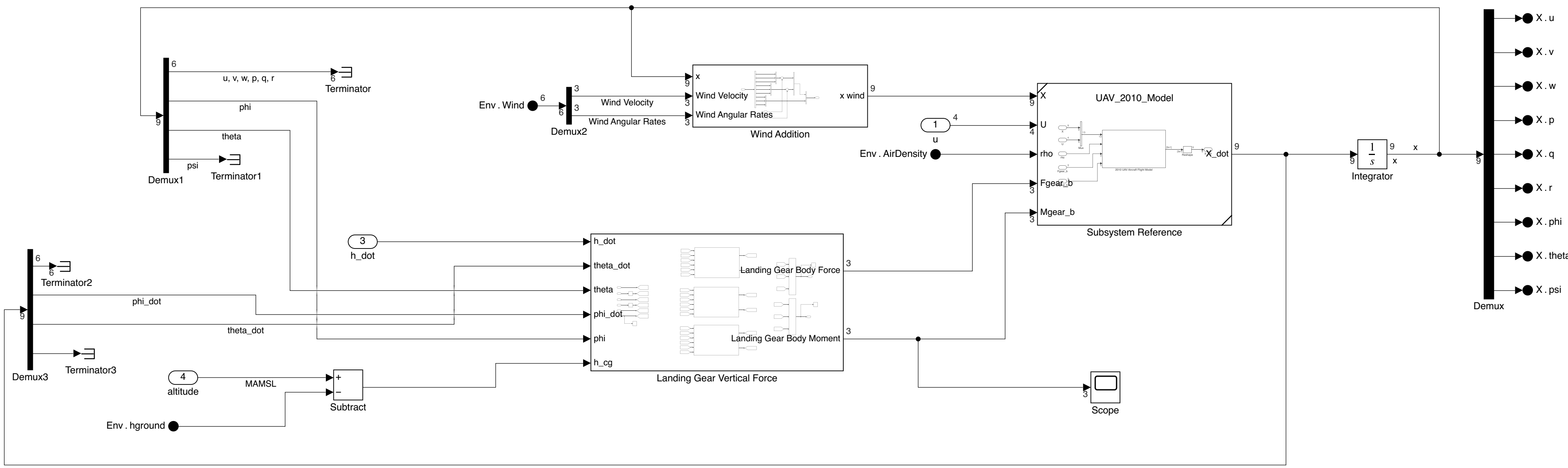
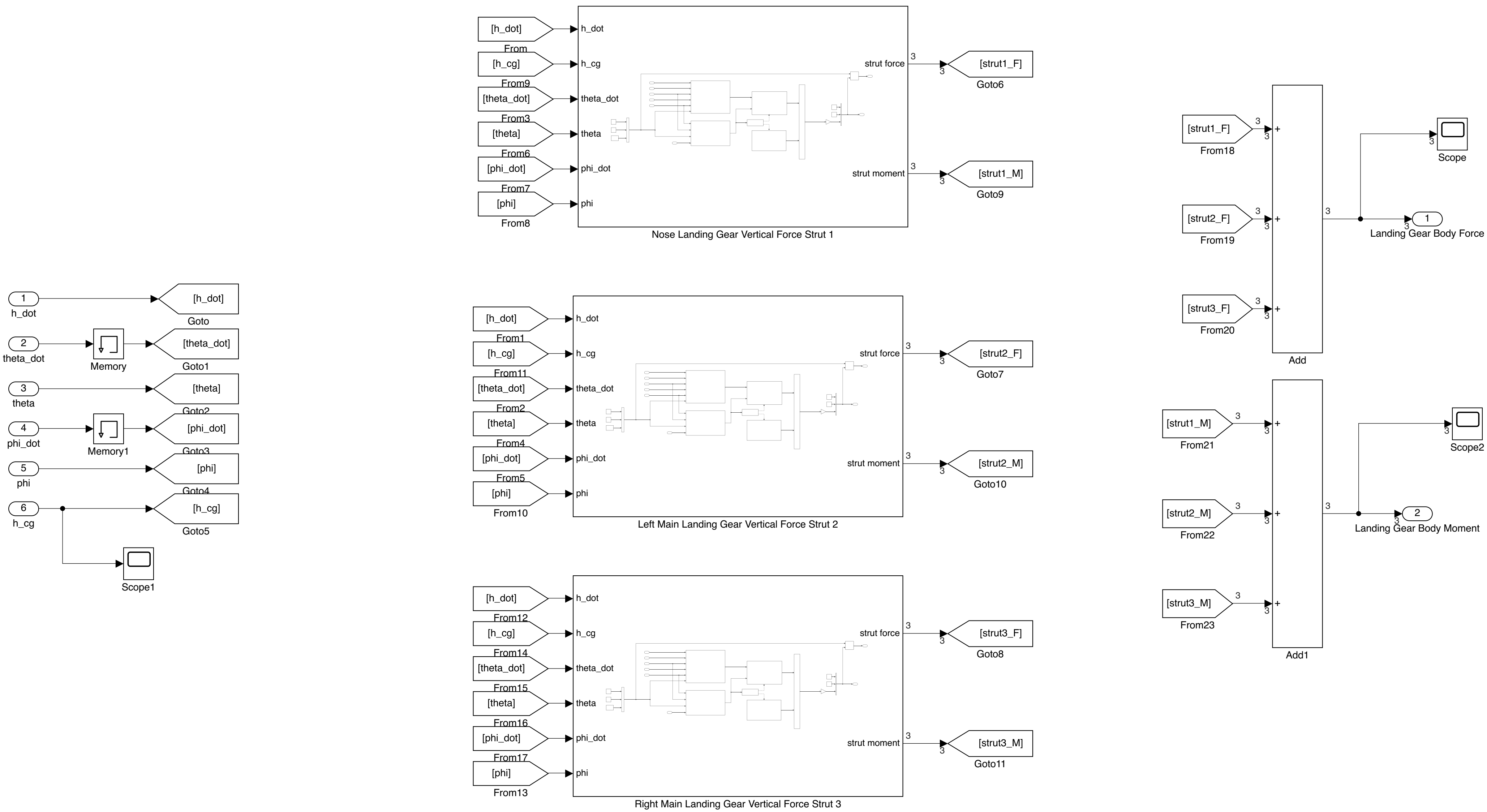
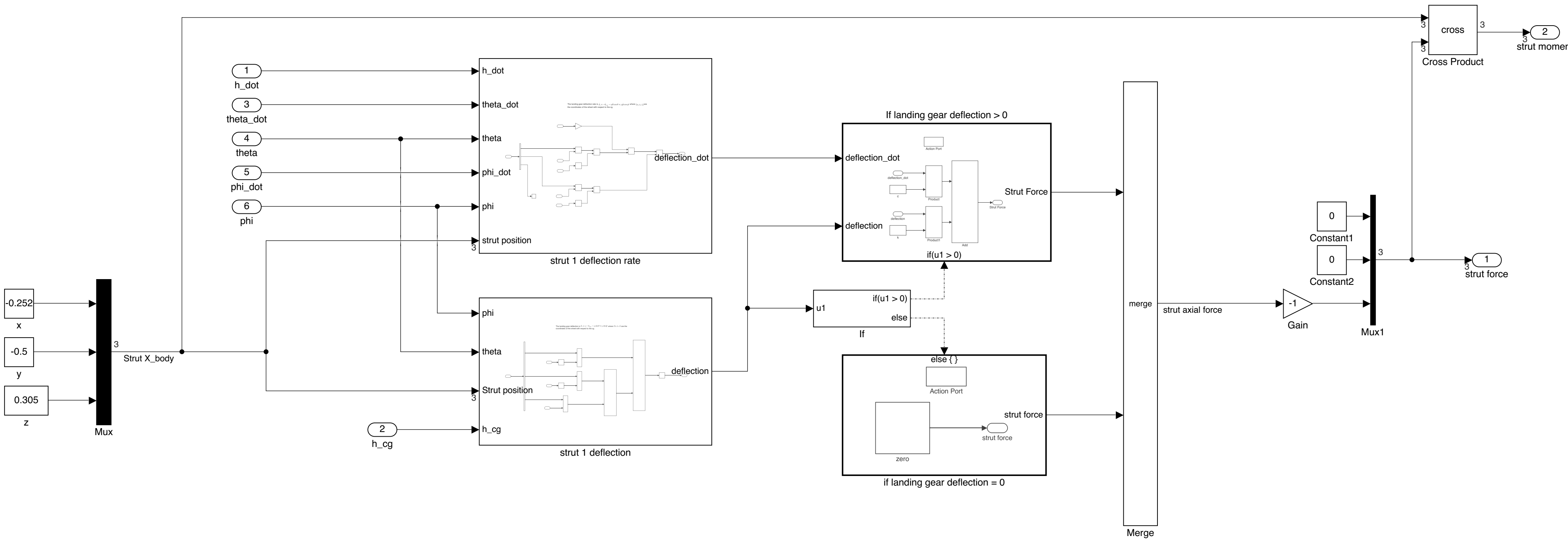


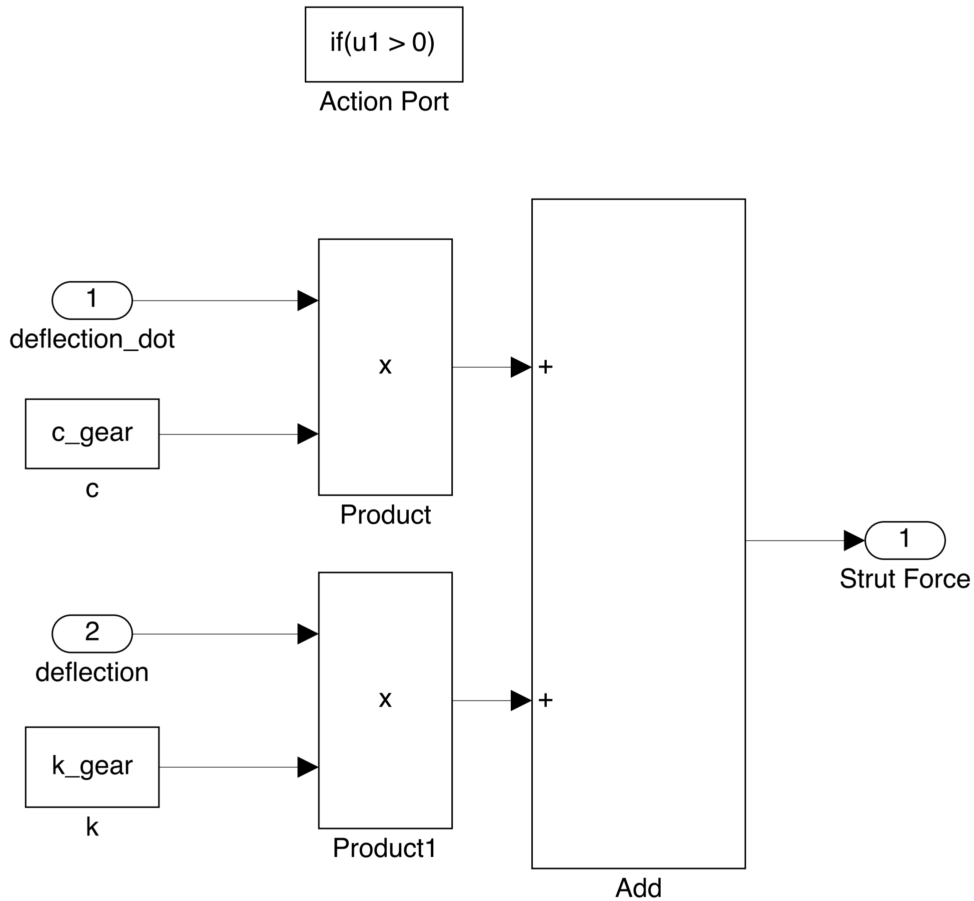
Demux1
u, v, w, p, q, r
Terminator
phi
theta
psi
Terminator1
Env . Wind
Demux2
Wind Velocity
Wind Angular Rates
x
Wind Addition
x wind
UAV_2010_Model
X
U
rho
Fgear_b
Mgear_b
X_dot
Subsystem Reference
u
Env . AirDensity
Integrator
Demux
X . u
X . v
X . w
X . p
X . q
X . r
X . phi
X . theta
X . psi
h_dot
theta_dot
phi_dot
h_cg
Landing Gear Body Force
Landing Gear Body Moment
Landing Gear Vertical Force
Demux3
Terminator2
Terminator3
altitude
MAMSL
Subtract
Env . hground
Scope

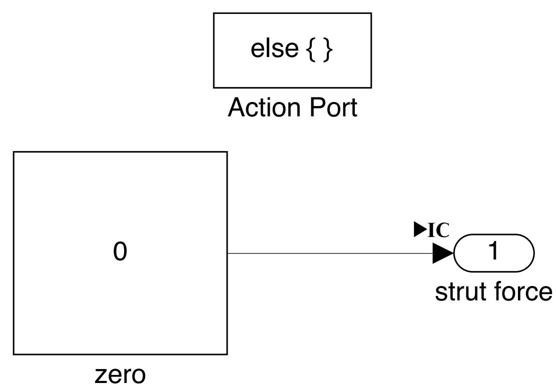


1
h_dot
[h_dot]
Goto
2
theta_dot
Memory
[theta_dot]
Goto1
3
theta
[theta]
Goto2
4
phi_dot
Memory1
[phi_dot]
Goto3
5
phi
[phi]
Goto4
6
h_cg
[h_cg]
Goto5
Scope1
[h_dot]
From
[h_cg]
From9
[theta_dot]
From3
[theta]
From6
[phi_dot]
From7
[phi]
From8
h_dot
h_cg
theta_dot
theta
phi_dot
phi
strut force
strut moment
Nose Landing Gear Vertical Force Strut 1
[strut1_F]
Goto6
[strut1_M]
Goto9
[h_dot]
From1
[h_cg]
From11
[theta_dot]
From2
[theta]
From4
[phi_dot]
From5
[phi]
From10
h_dot
h_cg
theta_dot
theta
phi_dot
phi
strut force
strut moment
Left Main Landing Gear Vertical Force Strut 2
[strut2_F]
Goto7
[strut2_M]
Goto10
[h_dot]
From12
[h_cg]
From14
[theta_dot]
From15
[theta]
From16
[phi_dot]
From17
[phi]
From13
h_dot
h_cg
theta_dot
theta
phi_dot
phi
strut force
strut moment
Right Main Landing Gear Vertical Force Strut 3
[strut3_F]
Goto8
[strut3_M]
Goto11
[strut1_F]
From18
[strut2_F]
From19
[strut3_F]
From20
Add
Scope
1
Landing Gear Body Force
[strut1_M]
From21
[strut2_M]
From22
[strut3_M]
From23
Add1
Scope2
2
Landing Gear Body Moment

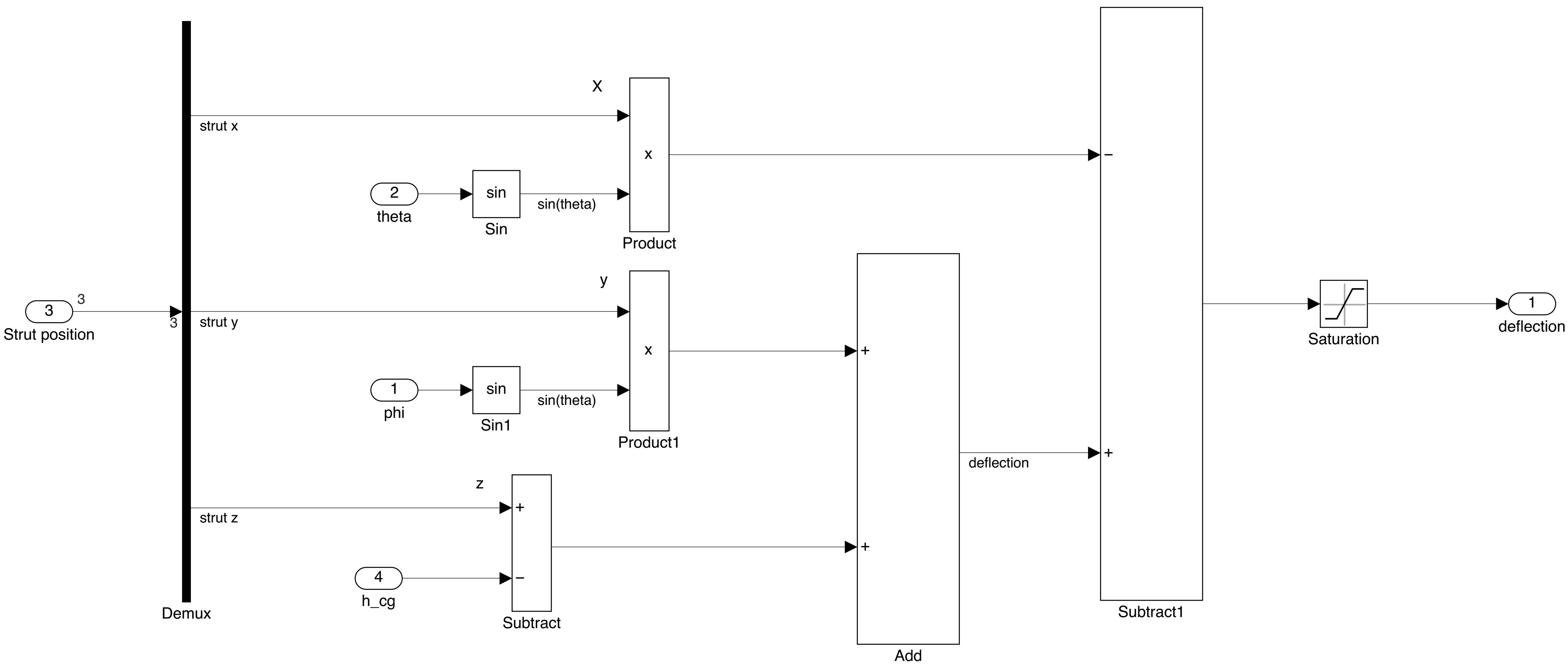


-0.252
x
-0.5
y
0.305
z
Mux
3
Strut X_body
1
h_dot
3
theta_dot
4
theta
5
phi_dot
6
phi
2
h_cg
h_dot
theta_dot
theta
phi_dot
phi
strut position
deflection_dot
strut 1 deflection rate
phi
theta
Strut position
h_cg
deflection
strut 1 deflection
If landing gear deflection > 0
deflection_dot
deflection
Strut Force
if(u1 > 0)
u1
if(u1 > 0)
else
If
else { }
Action Port
zero
strut force
strut force
if landing gear deflection = 0
merge
Merge
strut axial force
-1
Gain
0
Constant1
0
Constant2
Mux1
1
strut force
cross
Cross Product
2
strut moment

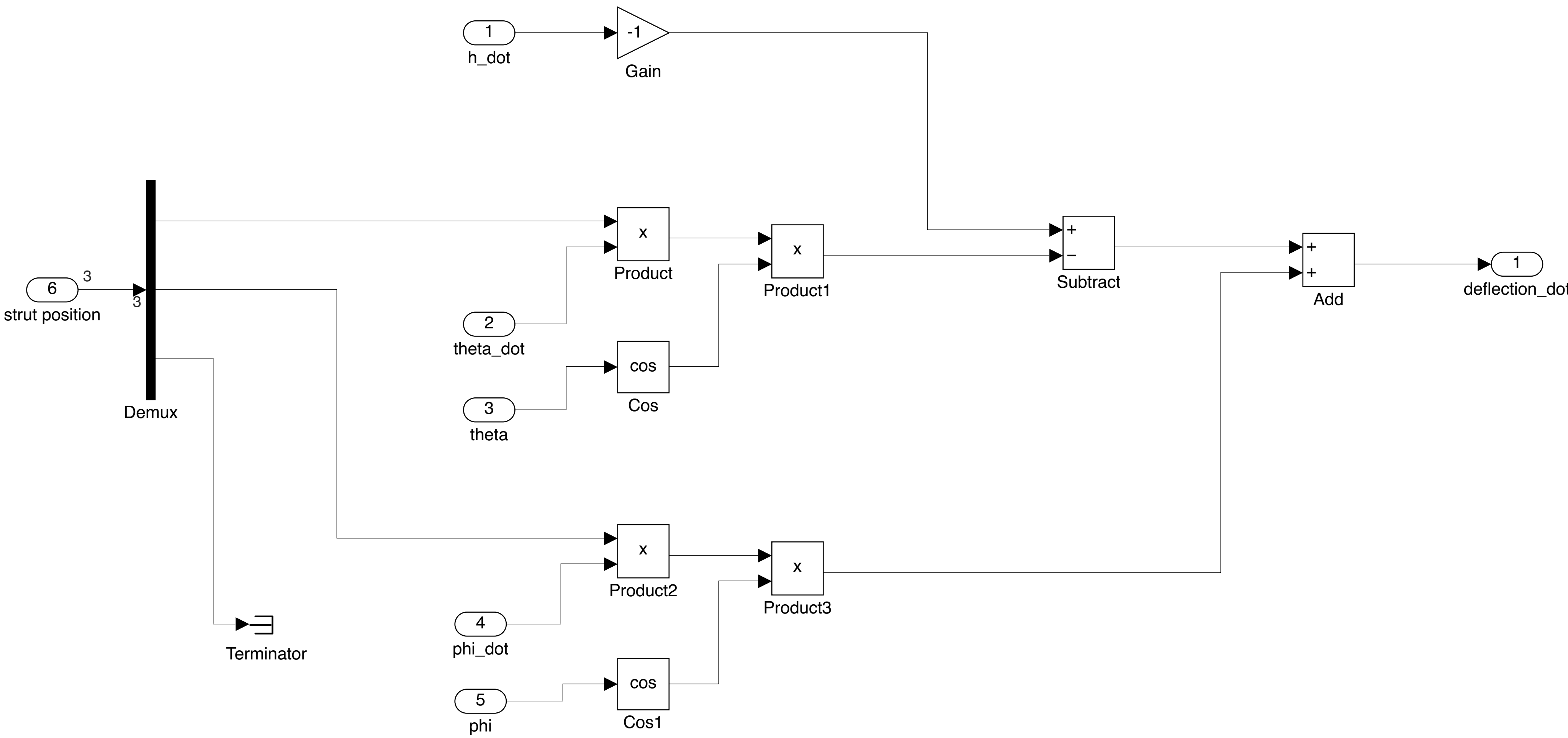

if(u1 > 0)
Action Port
1
deflection_dot
c_gear
c
x
Product
+
2
deflection
k_gear
k
x
Product1
+
Add
1
Strut Force

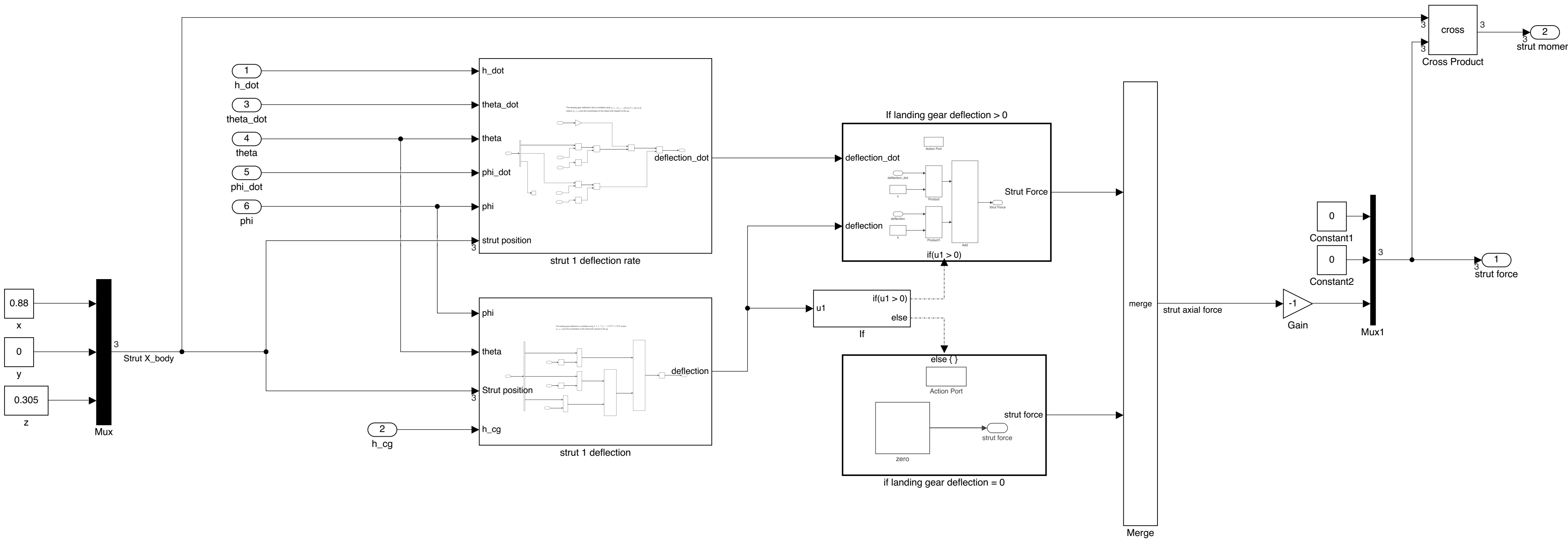

else { }
Action Port
0
zero
IC
1
strut force

The landing gear deflection is $\delta_t = z_i - h_{c.g.} - x_i \sin\theta + y_i \sin\phi$ where $[x_i, y_i, z_i]$ are the coordinates of the wheel with respect to the cg.

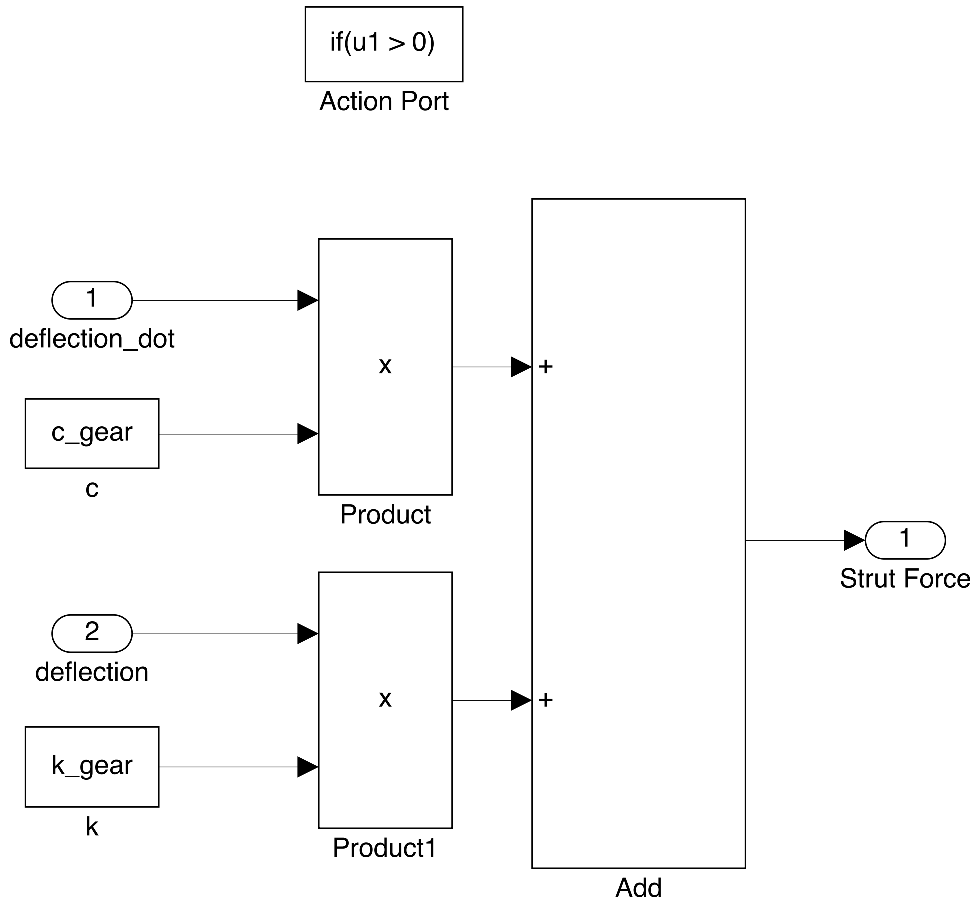


The landing gear deflection rate is $\dot{\delta}_t = -\dot{h}_{c.g.} - x_i\dot{\theta}\cos\theta + y_i\dot{\phi}\cos\phi$ where $[x_i, y_i, z_i]$ are the coordinates of the wheel with respect to the cg.

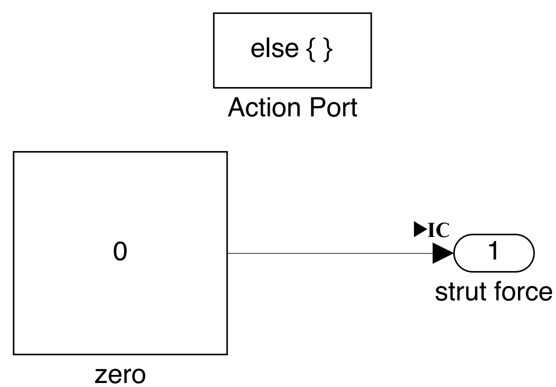




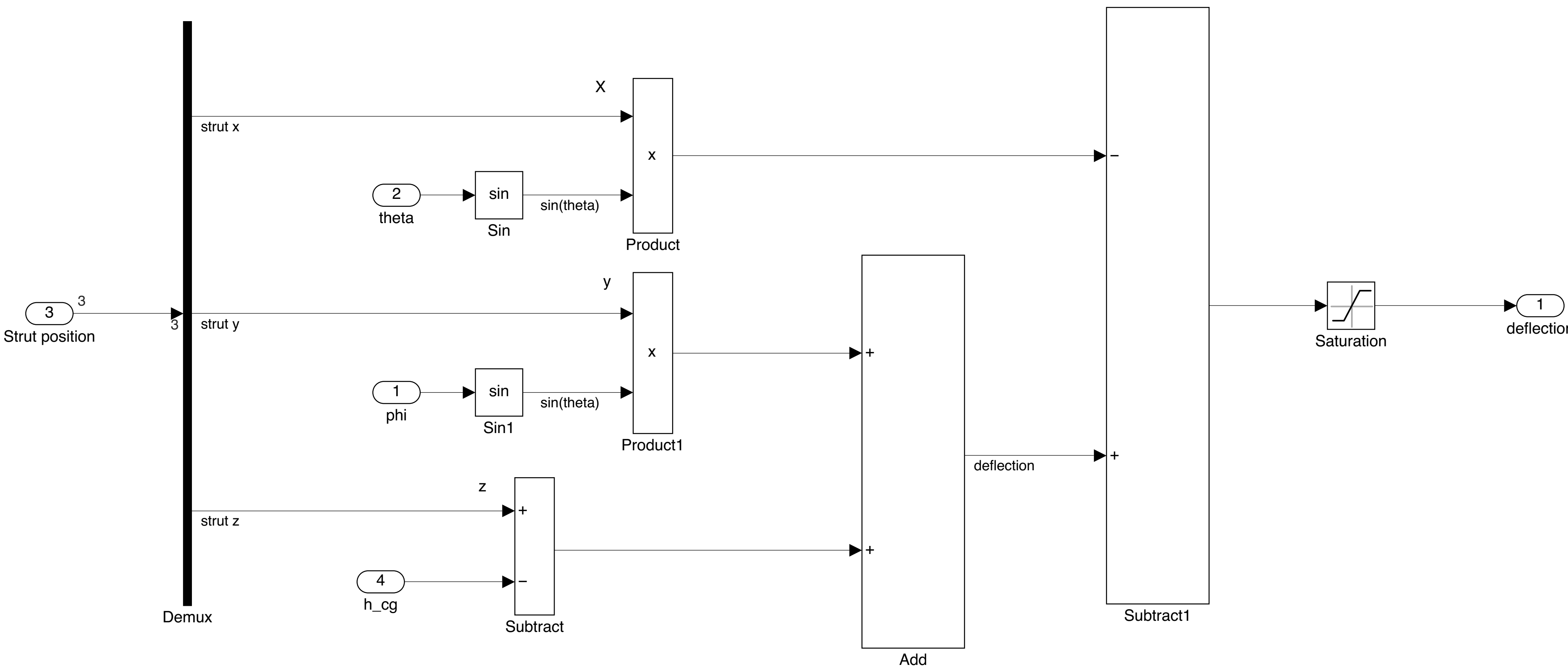
0.88
x
0
y
0.305
z
Mux
3
Strut X_body
1
h_dot
3
theta_dot
4
theta
5
phi_dot
6
phi
2
h_cg
h_dot
theta_dot
theta
phi_dot
phi
strut position
deflection_dot
strut 1 deflection rate
phi
theta
Strut position
h_cg
deflection
strut 1 deflection
If landing gear deflection > 0
deflection_dot
deflection
Strut Force
if(u1 > 0)
u1
if(u1 > 0)
else
If
else { }
Action Port
zero
strut force
if landing gear deflection = 0
merge
Merge
strut axial force
-1
Gain
0
Constant1
0
Constant2
Mux1
cross
Cross Product
2
strut moment
1
strut force



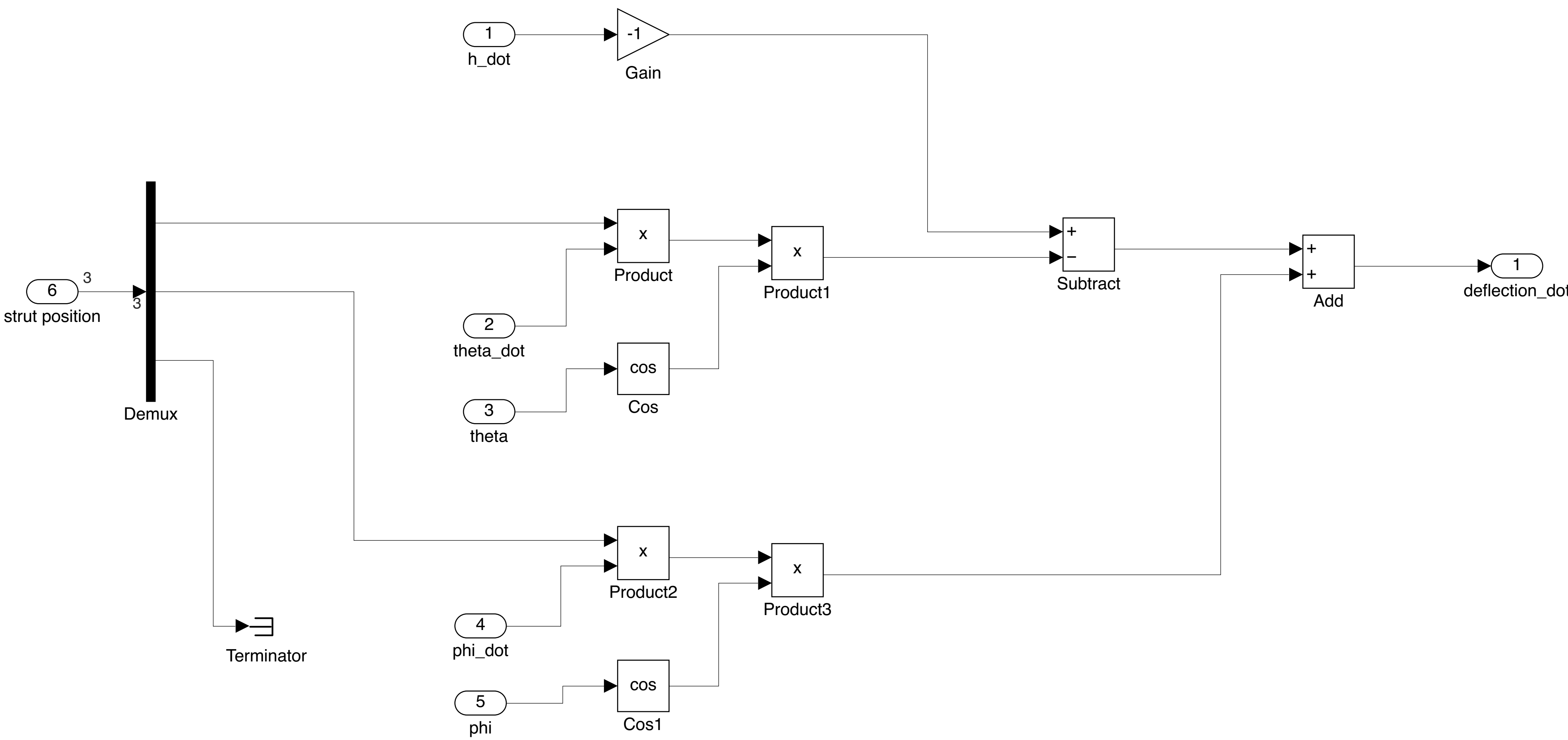
if(u1 > 0)
Action Port
1
deflection_dot
c_gear
c
x
Product
+
+
Add
1
Strut Force
2
deflection
k_gear
k
x
Product1

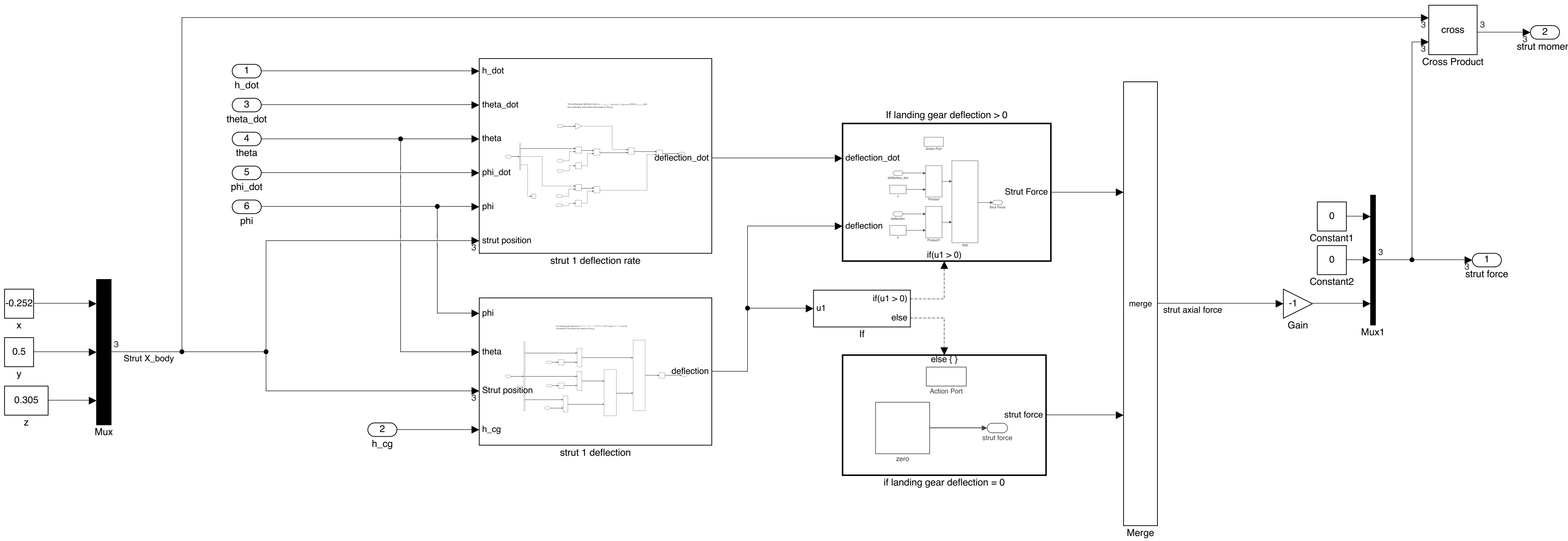

else { }
Action Port
0
zero
IC
1
strut force

The landing gear deflection is modelled using $\delta_t = z_i - h_{c.g.} - x_i \sin\theta + y_i \sin\phi$ where $[x_i, y_i, z_i]$ are the coordinates of the wheel with respect to the cg.

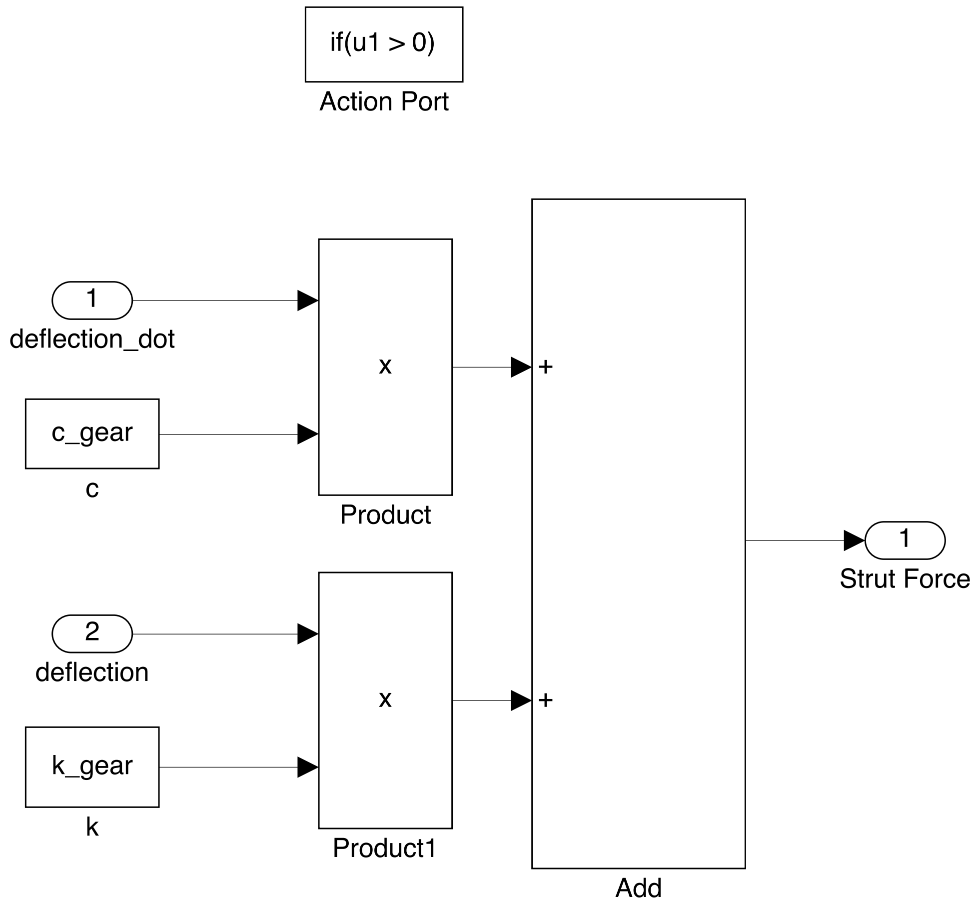


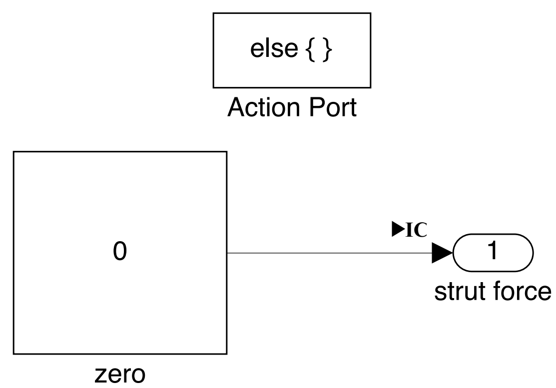
The landing gear deflection rate is modeled using $\dot{\delta}_t = -\dot{h}_{c.g.} - x_i\dot{\theta}\cos\theta + y_i\dot{\phi}\cos\phi$
where $[x_i, y_i, z_i]$ are the coordinates of the wheel with respect to the cg.



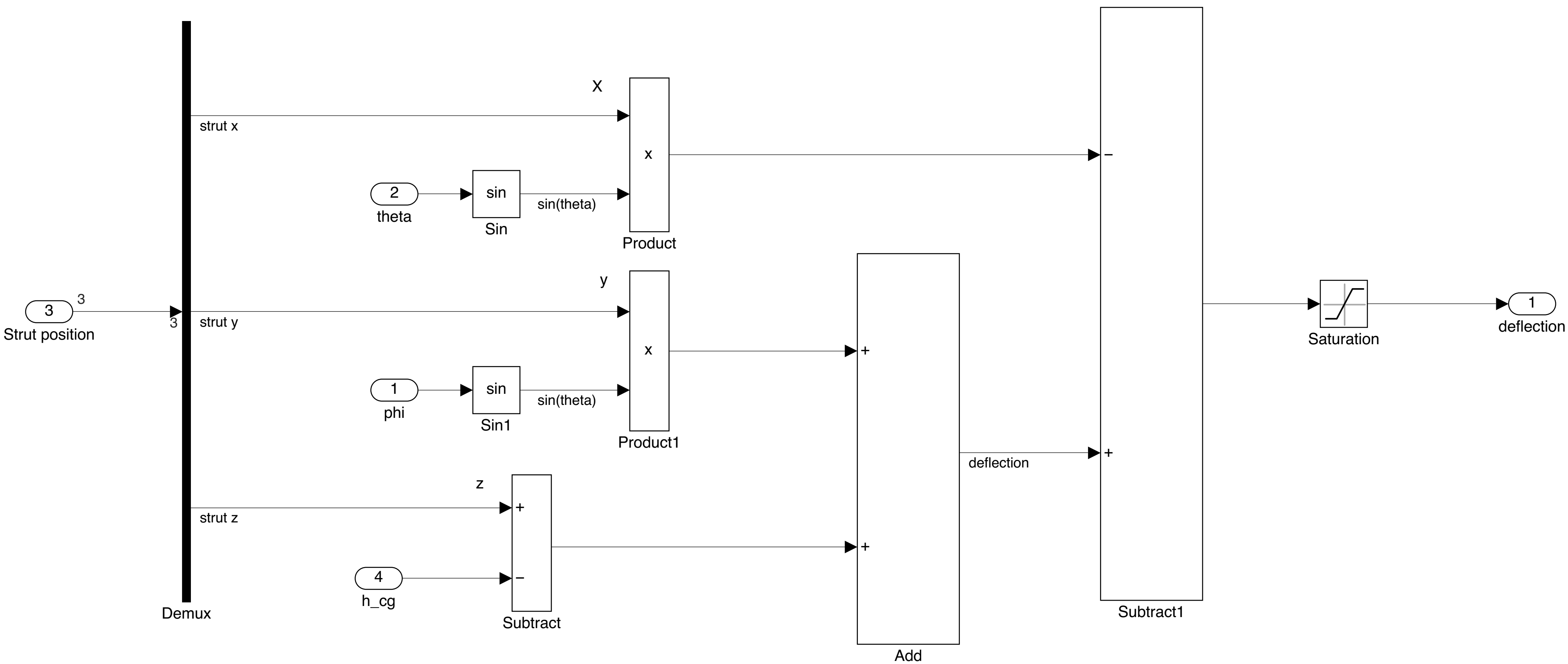


-0.252
x
0.5
y
0.305
z
Mux
3
Strut X_body
1
h_dot
3
theta_dot
4
theta
5
phi_dot
6
phi
2
h_cg
h_dot
theta_dot
theta
phi_dot
phi
strut position
deflection_dot
strut 1 deflection rate
phi
theta
Strut position
h_cg
deflection
strut 1 deflection
If landing gear deflection > 0
deflection_dot
deflection
Strut Force
if(u1 > 0)
u1
if(u1 > 0)
else
If
else { }
Action Port
zero
strut force
strut force
if landing gear deflection = 0
merge
Merge
strut axial force
-1
Gain
0
Constant1
0
Constant2
Mux1
cross
Cross Product
2
strut moment
1
strut force

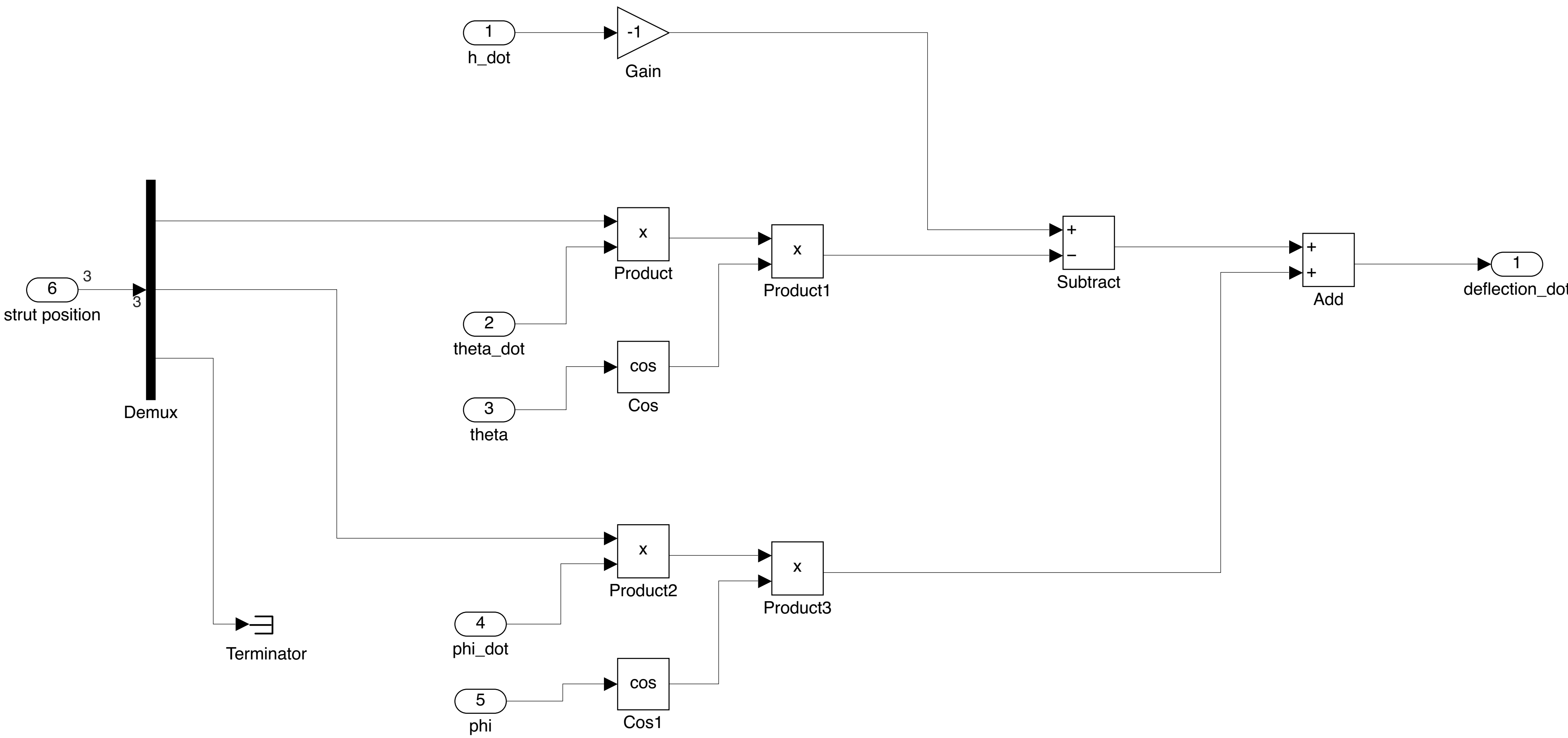

if(u1 > 0)
Action Port
1
deflection_dot
c_gear
c
x
Product
+
2
deflection
k_gear
k
x
Product1
+
Add
1
Strut Force

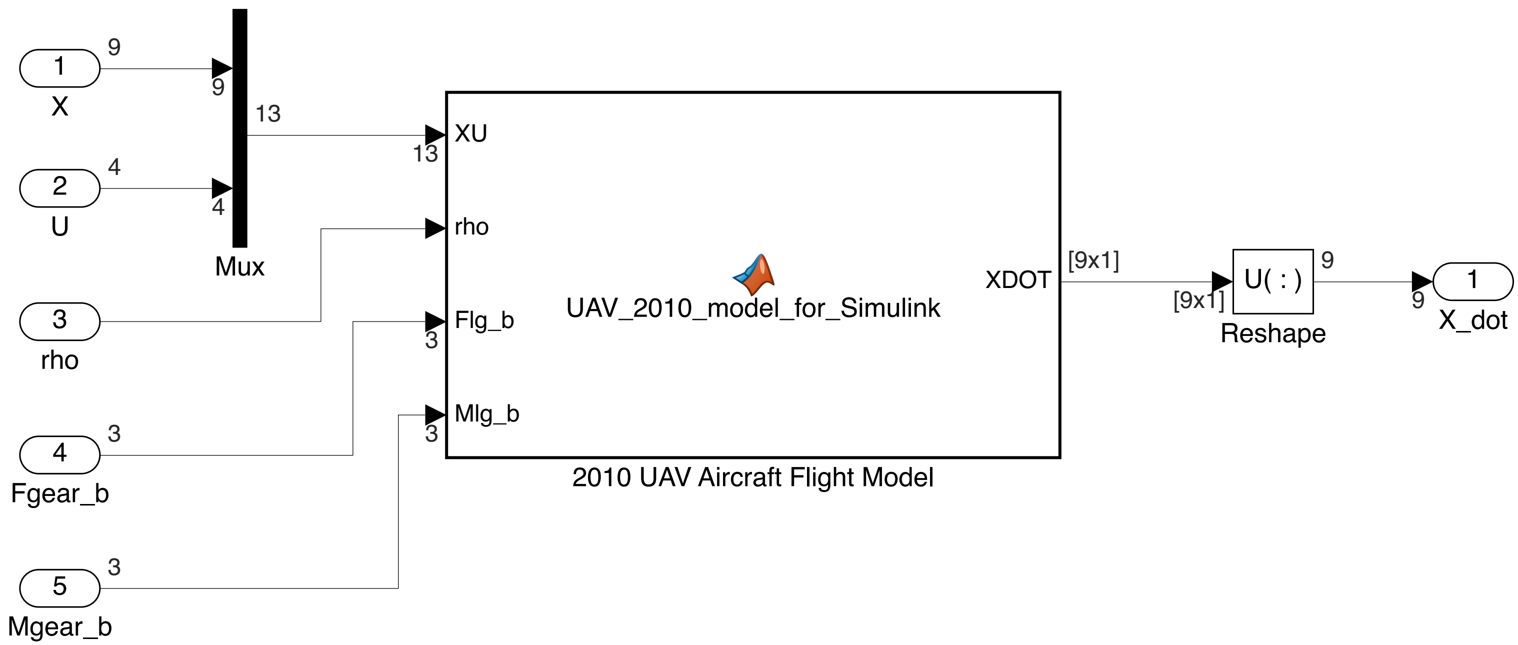

else { }
Action Port
0
zero
IC
1
strut force

The landing gear deflection is $\delta_t = z_i - h_{c.g.} - x_i \sin\theta + y_i \sin\phi$ where $[x_i, y_i, z_i]$ are the coordinates of the wheel with respect to the cg.



The landing gear deflection rate is $\dot{\delta}_t = -\dot{h}_{c.g.} - x_i\dot{\theta}\cos\theta + y_i\dot{\phi}\cos\phi$ where $[x_i, y_i, z_i]$ are the coordinates of the wheel with respect to the cg.



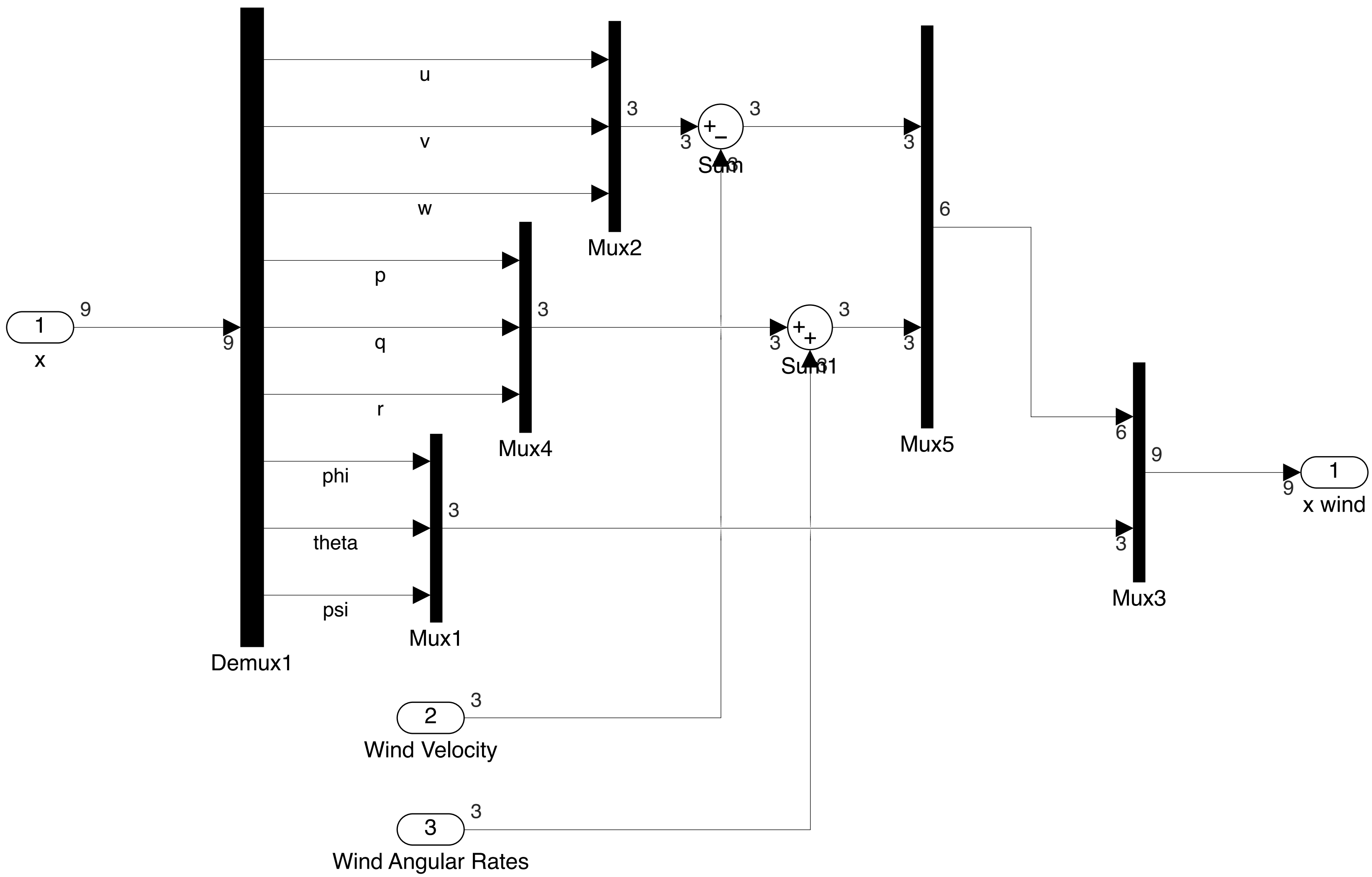

1
X
2
U
3
rho
4
Fgear_b
5
Mgear_b
9
9
4
4
3
3
Mux
13
13
XU
rho
Flg_b
3
Mlg_b
3
UAV_2010_model_for_Simulink
XDOT
2010 UAV Aircraft Flight Model
[9x1]
[9x1]
U(:)
Reshape
9
9
1
X_dot

```
function XDOT  = UAV_2010_model_for_Simulink(XU, rho, Flg_b, Mlg_b)

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

U = XU(10:13);

u1 = U(1); %d_A (aileron)
u2 = U(2); %d_T (stabilizer)
u3 = U(3); %d_R (rudder)
u4 = U(4); %d_th1 (throttle 1)

%----------------------CONSTANTS-------------------------------------------
% Nominal vehicle constants
m = 16.55612;            % Aircraft total mass (kg)

cbar = 0.7020814;        % Mean Aerodynamic Chord (m)
S = 1.309933;            % Wing planform area

Xcg = 0.2*cbar;          % x position of CoG in Fm (m)
Ycg = 0*cbar;            % y position of CoG in Fm (m)
Zcg = -0.10*cbar;        % z position of CoG in Fm (m)

% Engine constants
Xapt1 = -0.3556;         % x position of engine 1 force in Fm (m)
Yapt1 = 0;               % x position of engine 1 force in Fm (m)
Zapt1 = -0.0508;         % x position of engine 1 force in Fm (m)

% Other constants
g = 9.81;                % grav constant

%---------------------1. CONTROL LIMITS/SATURATION-------------------------


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

% Calculate alpha and beta
alpha = atan2(x3,x1)*180/pi;
beta = asin(x2/Va)*180/pi;

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

% Define vectors wbe_b and V_b
wbe_b = [x4;x5;x6];
V_b = [x1;x2;x3];

%---------------------3. AERODYNAMIC FORCE COEFFICIENTS--------------------
```

```
% Calculate the CL_wb

CL_noElev = -0.00002271*alpha^3 - 0.0002302*alpha^2 + 0.06565*alpha + 0.03959; % CL

CL_elevEffect = (-0.00001250958*alpha^3 + 0.000007639437*alpha^2 + 0.000286478898*alpha + 0.625574419317)*u2; % elevator effect on

% Total lift force coefficient
CL = CL_noElev + CL_elevEffect;

% Total drag force
CD = 0.000857*alpha^2 - 0.0004961*alpha + 0.0217;   % Curve fit of CD vs alpha

% % Total side force
if beta > 30.857                                    % piecewise parametrization of side force coefficient
    CY = 0.004409*beta -0.1694;
elseif beta <= 30.857 && beta >= -30.857
    CY = -0.001126*beta + 0.001396;
else
    CY = 0.004409*beta -0.1694;
end
%----------------------4. DIMENSIONAL AERODYNAMIC FORCES--------------------
% Calculate the actual dimensional forces in F_s (stability axis)
FA_s = [-CD*Q*S;
        CY*Q*S;
        -CL*Q*S];

rad = pi/180;

C_bs = [cos(alpha*rad) 0 -sin(alpha*rad);
        0 1 0;
        sin(alpha*rad) 0 cos(alpha*rad)];

FA_b = C_bs*FA_s;

%----------------------5. AERODYNAMIC MOMENT COEFFICIENT ABOUT CG --------------------
% Calculate CM = [Cl;Cm;Cn] about aerodynamic center in Fb

if beta > -17.409 && beta < 17.409                         % piecewise parameterization of CR
    eta11 = -0.0001058*(beta) + 0.0002933;
elseif beta <= -17.409
    eta11 = 0.0005473*beta + 0.01108;
else
    eta11 = 0.0005473*beta -0.01108;
end

eta21 = CMBA_lookup_table_2010_UAV(alpha, beta);           % CM lookup table
eta31 = CNBA_lookup_2010_UAV(beta);                        % CN lookup table

CMcg_b = [eta11 - 0.02864*u1 + dCRBA_dRudder(u3, alpha);       % all moments about CG in body axis including control surface effec
          eta21 + dCMSA_dElevator(u2, alpha, beta);
          eta31 + dCNBA_dRudder(alpha, u3)];

%----------------------6. AERODYNAMIC MOMENT ABOUT CG --------------------
% Normalize to an aerodynamic moment
MAcg_b = CMcg_b*Q*S*cbar;

% %----------------------7. AERODYNAMIC  COEFFICIENT ABOUT CG --------------------------
% rcg_b = [Xcg;Ycg;Zcg];
% rac_b = [Xac;Yac;Zac];
```

```
% MAcg_b = MAac_b + cross(FA_b,rcg_b - rac_b);

%---------------------8. ENGINE FORCE AND MOMENT ------------------------------------
% effect of engine. First, calculate thrust of engine
% engine_coeffs = [1.012e-09, -5.527e-05, 1.044];
%
% RPM = 158000*u4;
% if u4 > 1
%     RPM = 158000;
% end
% if RPM >= 0
%     F1 = (engine_coeffs(1)*RPM^2 + engine_coeffs(2)*RPM + engine_coeffs(3))*4.44822;
% else
%     F1 = 0;
% end

F1 = u4*m*g;

% assuming engine thrust is aligned with Fb, we have
FE1_b = [F1;0;0];
FE_b = FE1_b;

% engine moment due to offset of engine thrust from CoG
mew1 = [Xcg - Xapt1;
        Yapt1 - Ycg;
        Zcg - Zapt1];

MEcg1_b = cross(mew1,FE1_b);

MEcg_b = MEcg1_b;

%---------------------8.5. LANDING GEAR FORCE AND MOMENT ----------------------


%---------------------9. 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;

%---------------------10. GRAVITY EFFECTS ------------------------------------
% Inertia matrix
Ib = [1.1619 0 -0.0607;
        0 5.6276 0;
        -0.0607 0 6.1040];

% Inverse of inertia matrix
% symmetric about xz plane
invIb =    [0.861106618696125 0                  0.00856310153257778
            0                0.177695642902836 0
            0.00856310153257778  0                  0.163912152729854];

% Form R_b (all forces in Fb) and calculate udot, vdot, wdot
F_b = Fg_b + FE_b + FA_b + Flg_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 + Mlg_b;
x4to6dot = invIb*(Mcg_b - cross(wbe_b,Ib*wbe_b));

% Calculate phidot, thetadot, 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;
        x4to6dot;
        x7to9dot];

%     function tan = atan2_0_pi(y,x)
%         if x == 0
%             if y == 0
%                 tan = 0;
%             elseif y > 0
%                 tan = pi/2;
%             else
%                 tan = 3*pi/3;
%             end
%         elseif x > 0
%             if y >= 0
%                 tan = atan(y/x);
%             else
%                 tan = atan(y/x) + 2*pi;
%             end
%         elseif x < 0
%             if y == 0
%                 tan = pi;
%             else
%                 tan = atan(y/x) + pi;
%             end
%         end
%     end

end
```



1
x
9
9
Demux1
u
v
w
p
q
r
phi
theta
psi
Mux2
Mux4
Mux1
3
Sum
Sum1
Mux5
6
Mux3
9
1
x wind
2
Wind Velocity
3
Wind Angular Rates