AAE421 Flight Dynamics and Control Analysis Software

1. CessnaInitialize.m
   A MATLAB script to compute the initial conditions for Simulink model FlatEarth.mdl. It calls the utility Cessna182 to set up the 67 Cessna 182 aircraft constants. The first 57 of these constants are used by FlatEarth.mdl. In addition, it trims the aircraft at a given airspeed and altitude using QuickTrim.m.

2. FlatEarth.mdl
   Simulink model to generate time responses of the 12th order nonlinear equations of motion of the flat earth equations of motion. This model used s-function aircraft.m. Time responses are stored in the MATLAB workspace for subsequent plotting by FlatEarthAnal.m.

3. FlatEarth2.mdl
   This version of the FlatEarth.mdl has the inputs and outputs clearly identified and can be used by MATLAB function linmod to get the linearized equations of motion (a, b, c, and d matrices) from the nonlinear equations of motion ([a,b,c,d]=linmod('FlatEarth2',x1,u1)).

4. FlatEarthAnal.m
   This script does subsequent analysis of results from FlatEarth.mdl. It linearizes the nonlinear equations of motion. Furthermore, it generates time responses of the linearized system and overplots the time histories on top of the nonlinear time histories previously generated by FlatEarth.mdl. It computes eignevalues of the linear system.

5. CessnaLongSC.m
   A MATLAB script to compute linear equations of longitudinal motion, time responses, etc. in the manner described in our textbook by Roskam (see page 320). This is a different approach then used by FlatEarthAnal.m (which uses linmod to linearize the nonlinear equations of motion embedded in aircraft.m). This script uses the quadratic drag polar rather then the linear drag polar. It uses the constants defined by Cessna182.m.

6. CessnaLatSC.m
   A MATLAB script to compute the linear equations of lateral-directional motion, time responses, etc. in the manner described in our textbook by Roskam (see page 349). This is a different approach then used by FlatEarthAnal.m (which uses linmod to linearize the nonlinear equations of motion embedded in aircraft.m). It uses the constants defined by Cessna182.m.

7. CessnaCrosswind.m
   A MATLAB script to investigation of lateral-directional trim in a crosswind for the Cessna 182. It uses the constants defined by Cessna182.m.

Utilities

1. Cessna182.m

```
function [constant]=Cessna182
% function [constant]=Cessna182
% Version 1.0, 2/14/01
% Make a vector of constants for the Cessna 182 aircraft
% for the cruise configuration. Ref Roskam Page 480
% The order of these constants is critical. Do not change the order
% or code in other simulation functions will be in error.
```

2. QuickTrim.m

   This is a quick and dirty trim routine for the Cessna 182. It is not very general and should probably be rewritten to make fewer assumptions.

```
function [x,u,CL,CD,CM,alphadeg]=QuickTrim(Vt,Hp,constant)
% function [x,u,CL,CD,CT,CM,alphadeg]=QuickTrim(Vt,Hp,constant)
% Quick and dirty trim routine.
% constant is an array of aircraft specific sconstants.
% Vt and Hp are trim true airspeed (ft/sec) and pressure altutude (ft).
% x is the 12x1 state vector which is initialized in this routine.
% u is the 4x1 control vector which is also initialized by this routine.
% CL and CD are trim lift and drag coefficient.
% alphadeg is the trim angle of attack in degrees.
% The trim condition is assumed to be straight-line flight with
% wings level, constant attitude,
% constant altitude, and non-accelerating.
% It assumes CY0, Cl0, CN0 are all zero and an initial northerly heading.
% Flight in the troposphere is assumed.
% This is not the most general trim routine. It is a quick-trim routine.
% It also assumes that
% Xcg=xref and that dT=0 (thrust inclination angle) and that
% alpha is very small.
```

3. aircraft.m

   An s-function programmed using Simulink protocol containing the 12th order nonlinear equations of motion. These equations are not aircraft specific. However, aircraft.m calls aeroforce.m, aeromoment.m, thrust.m. These in turn use the constants defined by Cessna182.m.

```
function [sys,x0,str,ts] = aircraft(t,x,u,flag,constant,xIC)
%   function [sys,x0,str,ts] = aircraft(t,x,u,flag,constant,xIC)
%   Version 1.0, 2/14/01
%   S-file for six degree of freedom aircraft simulation.
%   constant is a vector of constants that describe the aircraft.
%   xIC is the initial condition on the state vector.
%   x=[U,V,W,P,Q,R,Phi,Theta,Psi,Xprime,Yprime,Hprime]
% Units are [ft/sec, ft/sec, ft/sec, rad/sec, rad/sec, rad/sec, rad, rad, rad, ft, ft, ft]
% u=[deltaE,deltaA,deltaR,bhp]
% Units are [rad,rad,rad,bhp]
```

4. aeroforce.m (compute aerodynamic forces components, , dimensional and nondimensional)

```
function
[Fax,Fay,Fazprime,CX,CY,CZprime]=aeroforce(x,u,constant,alpha,beta,Vt,qbar)
% function
[Fax,Fay,Fazprime,CX,CY,CZprime]=aeroforce(x,u,constant,alpha,beta,Vt,qbar)
% Version 1.0, 2/14/01
% Compute aircraft aerodynamic forces in the body axis system.
% FAvector=Fax*i+Fay*j+Faz*k
% CX, CY, CZprime are nondimensional versions of Fax, Fay, and Fazprime.
% x=[U,V,W,P,Q,R,Phi,Theta,Psi,Xprime,Yprime,Hprime]
% Units are [ft/sec, ft/sec, ft/sec, rad/sec, rad/sec, rad/sec, rad, rad, rad, ft, ft, ft]
% u=[deltaE,deltaA,deltaR,deltaT]
% Units are [rad,rad,rad,bhp]
% Fazprime does not contain the CLalphadot term.
```

5. aeromoment.m (compute aerodynamic moment components, dimensional and nondimensional)

```
function
[La,Ma,Na,Croll,Cpitch,Cyaw]=aeromoment(x,u,constant,alpha,beta,Vt,qbar,alph
adot,CX,CY,CZprime);
% function
[La,Ma,Na,Croll,Cpitch,Cyaw]=aeromoment(x,u,constant,alpha,beta,Vt,qbar,alph
adot,CX,CY,CZprime);
% Version 1.0, 2/14/01
% Compute aircraft aerodynamic moments in the body axis system about the c.g.
% x=[U,V,W,P,Q,R,Phi,Theta,Psi,Xprime,Yprime,Hprime]
% Units are [ft/sec, ft/sec, ft/sec, rad/sec, rad/sec, rad/sec, rad, rad, rad, ft, ft, ft]
% u=[deltaE,deltaA,deltaR,deltaT]
% Units are [rad,rad,rad,bhp]
```

8. thrust.m (compute thrust force and moment components)

```
function [Ftx,Fty,Ftz,Lt,Mt,Nt]=thrust(u,constant,Vt)
% [Ftx,Fty,Ftz,Lt,Mt,Nt]=thrust(u,constant,Vt)
% u(4)=bhp (hp)
% Vt=airspeed in ft/sec
% constant(23)=phiT, thrust inclination angle, RADIANS
% constant(24)=dT, thrust offset distance, ft
% constant(8)= eta, propeller efficiency
% Propeller efficiency is assumed constant.
```

9. TransformB2NED .m

```
function [TransMatrixB2NED]=TransformB2NED(phi,theta,psi)
% function [TransMatrixB2NED]=TransformB2NED(phi,theta,psi)
% Generate the transformation matrix to transform a vector from
% body axis coordinates to North-East-Down coordinates.
% V(NED)=TransMatrixB2NED*V(body)
```

10. 4. TransformB2Wind .m

```
function [TransMatrixB2Wind]=TransformB2Wind(alpha,beta)
% function [TransMatrixB2Wind]=TransformB2Wind(alpha,beta)
% Generate the transformation matrix to transform a vector from
% body-axis coordinates to wind-axis coordinates.
% V(wind)=TransMatrixB2Wind*V(body)
```

11. TransformNED2B.m

```
function [TransMatrixNED2B]=TransformNED2B(phi,theta,psi)
% function [TransMatrixNED2B]=TransformNED2B(phi,theta,psi)
% Generate the transformation matrix to transform a vector from
% North-East-Down coordinates to body axis coordinates.
% V(body)=TransMatrixNED2B*V(NED)
```

12. TransformWind2B.m

```
function [TransMatrixWind2B]=TransformWind2B(alpha,beta)
% function [TransMatrixWind2B]=TransformWind2B(alpha,beta)
% Generate the transformation matrix to transform a vector from
% wind-axis coordinates to body-axis coordinates.
% V(body)=TransMatrixWind2B*V(wind)
```

Reference

Jan Roskam, Airplane Flight Dynamics and Automatic Flight Controls, Part 1, 1994.