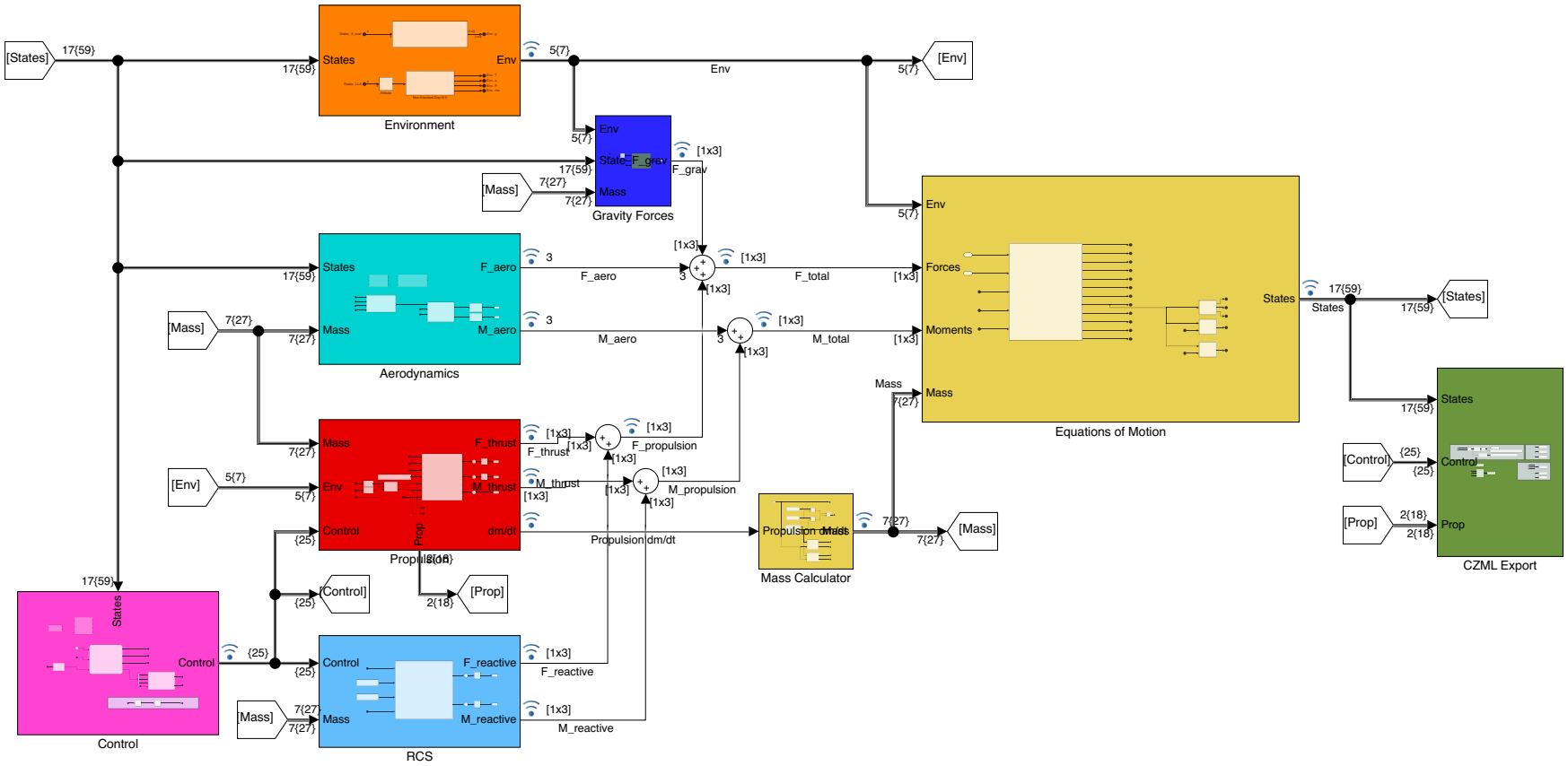
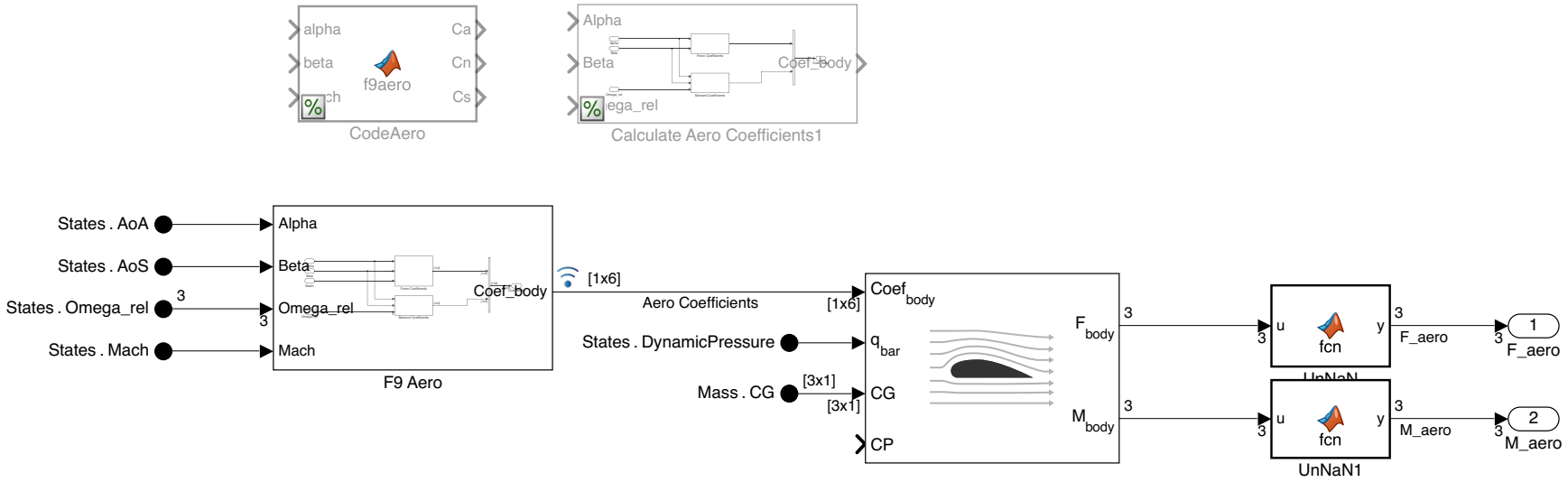
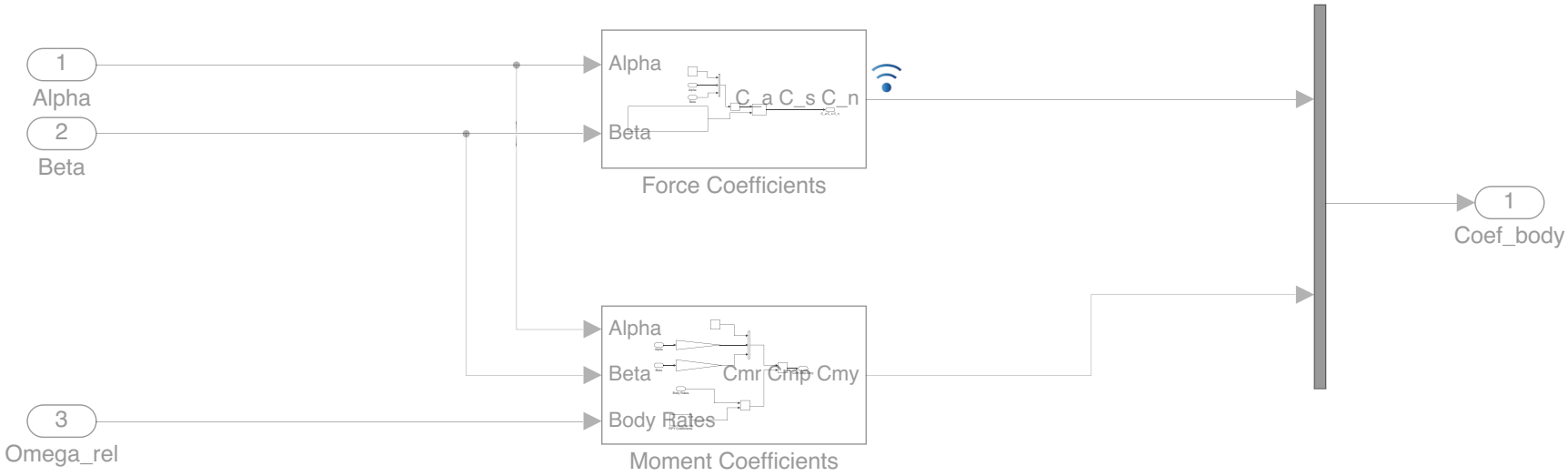
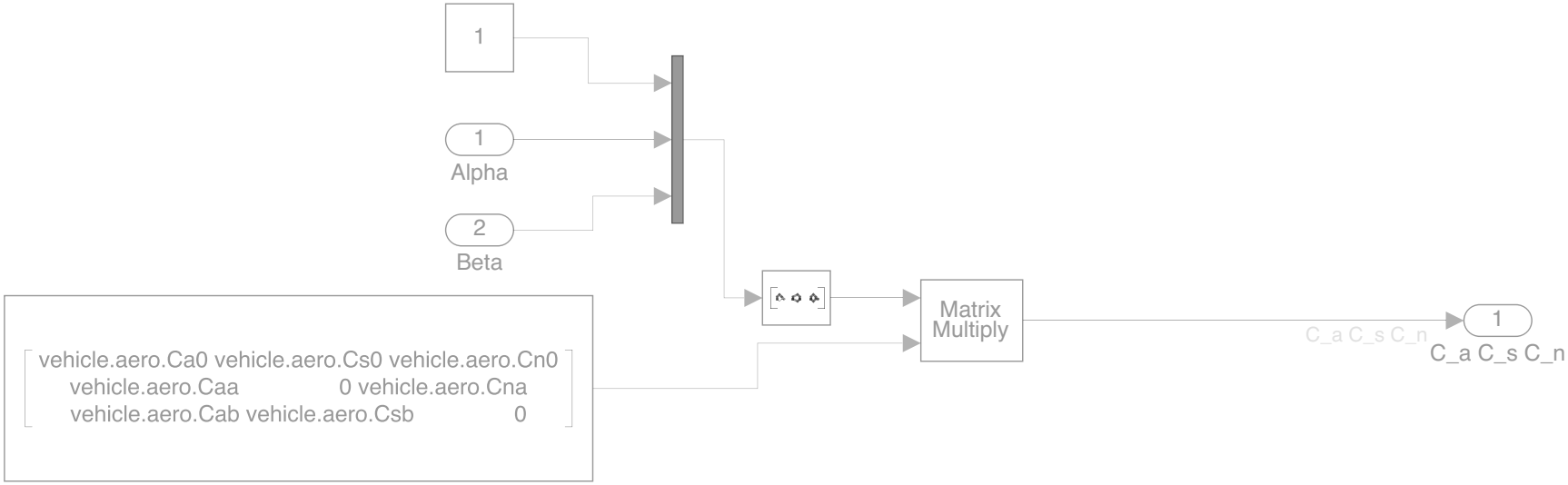
eom_6dof_f9_prototype_bus

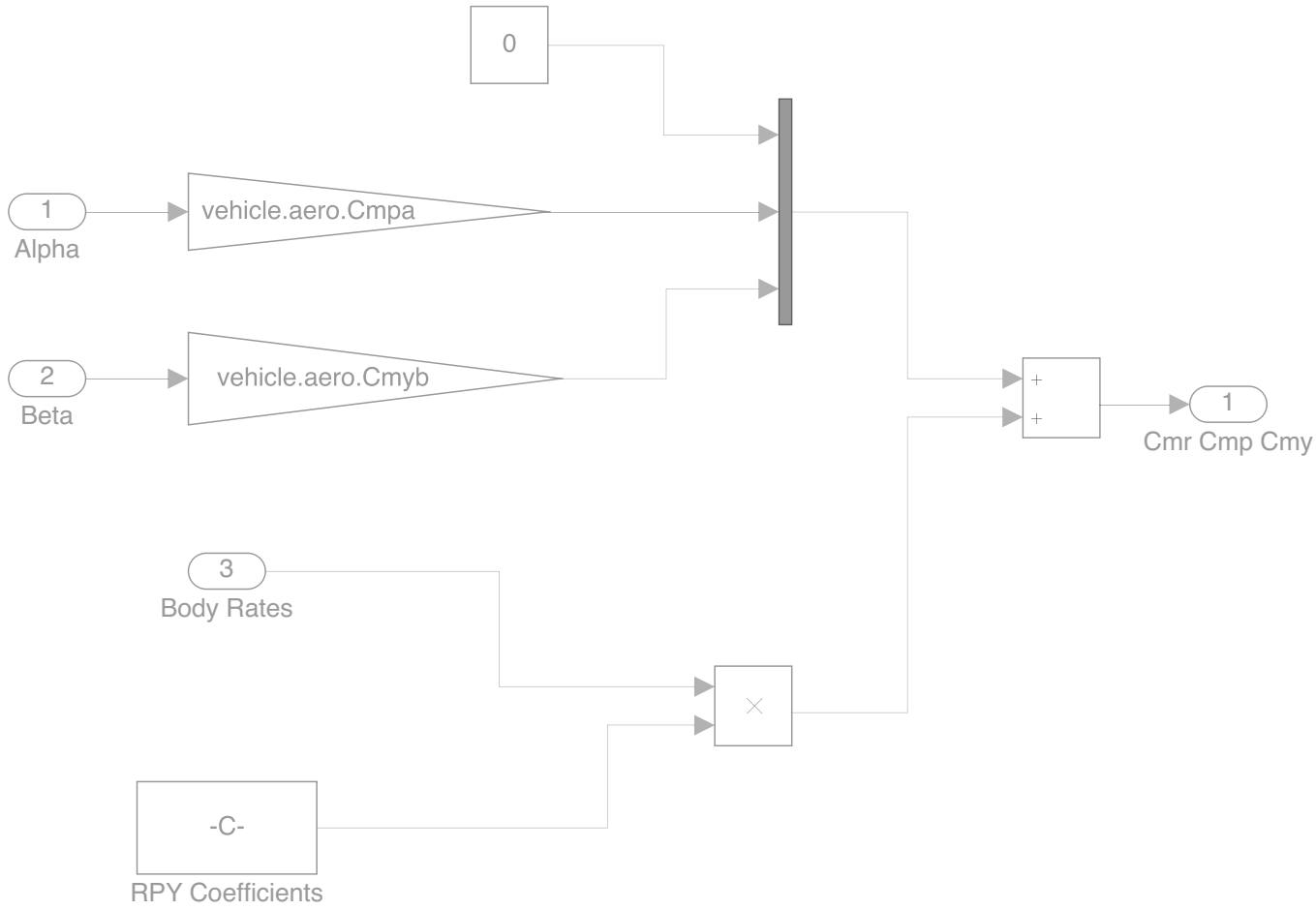


/Users/smiley/Development/regnc/regnc-playground/eom_6dof_f9_prototype_bus.slx

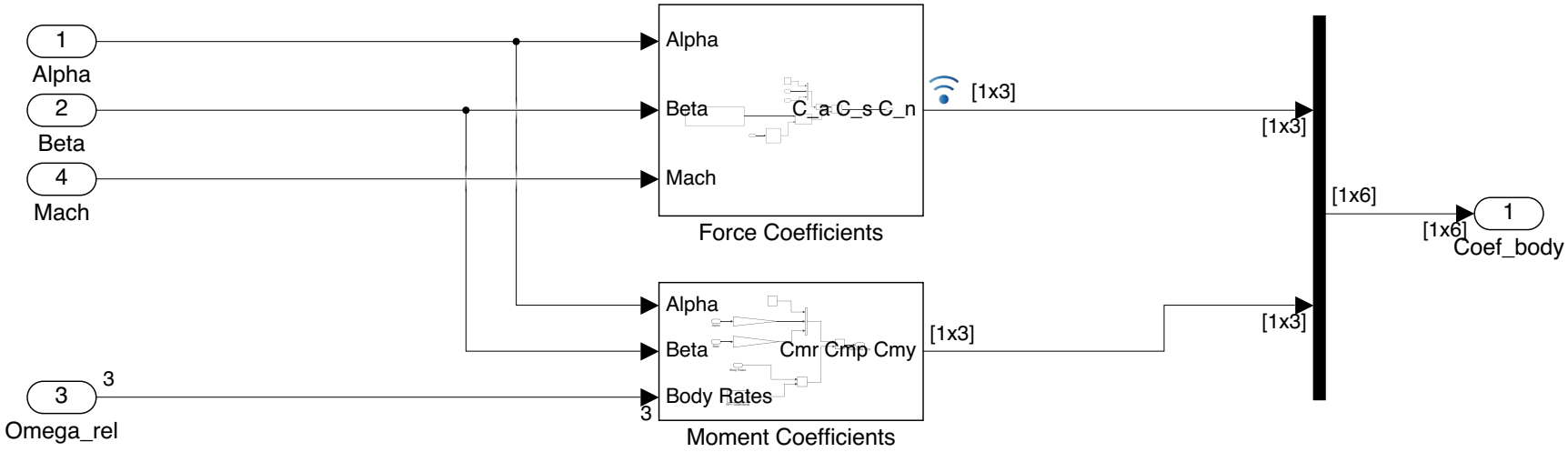

alpha
beta
ch
f9aero
Ca
Cn
Cs
CodeAero
Alpha
Beta
ega_rel
Coef_body
Calculate Aero Coefficients1
States . AoA
States . AoS
States . Omega_rel
States . Mach
Alpha
Beta
Omega_rel
Mach
Coef_body
F9 Aero
[1x6]
Aero Coefficients
[1x6]
States . DynamicPressure
Mass . CG
[3x1]
[3x1]
Coef body
q bar
CG
CP
F body
M body
u
y
fcn
F_aero
M_aero
UnNaN1
1
2

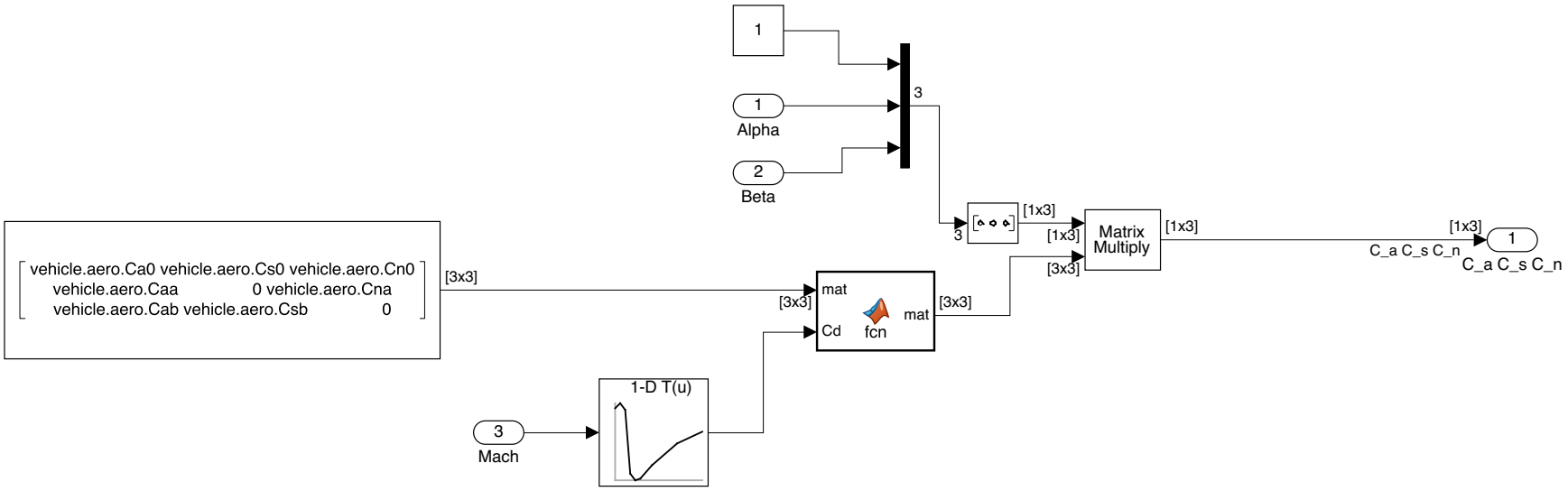

1
Alpha
2
Beta
3
Omega_rel
Alpha
Beta
C_a C_s C_n
Force Coefficients
Alpha
Beta
Body Rates
Cmr Cmp Cmy
Moment Coefficients
1
Coef_body


1
1
Alpha
2
Beta
vehicle.aero.Ca0 vehicle.aero.Cs0 vehicle.aero.Cn0
vehicle.aero.Caa 0 vehicle.aero.Cna
vehicle.aero.Cab vehicle.aero.Csb 0
Matrix Multiply
C_a C_s C_n
1
C_a C_s C_n

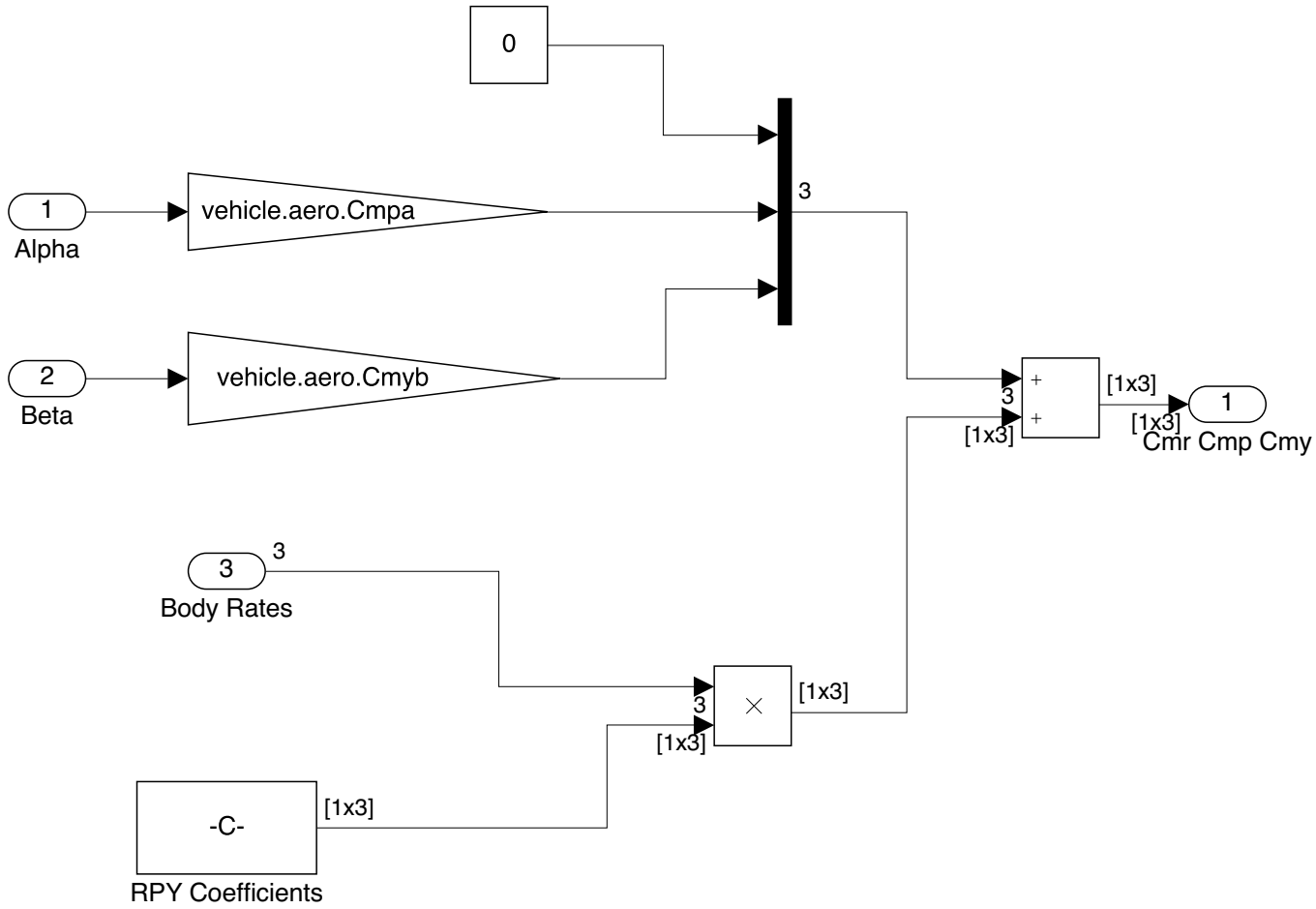

0
1
Alpha
vehicle.aero.Cmpa
2
Beta
vehicle.aero.Cmyb
1
Cmr Cmp Cmy
3
Body Rates
-C-
RPY Coefficients

```
function [Ca, Cn, Cs] = f9aero(alpha, beta, mach)
    Ca = getDragCoefficient(alpha, mach);
    Cn = getLiftCoefficient(alpha, mach);
    Cs = getLiftCoefficient(beta, mach);
end

function Cd = getBaseCd(baseCd, mach)
    Cd = baseCd;
    if mach > 1.0
        Cd = baseCd * 1.4 * exp(0.3 / mach);
    end
end

function Cd = getDragCoefficient(alpha, mach)
    baseCd = getBaseCd(0.4, mach);
    isRetrograde = false;

    if alpha > pi/2 || alpha < -pi/2
        % TODO landing legs and grid fins
        baseCd = getBaseCd(0.6, mach);
        isRetrograde = true;
    end

    dragCoefficient = abs(baseCd * cos(alpha));
    dragPreservation = 1.0;

    if isRetrograde
        % todo stuff
    end

    Cd = abs(dragCoefficient);
end

function Cl = getLiftCoefficient(alpha, mach)
    baseCl = getBaseCd(0.6, mach);
    Cl = baseCl * sin(alpha*2);
end
```


1
Alpha
2
Beta
4
Mach
3
Omega_rel
Alpha
Beta
Mach
C_a C_s C_n
Force Coefficients
[1x3]
Alpha
Beta
Body Rates
Cmr Cmp Cmy
Moment Coefficients
[1x3]
[1x3]
[1x3]
[1x6]
[1x6]
1
Coef_body

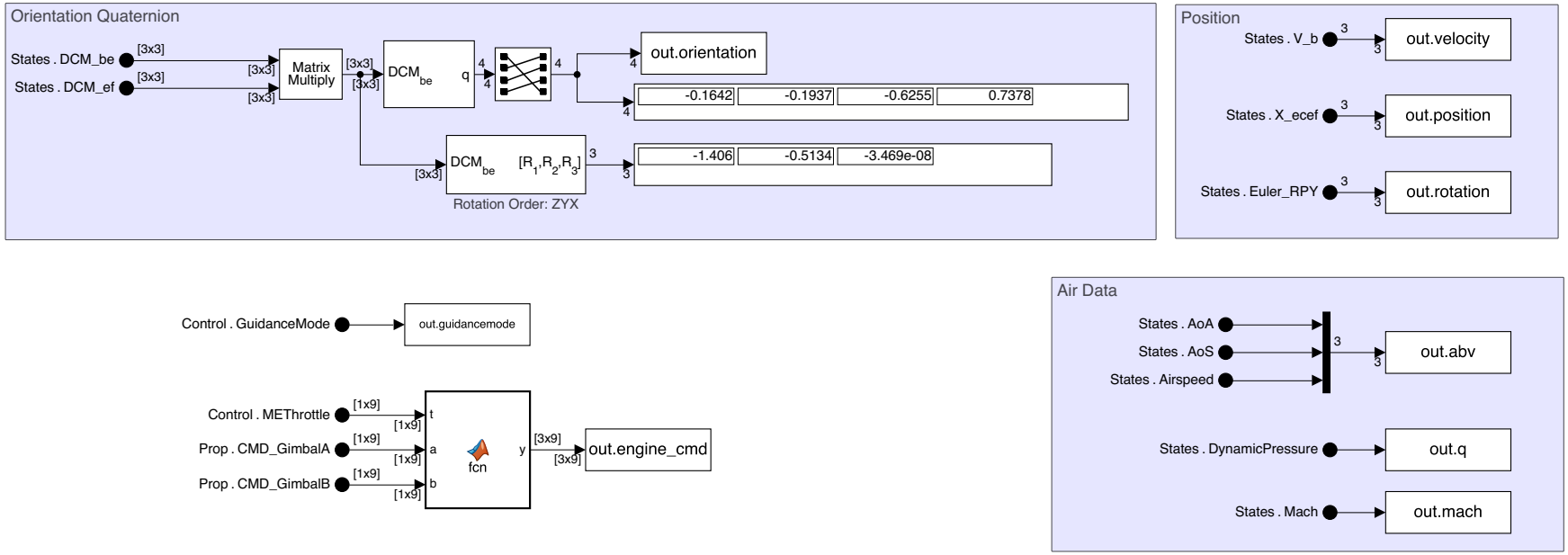

1
1
Alpha
2
Beta
3
[1x3]
3
[1x3]
[3x3]
Matrix
Multiply
[1x3]
[1x3]
1
C_a C_s C_n
C_a C_s C_n
vehicle.aero.Ca0 vehicle.aero.Cs0 vehicle.aero.Cn0
vehicle.aero.Caa 0 vehicle.aero.Cna
vehicle.aero.Cab vehicle.aero.Csb 0
[3x3]
mat
[3x3]
Cd
fcn
mat
[3x3]
1-D T(u)
3
Mach

```
function mat = fcn(mat, Cd)
    mat(1,1) = Cd;
end
```

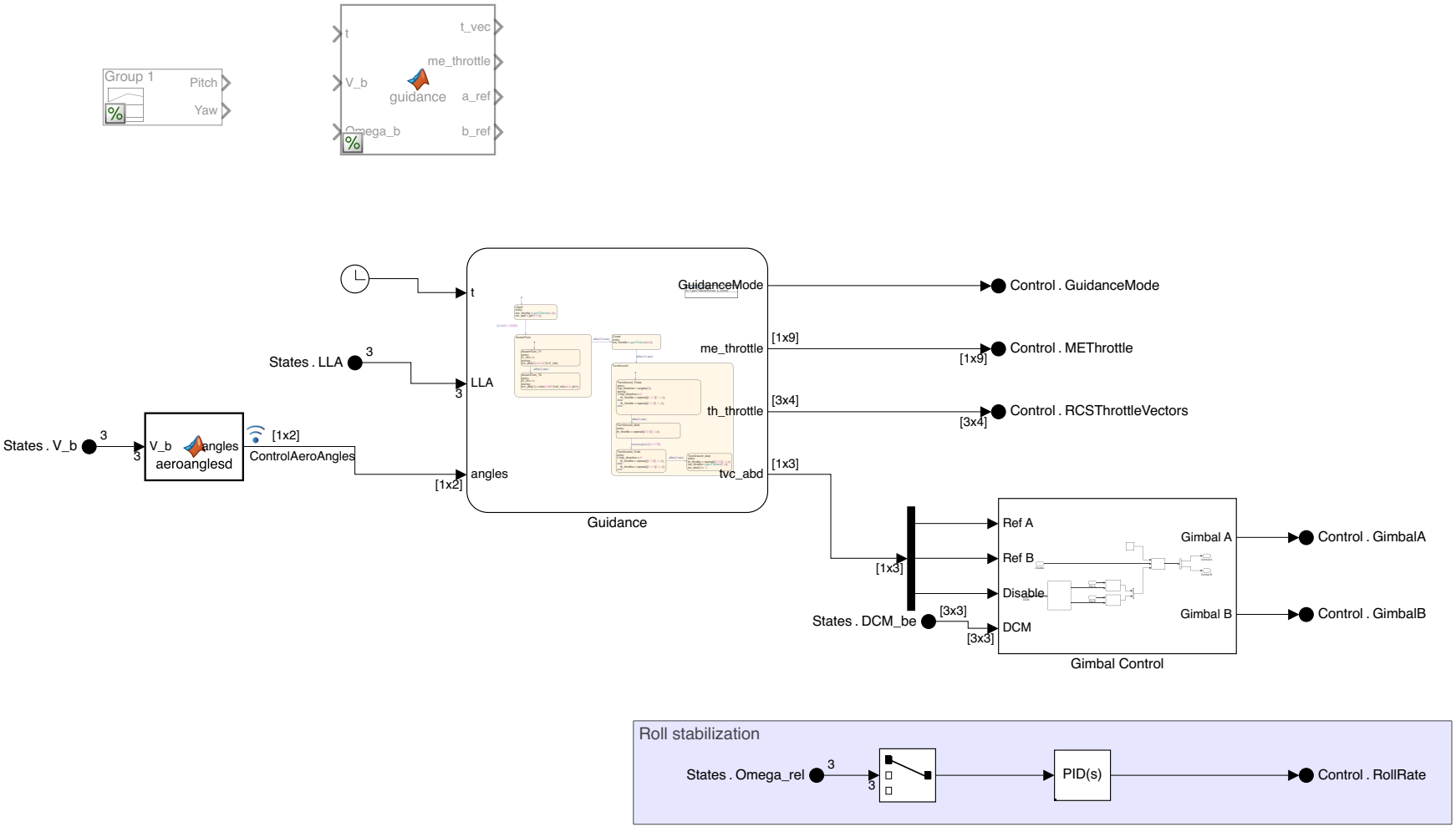

0
1
Alpha
vehicle.aero.Cmpa
3
2
Beta
vehicle.aero.Cmyb
3
[1x3]
[1x3]
1
[1x3]
Cmr Cmp Cmy
3
3
Body Rates
3
[1x3]
[1x3]
-C-
[1x3]
RPY Coefficients

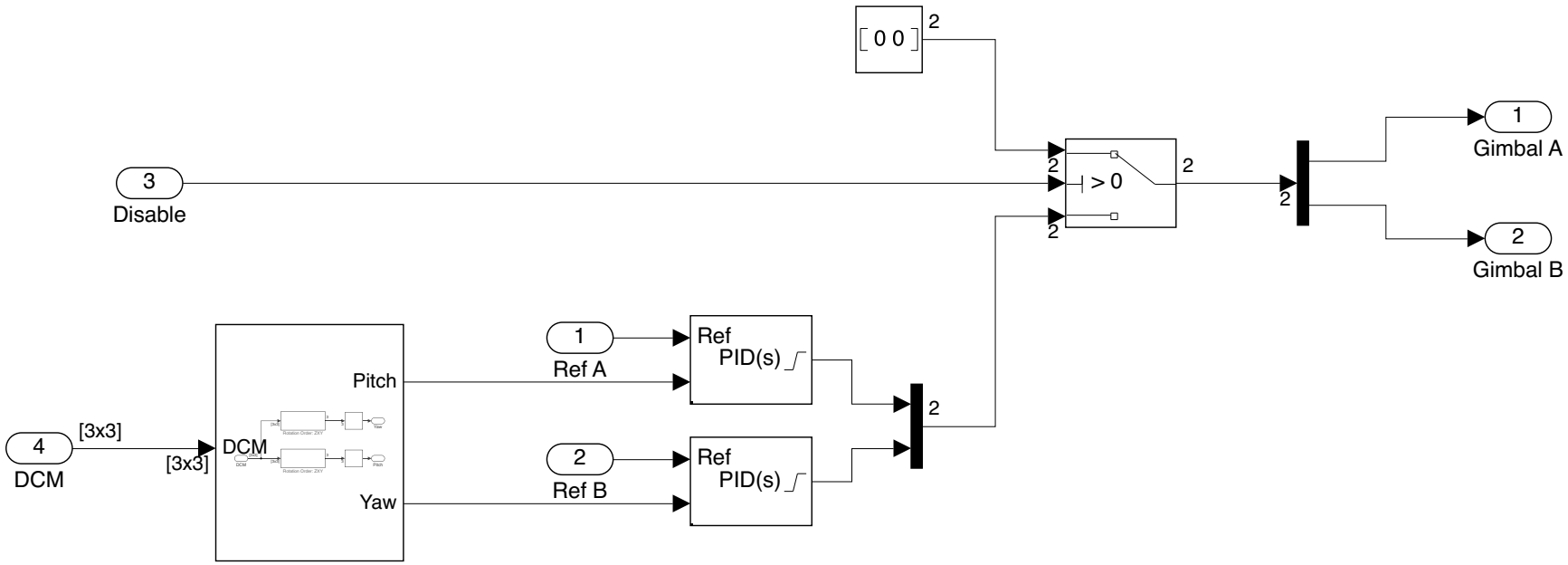
```
function y = fcn(u)
    u(isnan(u)) = 0;
y = u;
```

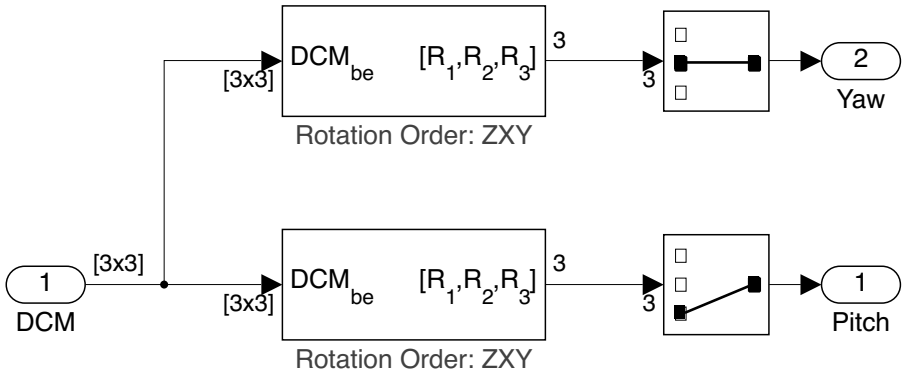
```
function y = fcn(u)
    u(isnan(u)) = 0;
y = u;
```

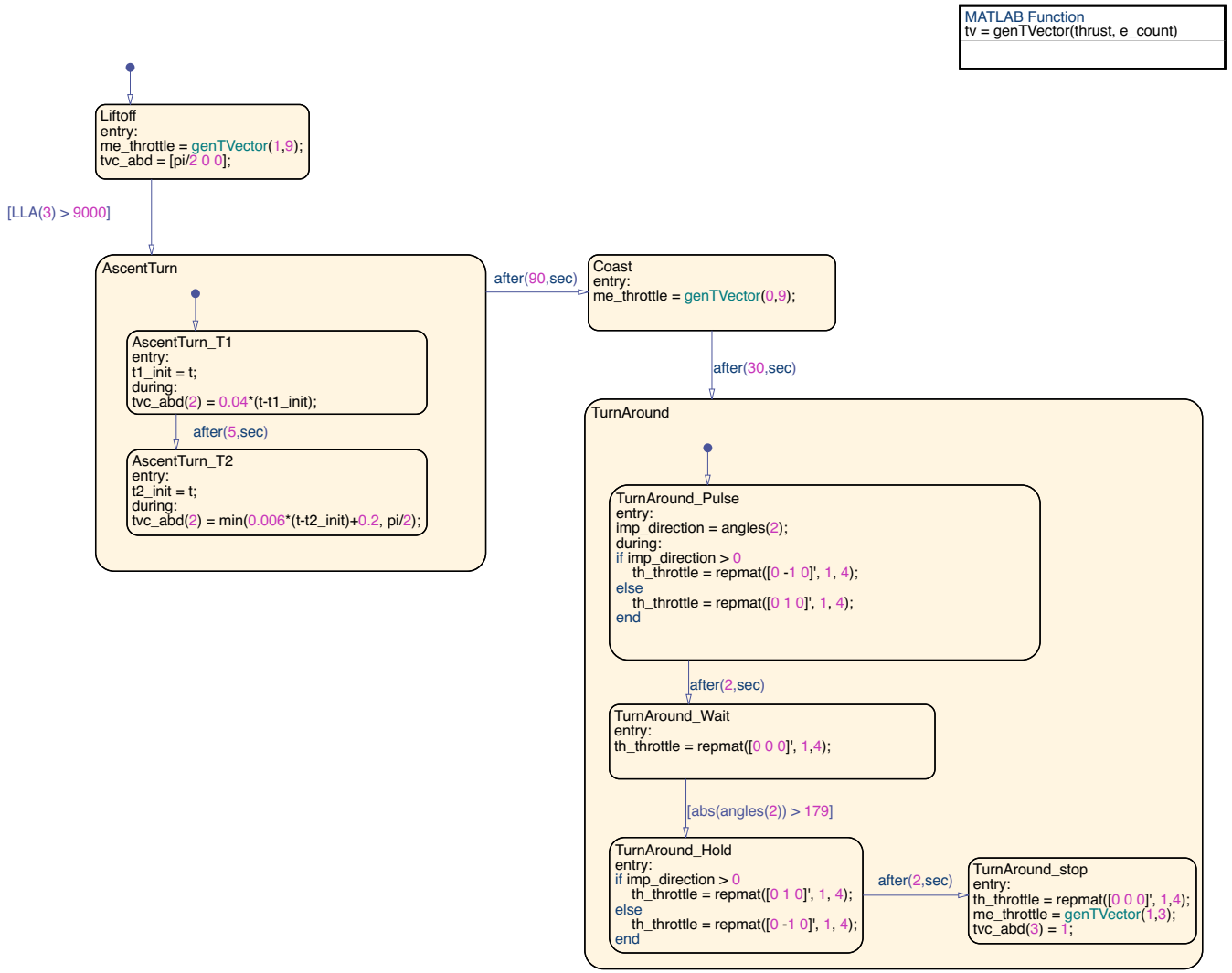

Orientation Quaternion
States . DCM_be
States . DCM_ef
[3x3]
Matrix Multiply
DCM_be
q
out.orientation
-0.1642
-0.1937
-0.6255
0.7378
DCM_be
[R1,R2,R3]
-1.406
-0.5134
-3.469e-08
Rotation Order: ZYX
Position
States . V_b
States . X_ecef
States . Euler_RPY
out.velocity
out.position
out.rotation
Control . GuidanceMode
out.guidancemode
Control . METhrottle
Prop . CMD_GimbalA
Prop . CMD_GimbalB
[1x9]
t
a
b
fcn
y
[3x9]
out.engine_cmd
Air Data
States . AoA
States . AoS
States . Airspeed
out.abv
States . DynamicPressure
out.q
States . Mach
out.mach

```
function y = fcn(t, a, b)
    y = [t;a;b]
end
```


Group 1
Pitch
Yaw
t
V_b
Omega_b
t_vec
me_throttle
guidance
a_ref
b_ref
States . LLA
3
t
LLA
GuidanceMode
me_throttle
th_throttle
tvc_abd
angles
[1x2]
Guidance
Control . GuidanceMode
[1x9]
Control . METhrottle
[3x4]
Control . RCSThrottleVectors
States . V_b
V_b
angles
aeroanglesd
[1x2]
ControlAeroAngles
[1x3]
Ref A
Ref B
Disable
DCM
Gimbal A
Gimbal B
States . DCM_be
[3x3]
Gimbal Control
Control . GimbalA
Control . GimbalB
Roll stabilization
States . Omega_rel
PID(s)
Control . RollRate

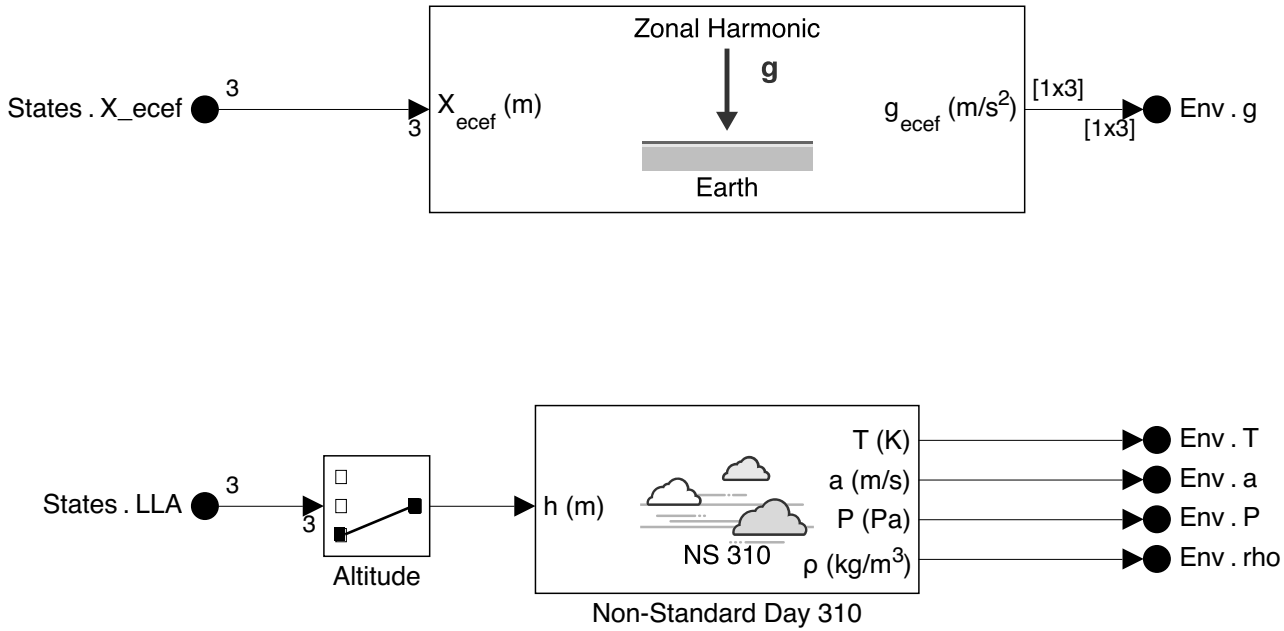

[0 0]
2
3
Disable
> 0
1
Gimbal A
2
Gimbal B
4
DCM
[3x3]
[3x3]
DCM
Pitch
Yaw
1
Ref A
2
Ref B
Ref
PID(s)
Ref
PID(s)

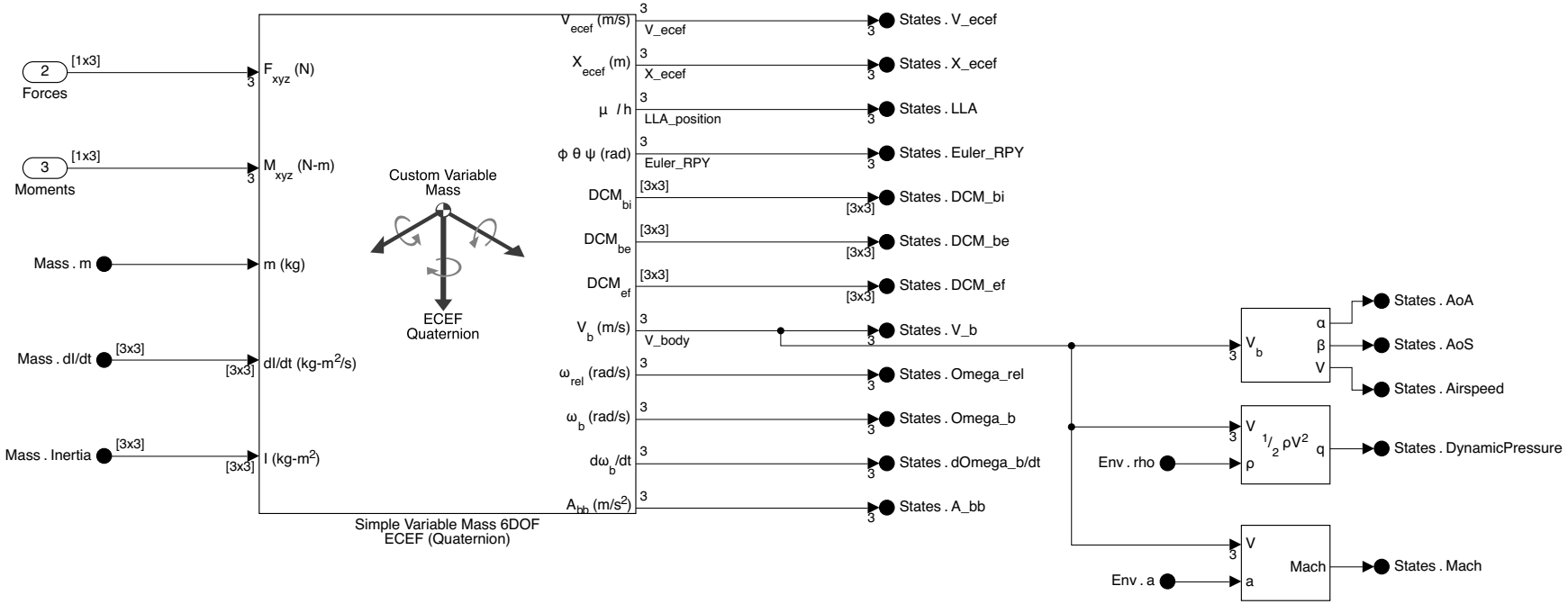

1
DCM
[3x3]
[3x3]
DCM be
[R1,R2,R3]
3
Rotation Order: ZXY
3
2
Yaw
[3x3]
[3x3]
DCM be
[R1,R2,R3]
3
Rotation Order: ZXY
3
1
Pitch

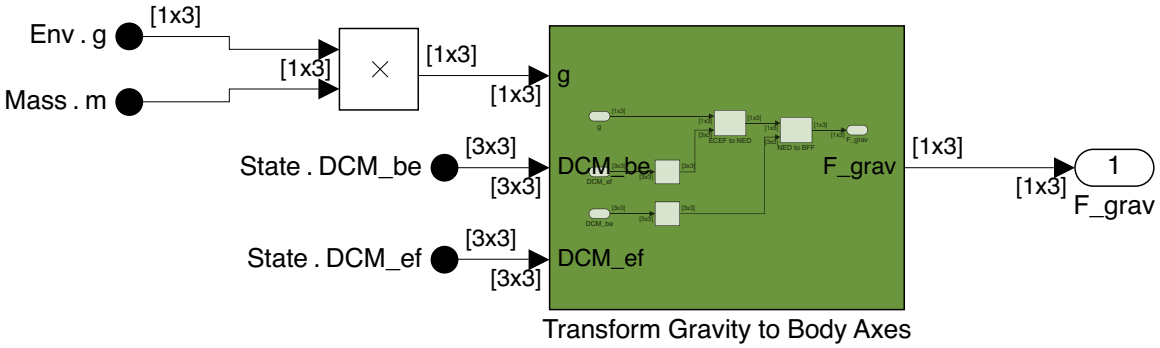

MATLAB Function
tv = genTVector(thrust, e_count)
Liftoff
entry:
me_throttle = genTVector(1,9);
tvc_abd = [pi/2 0 0];
[LLA(3) > 9000]
AscentTurn
AscentTurn_T1
entry:
t1_init = t;
during:
tvc_abd(2) = 0.04*(t-t1_init);
after(5,sec)
AscentTurn_T2
entry:
t2_init = t;
during:
tvc_abd(2) = min(0.006*(t-t2_init)+0.2, pi/2);
after(90,sec)
Coast
entry:
me_throttle = genTVector(0,9);
after(30,sec)
TurnAround
TurnAround_Pulse
entry:
imp_direction = angles(2);
during:
if imp_direction > 0
th_throttle = repmat([0 -1 0]', 1, 4);
else
th_throttle = repmat([0 1 0]', 1, 4);
end
after(2,sec)
TurnAround_Wait
entry:
th_throttle = repmat([0 0 0]', 1,4);
[abs(angles(2)) > 179]
TurnAround_Hold
entry:
if imp_direction > 0
th_throttle = repmat([0 1 0]', 1, 4);
else
th_throttle = repmat([0 -1 0]', 1, 4);
end
after(2,sec)
TurnAround_stop
entry:
th_throttle = repmat([0 0 0]', 1,4);
me_throttle = genTVector(1,3);
tvc_abd(3) = 1;

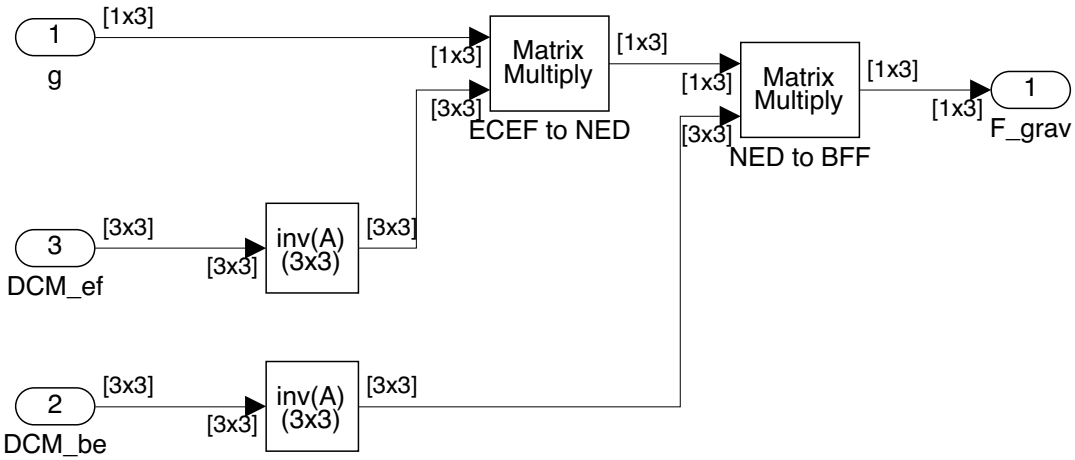
```
function [t_vec, me_throttle, a_ref, b_ref] = guidance(t, V_b, Omega_b)
    a_ref = pi/2;
    b_ref = 0;
    t_vec = repmat([1e-16 1e-16 1e-16]', 1, 4);

    if t < 160
        % Launch Guidance
        b_ref = 0;

        if t >= 45 && t < 50
            b_ref = 0.04*(t-45);
        elseif t >= 50
            b_ref = min(0.006*(t-50)+0.2, pi/2);
        end
    elseif t >= 160
        t_vec = turnaround(t);
    end

    % Throttle
    me_throttle = repmat(1, 1, 9);
    if t >= 40 && t < 55
        me_throttle = repmat(0.7, 1, 9);
    elseif t >= 160
        me_throttle = repmat(0,1,9);
    end
end

function t_vec = turnaround(t)

end
```

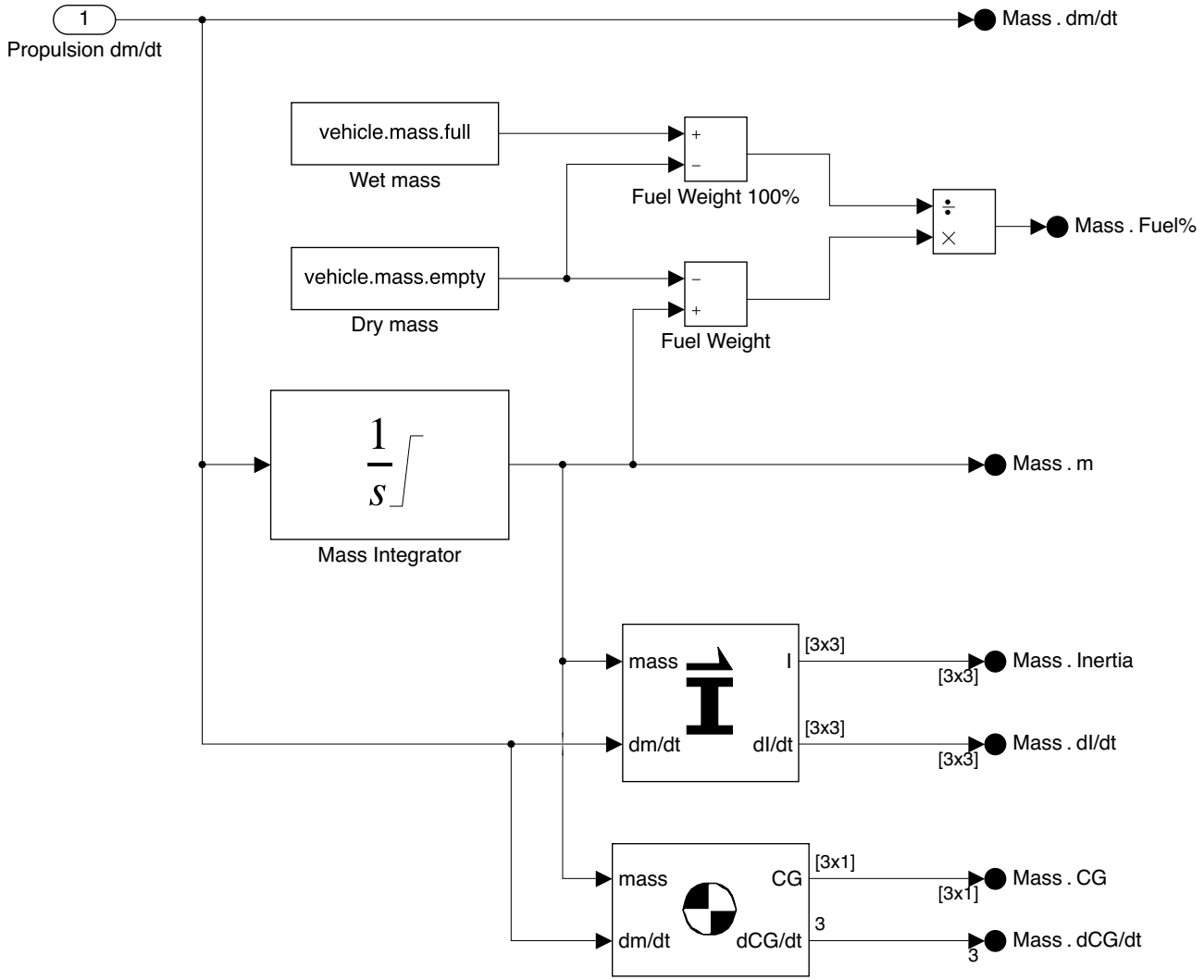
```
function angles = aeroanglesd(V_b)
    angles = zeros(1,2);
    bx = [1 0 0];

    x_vel = [V_b(1) V_b(2) 0];
    angles(2) = vecangle360(bx, x_vel, [0 0 1]);

    y_vel = [V_b(1) 0 V_b(3)];
    angles(1) = vecangle360(bx, y_vel, [0 1 0]);
end

% Vector angles
function a = vecangle360(v1,v2,n)
    x = cross(v1,v2);
    c = sign(dot(x,n)) * norm(x);
    a = atan2d(c,dot(v1,v2));
end
```

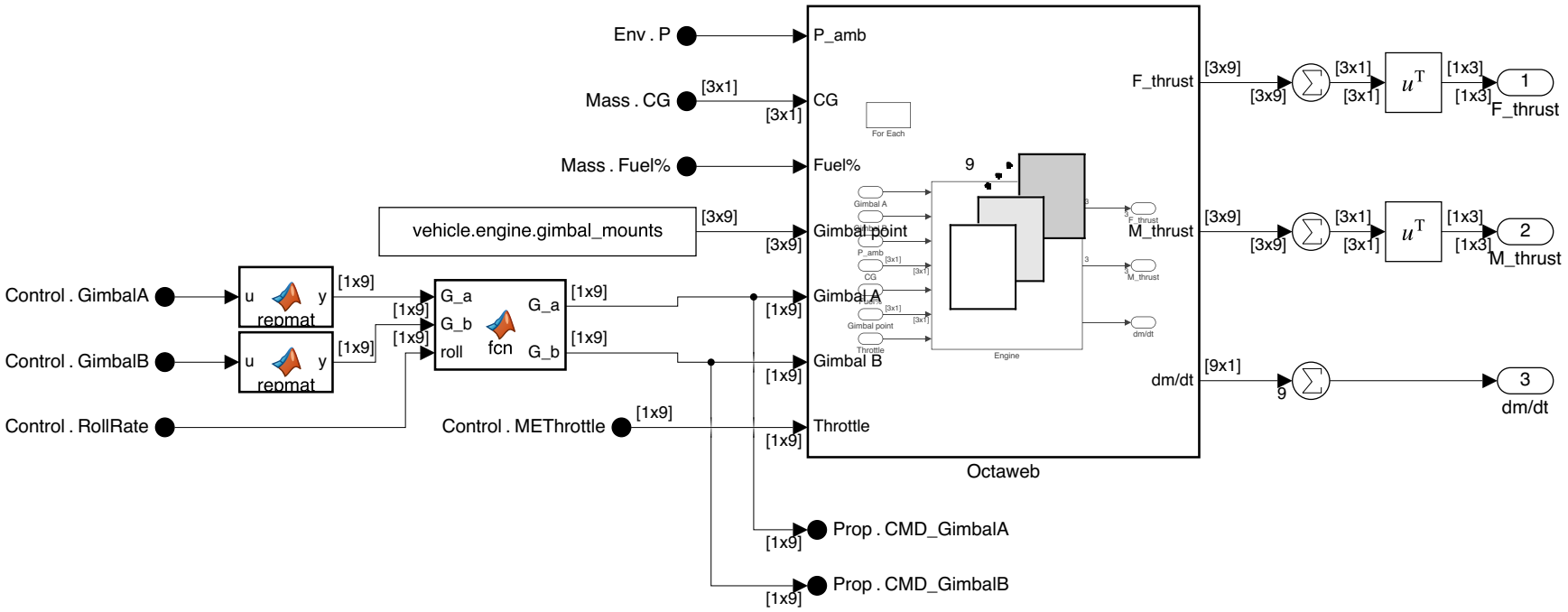

Zonal Harmonic
g
States . X_ecef
3
3
X ecef (m)
Earth
g ecef (m/s2)
[1x3]
[1x3]
Env . g
States . LLA
3
3
Altitude
h (m)
NS 310
T (K)
a (m/s)
P (Pa)
ρ (kg/m3)
Env . T
Env . a
Env . P
Env . rho
Non-Standard Day 310


2
Forces
[1x3]
3
F_xyz (N)
3
Moments
M_xyz (N-m)
Mass . m
m (kg)
Mass . dI/dt
dI/dt (kg-m²/s)
Mass . Inertia
I (kg-m²)
Custom Variable Mass
ECEF Quaternion
Simple Variable Mass 6DOF ECEF (Quaternion)
V_ecef (m/s)
X_ecef (m)
μ / h
LLA_position
ϕ θ ψ (rad)
Euler_RPY
DCM_bi
DCM_be
DCM_ef
V_b (m/s)
V_body
ω_rel (rad/s)
ω_b (rad/s)
dω_b/dt
A_bb (m/s²)
States . V_ecef
States . X_ecef
States . LLA
States . Euler_RPY
States . DCM_bi
States . DCM_be
States . DCM_ef
States . V_b
States . Omega_rel
States . Omega_b
States . dOmega_b/dt
States . A_bb
α
β
V
States . AoA
States . AoS
States . Airspeed
1/2 ρV² q
States . DynamicPressure
Env . rho
Mach
Env . a
States . Mach


Env . g
[1x3]
Mass . m
[1x3]
[1x3]
[1x3]
[1x3]
g
State . DCM_be
[3x3]
[3x3]
DCM_be
State . DCM_ef
[3x3]
[3x3]
DCM_ef
F_grav
[1x3]
[1x3]
1
F_grav
Transform Gravity to Body Axes


1
g
[1x3]
[1x3]
Matrix Multiply
[1x3]
[3x3]
ECEF to NED
[1x3]
Matrix Multiply
[1x3]
[3x3]
NED to BFF
[1x3]
[1x3]
1
F_grav
3
DCM_ef
[3x3]
[3x3]
inv(A) (3x3)
[3x3]
2
DCM_be
[3x3]
[3x3]
inv(A) (3x3)
[3x3]

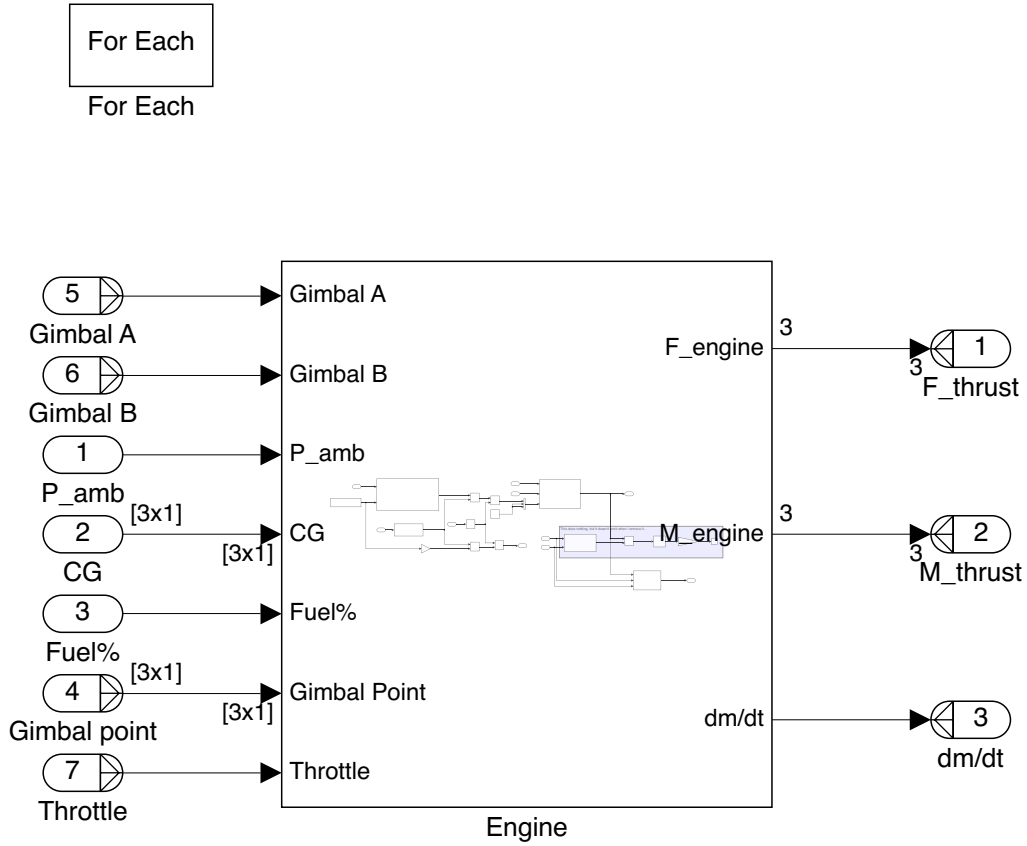

1
Propulsion dm/dt
Mass . dm/dt
vehicle.mass.full
Wet mass
Fuel Weight 100%
Mass . Fuel%
vehicle.mass.empty
Dry mass
Fuel Weight
1/s
Mass Integrator
Mass . m
mass
I
dm/dt
dI/dt
[3x3]
Mass . Inertia
Mass . dI/dt
mass
CG
dm/dt
dCG/dt
[3x1]
3
Mass . CG
Mass . dCG/dt

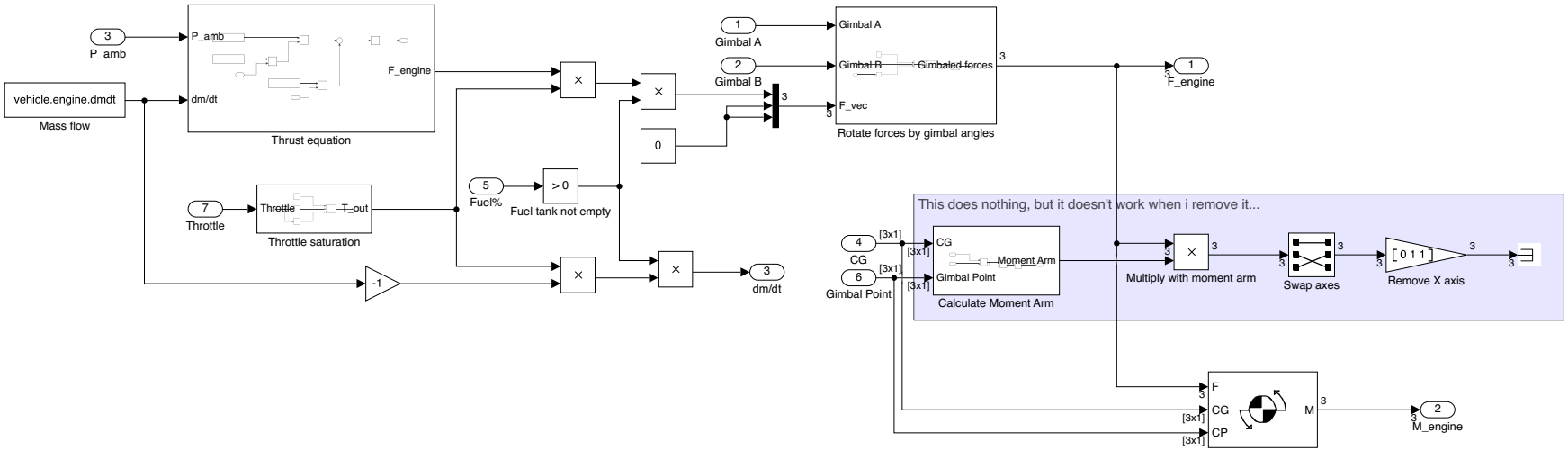

Env . P
Mass . CG
[3x1]
Mass . Fuel%
vehicle.engine.gimbal_mounts
[3x9]
Control . GimbalA
Control . GimbalB
Control . RollRate
u
y
repmat
[1x9]
G_a
G_b
roll
fcn
Control . METhrottle
P_amb
CG
Fuel%
Gimbal point
Gimbal A
Gimbal B
Throttle
For Each
9
Engine
F_thrust
M_thrust
dm/dt
[9x1]
Octaweb
u^T
[1x3]
1
2
3
Prop . CMD_GimbalA
Prop . CMD_GimbalB

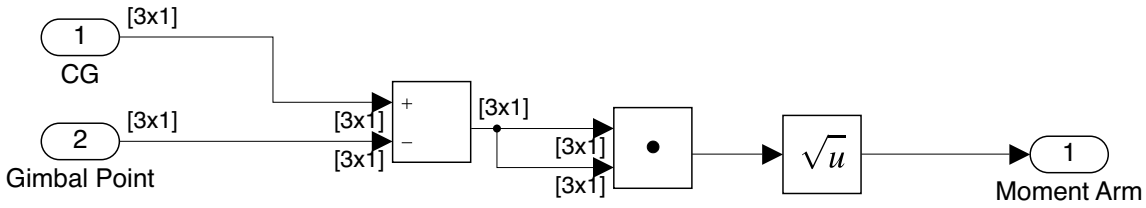
```
function y = repmat(u)
y = repmat(u, 1, 9);
```

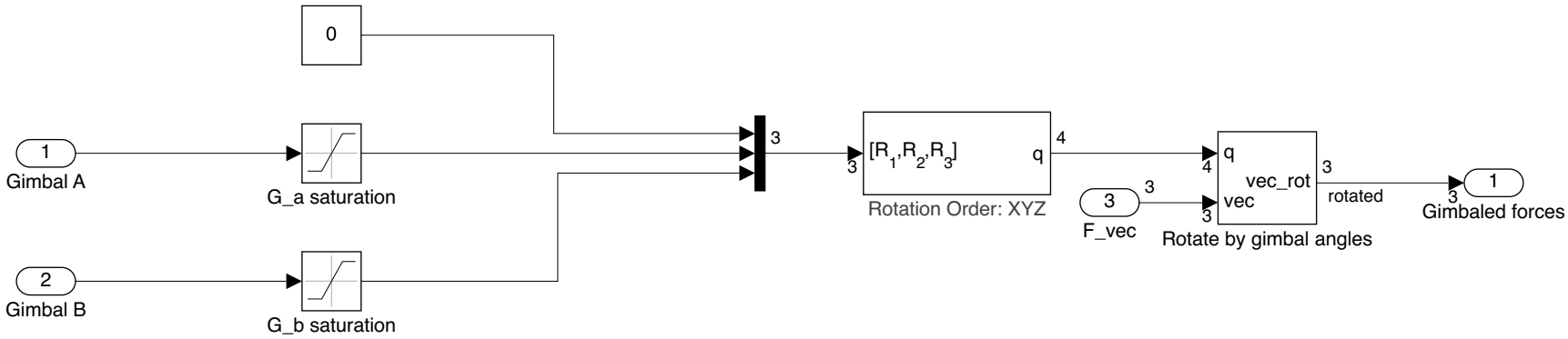
```
function y = repmat(u)
y = repmat(u, 1, 9);
```

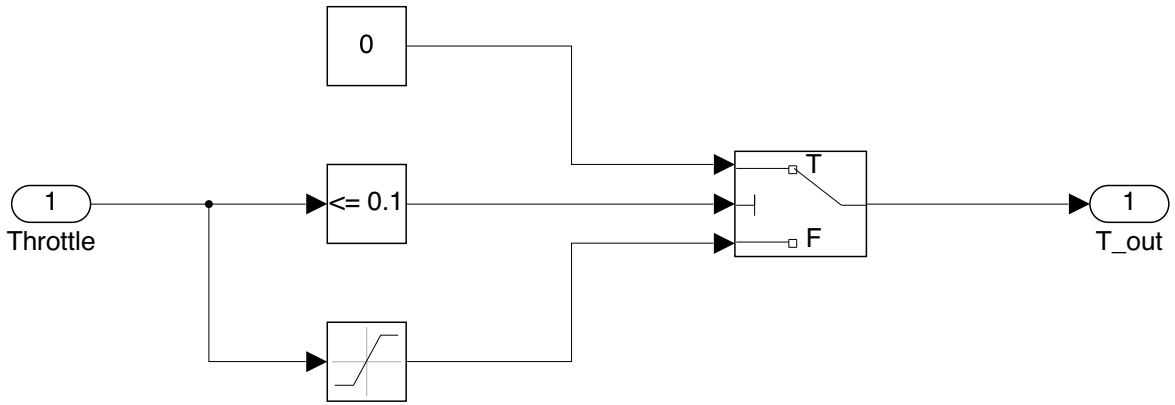
```
function [G_a, G_b]= fcn(G_a, G_b, roll)
    G_a(1) = G_a(1) + roll;
    G_b(3) = G_b(3) - roll;
    G_a(5) = G_a(5) - roll;
    G_b(7) = G_b(7) + roll;
end
```

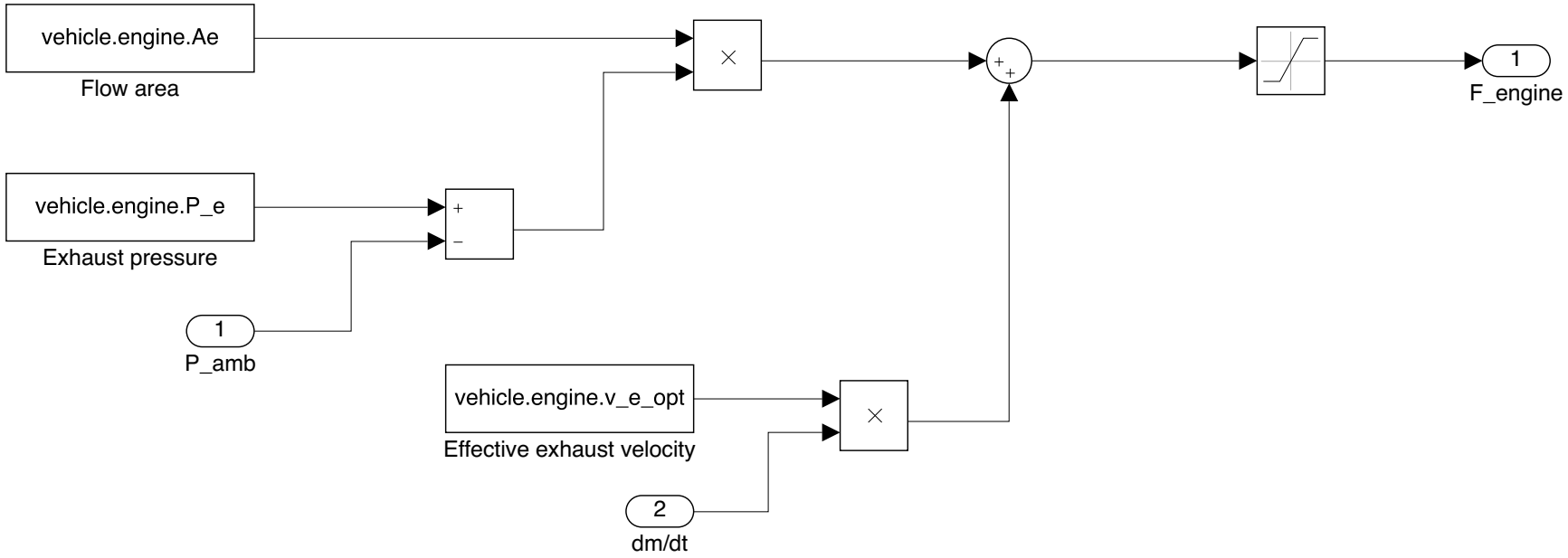

For Each
For Each
5
Gimbal A
6
Gimbal B
1
P_amb
2
CG
3
Fuel%
4
Gimbal point
7
Throttle
[3x1]
[3x1]
[3x1]
[3x1]
Gimbal A
Gimbal B
P_amb
CG
Fuel%
Gimbal Point
Throttle
F_engine
M_engine
dm/dt
3
3
3
1
F_thrust
2
M_thrust
3
dm/dt
Engine

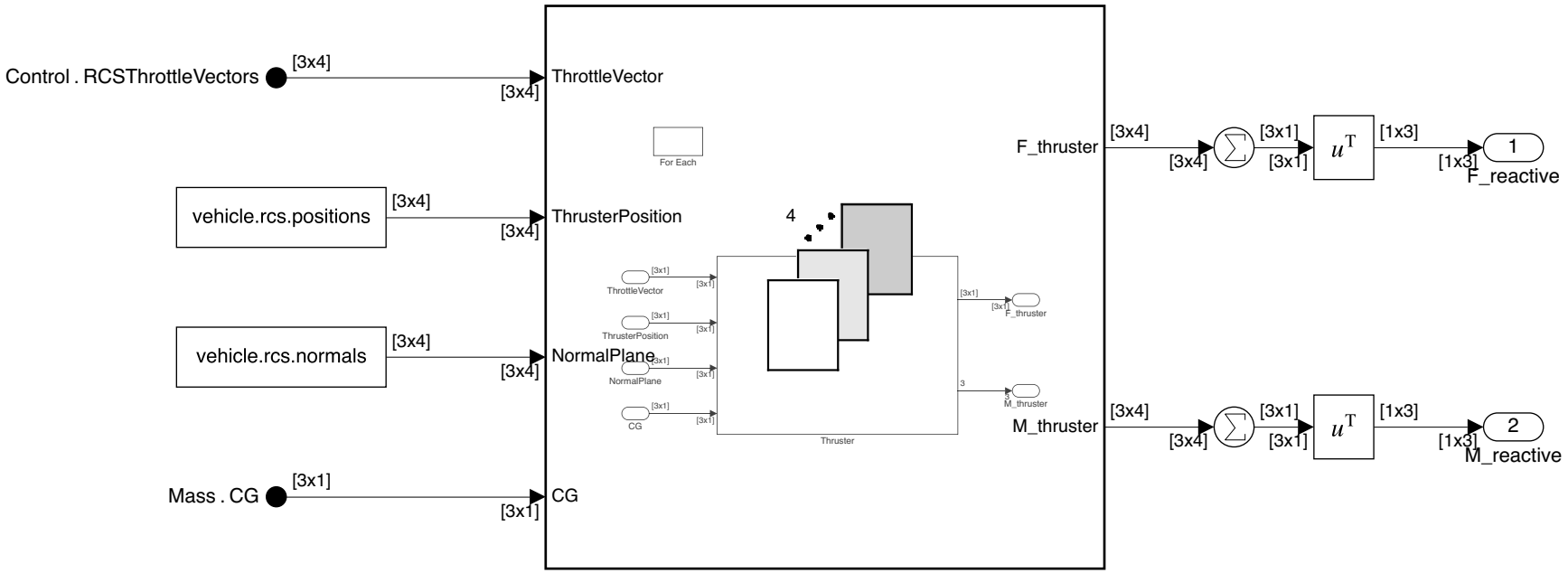

3
P_amb
vehicle.engine.dmdt
Mass flow
P_amb
dm/dt
F_engine
Thrust equation
7
Throttle
Throttle
T_out
Throttle saturation
-1
5
Fuel%
> 0
Fuel tank not empty
×
×
×
×
0
3
dm/dt
1
Gimbal A
2
Gimbal B
Gimbal A
Gimbal B
Gimbaled forces
F_vec
3
Rotate forces by gimbal angles
1
F_engine
This does nothing, but it doesn't work when i remove it...
4
CG
6
Gimbal Point
[3x1]
CG
Moment Arm
Gimbal Point
Calculate Moment Arm
Multiply with moment arm
Swap axes
[0 1 1]
Remove X axis
F
CG
CP
M
2
M_engine

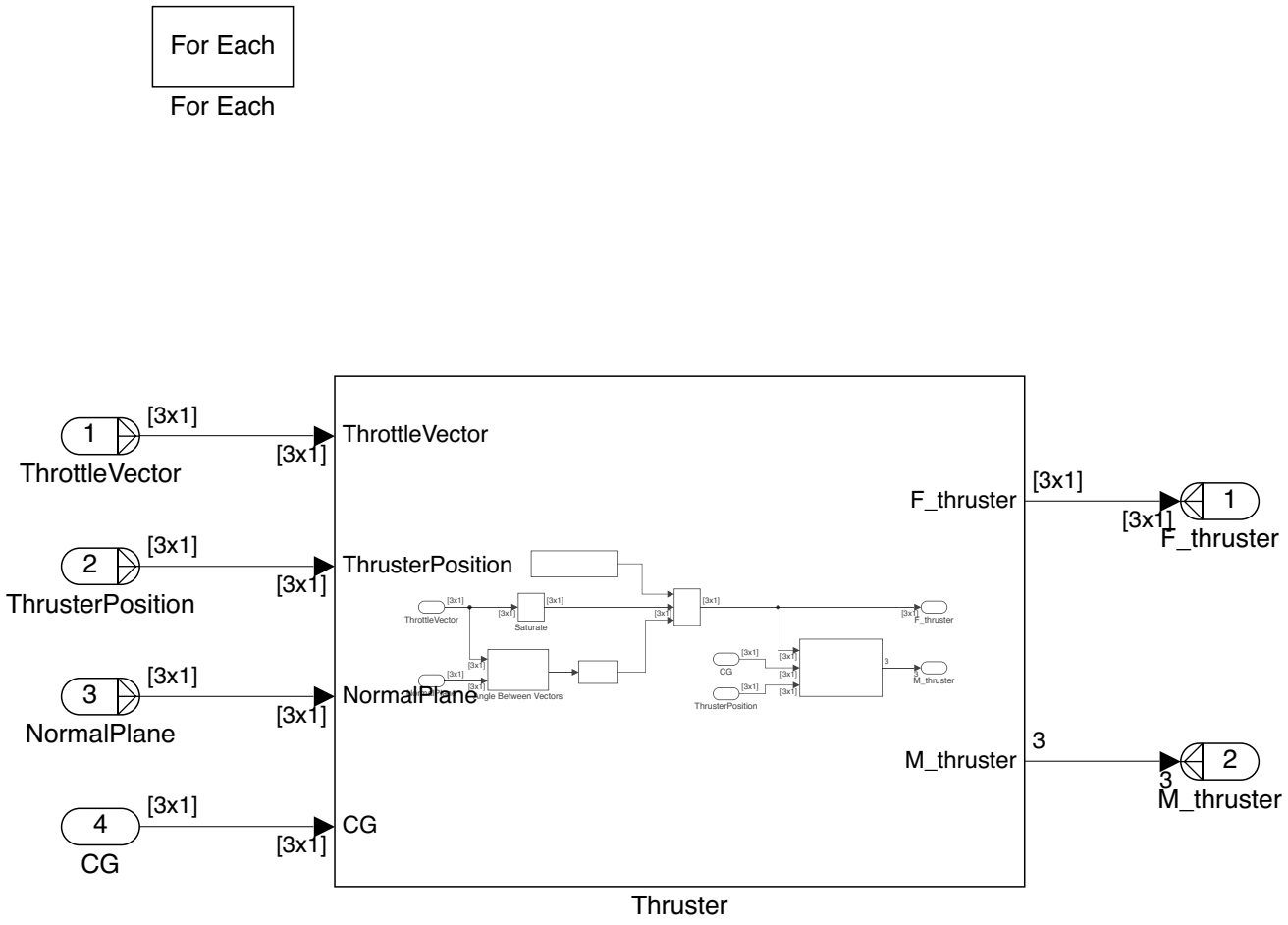

1
CG
[3x1]
2
Gimbal Point
[3x1]
+
−
[3x1]
[3x1]
[3x1]
[3x1]
[3x1]
$\sqrt{u}$
1
Moment Arm

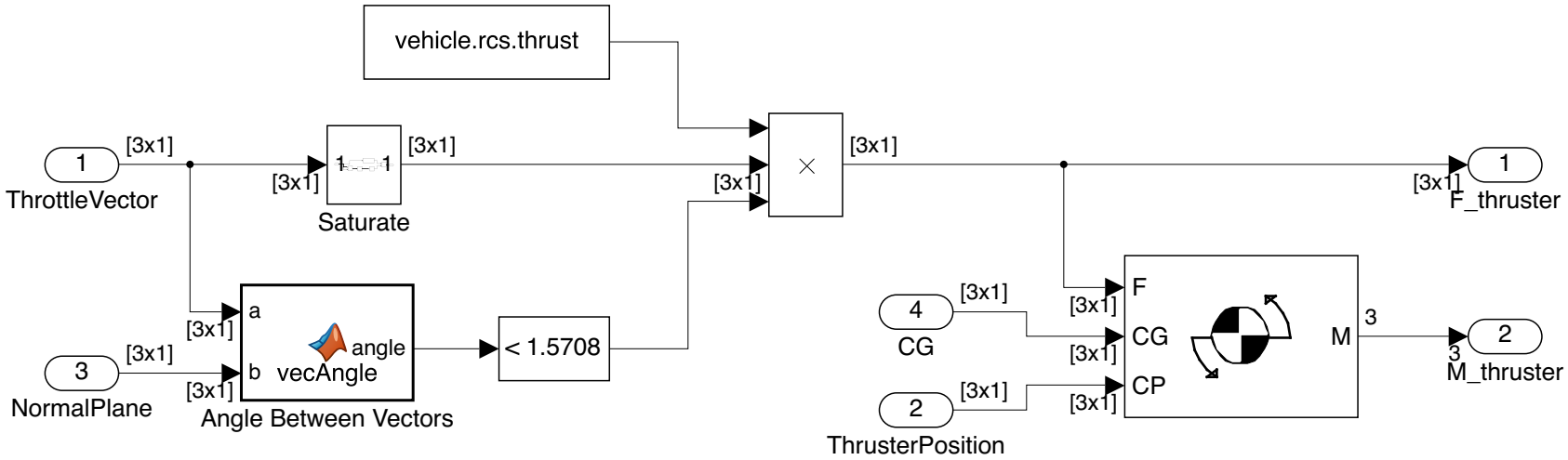

0
1
Gimbal A
G_a saturation
2
Gimbal B
G_b saturation
3
3
[R1,R2,R3]
q
4
Rotation Order: XYZ
3
F_vec
q
vec_rot
vec
3
3
rotated
Rotate by gimbal angles
1
Gimbaled forces


0
1
Throttle
<= 0.1
T
F
1
T_out

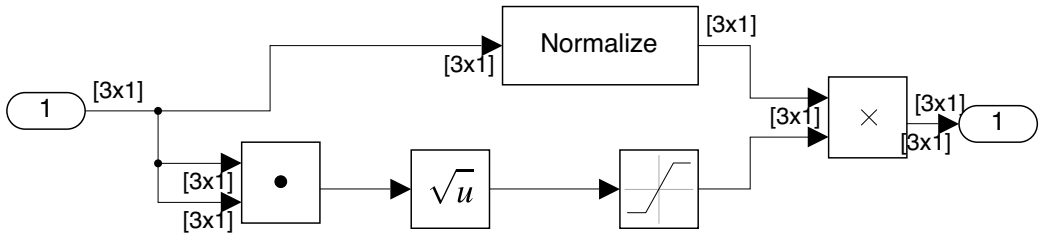

vehicle.engine.Ae
Flow area
vehicle.engine.P_e
Exhaust pressure
1
P_amb
vehicle.engine.v_e_opt
Effective exhaust velocity
2
dm/dt
1
F_engine


Control . RCSThrottleVectors
[3x4]
[3x4]
ThrottleVector
vehicle.rcs.positions
[3x4]
[3x4]
ThrusterPosition
vehicle.rcs.normals
[3x4]
[3x4]
NormalPlane
Mass . CG
[3x1]
[3x1]
CG
For Each
4
ThrottleVector
ThrusterPosition
NormalPlane
CG
Thruster
F_thruster
M_thruster
F_thruster
[3x4]
[3x4]
[3x1]
[3x1]
u^T
[1x3]
[1x3]
1
F_reactive
M_thruster
[3x4]
[3x4]
[3x1]
[3x1]
u^T
[1x3]
[1x3]
2
M_reactive
RCS Array


For Each
For Each
1
ThrottleVector
[3x1]
[3x1]
ThrottleVector
2
ThrusterPosition
[3x1]
[3x1]
ThrusterPosition
3
NormalPlane
[3x1]
[3x1]
NormalPlane
4
CG
[3x1]
[3x1]
CG
F_thruster
[3x1]
[3x1]
1
F_thruster
M_thruster
3
3
2
M_thruster
Saturate
Angle Between Vectors
Thruster


vehicle.rcs.thrust
1
ThrottleVector
[3x1]
Saturate
[3x1]
[3x1]
[3x1]
[3x1]
1
F_thruster
[3x1]
3
NormalPlane
[3x1]
a
b
angle
vecAngle
Angle Between Vectors
[3x1]
[3x1]
< 1.5708
4
CG
[3x1]
2
ThrusterPosition
[3x1]
F
CG
CP
M
3
3
2
M_thruster
[3x1]
[3x1]

```
function angle = vecAngle(a, b)
angle = atan2(norm(cross(a, b)), dot(a, b));
```


1
[3x1]
Normalize
[3x1]
[3x1]
[3x1]
[3x1]
[3x1]
[3x1]
[3x1]
[3x1]
1