INVKIN

Table of Contents

Função que calcula a cinemática inversa do robô planar 3R, que recebe a matriz de transformação do sistema do punho com relação a base, sua posição atual e limites de operação para devolver uma flag se foi possível chegar em soluções e, se sim, idem as devolve.

Calling Syntax

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

I/O Variables

`IN Double Matrix` **wrelb**: *W relative to B* Homogeneous Transformation Matrix 4x4

`IN Double Array` **current**: *Current angles* [$\theta_1 \theta_2 \theta_3$] [degrees degrees degrees]

`IN Double Array` **L**: *Ligaments length* [$L_1 L_2$] [meters meters]

`IN Double Matrix` **thetalim**: *Limite operation for N angles [2xN]* [$\theta_{1-Superior}$ ... $\theta_{N-Superior}$; $\theta_{1-Inferior}$ ... $\theta_{N-Inferior}$] [degrees]

`OU Double Array` **near**: *Nearest solution* [$\theta_1 \theta_2 \theta_3$] [degrees degrees degrees]

`OU Double Array` **far**: *Further solution* [$\theta_1 \theta_2 \theta_3$] [degrees degrees degrees]

`OU Bool` **sol**: *Solution* sol=0: No possible solution; sol=1: There was a solution

Example

```
wrelb = utoi([0.5 0.3 45]);
current = [0.3 0.5 0];
L = [0.5 0.3];
thetalim = [170 170 170; -170 -170 -170];
[near,far,sol]=invkin(wrelb,current,L,thetalim)
```

Hypothesis

RRR planar robot.

Limitations

A "Forma do usuário" é específica para o exercício de simulação e não tem validade para qualquer configuração de robô.

Version Control

1.0; Grupo 04; 2025/04/03 ; First issue.

Group Members

- Guilherme Fortunato Miranda

```
13683786
```

- João Pedro Dionizio Calazans

```
13673086
```

Function

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

Validity

It works in some years (not odds)

Main Calculations

```
uwrelb = itou(wrelb); % User wrelb
uwrelb(3) = uwrelb(3)*pi/180;

cos_2 = (uwrelb(1)^2+uwrelb(2)^2-L(1)^2-L(2)^2)/(2*L(1)*L(2));
sen_2 = real(sqrt(1-cos_2^2));
thetas = zeros(3);
thetas(2) = atan2(sen_2,cos_2);
k = zeros(2);
k(1) = L(1)+L(2)*cos_2;
k(2) = L(2)*sen_2;
gamma = atan2(k(2),k(1));
thetas(1) = atan2(uwrelb(2),uwrelb(1)) - gamma;
thetas(3) = uwrelb(3) - thetas(1) - thetas(2);
negative_thetas = zeros(3);
gamma = atan2(-k(2),k(1));
negative_thetas(1) =  atan2(uwrelb(2),uwrelb(1)) - gamma;
```

```
    negative_thetas(2) = atan2(-sen_2,cos_2);
    negative_thetas(3) = uwrelb(3) - negative_thetas(1) - negative_thetas(2);
    negative_distance  = sum(abs(negative_thetas(:,1)-current));
    positive_distance  = sum(abs(thetas(:,1)-current));
```

Output Data

```
    if (negative_distance > positive_distance)
        near = thetas(:,1)*180/pi;
        far = negative_thetas(:,1)*180/pi;
    else
        far = thetas(:,1)*180/pi;
        near = negative_thetas(:,1)*180/pi;
    end

    valid_pos = sum(all(thetas >= thetalim(1,:) & thetas <= thetalim(2,:)));
    valid_neg = sum(all(negative_thetas >= thetalim(1,:) & negative_thetas <=
thetalim(2,:)));

    destination_magnitude = sqrt(uwrelb(1)^2+uwrelb(2)^2);
    valid_magnitude = destination_magnitude > sqrt(L(1)^2+L(2)^2);

    if (valid_pos * valid_neg) + valid_magnitude > 1e-4
        sol = 0;
        near = NaN(1,3);
```

```
        far = NaN(1,3);
    else
        sol = 1;
    end

end
```

```
near =

         0
   90.0000
  -45.0000


far =

   61.9275
  -90.0000
   73.0725


sol =

     1
```

Published with MATLAB® R2024b