```
%% 3-D example with polynomial trajectory
% 1. Prameter setting
dim = 3;
knots = [0 2 4 7]; % knots
order = 8; % polynomial order
optimTarget = 'end-derivative'; % 'poly-coeff' or 'end-derivative'
maxConti = 4; % maximally imposed continuity between segment
objWeights = [0 1 1];  % 1 2 3 th order derivative
pTraj = PolyTrajGen(knots,order,optimTarget,dim,maxConti); % create object

% 2. Pin
% 2.1 FixPin
ts = [0 2 4 7]; % knots
Xs = [0 2 5 7 ; ...
        0 -1 3 -5 ; ...
        0 2 4 5]; % waypoints

Xdot = [ 0 0 0]'; % initial velocity

Xddot = [ 0 0 0]'; % initial acceleration

% Order 0 pin (waypoints)
for m = 1:size(Xs,2)
    pin = struct('t',ts(m),'d',0,'X',Xs(:,m));
    pTraj.addPin(pin);
end
% Order 1 pin
pin = struct('t',ts(1),'d',1,'X',Xdot);
pTraj.addPin(pin);
% Order 2 pin
pin = struct('t',ts(1),'d',2,'X',Xddot);
pTraj.addPin(pin);

% 2.2 LoosePin
passCube = [3 4.2 ; -3 -2 ; 1 2];
pin = struct('t',3,'d',0,'X',passCube);
pTraj.addPin(pin);

% 3. Solve
% set the objective function for penalizing the derivatives
pTraj.setDerivativeObj(objWeights);
pTraj.solve;

% 4. evaluate
t_eval = 3; % evluation time
d_eval = 1; % which order to be evaluated for derivative (0,1,2,..)
xdot_eval = pTraj.eval(t_eval,d_eval);
```