The quarterly newsletter for JSBSim, an open source flight dynamics model in C++

Back of the Envelope

VOLUME 2, ISSUE 2
15 JULY 2005

Extensive New Features in JSBSim 0.9.9

Quite a few new features have been added to JSBSim in the past several months. Some of the new features are discussed in this issue.

Server

JSBSim has for a long time had the capability to output data *from* itself at runtime via a socket. More recently, the capability has been added to connect *to* JSBSim via socket for data input. At this time, the capability is aimed at user interaction and debugging or investigations more than interprocess communications. The aircraft config file can now interpret a new element that defines how an outside process can connect to a running instance of JSBSim:

```
<input port="port number"/>
```

A new C++ class has been added, FGInput, and the socket class, FGfdmSocket, has been modified to handle inputs as well as outputs. The JSBSim input command interpreter can understand several commands:

```
get property_name
set property_name
hold
resume
info
help
quit
```

In the case of the get command, if the property name supplied is not found, JSBSim will attempt to return the names of all properties that contain the supplied string – which provides a rudimentary search capability.

One way that this new capability can be used with JSBSim in a standalone mode is through the use of the telnet command – which is much more versatile than I knew. The JSBSim standalone application can be started with some new options: *--realtime*, and *--suspend*. These options are useful when running with the server capability. The *--realtime* option causes JSBSim to run at realtime speed. The *--suspend* option causes JSBSim to go into *hold* after initialization. This allows the user to connect from another shell to the JSBSim server, using telnet. Running JSBSim with a script and for use with the server capability can be done as follows (all on one line):

```
./jsbsim
  --script=scripts/c1723.xml
  --realtime --suspend
```

With the above command line, JSBSim will initialize and hold. A connection via telnet to JSBSim from a different shell can then be made:

```
telnet host port
```

For example:

```
telnet localhost 1137
```

JSBSim responds over the socket:

```
$ telnet localhost 1137
Trying 127.0.0.1...
Connected to zeus.
Escape character is '^]'.
Connected to JSBSim server
JSBSim>
```

Typing *info* gives important information about the simulation run:

```
JSBSim> info
JSBSim version: 0.9.9
Config File version: 2.0
Aircraft simulated: Cessna C-172
Simulation time:    0.008
```

Property values can be set or retrieved using the *get* and *set* commands. If the exact property name is unknown, the *get* command can also be used to search for all properties that contains the given string (this capability can only be used while in *hold*):

```
JSBSim> get /inertia
inertia/mass-slugs
inertia/weight-lbs
inertia/cg-x-ft
inertia/cg-y-ft
inertia/cg-z-ft
```

The value for a specific property can be retrieved using the *get* command:

```
JSBSim> get inertia/cg-x-ft
```

(Continued on page 2)

JSBSim has for a long time had the capability to output data from itself at runtime via a socket. More recently, the capability has been added to connect to JSBSim via socket for data input.

Inside this issue:

(Continued from page 1)

```
inertia/cg-x-ft =  43.886700
```

Setting a property value is done as follows (for example):

```
JSBSim> set ap/elevator_cmd 1
```

Note that the "=" symbol is not used.

Input is welcome on how this capability can be made more useful or expanded.

Sensors

Up to this point, any of the more sophisticated flight control systems defined within a JSBSim aircraft configuration use "perfect sensors" – that is, data directly from the equations of motion.

Much discussion over the past couple of months has lead to the creation of an early sensors class. Up to this point, any of the more sophisticated flight control systems defined within a JSBSim aircraft configuration use "*perfect sensors*" – that is, data directly from the equations of motion. If the flight control system (FCS) needed dynamic pressure (qbar) or angle of attack (alpha) it simply used the same values calculated (and subsequently used) by the equations of motion. This is not strictly appropriate, however, because a flight control system will use sensor inputs that often follow a circuitous path to the flight control laws themselves. The nature of the sensor may induce a lag in the signal, noise and signal drift can creep in, and the signal may need to be converted so it can be represented in a set number of bits. Lastly, the signal may be processed by a data acquisition unit that distributes the signal to various aircraft systems (flight control, display instruments, telemetry, etc.) at a particular rate, and so signals are updated at a specific rate and perhaps are also delayed by transport lag.

The new FGSensors class addresses most of these. The flight control system specification works as before, but one now has the option of inserting a sensor object into the FCS definition (see Fig. 1).

```
<sensor name="name">
  <input> property </input>
  <lag> number </lag>
  <noise variation="PERCENT|ABSOLUTE"> number </noise>
  <quantization name="name">
    <bits> number </bits>
    <min> number </min>
    <max> number </max>
  </quantization>
  <drift_rate> number </drift_rate>
  <bias> number </bias>
</sensor>
```

Example:

```
<sensor name="aero/sensor/qbar">
  <input> aero/qbar </input>
  <lag> 0.5 </lag>
  <noise variation="PERCENT"> 2 </noise>
  <quantization name="aero/sensor/quantized/qbar">
    <bits> 12 </bits>
    <min> 0 </min>
    <max> 400 </max>
  </quantization>
  <bias> 0.5 </bias>
</sensor>
```

Figure 1. Sensor definition for JSBSim (v0.9.9)

The only required element in the sensor definition is the input element. In that case, no degradation would be modeled, and the output would simply be the input.

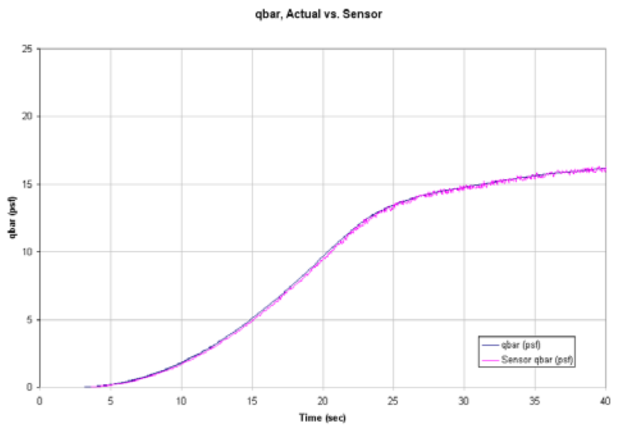


Figure 2. Sensor output compared with "actual" for qbar

For noise, if the type is PERCENT, then the value supplied is understood to be a percentage variance. That is, if the number given is 0.05, the variance is understood to be ±0.05 percent maximum variance. So, the actual value for the sensor will be <u>anywhere</u> from 0.95 to 1.05 of the actual "perfect" value at any time - even varying all the way from 0.95 to 1.05 in adjacent frames - whatever the delta time.

The sensor class is also the first class that features built-in malfunction flags. A sensor can be

(Continued on page 3)

About this newsletter ...

Edited by Jon Berndt

"*Back of the Envelope*" is a communication tool written generally for a wider audience than core JSBSim developers, including instructors, students, and other users. The articles featured will likely tend to address questions and comments raised in the mailing lists and via email. If you would like to suggest (or even author) an article for a future issue, please email the editor at: jsb@hal-pc.org.

(Continued from page 2)

failed low, high, or stuck. The implementation of sensors, functions, malfunctions, etc. open up a wide variety of possibilities for JSBSim. For instance, it is now conceivable that 3 identical *strings* for a set of flight control laws could be defined, with comparison logic that throws out bad sensor values, and malfunction logic that exercises the comparison logic. This class will be documented in more detail in the future and posted on the web site and in the header.

New Flight Control System Component: FCS Function

There may be times when none of the individual flight control components in the set of JSBSim FCS components will suffice for a particular purpose. In that event, a new FCS component has been created, the function component FGFCSComponent. The function component allows an arbitrary function to be represented in a flight control specification. The XML format of the component is as follows (optional items in [square brackets]):

```
<component name="name" type="FUNCTION">
  [<input> property </input>]
  <function name="name">
    … function body
  </function>
  [<output> property </output>]
  [<clipto>
    … clipping limits
  </clipto>]
</component>
```

If there is no input value supplied, the value of the function is calculated, and the function can be used elsewhere as needed. If an input value is supplied, then the component acts as a calculated gain function, with the input multiplied by the function value to arrive at the component output value.

Real-Time Operation

Up to this point, the JSBSim standalone executable has been a *batch* executable only that runs as fast as the CPU will allow it to. In some situations it is desirable to run in real-time. For instance, an application might require JSBSim to output a signal via a socket, perhaps using JSBSim environment data to *stimulate* a HUD or other device. The JSBSim executable can now be told to run at real-time speed with the use of the *--realtime* option:

```
./jsbsim.exe --script=scripts/c1723.xml --realtime
```

The JSBSim executable can now be told to run at real-time speed with the use of the --realtime option.

Multiple Output Streams

JSBSim now allows multiple <output></output> elements to be specified in a configuration file. Multiple output definitions enable multiple output streams. For instance, one might want a several specific outputs to be transmitted to specific devices, and have a set of data sent to an output file for later analysis. This arrangement would be defined as in the example below:

```
<output name="OutputFile" type="CSV" rate="20">
    <ground_reactions> ON </ground_reactions>
</output>
<output name="localhost" type="SOCKET" port="1138" rate="2">
    <property> position/h-agl-ft </property>
    <property> velocities/vc-kts </property>
    <property> attitude/phi-rad </property>
    <property> fcs/attitude/sensor/phi-rad </property>
</output>
<output name="localhost" type="SOCKET" port="1139" rate="1">
    <velocities> ON </velocities>
</output>
```

(Continued on page 4)

The cornering friction coefficient can now be specified as a function.

(Continued from page 3)

Landing Gear Up-Damping

Some landing gear struts feature different damping characteristics depending on whether the strut is compressing or extending. The landing gear definition now permits the specification of a different damping coefficient for strut extension (uncompression) using the *damping_coeff_rebound* element keyword:

```
<damping_coeff_rebound unit="LBS/FT/SEC">3200</damping_coeff_rebound>
```

Landing Gear Side Force Table Specification

As a first step in improving the landing gear model, a new capability has been added to the landing gear specification. The cornering friction coefficient can now be specified as a function. It is important to note that a function definition *can* consist of only a table – it is expected that a lookup table will define the landing gear side force function in most cases. Example:

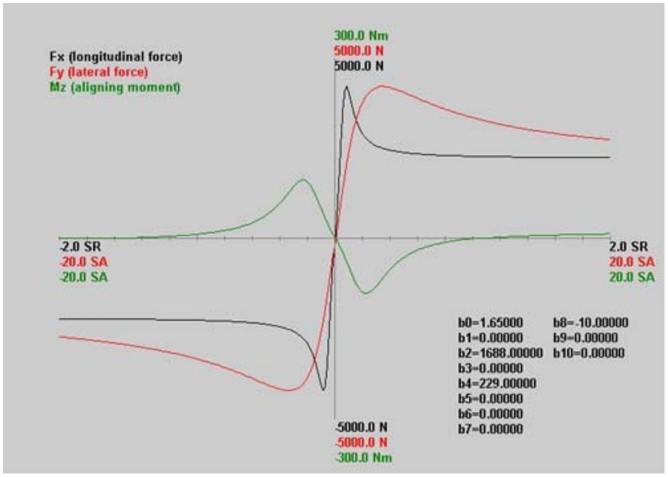
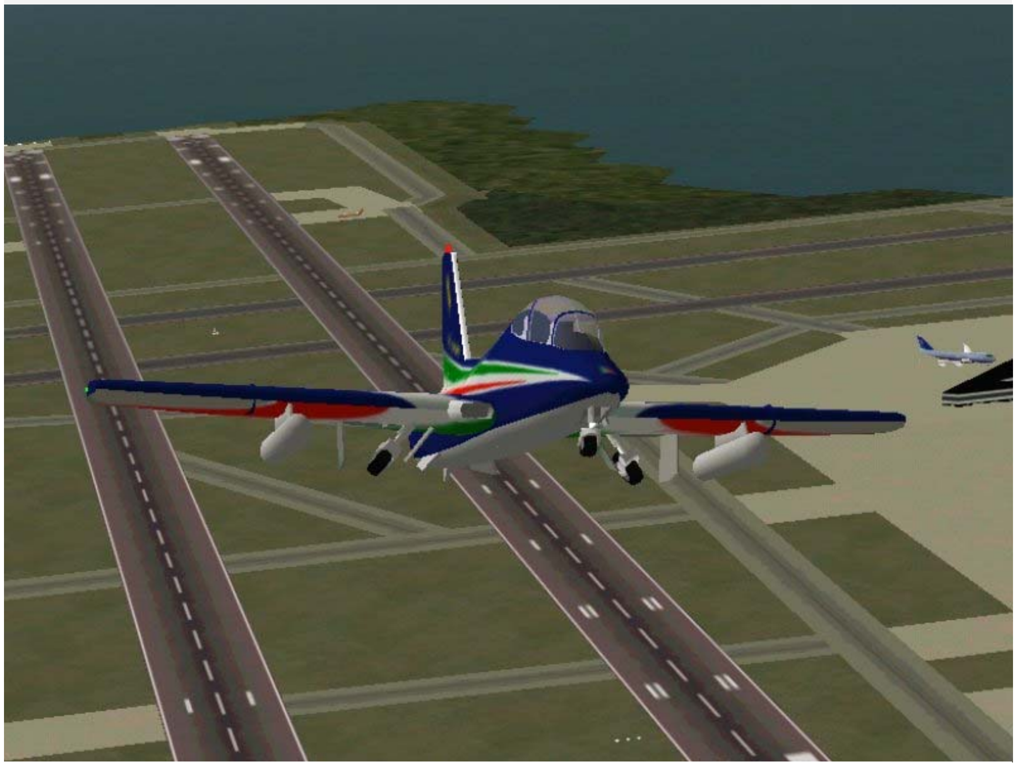


Figure 3. Pacejka curves. The cornering force is the red curve.

```
<contact type="BOGEY" name="NOSE">
  <location unit="IN">
      <x> -6.8 </x>
      <y> 0.0 </y>
      <z> -20.0 </z>
  </location>
  <static_friction>0.8</static_friction>
  <dynamic_friction>0.5</dynamic_friction>
  <function type="CORNERING_COEFF">
    <table>
      <independentVar>gear/slip-angle-deg</independentVar>
      <tableData>
        …
      </tableData>
    </table>
  </function>
  <rolling_friction>0.02</rolling_friction>
  <spring_coeff unit="LBS/FT">1800</spring_coeff>
  <damping_coeff unit="LBS/FT/SEC">600</damping_coeff>
  <max_steer unit="DEG">10</max_steer>
  <brake_group>NONE</brake_group>
  <retractable>0</retractable>
</contact>
```

Note that the Pacejka curve shown in Figure 3 (above) shows *forces* instead of coefficients. The graph shown is for a race car application, but in an aircraft each gear may have a widely varying normal force. So, the curve for an aircraft will be for the coefficient, rather than the force. Multiplying the side force coefficient by the normal force gives the actual side force.


MB-339 PAN model takes off

Frecce Tricolori MB339 PAN Jet Model

A joint project between the HCI Lab of the University of Udine and the aerobatic team of the Italian Air Force (the Frecce Tricolori) has produced a detailed, flyable model of the MB-339 PAN jet. The leader of the Frecce Tricolori (Capt. Tammaro) was a member of the development team and beta-tester of the model. Development took about 1 year, at the end of which the Frecce Tricolori officially approved the public release of the model.

MB-339 PAN is a free multi-platform simulation of the jet used by the aerobatic team of the Italian Air Force. It runs on Windows, Linux, Mac, SGI, Solaris and is based on the FlightGear Open Source Software. The JSBSim flight model was created initially using Aeromatic and then further refined.

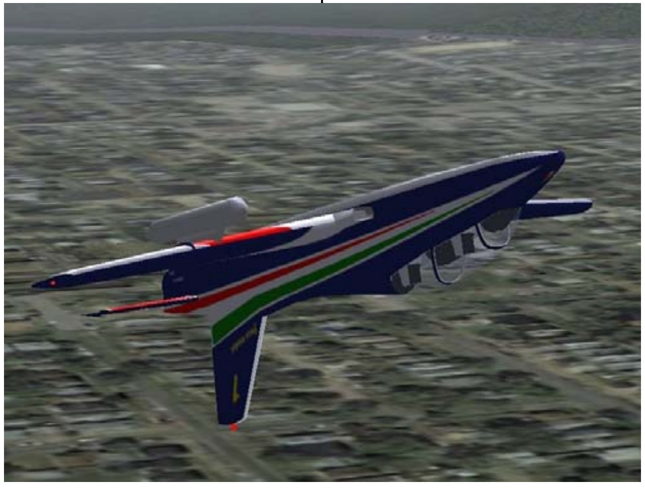
Features:

- Accurate flight dynamics and engine modeling
- About 10000 polygons for the aircraft geometry with lots of animated parts:
  - Flaps, ailerons, airbrake, gear and thruster.
- 5000 additional polygons for the full 3D virtual cockpit:
  - Stick, throttle, flaps, parking brakes and gear levers, including the rudder pedals.
  - AHI, AOA, HSI, altimeter, climb rate, acceleration, airspeed and Mach indicator.
  - Engine instruments and other gauges.

You can download the model and see more screenshots here:

http://hcilab.uniud.it/pan

(Continued on page 6)


MB339 PAN model in inverted flight

Aermacchi MB-339 PAN

Dimensions
Length 36.8 ft 11.2 m
Height 12.8 ft 3.9 m
Wing span 36.8 ft 11.2 m
Wing area 207.7 sq ft 19.3 m^2

Masses
Empty Mass 7350 lbs 3334 kg
Clean T.O.M. 10910 lbs 4950 kg
Max T.O.M. 14000 lbs 6350 kg
Max wpn load 4000 lbs 1815 kg

Fuel
Internal tanks 3154 lbs 1430 kg
External tanks 1151 lbs 522 kg

Engine
Type Rolls Royce Viper 632 turbojet
Thrust 4000 lbs 1815 kg

Structural Limits
Max Speed 500 Kias
Load Factor +8 -4.0

Clean Performance
T.O. ground run 2035 ft 620 m
ROC (SL-ISA) 5350 ft/min 27 m/s
Service ceiling 45000 ft 13700 m
Max level speed 470 Kts
Max sustained load factor 5.1 g
Max sust. turn rate (15K') 12°/sec
Approach speed at 50' alt. 102 kt
Landing ground run 1575 ft 480 m

A lot of information about the L410 can be obtained at the Letecke Zavody web site:

http://www.let.cz

Also, the FAA has issued a Type Certificate Data Sheet (TCDS) for the L420 that has some detailed information for the L410. See:

http://www.airweb.faa.gov

TCDS number: A42CE.

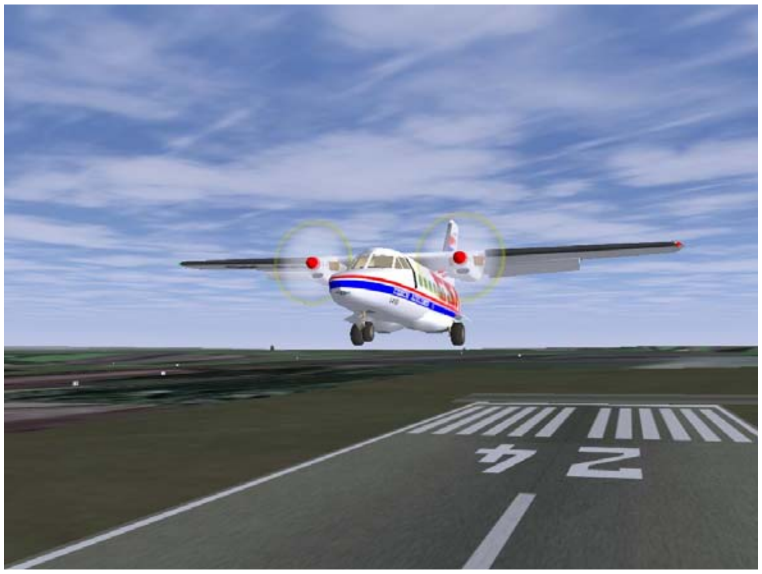
About the L410 Turbolet (from Airliners.net):

"The L410 is a very successful Czech commuter which was first built in response to Soviet requirements, but has sold widely around the globe.

The first design studies of the original 15 seat L410 began in 1966. The resulting conventional design was named the Turbolet, and was developed to be capable of operations from unprepared runway strips. The powerplant chosen was the all new Walter or Motorlet M 601, but this engine was not sufficiently developed to power the prototypes, and Pratt & Whitney Canada PT6-A27's were fitted in their place. The first flight occurred on April 16 1969, and series production began in 1970. Initial production L410's were also powered by the PT6A, and it was not until 1973 that production aircraft L410M's featured the M 601.

(Continued from page 5)

L410 Turbolet



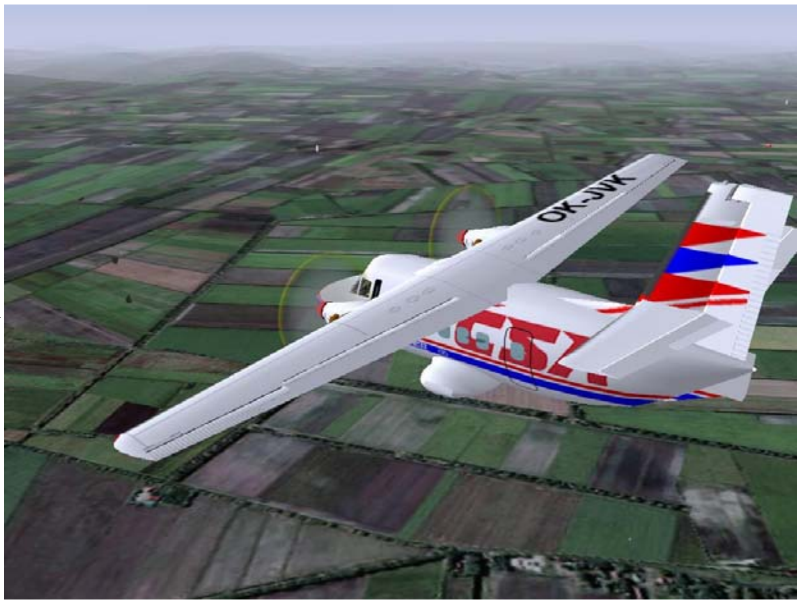
A beautifully detailed model of the Letecke Zavody L410 Turbolet commuter aircraft has been created for FlightGear/JSBSim by the father/son team of Jiri Javurek and Jiri Javurek of the Czech Republic. This is one of the finest and most complete 3D models for FlightGear that this writer has seen. As for the JSBSim flight model, this model also features the most sophisticated use of the JSBSim flight control components this writer has ever seen.

To use this aircraft model, you must incorporate some code changes that the authors have made into your version of FlightGear. The code changes are included in the tar archive, which can be downloaded at the authors' web site here:

http://javky.rozhled.cz/index.php?id=fgfsl410&typ=P

One of the changes in this model is the addition of a new class to simulate turboprop engine. Jiri Javurek describes their design process:

When we began to write the L410 model for FlightGear our goal was to make it as realistic as possible. JSBSim is a great flight dynamics model, but there wasn't an appropriate turboprop engine model. The piston engine had very different behavior and the turbine engine was difficult to connect to a propeller thruster. We decided to write a new turboprop model for small two shaft engines like the PT6 from Pratt & Whitney Canada or the Walter M601.



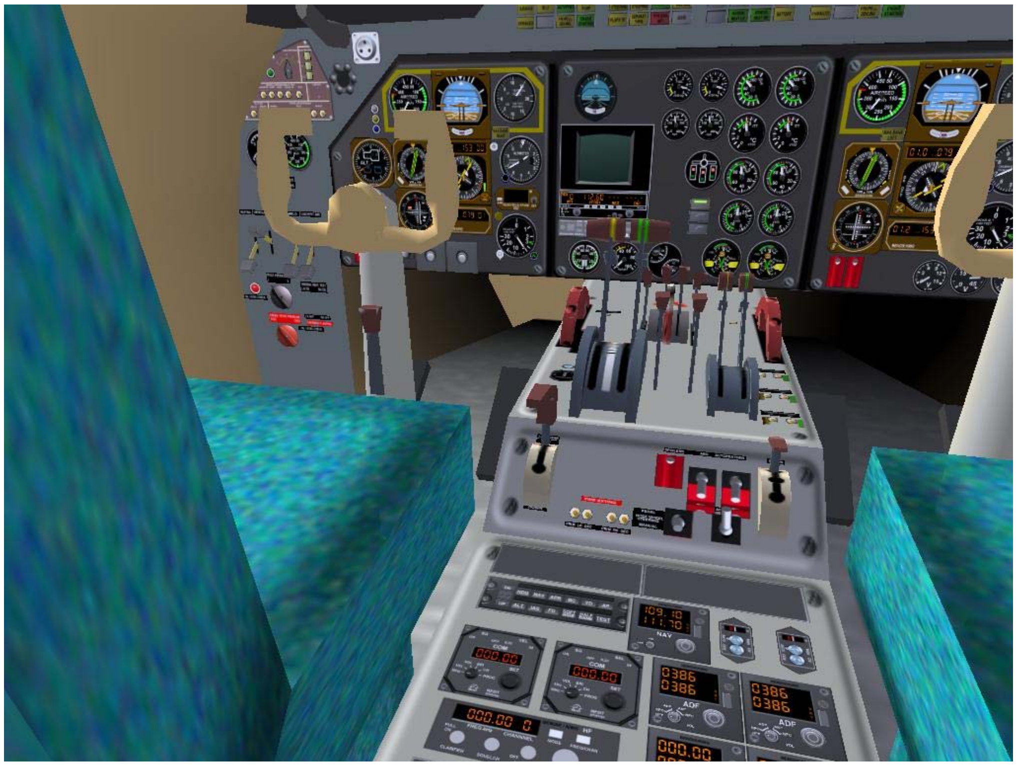
The engine is a two shaft engine. On the first shaft there is a compressor. This shaft is powered by the first turbine rotor (together called a gas generator). On the second shaft there is the second turbine rotor and this shaft (through a speed reducer) powers the propeller. You can see detailed parameters at http://www.walterengines.com.

We have obtained detailed parameters for the M601 and have rewritten the SimTurbine class, adding some features to the propeller in the process.

The main features of the complete system of engine and propeller are described.

The engine with propeller can work in 3 ranges:

Normal - the pitch of propeller blades is controlled by an automatic regulator to reach constant speed.
Beta range - small pitch angles for taxiing and negative angle for braking.
Feather position.



The main output values are:

- Speed of the compressor shaft.
- Speed of the propeller.
- Torque of the propeller shaft
- Temperature behind the first turbine
- Oil pressure
- Oil temperature
- Fuel flow
- Engine-is-starting state (with a timer)

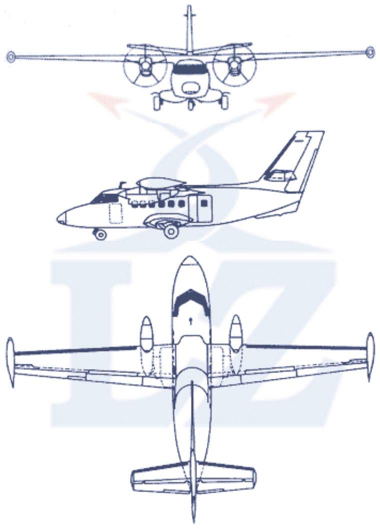
These features enable simulation of a lot of different situations and failures:

Manual feathering, Automatic feathering (used at take-off), Emergency feathering, Reversing, Slow taxiing, Intervention of electronic limiter by overrun of torque and so on.

Planned features: Extraordinary increase mode (used for the case when one engine malfunctions during take off). Water injection used for increasing power by high temperatures.

We hope that the new engine model could also be useful for additional engines and aircrafts.

Letecke Zavody L410 Turbolet



Dimensions
Length 47.5 ft 14.5 m
Height 19.1 ft 5.8 m
Wing span 63.9 ft 19.5 m
Wing area 378.7 sq ft 35.2 m^2

Masses
Empty Mass 9300 lbs 4225 kg
Max T.O.M. 14500lbs 6600 kg

Engine
Type Walter M601B Turboprop
Thrust 544ekW (730eshp)

Clean Performance
T.O. ground run 1850 ft 565 m
Service ceiling 19700 ft 6000 m
Max level speed 197 Kts
Landing ground run 1400 ft 425 m

Students and Industry Opportunities with JSBSim?

There is a new cooperative effort in the beginning stages, one to allow people in industry to assist students at various learning institutions. Facility involvement is also highly desired.

Most students graduating from an institution of higher education, with either an undergraduate or graduate level degree, are required to complete a senior project. Often, the students struggle with ideas.

What is being set up is a place for people in industry, or just about anyone with a keyboard and an email address, to be able to formally suggest senior projects. JSBSim (with or without FlightGear) has been used in various ways in academic projects for years.

Along with suggested ideas might be guidelines and helpful hints for the students. The projects will probably involve use of FlightGear, JSBSim, and DATCOM+, along with other tools. Programming abilities will generally NOT be required, as the student should be able to complete the tasks with existing tools.

You can check the web page http://www.holycows.net/students/ for more details, and a list of suggested projects so far. If you have a suggestion for a project, send email to Bill Galbraith at billg@holycows.net.

Simulate This! Heavy Lift Launch Vehicle

The Vision for Space Exploration that the U.S. National Aeronautics and Space Administration (NASA) is currently planning for is giving rise to some interesting space vehicle and rocket launcher designs. The concept presented at right features 4 solid rocket boosters (SRBs) which each produce over 3 million lbs. thrust at liftoff. This concept also features 3 or 4 space shuttle main engines (SSMEs) which each produce about 400,000 lbs. thrust at liftoff. The combined thrust of this concept would be well over 10 million lbs at liftoff—if flown this would be the most monstrous aerospace vehicle ever launched, by far.

This writer has been interested in in modeling rocket flight in JSBSim for a long time. It can be done, but modeling a full ascent of such a vehicle will require some enhancements that can hopefully be made over time. Some features that would be needed include:

- Ability to model a multi-body vehicle
- Ability to model different aerodynamic and mass properties over time
- Ability to model changing modes of flight control depending on phase of flight
- Ability to model a solid rocket booster, and its changing thrust profile, mass properties, etc.
- Ability to model gimballing rocket nozzles

Some early work was done for JSBSim to allow parent/child relationships with *spawned executive* objects. The point was reached where *child FDMs* were able to be read by the parser. Development of that feature did not advance at the time. Further definition and development of that capability will probably be put off until after JSBSim v1.0 is released as production code. Flight control, aerodynamic, and mass properties modeling for a multi-body vehicle are also conceptually defined, and quite possible.

The capabilities described above would also be useful in modeling captive/carry flights, such as a B-52 mothership carrying an X-15 research aircraft for drop-test flights.

Highlighted References

Online:

Bernard Etkin's well-known flight dynamics textbook is being updated and will be released in a new paperback edition by Dover this October (ISBN 0486445224).

Information on Pacejka's Magic Formula (tire modeling):

www.racer.nl/reference/pacejka.htm

Walter Engines:

www.walterengines.com

Visit us on the web at: www.jsbsim.org