JSBSim

An Open Source Flight Dynamics Model (FDM)

Agostino De Marco

7 April 2014

What is JSBSim?

JSBSim is a high fidelity, 6 DoF, flight dynamics and control software library written in the C++ programming language. Simulation features:

- Vehicle agnostic (completely data-driven)
- Standard or EGM96 gravitational model
- Geodetic (WGS-84) and geocentric position
- Standard or custom table-driven atmosphere
- MIL-F-8785C Turbulence model
- Mass properties update continuously
- Selectable integrators (explicit integration)

What is JSBSim?

- Flight dynamics and control S/W library
- ~50,000 lines of C++ code
- ~80 C++ classes
- In development since 1997
- Data driven
- XML configuration files

http://www.jsbsim.org

Goals of JSBSim

- To make simulation – of aerospace vehicles in particular – more accessible
- Ease of use and easy to learn (targeted at upper level college students)
- Minimize input required to model a vehicle and a scenario
- Provide for quick and easy analysis, debugging

JSBSim most notable uses

- FlightGear www.flightgear.org
- Outerra www.outerra.org
- BoozSimulator (Paparazzi) wiki.paparazziuav.org/wiki/BoozSimulator
- OpenEaagles www.openeaagles.org

A team of main developers, and a large base of users

An **open source**,
platform-independent,
flight dynamics & control software library in C++

JSBSim

Home | About Us | News | Download | Documentation | Mailing List | Links | SF.net/JSBSim | Bugs | Suggestions | Wiki | Aeromatic | MATLAB

Jon Berndt,
(Texas, U.S.A.)
Lead S/W Architect & Development Coordinator

Jon designed the original architecture, and continues to refine it, with inputs from the other team members. He has worked with military and space training and engineering simulators for many years. Jon is an aero engineer (University of Minnesota).

His **web site**.

Tony Peden
(Washington, U.S.A.)
Co-Author

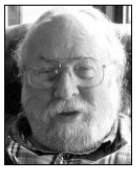
Tony has been contributing to the growth of JSBSim almost from day 1. He is responsible for integrating JSBSim with FlightGear, and for initialization and trimming. Tony also implemented David's property system into JSBSim. Tony hails from Ohio State University, with a degree in Aero and Astronautical Engineering.

David Culp
(U.S.A.)
Developer

David developed the turbine simulation for JSBSim, as well as aircraft models that use it, including the T-38. He has experience flying many types of military and commercial aircraft, including the T-38, and the Boeing 707, 727, 737, 757, 767, the SGS 2-32, and the OV-10. David is an aero engineer (USAF Academy).

Agostino De Marco
(Naples, Italy)
Developer/User

Agostino De Marco is a professor of aerospace engineering at the University of Naples in Italy.



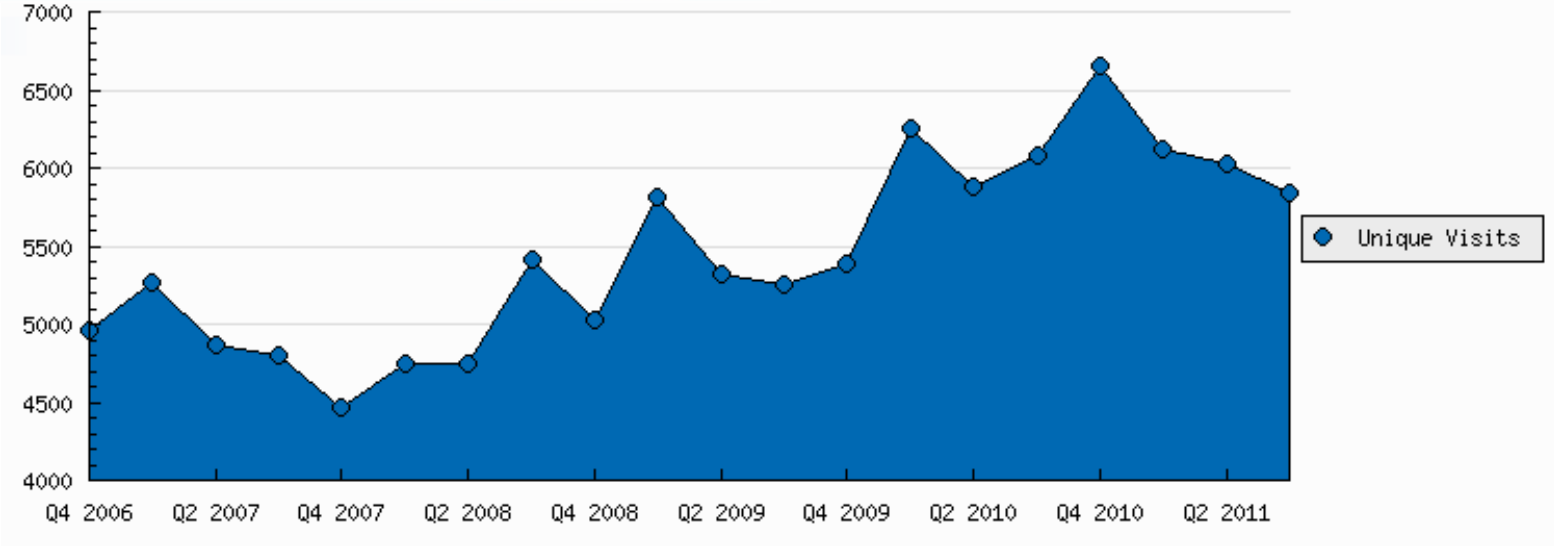
Lee Duke
(Glasgow, Virginia)
User/Developer

Lee Duke, the Chief Engineer of Rain Mountain Systems since 2004, retired from the NASA Dryden Flight Research Center in 2002 where he worked in flight test, flight controls, modeling and simulation, atmospheric flight dynamics, flight systems, and applications of artificial intelligence to aircraft systems. He is a member of IEEE, AIAA, and AUVSI.

Mathias Froehlich
(Germany)
Developer

Mathias improved and corrected the equations of motion for an early version JSBSim, among other things. Mathias is a mathematician.

JSBSim users in the world



Who is a JSBSim *Developer*?

- Developers:
  - Add or modify C++ code
  - Integrate JSBSim into new simulation architectures
  - Compile the code
  - Interact with the code using the API
  - Use models (XML files) for testing new features

Who is a JSBSim *User*?

- Users:
  - Do not write C++ code
  - Do not use compilers
  - Write or use XML models
  - Interact with the simulation using properties and XML files
  - Run the simulation to produce data
  - Analyze the data resulting from the runs
  - Unix command skills useful

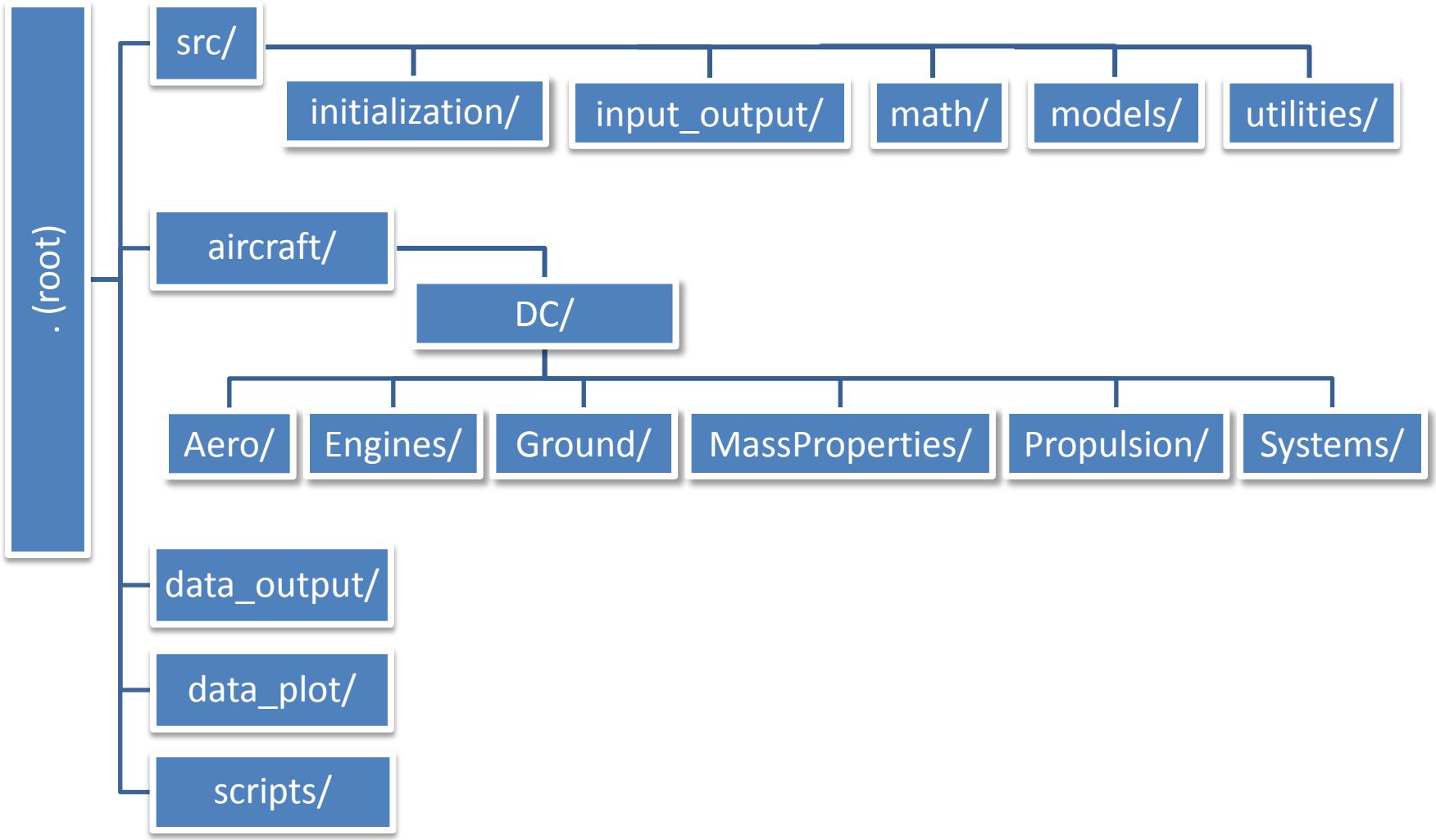
JSBSim ‘running modes’

- JSBSim can be run by itself as a standalone application (*batch mode*), and told to connect to FlightGear via socket, subsequently directing FlightGear what to display.
- The JSBSim executable for batch mode runs has a *reset capability*:
  - Reset integrator past states
  - Reset flight control component past states
  - Reconfigure aircraft settings in scripts
  - Trim aircraft
- Scripted runs are possible where the aircraft configuration file is loaded once, but multiple runs are made, such as for a set of Monte Carlo runs.

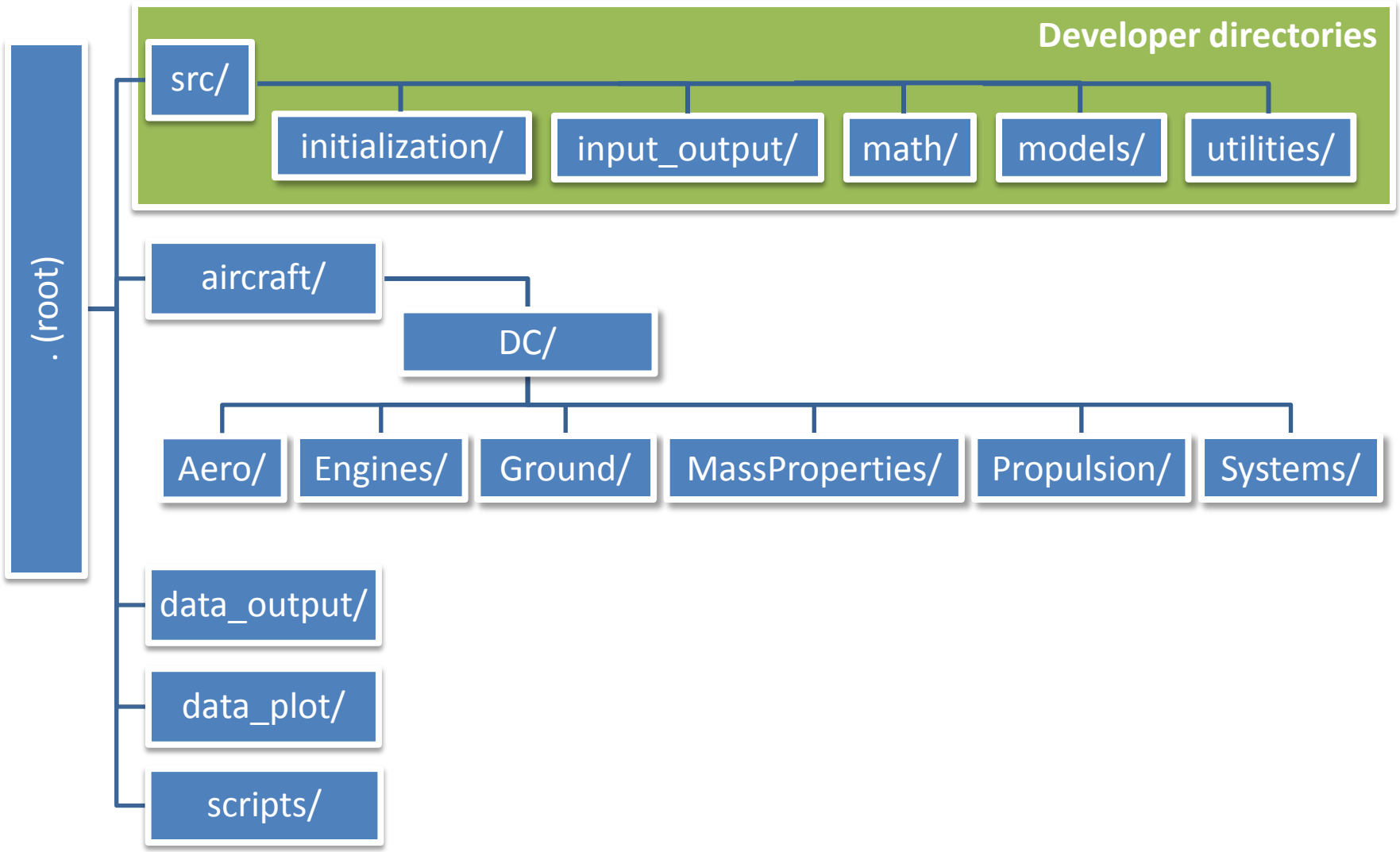
JSBSim in *batch mode*

```
JSBSim version 1.0 Feb 21 2012 11:15:51

Usage: jsbsim [script file name] [output file names] <options>

options:
  --help  returns this message
  --version  returns the version number
  --outputlogfile=<filename>  sets (overrides) the name of the first data output file
  --logdirectivefile=<filename>  specifies the name of a data logging directives file
                                 (can appear multiple times)
  --root=<path>  specifies the JSBSim root directory (where aircraft/, engine/, etc. reside)
  --aircraft=<filename>  specifies the name of the aircraft to be modeled
  --script=<filename>  specifies a script to run
  --realtime  specifies to run in actual real world time
  --nice  specifies to run at lower CPU usage
  --nohighlight  specifies that console output should be pure text only (no color)
  --suspend  specifies to suspend the simulation after initialization
  --initfile=<filename>  specifies an initilization file
  --catalog specifies that all properties for this aircraft model should be printed
           (catalog=aircraftname is an optional format)
  --property=<name=value> e.g. --property=simulation/integrator/rate/rotational=1
  --simulation-rate=<rate (double)> specifies the sim dT time or frequency
                    If rate specified is less than 1, it is interpreted as
                    a time step size, otherwise it is assumed to be a rate in Hertz.
  --end-time=<time (double)> specifies the sim end time

NOTE: There can be no spaces around the = sign when
      an option is followed by a filename
```

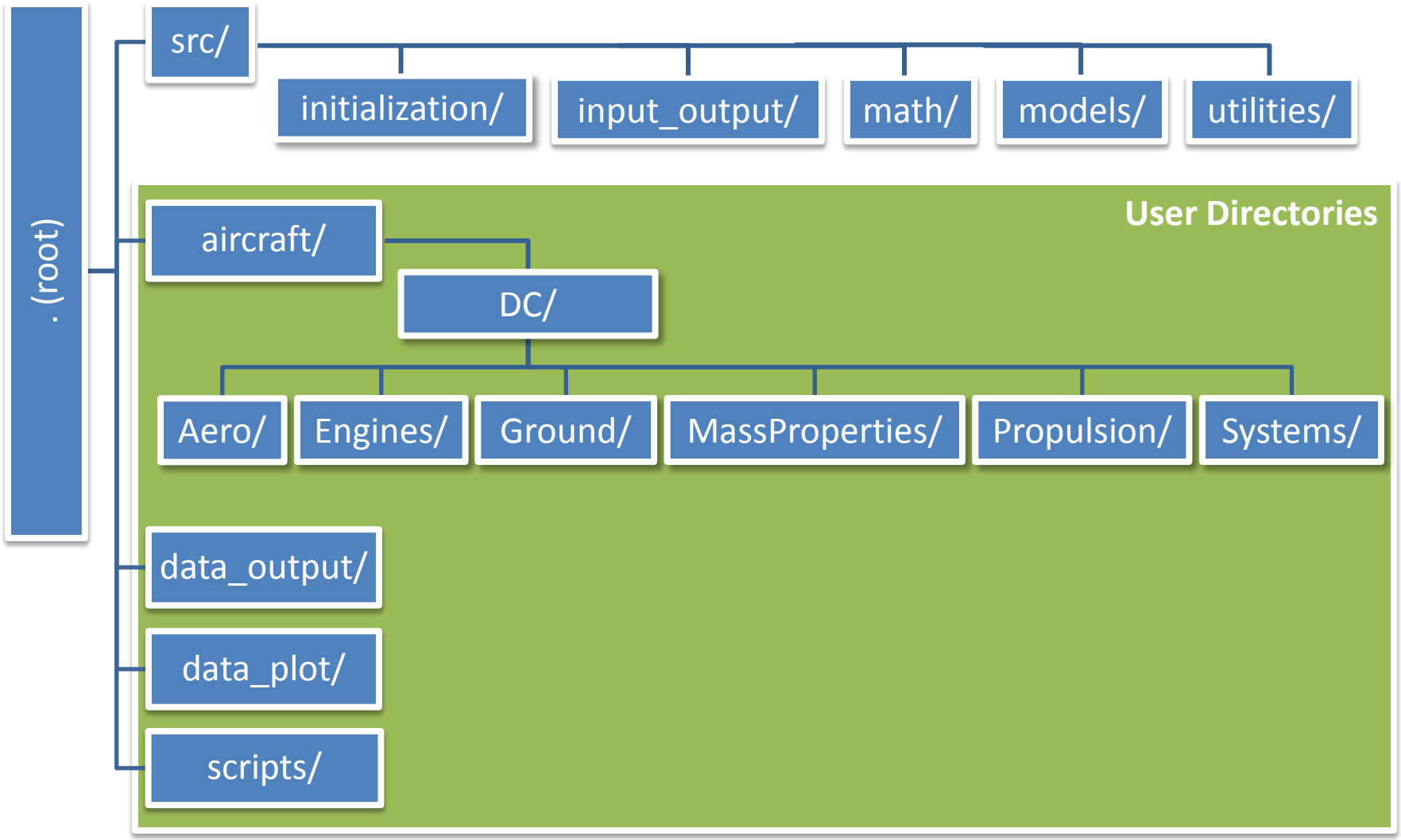
JSBSim Directory Tree



JSBSim Directory Tree



JSBSim Directory Tree



Files Needed by Users

- Model files (vehicle, engine, systems, aero, etc.)
- Script files
- Data logging files
- Data plotting files
- Shell scripts for running monte carlo analyses

Tools Needed

- Bash shell (cygwin or linux)
- Gnuplot
- Prep_plot

- Optional (pick one):
  - Microsoft Visual C++ 2010/2012 Express (free, devel.)
  - Notepad++ (free, user/devel.)
  - Matlab (non-free, users, post-processing)

User Workflow

- Setup scenario (edit model, scripts, etc.)
  - Run simulation
  - Auto-generate plots (e.g. Matlab)
  - Analyze plots
  - Modify input conditions / parameters
  - Rerun
- When ready: make runs for the record
- Plot data
- Archive CSV data with input files and run information in zip file to reduce storage reqmts.
- Present results

Analysis Computerized Model CM and Traceability

- Must be able to recreate runs
- For any run must know
  - Vehicle configuration
  - Dispersions
  - Environment setup
  - Other inputs and assumptions, etc.
- Save all run data, note executable version used and archive input data files w/data

Key Concepts in JSBSim

- XML
- Properties
- Functions

What is XML

- eXtensible Markup Language
- A *way* of encoding data
- Range of tools, technology, and code already available:
  - Schema
  - Transforms
  - Editors
  - Parsers (JSBSim uses the eXpat library)
- See:
  - W3 Schools (http://www.w3schools.com/xml/)
  - XML.org (http://www.xml.org)

XML Elements

Format:

```
<element> content </element>
```

Example:

```
<function>
  <product>
    <cos>   0.707 </cos>
    <value> aero/qbar-area </value>
  </product>
</function>
```

XML Attributes

Format:

```
<element attribute="value">
   content
</element>
```

Example:

```
<ixx dispersion="700"
     type="uniform"
     unit="SLUG*FT2"> 14009.9 </ixx>
```

Early JSBSim Data Specification

```
<COEFFICIENT NAME="CLalpha" TYPE="VECTOR">
  Lift_due_to_alpha
  8
  velocities/mach-norm
  aero/qbar-psf | metrics/Sw-sqft | aero/alpha-rad
  0.00 4.50
  0.40 3.80
  0.60 3.60
  1.05 4.50
  1.40 4.00
  2.80 2.50
  6.00 1.10
  9.00 1.00
</COEFFICIENT>
```

Current JSBSim Data Spec

```
<function name="aero/force/CLDf">
  <description>
    Delta lift due to flap deflection
  </description>
  <product>
    <p> aero/function/ground-effect-factor-lift </p>
    <p> aero/qbar-area </p>
    <table>
      <independentVar> fcs/flap-pos-deg </independentVar>
      <tableData>
         0.0  0.0
        10.0  0.25
        20.0  0.30
        30.0  0.35
      </tableData>
    </table>
  </product>
</function>
```

Properties

A Property is *used* like a Variable

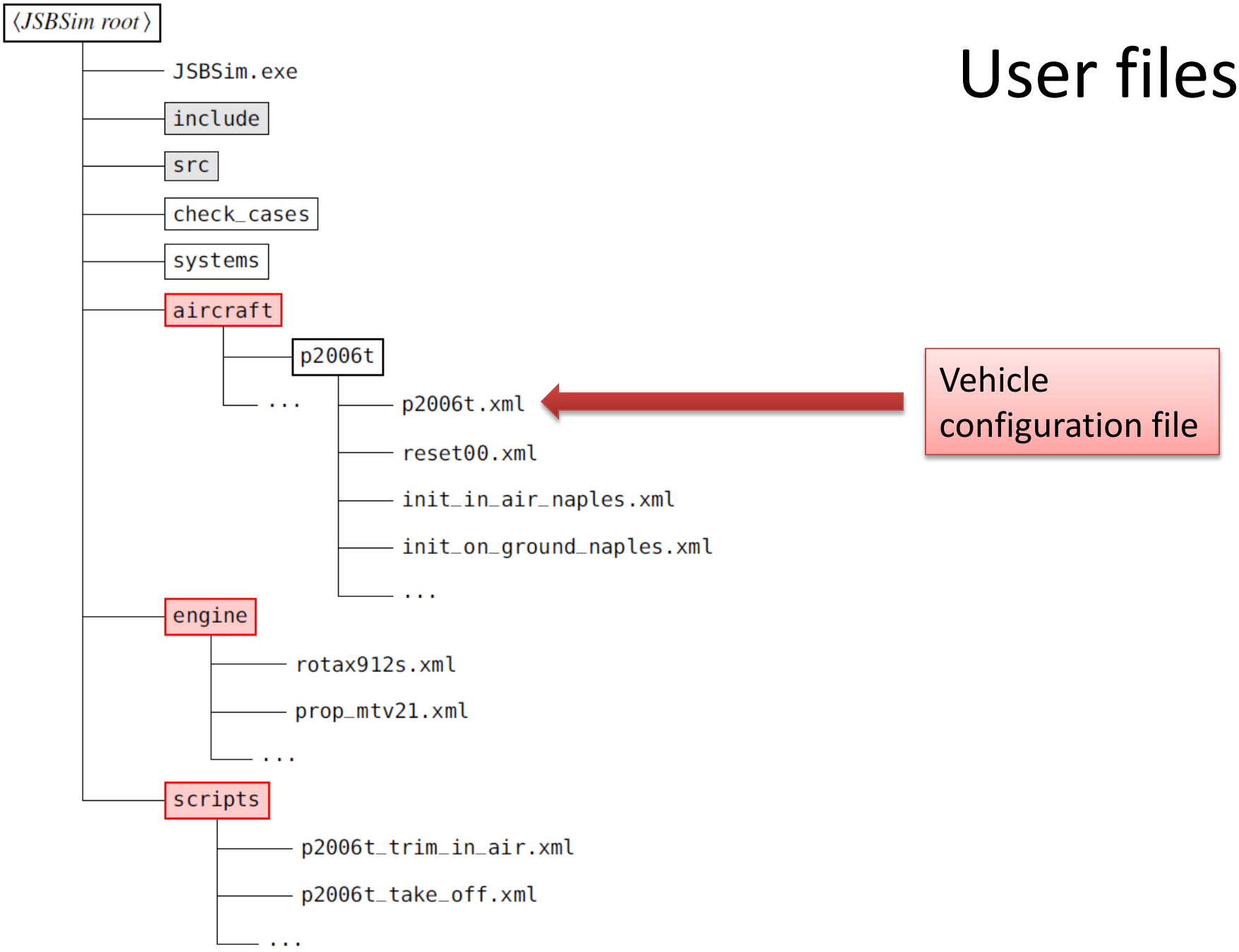
- Property names are defined in a hierarchical style:
  - attitude/phi-rad
  - velocities/u-fps
- A property can have read and/or read-write access
- There are native and created properties
- [Programmatically, a property may be bound to a program variable or a function – this is a concern mostly for developers]

Aircraft FDM catalog

```
$ JSBSim --aircraft=c172p --catalog > c172p_catalog.txt
```

- Exports a ‘catalog’ of JSBSim internal *properties* related to the selected aircraft (Cessna 172)
- Properties are nothing but variable/value pairs handled during simulations
- Each property value is ‘exposed’ to the user, i.e. can be grabbed and used for specific purposes

User files

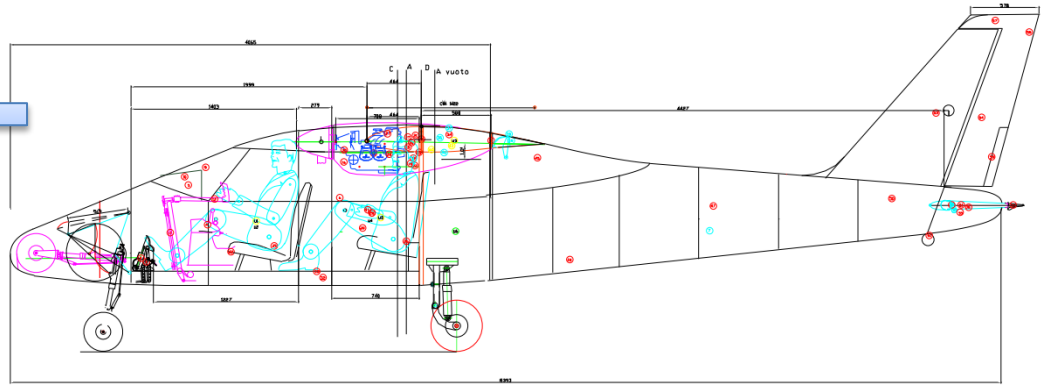

⟨JSBSim root⟩
JSBSim.exe
include
src
check_cases
systems
aircraft
p2006t
...
p2006t.xml
reset00.xml
init_in_air_naples.xml
init_on_ground_naples.xml
...
engine
rotax912s.xml
prop_mtv21.xml
...
scripts
p2006t_trim_in_air.xml
p2006t_take_off.xml
...
Vehicle configuration file

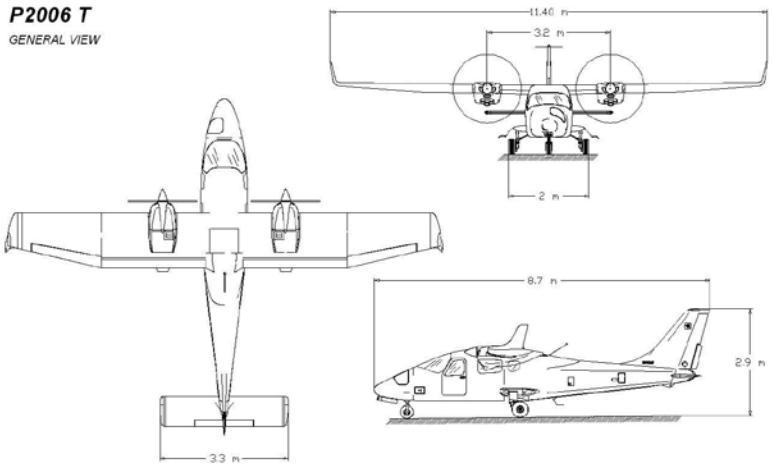
Vehicle configuration file

```
<fdm_config>
  <fileheader> … </fileheader>                    <!-- 0 or 1 instance  -->
  <metrics> … </metrics>                          <!-- 1 instance       -->
  <mass_balance> … </mass_balance>                <!-- 1 instance       -->
  <ground_reactions> … </ground_reactions>        <!-- 1 instance       -->
  <external_reactions> … </external_reactions>    <!-- 0 or 1 instance  -->
  <buoyant_forces> … </buoyant_forces>            <!-- 0 or 1 instance  -->
  <propulsion> … </propulsion>                    <!-- 0 or 1 instance  -->
  <system> … </system>                            <!-- 0 to n instances -->
  <autopilot> … </autopilot>                      <!-- 0 or 1 instance  -->
  <flight_control> … </flight_control>            <!-- 0 or 1 instance  -->
  <aerodynamics> … </aerodynamics>                <!-- 1 instance       -->
  <input> … </input>                              <!-- 0 or 1 instance  -->
  <output> … </output>                            <!-- 0 to n instances -->
</fdm_config>
```

Vehicle configuration

```
<metrics>
    <wingarea unit="M2"> 14.76 </wingarea>
    <wingspan unit="M">  11.4  </wingspan>
    <chord unit="M">      1.36 </chord>
    <htailarea unit="M2"> 2.57 </htailarea>
    <htailarm unit="M">   4.7  </htailarm>
    <vtailarea unit="M2"> 1.01 </vtailarea>
    <vtailarm unit="M">   1.04 </vtailarm>

    <location name="AERORP" unit="M">
        <x> 3.3  </x>
        <y> 0.0  </y>
        <z> 0.85 </z>
    </location>
    <location name="EYEPOINT" unit="M">
        <x> 2.15 </x>
        <y> 0.0  </y>
        <z> 0.72 </z>
    </location>
</metrics>

<mass_balance>
    <ixx unit="KG*M2"> 1617  </ixx>
    <iyy unit="KG*M2"> 1927  </iyy>
    <izz unit="KG*M2"> 2931  </izz>
    <ixy unit="KG*M2">    0  </ixy>
    <iyz unit="KG*M2">    0  </iyz>
    <ixz unit="KG*M2"> -221.3</ixz>
    <emptywt unit="KG"> 760 </emptywt>
            <location name="CG" unit="M">
        <x> 3.25 </x>
        <y> 0.0  </y>
        <z> 0.56 </z>
    </location>
            <pointmass name="PILOT">
                    <weight unit="KG">90</weight>
                    <location unit="M">
                            <x>  2.15 </x>
                            <y> -0.5  </y>
                            <z>  0.7  </z>
                    </location>
            </pointmass>
            <pointmass name="CO-PILOT">
        <weight unit="KG">90</weight>
        <location unit="M">
            <x> 2.15 </x>
            <y> 0.5 </y>
            <z> 0.7 </z>
```





Geometry

Masses

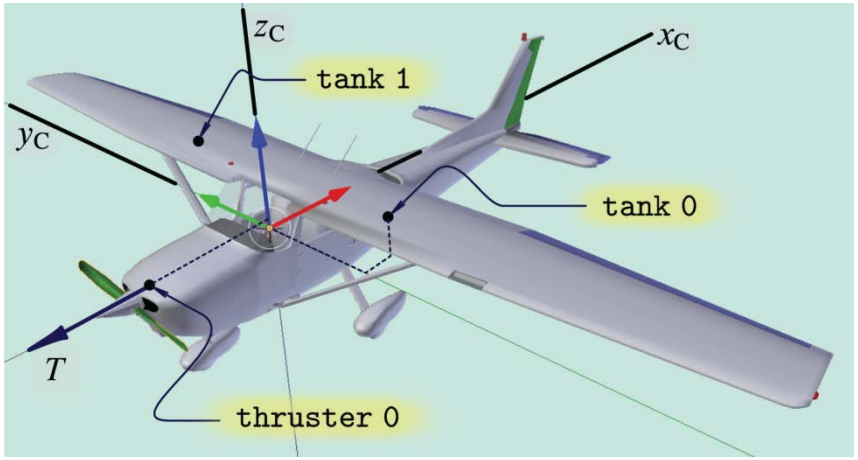
Vehicle configuration

Propulsion configuration files:

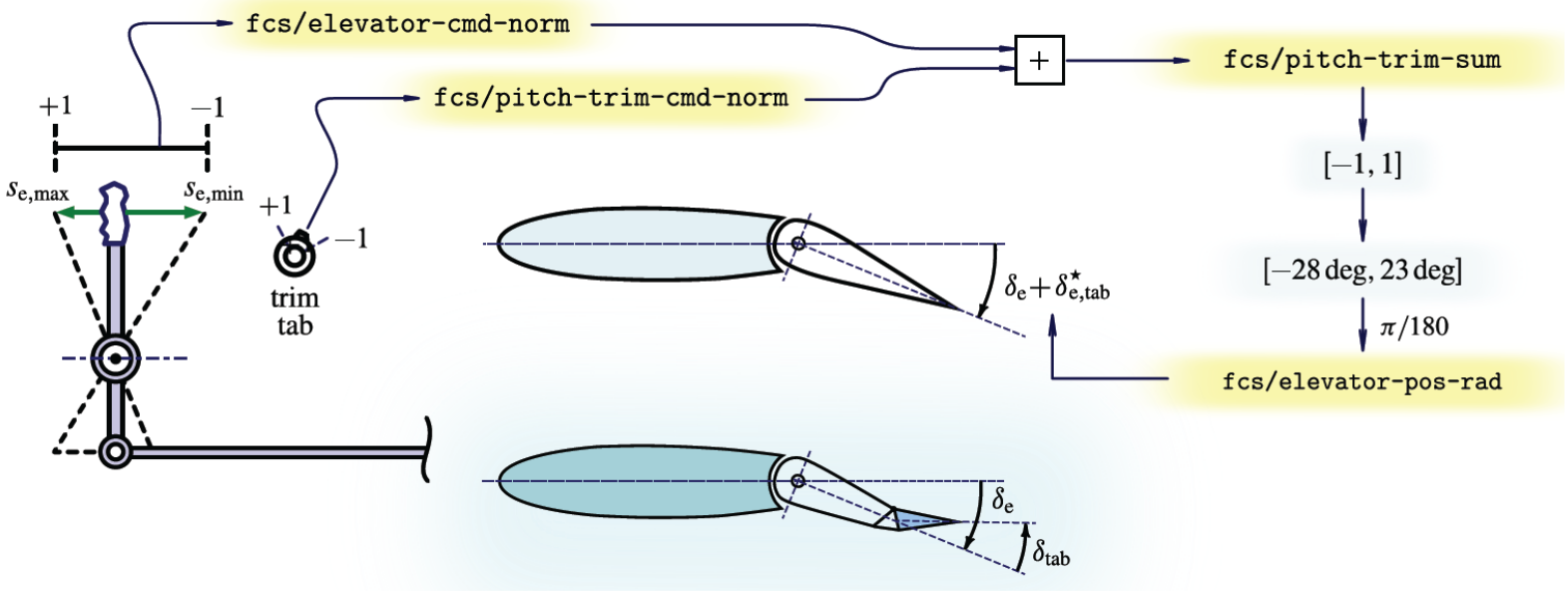
```
<?xml version="1.0"?>


<piston_engine name="ROTAX 912 S3">
  <minmp unit="INHG"> 18.0 </minmp>
  <maxmp unit="INHG"> 29.5 </maxmp>
  <displacement unit="IN3"> 82.6 </displacement>
  <cycles>          4.0 </cycles>
  <bore unit="IN"> 3.31</bore>
  <stroke unit="IN">2.4</stroke>
  <compressionratio>10.5</compressionratio>
  <maxhp>         95.30 </maxhp>
  <idlerpm>       900.0 </idlerpm>
  <maxrpm>       5800.0 </maxrpm>
  <maxthrottle>     1.0 </maxthrottle>
  <minthrottle>     0.1 </minthrottle>
  <sparkfaildrop>   0.0 </sparkfaildrop>
</piston_engine>
```

```
<?xml version="1.0"?>

<propeller name="MTV-21-A-C-F">
  <ixx unit="KG*M2"> 0.3 </ixx>
  <diameter unit="M">  1.78 </diameter>
  <numblades> 2 </numblades>
  <minpitch> 10.0 </minpitch>
  <maxpitch> 30.0 </maxpitch>


  <table name="C_THRUST" type="internal">
          <tableData>
                  0.40000       0.10791
                  0.50044       0.10426
                  0.59935       0.099004
                  0.69968       0.093108
                  0.80003       0.086684
                  0.89901       0.08017
                  0.99801       0.07396
                  1.09840       0.067659
                  1.19880       0.061796
                  1.30000       0.056902

     </tableData>
  </table>


  <table name="C_POWER" type="internal">
     <tableData>
         0.40000        0.052271
         0.50044        0.063186
         0.59935        0.071859
         0.69968        0.078893
         0.80003        0.083984
         0.89901        0.087283
         0.99801        0.089389
         1.0984         0.09
         1.1988         0.089717
         1.3            0.089583
     </tableData>
  </table>

</propeller>
```

Vehicle configuration



- Fuel tanks
- Thrusters
- Flight Control System (FCS)

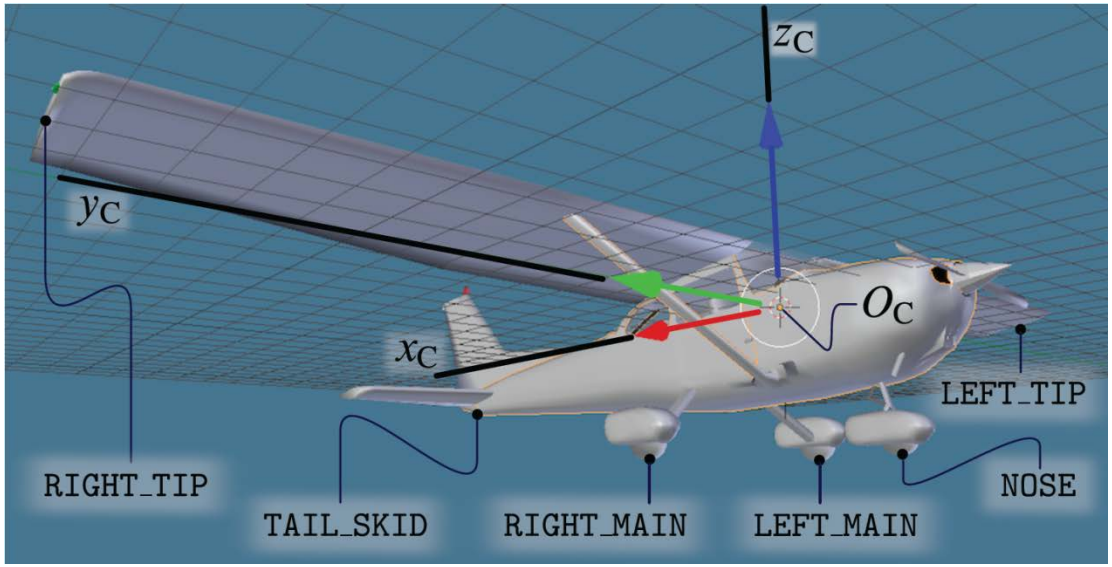


Vehicle configuration

FCS:

```
<flight_control name="FCS: p2006t">

    <channel name="Pitch">

        <summer name="Pitch Trim Sum">
            <input>fcs/elevator-cmd-norm</input>
            <input>fcs/pitch-trim-cmd-norm</input>
            <clipto>
                <min>-1</min>
                <max> 1</max>
            </clipto>
        </summer>

        <aerosurface_scale name="Elevator Control">
            <input>fcs/pitch-trim-sum</input>
            <gain>0.01745</gain>
            <range>
                    <min>-15</min>
                    <max> 4</max>
            </range>
            <output>fcs/elevator-pos-rad</output>
            </aerosurface_scale>

            <aerosurface_scale name="Elevator Position Normalized">
            <input>fcs/elevator-pos-deg</input>
            <domain>
                    <min>-15</min>
                    <max> 4</max>
            </domain>
            <range>
                <min>-1</min>
                <max> 1</max>
            </range>
            <output>fcs/elevator-pos-norm</output>
        </aerosurface_scale>

    </channel>
```

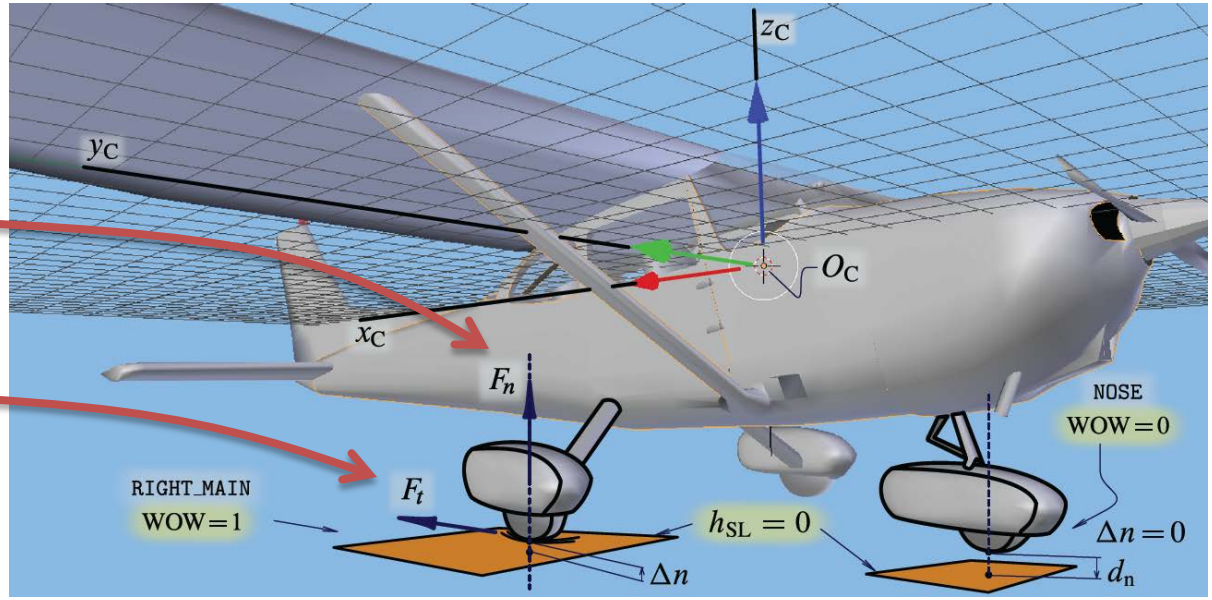
Vehicle configuration



Landing gears and terrain contact

$$F_n = k\,\Delta n + b\,\frac{d\Delta n}{dt}$$

$$F_t = \mu\,F_n$$



Sim configuration

Initialization file:

```
<?xml version="1.0"?>
<initialize name="myreset">
  <!--
  This file sets up the aircraft @ 7000 ft
 altitude; @236 ft/s = 140 knots  (cruise speed);
  @ Naples.
  -->
  <ubody unit="FT/SEC">  202.5  </ubody>
  <vbody unit="FT/SEC">    0.0  </vbody>
  <wbody unit="FT/SEC">    0.0  </wbody>
  <latitude unit="DEG">   40.89 </latitude>
  <longitude unit="DEG">  14.28 </longitude>
  <phi unit="DEG">         0.0  </phi>
  <theta unit="DEG">       0.0  </theta>
  <psi unit="DEG">       150.0  </psi>
  <altitude unit="FT">  2320.0  </altitude>
</initialize>
```

Sim configuration

JSBSim Script file

AC model and initialization file selection

Initial, final time, integration interval

Event scheduling

Example of scripted trim
(default simple algorithm)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="http://jsbsim.sourceforge.net/JSBSimScript.xsl"?>
<runscript xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://jsbsim.sf.net/JSBSimScript.xsd"
    name="P2006T test">

    <use aircraft="p2006tv6" initialize="myreset"/>
    <run start="0.0" end="100" dt="0.0083333">

        <property> simulation/notify-time-trigger </property>
        <property value="1"> simulation/run_id </property>

        <event name="trim e setaggio comandi">
                <description>trim the aircraft</description>
                <condition>
                        simulation/sim-time-sec ge 0.0
                </condition>

                <set name="fcs/mixture-cmd-norm[0]" value="1.0"/>
                <set name="fcs/mixture-cmd-norm[1]" value="1.0"/>
                <set name="propulsion/magneto_cmd" value="3"/>
                <set name="fcs/throttle-cmd-norm[0]" value="0.0"/>
                <set name="fcs/throttle-cmd-norm[1]" value="0.0"/>
                <set name="propulsion/starter_cmd" value="1"/>
                <set name="fcs/mixture-cmd-norm" value="1.0"/>
                <set name="fcs/throttle-cmd-norm" value="1.0"/>
                <set name="simulation/do_simple_trim" value="0"/>
        </event>

         <event name="elevator step">
         <description>azione sul comando dell'elevatore</description>
         <condition>
              simulation/sim-time-sec >= 0.05
       </condition>

         <set name="fcs/elevator-cmd-norm" value="-0.2" action="FG_RAMP" tc="0.5"/>
         </event>

         <event name="remove">
         <description>rilascia l'elevatore</description>
         <condition>
              simulation/sim-time-sec >= 10.2
         </condition>

         <set name="fcs/elevator-cmd-norm" value="-0.02" action="FG_RAMP" tc="0.8"/>
         </event>
  </run>
</runscript>
```

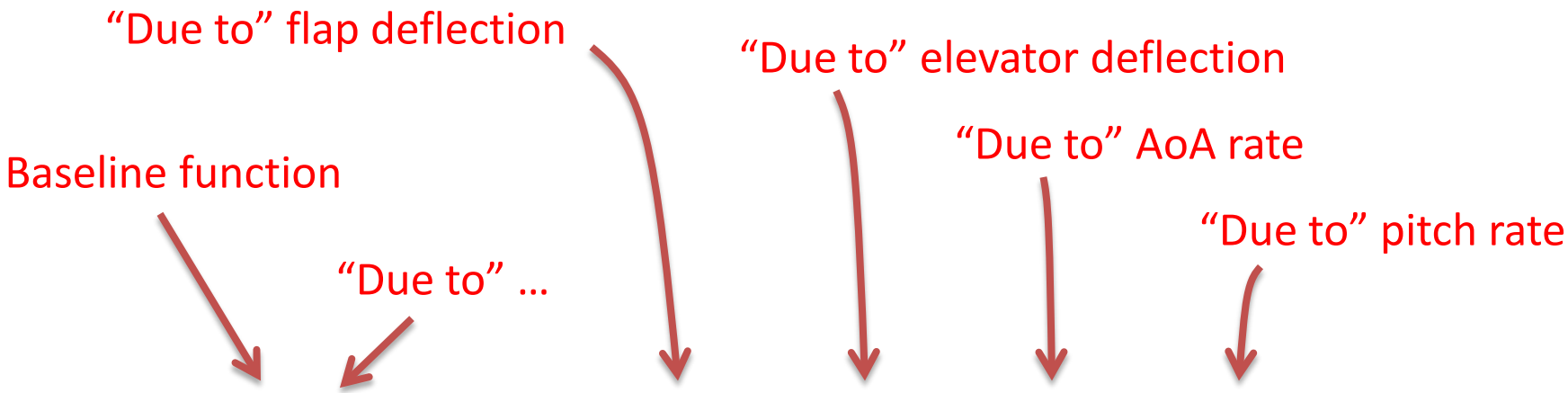
Functions

Arbitrary functions can be defined in XML. Functions are used to:

- Define aerodynamics forces and moments
- Define specific control system algorithms
- Set property values in scripted events

Functions support most routines available in the C language standard library (e.g. cos, sin, tan, exp. pow, mod, etc.)

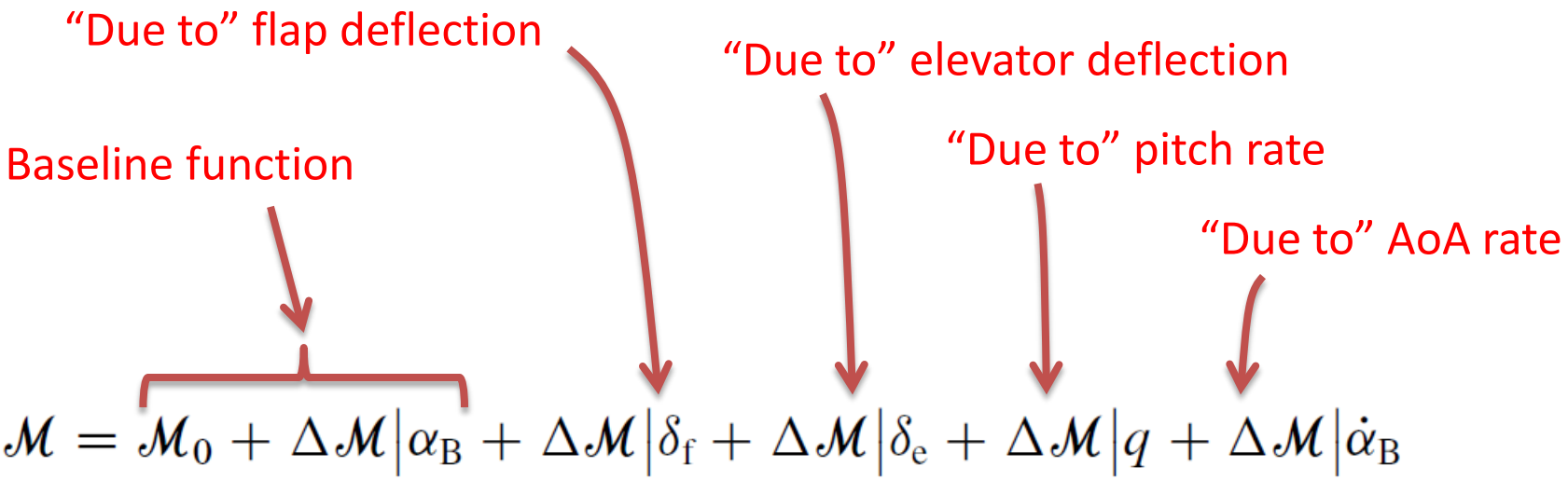
Example: Lift build-up



$$L = L_{\text{basic}}\left|\left(\alpha_{\text{B}}, \phi_{\text{hyst}}\right) + \Delta L\right|\delta_{\text{f}} + \Delta L\left|\delta_{\text{e}} + \Delta L\right|\dot{\alpha}_{\text{B}} + \Delta L\left|q\right.$$

$$= \bar{q}_{\infty} S \left[C_{L,\text{basic}}\left(\alpha_{\text{B}}, \phi_{\text{hyst}}\right) + \Delta C_L\left(\delta_{\text{f}}\right) + \Delta C_L\left(\delta_{\text{e}}\right) + \Delta C_L\left(\dot{\alpha}_{\text{B}}\right) + \Delta C_L\left(q\right)\right]$$

Example: Pitching Mom. build-up



$$\mathcal{M} = \mathcal{M}_0 + \Delta\mathcal{M}\big|\alpha_\mathrm{B} + \Delta\mathcal{M}\big|\delta_\mathrm{f} + \Delta\mathcal{M}\big|\delta_\mathrm{e} + \Delta\mathcal{M}\big|q + \Delta\mathcal{M}\big|\dot{\alpha}_\mathrm{B}$$

$$= \bar{q}_\infty S\,\bar{c}\left[C_{\mathcal{M}0} + C_{\mathcal{M}_\alpha}\sin\alpha_\mathrm{B} + \Delta C_\mathcal{M}(\delta_\mathrm{f}) + C_{\mathcal{M}_{\delta_\mathrm{e}}}\delta_\mathrm{e}\right.$$

$$\left. + C_{\mathcal{M}_q}\frac{q\bar{c}}{2V} + C_{\mathcal{M}_{\dot{\alpha}_\mathrm{B}}}\frac{\dot{\alpha}_\mathrm{B}\bar{c}}{2V}\right]$$

XML Function Example: Aero

```
<function name="aero/moment/Df">
  <description> Pitch moment due to flap deflection </description>
  <product>
    <property> aero/qbar-area </property>
    <property> metrics/cbarw-ft </property>
    <table>
      <independentVar> fcs/flap-pos-deg </independentVar>
      <tableData>
         0.0   0.0
        10.0  -0.0654
        20.0  -0.0981
        30.0  -0.1140
      </tableData>
    </table>
  </product>
</function>
```

Represents: $M_{\delta f} = qbar \bullet S_w \bullet cbar \bullet C_{m\delta f}$

XML Function Example: Aero

```
<function name="aero/moment/Df">
  <description> Pitch moment due to flap deflection </description>
  <product>
    <property> aero/qbar-area </property>
    <property> metrics/cbarw-ft </property>
    <table>
      <independentVar> fcs/flap-pos-deg </independentVar>
      <tableData>
         0.0   0.0
        10.0  -0.0654
        20.0  -0.0981
        30.0  -0.1140
      </tableData>
    </table>
  </product>
</function>
```

Represents: $M_{\delta f} = qbar \cdot S_w \cdot cbar \cdot C_{m\delta f}$

XML Function Example: Aero

```
<function name="aero/moment/Df">
  <description> Pitch moment due to flap deflection </description>
  <product>
    <property> aero/qbar-area </property>
    <property> metrics/cbarw-ft </property>
    <table>
      <independentVar> fcs/flap-pos-deg </independentVar>
      <tableData>
         0.0   0.0
        10.0  -0.0654
        20.0  -0.0981
        30.0  -0.1140
      </tableData>
    </table>
  </product>
</function>
```

Represents: $M_{\delta f} = qbar \cdot S_w \cdot cbar \cdot C_{m\delta f}$

XML Function Example: Aero

```
<function name="aero/moment/Df">
  <description> Pitch moment due to flap deflection </description>
  <product>
    <property> aero/qbar-area </property>
    <property> metrics/cbarw-ft </property>
    <table>
      <independentVar> fcs/flap-pos-deg </independentVar>
      <tableData>
         0.0   0.0
        10.0  -0.0654
        20.0  -0.0981
        30.0  -0.1140
      </tableData>
    </table>
  </product>
</function>
```

Represents: $M_{\delta f} = qbar \bullet S_w \bullet cbar \bullet C_{m\delta f}$

XML Function Example: Aero

```
<function name="aero/moment/Df">
  <description> Pitch moment due to flap deflection </description>
  <product>
    <property> aero/qbar-area </property>
    <property> metrics/cbarw-ft </property>
    <table>
      <independentVar> fcs/flap-pos-deg </independentVar>
      <tableData>
         0.0   0.0
        10.0  -0.0654
        20.0  -0.0981
        30.0  -0.1140
      </tableData>
    </table>
  </product>
</function>
```

Represents: $M_{\delta f} = qbar \bullet S_w \bullet cbar \bullet C_{m\delta f}$

XML System Function Example

```
<fcs_function name="guidance/angle-to-roll-rad“>
  <function>
    <acos>
      <sum>
        <product>
          <p> guidance/x1a </p>
          <p> guidance/x2a </p>
        </product>
        <product>
          <p> guidance/y1a </p>
          <p> guidance/y2a </p>
        </product>
      </sum>
    </acos>
  </function>
</fcs_function>
```

XML Script Function

```
<event name="Patrick Atmosphere Temperature and Pressure" continuous="tru
  <description> Implements a Patrick atmosphere by overriding temp. and p
  <condition> simulation/sim-time-sec ge 0.0 </condition>
  <set name="atmosphere/override/temperature">
    <function name="atmosphere/override/temperature-function">
      <sum>
        <table name="atmosphere/override/temperature-mean">
          <independentVar lookup="row"> position/h-agl-ft </independentVa
          <independentVar lookup="column"> simulation/month </independent
          <tableData>
                <!-- This data is from the Patrick AFB Range Reference At
                <!--     Annual    January   February     March      Apri
                          0          1          2          3          4
                  0.0    529.128    518.220    517.824    522.468    527.
                  9.8    528.480    517.788    517.446    522.090    527.
               6562.0    511.776    505.674    504.846    507.528    510.
                  …          …          …          …          …          …
             229670.0    394.794    397.080    398.826    399.474    392
          </tableData>
        </table>
```

Flight Control System Components in JSBSim

```
<sensor name="fcs/attitude/sensor/phi-rad">
  <input> attitude/phi-rad </input>
  <lag> 0.5 </lag>
  <noise variation="PERCENT"> 0.05 </noise>
  <quantization name="attitude/sensor/quantized/phi-rad">
    <bits> 12    </bits>
    <min>  -3.14 </min> <!-- -180 degrees -->
    <max>   3.14 </max> <!-- +180 degrees -->
  </quantization>
  <bias> 0.001 </bias>
</sensor>


<pid name="fcs/roll-ap-error-pid">
  <input> fcs/attitude/sensor/phi-rad </input>
  <kp> 2.0 </kp>
  <ki> 0.2</ki>
  <kd> 1 </kd>
</pid>
```

Flight Control System Components in JSBSim

```
<sensor name="fcs/attitude/sensor/phi-rad">
  <input> attitude/phi-rad </input>
  <lag> 0.5 </lag>
  <noise variation="PERCENT"> 0.05 </noise>
  <quantization name="attitude/sensor/quantized/phi-rad">
    <bits> 12    </bits>
    <min>  -3.14 </min> <!-- -180 degrees -->
    <max>   3.14 </max> <!-- +180 degrees -->
  </quantization>
  <bias> 0.001 </bias>
</sensor>


<pid name="fcs/roll-ap-error-pid">
  <input> fcs/attitude/sensor/phi-rad </input>
  <kp> 2.0 </kp>
  <ki> 0.2</ki>
  <kd> 1 </kd>
</pid>
```

Complex Functions

```
<fcs_function name="guidance/heading-to-waypoint">
  <function>
    <atan2> <!-- atan2 (deltaY, deltaX )-->
      <product>
        <sin><property> fcs/delta-lon-rad </property></sin>
        <cos><property> ap/wp_latitude_rad </property></cos>
      </product>
      <difference>
        <product>
          <cos><property> position/lat-gc-rad </property></cos>
          <sin><property> ap/wp_latitude_rad </property></sin>
        </product>
        <product>
          <sin><property> position/lat-gc-rad </property></sin>
          <cos><property> ap/wp_latitude_rad </property></cos>
          <cos><property> fcs/delta-lon-rad </property></cos>
        </product>
      </difference>
    </atan2>
  </function>
</fcs_function>
```

Output

JSBSim has a versatile data logging system.

- Any number of “output” sections can be specified in the configuration file, but it is preferred to create individual output *directive* files.
- Logical data sets can be output, and/or individual parameters.
- Output can be sent to one or many (in any combination):
    - Socket (local or remote)
    - File
    - Console
- Normally, file is sent to a CSV data file

Output Example

```
<output name="DC.csv" type="CSV" rate="30">

  <massprops>         OFF </massprops>
  <rates>             ON  </rates>
  <velocities>        ON  </velocities>
  <forces>            ON  </forces>
  <moments>           ON  </moments>
  <position>          ON  </position>
  <propulsion>        ON  </propulsion>
  <aerosurfaces>      OFF </aerosurfaces>
  <aerodynamics>      OFF </aerodynamics>
  <fcs>               ON  </fcs>
  <ground_reactions> OFF </ground_reactions>
  <atmosphere>        OFF </atmosphere>

  <property> velocities/ve-kts     </property>
  <property> velocities/mach       </property>
  <property> flight-path/gamma-deg </property>

  …
</output
```

Gravity Model

JSBSim can use either a classical gravity model (proportional to $1/r^2$), or the model used with the WGS-84 datum, the Earth Gravitation Model of 1996 (EGM96 – the default). For the Dream Chaser ascent abort simulation runs, we use the latter. The implementation does not use the full set of 130,676 coefficients, but only the first coefficient, which is two orders of magnitude larger than the next coefficient.

The gravity model is selected via the following property usually set in a script (0 = standard, 1 = EGM96):

simulation/gravity-model

[Reference: Aircraft Control and Simulation, *Stevens and Lewis*]

Atmosphere Model

The atmosphere model that is standard in JSBSim is the 1976 standard atmosphere as defined in “U.S. Standard Atmosphere, 1976", NASA TM-X-74335.

We currently use the Patrick AFB Range Reference atmosphere for ascent abort runs. This is satisfactory for aborts taking place entirely within a short range around the Florida launch site.

For more geographically expansive aborts (Charleston, TAL, etc.) we plan to use the GRAM 2007 atmosphere model. GRAM is the “gold standard” atmosphere model, and is a product of NASA Marshall Spaceflight Center.

Example: Tecnam P2006T

Cockpit

Baggage compartment

Table 1 P2006T aircraft geometric characteristics

Parameter	Value	Parameter	Value
Wing span	37.40 ft (11.4 m)	Fuselage length	28.50 ft (8.7 m)
Wing area, S	159.31 ft^2 (14.8 m^2)	Cabin width	48.03 in (1.22 m)
MAC, c	4.40 ft (1.34 m)	Cabin length (with baggage)	11 ft (3.35 m)
Wing aspect ratio, AR	8.76 (8.76)	Fuselage height	9.35 ft (2.85 m)

Table 2 P2006T aircraft weights and loading

Parameter	Value
MTOW, W_{TO}	2601 lb (1180 kg)
Maximum ramp weight	2601 lb (1180 kg)
Standard equipped weight	1675 lb (760 kg)
Standard useful load	926 lb (420 kg)
Limit load factors, n	+3.8 g/ − 1.9 g

Table 3 P2006T aircraft propulsion characteristics

Parameter	Value
Engine model	Rotax 912S
Takeoff power	100 hp (73 kW)
Maximum continuous power	92.4 hp (69 kW)
Propeller (two blades, constant speed, full feathering)	MTV-21-A-C-F/CF178-05

Example: Tecnam P2006T

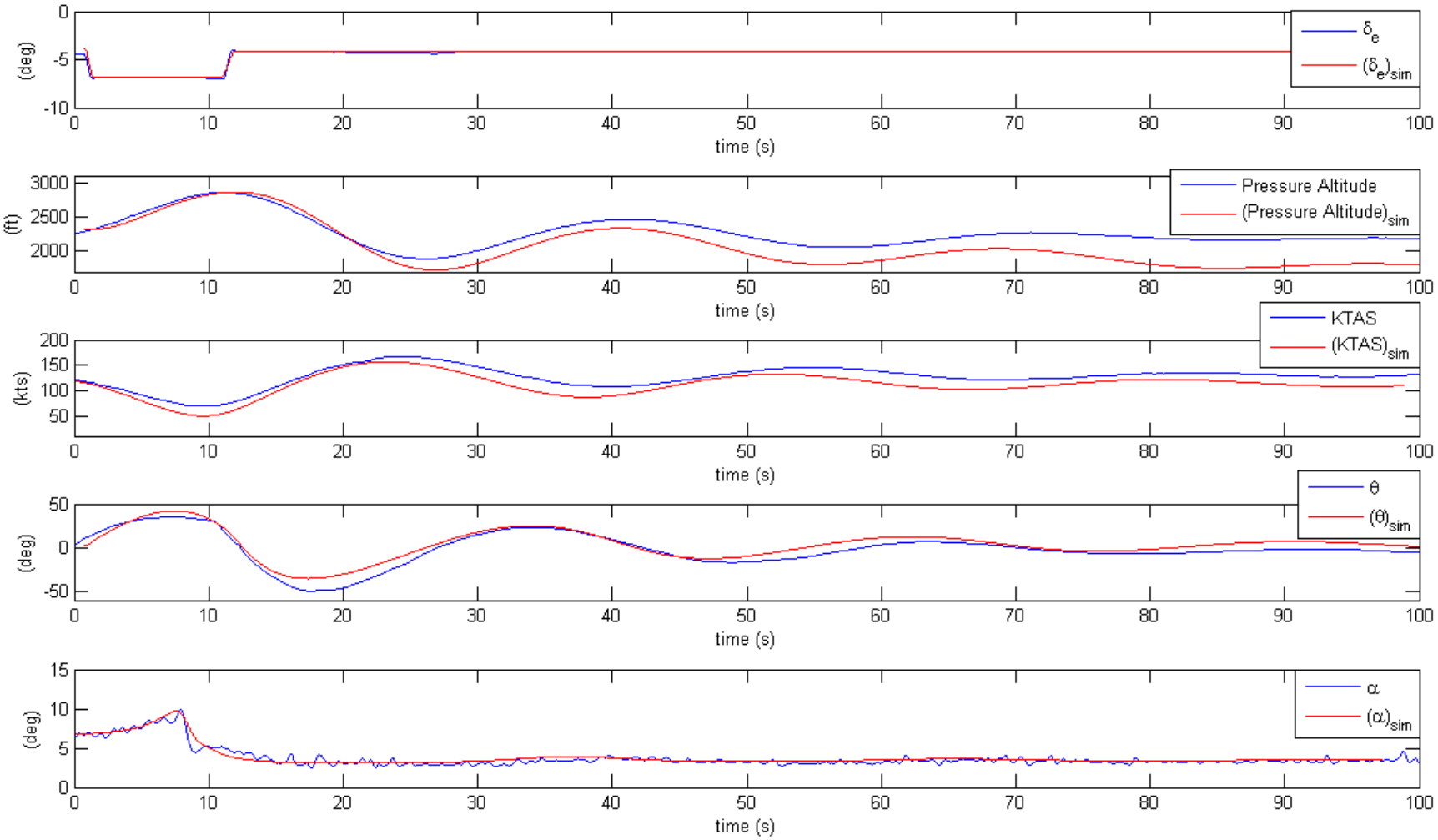
Table 5 Selected c.g. range for flight tests (useful load 420 kg)

	Max forward	Max aft
X_{cg}/c	16.5%	31%
Load	Pilot (90 kg)	Pilot (90 kg)
Condition	3 crew (270 kg)	2 crew, rear (160 kg)
	No baggage	80 kg baggage
	6- kg fuel	90 kg fuel

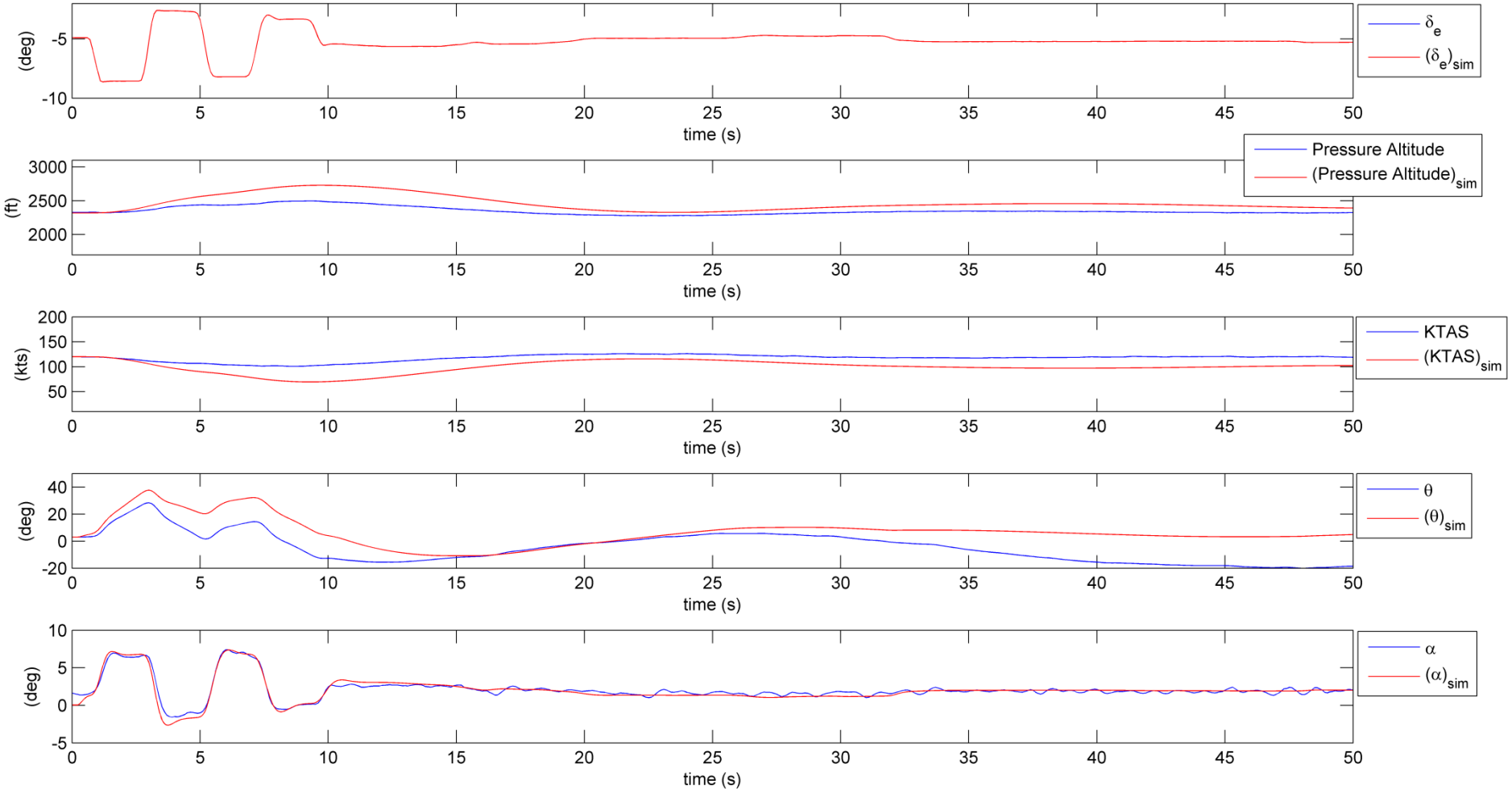
Table 4 P2006T Performances as measured from flight-certification tests

Parameter	Value
Max speed at S/L (full throttle, max RPM)	154 kt
Cruise speed (75%, 7000 ft)	145 kt
Cruise speed (65%, 9000 ft)	135 kt
Stall speed flap down	47 kt
V_A (maneuvering speed)	116 kt
V_{NE} (never exceed speed)	168 kt
Max RC, S/L	1210 ft/ min
Max RC, S/L ,OEI	350 ft/ min
Service ceiling (twin engine)	12,800 ft
Single-engine ceiling	6600 ft
Takeoff distance	1260 ft (384 m)
Takeoff ground run	968 ft (295 m)
Landing distance	1263 ft (385 m)
Landing ground run	734 ft (224 m)

Pull manouever



More elaborated command hystory



Conclusions

An easy-to-use open source software

Capability to model basic, intermediate or advanced vehicles

No need to use commercial software

Interface with FlightGear

JSBSim: www.jsbsim.org