Transformation matrix

```
T =

    0.9859    0.1676   -0.0000   57.6958
    0.1676   -0.9859   -0.0000 -494.7989
   -0.0000    0.0000   -1.0000  432.2000
         0         0         0    1.0000
```

```
[ 0.985850, 0.167629, -0.000000, 57.695774 ;
  0.167629, -0.985850, -0.000000, -494.798913 ;
  -0.000000, 0.000000, -1.000000, 432.200000 ;
  0.000000, 0.000000, 0.000000, 1.000000 ];
```

Starting joints

-70.710000, -90.000000, -90.000000, -90.000000, 90.000000, -170.360000

8 unique IK solutions

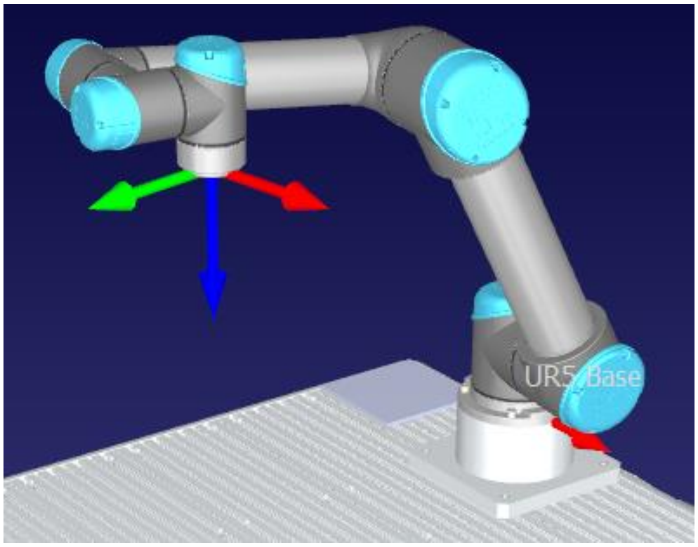
```
ik_sols =

   84.0194  -63.1393   56.3819   96.7574   90.0000  -15.6306
   84.0194   -9.2183  -56.3819  155.6002   90.0000  -15.6306
   84.0194  -90.0000   89.9562  -89.9562  -90.0000  164.3694
   84.0194   -4.6299  -89.9562    4.5861  -90.0000  164.3694
  -70.7100 -175.3701   89.9562  175.4139   90.0000 -170.3600
  -70.7100  -90.0000  -89.9562  -90.0438   90.0000 -170.3600
  -70.7100 -170.7817   56.3819   24.3998  -90.0000    9.6400
  -70.7100 -116.8607  -56.3819   83.2426  -90.0000    9.6400
```

RoboDK 21 IK solutions (only 8 unique)

```
ik_all =

  Columns 1 through 14

  -70.7100   84.0118   84.0118   84.0118  -70.7100   84.0118  -70.7100  -70.7100   84.0118   84.0118   84.0118  -70.7100   84.0118   84.0118
  -90.0000   -9.0246  -90.0000  -63.4657 -116.5343   -4.5634 -170.9754 -175.4366  -90.0000   -9.0246  -63.4657 -175.4366   -4.5634  -90.0000
  -90.0000  -56.9154   90.0000   56.9154  -56.9154  -90.0000   56.9154   90.0000 -270.0000  -56.9154   56.9154   90.0000  -90.0000   90.0000
  -90.0000  155.9400  -90.0000   96.5503   83.4497    4.5634   24.0600  175.4366  -90.0000 -204.0600 -263.4497 -184.5634    4.5634  -90.0000
   90.0000   90.0000  -90.0000   90.0000  -90.0000  -90.0000  -90.0000   90.0000  -90.0000   90.0000   90.0000   90.0000  270.0000  270.0000
 -170.3600  -15.6382  164.3618  -15.6382    9.6400  164.3618    9.6400 -170.3600  164.3618  -15.6382  -15.6382 -170.3600  164.3618  164.3618

  Columns 15 through 21

   84.0118   84.0118   84.0118   84.0118   84.0118   84.0118   84.0118
  -90.0000   -4.5634  -90.0000  -90.0000   -4.5634  -90.0000  -90.0000
 -270.0000  -90.0000   90.0000 -270.0000  -90.0000   90.0000 -270.0000
  -90.0000    4.5634  -90.0000  -90.0000    4.5634  -90.0000  -90.0000
  270.0000  -90.0000  -90.0000  -90.0000  270.0000  270.0000  270.0000
  164.3618 -195.6382 -195.6382 -195.6382 -195.6382 -195.6382 -195.6382
```

84.0194 -63.1393 56.3819 96.7574 90.0000 -15.6306

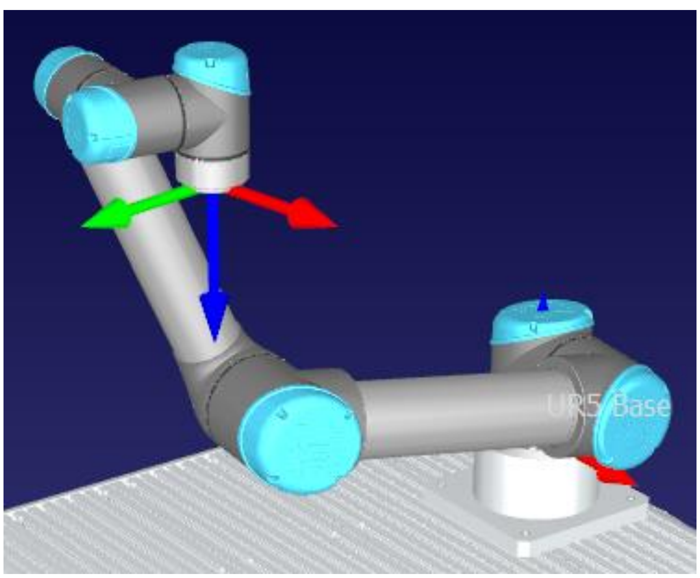
	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
4	4	○	●	●	84.0	-63.5	56.9	96.6	90.0	-15.6



Rear/Up/Flip

84.0194 -9.2183 -56.3819 155.6002 90.0000 -15.6306

	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
2	6	○	○	●	84.0	-9.0	-56.9	155.9	90.0	-15.6

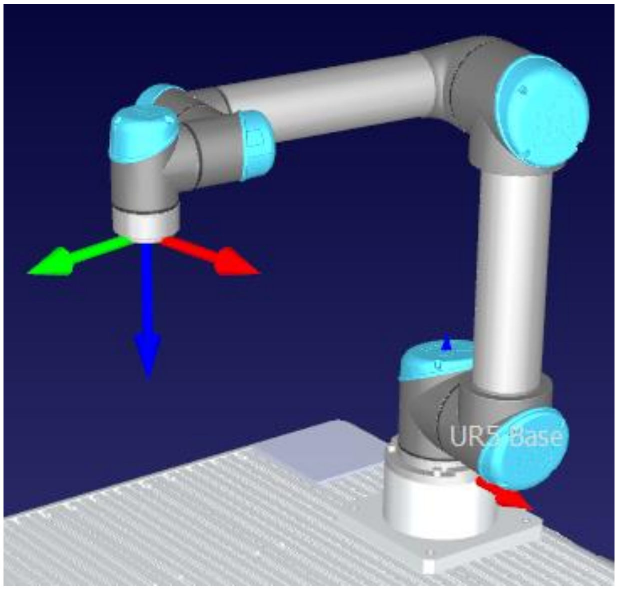


Rear/Down/Flip

If the light is green, then the robot is in the first listed configuration (first from the pair F/R, U/D, F/N). In the first case robot is in configuration **Rear, Up, Flip**, in the second case, robot is in configuration **Rear, Down, Flip.**

84.0194 -90.0000 89.9562 -89.9562 -90.0000 164.3694

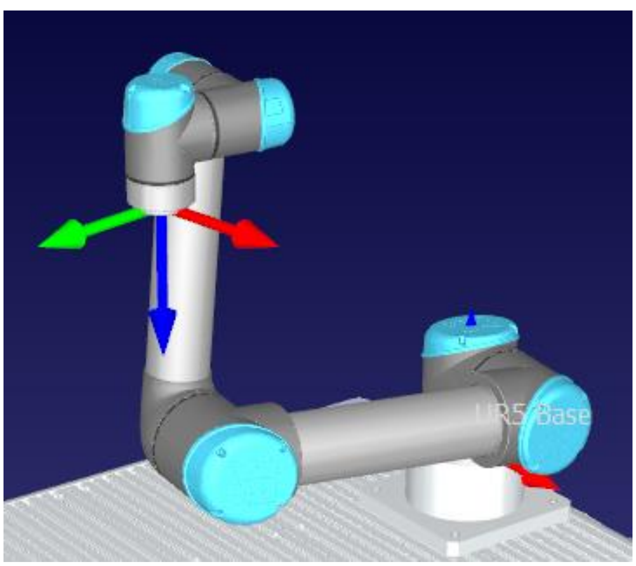
	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
3	5	○	●	○	84.0	-90.0	90.0	-90.0	-90.0	164.4



Rear/Up/ No Flip

84.0194 -4.6299 -89.9562 4.5861 -90.0000 164.3694

	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
6	7	○	○	○	84.0	-4.6	-90.0	4.6	-90.0	164.4

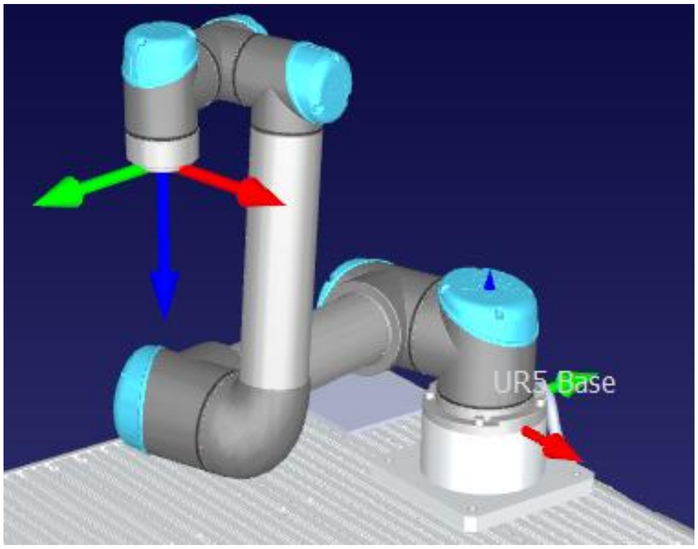


Rear/Down/ No Flip

In the first case robot is in configuration **Rear, Up, No flip**, in the second case, robot is in configuration **Rear, Down, No flip.**

-70.7100 -175.3701   89.9562 175.4139   90.0000 -170.3600

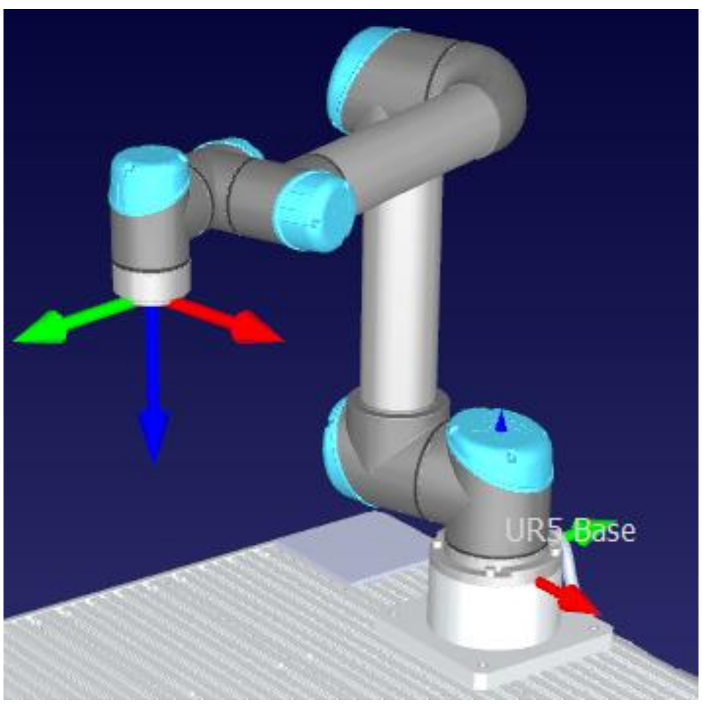
	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
8	0	●	●	●	-70.7	-175.4	90.0	175.4	90.0	-170.4



Front/Up/Flip

-70.7100 -90.0000 -89.9562 -90.0438   90.0000 -170.3600

	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
1	2	●	○	●	-70.7	-90.0	-90.0	-90.0	90.0	-170.4

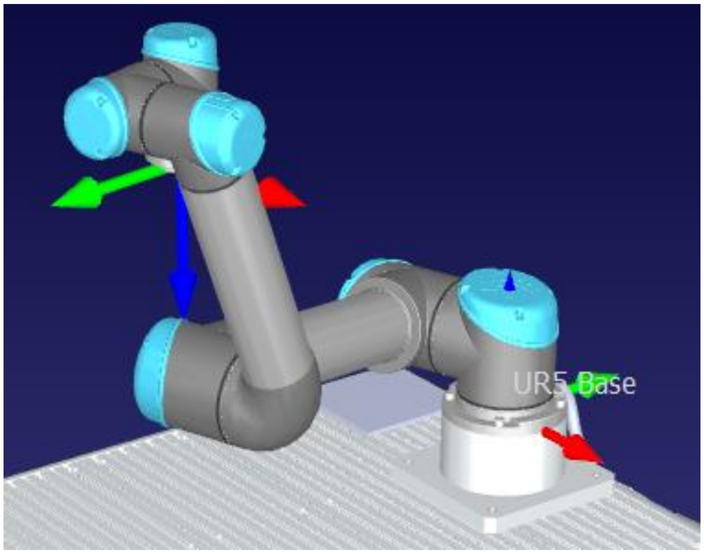


Front/Down/Flip

In the first case robot is in configuration **Front, Up, Flip,** in the second case, robot is in configuration **Front, Down, Flip.**

-70.7100 -170.7817   56.3819   24.3998 -90.0000   9.6400

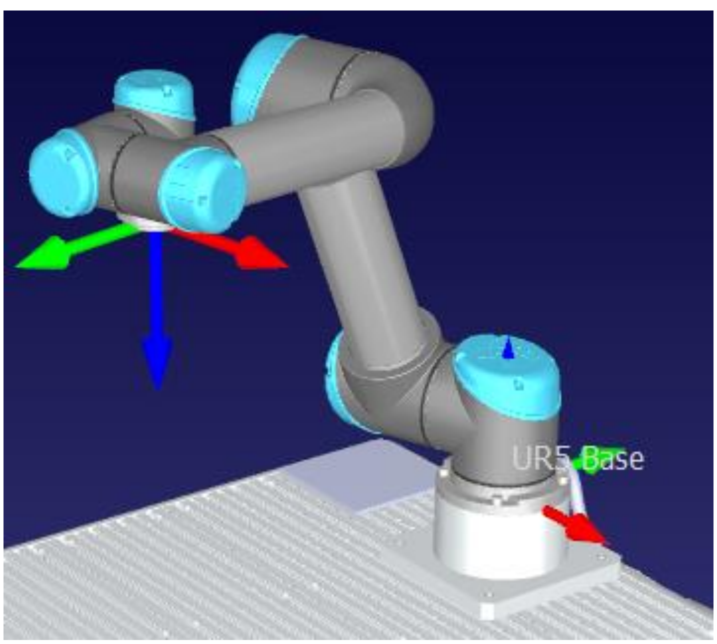
	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
7	1	●	●	○	-70.7	-171.0	56.9	24.1	-90.0	9.6



Front/Up/ No Flip

-70.7100 -116.8607 -56.3819   83.2426 -90.0000   9.6400

	id	F/R	U/D	F/N	J1	J2	J3	J4	J5	J6
5	3	●	○	○	-70.7	-116.5	-56.9	83.4	-90.0	9.6



Front/Down/ No Flip

In the first case robot is in configuration **Front, Up, No flip**, in the second case, robot is in configuration **Front, Down, No flip.**

Python Code

JointsConfig(*joints*)

Returns the robot configuration state for a set of robot joints. The configuration state is defined as: [REAR, LOWERARM, FLIP, turns]. The turns are reserved for future use.

Example:

```
# Retrieve all solutions for a given pose:
all_solutions = robot.SolveIK_All(pose, toolpose, framepose)
joints = []

# Iterate through each solution
for j in all_solutions:
    # Retrieve flags as a list for each solution
    conf_RLF = robot.JointsConfig(j).list()

    # Breakdown of flags:
    rear  = conf_RLF[0] # 1 if Rear , 0 if Front
    lower = conf_RLF[1] # 1 if Lower, 0 if Upper (elbow)
    flip  = conf_RLF[2] # 1 if Flip , 0 if Non flip (Flip is usually when Joint 5 is negative)

    # Look for a solution with Front and Elbow up configuration
    #if conf_RLF[0:2] == [0,0]:
    if rear == 0 and lower == 0:
        print("Solution found!")
        joints = j
        break
```

Sometimes when robot is moving on a specific path we get a sudden change of configuration because it goes through singularity. I would like to try to fix a desired robot configuration for some tasks with the help of RoboDK API for Matlab. When I am calculating Inverse kinematics in Matlab I get 8 unique solutions, but when I use RoboDK IK solver I get up to 21 solutions. So my first question is, how can I return only 8 unique solutions with robodk ik solver into Matlab?

Then I went checking how robodk is showing different configurations. Please take a look at the attached document and confirm if my intepration is correct (with those green lights).

For the last question. It seems that there is already a solution for Python which can recognize different configurations (last page in the attached document). Is it possible to implement this in Matlab? If not, where can I find the program which uses function JointsConfig?

To summarize. When I get 8 unique IK solutions I would like to know for each solution in which configuration (f,r,u,d,f,nf) the robot is (in Matlab). I am already trying to manually limit joints for each configuration, but I think RoboDK can handle this better.