Code for ikinelbow:

```
% ikinelbow computes the inverse kinematics for the 7 DoF Baxter robot
% constrained to be 6 DoF with theta3=0
%
%     theta = ikinelbow(a, d, Tw_tool, LR, UD, NF) Calculates a vector of
%     theta (joint angle) values given specification of the robot geometry
%     and a desired end position and orientation
%
%     a = non zero a DH parameters in sequential order starting with a1
%     i.e. (a1, a2)
%   d = non zero d DH parameters in sequential order starting with d1
%   i.e. (d1, d4, d6)
%   Tw_tool  = Homogeneous transformation matrix which specifies a desired
%   location and orientation of the end effector from the tool frame with
%   respect to the world frame
%   LR = 1 if specifying the "lefty" solution and 0 if specifying the
%   "righty" solution
%   UD = 1 if specifying the "elbow up" solution and 0 if specifying the
%   "elbow down" solution
%   NF = 1 if specifying the "flip" solution and 0 if specifying the
%   "no flip" solution
%
%     Ryan Dalby
%     ME EN 6220
%     11/17/2020
function [theta] = ikinelbow(a,d,Tw_tool,LR,UD,NF)
theta = zeros(1,6);
R_w_tool = Tw_tool(1:3,1:3);
d_w_wtool = Tw_tool(1:3,4);
a1 = a(1);
a2 = a(2);
d1 = d(1);
d4 = d(2);
d6 = d(3);

%
% Information for transformation between world frame and frame 0 and also
%
R_w_0 = [sqrt(2)/2 sqrt(2)/2 0;...
         -sqrt(2)/2 sqrt(2)/2 0;...
         0 0 1];
L = 221; % mm
h = 22; % mm
```

```
H = 1104; % mm
d_w_w0 = [L; h; H];

%
% Information for transformation between frame 6 and tool frame
%
R_6_tool = [1 0 0;...
         0 1 0;...
         0 0 1];
d_6_4tool = [0; 0; d6];

%
% Extract necessary information to use solution used in ikinebaxter
%
R_0_6 = transpose(R_w_0)*R_w_tool*transpose(R_6_tool);
d_0_0tool = transpose(R_w_0)*(d_w_wtool-d_w_w0);
d_0_4tool = R_0_6 * d_6_4tool;
d_0_04 = d_0_0tool - d_0_4tool;

%
% Workspace determination
%
% Assume that d4=374.29mm, a2=370.82mm, and a1=69mm
d_0_06 = d_0_04;
d_0_06_mag = norm(d_0_06);
inner_primary_workspace_diameter = a1+a2-d4;
outer_primary_workspace_diameter = a2+d4-a1;
% Make sure we are in primary workspace or exit function
if (d_0_06_mag < inner_primary_workspace_diameter) || (d_0_06_mag >
outer_primary_workspace_diameter)
    disp('Outside of primary workspace');
    return;
end


%
% Regional structure inverse kinematics
%

% theta 1 determination
theta(1) = atan2(d_0_04(2), d_0_04(1));
if (~LR)
    theta(1) = theta(1) + pi;
```

```
end

% theta 3 and theta 2 determination
d_0_01 = [a1*cos(theta(1)); a1*sin(theta(1)); d1];
R_1_0 = [cos(theta(1)) sin(theta(1)) 0;...
         0 0 -1;...
         -sin(theta(1)) cos(theta(1)) 0];
d_1_14 = R_1_0 * (d_0_04-d_0_01);

r2 = d_1_14(1)^2 + d_1_14(2)^2;

alpha = 2*atan2(sqrt((a2+d4)^2 - (r2)) , sqrt((r2) - (a2-d4)^2));
if(~UD)
    alpha = alpha * -1;
end
theta(3) = alpha - pi/2;

psi = atan2((d4*sin(alpha)),(a2+d4*cos(alpha)));
phi = atan2(d_1_14(2),d_1_14(1));
theta(2) = phi - psi;

%
% Orientation structure inverse kinematics
%
R_0_1 = transpose(R_1_0);
R_1_2 = [cos(theta(2)) -sin(theta(2)) 0;...
         sin(theta(2)) cos(theta(2)) 0;...
         0 0 1];
R_2_3 = [cos(theta(3)) 0 -sin(theta(3));...
         sin(theta(3)) 0 cos(theta(3));...
         0 -1 0];
R_0_3 = R_0_1*R_1_2*R_2_3;

R_3_6 = transpose(R_0_3) * R_0_6;

% theta 4 determination
theta(4) = atan2(-R_3_6(2,3),-R_3_6(1,3));
if(NF)
    theta(4) = theta(4) + pi;
end

% theta 5 and theta 6 determination
R_3_4 = [cos(theta(4)) 0 sin(theta(4));...
```

```
         sin(theta(4)) 0 -cos(theta(4));...
         0 1 0];

R_4_6 = transpose(R_3_4) * R_3_6;

theta(5) = atan2(-R_4_6(1,3),R_4_6(2,3));

theta(6) = atan2(-R_4_6(3,1),-R_4_6(3,2));
end
```