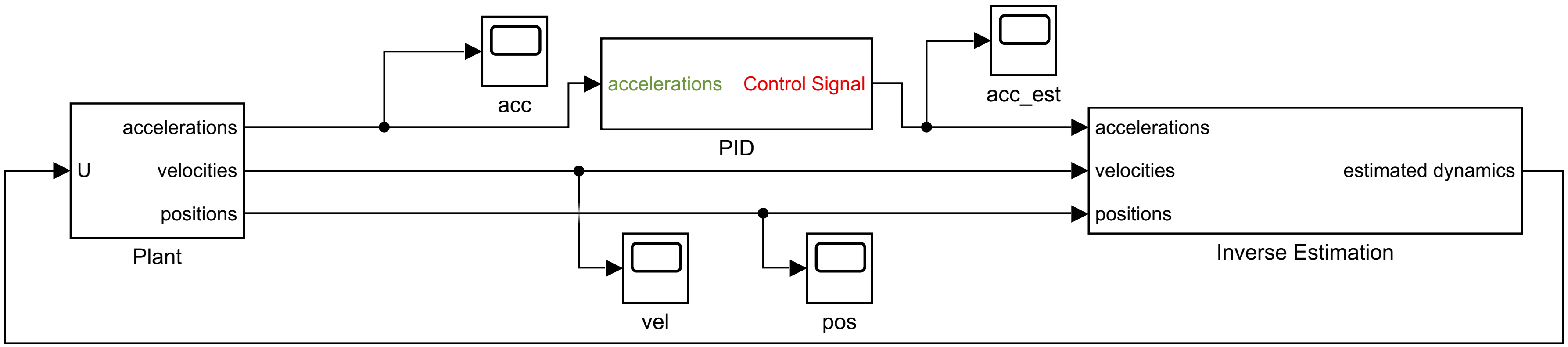
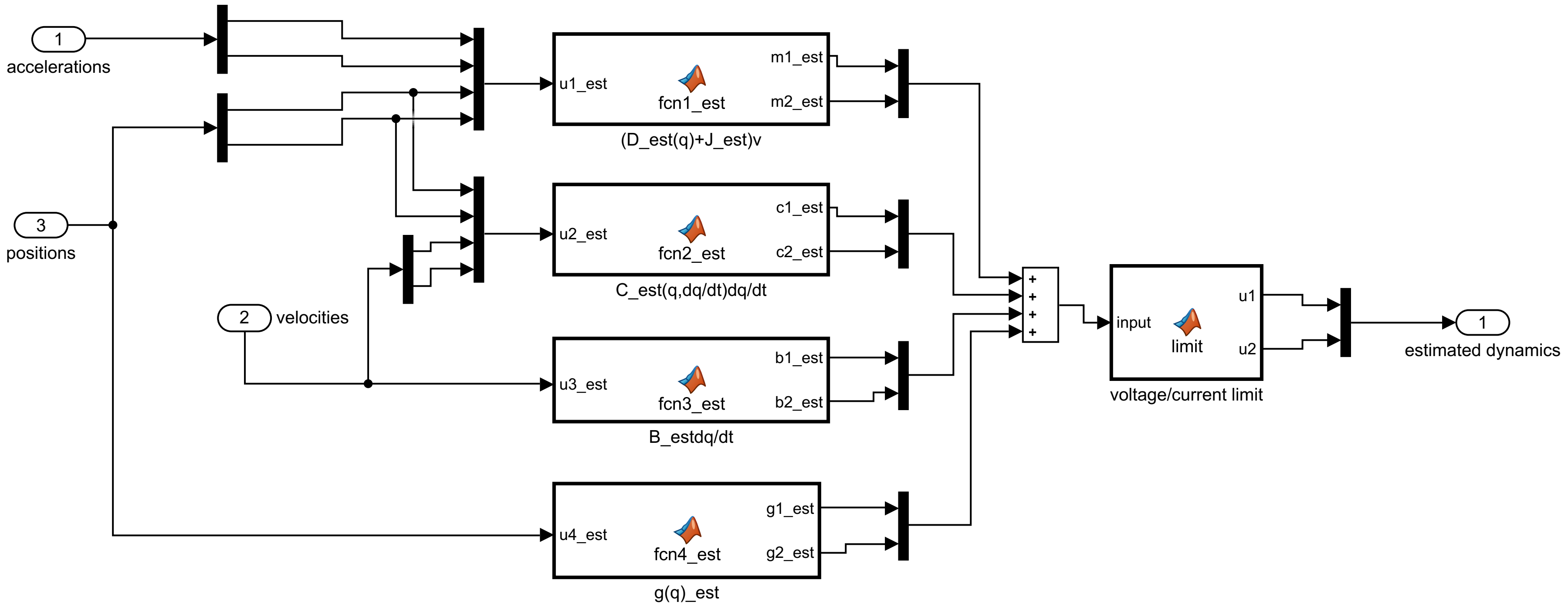

acc
accelerations
Control Signal
PID
acc_est
accelerations
velocities
positions
U
Plant
accelerations
velocities
positions
estimated dynamics
Inverse Estimation
vel
pos


1
accelerations
3
positions
2
velocities
u1_est
fcn1_est
m1_est
m2_est
(D_est(q)+J_est)v
u2_est
fcn2_est
c1_est
c2_est
C_est(q,dq/dt)dq/dt
u3_est
fcn3_est
b1_est
b2_est
B_estdq/dt
u4_est
fcn4_est
g1_est
g2_est
g(q)_est
+
+
+
+
input
limit
u1
u2
voltage/current limit
1
estimated dynamics

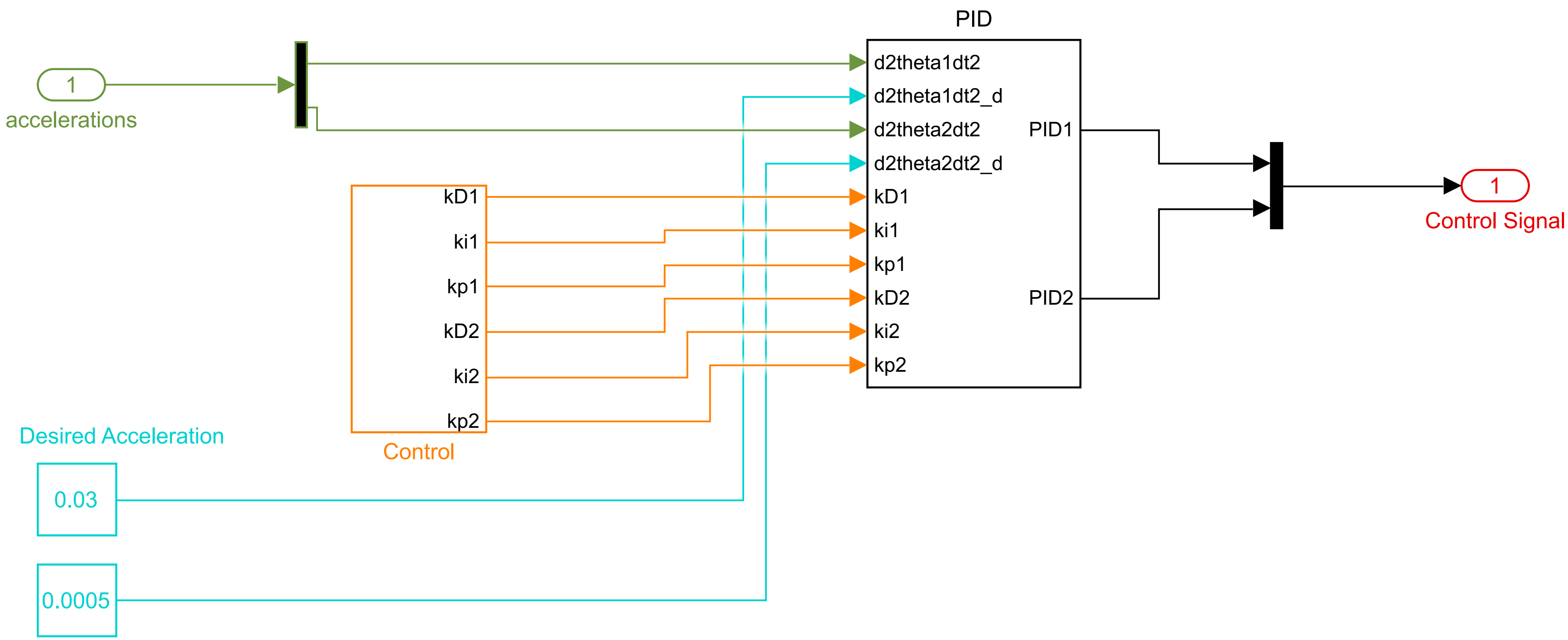
```
function [m1_est, m2_est] = fcn1_est(u1_est)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

if length(u1_est) == 4

    % error margin
    err = 1.01; % error for everything

    acc1_est = u1_est(1); % estimated acceleration for first actuator
    acc2_est = u1_est(2); % estimated acceleration for second actuator
    pos2 = err*u1_est(4); % second actuator position with noise

    % motor variables
    j1 = err*3060e-7; % rotor inertia
    j2 = j1; % rotor inertia

    % Link 1
    I133 = err*0.000891271;
    L1 = err*0.2;
    Lc1 = err*0.1;
    m1 = err*0.358306058;
    % Link 2
    I233 = err*0.000552127;
    L2 = err*0.2;
    Lc2 = err*0.073;
    m2 = err*0.308756558;

    % external variables
    r1 = err*1/4; % first actuator reduction
    r2 = err*1/3; % second actuator reduction

    % matrix variables
    d11_est = m1*Lc1^2 + m2*(L1^2 +Lc2^2 +2*L1*Lc2*cos(pos2)) + I133 +I233;
    d12_est = m2*(Lc2^2 +L1*Lc2*cos(pos2)) +I233;
    d21_est = m2*(Lc2^2 +L1*Lc2*cos(pos2)) +I233;
    d22_est = m2*Lc2^2 +I233;
    j11_est = 1/(r1^2)*j1;
    j22_est = 1/(r2^2)*j2;

    M_est = [(d11_est +j11_est) d12_est;...
        d21_est (d22_est +j22_est)]; % D(q)+J/r^2  (2x2) matrix

    Mmul = M_est*[acc1_est; acc2_est];

    m1_est = Mmul(1,:);
    m2_est = Mmul(2,:);

else

    m1_est = 0;
    m2_est = 0;

end
```

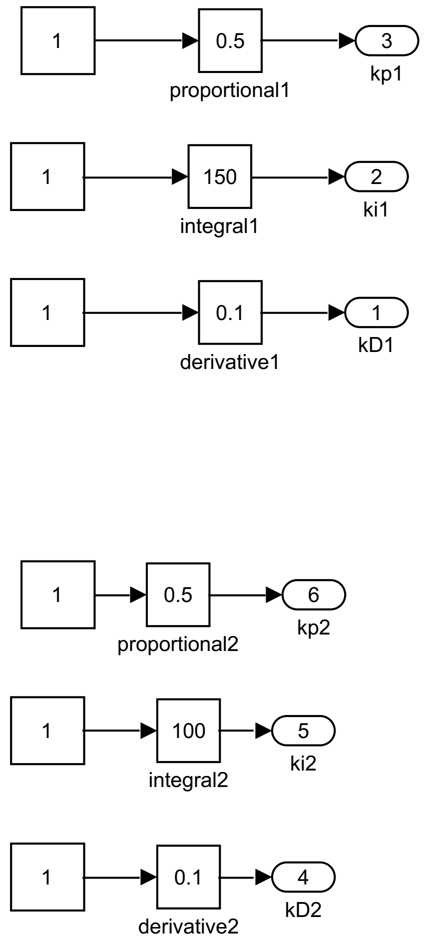
```
function [b1_est, b2_est] = fcn3_est(u3_est)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

if length(u3_est) == 2

    % error margin
    err = 1.01; % error for everything

    vel1 = err*u3_est(1); % first actuator velocity
    vel2 = err*u3_est(2); % second actuator velocity

    % motor variables
    kb = err*1/(44*2*pi/60); % back emf constant
    km = err*217e-3; % torque constant
    r = err*2.30; % motor resistance
    mt = err*14.9e-3; % mechanical time constant
    j1 = err*3060e-7; % rotor inertia
    b1 = mt*j1; % motor friction
    b2 = b1; % motor friction

    % external variables
    r1 = err*1/4; % first actuator reduction
    r2 = err*1/3; % second actuator reduction

    % matrix variables
    b1_est = 1/(r1^2)*(b1 +km*kb/r);
    b2_est = 1/(r2^2)*(b2 +km*kb/r);

    B_est = [b1_est 0; 0 b2_est];

    Bmul_est = B_est*[vel1; vel2];

    b1_est = Bmul_est(1,:);
    b2_est = Bmul_est(2,:);

else

    b1_est = 0;
    b2_est = 0;

end
```

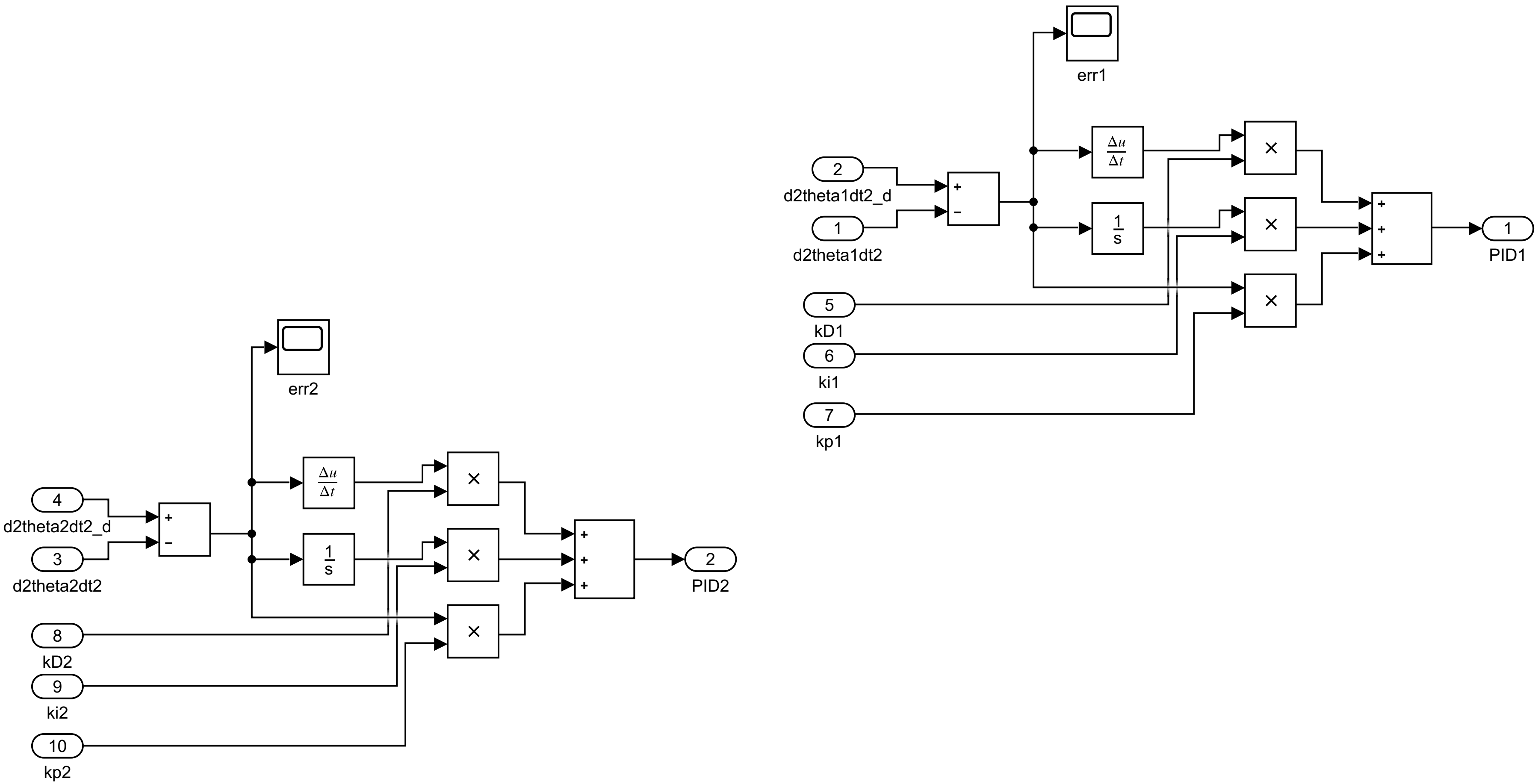
```
function [c1_est, c2_est] = fcn2_est(u2_est)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

if length(u2_est) == 4

    % error margin
    err = 1.01; % error for everything

    pos2 = err*u2_est(2); % second actuator position
    vel1 = err*u2_est(3); % first actuator velocity
    vel2 = err*u2_est(4); % second actuator velocity

    % Link 1
    I133 = err*0.000891271;
    L1 = err*0.2;
    Lc1 = err*0.1;
    m1 = err*0.358306058;
    % Link 2
    I233 = err*0.000552127;
    L2 = err*0.2;
    Lc2 = err*0.073;
    m2 = err*0.308756558;

    % matrix variables
    c111_est = 0;
    c112_est = m2*L1*Lc2*sin(pos2);
    c121_est = -c112_est;
    c211_est = -c112_est;
    c122_est = 0;
    c212_est = 0;
    c221_est = -c112_est;
    c222_est = 0;

    C_est = [(c111_est*vel1 +c211_est*vel2) (c121_est*vel1 +c221_est*vel2); ...
        (c112_est*vel1 +c212_est*vel2) (c122_est*vel1 +c222_est*vel2)]; % C (2x2) matrix

    Cmul_est = C_est*[vel1; vel2];

    c1_est = Cmul_est(1,:);
    c2_est = Cmul_est(2,:);

else

    c1_est = 0;
    c2_est = 0;

end
```

```
function [g1_est, g2_est] = fcn4_est(u4_est)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

if length(u4_est) == 2

    % error margin
    err = 1.01; % error for everything

    pos1 = err*u4_est(1); % first actuator position
    pos2 = err*u4_est(2); % second actuator position

    % Link 1
    I133 = err*0.000891271;
    L1 = err*0.2;
    Lc1 = err*0.1;
    m1 = err*0.358306058;
    % Link 2
    I233 = err*0.000552127;
    L2 = err*0.2;
    Lc2 = err*0.073;
    m2 = err*0.308756558;

    % external variables
    g = err*9.81; % gravitational acceleration

    % matrix variables
    phi1_est = (m1*Lc1 +m2*L1)*g*cos(pos1) +m2*g*Lc2*cos(pos1 +pos2);
    phi2_est = m2*Lc2*cos(pos1 +pos2);

    g1_est = phi1_est;
    g2_est = phi2_est;

else

    g1_est = 0;
    g2_est = 0;

end
```

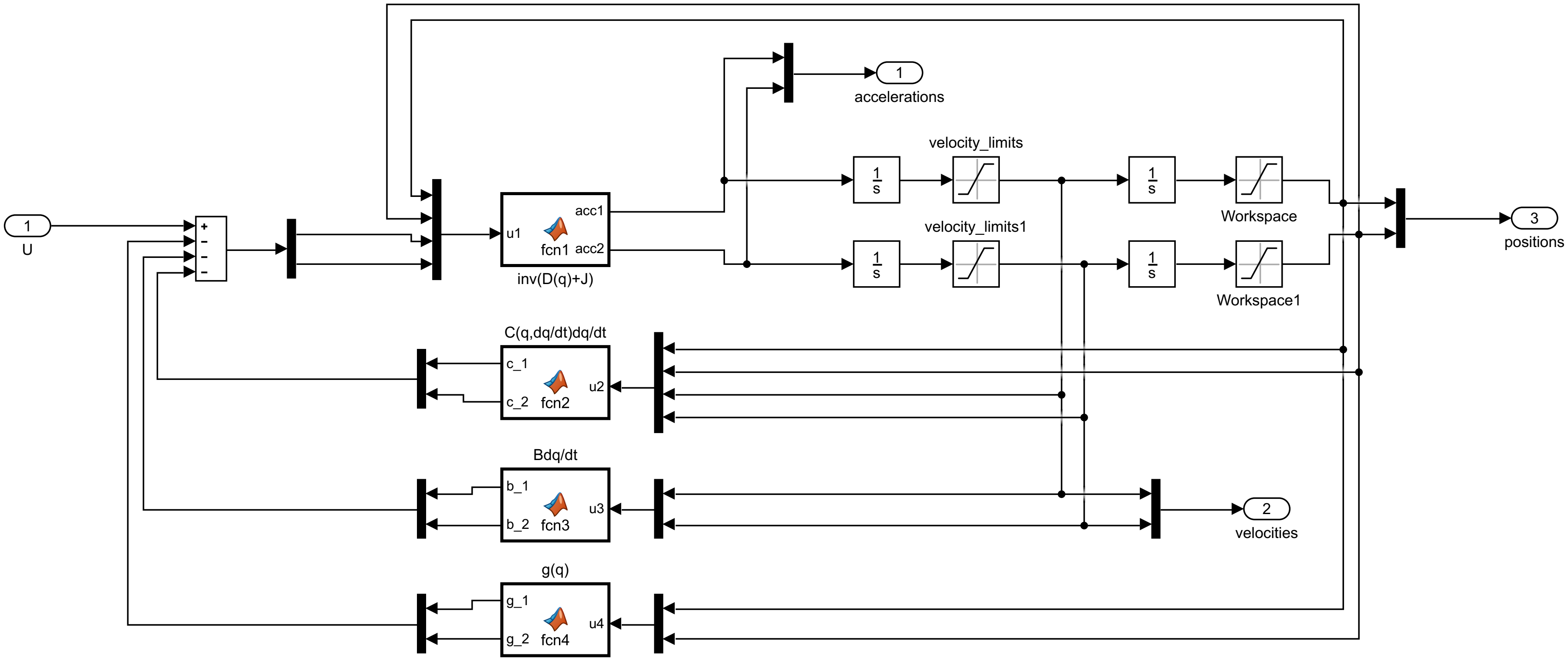
```
function [u1, u2] = limit(input)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

% Here we estimate the required voltage! In this example, the controller
% works on accelerations, not directly on the voltage. The reason is, the
% controller can suddenly estimate excessive/impossible/dangerous voltages.

if length(input) == 2

u1 = input(1);
u2 = input(2);

% error margin
err = 1.01; % error for everything

% motor variables
km = err*217e-3; % torque constant
r = err*2.30; % motor resistance

% external variables
r1 = err*1/4; % first actuator reduction
r2 = err*1/3; % second actuator reduction

% volt limits
vlim = 48; % absolute limit

volt1 = u1*r1*r/km;
volt2 = u2*r2*r/km;

if volt1 >= vlim
    volt1 = vlim;
elseif volt1 <= -vlim
    volt1 = -vlim;
else
    volt1 = volt1; %#ok<ASGSL>
end

if volt2 >= vlim
    volt2 = vlim;
elseif volt2 <= -vlim
    volt2 = -vlim;
else
    volt2 = volt2; %#ok<ASGSL>
end

u1 = volt1*km/(r*r1);
u2 = volt2*km/(r*r2);

else

    u1 = 0;
u2 = 0;

end
```


PID
1
accelerations
d2theta1dt2
d2theta1dt2_d
d2theta2dt2
d2theta2dt2_d
kD1
ki1
kp1
kD2
ki2
kp2
PID1
PID2
kD1
ki1
kp1
kD2
ki2
kp2
Control
Desired Acceleration
0.03
0.0005
1
Control Signal


1
0.5
3
proportional1
kp1
1
150
2
integral1
ki1
1
0.1
1
derivative1
kD1
1
0.5
6
proportional2
kp2
1
100
5
integral2
ki2
1
0.1
4
derivative2
kD2


err1
2
d2theta1dt2_d
1
d2theta1dt2
5
kD1
6
ki1
7
kp1
1
PID1
err2
4
d2theta2dt2_d
3
d2theta2dt2
8
kD2
9
ki2
10
kp2
2
PID2

Project/Plant



```
function [b_1, b_2] = fcn3(u3)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

if length(u3) == 2

    vel1 = u3(1); % first actuator velocity
    vel2 = u3(2); % second actuator velocity

    % motor variables
    kb = 1/(44*2*pi/60); % back emf constant
    km = 217e-3; % torque constant
    r = 2.30; % motor resistance
    mt = 14.9e-3; % mechanical time constant
    j1 = 3060e-7; % rotor inertia
    b1 = mt*j1; % motor friction
    b2 = b1; % motor friction

    % external variables
    r1 = 1/4; % first actuator reduction
    r2 = 1/3; % second actuator reduction

    % matrix variables
    b1 = 1/(r1^2)*(b1 +km*kb/r);
    b2 = 1/(r2^2)*(b2 +km*kb/r);

    B = [b1 0; 0 b2];

    Bmul = B*[vel1; vel2];

    b_1 = Bmul(1,:);
    b_2 = Bmul(2,:);

else

    b_1 = 0;
    b_2 = 0;

end
```

```
function [c_1, c_2] = fcn2(u2)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

if length(u2) == 4

pos2 = u2(2); % second actuator position
vel1 = u2(3); % first actuator velocity
vel2 = u2(4); % second actuator velocity

% Link 1
I133 = 0.000891271;
L1 = 0.2;
Lc1 = 0.1;
m1 = 0.358306058;
% Link 2
I233 = 0.000552127;
L2 = 0.2;
Lc2 = 0.073;
m2 = 0.308756558;

% matrix variables
c111 = 0;
c112 = m2*L1*Lc2*sin(pos2);
c121 = -c112;
c211 = -c112;
c122 = 0;
c212 = 0;
c221 = -c112;
c222 = 0;

C = [(c111*vel1 +c211*vel2) (c121*vel1 +c221*vel2);...
    (c112*vel1 +c212*vel2) (c122*vel1 +c222*vel2)]; % C (2x2) matrix

Cmul = C*[vel1; vel2];

c_1 = Cmul(1,:);
c_2 = Cmul(2,:);

else

c_1 = 0;
c_2 = 0;

end
```

```
function [g_1, g_2] = fcn4(u4)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

if length(u4) == 2

    pos1 = u4(1); % first actuator position
    pos2 = u4(2); % second actuator position

    % link variables
    % Link 1
    I133 = 0.000891271;
    L1 = 0.2;
    Lc1 = 0.1;
    m1 = 0.358306058;
    % Link 2
    I233 = 0.000552127;
    L2 = 0.2;
    Lc2 = 0.073;
    m2 = 0.308756558;

    % external variables
    g = 9.81; % gravitational acceleration

    % matrix variables
    phi1 = (m1*Lc1 +m2*L1)*g*cos(pos1) +m2*g*Lc2*cos(pos1 +pos2);
    phi2 = m2*Lc2*cos(pos1 +pos2);

    g_1 = phi1;
    g_2 = phi2;

else

    g_1 = 0;
    g_2 = 0;

end
```

```
function [acc1, acc2] = fcn1(u1)
% This block supports an embeddable subset of the MATLAB language.
% See the help menu for details.

% the input U(t) is the net torque on the joints after motor and link
% variables are all accounted for.

if length(u1) == 4

    pos2 = u1(2); % second actuator position
    cbg1 = u1(3); % array element to be multiplied with inv(M)
    cbg2 = u1(4); % array element to be multiplied with inv(M)

    % motor variables
    j1 = 3060e-7; % rotor inertia
    j2 = j1; % rotor inertia

    % Link 1
    I133 = 0.000891271;
    L1 = 0.2;
    Lc1 = 0.1;
    m1 = 0.358306058;
    % Link 2
    I233 = 0.000552127;
    L2 = 0.2;
    Lc2 = 0.073;
    m2 = 0.308756558;

    % external variables
    r1 = 1/4; % first actuator reduction
    r2 = 1/3; % second actuator reduction

    % matrix variables
    d11 = m1*Lc1^2 + m2*(L1^2 +Lc2^2 +2*L1*Lc2*cos(pos2)) + I133 +I233;
    d12 = m2*(Lc2^2 +L1*Lc2*cos(pos2)) +I233;
    d21 = m2*(Lc2^2 +L1*Lc2*cos(pos2)) +I233;
    d22 = m2*Lc2^2 +I233;
    j11 = 1/(r1^2)*j1;
    j22 = 1/(r2^2)*j2;

    M = [(d11 +j11) d12; d21 (d22 +j22)]; % D(q)+J/r^2  (2x2) matrix

    accel = inv(M)*[cbg1; cbg2]; %#ok<MINV>

    acc1 = accel(1,:);
    acc2 = accel(2,:);

else

    acc1 = 0;
acc2 = 0;

end
```