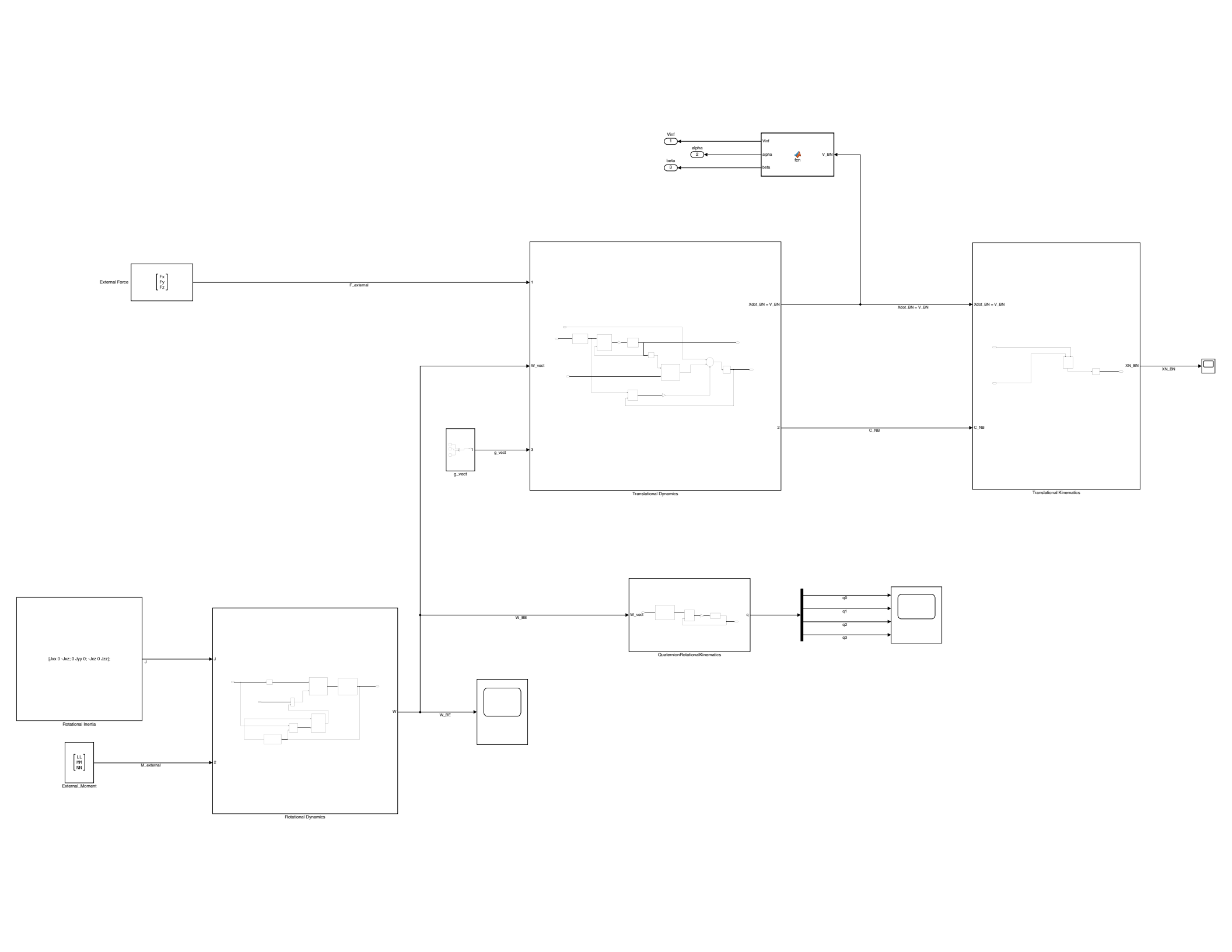
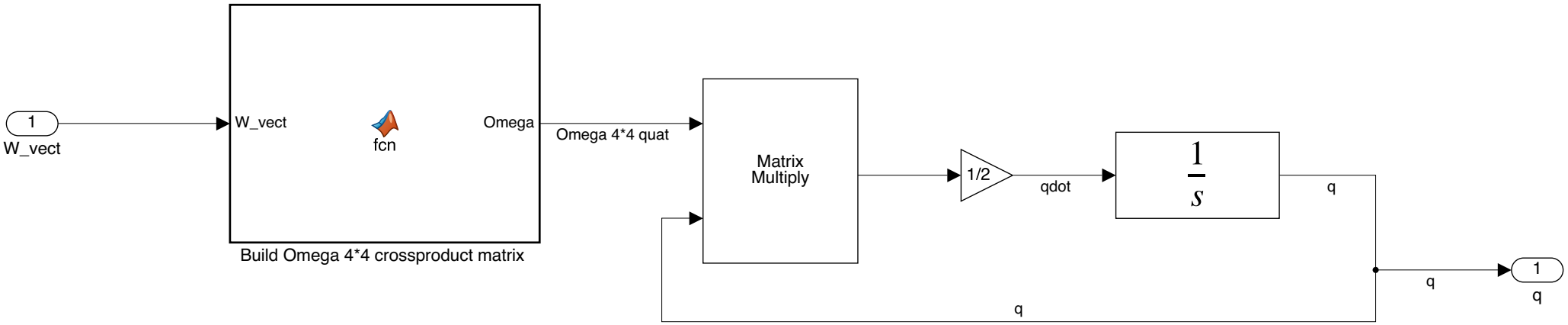
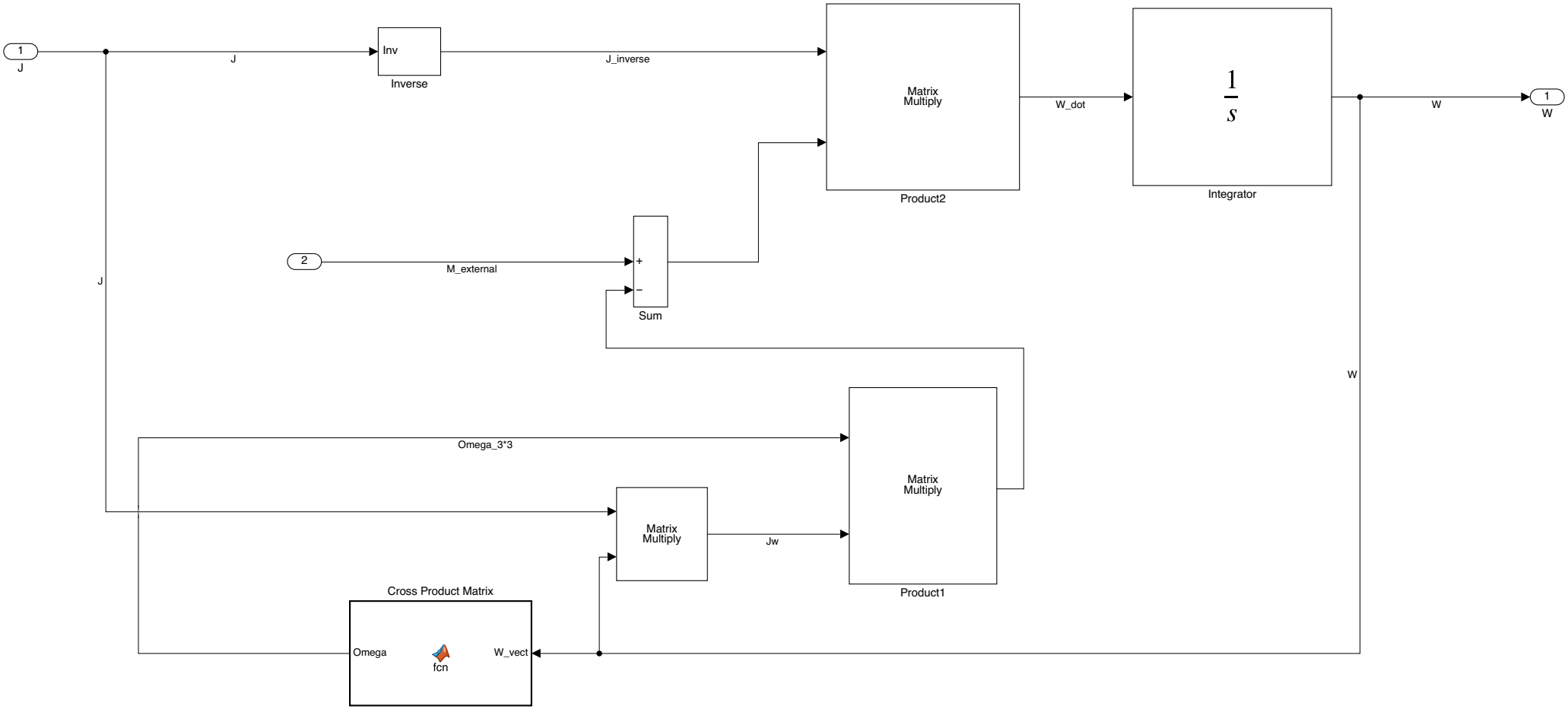

Vinf
1
alpha
2
beta
3
Vinf
alpha
beta
fcn
V_BN
External Force
Fx
Fy
Fz
F_external
1
W_vect
3
g_vect
1
g_vect
Translational Dynamics
Xdot_BN = V_BN
Xdot_BN = V_BN
Xdot_BN = V_BN
XN_BN
XN_BN
2
C_NB
C_NB
Translational Kinematics
[Jxx 0 -Jxz; 0 Jyy 0; -Jxz 0 Jzz];
J
Rotational Inertia
J
W
W_BE
W_BE
External_Moment
LL
MM
NN
M_external
2
Rotational Dynamics
W_BE
W_vect
q
QuaternionRotationalKinematics
q0
q1
q2
q3

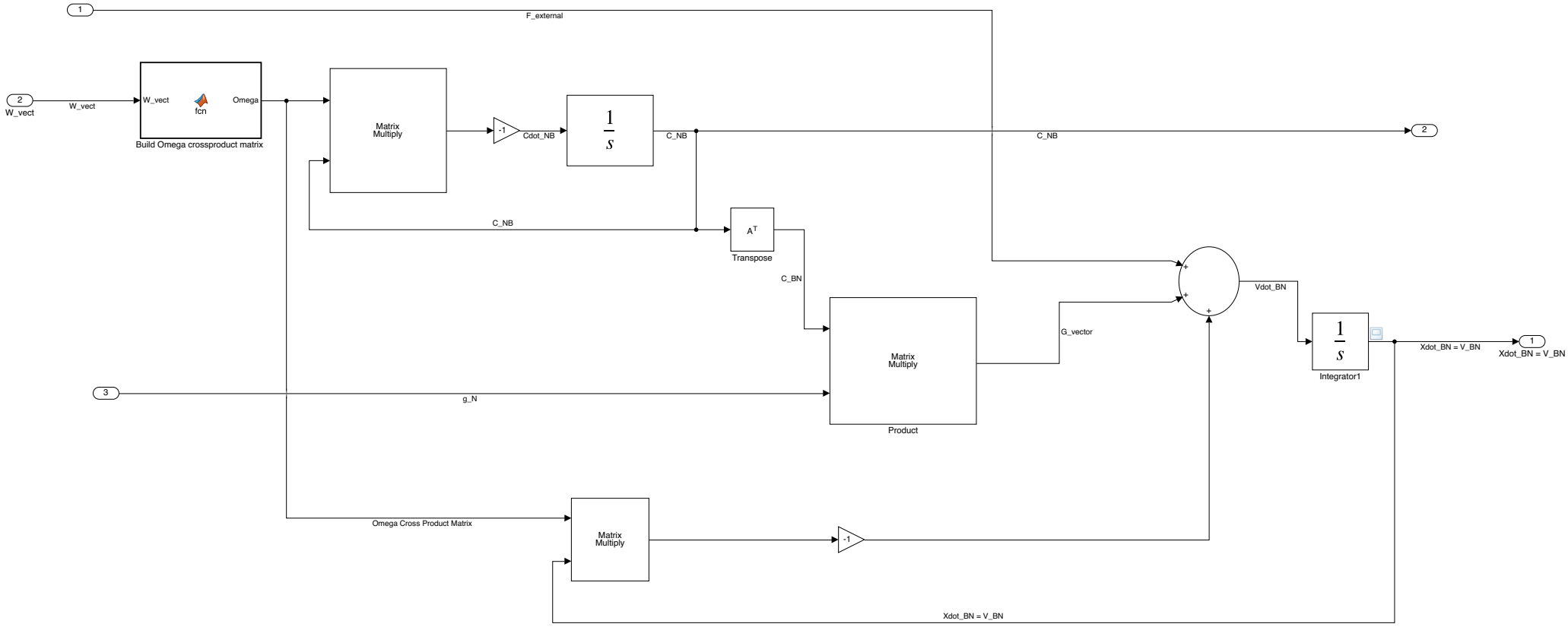
```
function [Vinf,alpha,beta]= fcn(V_BN)

alpha = atan2(V_BN(3),V_BN(1));
u = norm(V_BN);
X = V_BN(2)/u;
beta = asin(X);
Vinf = u;
```


1
W_vect
W_vect
fcn
Omega
Build Omega 4*4 crossproduct matrix
Omega 4*4 quat
Matrix
Multiply
1/2
qdot
1
s
q
q
1
q
q

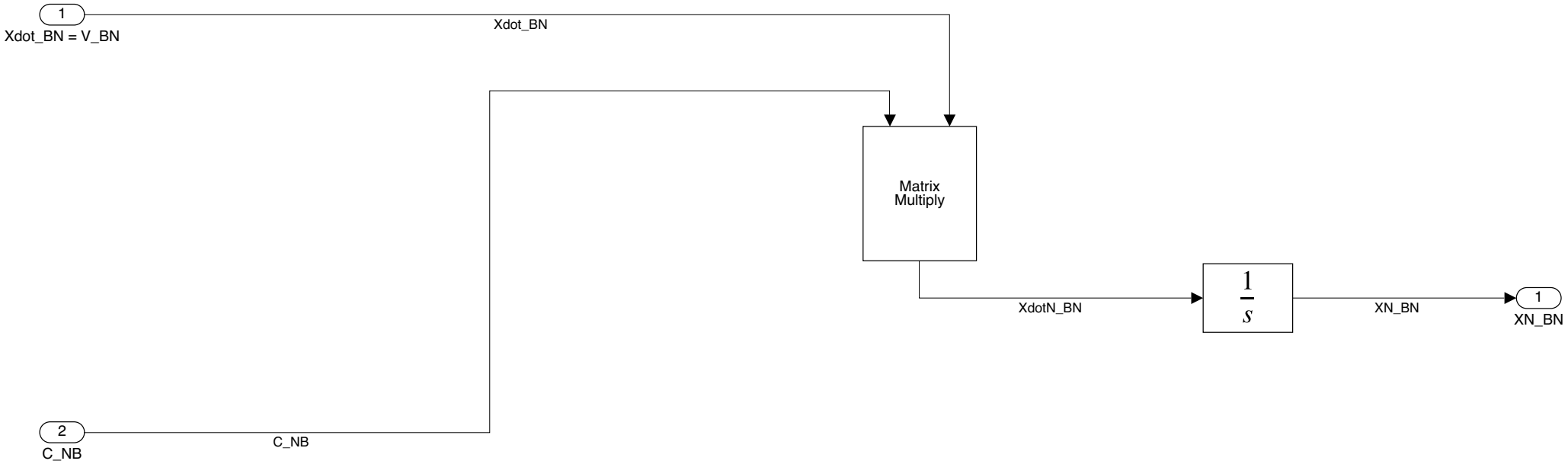
```
function Omega = fcn(W_vect)

wx = W_vect(1);
wy = W_vect(2);
wz = W_vect(3);


Omega = [0 -wx -wy -wz;
    wx 0 wz -wy;
    wy -wz 0 wx;
    wz wy -wx 0];
```

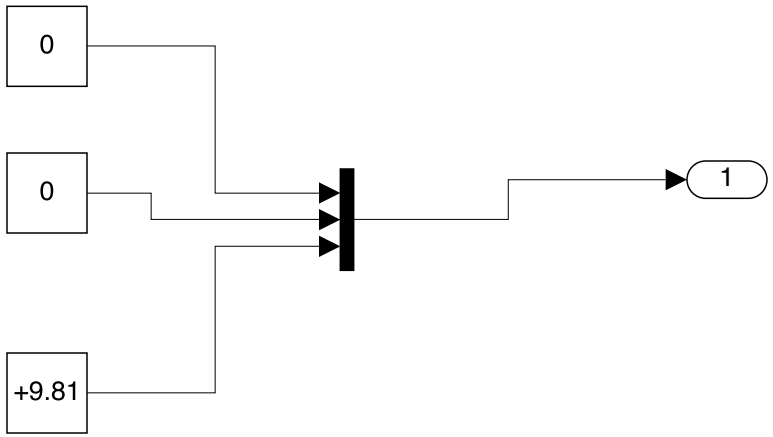

1
J
J
Inv
Inverse
J_inverse
Matrix
Multiply
Product2
W_dot
1
s
Integrator
W
1
W
2
M_external
+
−
Sum
J
W
Omega_3*3
Matrix
Multiply
Product1
Matrix
Multiply
Jw
Cross Product Matrix
Omega
fcn
W_vect

```
function Omega = fcn(W_vect)

p = W_vect(1);
q = W_vect(2);
r = W_vect(3);


Omega = [0 -r q;
    r 0 -p;
    -q p 0];
```


1
F_external
2
W_vect
W_vect
W_vect
fcn
Omega
Build Omega crossproduct matrix
Matrix Multiply
-1
Cdot_NB
1
s
C_NB
C_NB
2
C_NB
A^T
Transpose
C_BN
3
g_N
Matrix Multiply
Product
G_vector
Vdot_BN
1
s
Integrator1
Xdot_BN = V_BN
1
Xdot_BN = V_BN
Omega Cross Product Matrix
Matrix Multiply
-1
Xdot_BN = V_BN

```
function Omega = fcn(W_vect)

p = W_vect(1);
q = W_vect(2);
r = W_vect(3);


Omega = [0 -r q;
    r 0 -p;
    -q p 0];
```


1
Xdot_BN = V_BN
Xdot_BN
Matrix
Multiply
XdotN_BN
1
s
XN_BN
1
XN_BN
2
C_NB
C_NB


0
0
+9.81
1