Introduction to Robotics

Lab 6

Shaaf Farooque, Mysha Zulfiqar

```
%Task 6.1
syms x y z phi

l_1=10;
l_2=4.5;
l_3=10.7;
l_4=10.5;
l_5=9.5;

theta1_1 = mod(atan2(y,x) + pi, 2*pi) - pi;
theta1_2 = mod(atan2(y,x) +pi + pi, 2*pi) - pi;

r = sqrt(x^2 + y^2);
s = z - (l_1+l_2);

r_ = r - l_5*cos(phi);
s_ = s - l_5*sin(phi);

D = ((r_*r_) + (s_*s_) - l_3^2 - l_4^2)/(2*l_3*l_4);

num = sqrt(1-D*D);

theta3_1 = mod(atan2((num),D) + pi, 2*pi) - pi;
theta3_2 = mod(atan2(-(num),D) + pi, 2*pi) - pi;

theta2_1 = mod(atan2(s_,r_) - atan2(l_4*sin(theta3_1), l_3 + l_4*cos(theta3_1)) +
pi, 2*pi) - pi;
theta2_2 = mod(atan2(s_,r_) - atan2(l_4*sin(theta3_2), l_3 + l_4*cos(theta3_2)) +
pi, 2*pi) - pi;

theta4_1 = mod((phi - theta2_1 - theta3_1) + pi, 2*pi) - pi;
theta4_2 = mod((phi - theta2_2 - theta3_2)  + pi, 2*pi) - pi;

angles(1,:) = [theta1_1+pi/2 -theta2_1+pi/2  -theta3_1 -theta4_1];
angles(2,:) = [theta1_1+pi/2 -theta2_2+pi/2  -theta3_2 -theta4_2];
angles(3,:) = [theta1_2+pi/2 -(-theta2_1+pi)+pi/2 -theta3_1 theta4_1];
angles(4,:) = [theta1_2+pi/2 -(-theta2_2+pi)+pi/2 theta3_2 theta4_2];
angles
```

```
angles =
```

$$\begin{pmatrix} \sigma_2 & \frac{3\pi}{2} - \sigma_5 & \pi - \sigma_7 & \pi - \sigma_3 \\ \sigma_2 & \frac{3\pi}{2} - \sigma_6 & \pi - \sigma_8 & \pi - \sigma_4 \\ \sigma_1 & -\frac{3\pi}{2} + \sigma_5 & \pi - \sigma_7 & -\pi + \sigma_3 \\ \sigma_1 & -\frac{3\pi}{2} + \sigma_6 & -\pi + \sigma_8 & -\pi + \sigma_4 \end{pmatrix}$$

where

$$\sigma_1 = -\frac{\pi}{2} + (2\pi + \mathrm{atan2}(y, x) \bmod 2\pi)$$

$$\sigma_2 = -\frac{\pi}{2} + (\pi + \mathrm{atan2}(y, x) \bmod 2\pi)$$

$$\sigma_3 = \phi + 3\pi - \sigma_5 - \sigma_7 \bmod 2\pi$$

$$\sigma_4 = \phi + 3\pi - \sigma_6 - \sigma_8 \bmod 2\pi$$

$$\sigma_5 = \pi - \mathrm{angle}\left(\frac{107}{10} - \frac{21\cos(\sigma_7)}{2} - \frac{21\sin(\sigma_7)\,\mathrm{i}}{2}\right) + \sigma_9 \bmod 2\pi$$

$$\sigma_6 = \pi - \mathrm{angle}\left(\frac{107}{10} - \frac{21\cos(\sigma_8)}{2} - \frac{21\sin(\sigma_8)\,\mathrm{i}}{2}\right) + \sigma_9 \bmod 2\pi$$

$$\sigma_7 = \pi + \mathrm{atan2}\left(\sqrt{1 - {\sigma_{10}}^2}, \sigma_{10}\right) \bmod 2\pi$$

$$\sigma_8 = \pi + \mathrm{atan2}\left(-\sqrt{1 - {\sigma_{10}}^2}, \sigma_{10}\right) \bmod 2\pi$$

$$\sigma_9 = \mathrm{atan2}\left(z - \frac{19\sin(\phi)}{2} - \frac{29}{2}, \sqrt{x^2 + y^2} - \frac{19\cos(\phi)}{2}\right)$$

$$\sigma_{10} = \frac{10\left(\frac{19\cos(\phi)}{2} - \sqrt{x^2 + y^2}\right)^2}{2247} + \frac{10\left(\frac{19\sin(\phi)}{2} - z + \frac{29}{2}\right)^2}{2247} - \frac{11237}{11235}$$

```
%4 Solutions exist.
```

```
%Task 6.2
test1 = [-13, -13, 4]; %third quadrant
```

```
angles1 = findJointAngles(test1(1),test1(2),test1(3),0)
```

```
angles1 = 4×4
   -0.7854    3.2931   -1.7297    0.0074
   -0.7854    1.5856    1.7297   -1.7444
    2.3562   -3.2931   -1.7297   -0.0074
    2.3562   -1.5856   -1.7297    1.7444
```

```
disp(['x = ',num2str(test1(1)), ' y = ', num2str(test1(2)),' z =
',num2str(test1(3))])
```

```
x = -13 y = -13 z = 4
```

```
[x1, y1, z1, ~] = pincherFK(angles1(1,:))
```

```
x1 = -13
y1 = -13
z1 = 4
```

```
test1 = [13, -5, 4]; %fourth quadrant
angles1 = findJointAngles(test1(1),test1(2),test1(3),0)
```

```
angles1 = 4×4
    1.2036    3.7310   -2.0067   -0.1535
    1.2036    1.7539    2.0067   -2.1898
    4.3452   -3.7310   -2.0067    0.1535
    4.3452   -1.7539   -2.0067    2.1898
```

```
disp(['x = ',num2str(test1(1)), ' y = ', num2str(test1(2)),' z =
',num2str(test1(3))])
```

```
x = 13 y = -5 z = 4
```

```
[x2, y2, z2, ~] = pincherFK(angles1(1,:))
```

```
x2 = 13.0000
y2 = -5
z2 = 4.0000
```

```
test1 = [10, 12, 6]; %first quadrant
angles1 = findJointAngles(test1(1),test1(2),test1(3),0)
```

```
angles1 = 4×4
    2.4469    3.5550   -2.1082    0.1240
    2.4469    1.4800    2.1082   -2.0174
   -0.6947   -3.5550   -2.1082   -0.1240
   -0.6947   -1.4800   -2.1082    2.0174
```

```
disp(['x = ',num2str(test1(1)), ' y = ', num2str(test1(2)),' z =
',num2str(test1(3))])
```

```
x = 10 y = 12 z = 6
```

```
[x3, y3, z3, ~] = pincherFK(angles1(1,:))
```

```
x3 = 10
y3 = 12.0000
```

```
z3 = 6.0000
```

```
test1 = [-5, 4, 10]; %second quadrant
angles1 = findJointAngles(test1(1),test1(2),test1(3),0)
```

```
angles1 = 4×4
    4.0376   -1.2639   -2.6207   -0.8278
    4.0376    2.4694    2.6207    2.7639
    0.8961    1.2639   -2.6207    0.8278
    0.8961   -2.4694   -2.6207   -2.7639
```

```
disp(['x = ',num2str(test1(1)), ' y = ', num2str(test1(2)),' z =
',num2str(test1(3))])
```

```
x = -5 y = 4 z = 10
```

```
[x4, y4, z4, ~] = pincherFK(angles1(1,:))
```

```
x4 = -5.0000
y4 = 4.0000
z4 = 10.0000
```

```
%Task 6.3
%Function in the last section
```

```
arb = Arbotix('port', 'COM4', 'nservos', 5)
```

```
Warning: instrfind will be removed in a future release. For serialport, tcpclient, tcpserver, udpport,
visadev, aardvark, and ni845x objects, use serialportfind, tcpclientfind, tcpserverfind, udpportfind,
visadevfind, aardvarkfind, and ni845xfind instead.
serPort COM4is in use.   Closing it.
Warning: serial will be removed in a future release. Use serialport instead.
If you are using serial with icdevice, continue using serial in this MATLAB release.
i = 4

arb =
Arbotix chain on serPort COM4 (open)
 5 servos in chain
```

```
%Task 6.4
%5 random points (x, y, z, phi). Unit : cm
point1 = [12, 7, 5, 0];
point2 = [-10, 8, 10, 0];
point3 = [-13, -13, 4, 0];
point4 = [8, -12, 6, 0];
point5 = [15, 0, 7, 0];

angles1 = findOptimalSolution(point1, arb.getpos())
```

```
angles1 = 1×4
    2.0989    0.7254    1.7125    0.7037
```

```
angles2 = findOptimalSolution(point2, arb.getpos())
```

```
angles2 = 1×4
    0.8961   -0.3444   -1.7305   -1.0667
```

```
angles3 = findOptimalSolution(point3, arb.getpos())
```

```
angles3 = 1×4
   -0.7854    1.1118    1.0374    0.9924
```

```
angles4 = findOptimalSolution(point4, arb.getpos())
```

```
angles4 = 1×4
    0.5880    0.6912    1.6409    0.8095
```

```
angles5 = findOptimalSolution(point5, arb.getpos())
```

```
angles5 = 1×4
    1.5708    0.6716    1.5518    0.9182
```

```
% arb.setpos([angles1 0], [60 60 60 60 60])
% arb.setpos([angles2 0], [60 60 60 60 60])
% arb.setpos([angles3 0], [60 60 60 60 60])
% arb.setpos([angles4 0], [60 60 60 60 60])
arb.setpos([angles5 0], [60 60 60 60 60])
```

Desired Position(cm)	Actual Position(cm)	Absolute Euclidean Error(cm)
(12,7,5)	(11,8,3)	2.4495
(-10,8,10)	(-9.5,9,8)	2.2913
(-13,-13,4)	(-12,-12,2.5)	2.0616
(8,-12,6)	(9,-10,3.5)	3.3541
(15,0,7)	(13.5,0,4)	3.3541

```
error1 = euclideanError(point1(1:3), [11 8 3])
```

```
error1 = 2.4495
```

```
error2 = euclideanError(point2(1:3), [-9.5 9 8])
```

```
error2 = 2.2913
```

```
error3 = euclideanError(point3(1:3), [-12 -12 2.5])
```

```
error3 = 2.0616
```

```
error4 = euclideanError(point4(1:3), [9 -10 3.5])
```

```
error4 = 3.3541
```

```
error5 = euclideanError(point5(1:3), [13.5 0 4])
```

```
error5 = 3.3541
```

```

mean([error1,error2, error3,error4,error5])
```

```
ans = 2.7021
```

Mean euclidean error is 2.7cm.

Possible sources of error:

One significant source is kinematic modeling error, the solutions from findJointAngles and findOptimalSolution are ideal solutions. In practice, there may be some error due to frame misalignment. Moreover, motor control limitations, including overshoot or friction, can make the robot miss target positions.

```
function angles = findJointAngles(x,y,z,phi)
    angles = zeros(4,4);
    l_1=10;
    l_2=4.5;
    l_3=10.7;
    l_4=10.5;
    l_5=9.5;

    theta1_1 = mod(atan2(y,x) + pi, 2*pi) - pi;
    theta1_2 = mod(atan2(y,x) +pi + pi, 2*pi) - pi;

    r = sqrt(x^2 + y^2);
    s = z - (l_1+l_2);

    r_ = r - l_5*cos(phi);
    s_ = s - l_5*sin(phi);

    D = ((r_*r_) + (s_*s_) - l_3^2 - l_4^2)/(2*l_3*l_4);

    num = sqrt(1-D*D);

    theta3_1 = mod(atan2((num),D) + pi, 2*pi) - pi;
    theta3_2 = mod(atan2(-(num),D) + pi, 2*pi) - pi;

    theta2_1 = mod(atan2(s_,r_) - atan2(l_4*sin(theta3_1), l_3 + l_4*cos(theta3_1))
+ pi, 2*pi) - pi;
    theta2_2 = mod(atan2(s_,r_) - atan2(l_4*sin(theta3_2), l_3 + l_4*cos(theta3_2))
+ pi, 2*pi) - pi;

    theta4_1 = mod((phi - theta2_1 - theta3_1) + pi, 2*pi) - pi;
    theta4_2 = mod((phi - theta2_2 - theta3_2)  + pi, 2*pi) - pi;

    angles(1,:) = [theta1_1+pi/2 -theta2_1+pi/2  -theta3_1 -theta4_1];
    angles(2,:) = [theta1_1+pi/2 -theta2_2+pi/2  -theta3_2 -theta4_2];
    angles(3,:) = [theta1_2+pi/2 -(-theta2_1+pi)+pi/2 -theta3_1 theta4_1];
    angles(4,:) = [theta1_2+pi/2 -(-theta2_2+pi)+pi/2 theta3_2 theta4_2];
    % angles = mod(angles + pi, 2*pi) - pi;

end
```

```matlab
function solution = findOptimalSolution(desiredPos, currentPos)
     x = desiredPos(1);
     y = desiredPos(2);
     z = desiredPos(3);
     phi = -pi/2;


    solutions = findJointAngles(x, y, z, phi);

    B = cellfun(@checkJointLimits, num2cell(solutions, 2), 'UniformOutput', false);
    B = cell2mat(B);  % B is now n_solutions × 4 logical matrix

    % Identify solutions where ALL joints are within limits
    valid_solutions_mask = all(B, 2);  % True only if all joints in a row are true

    % Keep only valid solutions (original joint angles)
    valid_solutions = solutions(valid_solutions_mask, :);

    % If no valid solutions, return empty or handle error
    if isempty(valid_solutions)
        solution = [];  % Or throw an error/warning as needed
        return;
    end

    % Absolute errors for all joints of all valid solutions
    delta = abs(valid_solutions - currentPos(1, 1:4));

    s = sum(delta, 2);

    % Find the solution with the minimum total error
    [~, idx] = min(s);
    solution = valid_solutions(idx, :); % Return the actual joint angles
end

function isValid = checkJointLimits(jointAngles)
    % joint angle limits in radians (-150 to 150 degrees)
    thetaLimits = [-150*pi/180, 150*pi/180]; % Convert degrees to radians

    % Check if all joint angles are within limits
    isValid = all(jointAngles >= thetaLimits(1) & jointAngles <= thetaLimits(2));
end

function vs = skew(v)
    vs = [0 -v(3) v(2);
          v(3) 0 -v(1);
         -v(2) v(1) 0];
end
```

```
function error = euclideanError(point1, point2)
    diff = point1 - point2;
    error = sqrt(sum(diff.^2));
end


function T = exp(S, theta)
    w_skew = skew(S(1:3)/norm(S(1:3)));
    exp_w = eye(3) + w_skew*sin(theta) + w_skew^2*(1-cos(theta));
    G = eye(3)*theta + (1-cos(theta))*w_skew+(theta-sin(theta))*w_skew^2;
    T = [exp_w, G*S(4:6);
         zeros(1,3), 1];
end

function [S1, S2, S3, S4, Tsb] = getTsb()
    l_1=10;
    l_2=4.5;
    l_3=10.7;
    l_4=10.5;
    w1 = [0;0;1];
    q1 = [0;0;l_1];
    S1 = [w1;cross(-w1,q1)];

    w2 = [1;0;0];
    q2 = [0;0;l_1+l_2];
    S2 = [w2;cross(-w2,q2)];

    w3 = [1;0;0];
    q3 = [0;0;l_1+l_2+l_3];
    S3 = [w3;cross(-w3,q3)];

    w4 = [1;0;0];
    q4 = [0;0;l_1+l_2+l_3+l_4];
    S4 = [w4;cross(-w4,q4)];

    syms theta_1 theta_2 theta_3 theta_4
    Tsb = exp(S1,theta_1)*exp(S2,theta_2)*exp(S3,theta_3)*exp(S4,theta_4);
end


function zeroConfig= getZeroConf()
    l_1=10;
    l_2=4.5;
    l_3=10.7;
    l_4=10.5;
    l_5=9.5;
    zeroConfig = [1 0 0 0;
         0 1 0 0;
         0 0 1 l_1 + l_2 + l_3 + l_4 + l_5;
         0 0 0 1];
```

```
end
function [x,y,z,R] = pincherFK(jointAngles)
    M = getZeroConf();
    syms theta_1 theta_2 theta_3 theta_4
    [~, ~, ~, ~, Tsb] = getTsb();
    T = double(subs(Tsb,[theta_1 theta_2 theta_3 theta_4],jointAngles(1:4))*M);
    x = T(1,4);
    y = T(2,4);
    z = T(3,4);
    R = T(1:3, 1:3);
end
```