# Aircraft – A Modelica Library for Aircraft Dynamics Simulation

Andreas Idebrant, Peter Fritzson

MathCore Engineering AB, Teknikringen 1B,
SE-58320 Linköping, Sweden
Email: {idebrant, petfr}@mathcore.com; http://www.mathcore.com

## ABSTRACT

The `Aircraft` library is a versatile Modelica library for modeling and simulating aircraft dynamics applications. It is structured into a number of sublibraries, that contain models for describing the aerodynamics, atmosphere aerodynamic impact on aircraft, bodies, engine models, coordinate transformation models, etc. The library is very easy to use, and automatically computes up-to-date center-of-mass and moment of inertia depending on the mass and position of components included in the application model. External components written in C can be included, e.g. external controller models. The library has successfully been used to model the flight dynamics of a generic aircraft whose visual appearance has superficial similarities to the Swedish JAS Gripen aircraft, using the MathModelica tool for modeling, simulation, and 3D visualization.

## KEYWORDS

Modelica, Aircraft, Dynamics, MathModelica, Equation-based, Object-Oriented

## 1 INTRODUCTION

The `Aircraft` library was developed in 2003 at MathCore Engineering AB, Linköping, Sweden, in cooperation with FOI (The Swedish Defense Institute) in Stockholm. The project was financed by FMV (Swedish Defence Material Administration) and conducted by FOI. The developed library is a property of FOI. The goal was to make a generic and easy-to-use aircraft model in the object-oriented modeling language Modelica. The GAM data [1] and the ADMIRE model [2] have been used as a reference. Some design principles have been adopted from
[3] and [4]. The result is the `Aircraft` library containing about 50 component classes for modeling and simulating an aircraft in various configurations.

This paper does not give any introduction to the Modelica language or the MathModelica [9] tool. For more information about Modelica there are currently two books available: “Principles of Object-Oriented Modeling and Simulation with Modelica 2.1” [5] and “Introduction to Physical Modeling with Modelica” [6].

## 2 LIBRARY STRUCTURE

The `Aircraft` library currently consists of the following subpackages:

| | |
|---|---|
| **Aerodynamics** | Aerodynamic models |
| **Atmosphere** | Aerodynamic impact on aircraft |
| **Bodies** | Aircraft bodies without aerodynamic influence |
| **Examples** | Example models |
| **Functions** | Common functions |
| **Icons** | Icons specific for this package |
| **Interfaces** | Connector interfaces |
| **Propulsion** | Engine models |
| **Test** | Test models to verify their behavior |

| | |
|---|---|
| **Transformations** | Transformation models for frames |
| **Types** | Types specific for the main package |
| **Utilities** | Utility components used for aircrafts |

In the following we will briefly describe a few of these subpackages, but will first take a quick look at a complete aircraft system model, including an aircraft body.

## 3 A COMPLETE AIRCRAFT MODEL

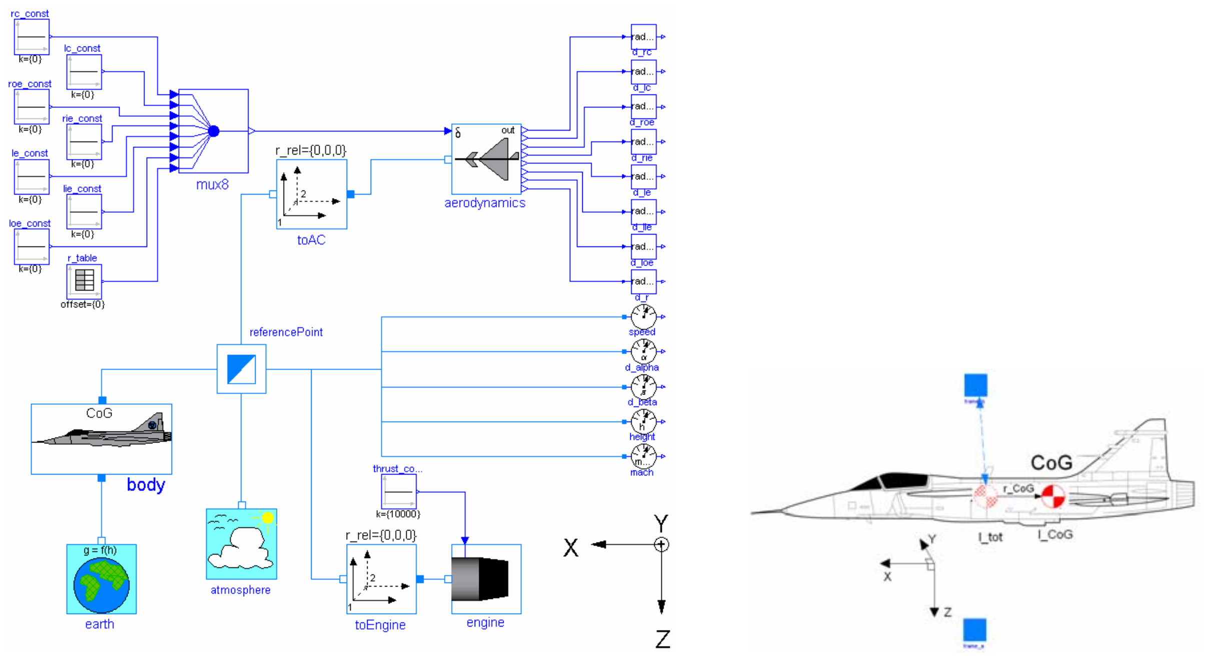


Figure 1. Left: The structure of a complete aircraft system model. Right: The actual aircraft body.

The connection diagram to the left in Figure 1 shows an example of a whole aircraft system model, including the aircraft body, gravitational and aerodynamic influence, controller, propulsion unit, etc. The picture to the right shows an aircraft body viewed from the side.

## 4 AERODYNAMICS SUBPACKAGE

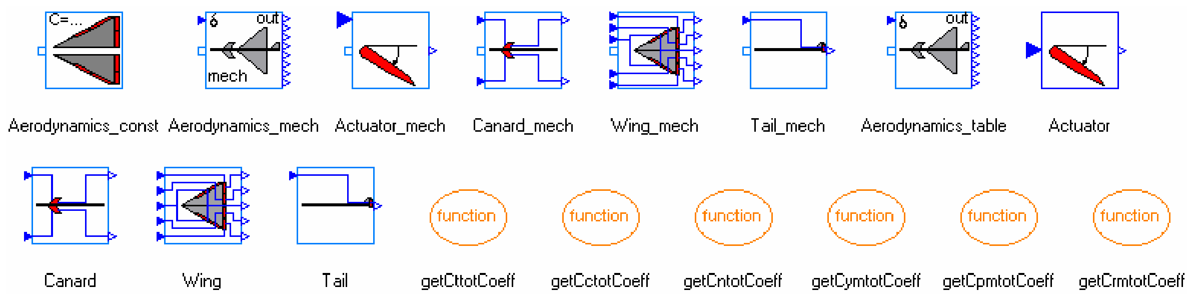


Figure 2. Overview of the `Aerodynamics` subpackage.

The `Aerodynamics` subpackage basically consists of three types of different aerodynamic models:

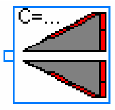
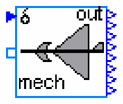
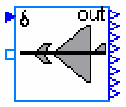
| | | |
|---|---|---|
|  | `Aerodynamics_const` | Constant aerodynamic force and moment coefficients where a certain flight mode can be studied if you know the value of the coefficients. |
|  | `Aerodynamics_mech` | Consists of components with mechanical flanges. This means that each rudder can be placed in an arbitrary position and orientation on a wing. The rudders contain equations for the force acting on the effective area of the rudder. The components with mechanical flanges have the extension "`_mech`". |
|  | `Aerodynamics_table` | Contains block components similar to MATLAB/Simulink models. The aerodynamic coefficients are calculated using external function calls to C-functions using GAM data [1]. The general data and conventions are described in "Report on the usage of the Generic Aerodata Model". |

Table 1. Overview of the `Atmosphere` package.

The last two objects have inputs for controlling the rudders and outputs from the rudder deviations.

## 5 ATMOSPHERE SUBPACKAGE

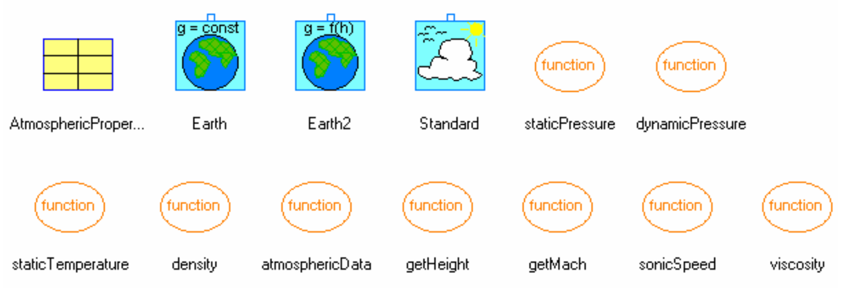


Figure 3. Overview of the `Atmosphere` package.

The `Atmosphere` package contains objects to build a model of the surrounding environment. Some properties that can be considered as constants are collected in the `AtmophericProperties` model. A standard atmosphere is modeled in the `Standard` component and calculates pressure, temperature, and density. The different quantities needed are calculated using the different functions placed in the `Atmosphere` package. There are also help functions for calculating different quantities placed in the package. The two earth models are used to model a constant gravity field and an altitude dependent gravity field respectively. They contribute the gravitational constant, `g`, to the system.

## 6 BODIES SUBPACKAGE

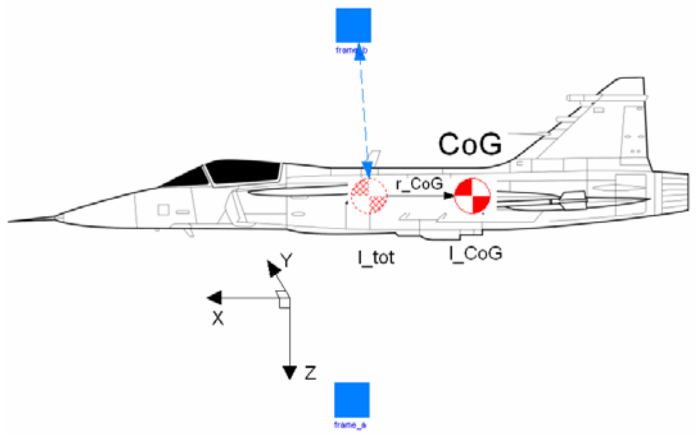


Figure 4. An aircraft body with variable center of gravity (CoG) and mass moment of inertia (I_CoG).

Currently the `Bodies` package contains one aircraft `Body` class containing aircraft body dynamics, specified in 6 degrees of freedom. It can handle movable center of gravity and also variable mass moment of inertia.

There are two connectors on the body. The lower should be connected to a model that calculates the gravitational acceleration, e.g. the earth models in `Atmosphere` package. The upper connector indicates the center of gravity (`CoG`) of the body when nothing else is attached. This connector should be connected to a reference point where the body fixed coordinate system is attached, e.g. an `ApplicationPoint` in the `Transformations` library. It is of course possible to use a transformation object to place the `CoG` in an arbitrary way.

To be able to calculate the body dynamics correctly we need to know the total center of gravity and total mass moment of inertia in that point. To be able to do this, two quantities were introduced in the connector type `Frame`. These are mass (`mass`), and the product of mass and position to initial center of gravity (`md`). These are flow variables, which means that they are summed up in a junction point and in this case the total sum are located in `frame_b` at the top in Figure 4.

The total mass (`m_tot`) is calculated using the mass at `frame_b` (the upper connector in Figure 4) and the total empty body mass:

```
    m_tot = frame_b.m + mass;                                  (1)
```

In the same way the mass moment of inertia at `frame_b` is summed:

```
    I_tot = frame_b.I + I_0;                                   (2)
```

Position of center of gravity from initial center of gravity (`r_CoG` in Figure 4) is calculated by summing the total mass of components multiplied by their distance to the initial center of gravity divided by the total mass, i.e.

```
    r_CoG = frame_b.md/m_tot;                                  (3)
```

The total mass moment of inertia around `CoG` is calculated according to below:

```
    I_CoG = I_tot - m_tot*skew(r_CoG)*skew(r_CoG);             (4)
```

At the end the differential equations for the movement are calculated:

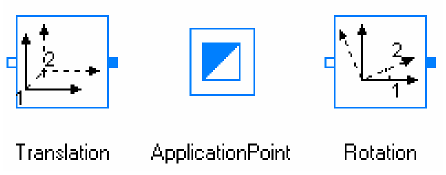
```
    der(r) = Tmat*(v + cross(w, r_CoG));
    der(Q) = -0.5*Q*{{0,p_w,q_w,r_w},{-p_w,0,-r_w,q_w},
             (-q_w,r_w,0,-p_w},{-r_w,-q_w,p_w,0}};             (5)
    F = m_tot*(der(v) - g + cross(w, v));
    M = I_CoG*der(w) + cross(w, I_CoG*w);
```

## 7 FRAME CONNECTORS

In the `Interfaces` subpackage there are four different connector models. The most fundamental connector is the `Frame` connector. The two connectors `Frame_a` (filled icon, positive side of component) and `Frame_b` (unfilled, negative) inherit from `Frame` and are identical apart from icons.

```
connector Frame  "Connector for aerodynamic and mechanical purposes"
  SI.Pressure          p    "Static pressure";
  SI.Density           rho  "Atmosphere density";
  SI.Temperature       T    "Static temperature";
  Types.TransferMatrix Tmat "Transfer matrix from local to inertial system";
  SI.Position          r[3] "Position of frame relative inertial frame";
  SI.AngularVelocity   w[3] "Angular velocity";
  SI.Velocity          v[3] "Translational velocity in earth frame";
  SI.Acceleration      g[3] "Gravitational acceleration";
  flow SI.Force[3]     F    "Force";
  flow SI.Torque[3]    M    "Torque";
  flow SI.Mass         m    "Mass";
  flow Real[3]         md(unit="kg.m") "Product mass*position (m*d) for calc CoG";
  flow SI.MomentOfInertia[3,3] I "Mass moment of inertia";
end Frame;
```

## 8 TRANSFORMATIONS, TYPES, AND AIRCRAFT ENGINE



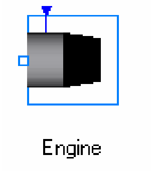


Figure 5. Left: Transformation and reference objects.  Right: Engine model in the `Propulsion` subpackage

The transformation and reference objects are essential for dealing with coordinate systems. They can move the coordinate system from one point to another. The `ApplicationPoint` serves as a reference point for the fixed coordinate system. It has a `Frame_ref` connector in the middle and inside the flow variables are all set to zero, since the connector should not produce any physical effect to the surroundings. The `Translation` object makes a translation from frame 1 (non-filled) to frame 2 (filled). If a rotation of the system is needed the `Rotation` object is used. Here it is possible to specify which axis (`x`, `y`, or `z`) the coordinate system is going to rotate about and which angle.

The `Aircraft` library makes extensive use of physical types. The Modelica standard library contains the packages `SIunits` and `NonSIunits` where most of the needed types can be found. However, there are a few missing and they are defined in the `Types` subpackage:

```
type TransferMatrix = Real[3,3]    "A standard 3x3 rotation matrix";
type Quaternion     = Real[4]      "Quaternion {q0,q1,q2,q3}";
type EulerAngle     = SI.Angle[3] "Euler angles {psi,theta,phi}";
```

The current engine model (see Figure 5, right) in the Propulsion subpackage is very simple. The more advanced engine model described in Figure 6 of [2] is not used here. There are two ways to configure this model. Either you choose to set a fixed force using the parameter `fixed_F` or you can use the block connection at the top to use an arbitrary signal. The block connection is compatible with the `Blocks` library from the Modelica standard library. As soon as the block connector is connected the parameter `fixed_F` is ignored.

## 9 UTILITIES SUBLIBRARY

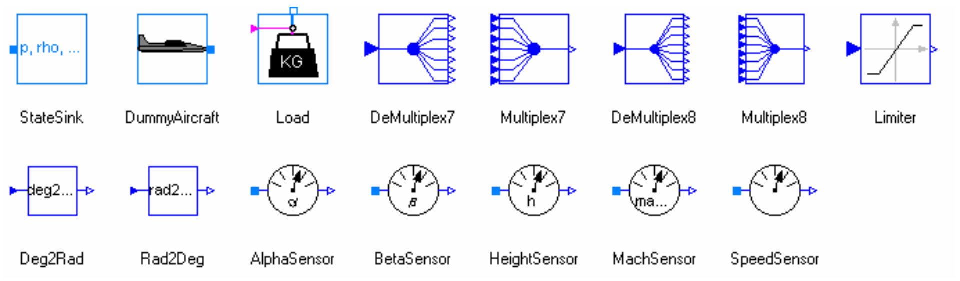


Figure 6. Components present in the `Utilities` library.

Some components not belonging to either of the other subpackages or just used for test purposes have been placed in the `Utilites` subpackage.

The `StateSink` component is just used to give the system values for the effort variables. For example, it is used in the `Test_Translation` model in the `Test` package to give a consistent system when testing the `Translation` model.

To be able to test certain model classes without having a full working aircraft a simplified aircraft, `DummyAircraft`, was invented initially to supply the system with the required dynamics data.

When simulating a change in load and total mass it is useful to make use of the `Load` model. It contains mass and mass moment of inertia. It can be placed arbitrary by using the transformation objects in the `Transformations` package. The pink connection to the upper left makes it possible to connect a Boolean component from the standard Modelica library and simulate a load drop, i.e. disconnect the mass. This can be useful when simulating a missile that is being fired or a tank being dropped at a certain event.

The multiplexers and demultiplexers are used as inputs and outputs for a controller. The limiter model limits a signal and there are also two models that convert between degrees and radians.

The five sensors are ideal and transfers a property in a frame connector to the physical value specified by their name.

## 10 AIRCRAFT EXAMPLES SUBPACKAGE

The `Examples` subpackage contains typical application models built using the component models of the `Aircraft` library, also including the following two simple examples:

1. The `Skid` model, where the tail rudder wiggles making the body to “skid”. It is possible to simulate the model for about 4 seconds before it fails.
2. The `SimpleControl` model, where a simple PID-controller has been defined controlling alpha-angle and height. This model can be simulated for about 50 seconds before it fails.

Recently a more realistic controller has been designed, which has been used for modeling more complex and prolonged flight scenarios, including a looping scenario.

## 11 MODEL TESTING AND VALIDATION

To make sure that the models are valid, each component has been tested thoroughly. Almost all tests have also been stored in the `Test` package. Often the name convention used is `Test_`*component*, where *component* is the name of the component being tested. Doing these tests, errors can be discovered in an early stage and are easier to pinpoint and correct.

The behavior of the components has been validated using “know-how”, e.g., if you move the tail rudder the aircraft should turn in the correct direction etc. A 3D-visualization tool, called MVIS[1], together with a simple 3D CAD-model of a fighter aircraft has been used for that purpose.

## 12 CONCLUSIONS

The Modelica `Aircraft` library has proved to be an easy-to-use and fairly comprehensive library for building models of aircraft dynamics. It has been successfully used in several flight scenarios.

## REFERENCES

[1] Backström, H. Report on the usage of the Generic Aero data Model. Saab Aircraft AB. Linköping, 1996.

[2] Forsell, Lars, et. al. The Aero-Data Model In a Research Environment (ADMIRE) for Flight Control Robustness Evaluation. August, 2001.

[3] Idebrant, Andreas. Multidomänmodellering och simulering av jaktrobot. Master thesis LiTH-ISY-EX-3085. In Swedish. Linköping, 2000.

[4] "Automatisierte Modellbildung der Flugsystemdynamik". Dr.-Ing Dieter Moormann. Fortschr.-Ber. VDI Reihe 8 Nr. 931. Düsseldorf: VDI Verlag 2002. ISBN 3-18-393108-7.

[5] Fritzson, Peter. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. 940 pages, ISBN 0-471-471631, Wiley-IEEE Press 2003.

[6] Tiller, Michael. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers 2001, ISBN 0-7923-7367-7.

[7] Ward, J.P., *Quaternions and Cayley Numbers*. Kluwer Academic Publishers, 1997.

[8] Nelson, Robert C.. *Flight Stability and Automatic Control*, Second Edition 1998.

[9] MathCore Engineering AB. *MathModelica User’s Guide*. www.mathcore.com, 2003.

[1] MVIS or Modelica VISualizer is a tool for automatic 3D-visualization of 3D multi-body models in the Modelica standard library. It is also possible to use CAD-models in conjunction with real simulation data. More information is available at MathCore web site: http://www.mathcore.com.