```
classdef  FunctionHolder
    properties
        a1=1;a2=0.5;a3=0.5;a4=0.2;
        realpathPoints=[-0.3 0.3;0.6 0.6;0.5 0.5];  %position of end effector
        OrientZ=[0 0];                              %Orientation of end effector in rad
    end
    methods
        function R=Rotationmatrix(~,alpha,theta)
             R=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha);
                sind(theta) cosd(theta)*cosd(alpha) -cosd(theta)*sind(alpha);
                0 sind(alpha) cosd(alpha)];
        end
        function T=TransformationMatrix(a,alpha,theta,d)
            T=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta);
                sind(theta) cosd(theta)*cosd(alpha) -cosd(theta)*sind(alpha) a*sind(theta);
                0 sind(alpha) cosd(alpha) d;0 0 0 1];
        end
        function I=InverseKinematics(~,T,a1,a2,a3)
            theta3=acos((T(1,4)^2+T(2,4)^2-a2^2-a3^2)/(2*a2*a3));
            psi=asin((a2+a3*cos(theta3))/sqrt((a2+a3*cos(theta3))^2+(a3*sin(theta3))^2));
            theta1=psi-asin(T(1,4)/sqrt((a2+a3*cos(theta3))^2+(a3*sin(theta3))^2));
            d2=T(3,4)-a1;
            theta4=acos(T(1,1)-theta1-theta3);
            I=[theta1 d2 theta3 theta4];
        end
        function X = CoefficientsOfCubic(~,t1,t2,th1,th2,thd1,thd2)   %for path planning
            syms a0 a1 a2 a3
            eqn1 = a0+t1*a1 +t1^2*a2 + t1^3*a3 == th1;
            eqn2 = a1 +2*t1*a2 + 3*t1^2*a3 == thd1;
            eqn3 =a0+t2*a1 +t2^2*a2 + t2^3*a3 == th2;
            eqn4 = a1 +2*t2*a2 + 3*t2^2*a3 == thd2 ;
            [A,B] = equationsToMatrix([eqn1, eqn2, eqn3, eqn4], [a0, a1, a2 ,a3]);
            X = linsolve(A,B);
        end
    end
end
```