On Some Trim Strategies for Nonlinear Aircraft Flight Dynamics Models with the Open Source Software JSBSim

James M. Goppert,[†] **Agostino De Marco**,[*] Inseok Hwang[†]

[*] JSBSim Development Team,
University of Naples "Federico II",
Department of Aerospace Engineering (DIAS), dias.unina.it
Aircraft Design & Aeroflightdynamics Group (ADAG) dias.unina.it/adag

[†] School of Aeronautics and Astronautics, Purdue University
West Lafayette, Indiana, USA engineering.purdue.edu/AAE

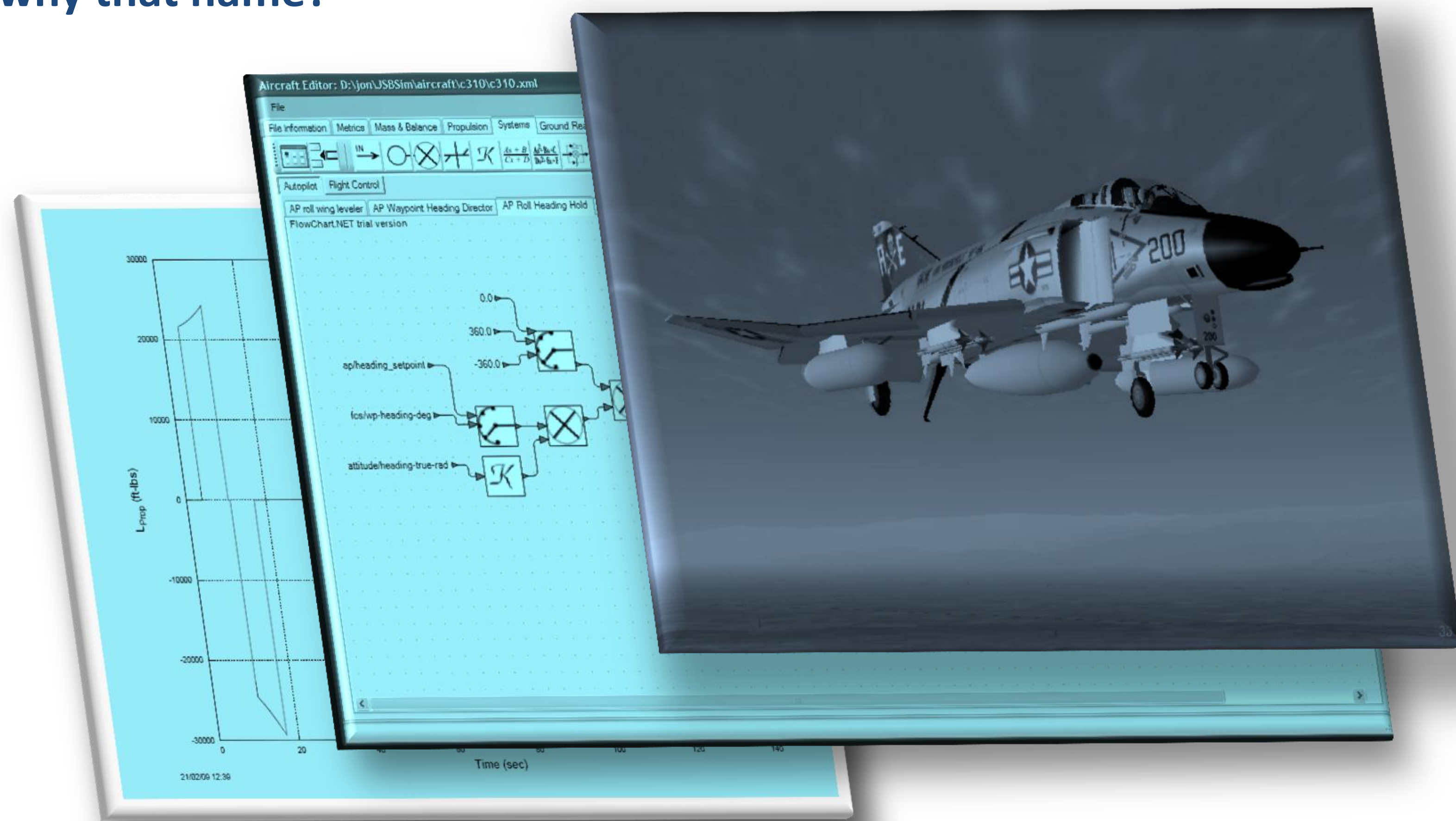
Layout of the presentation

- A quick introduction to JSBSim, an open source Flight Dynamics Model (FDM) software library
- Implementation of a Trim algorithm for JSBSim, based on a probabilistic Nelder Mead solver.
- An aicraft trimming/linearization GUI and an open source equivalent of the Matlab/Simulink aerospace toolbox.

Layout of the presentation

- **A quick introduction to JSBSim, an open source Flight Dynamics Model (FDM) software library**
- **Implementation of a Trim algorithm for JSBSim, based on a probabilistic Nelder Mead solver.**
- **An aicraft trimming/linearization GUI and an open source equivalent of the Matlab/Simulink aerospace toolbox.**

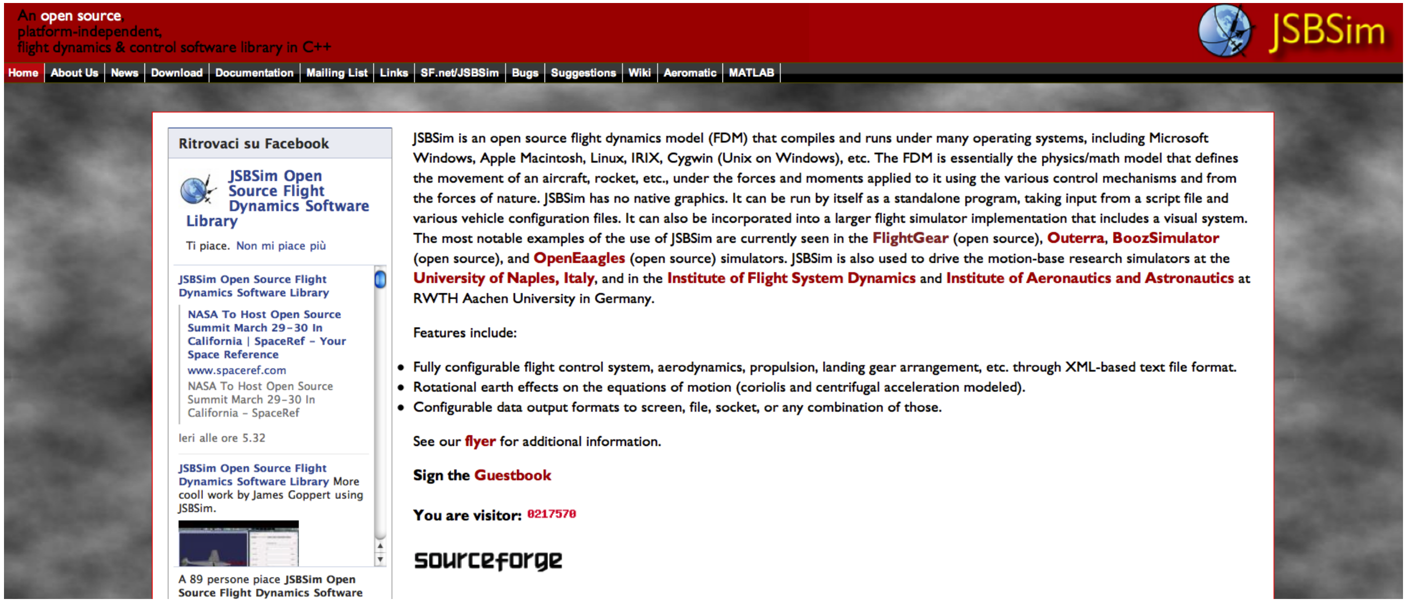
JSBSim … why that name?



Author and Development Team Lead: **Jon S. Berndt**

JSBSim

What is JSBSim?



http://www.jsbsim.org

- **Flight dynamics and control S/W library**
- **~50,000 lines of C++ code (~20,000 effective code lines)**
- **~80 C++ classes**
- **In development since 1997** (Current version effectively 1.0)
- **Data driven**
- **XML configuration files**

JSBSim

PURDUE UNIVERSITY

JSBSim 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

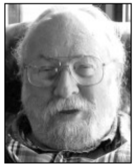
Tony has been contributing to the growth of JSBSim almost from day 1. He is responsible for integrating JSBSim with FlightGear, and for initialization and trimming. Tony also implemented David's property system into JSBSim. Tony hails from Ohio State University, with a degree in Aero and Astronautical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



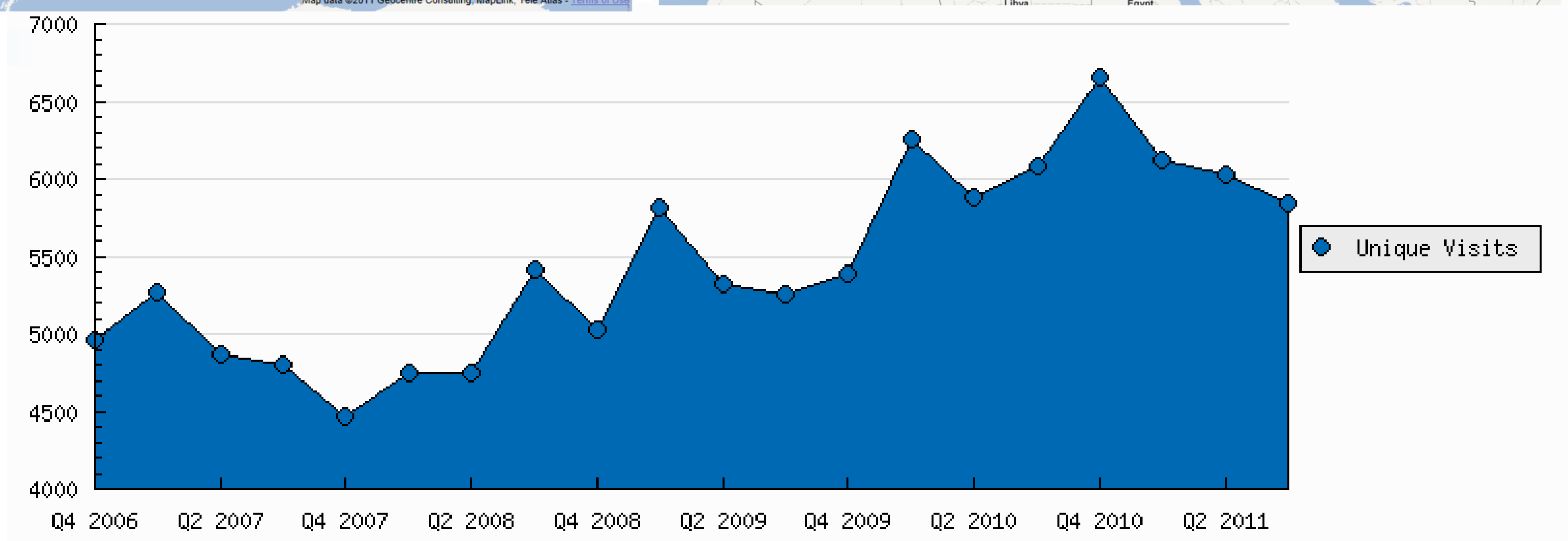
Lee Duke
(Glasgow, Virginia)
User/Developer

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



JSBSim – Examples of use

- FlightGear
- OpenEaagles
- Air Traffic Simulation
- 6DoF desktop simulations, flight sims, various studies and Investigations
- UAV (HITL, pilot training, autopilot development)
- Range safety ballistic trajectory study

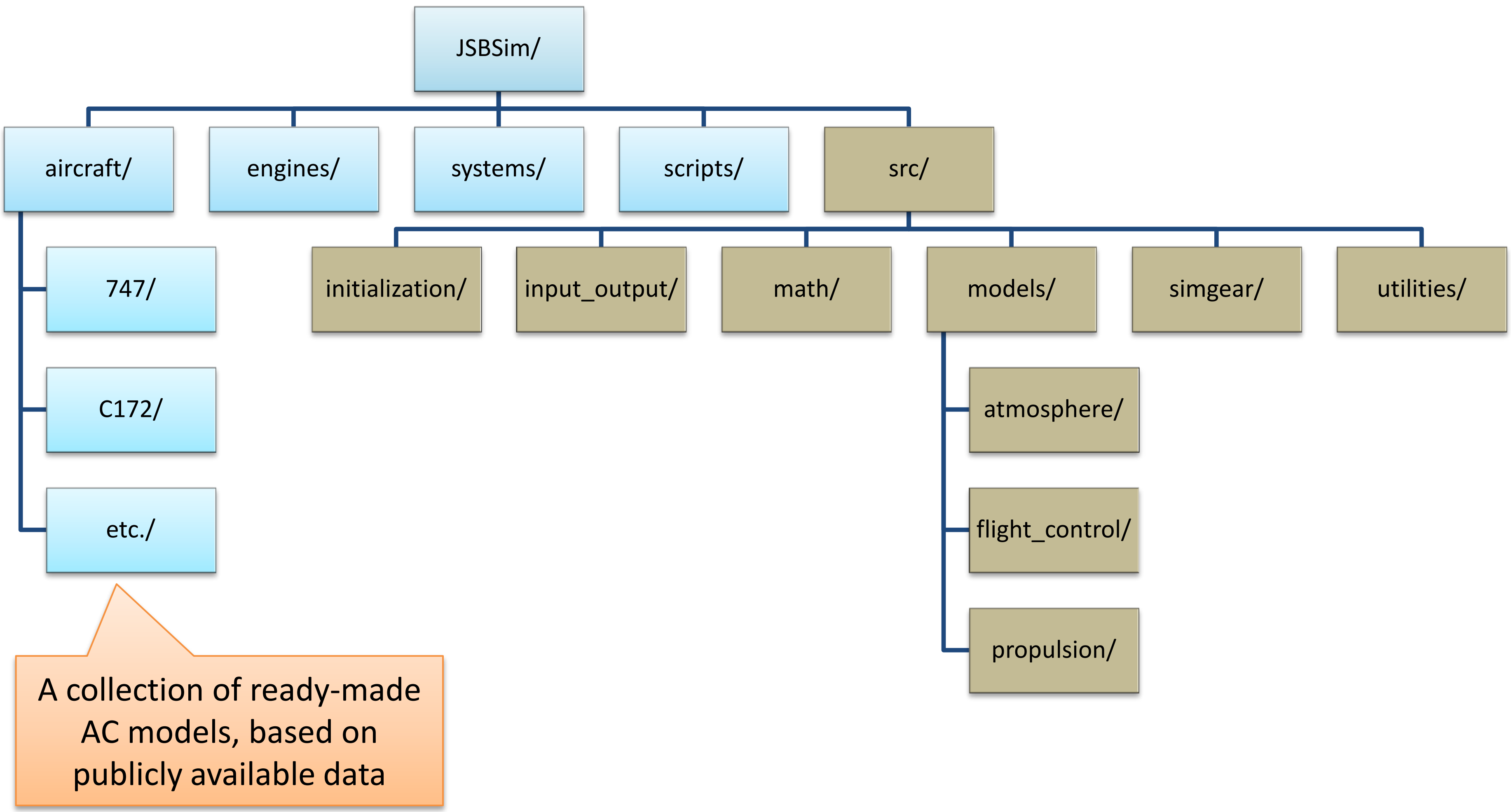
A closer look to JSBSim

- Open Source tools are all that is needed to build and use it.
- JSBSim runs on Windows, Mac, Linux, IRIX, etc.
- JSBSim is scriptable.
- JSBSim can be run in “standalone mode” (from a console or from a stub application) or integrated within a larger application framework such as OpenEaagles, or a simulation such as FlightGear.

A closer look to JSBSim – Running modes

- JSBSim can be run by itself as a standalone application, and told to connect to FlightGear via socket, subsequently directing FlightGear what to display.
- Some effort has been expended on refining the reset capability in JSBSim.
  - Reset integrator past states
  - Reset flight control component past states
  - Reconfigure aircraft settings in scripts
  - Trim aircraft
- This now permits scripted runs where the aircraft configuration file is loaded once, but multiple runs are made, such as for a set of Monte Carlo runs.

A closer look to JSBSim – Directory structure



A closer look to JSBSim – The simplest possible code

```
#include <FGFDMExec.h>                 // Include the executive header
int main(int argc, char **argv)        // Pass a script name via argv
{
  JSBSim::FGFDMExec FDMExec;           // Instantiate the Executive
  bool result = true;
  FDMExec.LoadScript(argv[1]);         // Load a script
  while (result)
    result = FDMExec.Run();            // Run until the script completes
}
```

The above code will model anything from a ball, an aircraft, and a car, to a rocket. The vehicle and simulation run specifics are all read from configuration files coded in XML format.

JSBSim Vehicle Configuration File Format

```
<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>
```

Configuration File Format (cont'd)

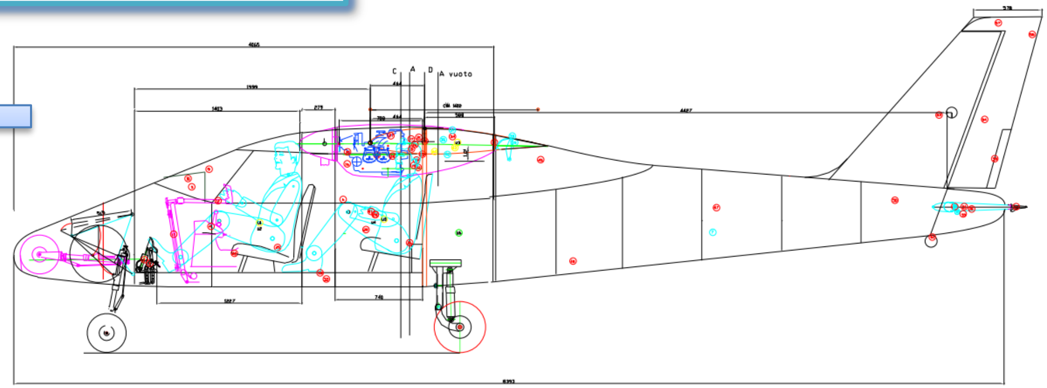
Geometry

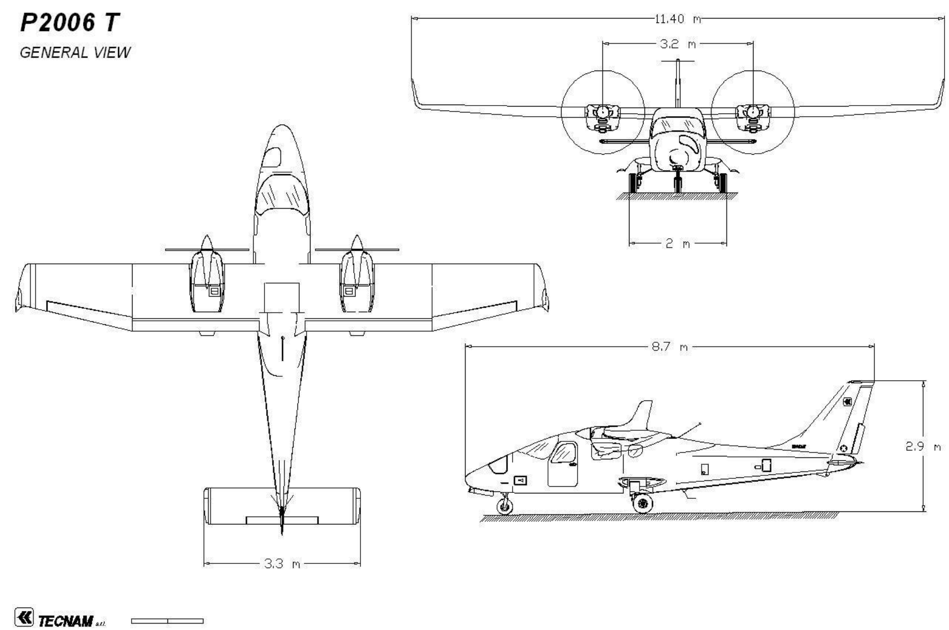
Masses

```
<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>
```





JSBSim propulsion configuration files

```xml
<?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
<?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>
```

JSBSim FCS
Modelling section

```
<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>
```

JSBSim – System modelling

- JSBSim models a set of control system components that can be linked together to build control laws.
- Any number of **<system>** elements can be specified in a configuration file.
- The way that people have used this capability has in turn driven the development and refinement of the **<system>** specification.
- autopilot control laws have been written that are generic, and feature gains and other values that can be set for a specific aircraft.
- Work is underway towards a set of common GNC capabilities, defined in files that can be included by any aircraft model

JSBSim Initialization file

```xml
<?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>
```

Initial values are often required to be close enough to equilibrium values.

Often a trim step is required when simulations start.

JSBSim Script file

AC model and initialization file selection

Initial, finel 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>
```

Layout of the presentation

- A quick introduction to JSBSim, an open source Flight Dynamics Model (FDM) software library
- **Implementation of a Trim algorithm for JSBSim, based on a probabilistic Nelder Mead solver.**
- An aicraft trimming/linearization GUI and an open source equivalent of the Matlab/Simulink aerospace toolbox.

The mathematical problem of finding trim conditions

Aircraft Equations of Motion

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

Equilibrium Flight Equations

$$\mathbf{0} = \mathbf{f}(\mathbf{x}_{\text{eq}}, \mathbf{u}_{\text{eq}})$$

$$\mathbf{x} = \big[V_t, \alpha, \beta, p, q, r, \ldots \; \psi, \theta, \phi, \text{altitude}, \text{longitude}, \text{latitude}, \ldots \; \text{rpm}, \text{prop pitch}\big]^T$$

$$\mathbf{u} = \big[\text{throttle}, \text{aileron}, \text{elevator}, \text{rudder}\big]^T$$

Design vector

$$\mathbf{d} = \left[\tilde{\mathbf{x}}^T, \tilde{\mathbf{u}}^T\right]^T$$

Solve $\mathbf{f}(\mathbf{d}) = \mathbf{0}, \; \mathbf{d} \in D$

$\tilde{\mathbf{x}}$, $\tilde{\mathbf{u}}$ are vectors obtained by taking out from ***x*** and ***u*** some trim design objectives

For instance, in most cases the airspeed V_t is a design quantity, i.e. trim conditions are searched for at given flight speeds. In some cases, when a wings-level flight condition is desired, the roll angle ϕ is known and set to zero.

Finding trim conditions, a constrained optimization problem

Aircraft Equations of Motion

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

Design vector

$$\mathbf{d} = \left[\tilde{\mathbf{x}}^T, \tilde{\mathbf{u}}^T \right]^T$$

Solving **f**(**d**) = 0

means finding the **minimum** of scalar *cost function*

$$J = \dot{V}_t^2 + \dot{\alpha}^2 + \dot{\beta}^2 + \dot{p}^2 + \dot{q}^2 + \dot{r}^2$$

for **d** in *D* (design space) is subject to constraints.

$$\min_{\mathbf{d} \in D} J(\mathbf{d}) = 0 \qquad \text{(if exists)}$$

Single terms in the sum that defines *J* are given by the first six components of the state function **f**.

Trim design objectives define the kind of trim condition desired.

Reduced order design vector

In our trim algorithm we always assign the airspeed V_t and impose some specific **constraints**. This reduces the dimension of the design vector, which is

$$\mathbf{d} = \big[\, \alpha, \beta, \ldots \\ \text{throttle, elevator, aileron, rudder} \big]^T$$

Rate of climb constraint

$$\tan\theta = \frac{ab + \sin\gamma\sqrt{a^2 - \sin^2\gamma + b^2}}{a^2 - \sin^2\gamma}$$

$$\theta \neq \pm\pi/2$$
$$a = \cos\alpha\,\cos\beta$$
$$b = \sin\phi\,\sin\beta + \cos\phi\,\sin\alpha\,\cos\beta$$

Simplified when zero ROC is enforced.

Stevens B. L. and Lewis F. L., *Aircraft Control and Simulation*, Wiley, New York, 2003.

Turn coordination constraint

$$\tan\phi = \Gamma \frac{\cos\beta}{\cos\alpha} \times \frac{(a-b^2)+\tan\alpha\sqrt{c(1-b^2)+\Gamma^2\sin^2\beta}}{a^2-b^2(1+c\tan^2\alpha)}$$

General expression derived by Stevens and Lewis

$$\Gamma = \frac{\dot{\psi} V_t}{g}$$

$$a = 1-\Gamma\tan\alpha\sin\beta$$

$$b = \frac{\sin\gamma}{\cos\beta}$$

$$c = 1+\Gamma^2\cos^2\beta$$

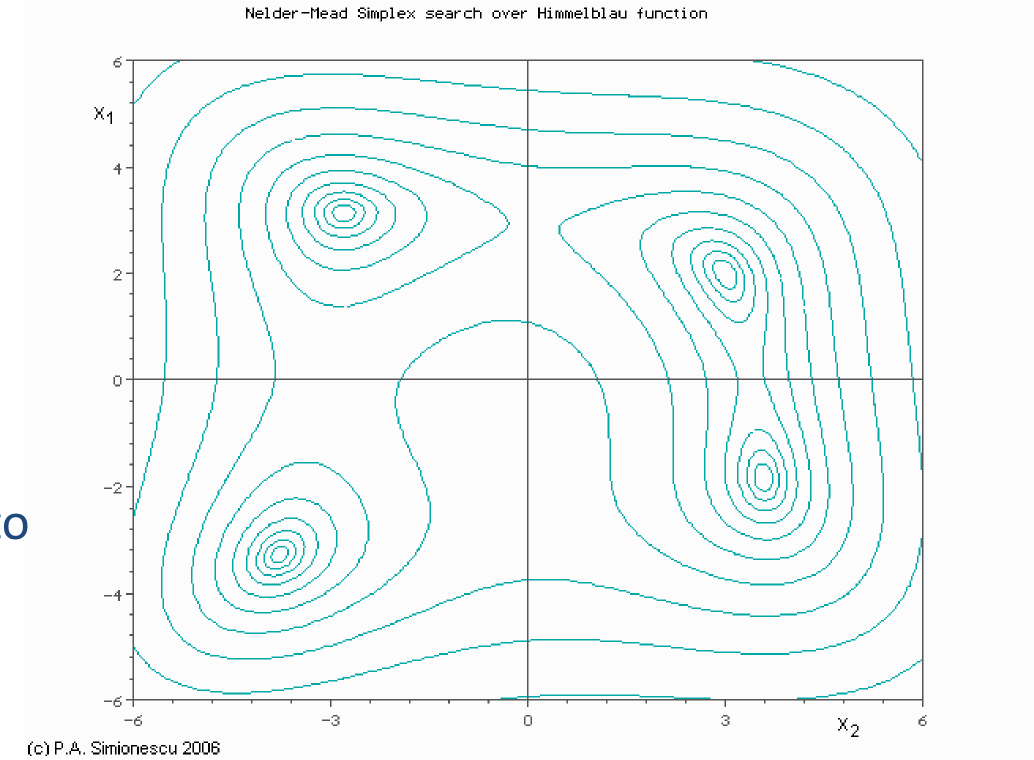
Simplified when zero sideslip is enforced.

V_t is the tangential velocity in the turn

psi-dot is the yaw rate during the turn

Trim search: Nelder Mead Simplex minimization

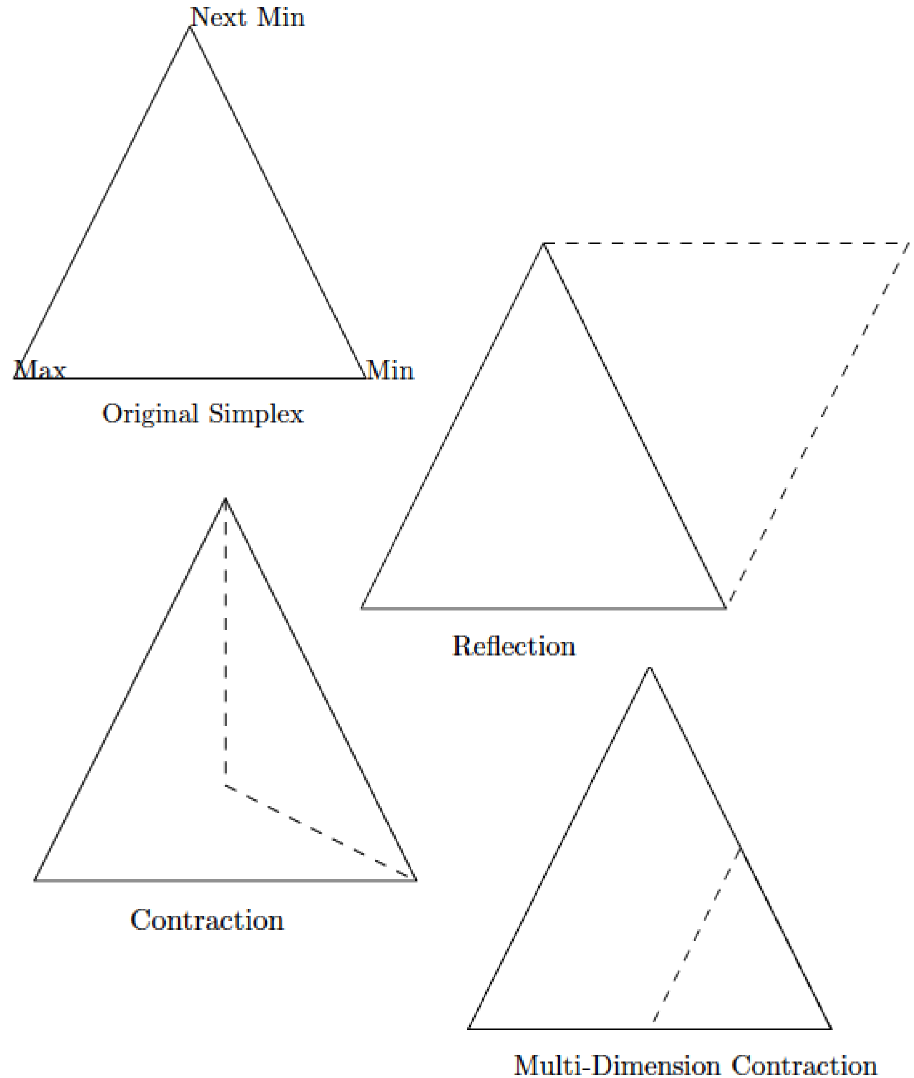
- Does not require derivative of function
- No need to compute Jacobian
- Helpful for complex systems
- Can become stuck at local minima
- Important to introduce constraints to reduce dimension of design space
- Treatment of bounds with a penalty approach



Nelder J. A. and Mead R., “Simplex method for function minimization”. *Computer Journal*, 1965.

Simplex minimization

- If **d** is a vector of an *n*-dimensional space *D*, a simplex is a set of is n+1 points of *D*.
- In the example we are looking for a design vector of 2 components. Hence the simplex has 3 points.
- *H* (High point) is where *J*(**d**) is the highest value for the given simplex.
- *L* (Low point) is where *J*(**d**) is the lowest value for the given simplex.
- Important to find good strategies to *move* the simplex in order ti find the minimum of *J*.
- Controlled Random Search (CRS), and Simplex Simulated Anealing (SSA) methods



Kvasnicka V. and Pospichal J., "A hybrid of simplex method and simulated annealing". *Chemometrics and Intelligent Laboratory Systems*, Vol. 39, No. 2, 1997.

Trim: Algorithm Designed for JSBSim

Sample n+1 points of the n-dimensional domain *D*, create simplex

While cost between high and low vertex Is greater than a tolerance, try a reflection

Try an expansion

If not a good step, try a contraction

If not a good step, try a multi-contraction

If simplex is stuck at a local minimum, resample

$S \leftarrow$ set of $(n+1)$ user defined points of D
while $|x_H - x_L| > \varepsilon$ **do**
  $x^\star \leftarrow$ randomized-reflection(S)
  if $f(x^\star) < f(x_L)$ **then**
    $x^{\star\star} \leftarrow$ randomized-expansion(S)
    if $f(x^{\star\star}) < f(x^\star)$ **then**
      $x_H \leftarrow x^{\star\star}$
    else
      $x_H \leftarrow x^\star$
    end if
  else
    $x_{0,H} \leftarrow x_H$
    randomized-contraction(S)
    if $x_H \geq x_{0,H}$ **then**
      randomized-multi-contraction(S)
    end if
  end if
  if iterations > max iterations **then**
    $S \leftarrow$ re-sample population, centered at current minimum vertex
  end if
end while

Constrained cost function implemented in JSBSim

Constrain design vector

Compute cost

Propagate simulation

Check if cost has converged

Continue to propagate, constrain cycle

$x \leftarrow \text{constrain}(D)$

$\text{cost}_0 \leftarrow f(x)$

$\text{propagate}(x)$

$x \leftarrow \text{constrain}(D)$

$\text{cost}_1 \leftarrow f(x)$

while $|\text{cost}_0 - \text{cost}_1| < \varepsilon$ **do**

$\quad \text{cost}_0 \leftarrow f(x)$

$\quad \text{propagate}(x)$

$\quad x \leftarrow \text{constrain}(D)$

$\quad \text{cost}_1 \leftarrow f(x)$

end while

return cost_1

Linearization: FGStateSpace Class

- Component based access to JSBSim's dynamics model.
- Ability to specify input, output, state vectors.
- Ability to generate state-space representation using internal JSBSim derivatives or finite difference approximation of the derivative.

More details in source code, https://github.com/jgoppert/jsbsim

Look on YouTube for: “JSBSim“ and “James Goppert“ for video demonstrations.

F16 trim example, input

```
JSBSim Trimming Utility

input ( press enter to accept [default] )

  debug level      [          0] :
model selection
  aircraft       [         f16] :
  successfully loaded: General Dynamics F-16A

flight conditions:
  altitude, ft        [       10000] :
  velocity, ft/s        [         600] :
  gamma, deg       [          0] :
  mode < non-turning(0), rolling(1), pitching(2), yawing(3) > [          0]  : 3
  yaw rate, rad/s [          0]  : .1

solver properties:
  show converge status?  [          0] :
  show simplex?     [          0] :
  pause?        [          0] :
  relative tolerance    [1.192093e-07] :
  absolute tolerance    [1.000000e-02] :
  max iterations      [        2000] :
  convergence speed   [1.100000e+00] :
  randomization ratio   [0.000000e+00] :
```

JSBSim F16A model selected

Selected altitude, velocity, flight path angle

Coordinate turn mode selected

Set of properties related to the trim algorithm

F16 trim example, output

Trim state

```
vt           : 600.000
alpha, deg   : 7.147
theta, deg   : 5.826
q, rad/s     : 0.088
thrust, lbf  : 6400.434
beta, deg    : 2.800
phi, deg     : 62.231
p, rad/s     : -0.012
r, rad/s     : 0.046
```

Normalized input vector

```
input
   throttle cmd, % : 35.450
   elevator cmd, % : -45.066
   aileron cmd, %  : 5.919
   rudder cmd, % : 15.986
```

State derivatives

```
aircraft d/dt state
   d/dt vt            : -1.247e-02
   d/dt alpha, deg/s  : -6.958e-03
   d/dt theta, deg/s  : -1.636e-03
   d/dt q, rad/s^2    : 3.526e-03
   d/dt thrust, lbf   : 0.000e+00
   d/dt beta, deg/s   : -5.550e-01
   d/dt phi, deg/s    : -1.329e-01
   d/dt p, rad/s^2    : -1.244e-02
   d/dt r, rad/s^2    : 4.625e-02
```

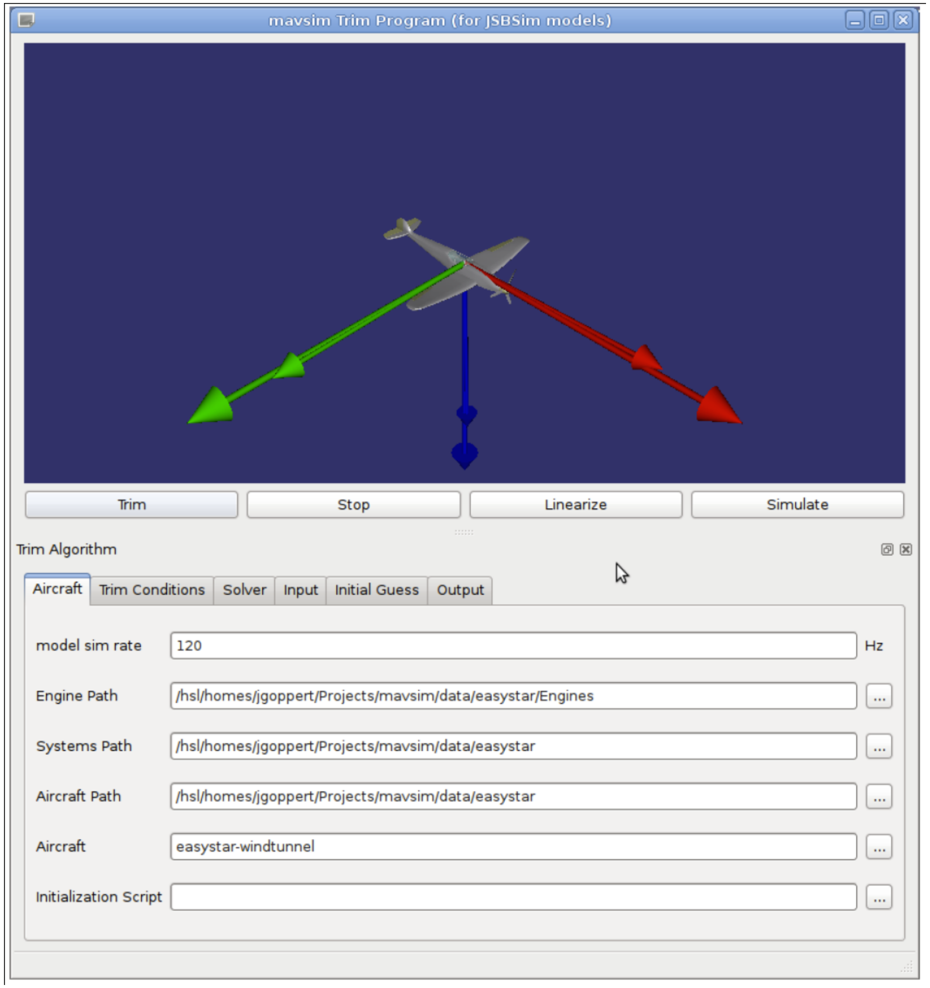
```
A=
[    0.211,    5.681,   -31.999,   -1.560,  -32.072,   -1.560,   -0.069,   -0.000,
    10.872,    0.147,    0.000,   -0.000,    0.000,    0.000;
    -0.000,   -0.881,    0.000,    0.995,   -0.000,    0.995,   -0.000,   -0.000,
     0.078,    0.001,    0.000,   -0.000,    0.000,    0.000;
    -0.000,    0.000,   -0.000,    1.000,   -0.000,    1.000,    0.000,   -0.000,
     0.000,    0.000,    0.000,    0.000,    0.000,    0.000;
     0.000,   -5.278,   -0.004,    1.182,    0.051,    1.182,    0.053,   -0.000,
    -0.132,   -0.112,   -0.000,    0.000,    0.000,   -0.000;
    -0.000,    0.000,   -0.000,    1.000,   -0.000,    1.000,    0.000,   -0.000,
     0.000,    0.000,    0.000,    0.000,    0.000,    0.000;
     0.000,   -5.278,   -0.004,    1.182,    0.051,    1.182,    0.053,   -0.000,
    -0.132,   -0.112,   -0.000,    0.000,    0.000,   -0.000;
     0.000,   -0.000,    0.000,   -0.000,    0.000,   -0.000,   -0.294,    0.053,
     0.020,   -0.917,    0.000,    0.000,    0.000,    0.000;
     0.000,    0.000,    0.000,   -0.000,    0.000,   -0.000,    0.000,   -0.000,
     1.000,    0.021,   -0.000,   -0.000,   -0.000,    0.000;
```

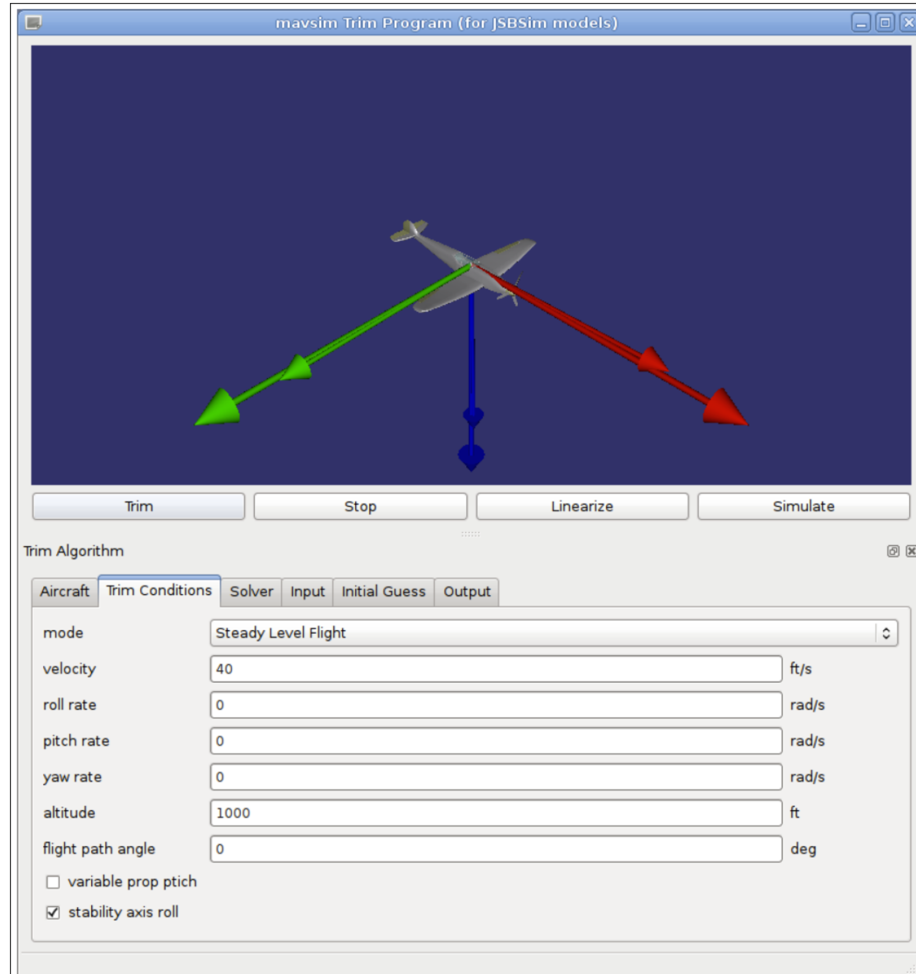
Excerpt of linearized model

Layout of the presentation

- A quick introduction to JSBSim, an open source Flight Dynamics Model (FDM) software library
- Implementation of a Trim algorithm for JSBSim, based on a probabilistic Nelder Mead solver.
- **An aicraft trimming/linearization GUI and an open source equivalent of the Matlab/Simulink aerospace toolbox.**

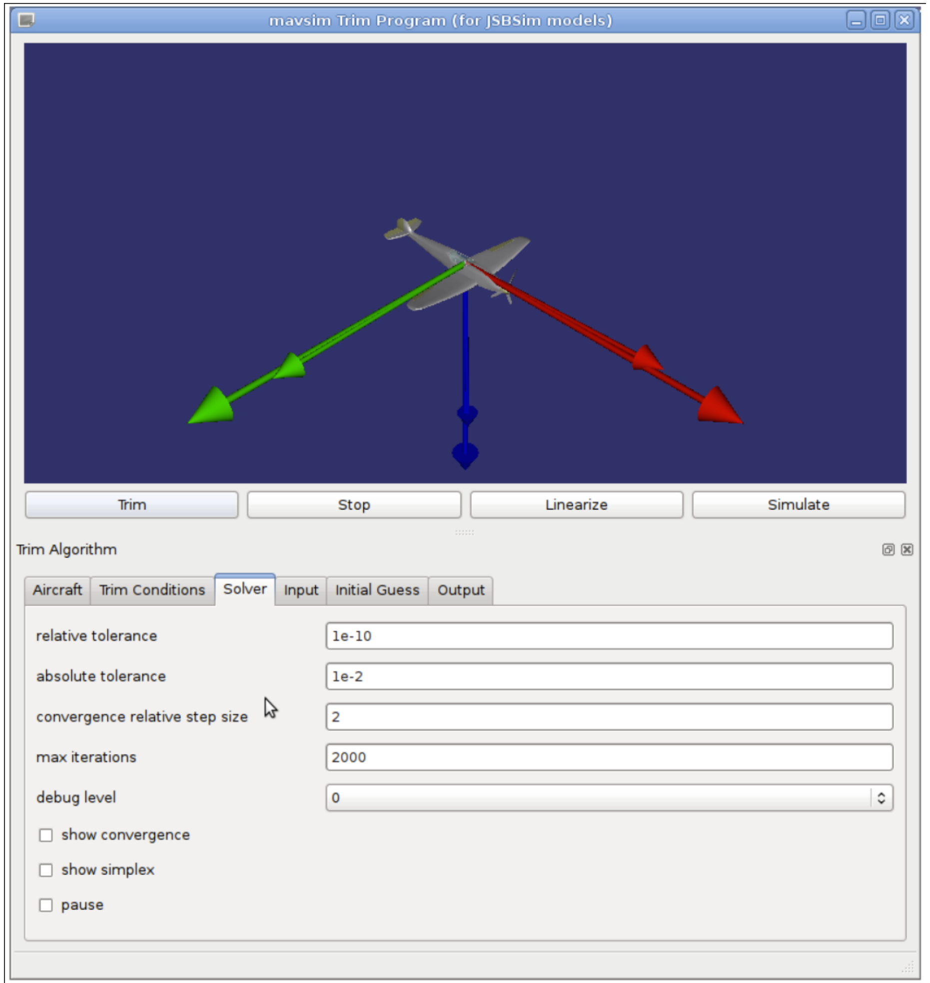
Trim GUI, a Qt-based application (portable)

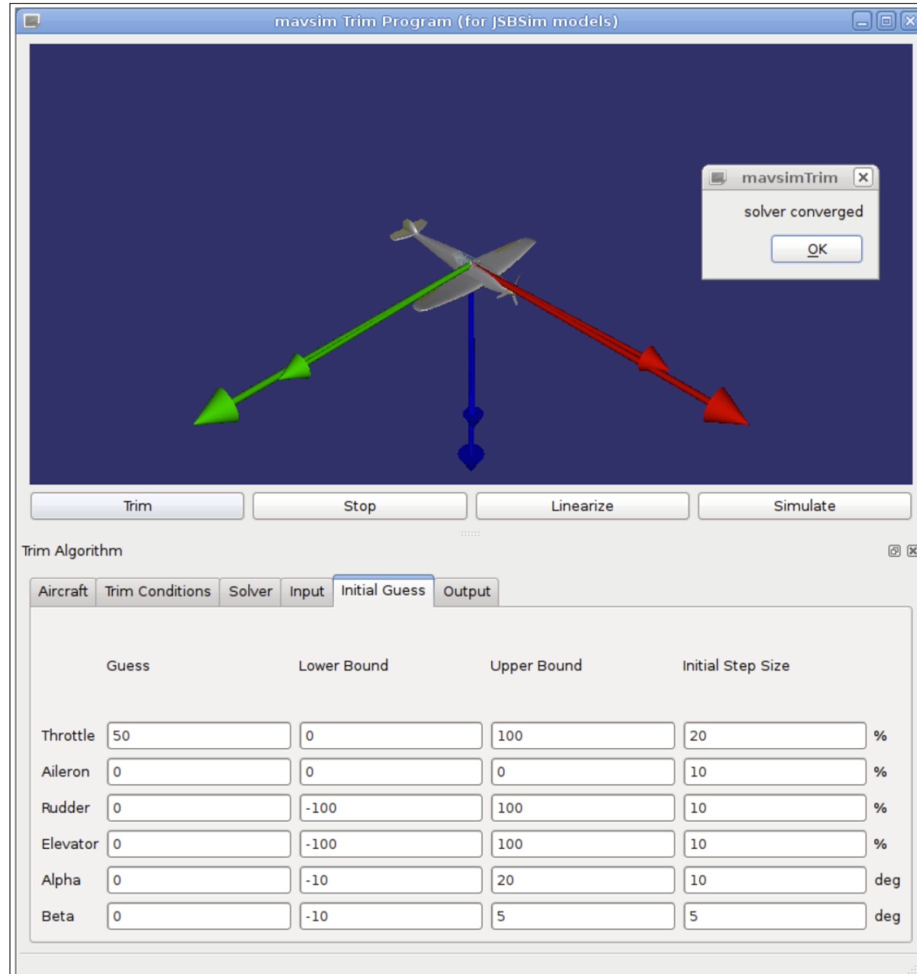




More details in source code, https://github.com/jgoppert/jsbsim

Trim GUI, a Qt-based application (portable)

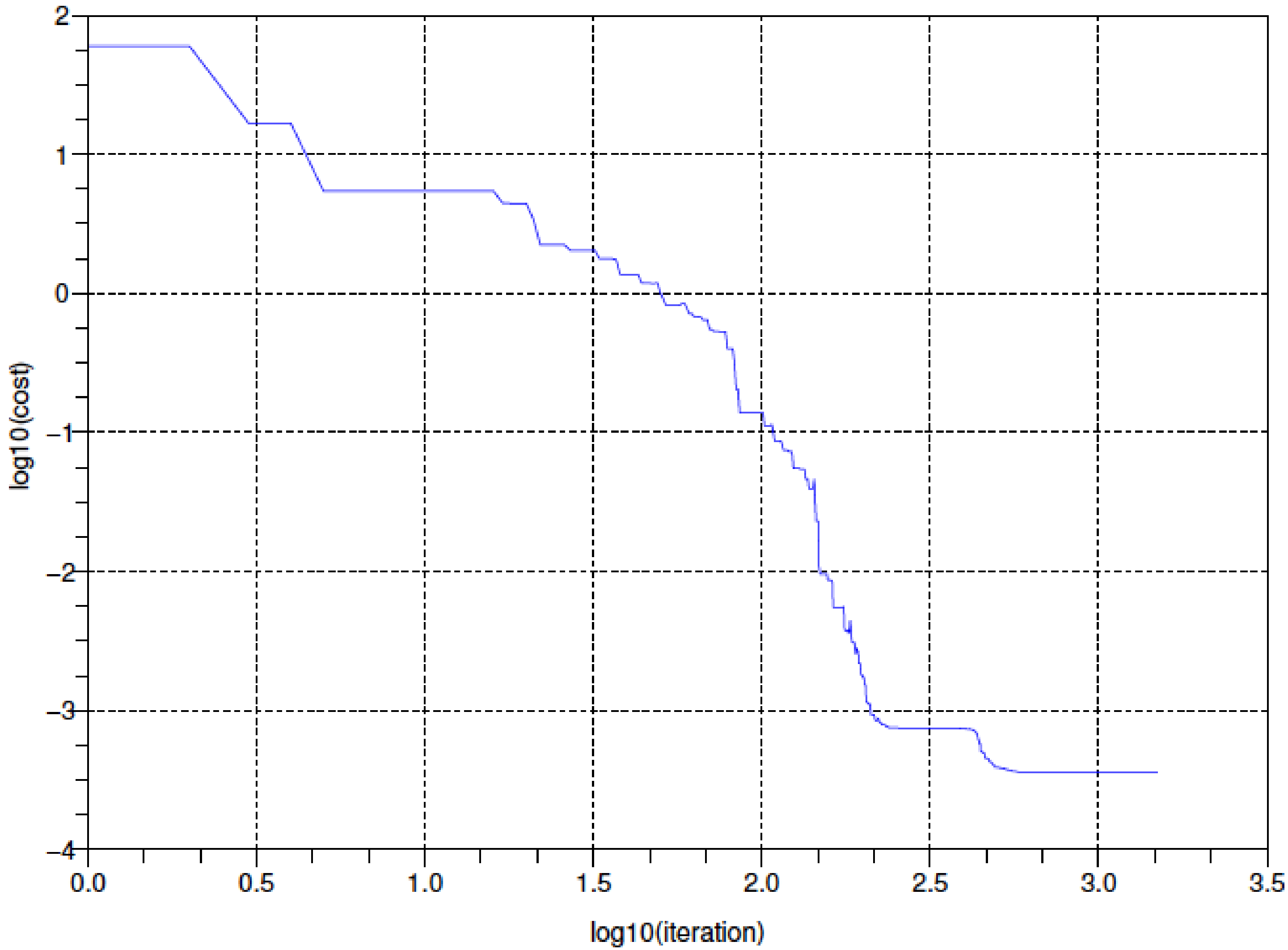




More details in source code, https://github.com/jgoppert/jsbsim

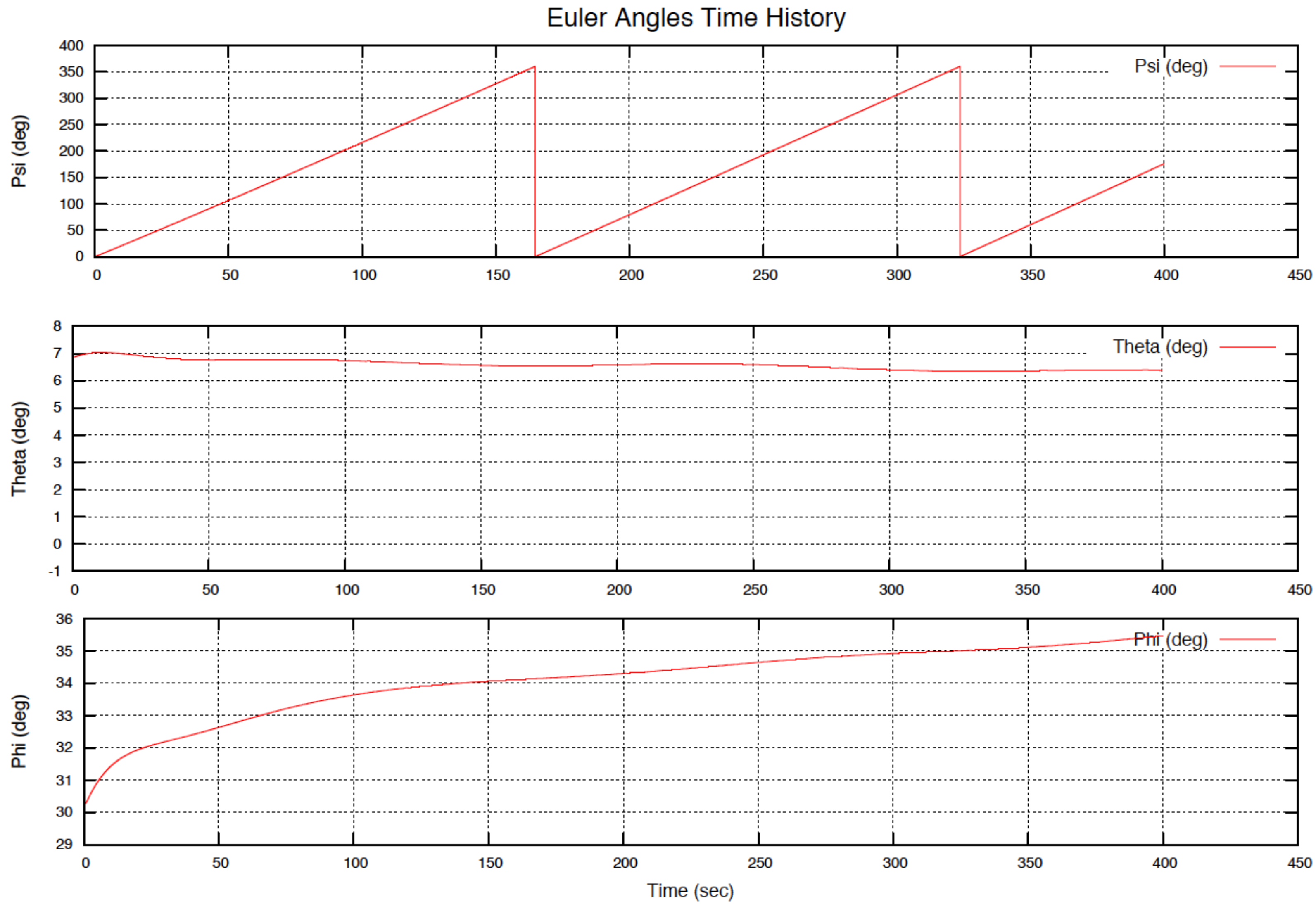
F16 trim example, Simplex Cost Convergence History

30 deg Bank Turn
@ 500 ft/s



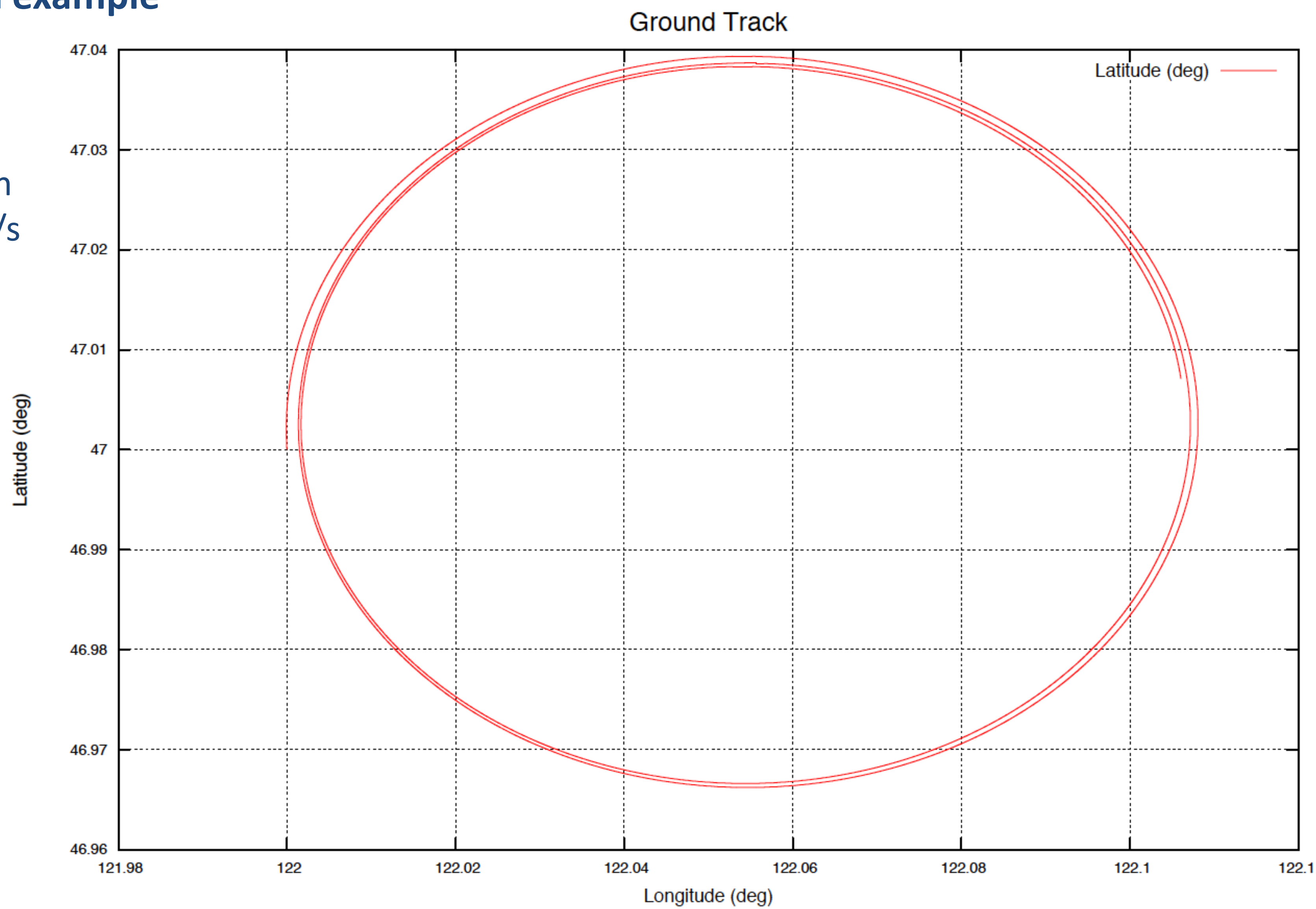
F16 trim example

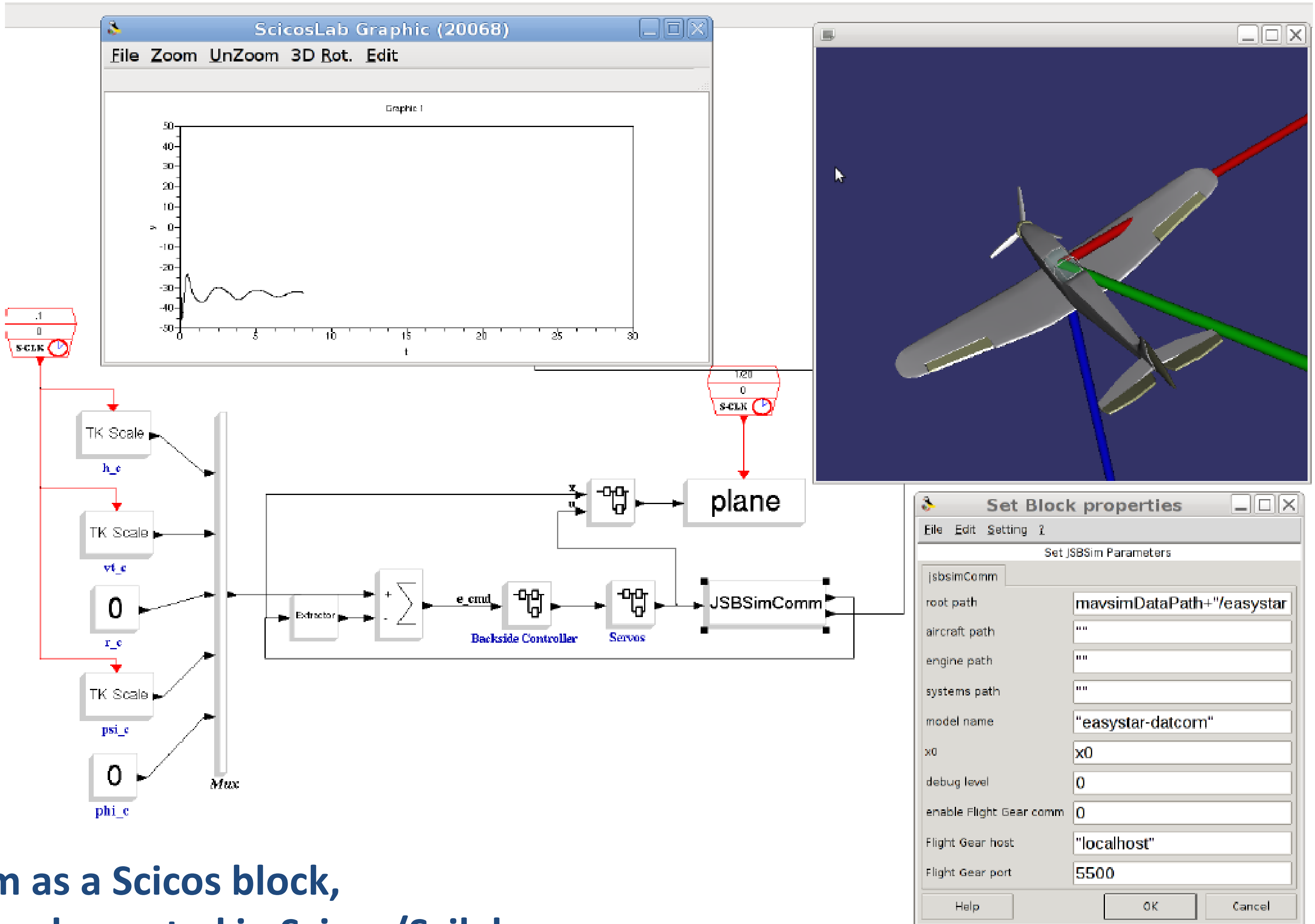
30 deg
Bank Turn
@ 500 ft/s



F16 trim example

30 deg
Bank Turn
@ 500 ft/s





JSBSim as a Scicos block, trim implemented in Scicos/Scilab

Conclusions

- A simplex based trimming method for the JSBSim flight dynamics library. Based upon the Nelder-Mead simplex method, with randomization added.
- A generic state space interface to the JSBSim library. Allows users to create custom state space representations, which are useful for control design and dynamics analysis.
- A multiplatform GUI to interface to the trimming and linearization code. The GUI makes finding a trim condition and creating a linear model simple, and helps the user with more intuitive feedback.
- A Scicos block to interface to JSBSim. A linear model can be used to design a controller in Scicos/ScicosLab and the JSBSim aircraft model can be simulated with dynamics and control elements within the Scicos environment.

Thank you

Thank you

Thank you

Thank you

Thank you

JSBSim – A selection of papers

See: [1] Berndt J. S., "JSBSim: An Open Source Flight Dynamics Model in C++." AIAA 2004-4923, AIAA Modeling and Simulation Technologies Conference and Exhibit 16 - 19 August 2004, Providence, Rhode Island, USA.

[2] Coiro D. P., De Marco A., Nicolosi F., "A 6DOF Flight Simulation Environment for General Aviation Aircraft with Control Loading Reproduction." AIAA 2007-6364, AIAA Modeling and Simulation Technologies Conference and Exhibit 20-23 August 2007, Hilton Head, South Carolina, USA.

[3] Berndt J. S., De Marco A., "Progress on and Usage of the Open Source Flight Dynamics Model Software Library, JSBSim." AIAA 2009-5600, AIAA Modeling and Simulation Technologies Conference and Exhibit 10-13 August 2009, Chicago, Illinois, USA.

[3] Agte J., Borer N. K., de Weck O., "A Simulation-based Design Model for Analysis and Optimization of Multi-State Aircraft Performance." AIAA 2010-2997, 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 12-15 April 2010, Orlando, Florida.

JSBSim use in simulation-based design

JSBSim has been used in simulation–based aircraft design and analysis approaches.

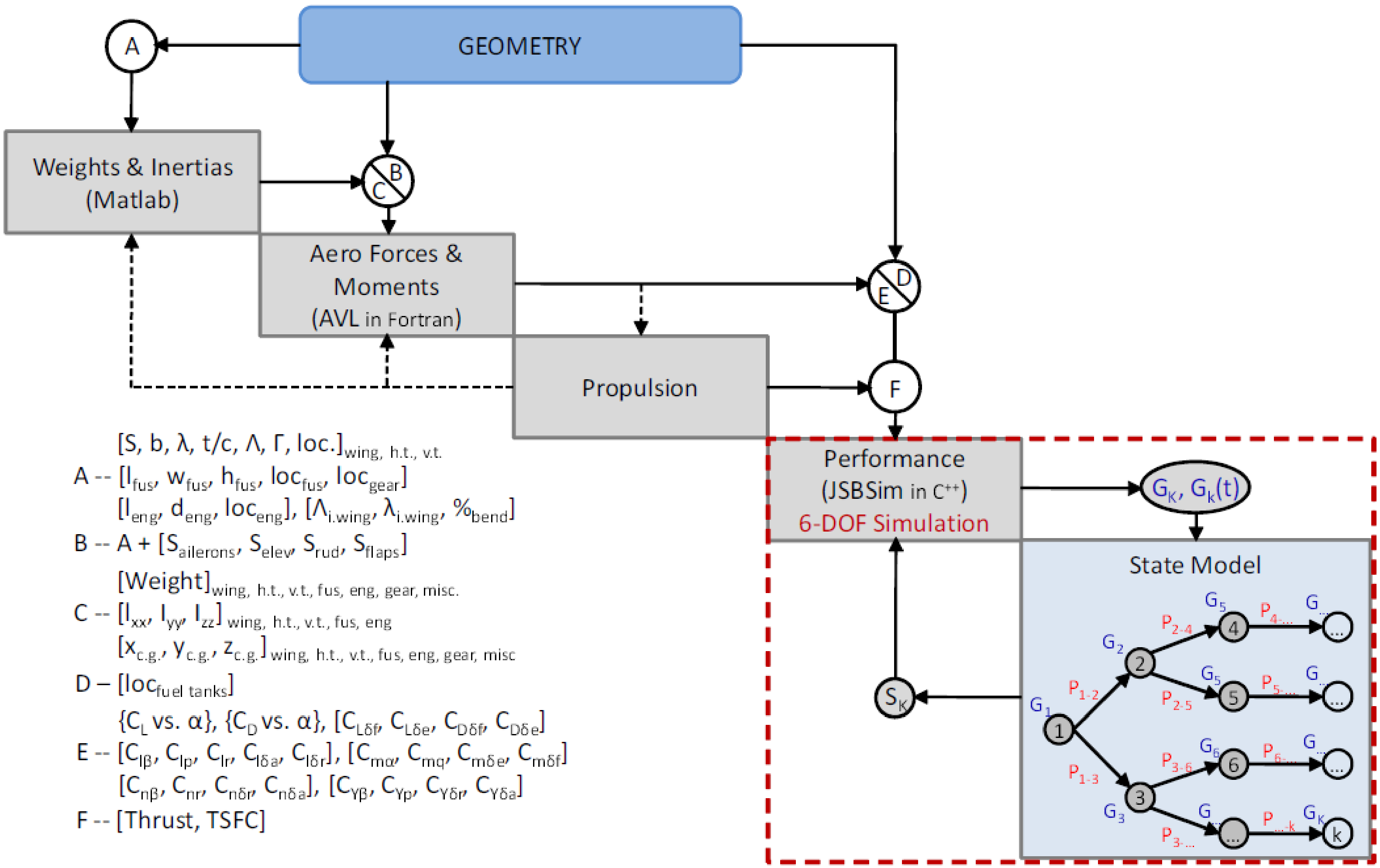
The focus is on the evaluation of aircraft as multi-state systems, i.e. one having a finite set of performance levels or ranges. Sometimes these ranges are differentiated by distinct levels of failure.

In order to accurately examine numerous aircraft performance states, a multi-disciplinary design model is used, a 6-DoF flight simulator integrated with a vortex lattice aerodynamics solver and a tool for calculation of weights and inertias.

The JSBSim batch running mode facilitates a global approach for concurrent analysis of aircraft expected performance and availability. Namely, by allowing systematic calculation of performance metrics for differing aircraft states, the relationship between an aircraft's global design variables and its performance and availability may be established.

Such an approach allows designers to identify those elements that might drive system loss probability through an analysis of performance changes across system states and their respective sensitivity to design variables.

JSBSim use in simulation-based design



Aircraft integrated system model used at Draper Laboratory with behavioral–Markov failure modelling