMARY AND OUTLOOK

In this paper an overview of the Modelica-based Flight Dynamics Library has been given. This library allows for intuitive construction of multi-disciplinary models for use in the aircraft and flight control laws design process. To this end, the library offers the following unique features:

- full compatibility with other libraries based on the Modelica language, allowing for development of truly multi-disciplinary aircraft models on a common modelling platform;
- intuitively structured models due to a physics-oriented break-down into model components and interactions;
- construction of rigid just as well as fully flexible aircraft models, including unsteady aerodynamic effects;
- easy implementation of multiple aircraft models usiug the same set of environment models (earth, atmosphere, etc.);
- automatic generation of efficieut simulation code for various engineeriug environuents (usiug a Modelica tool like Dymola);
- automatic generation of iuverse model code, e.g. for use in nonlinear control laws;
- automatic generation of trimming and linearisation scripts for use with the model in Matlab/Simulink;
- easy integration with desktop visualisation.

The Flight Dynamics Library as presented in this paper has undergone major restructuring as compared with previons versions, and exploits features of the latest Modelica language standard [24]. In the near future it will be completed with components from previous library versions that so far have not been included. Furthermore, one or more example models based on public domain data will be implemented and documented, allowing for commercial release of the library.

7. REFERENCES

[1] Aerolabs AG – Professional Solutions for the Engineering Industry: `http://www.aerolabs.de`.

[2] anon. Department of Defence World Geodetic System 1984 – Its Definition and Relationships with Local Geodetic Systems, Third Editiou, Amandment 1. Technical Report NIMA TR8350.2, National Imagery and Mapping Agency, Bethesda, MD, January 2000.

[3] J. Bals, G. Hofer, A. Pfeiffer, and C. Schallert. Virtual Iron Bird A Multidisciplinary Modelling And Simulation Platform For New Aircraft System Architectures. Iu *DGLR Luft- und Raumfahrtkongress 2005, Friedrichshafen, DGLR-Johrbuch 2005.*

[4] M. Bauschat, W. Mönnich, D. Willemsen, and G. Looye. Flight testing Robust Autoland Control Laws. In *Proc. of the AIAA Guidance, Navigation and Control Conference 2001, Montreol CA*, 2001.

[5] C.S. Buttrill, T.A. Zeiler, and P.D. Arbuckle. Nonlinear Simulation of a Flexible Aircraft in Maneuvering Flight. Iu *Proc. of the AIAA Flight Simulation Technologies Conference*, Monterey, CA, August 1987. AIAA-87-2501.

[6] H. Elmqvist. *A Struetured Model Language for Large Continuous Systems.* PhD thesis, Lund Institute of Technology, Lund, Sweden, 1978.

[7] H. Elmqvist. Object-oriented modeling and automatic formula manipulation in dymola. In *SIMS '93, Scandinavian Simulation Society*, Kongberg, Norway, Juue 1993. available from `http://www.dynasim.se/ publications.html`.

[8] Dale Enns, Dan Bugajski, Russ Hendrick, and Gunter Stein. Dynamic Inversion: An Evolving Methodology for Flight Control Design. In *AGARD Conference Proc. 560: Active Control Technology: Applications and Lessons Learned*, pages 7–1 – 7–12, Turin, Italy, May 1994.

[9] Christopher Fielding, Andras Varga, Samir Bennani, and Michiel Selier (Eds.). *Advanced Techniques for Clearanee of Flight Control Laws.* Lecture Notes in Control and Information Sciences 283. Springer Verlag, London, 2002.

[10] H. Friehmelt and P. Huber. Vector- Die X31A fliegt zu neuen bahubrechenden Technologiedemonstratioueu. In *DGLR Luft- und Raumfahrtkongress 2001, Hamburg 17.-20. Sept. 2001.*

[11] Joint Aviation Authorities Committee. Joint Aviation Requirements, JAR–AWO All Weather Operations. Technical report, JAAC, August 1996. Change 2.

[12] G.H. Kaplan. The IAU Resolutions on Astronomical Constants, Time Scales, and the Fundamental Reference Frame. Circular No. 163; Uuited States Naval Observatory; Washington, DC. December 10, 1981.

[13] Thiemo Kier and Gertjan Looye. Development Of Aircraft Flight Loads Analysis Models And Associated Uncertainties For Pre-Design Studies. In *Proc. of the International Forum on Aeroelasticity and Structural Dynamics (IFASD)*, Munich, Germany, June 2005.

[14] G. Looye, A. Varga, D. Moormann, and S. Bennani. Post-design stability robustness assessment of the RCAM controller design entries. Technical Report TP-088-35, GARTEUR, April 1997. available from `www.nlr.nl/hostedsites/ garteur`.

[15] Gertjan Looye. Integrated Flight Mechanics and Aeroelastic Aircraft Modeling nsing Object-Oriented Modeling Techniques. In *Proc. of the AIAA Modeling and Simulation Technologies Conference*, Portland, USA, August 1999. AIAA-99-4192.

[16] Gertjan Looye. Design of Robust Autopilot Control Laws with Nonlinear Dynamic Inversion. *at – Automatisierungstechnik*, 49(12), 2001.

[17] Gertjan Looye. Integration of Rigid and Aeroelastic Aircraft Models using the Residualised Model Method. In *Proc. of the Int. Forum on Aeroelosticity and Structural Dynomics (IFASD)*, Munich, Germany, June 2005.

[18] Gertjan Looye, Michael Thümmel, Matthias Kurze, Martin Otter, and Johann Bals. Nonlinear Inverse Models for Control. In *Proc. of the third internationol Modelica conference*, Hamburg, March 2005.

[19] Jean-François Magni, Samir Bennani, and Jan Terlouw (Eds). *Robust Flight Control – A Design Challenge.* Lecture Notes in Control and Informatiou Sciences 224. Springer Verlag, London, 1997.

[20] The Math Works Iuc. *MATLAB – External Interfaces Reference – Version 7*, March 2005.

[21] The Math Works Inc. *Simulink Reference*, 2005.

[22] McNeal-Schwendler Corp.. `http://www.mscsoftware.com`. On-line informatiou on NASTRAN.

[23] Modelica Design Group. Modelica: Lauguage design for multi-domain modeling. `http://www.modelica.org`.

[24] Modelica Design Group. Modelica - A Unified Object-Oriented Language for Physical Systems Modeling – Language Specification Version 2.2, February 2005.

[25] D. Moormann, P.J. Mosterman, and G. Looye. Object-oriented eomputational model building of aircraft flight dynamics and systems. *Aerospace Science and Technology*, 3(3), April 1999.

[26] Dieter Moormann. Physical modeling of controlled aireraft. In *Proc. of the CESA'96 IMACS Multiconference on Computational Engineering in Systems Applications*, pages 970–975, Lille-France, July 1996.

[27] Dieter Moormann. *Automatisierte Modellbildung der Flugsystemdynamik.* VDI Verlag, Reihe 8, Dissertation RWTH Aachen, Inst. für Flugdynamik, Düsseldorf, 2002.

[28] S.J. Rasmussen and S.G. Breslin. AVDS: a Flight Systems Design Tool for Visualization and Engineer-in-the-Loop Simulation. AIAA 97-3467. 1997.

[29] Christian Reschke. Flight Loads Analysis with Inertially Coupled Equations of Motiou. In *Proc. of the AIAA Modeling and Simulation Technologies Conference and Exhibit 2005, San Francisco CA*, 2005.

[30] W.F.J.A. Rouwhorst. Robust and Efficieut Autopilots control Laws design, demoustratiug the use of modern robust control design methodologies in the autoland system design process – the REAL Project. In *Proc. of the Aeronauties Days 2001, Hamburg Germany*, January 2001.

[31] R. Steinhauser, G. Looye, and O. Brieger. Design and Evaluation of Control Laws for the X-31a with Reduced Vertical Tail. In *Proc. of the AIAA Guidance and Control Conference*, Providence, Rhode Island, USA, August 2004.

[32] Brian L. Stevens and Frank L. Lewis. *Aircraft Control and Simulation.* Wiley-Interscience Publication. John Wiley & Sons, Inc., New York, 2000.

[33] Martin R. Waszak and Dave K. Schmidt. Flight Dynamics of Aeroelastic Vehicles. *Journal of Aircraft*, 25(6):563–571, June 1988.