Software for Simulating Six Degree of Freedom Motion of a Rigid Aircraft

Professor Dominick Andrisani, II
andrisan@purdue.edu

School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN

January 2003

Introduction

The software described here allows for six degree of freedom simulation of the arbitrary motion of a rigid aircraft. The nonlinear differential equations are of the form

$$\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{g}(\mathbf{x},\mathbf{u}), \quad \mathbf{x}(0) \\
\bar{\mathbf{y}} &= \bar{\mathbf{h}}(\bar{\mathbf{x}},\bar{\mathbf{u}}) \\
\mathbf{u}(\mathbf{t}) &= \textbf{specified}
\end{aligned}$$

where x, y and u are the state, output and control vectors respectively and x(0) is the initial condition.

Two versions of some of the MATLAB software are provided for students who have access to either MATLAB 5 or MATLAB 6. For example

Step3_Simulink_SimMATLAB5.m

and Step3_Simulink_SimMATLAB6.m

are two versions of the same script file. In the subsequent documentation these files are sometimes referred to Step3_Simulink_SimMATLABx.m.

Two different ways of getting the solution to these nonlinear differential equations are provided. The first method is by numerical integration. SIMULINK is used for this. The second method uses linearization to obtain the total approximate solution. These methods are depicted in Figure 1. On the figure are the names of MATLAB scripts (ending with .m) and SIMULINK model files (ending with .mdl) that implement the various functions in each block.

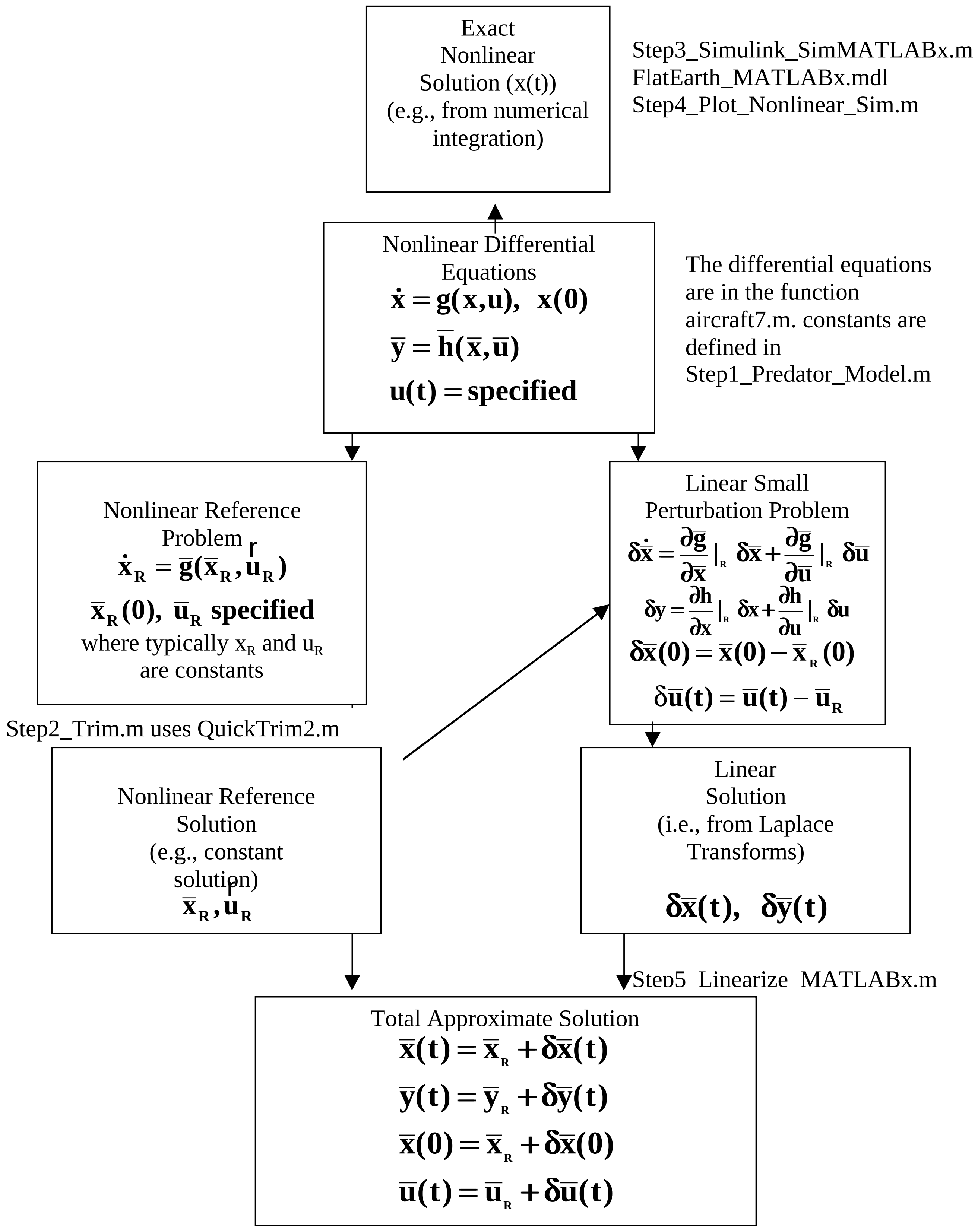

Exact Nonlinear Solution (x(t)) (e.g., from numerical integration)
Step3_Simulink_SimMATLABx.m
FlatEarth_MATLABx.mdl
Step4_Plot_Nonlinear_Sim.m
Nonlinear Differential Equations
$\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x},\mathbf{u}),\ \ \mathbf{x}(0)$
$\bar{\mathbf{y}} = \bar{\mathbf{h}}(\bar{\mathbf{x}},\bar{\mathbf{u}})$
$\mathbf{u}(t) = \text{specified}$
The differential equations are in the function aircraft7.m. constants are defined in Step1_Predator_Model.m
Nonlinear Reference Problem
$\dot{\mathbf{x}}_R = \bar{\mathbf{g}}(\bar{\mathbf{x}}_R, \bar{\mathbf{u}}_R)$
$\bar{\mathbf{x}}_R(0),\ \bar{\mathbf{u}}_R$ specified
where typically x_R and u_R are constants
Linear Small Perturbation Problem
$\delta\dot{\bar{\mathbf{x}}} = \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{x}}}|_R\ \delta\bar{\mathbf{x}} + \frac{\partial \bar{\mathbf{g}}}{\partial \bar{\mathbf{u}}}|_R\ \delta\bar{\mathbf{u}}$
$\delta\mathbf{y} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}|_R\ \delta\mathbf{x} + \frac{\partial \mathbf{h}}{\partial \mathbf{u}}|_R\ \delta\mathbf{u}$
$\delta\bar{\mathbf{x}}(0) = \bar{\mathbf{x}}(0) - \bar{\mathbf{x}}_R(0)$
$\delta\bar{\mathbf{u}}(t) = \bar{\mathbf{u}}(t) - \bar{\mathbf{u}}_R$
Step2_Trim.m uses QuickTrim2.m
Nonlinear Reference Solution (e.g., constant solution)
$\bar{\mathbf{x}}_R, \bar{\mathbf{u}}_R$
Linear Solution (i.e., from Laplace Transforms)
$\delta\bar{\mathbf{x}}(t),\ \ \delta\bar{\mathbf{y}}(t)$
Step5_Linearize_MATLABx.m
Total Approximate Solution
$\bar{\mathbf{x}}(t) = \bar{\mathbf{x}}_R + \delta\bar{\mathbf{x}}(t)$
$\bar{\mathbf{y}}(t) = \bar{\mathbf{y}}_R + \delta\bar{\mathbf{y}}(t)$
$\bar{\mathbf{x}}(0) = \bar{\mathbf{x}}_R + \delta\bar{\mathbf{x}}(0)$
$\bar{\mathbf{u}}(t) = \bar{\mathbf{u}}_R + \delta\bar{\mathbf{u}}(t)$

Figure 1 Solution Procedures for Nonlinear Differential Equations

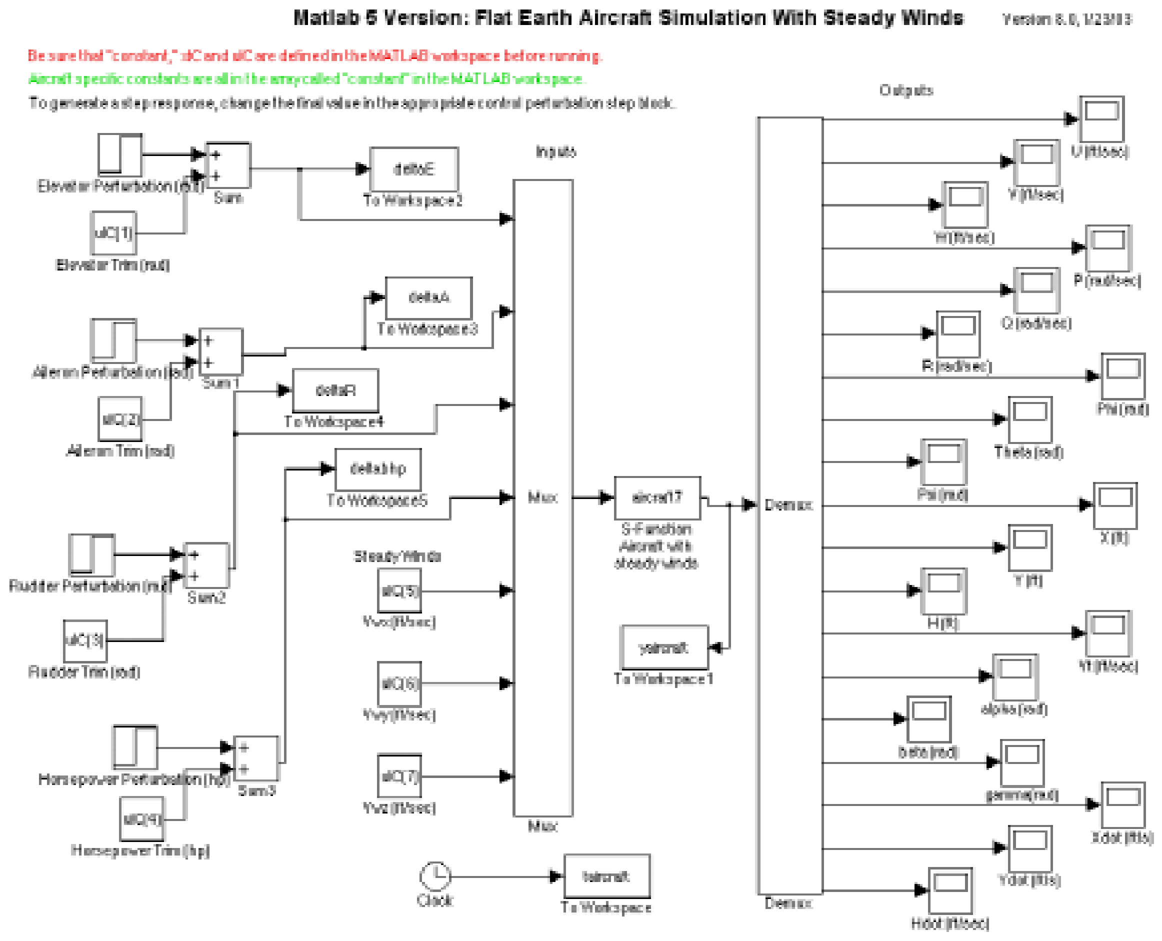
Software Availability

Wed Address
http://roger.ecn.purdue.edu/~andrisan/Courses/AAE565_S2003/Index.html

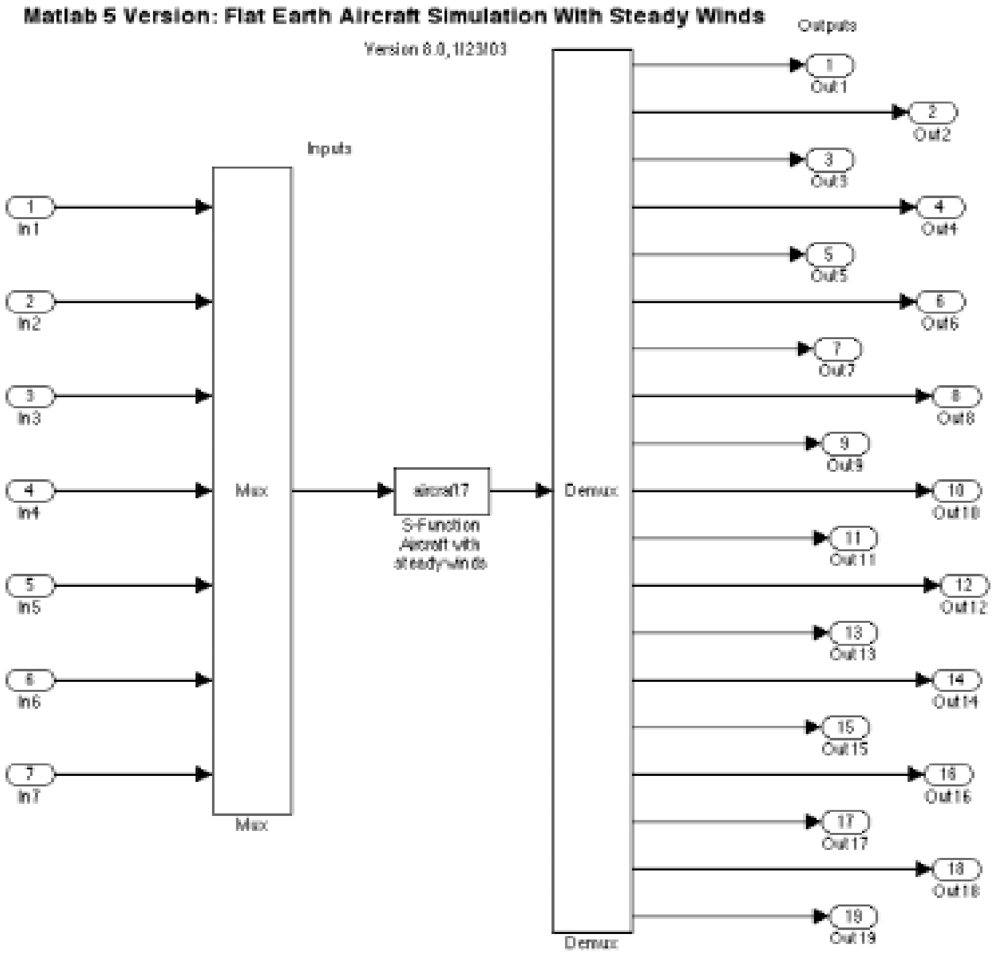
All MATLAB Scripts and models for the nonlinear simulation of a Predator UAV in a single zip file can be downloaded in a single zipped file (FlatEarth_Predator_V8.zip). Alternately, you can view and save each file separately using a link to the folder that has all the individual MATLAB scripts and models.

SIMULINK Models

FlatEarth_MATLABx.mdl



FlatEarth_ports_MATLABx.mdl



MATLAB Scripts

Step0_ReadMe.m   Version 8.0 1/23/03

These MALAB scripts are designed to analyze and simulate arbitrary rigid aircraft over the flat earth. A vehicle description file called Basic_Constants_Predator is included allowing analysis of the Predator UAV.

It is important that these MATLAB scripts be run in order.

- Step1_Predator_Model.m
- Step2_Trim.m
- Step3_Simulink_SimMATLAB5.m or Step3_Simulink_SimMATLAB6.m
- Step4_Plot_Noninear_Sim.m (optional step)
- Step5_Linearize_MATLAB5.m or Step5_Linearize_MATLAB6.m
- Step6_Longitudinal_Design.m
- Step7_Lateral_Design.m (can be run after Step5)

MATLAB 5 and 6 versions of several files are provided allowing users of either version of MATLAB to use the software.

Step1_Predator_Model.m   Version 8.0 1/23/03

OBJECTIVE: Develop the aerodynamic and thrust model for a particular aircraft.

INPUTS: a file of basic vehicle constants, e.g. Basic_Constants_Predator.m

OUTPUTS: In the MATLAB workspace will be defined an array called constant.
Note: memory is cleared at the start of this script.

Step2_Trim.m   Version 8.0 1/23/03

OBJECTIVE: Trim the aircraft and develop initial conditions for the
SIMULINK nonlinear simulation.

INPUTS: the array called constant

OUTPUTS: In the MATLAB workspace will be defined initial conditions
on the state and controls (xIC and uIC). The array constant, xIC and uIC
are all required by the SIMULINK nonlinear simulation.

Step3_Simulink_SimMATLABx.m Version 8.0 1/23/03

OBJECTIVE: Simulate an aircraft in 6 degree of freedom motion
with nonlinear equations of motion and nonlinear
aerodynamic and thrust models. A flat earth and
rigid body dynamics are assumed.

INPUT: In the MATLAB workspace will be defined initial conditions
on the state and controls (xIC and uIC). The array constant, xIC and uIC
are all required by this SIMULINK nonlinear simulation.

OUTPUT: Arrays in the MATLAB workspace called taircraft and yaircraft
that contain an aray of time values and a matrix of output state
time histories.

Step4_Plot_Noninear_Sim.m Version 8.0 1/23/03

OBJECTIVE: Plot results from the nonlinear flat earth simulation.

INPUTS: In the MATLAB workspace must be an array of simulation times (taircraft)
and a matrix of simulation outputs (yaircraft). These are generally
created by the SIMULINK simulations FlatEarth_MATLABx.mdl.

OUTPUTS: Four figures, each with three subplots.

Step5_Linearize_MATLABx.m Version 8.0 1/23/03

OBJECTIVES: 1. Script to linearize the aircraft system assuming the nonlinear
aircraft model in SIMULINK model FlatEarth_MATLABx.mdl.
2. Perform a linear simulation for the same conditions as
used in the SIMULINK nonlinear simulation.
3. Overplot the nonlinear simulation results and the linear simulation
results in order to verify the accuracy of the linearization.
4. Split the 12th order system into the smaller longitudinal
and lateral-directional subsystems.
5. Save to binary files the longitudinal and lateral-directional
state space subsystems (modelLong.mat, modelLat.mat).
6. Determine if we are on the frontside or the backside of
the power required curve.

INPUTS: 1. In the MATLAB workspace must be an array of simulation times (taircraft)
and a matrix of simulation outputs (yaircraft). These are generally
created by the SIMULINK simulations FlatEarth_MATLABx.mdl.
2. In the MATLAB workspace will be defined initial conditions

on the state and controls (xIC and uIC) andthe array constant.

OUTPUTS: 3 figures of overplots,
plot of the power required curve,
2 output files (modelLong.mat, modelLat.mat),
transfer functions, poles, natural frequencis and damping ratios in the command window.

Step6_Longitudinal_Design.m Version 8.0 1/23/03

OBJECTIVE: Set up linear models for use in linear control system design
for the longitudinal subsystem

INPUTS: The file modelLong.mat created by Step5_Linearize_MATLABx.m

OUTPUTS: Many transfer functions, poles, natural frequencies and
damping ratios in the MATLAB command window. Also several linear
time invariant systems are created in the MATLAB workspace f
or subsequent analysis (Lsys, Ltfsys, Lzpksys).

NOTES: 1. This code can be run after Step5_Linearize_MATLABx.m has been run.
2. This code clears memory at the start if execution

Step7_lateral_Design.m   Version 8.0 1/23/03

OBJECTIVE: Set up linear models for use in linear control system design
for the lateral-directional subsystem

INPUTS: The file modelLat.mat created by Step5_Linearize_MATLABx.m

OUTPUTS: Many transfer functions, poles, natural frequencies and
damping ratios in the MATLAB command window. Also several linear
time invariant systems are created in the MATLAB workspace f
or subsequent analysis (LDsys, LDtfsys, LDzpksys).

NOTES: 1. This code can be run after Step5_Linearize_MATLABx.m has been run.
2. This code clears memory at the start if execution

Complete Listing of Scripts and Model Files

Basic_Constants_Predator.m
CL_0.M
CL_alpha.m
CL_alpha_dot.m
CL_de.m
CL_q.m
Check_Constants.m
Cl_beta.m
Cl_da.m
Cl_dr.m
Cl_p.m
Cl_r.m
Cm_0.m
Cm_a_dot.m
Cm_alpha.m
Cm_de.m
Cm_q.m
Cn_beta.m
Cn_da.m
Cn_dr.m
Cn_p.m
Cn_r.m
Cy_beta.m
Cy_da.m
Cy_dr.m
Cy_p.m
Cy_r.m
FlatEarth_MATLAB5.mdl
FlatEarth_MATLAB6.mdl
FlatEarth_ports_MATLAB5.mdl
FlatEarth_ports_MATLAB6.mdl

Make_Constants.m
NameScript.m
QuickTrim2.m

Step0_ReadMe.m
Step1_Predator_Model.m
Step2_Trim.m
Step3_Simulink_SimMATLAB5.m
Step3_Simulink_SimMATLAB6.m
Step4_Plot_Nonlinear_Sim.m
Step5_Linearize_MATLAB5.m
Step5_Linearize_MATLAB6.m
Step6_Longitudinal_Design.m
Step7_Lateral_Design.m
TransformB2NED.m
TransformB2Wind.m
TransformNED2B.m
TransformWind2B.m
aeroforce.m
aeromoment.m
aircraft7.m
interlim1.m
interlim2.m
interlim3.m
modelLat.mat
modelLong.mat
nufun.m
rhofun.m
selector.m
thrust.m

Brief Description of Other MATLAB Functions and Scripts

Basic_Constants_Predator.m contains basic constants for the Predator. Basic constants include things like aspect ratio of the vertical tail and horizontal tail area.

CL_0.M computes the stability derivative CL0.

CL_alpha.m computes the stability derivative CLa (lift curve slope).

CL_alpha_dot.m computes the stability derivative Clalphadot.

CL_de.m computes the control derivative Clde (elevator effectiveness).

CL_q.m computes the stability derivative CLq.

Check_Constants.m is a script to perform a believability check on the array called constant which contains the stablility and control derivatives and inertia properties.

Cl_beta.m computes the stability derivative Clbeta (dihedral effect).

Cl_da.m computes the control derivative Clda (aileron effectiveness).

Cl_dr.m computes the control derivative Cldr (rolling moment due to rudder).

Cl_p.m computes the stability derivative Clp (damping in roll).

Cl_r.m computes the stability derivative Clr.

Cm_0.m computes the stability derivative Cm0.

Cm_a_dot.m computes the stability derivative Cmalphadot (lag of downwash effect).

Cm_alpha.m computes the stability derivative Cmalpha (pitch stiffness).

Cm_de.m computes the control derivative Cmde (elevator effectiveness).

Cm_q.m computes the stability derivative Cmq (damping in pitch).

Cn_beta.m computes the stability derivative Cnbeta (weathercock stability derivative).

Cn_da.m computes the control derivative Cnda (adverse/proverse aileron yaw).

Cn_dr.m computes the control derivative Cndr (rudder effectiveness).

Cn_p.m computes the stability derivative Cnp.

Cn_r.m computes the stability derivative Cnr (damping in yaw).

Cy_beta.m computes the stability derivative Cybeta (side force due to sideslip).

Cy_da.m computes the control derivative Cyda.

Cy_dr.m computes the control derivative Cydr.

Cy_p.m computes the stability derivative Cyp.

Cy_r.m computes the stability derivative Cyr.

Make_Constants.m uses Basic_Constants_Predator to compute the array called constant.

NameScript.m helps in the believability check performed by Check_Constants.

QuickTrim2.m trims the aircraft at a given speed and altitude in steady level flight.

TransformB2NED.m computes transformation matrix from body to North-East-Down frame.

TransformB2Wind.m computes transformation matrix from body to wind frame.

TransformNED2B.m computes transformation matrix North-East-Down to body frame.

TransformWind2B.m computes transformation matrix wind to body frame.

aeroforce.m computes the aerodynamic force vector in body axis components.

aeromoment.m compute the aerodynamic moment vector in body axis components.

aircraft7.m contains the equations of motion and output equations in s-function format for use in an s-function by SIMULINK.

interlim1.m is a 1-dimensional table look-up function with limits. If the inputs are beyond the range specified in the table, the outputs are limited to last value in the table and a non-fatal warning message is generated. Extrapolation is never performed. The informed student should check out these non-fatal warnings to verify that the code handled this ambiguous situation in the proper manner.

interlim2.m is a 3-dimensional table look-up function with limits. If the inputs are beyond the range specified in the table, the outputs are limited to last value in the table and a non-fatal warning message is generated. Extrapolation is never performed. The informed student should check out these non-fatal warnings to verify that the code handled this ambiguous situation in the proper manner.

interlim3.m is a 3-dimensional table look-up function with limits. If the inputs are beyond the range specified in the table, the outputs are limited to last value in the table and a non-fatal warning message is generated. Extrapolation is never performed. The informed student should check out these non-fatal warnings to verify that the code handled this ambiguous situation in the proper manner.

modelLat.mat is a MALLAB binary file generated by the statement

```
save modelLat aLD bLD cLD dLD Vt Hp aircraft
```

It contains the lateral-directional subsystem in state space form (e.g., a, b, c, d matrices).

modelLong.mat is a MALLAB binary file generated by the statement

```
save modelLong aL bL cL dL Vt Hp aircraft
```

It contains the longitudinal subsystem in state space form (e.g., a, b, c, d matrices).

nufun.m calculates kinematic viscosity in the standard atmosphere.

rhofun.m calculates air density in the standard atmosphere.

selector.m creates a smaller state space subsystem from a larger state space system by selecting certain states, control and outputs to be retained in the smaller subsystem.

thrust.m generated the thrust force and moment in body axis components.