```
%gains for control law
Kp = 50;
Kd = 20;
Ki = 2;
I_trd = 0;
e_pow = 0.9;

%%%%%%%  ENVIRONMENT  %%%%%%%%

g = 9.81; %[N/kg] gracvitational acceleration

%%%%%%%  SYSTEM PROPERTIES %%%%%%%%

m_v = 1200; %[kg] mass of the pod
Mratio = 1; % ratio of ideal(measured/estimated) mass of the pod to the actual mass
of the pod; adjust this multiplier for Mideal/Mreal

I_vx = 250; %[kg*m^2] moment of inertia of the vehicle roll
I_vy = 1000; %[kg*m^2] moment of inertia of the vehicle pitch

%rail and magnet arms
m_r = 15; %[kg] effective mass of the rail
k_r = 500000; %[N/m] spring constant of the rail

m_e = 50; %[kg] mass of electormagnet module
k_b = 1500000; %[N/m] spring cosntant of magnet suspension beam

x_E = 1; %[m] offset position of electromagnets away from CM
y_E = 0.5; %[m] offset position of electromagnets away from CM
z_offset = 0.0; %[m] "vertical" distance between the center of mass and the chassis
level with magnets

m_eff = (m_e + m_v/4); %[N] effective gravitational force to be supported by each
magnet


%%%%%%%  MAGNET  %%%%%%%%
Imax = 100; %[A] maximum current we can supply
gmax = 0.030; %[m] gap width at which imax is needed for equilibrium

Ieq = 6; %[A] equilibrium current - experimental
geq = 0.020; %[A] equilibrium gap width - experimental
ng = 2; %power of gap width in force eq
Cmag = m_eff*g*geq^(ng)/Ieq; %magnet constant from experimental data


%magnet arms
```

```
%time
ti = 0; %[s] initial time
tf = 6; %[s] final time
tn = tf*2000; % number of time steps
t = linspace(ti,tf,tn); %time array
dt = (tf-ti)/tn; %[s] time step size

%disturbances

FdFR = zeros(size(t)); %[N] disturbance force
VdFR = zeros(size(t)); %[N*s] disturbance impulse

FdFL = zeros(size(t)); %[N] disturbance force
VdFL = zeros(size(t)); %[N*s] disturbance impulse

FdAR = zeros(size(t)); %[N] disturbance force
VdAR = zeros(size(t)); %[N*s] disturbance impulse

FdAL = zeros(size(t)); %[N] disturbance force
VdAL = zeros(size(t)); %[N*s] disturbance impulse



%%%%%%%  GAP VARIABLES  %%%%%%%

%front right
gFR_t = zeros(size(t)); %[m/s]
dgFR_dt = zeros(size(t)); %[m/s]
ddgFR_ddt = zeros(size(t)); %[m/s]

%front left
gFL_t = zeros(size(t)); %[m/s]
dgFL_dt = zeros(size(t)); %[m/s]
ddgFL_ddt = zeros(size(t)); %[m/s]

%aft right
gAR_t = zeros(size(t)); %[m/s]
dgAR_dt = zeros(size(t)); %[m/s]
ddgAR_ddt = zeros(size(t)); %[m/s]

%aft left; x is aligned with track, y is perpendicular, z is up
gAL_t = zeros(size(t)); %[m/s]
dgAL_dt = zeros(size(t)); %[m/s]
ddgAL_ddt = zeros(size(t)); %[m/s]

%rotation
alphaX_t = zeros(size(t)); %[rad]
dalphaX_dt = zeros(size(t)); %[rad/s]
ddalphaX_ddt = zeros(size(t)); %[rad/s2]
```

```
alphaY_t = zeros(size(t)); %[rad]
dalphaY_dt = zeros(size(t)); %[rad/s]
ddalphaY_ddt = zeros(size(t)); %[rad/s2]

alphaZ_t = zeros(size(t)); %[rad]
dalphaZ_dt = zeros(size(t)); %[rad/s]
ddalphaZ_ddt = zeros(size(t)); %[rad/s2]

%translation
zCM_t = zeros(size(t)); %[m] postion of the pod`s center of mass
dzCM_dt = zeros(size(t)); %[m/s]
ddzCM_ddt = zeros(size(t)); %[m/s2]

%%%%%%%  FORCE VARIABLES  %%%%%%%

g_ref = -0.01* ones(size(t)); %[m] reference position

IFR = zeros(size(t)); %[A] current
IFL = zeros(size(t)); %[A] current
IAR = zeros(size(t)); %[A] current
IAL = zeros(size(t)); %[A] current

FFR = zeros(size(t)); %[N] force
FFL = zeros(size(t)); %[N] force
FAR = zeros(size(t)); %[N] force
FAL = zeros(size(t)); %[N] force

%error accumulators
accFR = zeros(size(t)); % error accumulator
accFL = zeros(size(t)); % error accumulator
accAR = zeros(size(t)); % error accumulator
accAL = zeros(size(t)); % error accumulator

%%%%%%%  ICs  %%%%%%%

%assume velocities and accelerations are 0s
zCM_t(1) = -0.02 + z_offset;
alphaX_t(1) = 0;
alphaY_t(1) = 0;

Rx = [1 0 0; 0 cos(alphaX_t(1)) -sin(alphaX_t(1)); 0 sin(alphaX_t(1)) cos(alphaY_t ↙
(1))];
Ry = [cos(alphaY_t(1)) 0 sin(alphaY_t(1)); 0 1 0; -sin(alphaY_t(1)) 0 cos(alphaY_t ↙
(1))];
Rz = [cos(alphaZ_t(1)) -sin(alphaZ_t(1)) 0; sin(alphaZ_t(1)) cos(alphaZ_t(1)) 0; 0 0 ↙
1];

m_pos_0 = [x_E x_E -x_E -x_E; -y_E y_E -y_E y_E; -z_offset -z_offset -z_offset - ↙
```

```
z_offset];
m_curr_pos = [0 0 0 0; 0 0 0 0; zCM_t(1) zCM_t(1) zCM_t(1) zCM_t(1)] + ↙
Rx*Ry*Rz*m_pos_0; %first term is the position of CM

gFR_t(1) = m_curr_pos(3, 1);
gFL_t(1) = m_curr_pos(3, 2);
gAR_t(1) = m_curr_pos(3, 3);
gAL_t(1) = m_curr_pos(3, 4);

IFR(1) = m_eff*g*geq^(ng)/Cmag;
IFL(1) = m_eff*g*geq^(ng)/Cmag;
IAR(1) = m_eff*g*geq^(ng)/Cmag;
IAL(1) = m_eff*g*geq^(ng)/Cmag;

FFR(1) = m_eff*g;
FFL(1) = m_eff*g;
FAR(1) = m_eff*g;
FAL(1) = m_eff*g;


%%%%%%%  CONTROL LAW  %%%%%%%%

for i = 1:tn-1

    %update accumulators
    accFR(i+1) = accFR(i) + (g_ref(i)-gFR_t(i))*dt;
    accFL(i+1) = accFL(i) + (g_ref(i)-gFL_t(i))*dt;
    accAR(i+1) = accAR(i) + (g_ref(i)-gAR_t(i))*dt;
    accAL(i+1) = accAL(i) + (g_ref(i)-gAL_t(i))*dt;

    %control laws

    IFR(i+1) = m_eff*g*((-gFR_t(i))^(ng)/Cmag)*(1 + Kp*(g_ref(i)-gFR_t(i))^(e_pow) ↙
+Kd*(0-dgFR_dt(i)) + Ki*accFR(i+1));
    if (IFR(i+1) < I_trd)
        IFR(i+1) = 0;
    end
    FFR(i+1) = IFR(i+1)/((-gFR_t(i))^(ng)/Cmag);

    IFL(i+1) = m_eff*g*((-gFL_t(i))^(ng)/Cmag)*(1 + Kp*(g_ref(i)-gFL_t(i))^(e_pow) ↙
+Kd*(0-dgFL_dt(i))+ Ki*accFL(i+1));
    if (IFL(i+1) < I_trd)
        IFL(i+1) = 0;
    end
    FFL(i+1) = IFL(i+1)/((-gFL_t(i))^(ng)/Cmag);

    IAR(i+1) = m_eff*g*((-gAR_t(i))^(ng)/Cmag)*(1 + Kp*(g_ref(i)-gAR_t(i))^(e_pow) ↙
+Kd*(0-dgAR_dt(i))+ Ki*accAR(i+1));
    if (IAR(i+1) < I_trd)
```

```
        IAR(i+1) = 0;
    end
    FAR(i+1) = IAR(i+1)/((-gAR_t(i))^(ng)/Cmag);

    IAL(i+1) = m_eff*g*((-gAL_t(i))^(ng)/Cmag)*(1 + Kp*(g_ref(i)-gAL_t(i))^(e_pow) ↙
+Kd*(0-dgAL_dt(i))+ Ki*accAL(i+1));
    if (IAL(i+1) < I_trd)
        IAL(i+1) = 0;
    end
    FAL(i+1) = IAL(i+1)/((-gAL_t(i))^(ng)/Cmag);


    %CM kinematics rotation
    ddalphaX_ddt(i+1) = 1/I_vx *(-FFR(i+1)*y_E + FFL(i+1) * y_E - FAR(i+1) * y_E +↙
FAL(i+1) * y_E);
    dalphaX_dt(i+1) = dalphaX_dt(i) + ddalphaX_ddt (i+1) * dt;
    alphaX_t(i+1) = alphaX_t(i) + dalphaX_dt (i+1) * dt;

    ddalphaY_ddt(i+1) = 1/I_vy *(-FFR(i+1)*x_E - FFL(i+1) * x_E + FAR(i+1) * x_E +↙
FAL(i+1) * x_E);
    dalphaY_dt(i+1) = dalphaY_dt(i) + ddalphaY_ddt (i+1) * dt;
    alphaY_t(i+1) = alphaY_t(i) + dalphaY_dt (i+1) * dt;


    %CM kinematics translation
    ddzCM_ddt(i+1) = (-4*m_eff*g + FAL(i+1) + FAR(i+1) + FFR(i+1) + FFL(i+1))/↙
(4*m_eff);
    dzCM_dt(i+1) = dzCM_dt(i) + ddzCM_ddt(i+1) * dt;
    zCM_t(i+1) = zCM_t(i)+dzCM_dt(i+1) * dt;


    % %magnet kinematics; disregard or account for arm bending
    %
    % ddgFR_ddt(i+1) = ddzCM_ddt(i+1)- sin(alphaX_t(i+1))*y_E*ddalphaX_ddt(i+1) - sin↙
(alphaY_t(i+1))*x_E*ddalphaY_ddt(i+1);
    % dgFR_dt(i+1) = dgFR_dt(i) + ddgFR_ddt(i+1) * dt;
    %
    %
    % ddgFL_ddt(i+1) = ddzCM_ddt(i+1) + sin(alphaX_t(i+1))*y_E*ddalphaX_ddt(i+1) -↙
sin(alphaY_t(i+1))*x_E*ddalphaY_ddt(i+1);
    % dgFL_dt(i+1) = dgFL_dt(i) + ddgFL_ddt(i+1) * dt;
    %
    %
    % ddgAR_ddt(i+1) = ddzCM_ddt(i+1) - sin(alphaX_t(i+1))*y_E*ddalphaX_ddt(i+1) +↙
sin(alphaY_t(i+1))*x_E*ddalphaY_ddt(i+1);
    % dgAR_dt(i+1) = dgAR_dt(i) + ddgAR_ddt(i+1) * dt;
    %
    %
    % ddgAL_ddt(i+1) = ddzCM_ddt(i+1) + sin(alphaX_t(i+1))*y_E*ddalphaX_ddt(i+1) +↙
```

```
sin(alphaY_t(i+1))*x_E*ddalphaY_ddt(i+1);
    % dgAL_dt(i+1) = dgAL_dt(i) + ddgAL_ddt(i+1) * dt;
    %

    %%%%%%%  POSITION FROM ROTATION  %%%%%%%%

    Rx = [1 0 0; 0 cos(alphaX_t(i+1)) -sin(alphaX_t(i+1)); 0 sin(alphaX_t(i+1)) cos ↙
(alphaY_t(i+1))];
    Ry = [cos(alphaY_t(i+1)) 0 sin(alphaY_t(i+1)); 0 1 0; -sin(alphaY_t(i+1)) 0 cos ↙
(alphaY_t(i+1))];
    Rz = [cos(alphaZ_t(i+1)) -sin(alphaZ_t(i+1)) 0; sin(alphaZ_t(i+1)) cos(alphaZ_t ↙
(i+1)) 0; 0 0 1];

    m_pos_0 = [x_E x_E -x_E -x_E; -y_E y_E -y_E y_E; -z_offset -z_offset -z_offset - ↙
z_offset];
    m_curr_pos = [0 0 0 0; 0 0 0 0; zCM_t(i+1) zCM_t(i+1) zCM_t(i+1) zCM_t(i+1)] + ↙
Rx*Ry*Rz*m_pos_0;

    gFR_t(i+1) = m_curr_pos(3, 1);
    gFL_t(i+1) = m_curr_pos(3, 2);
    gAR_t(i+1) = m_curr_pos(3, 3);
    gAL_t(i+1) = m_curr_pos(3, 4);

    dgFR_dt(i+1) = (gFR_t(i+1) - gFR_t(i))/dt;
    dgFL_dt(i+1) = (gFL_t(i+1) - gFL_t(i))/dt;
    dgAR_dt(i+1) = (gAR_t(i+1) - gAR_t(i))/dt;
    dgAL_dt(i+1) = (gAL_t(i+1) - gAL_t(i))/dt;

    ddgFR_ddt(i+1) = (dgFR_dt(i+1) - dgFR_dt(i))/dt;
    ddgFL_ddt(i+1) = (dgFL_dt(i+1) - dgFL_dt(i))/dt;
    ddgAR_ddt(i+1) = (dgAR_dt(i+1) - dgAR_dt(i))/dt;
    ddgAL_ddt(i+1) = (dgAL_dt(i+1) - dgAL_dt(i))/dt;

end

%%%%%%%%  PLOTTING  %%%%%%%%
figure

subplot(2,3,1)
plot (t, gFR_t, 'r-')
hold on
plot (t, gFL_t, 'g-');
hold on
plot (t, gAR_t, 'b-');
hold on
plot (t, gAL_t, 'm-');
hold on
plot (t, g_ref, 'r--')
hold on
```

```
plot (t, zeros(size(t)), 'r--')
xlabel('Time [s]')
ylabel('Gap [m]')
hold off
legend('FR','FL', 'AR', 'AL')

subplot(2,3,2)
plot (t, FFR, 'r-')
hold on
plot (t, FFL, 'g-');
hold on
plot (t, FAR, 'b-');
hold on
plot (t, FAL, 'm-');
xlabel('Time [s]')
ylabel('Force [N]')
hold off
legend('FR','FL', 'AR', 'AL')

subplot(2,3,3)
plot (t, 180/3.1415*alphaX_t, 'r-')
hold on
plot (t, 180/3.1415*alphaY_t, 'g-');
hold on
plot (t, 180/3.1415*alphaZ_t, 'b-');
hold on
plot (t, zeros(size(t)), '--')
hold on
plot (t, 5*ones(size(t)), 'r--')
hold on
plot (t, -5*ones(size(t)), 'r--')
xlabel('Time [s]')
ylabel('Rotation Angles [deg]')
hold off
legend('alpha X','alpha Y', 'alpha Z')

% subplot(2,2,4)
% plot (t, IFR, 'r-')
% hold on
% plot (t, IFL, 'g-');
% hold on
% plot (t, IAR, 'b-');
% hold on
% plot (t, IAL, 'm-');
% xlabel('Time [s]')
% ylabel('Current [A]')
% hold off
% legend('FR','FL', 'AR', 'AL')
```

```
subplot(2,3,5)
plot (t, dgFR_dt, 'r-')
hold on
plot (t, dgFL_dt, 'g-');
hold on
plot (t, dgAR_dt, 'b-');
hold on
plot (t, dgAL_dt, 'm-');
hold on
plot (t, zeros(size(t)), 'r--')
xlabel('Time [s]')
ylabel('Gap Velocity [m/s]')
hold off
legend('FR','FL', 'AR', 'AL')

subplot(2,3,6)
plot (t, ddgFR_ddt, 'r-')
hold on
plot (t, ddgFL_ddt, 'g-');
hold on
plot (t, ddgAR_ddt, 'b-');
hold on
plot (t, ddgAL_ddt, 'm-');
hold on
plot (t, zeros(size(t)), 'r--')
xlabel('Time [s]')
ylabel('Gap Acceleration [m/s2]')
hold off
legend('FR','FL', 'AR', 'AL')
```