INVKIN

Table of Contents

Calcula a cinemática inversa do robô planar RRR tendo como entradas a matriz de transformação do punho para a base, indicando para onde se deseja levar o robô, os valores atuais dos ângulos das juntas, os comprimentos dos ligamentos e os limites de operação das juntas do robô.

Calling Syntax

[near, far, sol] = invkin(wrelb, current, L, thetalim)

I/O Variables

`IN Double Matrix` **wrelb**: Homogeneous Transformation Matrix 4x4

`IN Double Array` **current**: [theta1 theta2 theta3] [degrees degrees degrees]

`IN Double Array` **L**: [l1 l2] [meters meters]

`IN Double Matrix` **thetalim**: [lim1n lim1p; lim2n lim2p; lim3n lim3p] [degrees degrees; degrees degrees; degrees degrees]

`OU Double Array` **near**: [theta1 theta2 theta3] [degrees degrees degrees]

`OU Double Array` **far**: [theta1 theta2 theta3] [degrees degrees degrees]

`OU Double` **sol**: Number of solutions

Example

```
wrelb = [0 -1 0 1;
        1  0 0 0;
        0  0 1 0;
        0  0 0 1]
current = [0 0 0]
L = [0.5 0.3]
thetalim = [ -170 170; -170 170; -170 170]
[near, far, sol] = invkin(wrelb, current, L, thetalim)
near =
```

```
            0     0     0
```

far =

```
            0     0     0
```

sol =

```
            0
```

Hypothesis

RRR planar robot.

Version Control

1.0; Leonardo da Cunha Menegon, Michel Kagan, Vinícius Nardelli; 01/05/2023; First issue.

Function

```
function [near, far, sol] = invkin(wrelb, current, L, thetalim)
```

Validity

```
    arguments
        wrelb (4,4) {functions.mustBeHomTransfR}
        current (1,3) {mustBeNumeric, mustBeReal, mustBeFinite}
        L (1,2) {mustBeNumeric, mustBeReal, mustBeFinite} = [0.5, 0.3]
        thetalim (3, 2) {mustBeNumeric, mustBeReal, mustBeFinite} =
 repmat([-170, 170], 3, 1)
    end
```

Main Calculations

```
    x = wrelb(1, 4);
    y = wrelb(2, 4);
    phi = atan2d(wrelb(2, 1), wrelb(1, 1));
    sol = -1;
    soln = -1;
    solp = -1;

% Espaço de Trabalho Alcançavel
    raio_interno = abs(L(1) - L(2));
    raio_externo = L(1) + L(2);
    distancia = sqrt(x^2 + y^2);
    if (raio_interno > distancia || distancia > raio_externo)
        sol = 0;
        near = [0 0 0];
        far = [0 0 0];
% Cálculo das soluções
    else
```

```
        c2 = (x^2 + y^2 - L(1)^2 - L(2)^2)/(2*L(1)*L(2));
        s2p = sqrt(1 - c2^2);
        s2n = -sqrt(1 - c2^2);
        k1 = L(1) + L(2)*c2;
        k2p = L(2)*s2p;
        k2n = L(2)*s2n;
        gamap = atan2d(k2p, k1);
        gaman = atan2d(k2n, k1);
        theta2p = atan2d(s2p, c2);
        theta2n = atan2d(s2n, c2);
        theta1p = atan2d(y, x) - gamap;
        theta1n = atan2d(y, x) - gaman;
        theta3p = phi - theta1p - theta2p;
        theta3n = phi - theta1n - theta2n;
        vthetap = [theta1p theta2p theta3p];
        vthetan = [theta1n theta2n theta3n];
% Verificação dos limites de operação das juntas
        for i = 1:3
            if thetalim(i, 1) > vthetap(i) || vthetap(i) > thetalim(i,
 2)
                solp = 0;
            end
            if thetalim(i, 1) > vthetan(i) || vthetan(i) > thetalim(i,
 2)
                soln = 0;
            end
        end
% Nenhuma solução
        if solp == 0 && soln == 0
            sol = 0;
            near = [0 0 0];
            far = [0 0 0];
        end
% Uma solução
        if solp == 0 && soln ~=0
            sol = 1;
            near = [theta1n theta2n theta3n];
            far = near;
        end
        if soln == 0 && solp ~=0
            sol = 1;
            near = [theta1p theta2p theta3p];
            far = near;
        end
% Duas soluções
        if solp ~= 0 && soln ~= 0
            diffp = abs(theta1p - current(1)) + abs(theta2p -
 current(2)) + abs(theta3p - current(3));
            diffn = abs(theta1n - current(1)) + abs(theta2n -
 current(2)) + abs(theta3n - current(3));
            if diffp < diffn
                near = [theta1p theta2p theta3p];
                far = [theta1n theta2n theta3n];
            else
```

```
                far = [theta1p theta2p theta3p];
                near = [theta1n theta2n theta3n];
            end
            sol = 2;
        end
    end
```

Output Data

```
    [near, far, sol];

end
```

Published with MATLAB® R2020a