SUMMARY:

This design report outlines the framework of the non-linear simulation. It outlines the equations of motion that the simulation uses and the changes from the linear to the non-linear model Simulink framework. A description of how to run the model is provided for future operators as well as a block diagram of the model and the scripts of the functions required.

Revision History

Version	Date	Description
A	05 APR 09	Initial Release

Table of Contents

1.0 SCOPE **3**

2.0 REFERENCES **3**

3.0 DYNAMIC SIMULATION **3**

3.1 Non-linear equations of motion 4

3.2 The non-linear model framework 5

3.3 Running the simulation 7

3.4 The Find_Trim and cost.m functions 8

4.0 CONCLUSIONS **8**

5.0 RECOMMENDATIONS **8**

APPENDICES **9**

A: Geo_model.m 9

B: Non-linear model block diagram 15

C: Geo_Find_Trim.m 16

D: Geo_cost.m 16

1.0 Scope

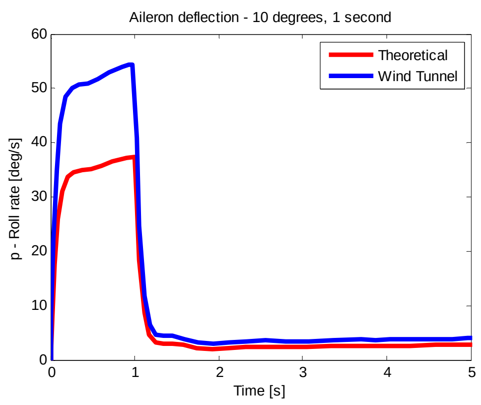
This design report outlines the framework of the non-linear simulation. It outlines the equations of motion that the simulation uses and the changes from the linear to the non-linear model Simulink framework. A description of how to run the model is provided for future operators as well as a block diagram of the model and the scripts of the functions required.

2.0 References

1. DR 83-14
2. DR 93-03
3. Koning, J. *Report - GeoSurv II flight control system project,* Carleton University / Delft University of Technology, July 2007

3.0 Dynamic Simulation

Dynamic simulation is a tool that the design team can use to simulate the flight of the GeoSurv II during the design face. This has a broad range of applications during the design process. It can be used to aid in control surface sizing and as a preliminary flight test bed for pilot testing and training. The model currently in use was developed by Justing Koning and Zhe Yan in previous project years. Koning developed a set of non-linear equations of motion and developed a simulation tool in MATLAB. It can simulate the effect of control surface deflection. It can also be used to test and verify wind tunnel data and compare any discrepancies between theoretical and experimental stability derivatives as well as compare aircraft performance with other aircraft. An example of this is the rolling moment due to aileron deflection. When comparing the theoretical values with wind tunnel data we get Figure 1 below.



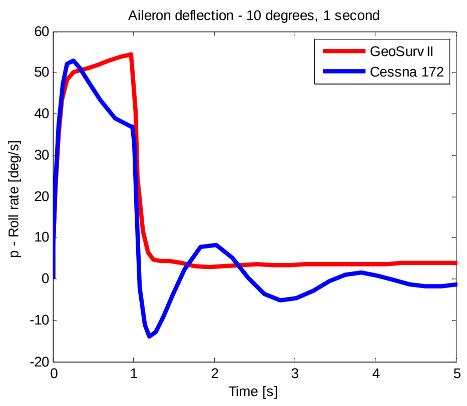


Figure 1: Roll rate comparison

In the future this tool will be useful for comparison between the dynamic model and prototype flight data. This will enable validation of all the aerodynamic design work done to date.

Last project year Yan was able to use the linear equations of motion in a Simulink model which takes input in real time and outputs a 3D visualization of the aircraft. The details of his work can be found in DR 83-14.

The linear simulation, however, has limitations. It is only valid for small perturbations. The model can only simulate small control surface deflections and stall cannot be simulated. The goal for this year was to further develop the model into a non-linear model. This would allow much more flexibility. Adverse weather conditions like high winds and gust conditions could be simulated as well as larger control surface deflections.

3.1 Non-linear equations of motion

The first step in developing the non-linear model was to develop the equations of motion that would be programmed into the MATLAB environment. The detailed derivation of the equations of motion in the longitudinal direction can be seen in DR 93-03. A similar derivation was performed for the lateral direction and the resulting equations that will be used in the non-linear model are:

Translational

$$\dot{u} = \frac{1}{m}\left[L\sin\alpha - D\cos\alpha + T\right] - g\sin\theta + vr - qw$$

$$\dot{v} = \frac{1}{m}\left[Y\cos\beta\right] + g\sin\phi\cos\theta + wp - ur$$

$$\dot{w} = \frac{1}{m}\left[-D\sin\alpha - L\cos\alpha\right] + g\cos\phi\cos\theta + uq - vp$$

where the lift, drag and side-force and non-dimensionalized as:

$$L = C_L\bar{q}S \qquad C_L = C_{L_\alpha}\alpha + C_{L_{\delta e}}\delta e + C_{L_{\delta f}}\delta f$$

$$D = C_D\bar{q}S \qquad C_D = C_{D_0} + C_{D_L}$$

$$Y = C_Y\bar{q}S \qquad C_Y = C_{Y_\beta}\beta + C_{Y_{\delta r}}\delta r + C_{Y_r}r$$

Rotational

$$\dot{p} = \frac{\ell}{I_{xx}} - \frac{\left(I_{zz} - I_{yy}\right)}{I_{xx}}rq$$

$$\dot{q} = \frac{M}{I_{yy}} - \frac{\left(I_{xx} - I_{zz}\right)}{I_{yy}}pr$$

$$\dot{r} = \frac{N}{I_{zz}} - \frac{\left(I_{yy} - I_{xx}\right)}{I_{zz}}pq$$

and the roll, pitch and yaw moments are non-dimensionalized as:

$$\ell = C_\ell\bar{q}Sb \qquad C_\ell = C_{\ell_{\delta a}}\delta a + C_{\ell_{\delta r}}\delta r + C_{\ell_\beta}\beta + C_{\ell_r}r + C_{\ell_p}p$$

$$M = C_M\bar{q}S\bar{c} \qquad C_M = C_{M_\alpha}\alpha + C_{M_{\delta e}}\delta e + C_{M_{\delta f}}\delta f + C_{M_q}q$$

$$N = C_N\bar{q}Sb \qquad C_N = C_{N_\beta}\beta + C_{N_{\delta r}}\delta r$$

These equations are valid at all times for the aircraft. With these equations the GeoSurv II can be much more accurately modeled over many more flight conditions than was previously possible.

3.2 The non-linear model framework

The equations of motion were then programmed into a MATLAB m-file that could be incorporated into a Simulink model. Like the linear model that preceded it, this model accepts inputs from a USB R/C device and outputs a 3D visualization in the Flightgear interface. A more detailed description of the workings of this system can be found in DR 83-14 which presents the framework for the linear model system as shown below in Figure 2.

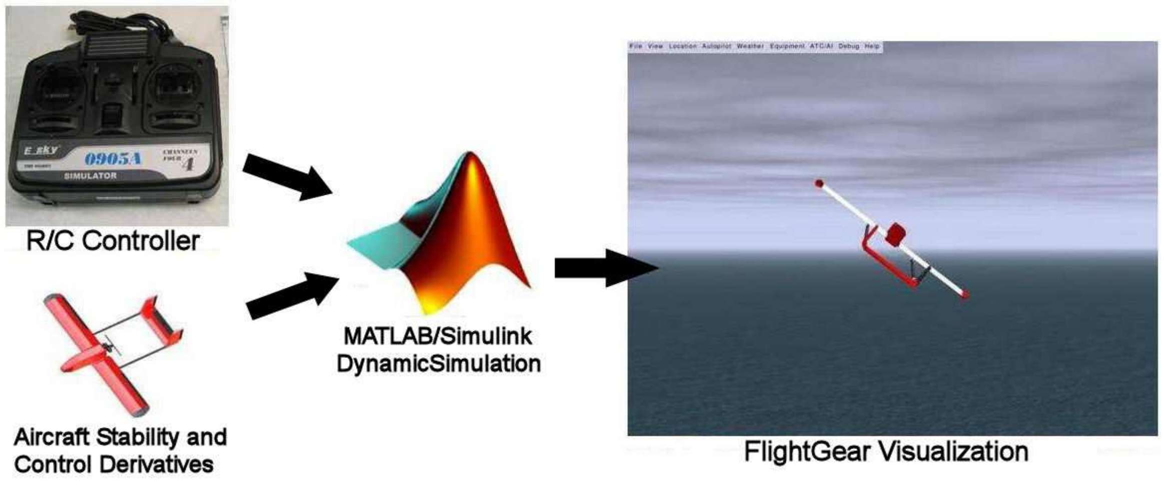

R/C Controller
Aircraft Stability and Control Derivatives
MATLAB/Simulink DynamicSimulation
FlightGear Visualization

Figure 2: Dynamic simulation overview

The main difference between the old linear model and the new non-linear model is the equations of motion block. The linear EOM block takes inputs of control surface deflections and outputs a forward speed, angle of attack, side-slip, pitch Euler angle, roll, pitch and yaw as shown in Figure 3.

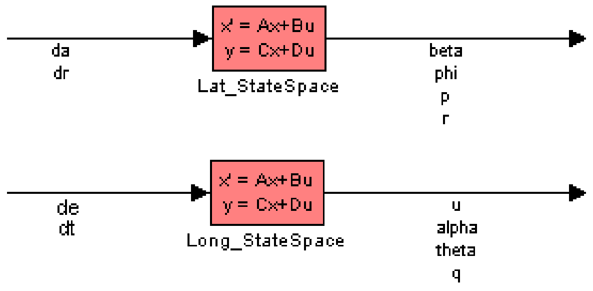


Figure 3: Linear equations of motion blocks

The non-linear EOM block takes inputs of control surfaces as well as the current speed and the orientation of the aircraft in the body and inertial frames. The outputs are the translational accelerations as well as the rotational accelerations in the body and inertial frames. This can be seen in Figure 4.

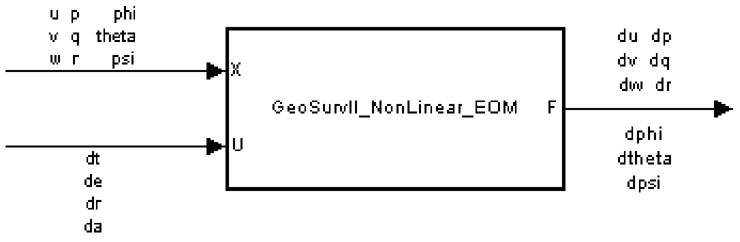


Figure 4: Linear equations of motion block

This EOM block expresses the non-linear equations in a format that MATLAB can manipulate. It uses the function m-file *Geo_model.m*. The script for *Geo_model.m* is displayed in Appendix A. There are two versions of the Simulink framework which can be used for analysis of the aircraft. The first, *GeoSurvII_NonLinear_RC.mdl* takes input from a USB R/C device and outputs to the FlightGear 3D visualization software. The second, *GeoSurvII_NonLinear_Signal.mdl* uses signal builder blocks to provide any specific input such as a step or doublet input. Again, this outputs to the FlightGear software. A block diagram of the Simulink framework can be seen in Appendix B. For more analytical purposes, placing scopes will allow output of plots of any parameter one desires. These include:

- Airspeed
- Ground speed
- Altitude
- Longitude and latitude
- Control surface deflections
- Roll, pitch and yaw rates

With this it should be possible to compare the aircraft performance to flight data, the linear model and other aircraft.

3.3 Running the simulation

The following instructions will describe how to run the non-linear simulation. T AeroSim blockset and FlightGear need to be installed in order to run the simulation. Starting in the *Dynamic Simulation\Non-linear* folder:

- Open one of the *.mdl* files.
- Enter a value for *dt* in the MATLAB command window. This will determine the speed of the simulation. *dt = 0.005* is a generally a good value.
- Run *Geo_Find_Trim.m*. This finds trim conditions.
- Enter *Geo_model(Xo,Uo)* in the MATLAB command window and make sure that the results are zero. This means the correct trim settings have been found by Find_Trim.
- In the FlightGear Advanced options:
    - *General* page, set *Control* to 'joystick'
    - *Flight Model* page, set *FDM* to 'external', and set *Model Hz* to '200'
    - *Initial Position page*, set to corresponding model values

  - *Input/Output* page, set *Protocol* to 'native-fdm', set *Medium* to 'socket', set *Direction* to 'in', set *Hz* to '200', set *Hostname* to '"localhost"', set *Port* to '5500', and select 'UDP'
- The model can now be run.

3.4 The Find_Trim and cost.m functions

The *Geo_Find_Trim.m* function is a trim finding routine that allows the operator to set the initial flight conditions of the aircraft. It uses the *cost.m* function to find the inputs for which the *Geo_model.m* outputs (the translational and rotational accelerations) are zero. Any speed or orientation value can be selected to allow the initial conditions to be any steady flight condition (level, climb, turn). Once a parameter has been modified in the Find_Trim function a constraint must be added to the cost function following the same format as the speed and theta angles already in the script. The script can be found in Appendix C.

4.0 Conclusions

The GeoSurv II Dynamic Simulation has been further developed from a linear model to a non-linear model. This will allow for much more accurate simulation of the GeoSurv II aircraft and provides much more flexibility for the designers to perform analyses.

5.0 Recommendations

After flight of the prototype, flight data should be compared to simulation for model validation. In addition, simulation of failure modes such as control surface failures should be simulated as well as demonstration of requirements for aircraft certification. Work should be done with the avionics group to incorporate the autopilot into the model. This will aid in autopilot refinement and full automated flight testing in a virtual environment. GeoSurv II sensors and their respective errors should be incorporated into the model to simulate the ability of the GeoSurv II to follow a pre-programmed flight path relying on sensors for navigation.

Appendices

A: Geo_model.m

```
function XDOT = Geo_model(X,U)

%GEO_MODEL Full 6 DOF model for GeoSurv II aircraft
%
%   [XDOT] = GEO_MODEL(X,U) returns the state derivatives, XDOT, given the
%   current state and inputs, X and U.  The constants are the nominal
%   constants.
%
%   State vector X and input vector U is defined as
%
%       X = [u;v;w;p;q;r;phi;theta;psi] = [x1;x2;x3;x4;x5;x6;x7;x8;x9]
%       U = [d_A;d_E;d_R;d_th] = [u1;u2;u3;u4]
%
%
%Created by Christopher Lum
%lum@u.washington.edu
%Modified for GeoSurv II aircraft by Guillaume Blouin


%-----------------------CONSTANTS-------------------------------
%Unit conversions
lb_to_kg = 0.45359237;          %Unit conversion (pounds to kilograms)
in_to_m = 0.0254;               %Unit conversion (inches to meters)
ft_to_m = 0.3048;               %Unit conversion (feet to meters)
kt_to_ms = 0.514444;            %Unit conversion (knots to meters/second)
slug_to_kg = 14.593903;         %Unit conversion (slugs to kilograms)

%Mass moments of inertia
Ixx = 25.2589*slug_to_kg*ft_to_m^2;
Iyy = 32.8379*slug_to_kg*ft_to_m^2;
Izz = 47.0740*slug_to_kg*ft_to_m^2;
Ixz = 0*slug_to_kg*ft_to_m^2;

%Nominal vehicle constants (problem 1)
m = 186*lb_to_kg;               %Aircraft total mass (kg)
cbar = 27*in_to_m;              %Mean Aerodynamic Chord (m)
lt = 108.55*in_to_m;            %Distance by AC of tail and body (m)
S = 36*ft_to_m^2;               %Wing planform area (m^2)
St = 9.5*ft_to_m^2;             %Tail planform area (m^2)
i = -2*pi/180;                  %Tail incidence angle (rad)
AR = 7.11;                      %Aspect ratio
e = 0.836;                      %Oswalt's efficiency factor

Xcg = 0.25*cbar;                %x position of CoG in Fm (m)
Ycg = 0;                        %y position of CoG in Fm (m)
Zcg = 0.037*cbar;               %z position of CoG in Fm (m)

%Engine inputs
Xapt = 0;                       %x position of engine force in Fm (m)
Yapt = 0;                       %y position of engine force in Fm (m)
```

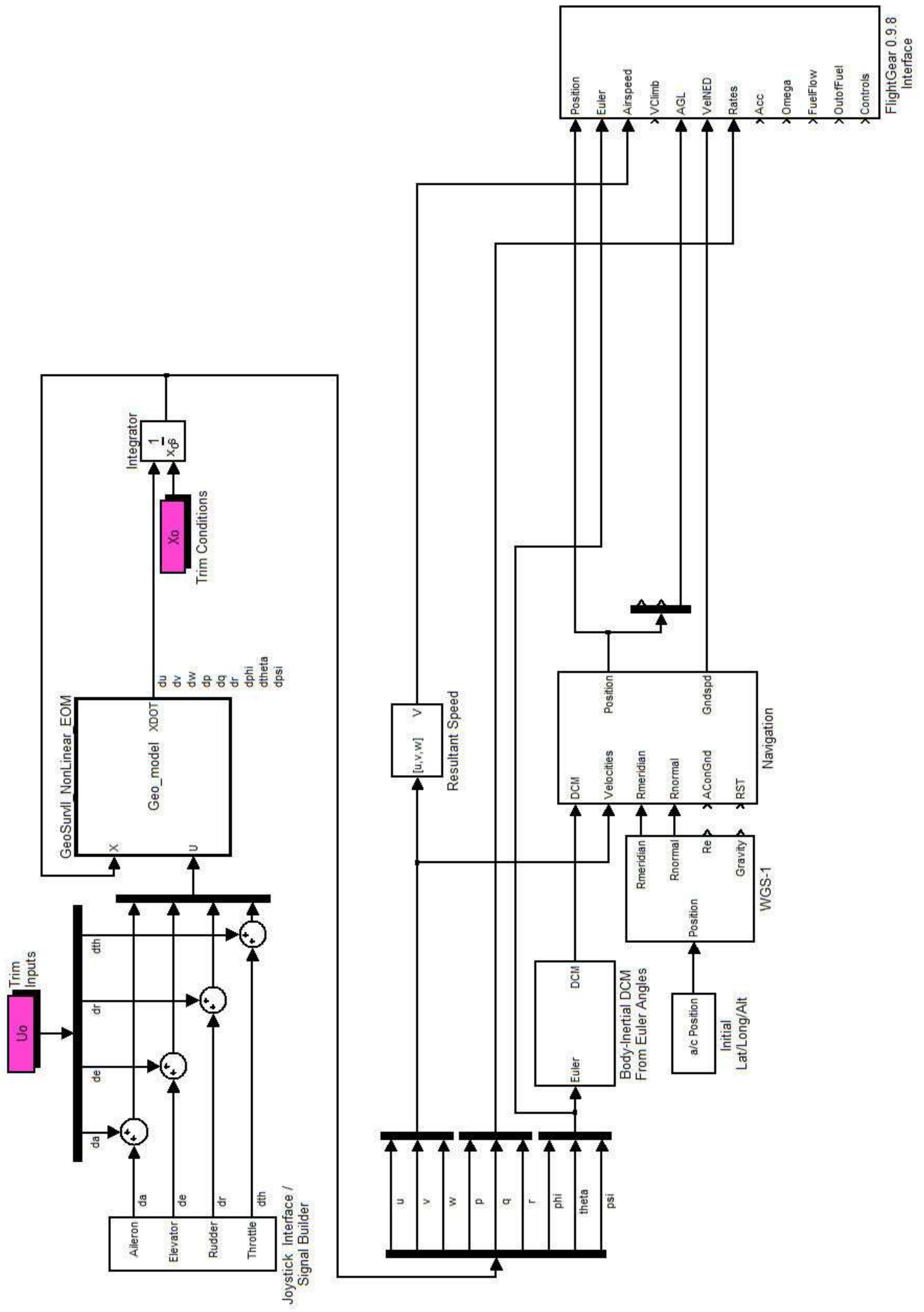
```
Zapt = 0.1405;                  %z position of engine force in Fm (m)

%Other constants
rho = 1.225;                    %Air density (kg/m^3)
g = 9.81;                       %Gravitational acceleration (m/s^2)
depsda = 0.33;                  %Change in downwash w.r.t. alpha (rad/rad)
alpha_L0 = -4*pi/180;           %Zero lift angle of attack (rad)

%Stability Derivatives (output from Justin Koning's Linear Model)
CXu = -0.096;
CXa = 0.15632;
CXadot = 0;
CXq = 0;
CXde = 0;

CZu = -0.85219;
CZa = -5.9298;
CZadot = -1.1763;
CZq = -5.6462;
CZde = -0.50358;

Cmu = 0;
Cma = -0.6094;
Cmadot = -4.496;
Cmq = -14.9865;
Cmde = -1.9304;

CYb = -0.5291;
CYbdot = 0;
CYp = 0.022775;
CYr = 0.4475;
CYda = 0;
CYdr = 0.13127;

Clb = -0.029855;
Clp = -0.49849;
Clr = 0.12639;
Clda = 0.2091;
Cldr = 0.0046136;

Cnb = 0.20858;
Cnbdot = 0;
Cnp = -0.057436;
Cnr = -0.21178;
Cnda = -0.006211;
Cndr = -0.0531;

%Convert CZ to CL
CLu = -CZu;
CLa = -CZa;
CLadot = -CZadot;
CLq = -CZq;
CLde = -CZde;
CLa_t = 3.544;
```

```
%Other Stability Derivatives (from AERO database)
CDo = 0.0222;
CMo = -0.1003;


%--------------------------STATE VECTOR----------------------------------
%Extract state vector
x1 = X(1);                              %u
x2 = X(2);                              %v
x3 = X(3);                              %w
x4 = X(4);                              %p
x5 = X(5);                              %q
x6 = X(6);                              %r
x7 = X(7);                              %phi
x8 = X(8);                              %theta
x9 = X(9);                              %psi

u1 = U(1);                              %d_A (aileron)
u2 = U(2);                              %d_E (elevator)
u3 = U(3);                              %d_R (rudder)
u4 = U(4);                              %d_th (throttle)


%---------------INTERMEDIATE VARIABLES------------------------
%Calculate airspeed
Va = sqrt(x1^2 + x2^2 + x3^2);

%Calculate alpha and beta
alpha = atan(x3/x1);
beta = asin(x2/Va);

%Calculate dynamic pressure
Q = 0.5*rho*Va^2;

%Also define the vectors wbe_b and V_b
wbe_b = [x4;x5;x6];
V_b = [x1;x2;x3];


%---------------AERODYNAMIC FORCE COEFFICIENTS----------------
%Calculate the CL_wb
if Va<(35.6*kt_to_ms)
end

if alpha<= 17*pi*180
    CL_wb=CLa*(alpha-alpha_L0);
elseif alpha>17*pi*180
    CL_wb=-5*10^-5*alpha^4+0.0034*alpha^3-0.092*alpha^2+1.1374*alpha-4.2971;
end

%Calculate CL_t
epsilon = depsda*(alpha - alpha_L0);
alpha_t = alpha - epsilon + i + CLq*x5*lt/Va;
CL_t = CLa_t*alpha_t + CLde*u2;
```

```
%Total lift force
CL = CL_wb + CL_t*(St/S);

%Total drag force (neglecting tail)
CD = CDo + CL^2/(pi*AR*e);

%Calculate sideforce
CY = CYb*beta + CYdr*u3 + CYr*x6;


%--------------DIMENSIONAL AERODYNAMIC FORCES---------------------
%Calculate the actual dimensional forces.  These are in F_s (stability axis)
FA_s = [-CD*Q*S;
         CY*Q*S;
        -CL*Q*S];

%Rotate these forces to F_b (body axis)
C_bs = [cos(alpha) 0 -sin(alpha);
        0 1 0;
        sin(alpha) 0 cos(alpha)];

FA_b = C_bs*FA_s;


%-------------AERODYNAMIC MOMENT COEFFICIENTS---------------
%Calculate the moments in Fb.  Define eta, dCMdx and dCMdu
eta11 = Clb*beta;
eta21 = CMo - (CLa_t*(St*lt)/(S*cbar))*(alpha - epsilon);
eta31 = Cnb*beta;

eta = [eta11;
       eta21;
       eta31];

dCMdx = (cbar/Va)*[Clp 0 Clr;
                   0 (Cmq*(St*lt^2)/(S*cbar^2)) 0;
                   Cnp 0 Cnr];

dCMdu = [Clda 0 Cldr;
          0 (-CLa_t*(St*lt)/(S*cbar)) 0;
          0 0 Cndr];

%Now calculate CM = [Cl;Cm;Cn] about Aerodynamic center in Fb
CMac_b = eta + dCMdx*wbe_b + dCMdu*[u1;u2;u3];

%Transfer now to CoG.
CX = -CD*cos(alpha) + CL*sin(alpha);
CZ = -CL*cos(alpha) - CD*sin(alpha);

A = (1/cbar)*[0 CZ -CY;
              -CZ 0 CX;
              CY -CX 0];

zeta = [(Xcg-0.12*cbar);
        -Ycg;
        Zcg];

CMcg_b = CMac_b + A*zeta;


%---------------DIMENSIONAL AERODYNAMIC MOMENTS---------------------
%Calculate the dimensional moments in Fb about cg
MAcg_b = CMcg_b*(Q*S*cbar);


%-----------------ENGINE FORCE & MOMENT----------------------------
%Now effect of engine.  First, calculate the thrust of engine
F = u4*m*g;

%Assuming that engine thrust is aligned with Fb, we have
FE_b = [F;0;0];

%Now engine moment due to offset of engine thrust from CoG.
mew = [Xcg - Xapt;
        Yapt - Ycg;
        Zcg - Zapt];


MEcg_b = cross(mew,FE_b);


%--------------------GRAVITY EFFECTS--------------------------------
%Calculate gravitational forces in the body frame.  This causes no moment
%about CoG.
g_b = [-g*sin(x8);
        g*cos(x8)*sin(x7);
        g*cos(x8)*cos(x7)];

Fg_b = m*g_b;


%-------------------STATE DERIVATIVES------------------------------
%Inertia matrix
Ib = m*[Ixx 0 Ixz;
        0 Iyy 0;
        Ixz 0 Izz];

%Inverse of inertia matrix
invIb = inv(Ib);

%Form F_b (all the forces in Fb) and calculate udot, vdot, wdot
F_b = Fg_b + FE_b + FA_b;
x1to3dot = (1/m)*F_b - cross(wbe_b,V_b);

%Form Mcg_b (all moments about CoG in Fb) and calculate pdot, qdot, rdot.
Mcg_b = MAcg_b + MEcg_b;
x4to6dot = invIb*(Mcg_b - cross(wbe_b,Ib*wbe_b));

%Calculate phidot,thetadot, and psidot
H_phi = [1 sin(x7)*tan(x8) cos(x7)*tan(x8);
```

```
         0 cos(x7) -sin(x7);
         0 sin(x7)/cos(x8) cos(x7)/cos(x8)];

x7to9dot = H_phi*wbe_b;

%Place in first order form
XDOT = [x1to3dot(1);                       %udot
        x1to3dot(2);                       %vdot
        x1to3dot(3);                       %wdot
        x4to6dot(1);                       %pdot
        x4to6dot(2);                       %qdot
        x4to6dot(3);                       %rdot
        x7to9dot(1);                       %phidot
        x7to9dot(2);                       %thetadot
        x7to9dot(3)];                      %psidot
```

B: Non-linear model block diagram



C: Geo_Find_Trim.m

```
%%
%unit conversion
kt_to_ms = 0.514444;
x1 = 60*kt_to_ms;                    %u
x2 = 0;                              %v
x3 = 0;                              %w
x4 = 0;                              %p
x5 = 0;                              %q
x6 = 0;                              %r
x7 = 0;                              %phi
x8 = 0;                              %theta
x9 = 0;                              %psi

u1 = 0;                              %d_A (aileron)
u2 = 0;                              %d_E (elevator)
u3 = 0;                              %d_R (rudder)
u4 = 0;                              %d_th (throttle)
guess=[x1,x2,x3,x4,x5,x6,x7,x8,x9,u1,u2,u3,u4];trim_values=fminsearch(@Geo_cost,guess,...
    optimset('TolX',1e-10,'MaxFunEvals',10000,'MaxIter',10000));

Xo=trim_values(1:9)
Uo=trim_values(10:13)
```

D: Geo_cost.m

```
function J=Geo_cost(XU);
X=XU(1:9);
U=XU(10:13);
%unit conversion
kt_to_ms = 0.514444;
%calculate derivative of states
xd=Geo_model(X,U);
% add extra constraints for steady flight
a=atan2(X(2),X(1));
theta=X(8);
Va=sqrt(X(1)^2+X(2)^2);
V=60*kt_to_ms;
Q=[xd;Va-V;(theta-a)];

J=Q'*Q;
```