AIAA-99-4192

Integrated Flight Mechanics and Aeroelastic Aircraft Modeling using Object-Oriented Modeling Techniques

Gertjan Looye*

German Aerospace Center
DLR-Oberpfaffenhofen
Institute for Robotics and System Dynamics
D-82234 Wessling, Germany
E-mail: gertjan.looye@dlr.de

Abstract In this paper a software implementation is proposed of integrated flight mechanics / aeroelastic aircraft models, using object-oriented modeling techniques. Model development involves integrating a rigid aircraft simulation model with aeroelastic flutter analysis models. The main application is primary flight control law design, in which most of the criteria are of flight mechanical nature. For this reason, the rigid aircraft model is taken as the basis, while the fidelity of the aeroelastic part may depend on the accuracy required.

Object-oriented modeling allows physical objects and phenomena to be implemented one-to-one into software objects, since interconnections can be defined freely (e.g. according to physical interactions like energy flow). This feature facilitates integration of model components from different engineering disciplines. A model compiler generates simulation code by symbolic manipulation of the model equations. The code can be exported to several (simulation) languages. This allows the same model to be used in different engineering environments. We illustrate this feature with a flutter analysis example.

1. Introduction

This paper proposes a software implementation of integrated flight mechanics / aeroelastic aircraft models, using object-oriented modeling techniques. The main model applications are simulation and flight control law design.

For slender airframe designs, like large civil transport aircraft, structural flexibility may considerably influence the flight dynamics. For this reason, aeroelastic effects have to be accounted for in the aircraft simulation model. Traditionally, flight control design models only contain rigid body dynamics and the design criteria are mostly of flight mechanical nature (e.g. handling qualities criteria). Therefore, we prefer a so-called 'bottom-up' approach in the model integration. This means that a full nonlinear rigid body simulation model is taken as the basis, and extended with an aeroelastic part of 'scalable' fidelity (e.g. number of modes, detail of unsteady aerodynamics taken into account, order reduced), derived from detailed flutter models.

In the first place, we need the equations of motion for flexible aircraft as a 'spine' for interconnecting the different model components. These equations are derived in refs [12, 20, 4].

Given the equations of motion, we can start thinking how to interconnect models that actually 'drive' these equations, like the aerodynamic model and thrust models. In this paper we will concentrate on implementation and interconnection of these submodels, rather than the modeling itself.

The total aircraft model consists of many submodels from different engineering disciplines. This is

[*] Research Engineer

an important reason why we use the object-oriented modeling language Dymola (DYnamic MOdeling LAnguage[1]) [5]. Moormann [14, 13] successfully implemented a rigid aircraft library using this language. This paper describes the further development to flexible aircraft. The Dymola language allows one-to-one implementation of physical objects and phenomena into software objects, because of freedom in defining the object interconnections. These interconnections are not limited to signal flows (like in most control-oriented graphical simulation tools) but represent physical system interactions, like energy flows, or kinematic constraints. The objects are defined and interconnected using a graphical interface. Model component libraries can be developed to allow easy re-use in other models [14, 13]. These libraries may contain different objects for a same sub-model with different levels of detail. These objects can be easily exchanged within the aircraft model. The modeling language is equation based, i.e. the model components can simply be written in the form of equations, without the need of 'bringing unknowns to the left-hand side'.

A model developed in Dymola is further processed using the DYnamic MOdeling LAboratory (same acronym: Dymola). The model compiler collects and sorts the equations from the objects and using symbolic algorithms generates simulation code in Ordinary Differential Equation (ODE) (or Differential Algebraic Equation (DAE)) form. As this is normally done by hand, this feature saves a lot of error-prone work. Finally, the simulation code can be exported as C, Fortran, or ACSL code, or directly to environments such as MATLAB/Simulink [11] and Matrix$_x$/SystemBuild [7]. This gives great flexibility in application of the model.

This paper is structured as follows. In Section 2 we review the equations of motion for flexible aircraft. In section 3 we briefly discuss aerodynamic model integration. Next we describe the implementation of an example model in Dymola. In section 5 we discuss code generation. In section 6 we present flutter analysis as an example application of the model. We will end with conclusions in section 7.

2. Equations of motion

Flight mechanics models are based on the nonlinear Newton-Euler equations of motion for a rigid body [3]. These equations describe the motion of a rigid aircraft with six-degrees of freedom, driven by thrust, aerodynamic and gravity forces. In the references [12, 20, 4] the equations of motion are extended to flexible aircraft. Ref. [4] presents a detailed derivation and shows how structural and rigid dynamics can be integrated in a mechanically correct way. The reader is referred to these references for a detailed discussion. We will only mention the most important assumptions made and cite the final equations from [20].

The three basic assumptions we make are (1) a reference system fixed relative to the earth surface is an inertial system; (2) structural deformations are sufficiently small so that linear elastic theory applies; (3) a set of vibration modes with eigen frequencies is available from e.g. finite element analysis. Figure 1 shows a free flying aircraft considered as a large set of lumped-masses dm_i (assumption 4), of which one is shown in the figure. The axes x_g, y_g, z_g form the inertial reference frame. The axes x_b, y_b, z_b form the body axes that move with the airframe. The inertial position of the origin is R_0. The angular rates of the body axes are $\Omega = [p,\ q,\ r]^T$, while the Eu-

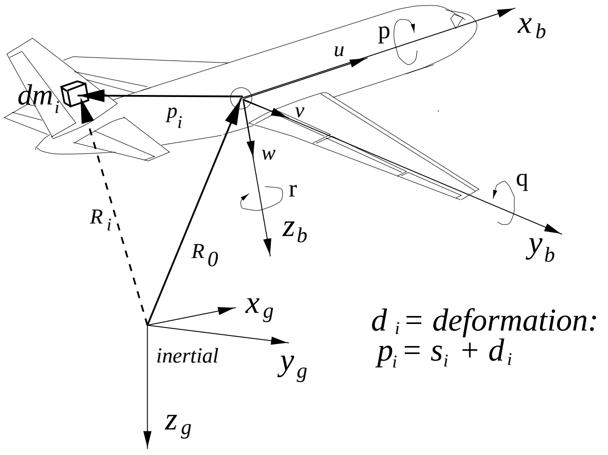


Figure 1: Flexible Aircraft

[1] to be updated to the Modelica standard in near future [6]

ler angles are $E = [\phi,\ \theta,\ \psi]^T$ (not depicted). The components of the inertial velocity vector along the body axes are $V = [u,\ v,\ w]^T$. The position of the mass element dm_i is p_i, consisting of its undeformed location s_i and its deformed translation d_i. The local deformation is written as a linear combination of mode shape deformation vectors Φ_{ij}, where j refers to the j^{th} elastic mode:

$$d_i = \sum_{j=1}^{N_e} \Phi_{ij} \eta_j \tag{1}$$

N_e is the number of modes taken into account and η_j is the generalized displacement of mode j. In refs[20, 4] the equations of motion are derived using Lagrangian mechanics. The most important issue is the choice of an appropriate body reference system. In all references, so-called *mean axes* are considered as the best choice, since this results in minimal inertial coupling between rigid and elastic dynamics. The orientation of mean axes is such that deformation induced translational and rotational momentum are always zero. In reality, these constraints are hard to satisfy, but with the assumption (3) that the deformations are small, and that deformation and deformation rates are co-linear, (assumption 5) it is sufficient when so-called *practical* mean axes constraints [20] are satisfied. If the mode shapes have been obtained from free-free finite element analysis, then a reference system with the axes in the direction of the rigid modes, and with the origin always in the center of gravity, satisfies these practical mean axes constraints.

Finally, we make the assumption that the inertia tensor is constant, although it will vary slightly due to deformation of the airframe (assumption 6). The resulting equations of motion are as follows:
* Kinematic relation R_0 and V:

$$\dot{R}_0 = R_{bg}^T(E)\ V \tag{2}$$

where R_{bg} is the rotation matrix from the inertial to the body mean axes.
* Kinematic relation E and Ω:

$$\dot{E} = R_{\phi b}(E)\ \Omega \tag{3}$$

where $R_{\phi b}$ is the transformation matrix from body angular rates into Euler angular rates.
* Rigid body force equation:

$$m[\dot{V} + \Omega \times V - R_{bg}(E)\ [0,\ 0,\ g]^T] = F_A + F_T \tag{4}$$

where F_A and F_T are aerodynamic and thrust force vectors respectively, and g is the gravity acceleration.
* Rigid body moment equation:

$$I\,\dot{\Omega} + \Omega \times I\Omega = M_A + M_T \tag{5}$$

where M_A and M_T are aerodynamic and thrust moment vectors respectively.
* Elastic degrees of freedom: $(j = 1...N_e)$

$$M_j(\ddot{\eta}_j + 2\zeta_j\omega_j\dot{\eta}_j + \omega_j^2\eta_j) = Q_{\eta_j} \tag{6}$$

where M_j is the generalized mass matrix, ζ_j the structural damping, ω_j the eigenfrequency, and Q_{η_j} the generalized aerodynamic force for mode j respectively.

The only coupling between rigid and flexible motions is via the aerodynamic forces and moments. Inertial coupling disappears through the assumptions made (5,6). Dropping these assumptions, the equations become considerably more complicated [4].

For flexible aircraft, the kinematics of local points on the airframe with respect to the mean axes are more complicated than for the rigid case. We need to know these local kinematics at the positions where we want to interconnect for example engine and sensor models. As we see from fig. 1, the kinematics of a local point are described by the movements of the mean axes (R_0), the undeformed location w.r.t. the mean axes (s_i), and the local airframe deformation (d_i, from eq. 1).

For our implementation (section 4) we will make use of multi-body principles. At the airframe locations where we intend to attach e.g. engine or sensor models, we define a local reference system which, contrary to the mean axes, is physically attached to the airframe. In undeformed position the orientation of the local and the mean axes are the same. Along the local body axes, we compute inertial kinematics of the local point (position, orientation, (angular) velocity, (angular) acceleration), and forces and moments. For example, the relation between absolute and relative velocities of the local and mean axes are given by:

$$\dot{d}_i = R_{bl} v_{i_l} - V + p_i \times \Omega \tag{7}$$

where the velocity of point i relative to the mean axes is:

$$\dot{d}_i = \sum_{j=1}^{N_e} \Phi_{ij} \dot{\eta}_j$$

v_{i_l} is the inertial velocity of point i along the local axes (l), R_{bl} is the time dependent rotation matrix from the local into the mean axes. This matrix depends on the local angular displacement vector ϕ_i. We will not cite the equations for acceleration here, but it will be clear that differentiating eq. 7 leads to a complicated expression [15].

An engine or sensor model now, is defined in its own reference system. Connecting a model to the airframe means that its local axis system and the local axis system defined on the airframe merge, i.e. kinematics of both axis systems are constraint to be equal. A vertical acceleration sensor for example, is modeled such that it outputs the acceleration along its vertical z-axis. Once interconnected with the airframe model, this acceleration is equal to the inertial acceleration in z-direction of the locally defined reference system on the airframe. As we will show in section 4, we can directly implement this interconnection procedure, without need for reordering equations or variables to sort out unknown variables; Dymola will do this automatically when generating simulation code.

3. Aerodynamic model

Although not exercised for the model example presented in this paper (since data was readily available), a major integration effort in developing an integrated flight mechanics/aeroelastic aircraft model is required in development of the aerodynamic model. Therefore we will briefly discuss some key aspects of the aerodynamic modeling.

Both for flight mechanic and aeroelastic models 'agreed-upon' aerodynamic modeling methods exist. In rigid models so-called stability and control derivatives are used, usually implemented in polynomial or tabular form. These derivatives are obtained from for example handbook methods, CFD computations, wind tunnel experiments, and flight testing. If necessary, flexibility of the airframe is accounted for using so-called rigid/flex static correction ratios that scale the derivatives.

The computation of aeroelastic forces and moments is highly complicated. A trade-off is necessary between accuracy and computational costs. For this reason the Doublet Lattice method[1] is a popular method in industry to compute aeroelastic models in sub-sonic flow regimes for flutter analysis. For our proposed model integration, we use those flutter models to develop the aeroelastic part of the aerodynamic model.

Integration of rigid and the aeroelastic aerodynamic models is for example discussed in refs [2, 22, 9]. The model consists of four parts:

$$\begin{bmatrix} Q_R \\ Q_\eta \end{bmatrix} = \begin{bmatrix} \mathrm{RR}(\underline{x}_a, \dot{\underline{x}}_a, ...) & + & \mathrm{RF}(\eta, \dot{\eta}, ...) \\ \mathrm{FR}(\underline{x}_a, \dot{\underline{x}}_a, ...) & + & \mathrm{FF}(\eta, \dot{\eta}, ...) \end{bmatrix} \tag{8}$$

where $Q_R = [F_A,\ M_A]^T$ are forces and moments driving rigid dynamics, $\underline{x}_a$ denotes rigid body aerodynamic states, such as airspeed V_A, angle of attack α and sideslip angle β. η is the vector with generalized displacements of the elastic modes. Of course the model depends on other variables as well, such as control inputs. RR (Rigid-Rigid) and RF (Rigid-Flex) describe the forces and moments driving the rigid part of the equations of motion, induced by rigid and flexible aircraft states respectively. FR (Flex-Rigid) and FF (Flex-Flex) describe the generalized forces driving the flexible equations of motion, induced by rigid and flexible aircraft states respectively.

The RR part is the same as in the rigid aircraft model, whereas the three other components are to be obtained from flutter models. It is generally known that the computation of generalized forces induced by rigid modes in flutter models is poor, but up to now we did not look into other modeling methods.

Flutter models are obtained in the frequency domain and therefore need to be transformed into the time domain. Several approaches have been developed that are based on rational function approximations in the Laplace domain[16, 8]. Ref. [17] compares several methods and concludes that Karpel's method[8] gives the most accurate results with the lowest number of additional aerodynamic lag states.

Rigid aerodynamic models implemented in look-up tables are usually valid over (a part of) the flight envelope, whereas flutter models only have very local validity. In practice, these models are computed over a grid of flight conditions and aircraft configurations. It is very difficult to parametrize these models as a function of such variables. Therefore, the most obvious way to increase validity of the aeroelastic parts in the aircraft model is interpolation between a set of locally derived aeroelastic models.

Finally, both the rigid and the aeroelastic parts of the aerodynamic model contain static aeroelastic information. In the rigid model this is represented by the rigid-flex ratios. In order that the trim state vector does not depend on the number of modes that is included, the refs. [22, 9] use a procedure developed by Adams at NASA Langley, in which the static influence is subtracted from the RF part of the aerodynamic model. Static aeroelastic deformation is represented by the rigid-flex ratios only.

4. Model implementation

Figure 2 shows the basic implementation of an example flexible aircraft model in the Dymola graphical interface. The aircraft model has been taken from ref. [21]. It has one asymmetrical mode and four symmetrical modes (see also ref.[20]). The quasi-steady aerodynamic model was derived using strip-theory.

The central object *flexbody* contains the equations of motion as discussed in section 2, and since all other objects are connected to it, it has indeed the function of a spine for the model. In addition two engine models are visible (the aircraft has four engines; each model actually represents a pair of engines on each wing), as well the aerodynamic, gravity, atmospheric, wind and systems (actuators for aerodynamic control surfaces) sub-models. In each object a local reference system is defined in which the sub-model is described.

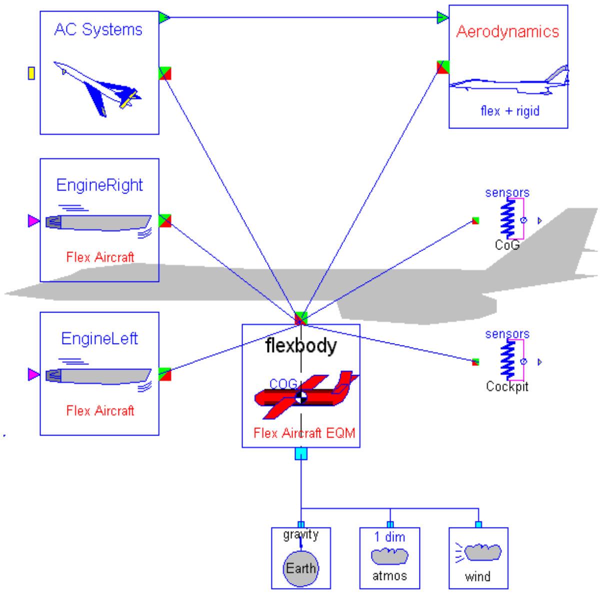


Figure 2: Model implementation (top level)

All objects have connection points to other objects. These connection points will be referred to as 'cuts'. Connections between cuts represent physical relations between sub-models, in this case kinematic constraints and energy flows. In these cuts the following variables are defined:

Forces and moments – F_l, M_l (force and moment vector along local reference system of the sub-model), Q_η (generalized aeroelastic forces).

Kinematic quantities – R_{gl} (rotation matrix from local into inertial axes), R_0 (inertial position vector of local axes origin), V, Ω (translational and rotational inertial velocities of the local axis system, described in local axes), a, α (translational and rotational inertial accelerations of the local axes), η, $\dot{\eta}$, $\ddot{\eta}$ (mode shape generalized coordinates and derivatives). Also environmental quantities are defined, but we will not discuss these. When connecting two cuts, the kinematic quantities in both cuts are set equal, so that the objects are forced to move exactly the same way. The forces and moments on the other hand, are summed to zero (can be interpreted as a local equilibrium condition).

Flexbody object

The *flexbody* object has two cuts: the variables in the lower cut are in earth fixed axes, in the upper one in body mean axes. Wind, atmosphere and gravity are modeled in earth fixed axes, and are therefore interconnected via the lower cut. The other objects move with the aircraft and drive the equations of motion via the mean axes. These are therefore interconnected via the upper cut. The kinematic relations between earth fixed and mean axes are part of the equations of motion, see section 2.

Figure 3 zooms in on the *flexbody* object (by right-clicking on it with the mouse in the graphical interface). The lower right block (*eqm*) contains the rigid equations of motion, entered in exactly the same form as in eqs. 2, 3, 4, 5. Again, the variables in the lower cut are in earth axes (inertial), whereas those in the upper cut are in body axes (6 DOF). The object to the left (*mass*) defines and computes mass dependent variables (empty aircraft weight, fuel weight, inertia tensor). The top-right object (*Flexible Modes*) contains the flexible equations of motion (eq. 6) in mean axes. By interconnecting the *eqm* and *Flexible Modes* objects, the mean axes merge with the body axes in the first object, i.e. the mean axes become a 6 DOF reference system. The variables in the upper cut of the second object are also in mean axes, but herein also generalized mode shape coordinates and derivatives (η, $\dot{\eta}$, $\ddot{\eta}$) are defined, as well as generalized aeroelastic forces (Q_η).

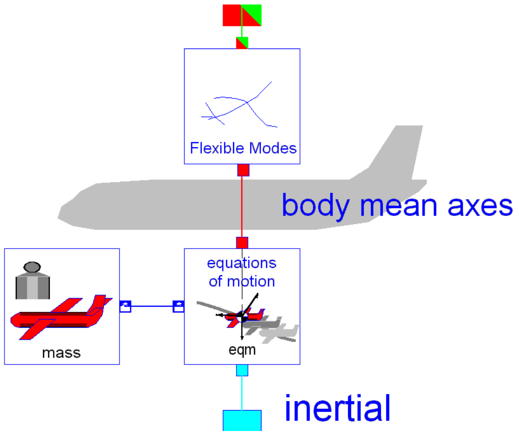


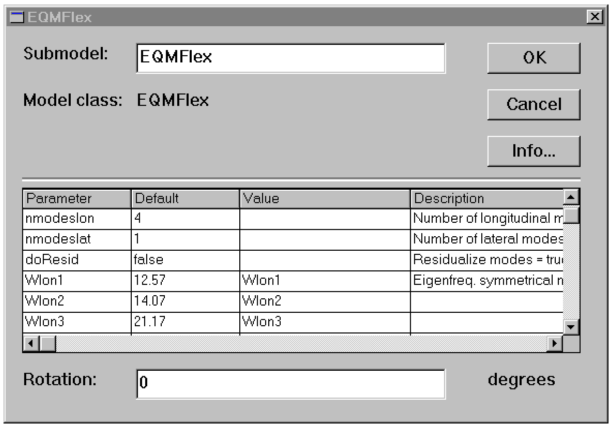
Figure 3: Equations of motion (EQM) object

All objects are of generic nature, and can be used in any aircraft model. When double-clicking on an object, a parameter window shows up that allows the user to enter aircraft-specific parameters. The window for the *Flexible Modes* object is shown in figure 4.

The number of longitudinal and lateral modes can be adapted via *nmodeslon* and *nmodeslat* respectively, as a means of scaling the fidelity of the elastic part of the model. The eigenfrequencies can be adapted as well (*Wlon1, Wlon2*, etc.). The parameter *doResid* specifies what to do with the modes to be removed. If *doResid* is true, $\dot{\eta}_j$ and $\ddot{\eta}_j$ in equation 6 are set to zero so that remains:

$$M_j(\omega_j^2 \eta_j) = Q_{\eta_j}$$

which will be solved by the model compiler. In the other case, η_j, $\dot{\eta}_j$ and $\ddot{\eta}_j$ are set to zero. The model compiler then automatically removes the equation in the simulation code, thus truncating mode j.



Submodel: EQMFlex

Model class: EQMFlex

Parameter	Default	Value	Description
nmodeslon	4		Number of longitudinal m
nmodeslat	1		Number of lateral modes
doResid	false		Residualize modes = tru
Wlon1	12.57	Wlon1	Eigenfreq. symmetrical n
Wlon2	14.07	Wlon2	
Wlon3	21.17	Wlon3	

Rotation: 0 degrees

Figure 4: Parameter dialog window *Flexible Modes*

Engine model object

Figure 5 shows the contents of an *Engine Model* object. The *Thrust* block contains a highly simplified engine model that only scales a throttle input into a thrust force (without dynamics). This thrust force is simply defined as $T = [T_x, 0, 0]$, where T_x is the thrust along the local x-axis. As discussed in the previous section, a local axis system (attached to the airframe) is defined at the engine location. This is done via four transformation objects, *TranslateFix, TranslateFlex, RotateFlex* and *RotateFix*. These relate the mean axes with the local axis system on the airframe via a fixed translation (from CoG to engine in undeformed position), a time dependent translation and rotation due to local airframe deformation, and a fixed rotation (engines are usually installed with a small angular offset) respectively. By connecting the engine model with the transformation objects, directional thrust variations with respect to the mean axes due to flexibility are automatically accounted for. The transformation objects are generic; they are used within the *sensors* object as well. Local airframe information (e.g. position, mode shape vectors) is specified via a parameter window.

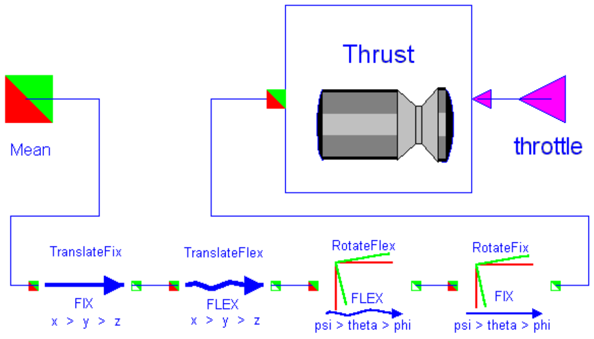


Figure 5: Engine implementation

Sensors object

In figure 2 two sensor objects are visible, *CoG* and *Cockpit*. Both are based on the generic object *sensors*, depicted in figure 6. On the right we see five (simple) sensor models that are modeled in their own axis systems. Like in the object *Engine model*, each local axis system is related to the mean axes via four transformations. The transformation objects are exactly the same as for the engines, only local position and mode shape information for the cockpit has been entered via the parameter window.

The transformation objects are based on a same parent object. This parent has the two cuts and implements general kinematic relations between the cut variables (e.g. eq. 7). The four transformation objects inherit these features and implement the actual kind of transformation (e.g. flexible translation).

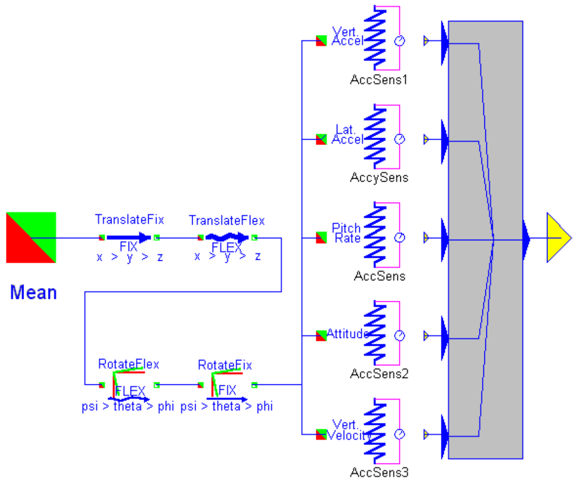


Figure 6: Sensor implementation

Aerodynamics object

The *Aerodynamics* object contains the aerodynamic model in the form of equation 8. The RR part is implemented in the form of stability and control derivatives that are interpolated from look-up tables. The RF, FR, and FF parts are implemented as constant matrices, valid for a single operating condition and aircraft configuration. The aerodynamic model is defined in mean axes.

5. Model processing

The model building process in Dymola is summarized in figure 7. At the top is the actual model implementation, as described in the previous section. A model is composed in the graphical user interface (GUI) from objects that are stored in libraries.

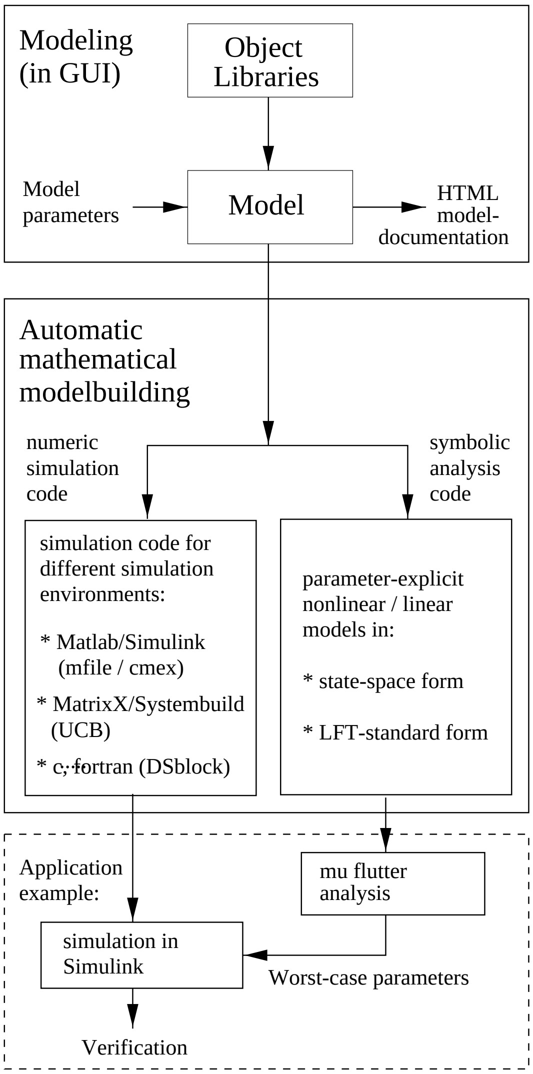


Figure 7: Model building process

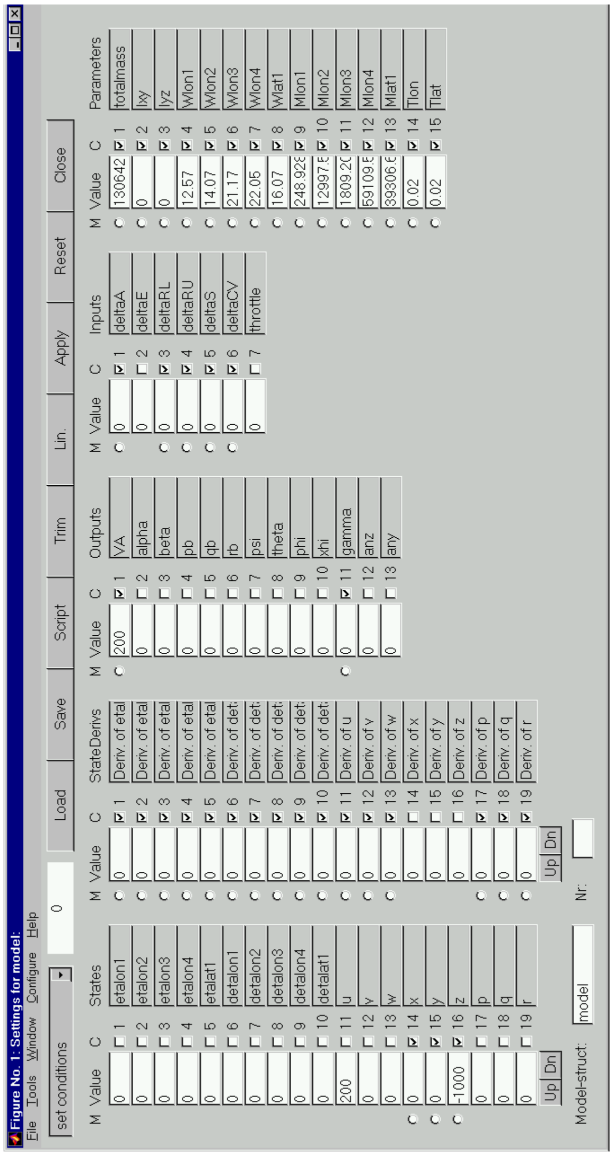
Parameters can be adjusted to aircraft-specific values. An HTML-code generation facility is available for model documentation, see for an example ref. [13]. Once finished, the model is loaded in the Dymola main program. The symbolic equation handler collects all equations from the model objects and solves them according to the inputs and outputs of the aircraft model (*Automatic mathematical model building* figure 7). Equations that are not necessary in the simulation code are automatically removed. Symbolic simulation code is generated in nonlinear state space form. This code can be exported in several languages, like c, Fortran, but also for specific environments like MATLAB/Simulink (m-code, or Simstruct c-code) and Matrix$_x$/SystemBuild (User Code Block). Another possibility is to generate symbolic analysis code in state-space from, which can be used to symbolically generate a model in the form of a Linear Fractional Transformation using symbolic mathematical software, see ref. [19, 18].

Dymola offers a built-in trimming/simulation and visualization facility for direct model evaluation. However, for application such as control law design, the export facility to for example MATLAB/Simulink is extremely useful. We used Simstruct code to simulate the aircraft in this environment. With this code also an m-file is generated in which parameter, state, input, and outputs names and initial values are defined. This information can for example be used to automatically generate a GUI in MATLAB for trimming and linearizing the model, see figure 8. Within this GUI, trim values can be entered and specified as 'fixed' or 'free' in the trim computation. For this trim computation a mex-function based on MINPACK has been implemented.

Finally, the aircraft model can be included into a simulink diagram with a block diagram of the control system to perform closed loop simulations.

6. Application example

As an application example, we use the model for flutter analysis in the μ-framework [10]. The aircraft dynamics are augmented by a pitch rate feedback via



M	Value	C	States
	0	□ 1	etalon1
	0	□ 2	etalon2
	0	□ 3	etalon3
	0	□ 4	etalon4
	0	□ 5	etalat1
	0	□ 6	detalon1
	0	□ 7	detalon2
	0	□ 8	detalon3
	0	□ 9	detalon4
	0	□ 10	detalat1
	200	□ 11	u
	0	□ 12	v
	0	□ 13	w
○	0	☑ 14	x
○	0	☑ 15	y
○	-1000	☑ 16	z
	0	□ 17	p
	0	□ 18	q
	0	□ 19	r

M	Value	C	StateDerivs
○	0	☑ 1	Deriv. of etal
○	0	☑ 2	Deriv. of etal
○	0	☑ 3	Deriv. of etal
○	0	☑ 4	Deriv. of etal
○	0	☑ 5	Deriv. of etal
○	0	☑ 6	Deriv. of det
○	0	☑ 7	Deriv. of det
○	0	☑ 8	Deriv. of det
○	0	☑ 9	Deriv. of det
○	0	☑ 10	Deriv. of det
○	0	☑ 11	Deriv. of u
○	0	☑ 12	Deriv. of v
○	0	☑ 13	Deriv. of w
	0	□ 14	Deriv. of x
	0	□ 15	Deriv. of y
	0	□ 16	Deriv. of z
○	0	☑ 17	Deriv. of p
○	0	☑ 18	Deriv. of q
○	0	☑ 19	Deriv. of r

M	Value	C	Outputs
○	200	☑ 1	VA
	0	□ 2	alpha
	0	□ 3	beta
	0	□ 4	pb
	0	□ 5	qb
	0	□ 6	rb
	0	□ 7	psi
	0	□ 8	theta
	0	□ 9	phi
	0	□ 10	xhi
○	0	☑ 11	gamma
	0	□ 12	anz
	0	□ 13	any

M	Value	C	Inputs
○	0	☑ 1	deltaA
	0	□ 2	deltaE
○	0	☑ 3	deltaRL
○	0	☑ 4	deltaRU
○	0	☑ 5	deltaS
○	0	☑ 6	deltaCV
	0	□ 7	throttle

M	Value	C	Parameters
○	130642	☑ 1	totalmass
○	0	☑ 2	Ixy
○	0	☑ 3	Iyz
○	12.57	☑ 4	Wlon1
○	14.07	☑ 5	Wlon2
○	21.17	☑ 6	Wlon3
○	22.05	☑ 7	Wlon4
○	16.07	☑ 8	Wlat1
○	248.92	☑ 9	Mlon1
○	12997.5	☑ 10	Mlon2
○	1809.20	☑ 11	Mlon3
○	59109.5	☑ 12	Mlon4
○	39306.6	☑ 13	Mlat1
○	0.02	☑ 14	Tlon
○	0.02	☑ 15	Tlat

Model-struct: model

Nr:

Figure 8: Screen shot of MATLAB trim GUI

the elevator. The pitch rate is sensed near the cockpit. Using the software described in ref. [19], which combines available symbolic and numeric computational software tools, we automatically generate a model in the form of a Linear Fractional Transformation (LFT) from the parametric Dymola model. The LFT, in which varying parameters are separated from the model and interconnected in a feedback path, is the standard form for μ-analysis. The LFT of the example aircraft model contains airspeed, structural eigenvalues and modal damping as varying parameters. With μ-analysis a worst-case flutter speed is computed in face of the uncertain structural parameters. The worst-case parameter combination is substituted into the Dymola generated simulink model, to verify the occurrence of flutter in a simulation. It turns out that the third symmetrical mode goes unstable, see figures 9, 10.

Since both the LFT model and simulation model have the same Dymola model as origin, this illustrates the advantage of single-source modeling capability (figure 7).

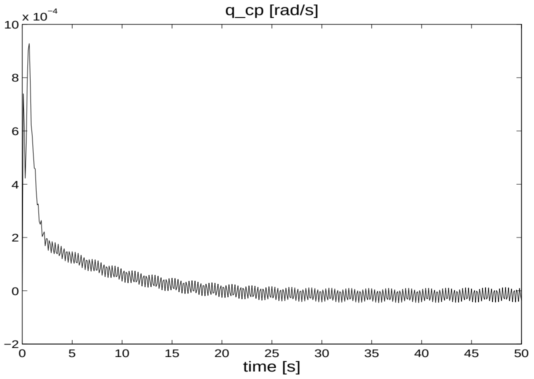


Figure 9: Pitch rate in cockpit: verification of flutter for worst model parameters

7. Conclusions

In this paper we have proposed a software implementation of integrated flight mechanics / aeroelastic aircraft models exploiting the features of object oriented modeling. Sub-models from different engineering disciplines can be implemented one-to-one

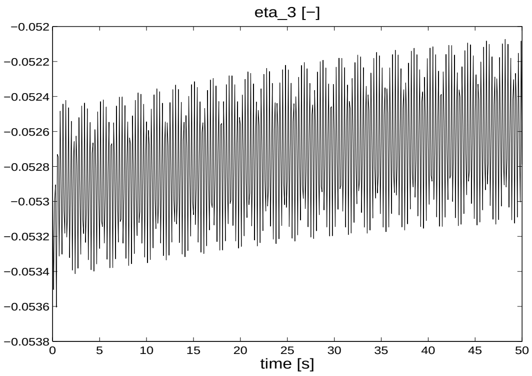


Figure 10: Gen. displacement of third symm. mode

into software objects, since we are able so specify the interconnections according to the way the objects physically interact.

The equations of motion form the core of the model implementation. The rigid simulation model is used as a basis, while the fidelity of the aeroelastic part may depend on the accuracy required by the application.

The implementation of an elastic aircraft model of limited complexity was demonstrated. Except for the aerodynamic model, other model components can be easily re-used in other aircraft models by adapting values in the object parameter window. At the moment, the fidelity of the aeroelastic model can only be adjusted via the number of modes. For more complex models this, for example, could also be done via the order of the rational function approximation of the unsteady aerodynamics, by implementing several aerodynamic models that are easily exchangable in the integrated aircraft model.

The package used for implementing the model, Dymola, is able to automatically generate simulation code for several engineering environments. The possibility to generate symbolic code facilitates parametric uncertainty modeling, since model parameters can be treated explicitly in e.g. a symbolic mathematical package. A great advantage is that the symbolic and the simulation model are generated from the same source. An example was given by generating an LFT model from the Dymola code to perform flutter analysis.

So far, the implementation is conceptual, since the complexity of the implemented aircraft model is low, and the model was readily at hand. Future work will involve actual modeling work and implementation of a more complex aircraft model.

Acknowledgment

The author wishes to thank Dieter Moormann from the same department for his support in all Dymola implementation work.

References

[1] E. Albano and W.P. Rodden. A Doublet-Lattice Method for Calculating Lifting Disturbances of Oscillating Surfaces in Subsonic Flows. *AIAA Journal*, 7(2):279–285, 1969. (Errata in AIAA Journal 7(11), November 1969, p.2192).

[2] P. Douglas Arbuckle, Carey S. Buttrill, and Thomas A. Zeiler. Simulation Model Building Procedure for Dynamic Systems Integration. *Journal of Guidance Control and Dynamics*, 12:894–900, 1989.

[3] Rudolf Brockhaus. *Flugregelung*. Springer-Verlag, Berlin Heidelberg, 1994.

[4] 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.

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

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

[7] Integrated Systems Inc. *SystemBuild User's Guide, version 6*, 1998.

[8] M. Karpel and S.T. Hoadley. Physically Weighted Minimum State Unsteady Aerodynamic Approximation. Technical Report NASA-TP-3025, NASA, 1990.

[9] E. Levretsky. High Speed Civil Transport (HCST) Flight Simulation and Analysis Software Development. In *Proceedings of the 36th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, Jan 1998. AIAA-98-0173.

[10] Rick Lind and Martin J. Brenner. Robust Flutter Margin Analysis that Incorporates Flight Data. Technical Report TP-1998-206543, NASA, Dryden Flight Research Center, Edwards CA, 1998.

[11] The MathWorks Inc. *Simulink User's Guide*, 1998.

[12] R.D. Milne. Dynamics of the Deformable Airplane (Part I and II). Technical Report R & M Number 3345, and AD-A953155, Her Majesty's Stationary Office, London, England, 1964.

[13] Dieter Moormann. GARTEUR Research Civil Aircraft Model (RCAM): Aircraft Physical Modeling Leading to Automatic Code Generation. Technical report, DLR-Oberpfaffenhofen, Institute for Robotics and System Dynamics, 1997. Hypertext document: `http://www.op.dlr.de/DD-DR-ER/research/flight/rcammod/node2.html`.

[14] D. Moormnann, P.J. Mosterman, and G. Looye. Object-oriented computational model building of aircraft flight dynamics and systems. *Aerospace Science and Technology*, 3:115–126, 1999. To appear early 1999.

[15] Martin Otter. *Objektorientierte Modellierung mechatronischer Systeme am Beispiel Geregelter Roboter*. PhD thesis, Fakultät für Maschinenbau der Ruhr-Universität Bochum, November 1993.

[16] K. Roger. Airplane Math Modeling Methods for Active Control Design. *AGARD*, (CP-228), August 1977.

[17] J. Schuler. *Flugregelung und aktive Schwingungsdämpfung für flexible Großraumflugzeuge – Modellbildung und Simulation*. PhD thesis, Institut für Flugmechanik und Flugregelung, Universität Stuttgart, 1997.

[18] A. Varga and G. Looye. Symbolic and numerical software tools for lft-based low order uncertainty modeling. In *Proc. CACSD'99 Symposium, Cohala Coast, Hawaii*, 1999.

[19] A. Varga, G. Looye, D. Moormann, and G. Grübel. Automated Generation of LFT-based Parametric Uncertainty Descriptions from Generic Aircraft Models. *Mathematical and Computer Modelling of Dynamical Systems*, 4(4):249–274, 1998. See also GARTEUR report TP-088-36 at `http://www.nlr.nl/public/hosted-sites/garteur/sum36.html`.

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

[21] M.R. Waszak and D.K. Schmidt. A Simulation Study of the Flight Dynamics of Elastic Aircraft: Volume 1 – Experiment, Results and Analysis. Technical Report NASA CP-4102, NASA, December 1987.

[22] B.A. Winther, P.J. Goggin, and J.R. Dykman. Reduced Order Dynamic Aeroelastic Model Development and Integration with Nonlinear Simulation. In *Proceedings of the AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1694 – 1704, Los Angeles, April 1998.