IFASD 2005

FLIGHTDYNLIB: AN OBJECT-ORIENTED MODEL COMPONENT LIBRARY FOR CONSTRUCTING MULTI-DISCIPLINARY AIRCRAFT DYNAMICS MODELS

Gertjan Looye, Simon Hecker, Thiemo Kier, Christian Reschke

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

Keywords: Object-oriented modelling, Modelica, Flight mechanics, Aeroelasticity, Model integration, Simulation, inverse Models

Abstract. In this paper a model component library for developing multi-disciplinary aircraft flight dynamics models is presented, named FlightDynLib. This library is based on the object-oriented modelling language Modelica that has been designed for modelling of large scale multi-physics systems. The flight dynamics library allows for graphical construction of complex rigid as well as flexible aircraft dynamics models and is fully compatible with other available libraries for electronics, thermodynamics, control systems, etc.

Object-oriented modelling allows physical objects and phenomena to be implemented one-to-one into software objects, since interconnections may be defined freely to directly represent the physical interactions (e.g. kinematic constraints, energy flows). Once a model is finished, dedicated symbolic algorithms collect, sort, and solve all model equations according to the selected inputs and outputs. From the solved equations efficient runtime code for model simulation is generated. This procedure may be used to obtain regular as well as inverse models, e.g. for trim computation or inversion-based control laws. Also dedicated representations for use in design of flight control laws can be generated, like symbolic parametric models for robustness analysis.

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 a model, and generation of simulation code will be demonstrated. The efficiency of the code results in short simulation times and allows for real-time applications.

I. 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. Most involved engineering disciplines develop models for their specific types of analysis. However, it has long since been recognised that considerable performance gains may be achieved 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. For cost reasons hardware rig testing and flight testing is more and more replaced with simulation. For this accurate dynamic models are need, especially in order to reliably predict the behaviour of the integrated aircraft (systems) in flight.

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. Also the translation of model components from the one environment (modelling formalism) into the other is sometimes applied. On

the other hand, general block diagram-based simulation tools (e.g. Matlab-Simulink) are used in more and more disciplines, allowing direct integration of models in one and the same simulation environment.

Tool- or code-based integration of simulation models are pragmatic approaches, but have a number of drawbacks. Coupling of tools is unlikely 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 model 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. Although the graphical possibilities of block diagram-oriented tools give considerable relief, being forced to formulate model components as ordinary differential equations with specific inputs and outputs may still require considerable effort to grasp the physics behind the model. The above will be illustrated on an aeroelastic example in Section II.

It is for these reasons that in the last decades several research groups have started developing dedicated physical modelling languages and tools. The principal feature of these languages is that the user is not forced to specify solved model (differential) equations and 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. discipline-specific modelling paradigms can still be used, but based on a common language base.

The physical languages are strongly supported by graphical tools. The second advantage for example allows that an electronic circuit still looks as such, and that a control law may still be implemented in the form of a block-diagram.

In 1996 several developers in the physical modelling field initiated the specification of a free common language standard, named *Modelica*. To this end, the non-profit Modelica organisation was founded (see `http://www.modelica.org`). By now, the language has achieved a high degree of maturity and several modelling tools ("Modelica compilers") have become available, featuring graphical modelling composition 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) 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.

Back in 1994 the DLR institute of Robotics and Mechatronics developed a first library (based on one of the predecessor languages of Modelica) for modelling of aircraft flight dynamics.[15] 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 model[5,6] and a first flexible aircraft shortly thereafter.[9] Since then, the library has been expanded and applied to complex aircraft models that include system hydraulics and electronics.[14] 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.

This paper discusses the latest developments, features, and example applications of the Flight Dynamics Library (FlightDynLib), with emphasis on flexible aircraft. In the following section a simple aeroelastic example is discussed in order to explain the basic principles

of physical or object-oriented modelling. Then the selection of a generic aircraft model structure is discussed (Section III), based on which the flight dynamics library has been organised (Section IV). In Section V the automatic generation of model code for model simulation and analysis is discussed, followed by some example applications. Finally, a summary is given in Section VII.

II. An aeroelastic example

CONSIDER the following generalised structural equation of motion:

$$M\ddot{\eta} + B\dot{\eta} + K\eta = Q \tag{1}$$

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:

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

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 1. The $\frac{1}{s}$ blocks are in-

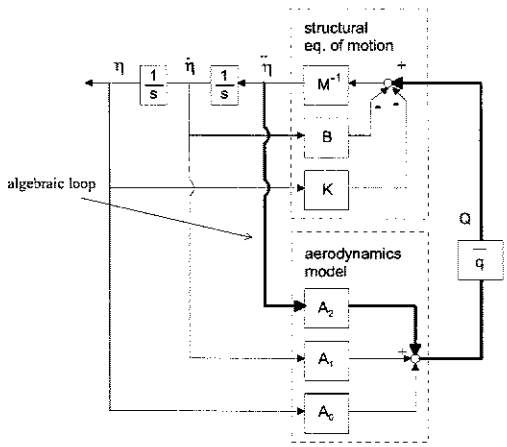

Figure 1. Implementation of the aeroelastic system using block diagram modelling

tegrators. 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:

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

Realizing this equation in a block diagram makes little sense, since structural and aerodynamics are mixed.

Besides manual derivation and coding (either via programming or by composing a block diagram), a third form of implementation is illustrated in Figure 2. The system is represented

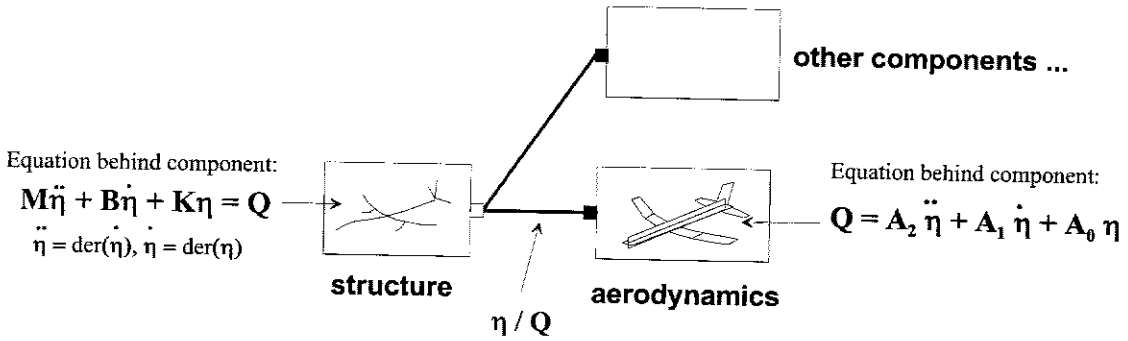

Figure 2. Implementation of the aeroelastic system using physical modelling

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 (separated by the "/") is handled as a "through" variable (i.e. summed up to zero between components). Note that addition of any other components (e.g. a propulsion model) is very straightforward, since the very same interconnection principle applies.

The implementation in Figure 2 illustrates the basic principle of physical modelling. The contents of the blocks consist of the physical modelling equations, as well as declarations of internal variables and parameters. Figure 3 illustrates the implementation of the aerodynamic model equations in the physical modelling language Modelica, introduced in Section I. Its graphical representation is captured in annotations, which have been left out here. The full specification of the Modelica language, as well as numerous references may be downloaded from `http://www.modelica.org`.

III. The basic structure of aircraft and environment models

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.

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 4 has been adopted. It consist of one (or multiple) aircraft, and environmental objects. The latter 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 mutual interactions, like towed gliders, wake vortices, air-to-air refuelling, releasing missiles, etc.

The earth model in Figure 4 has the following functions:

1. Provide the inertial reference. To this end, an Earth-Centered Inertial (ECI) is used. Its origin is attached to the Earth's center of mass, its orientation is fixed with respect to "fixed" reference stars.[7]

```
model aerodynamics

   /* Declarations:  */

   constant SI.Length cref =  10; /* Reference length, declared as Real,
                                     with SI unit 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 3. Modelica code behind example aerodynamics block in Figure 2

2. Provide the geodetic reference. As indicated in Figure 4, to this end the World Geodetic System 1984[1] (WGS-84) is used. The object implements an Earth-Centered Earth-Fixed (ECEF) reference frame, which has the same origin as the ECI, but rotates with the earth. Position and attitude of the ECEF are 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 equi-potential surfaces a function of longitude and latitude. Currently, a more simplified height and geocentric latitude-dependent (Ref.[17] - eqn.(1.4-16)) and a constant gravity model are available.

Double-clicking on the Earth 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

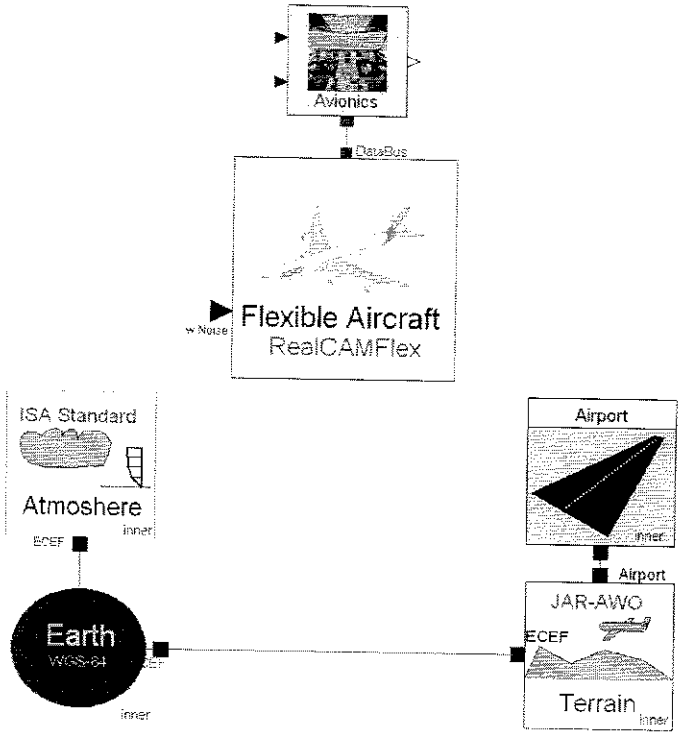

Figure 4. Top-level of model: aircraft and environment

many applications, but provides sufficient generality for use with 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.

The second environmental object in Figure 4 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 (local) earth surface. A simple earth boundary layer model logarithmically reduces the wind velocity to zero on the ground.

To the right in Figure 4 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 here is a terrain model as used for automatic landing control law design and certification, based on JAR-AWO specifications. 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.

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 runway as positioned in the JAR-AWO terrain model is included. 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.

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

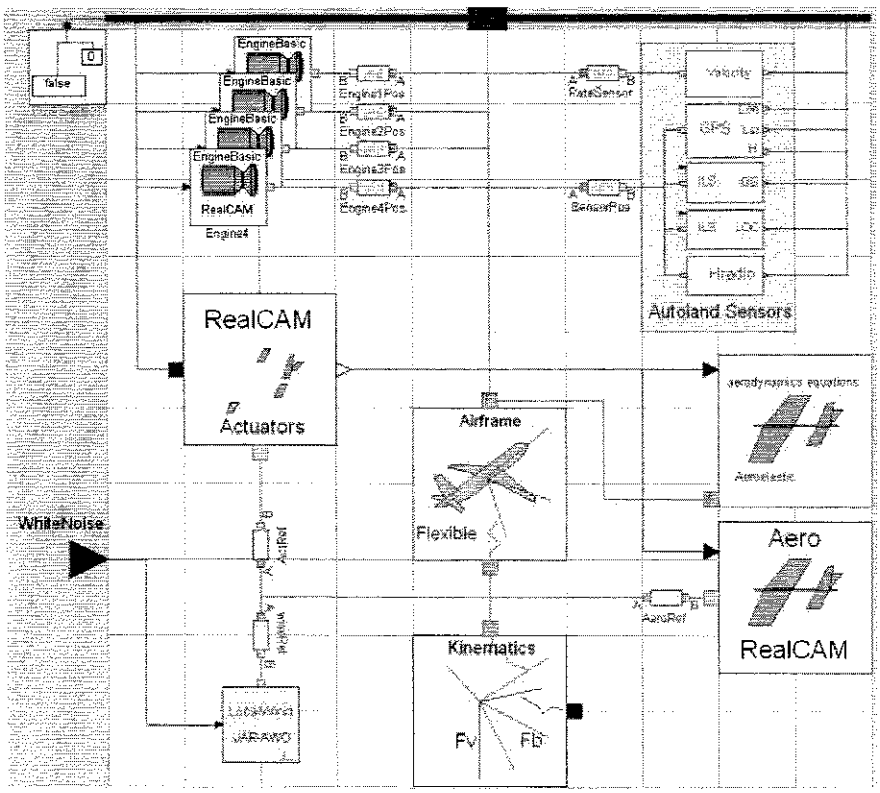

Figure 5. Structure of a flexible aircraft model

flexible ones. A typical model structure for a flexible transport aircraft is shown in Figure 5. The specific example was originally used in the EU project REAL (Robust and Efficient Autopilot control Laws design), 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.

The backbone of the model depicted in Figure 5 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 center of gravity. The object also defines a 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, right: NED reference frame). 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. 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 in the (undeformed) airframe may be more useful for referencing reasons. The local gravity acceleration is obtained by a call to the Earth object (Figure 4). 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.[3,18] The body axes system is hereby considered as a so-called mean reference 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 5) makes that the reference systems in both connectors merge, i.e. 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). 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 through variables (see Section II).

The airframe equations of motion are primarily driven by aerodynamic and propulsion forces and moments. These are computed in corresponding model components in Figure 5. 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 4.

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 5). 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 5 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 corrected for quasi-steady deformation of the airframe. The underlying model may be based on complex application rules, table look-ups, etc. The upper aerodynamics component computes unsteady (generalised) forces and moments as induced by flexible deformation of the airframe. These are corrected by removing quasi-steady effects, in this case, using the so-called residualised model method.[11] The unsteady aerodynamic data are read from a user-specified 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.

The engine models (top left) are connected to the airframe via a slightly different type of transformation. Instead of an offset, the number 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 for the airframe reference and its local deformation.

The very same principle applies to the sensor models that are in the top-right corner of Figure 5. 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 of 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 4), 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.

As already discussed at the beginning of this section, mean winds are computed in the atmosphere block at the top level of the model Figure 4. 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.

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 electrics 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.

Finally, the fat bar at the top of Figure 5 represents a so-called data bus. This bus includes signals that one would typically find on avionics buses in 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 and allows direct connection to elements from the Modelica block diagram library. This enables a control system composed using this library to directly communicate with the aircraft data bus.

IV. The DLR Flight Dynamics Library

THE top-level structure of the Flight Dynamics Library (FlightDynLib) is depicted in Figure 6. The "Modelica" branch in the depicted tree (top) contains the principal standard libraries delivered with Modelica (e.g. for block diagrams, electronics, etc.). 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 allow the user to develop his own. Each aircraft type or family namely tends to have its own 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.
- **Environment** contains all environment-related models as described at the beginning of the previous section.
- **Examples** contains actual implementations of aircraft models, as for example depicted in Figure 4 and 5.
- **Gear** currently contains a highly simplified landing gear model for which basic properties may be set and which may be attached to the Airframe object in Figure 5. Detailed models may be composed using among others the multi-body library by specialists in the field.
- **Interfaces** contains all library-specific connector types, as well as the data bus that was discussed at the end of the previous section.

- **Kinematics** contains the various types of models describing the kinematics of the body reference system with respect to the local NED, as well as the inertial reference. Versions with e.g. constrained degrees of freedom are available as well.
- **Propulsion** contains, 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 transformations between reference systems.
- **Types** contains some internal variable type definitions to which the user may add his own.
- **Utilities** contains miscellaneous functions e.g. for reading external data.

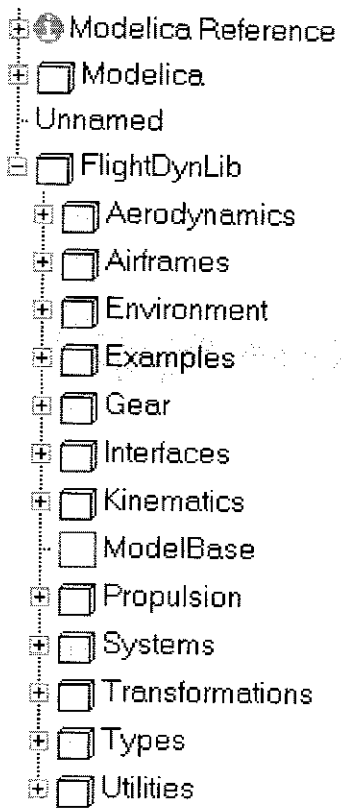

Figure 6. Top-level structure of the Flight Dynamics Library

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 7. 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 can be specified (e.g. truncation or residualisation).

V. Code generation for model application

AFTER model composition has finished, a model compiler sorts and solves all model equations according to specified inputs and outputs into Ordinary Differential Equa-

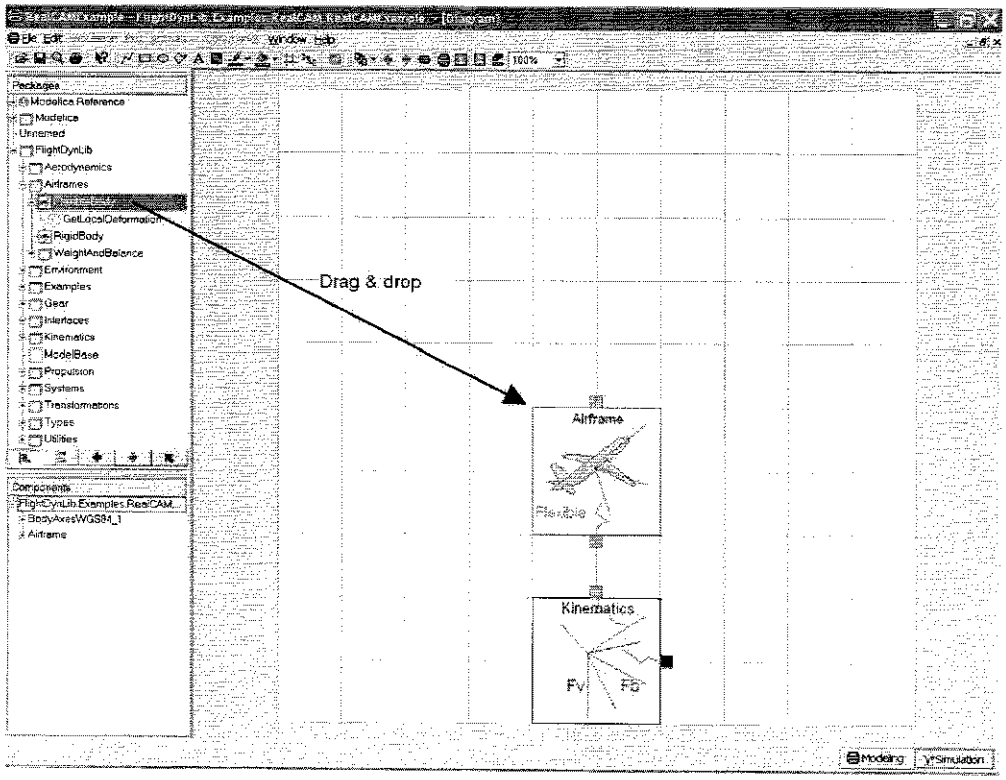

Figure 7. Graphical aircraft model composition using drag and drop from the library

tions (ODE's) or Differential Algebraic Equations (DAE's), suitable for use in a simulation environment. The modelling tool Dymola features, besides a graphical modelling environment and symbolic algorithms, extensive simulation and data analysis capabilities. However, the simulation may be used in other engineering environments and simulation tools as well, like for example Matlab-Simulink.

A simple way of specifying model inputs and outputs is illustrated in Figure 8. Here a so-called Avionics block has been connected to the bus connector of the aircraft (in this case a rigid one). At the main model level, also input and output connectors have been defined. The Avionics block injects pilot throttle and control surface input commands (from Thrust, 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").

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 (e.g., for given steady state flight condition parameters, compute corresponding control surface and throttle settings), as well as for automatic generation of control laws that are based on inverse model equations, like Nonlinear Dynamic Inversion (NDI[4]). Ref.[12] describes various types of inverse model-based control laws and their automatic generation from Modelica models. The basic principle is illustrated in Figure 9.

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 and rudder are selected as outputs. The OI and IO objects (available in the Utils branch in Figure 6) 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,

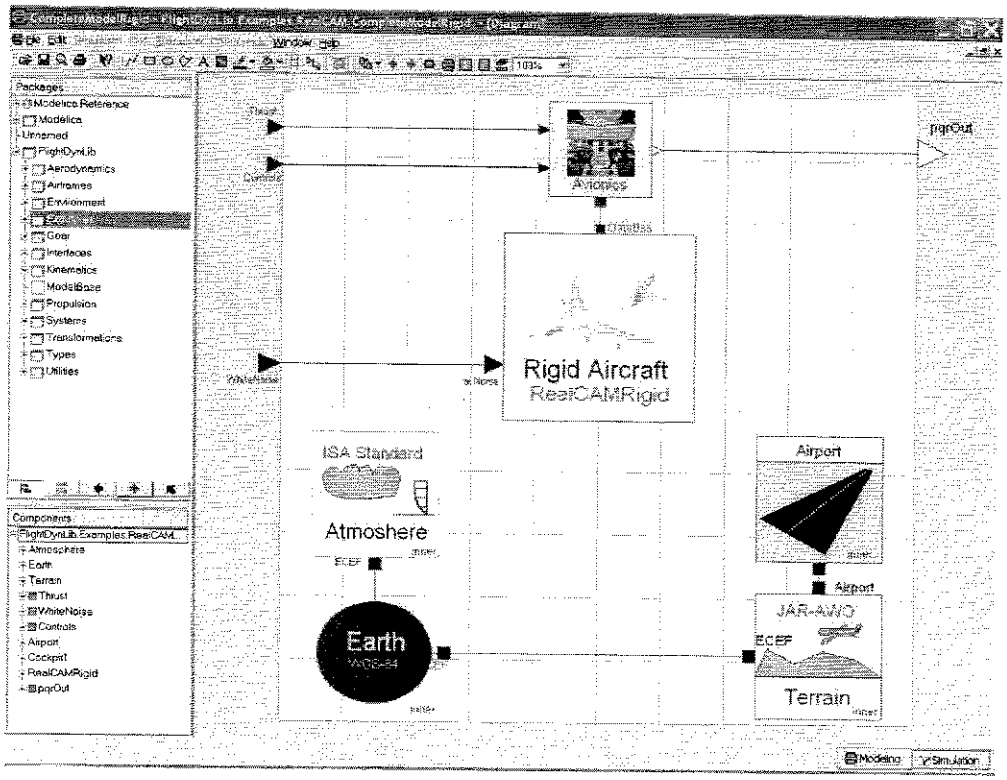

Figure 8. Specification of model inputs and outputs at the hierarchical top level

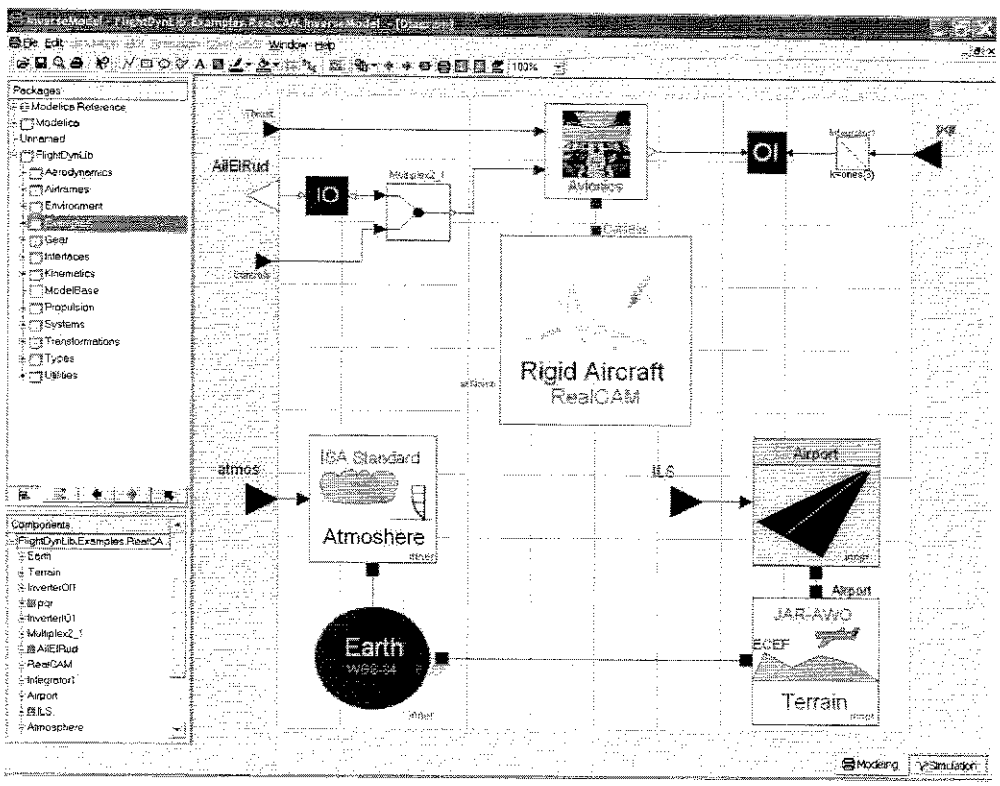

Figure 9. Reversal of inputs and outputs for inverse model generation

since the dynamic transfer between the control surfaces and the body angular rates (with not actuator dynamics) has a relative degree of 1 (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 input into body angular accelerations, since the integrator output is connected with the 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 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 Figure 9 have additional input connectors.

VI. Application examples

SINCE its first version in 1994, 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.

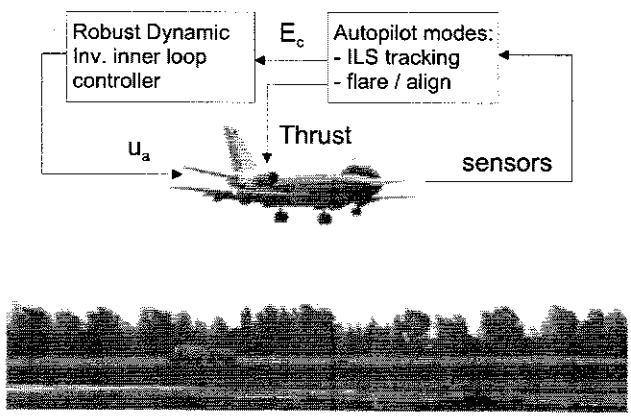

Figure 10. Automatic landing using automatically generated Nonlinear Dynamic Inversion control laws

- In the frame of the GARTEUR action group on Robust Flight Control, a generic military and civil aircraft model were implemented and simulation code was generated for use in two flight control law design challenges.[6,13] At the end of the project, 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.[8]

- In the frame of the German project "First shot approach in flight control laws design" a complex model of a small transport aircraft was developed, including detailed engine and actuation system models.[14]

- In the frame of the EU-project REAL (Robust and Efficient Auto pilot control Laws design) 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 in the frame of this project.[10] The control laws were successfully flight tested on DLR's test bed ATTAS (Advanced Technologies Testing Aircraft System[2]) during six automatic landings, see Figure 10.

- The same procedure for automatic generation of Nonlinear Dynamic Inversion control laws was applied to the thrust-vectored experimental fighter aircraft X-31A, in order to investigate reduced vertical tail configurations of this aircraft.[16] The control laws were exported from Dymola and implemented in the ground-based flight simulator in Patuxent River, MD, USA, and successfully evaluated by five test pilots.
- The example as presented in Section III has been implemented in a Real-time modelling environment with 3D visualisation of the aircraft, its environment, as well as structural deformations, see Figure 11. The model was augmented with inversion-based control laws as well as an automatic landing system and load alleviation control laws.

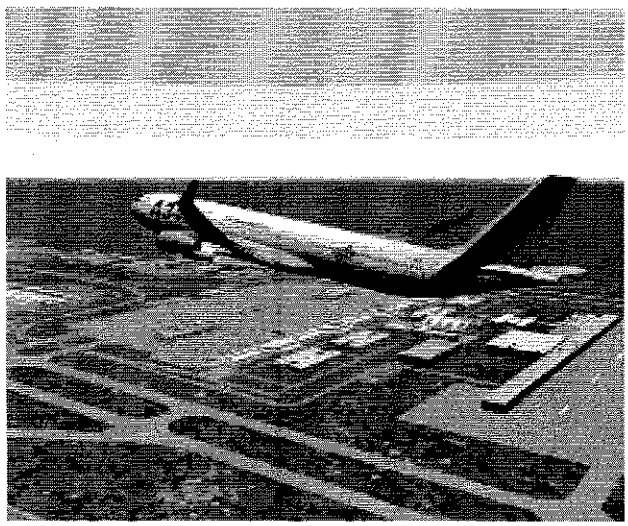

Figure 11. 3D-stereo visualisation of aircraft flight and structural dynamics

VII. Summary

IN this paper an overview of the Modelica-based DLR Flight Dynamics Library was given. This library allows for intuitive construction of multi-disciplinary aircraft models, including for example unsteady aeroelastic effects. The basic lay-out of aircraft models is highly physically oriented, greatly enhancing model visibility. The library is compatible with other Modelica libraries, allowing for development of true multi-domain models, based on one and the same modelling platform.

The symbolic algorithms and code generation capabilities of the modelling environment Dymola (a Modelica "compiler" and simulator) allow for generation of forward and inverse simulation models. The generated runtime model code may be used in various simulation tools.

References

[1]anon. Department of Defence World Geodetic System 1984 – Its Definition and Relationships with Local Geodetic Systems, Third Edition, Amandment 1. Technical Report NIMA TR8350.2, National Imagery

and Mapping Agency, Bethesda, MD, January 2000.

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

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

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

[5]Flight Mechanics Action Group 08. Robust Flight Control Design Challenge Problem Formulation and Manual: the High Incidence Research Model (HIRM). Technical Report TP-088-04, GARTEUR, April 1997. Version 3.

[6]Flight Mechanics Action Group 08. Robust Flight Control Design Challenge Problem Formulation and Manual: the Research Civil Aircraft Model (RCAM). Technical Report TP-088-03, GARTEUR, April 1997. Version 3.

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

[8]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`.

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

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

[11]Gertjan Looye. Integration of Rigid and Aeroelastic Aircraft Models using the Residualised Model Method. In *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics (IFASD)*, Munich, Germany, June 2005.

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

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

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

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

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

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

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