Contents

Define Ai

```
syms q1 q2 q3 L1

A0 = eye(4)

A1 = [cos(q1) 0 sin(q1) 0; ...
    sin(q1) 0 -cos(q1) 0; ...
    0 1 0 L1; ...
    0 0 0 1]

A2 = [cos(q2+pi/2) 0 sin(q2+pi/2) 0; ...
    sin(q2+pi/2) 0 -cos(q2+pi/2) 0; ...
    0 1 0 0; ...
    0 0 0 1]

A3 = [1 0 0 0; ...
    0 1 0 0; ...
    0 0 1 q3; ...
    0 0 0 1]
```

```
A0 =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1


A1 =

[cos(q1), 0,  sin(q1),  0]
[sin(q1), 0, -cos(q1),  0]
[      0, 1,        0, L1]
[      0, 0,        0,  1]


A2 =

[cos(q2 + pi/2), 0,  sin(q2 + pi/2), 0]
[sin(q2 + pi/2), 0, -cos(q2 + pi/2), 0]
[             0, 1,               0, 0]
[             0, 0,               0, 1]


A3 =
```

```
[1, 0, 0,  0]
[0, 1, 0,  0]
[0, 0, 1, q3]
[0, 0, 0,  1]
```

Calculate A

```
A01 = A0*A1
A012 = A01*A2
A0123 = A012*A3
A = A0123;
```

```
A01 =

[cos(q1), 0,  sin(q1),  0]
[sin(q1), 0, -cos(q1),  0]
[      0, 1,        0, L1]
[      0, 0,        0,  1]


A012 =

[cos(q1)*cos(q2 + pi/2),  sin(q1), cos(q1)*sin(q2 + pi/2),  0]
[cos(q2 + pi/2)*sin(q1), -cos(q1), sin(q1)*sin(q2 + pi/2),  0]
[        sin(q2 + pi/2),        0,        -cos(q2 + pi/2), L1]
[                     0,        0,                      0,  1]


A0123 =

[cos(q1)*cos(q2 + pi/2),  sin(q1), cos(q1)*sin(q2 + pi/2), q3*cos(q1)*sin(q2 + pi/2)]
[cos(q2 + pi/2)*sin(q1), -cos(q1), sin(q1)*sin(q2 + pi/2), q3*sin(q1)*sin(q2 + pi/2)]
[        sin(q2 + pi/2),        0,        -cos(q2 + pi/2),    L1 - q3*cos(q2 + pi/2)]
[                     0,        0,                      0,                         1]
```

calculate partial derivatives

```
dx = diff(A(1,4), q1)+diff(A(1,4), q2)+diff(A(1,4), q3)
dy = diff(A(2,4), q1)+diff(A(2,4), q2)+diff(A(2,4), q3)
dz = diff(A(3,4), q1)+diff(A(3,4), q2)+diff(A(3,4), q3)
```

```
dx =

cos(q1)*sin(q2 + pi/2) + q3*cos(q1)*cos(q2 + pi/2) - q3*sin(q1)*sin(q2 + pi/2)


dy =

sin(q1)*sin(q2 + pi/2) + q3*cos(q1)*sin(q2 + pi/2) + q3*cos(q2 + pi/2)*sin(q1)


dz =
```

```
q3*sin(q2 + pi/2) - cos(q2 + pi/2)
```

linear and angular velocities using the O and Z vector from the A Matrix

```
J1 = [cross(A0(1:3,3),(A(1:3,4)-A0(1:3,4))); A0(1:3,3)]
J2 = [cross(A01(1:3,3),(A(1:3,4)-A01(1:3,4))); A01(1:3,3)]
J3 = [A012(1:3,3); [0 0 0]']
J = [J1 J2 J3]
```

```
J1 =

-q3*sin(q1)*sin(q2 + pi/2)
 q3*cos(q1)*sin(q2 + pi/2)
                         0
                         0
                         0
                         1


J2 =

                                q3*cos(q1)*cos(q2 + pi/2)
                                q3*cos(q2 + pi/2)*sin(q1)
q3*sin(q1)^2*sin(q2 + pi/2) + q3*cos(q1)^2*sin(q2 + pi/2)
                                                  sin(q1)
                                                 -cos(q1)
                                                        0


J3 =

cos(q1)*sin(q2 + pi/2)
sin(q1)*sin(q2 + pi/2)
       -cos(q2 + pi/2)
                     0
                     0
                     0


J =

[-q3*sin(q1)*sin(q2 + pi/2),                                 q3*cos(q1)*cos(q2 + pi/2), cos(q1)*sin(q2 + pi/2)]
[ q3*cos(q1)*sin(q2 + pi/2),                                 q3*cos(q2 + pi/2)*sin(q1), sin(q1)*sin(q2 + pi/2)]
[                         0, q3*sin(q1)^2*sin(q2 + pi/2) + q3*cos(q1)^2*sin(q2 + pi/2),        -cos(q2 + pi/2)]
[                         0,                                                   sin(q1),                      0]
[                         0,                                                  -cos(q1),                      0]
[                         1,                                                         0,                      0]
```

all the links are 1 unit and Theta1 and Theta2 is at zero degrees, determine the numerical Jacobian Matrix

```
q1 = 0;
q2 = 0;
q3 = 0;
J = @(q1,q2,q3)([-q3*sin(q1)*sin(q2 + pi/2), q3*cos(q1)*cos(q2 + pi/2), cos(q1)*sin(q2 + pi/2);
q3*cos(q1)*sin(q2 + pi/2), q3*cos(q2 + pi/2)*sin(q1), sin(q1)*sin(q2 + pi/2);
```

```
0, q3*sin(q1)^2*sin(q2 + pi/2) + q3*cos(q1)^2*sin(q2 + pi/2),-cos(q2 + pi/2);
0, sin(q1), 0;
0, -cos(q1), 0;
1,0,0;]);
J(q1,q2,q3)
```

```
ans =

         0         0    1.0000
         0         0         0
         0         0   -0.0000
         0         0         0
         0   -1.0000         0
    1.0000         0         0
```

Determine the inverse Jacobian for the Matrix found in Question 4.

```
pinv(J(q1,q2,q3))
```

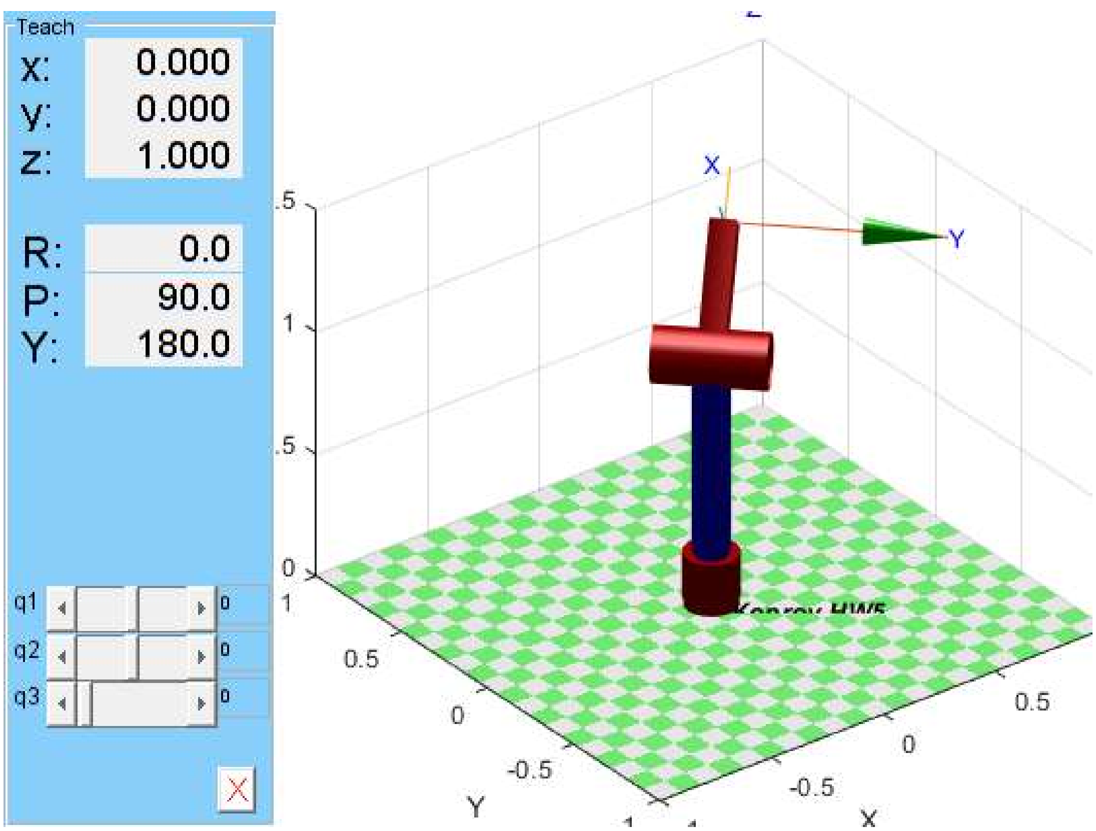
```
ans =

     0     0     0     0     0     1
     0     0     0     0    -1     0
     1     0     0     0     0     0
```

Write a MATLAB script to solve for the inverse kinematics, (q1, q2, q3) to position the robot at 0.25x + 0.25y + 1.354z

```
L1 = 1;
L(1) = Link('alpha', pi/2, 'a', 0, 'd', L1);
L(2) = Link('alpha', pi/2, 'a', 0, 'd', 0, 'offset', pi/2);
L(3) = Link('alpha', 0, 'a', 0, 'theta', 0);
L(3).qlim = [0 1];
R1 = SerialLink(L, 'name', "Koprov HW5");
R1.plotopt = {'workspace' [-1,1,-1,1,0,1.5]};
R1.teach

Xg = 0.25;
Yg = 0.25;
Zg = 1.354;

qk1 = 0;
qk2 = 0;
qk3 = 0;

doLoop = true;
while doLoop
    FK = R1.fkine([qk1 qk2 qk3]);
    xk = FK.t(1);
    yk = FK.t(2);
    zk = FK.t(3);

    J = R1.jacob0([qk1 qk2 qk3]);
    J_inv = pinv(J);
    X = Xg - xk;
```

```
    Y = Yg - yk;
    Z = Zg - zk;
    Xv = [X Y Z 0 0 0]';
    Q = J_inv*Xv;

    qk1 = qk1+Q(1);
    qk2 = qk2+Q(2);
    qk3 = qk3+Q(3);

    R1.plot([qk1 qk2 qk3])


    pos_error_x = xk -Xg;
    pos_error_y = yk -Yg;
    pos_error_z = zk -Zg;
    p_sse = sqrt(pos_error_x^2+pos_error_y^2+pos_error_z^2);

    if p_sse <= 0.001
        doLoop = false;
    end
end
disp([qk1*180/pi, qk2*180/pi, qk3])
```

```
   44.8644   45.0362    0.5003
```



Published with MATLAB® R2022a