MEAM 520

Lab 2: Inverse Kinematics

Xiaozhou Zhang / Xinlong Zheng

xzzhang@seas.upenn.edu

xinlongz@seas.upenn.edu

0 Prelab

0.1 Task 1

The coordinate frames are assigned as Figure 0-1.

The position of the red dot expressed in the frame 0 is

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} \cos\theta_1^* \cos(\theta_2^* + \theta_3^*) a_3 + \cos\theta_1^* \sin\theta_2^* a_2 \\ \sin\theta_1^* \cos(\theta_2^* + \theta_3^*) a_3 + \sin\theta_1^* \sin\theta_2^* a_2 \\ -\sin(\theta_2^* + \theta_3^*) a_3 + \cos\theta_2^* a_2 + d_1 \end{bmatrix}$$

0.2 Task 2

The numbers of inverse kinematics solutions exist in different regions are shown as Table 0-1 and the region is listed in the prelab's report by Xiaozhou Zhang.

Region	①	②	③	④	⑤	⑥
Number of IK solutions	0	2	∞	4	2	0

Table 0-1 Numbers of inverse kinematics solutions

0.3 Task 3

The solutions to inverse kinematics in a given position are shown as Table 2 (see next page) and the region is listed in the prelab's report by Xiaozhou Zhang.

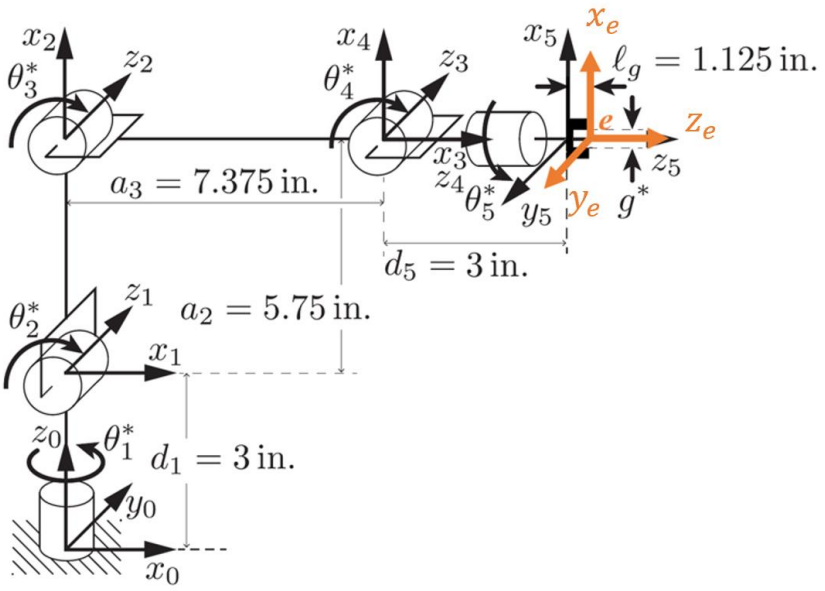

Figure 0-1 Coordinate frames assignment

Region	θ_1^*	θ_2^*	θ_3^*
②	$\mathrm{Atan2}(x,y)$	$\frac{\pi}{2}-\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$	$-\frac{\pi}{2}$
	$\pi+\mathrm{Atan2}(x,y)$	$\frac{3\pi}{2}+\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$	$-\frac{\pi}{2}$
③	Infinite Solutions	$\frac{\pi}{2}-\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$ $-\mathrm{Atan2}(\sqrt{1-E^2}/E)$	$\mathrm{Atan2}(D/\sqrt{1-D^2})$
		$\frac{\pi}{2}-\mathrm{Atan2}(z-d_1,\sqrt{x^2+y^2})$ $-\mathrm{Atan2}(-\sqrt{1-E^2},E)$	$\mathrm{Atan2}(D$ $/-\sqrt{1-D^2})$
④	$\mathrm{Atan2}(x,y)$	$\frac{\pi}{2}-\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$ $-\mathrm{Atan2}(\sqrt{1-E^2}/E)$	$\mathrm{Atan2}(D/\sqrt{1-D^2})$
		$\frac{\pi}{2}-\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$ $-\mathrm{Atan2}(E/-\sqrt{1-E^2})$	$\mathrm{Atan2}(D$ $/-\sqrt{1-D^2})$
	$\pi+\mathrm{Atan2}(x,y)$	$\frac{3\pi}{2}+\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$ $+\mathrm{Atan2}(E/\sqrt{1-E^2})$	$\mathrm{Atan2}(D/\sqrt{1-D^2})$
		$\frac{3\pi}{2}+\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$ $+\mathrm{Atan2}(E/\sqrt{1-E^2},)$	$\mathrm{Atan2}(D$ $/-\sqrt{1-D^2})$
⑤	$\mathrm{Atan2}(x,y)$	$\frac{\pi}{2}-\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$	$\frac{\pi}{2}$
	$\pi+\mathrm{Atan2}(x,y)$	$\frac{3\pi}{2}+\mathrm{Atan2}((z-d_1)/\sqrt{x^2+y^2})$	$\frac{\pi}{2}$

$*D=\frac{a_2^2+a_3^2-x^2-y^2-(z-d_1)^2}{2a_2a_3}$

$*E=\frac{a_2^2+x^2+y^2+(z-d_1)^2-a_3^2}{2a_2\sqrt{x^2+y^2+(z-d_1)^2}}$

Table 0-2 Solutions to inverse kinematics in a given position

1 Method

1.1 Sets of wrist center

Let's look at the plane formed by link 2 and link 3, e_n represents different positions of the origin of the frame e. The workspace of the arm in the $r - z_0$ plane is limited to the ring area between the red circle and the blue circle, and the black line represents the set of positions that the center of the wrist can take in the plane $r - z_0$ in order for frame e to have its origin in the plane $r - z_0$, rotate the plane by axes z_0 , we get all sets of the center of the wrist. The black lines are the parts of the cycle that in the reachable space of the wrist center, shown as Figure 1-1.

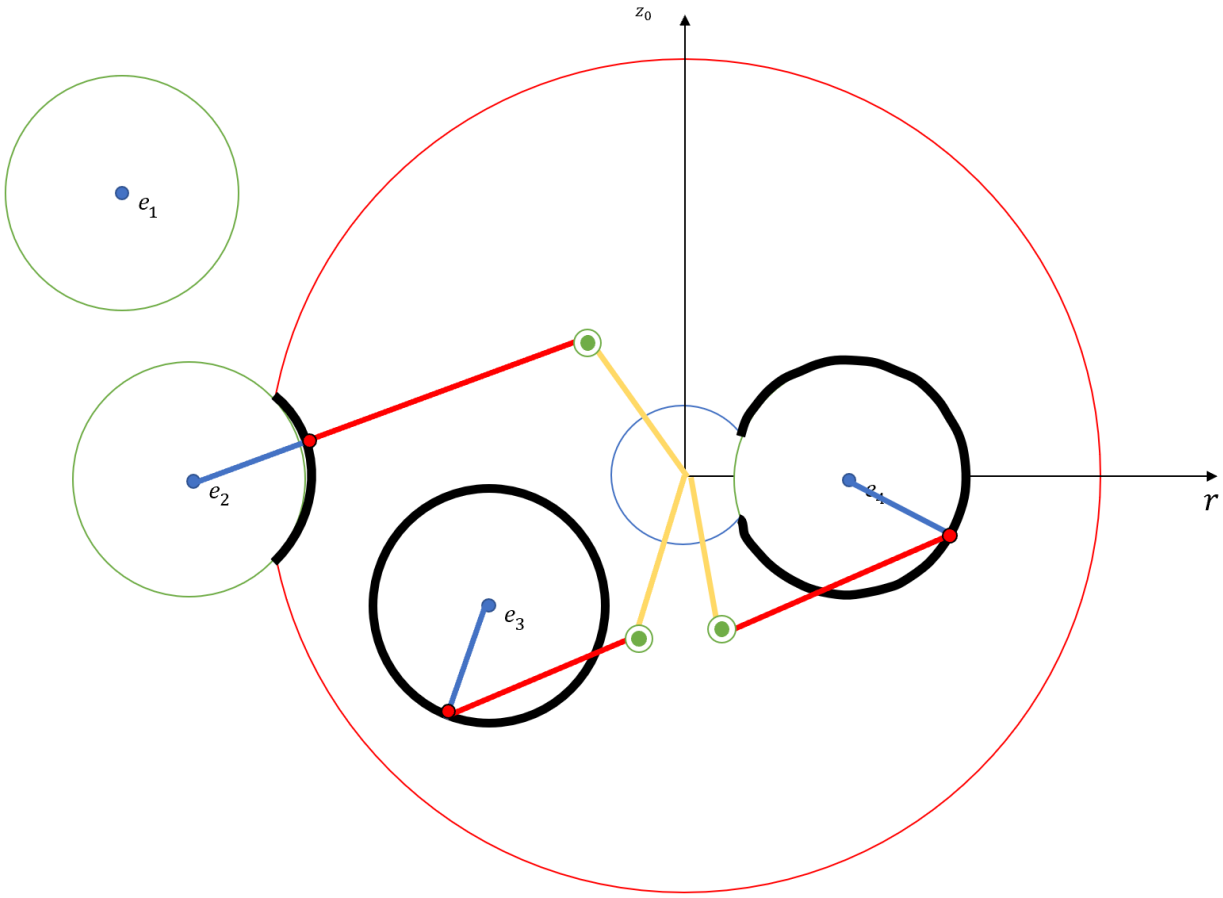

Figure 1-1 The sets of wrist center

1.2 Compute the wrist joints angles

We are taking algebraic approach.

With the homogenous transformation, we can get the position vector $o = [o_x\ o_y\ o_z]^T$ and the orientation matrix R of the target, both expressed in the base frame. We set $o_c^0 = [x\ y\ z]^T$ as the wrist center's position, expressed in the base frame. We set $[d_1\ a_2\ a_3\ d_5\ l_g]$ as the dimension of the robot.

The origin of the end effector's frame is obtained by a translation of distance $d_5 + l_g$ along z_5 from o_c^0. In this case, z_5 and z_e are the same axis, and the third column of R expresses the direction of z_e with respect of the base frame. Therefore, we have (SHV p96)

$$o_c^0 = o - (d_5 + l_g)R\begin{bmatrix}0\\0\\1\end{bmatrix}$$

Once o_c^0 is determined, use the method in Prelab to compute θ_1^*, θ_2^* and θ_3^*. The orientation transformation R_3^0 is depend only on the first three joint variables. The way to determine the rotation matrix is stated in the lab1's report of Xinlong Zheng. We can now determine the orientation of the end effector relative to the frame $o_3x_3y_3z_3$ from the expression (SHV p97)

$$R_e^3 = (R_3^0)^T R = (R_0^1 R_1^2 R_2^3)^T R$$

From what is done in the lab1, we have

$$R_4^3 = \begin{bmatrix} s_4 & 0 & c_4 \\ -c_4 & 0 & s_4 \\ 0 & -1 & 0 \end{bmatrix}$$

$$R_e^4 = \begin{bmatrix} c_5 & -s_5 & 0 \\ s_5 & c_5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_e^3 = R_4^3 R_e^4 = \begin{bmatrix} s_4c_5 & -s_4s_5 & c_4 \\ -c_4c_4 & c_4s_5 & s_4 \\ -s_5 & -c_4 & 0 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

in which,

$$s_4 = sin(\theta_4^*),$$

$$c_4 = \cos(\theta_4^*),$$

$$s_5 = \sin(\theta_5^*),$$

$$c_5 = cos(\theta_5^*).$$

It is obvious that

$$\theta_4^* = \text{Atan2}(r_{23}/r_{13})$$

$$\theta_5^* = \text{Atan2}(-r_{31}/-r_{32})$$

1.3 Orientation check

For lynx robot, if the z_e axis is not vertical to $z_3(z_2, z_1)$, then the orientation is not reachable because the setup only has 5 DOF.

For a given transformation matrix,

$$T_6^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We assume that it represents a target lies within the workspace and is not in a singular configuration (arms fully extended).

The z_e axis should be vertical to $z_3(z_2, z_1)$ to make sure the orientation reachable. In another word, the z_e axis should lie in the plane formed by link 2 and link 3.

The plane formed by link 2 and link 3 is determined by the position vector $[t_1 \quad t_2 \quad t_3]^T$, and the plane where z_e axis lies in is determined by orientation matrix on the upper left corner. We need to check for the whether these two plane are coplanar. In a mathematical way, that is whether

$$r_{13} t_2 = t_1 r_{23}$$

If the equation is valid, then the orientation is feasible. If the equation is not valid, then the orientation is not feasible.

The way to consider whether the orientation is feasible including singular configuration is presented in section 2.1.

2 Evaluation

2.1 Compute inverse kinematics

The task is asked to take a homogenous transformation matrix as a target, and return a Boolean value, which indicates weather the target is achievable by the robot, ignoring joint limits, and a vector of 5 joints values which bring the gripper to the target, considering joint limits. There might be some case which the Boolean value indicates that the target is achievable, but the vector of 5 joints values is not outputted since it is actually not achievable by the physical robot due to joint limits.

The program flow chat is shown as Figure 2-1.

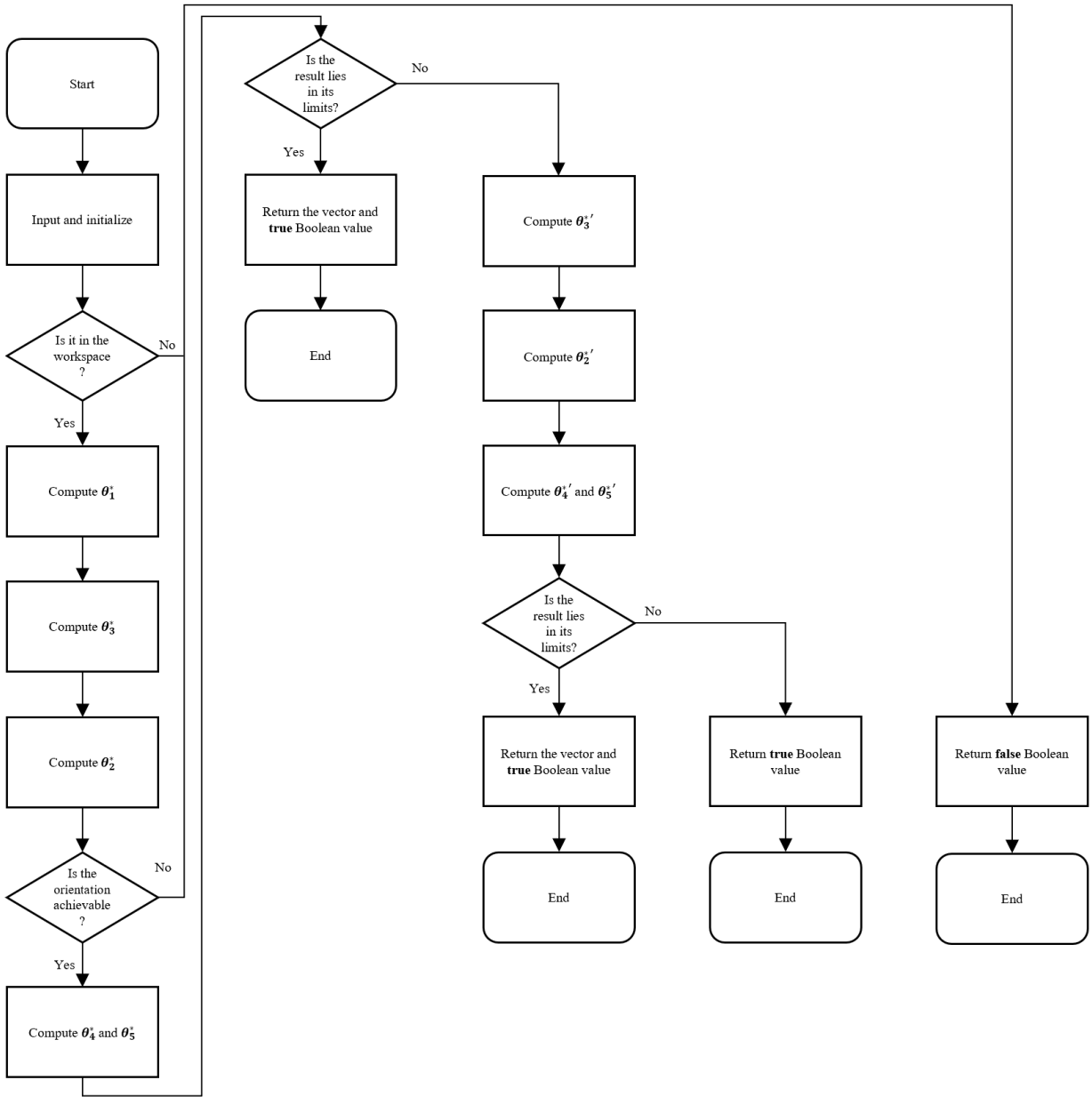

Figure 2-1 Flow chat of inverse kinematics program

(1) Input and initialize the decoupling
In this program, we take a homogenous transformation matrix as a target. Using the method stated in section 1.2 to get the wrist center's position $o_c^0 = [x\ y\ z]^T$.
Now we finish the kinematic decoupling by getting the position of the wrist center.

(2) Judge if the target is located in the workspace
Consider the plane formed by link 2 and link 3, which is shown as Figure 2-2.
By the law of cosine, we know

$$\cos(\alpha) = \frac{-r^2 - s^2 + a_2^2 + a_3^2}{2a_2a_3} = \frac{a_2^2 + a_3^2 - x^2 - y^2 - (z - d_1)^2}{2a_2a_3}$$

If $|\cos(\alpha)| >= 1$, such triangle cannot be formed, which means the position of the wrist center cannot be achieved by the robot, therefore the target is not located in the workspace of the robot. Thus, we return a false Boolean value to indicate that the target is not achievable even by the simulated robot.

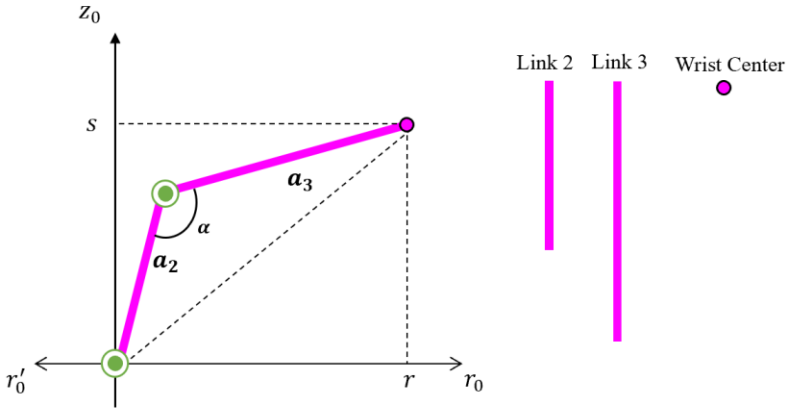

Figure 2-2 Projecting onto the plane formed by link 2 and link 3

(3) Compute θ_1^*
As what is stated in the prelab, the process of computing θ_1^* is shown as Figure 2-3.

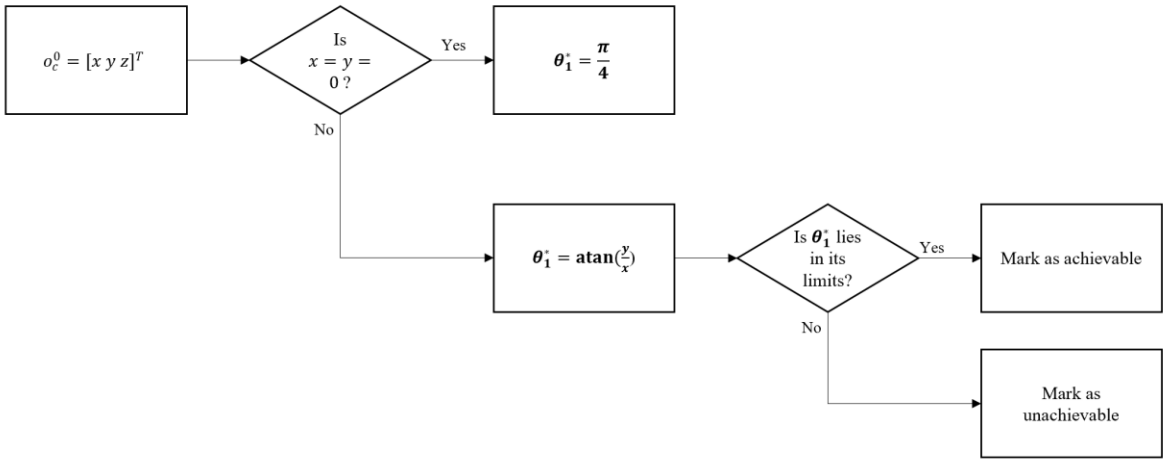

Figure 2-3 Flow chat of computing θ_1^*

When $x = y = 0$, there are infinite solutions for θ_1^*, we assign one solution, $\frac{\pi}{4}$, for it arbitrarily. Generally, there are two solutions for θ_1^*. We want to output the one which lies within the joint limits. We choose the function *atan(y/x)* rather than *atan2(y,x)* in MATLAB to ensure so. Because the range of joint 1 is a subset of the range of *atan(y/x)*, if the computed θ_1^* doesn't lie in its limits, then another solution would be the same, thus make the target unachievable by the physical robot. The relationship between the two solutions is shown in a polar coordinate system in Figure 2-4.

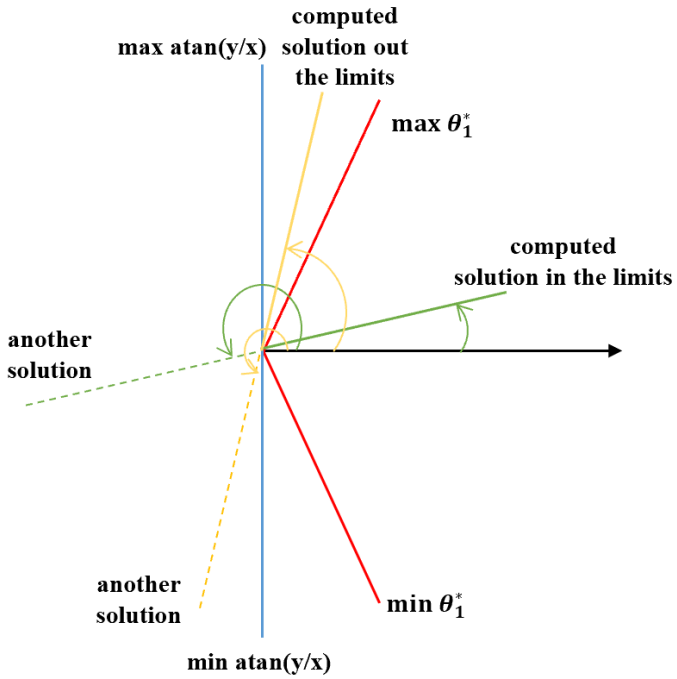

Figure 2-4 Select θ_1^* to output

(4) Compute θ_3^* and θ_2^*

As what is stated in the prelab, the process of computing θ_2^* and θ_3^* is shown as Figure 2-5.

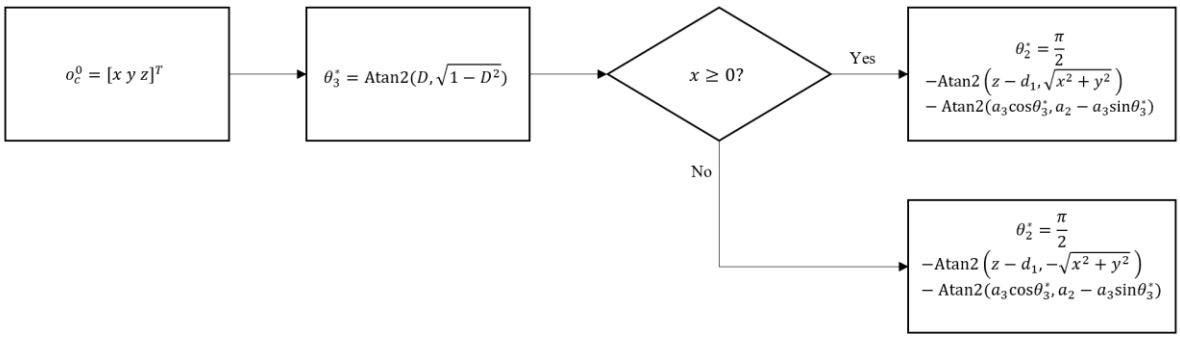

Figure 2-5 Flow chat of computing one solution for θ_2^* and θ_3^*

The method of computing θ_3^* is the same while which of θ_2^* is different from the prelab. We compute θ_2^* as a function of θ_3^* to make sure the correspondence relationship between each other. We choose one solution for θ_3^*, $\text{Atan2}(D, \sqrt{1-D^2})$, first, to complete the inverse kinematics. Like what is done in the prelab, consider the plane formed by link 2 and link 3, which is shown as Figure 2-6.

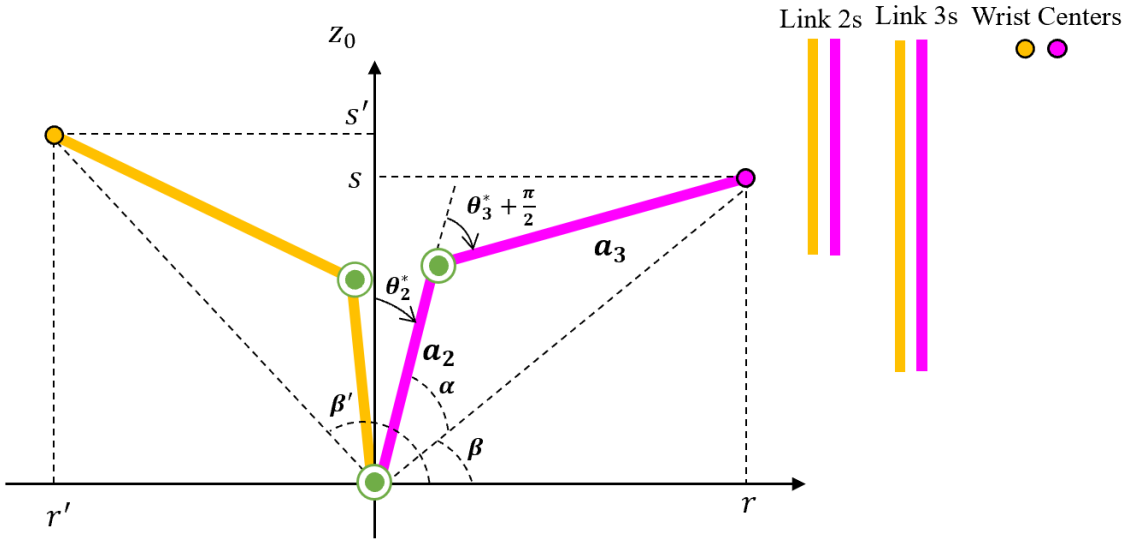

Figure 2-6 Projecting onto the plane formed by link 2 and link 3

Since θ_1^* is determined but maybe flipped from the original solution (caused by the function *atan(y/x)*), there is only one solution for θ_2^* but maybe two different expressions for θ_1^*. It is obvious that

$$\theta_2^* = \frac{\pi}{2} - \beta - \alpha$$

$$\alpha = \text{Atan2}(a_3 cos\theta_3^*, a_2 - a_3 sin\theta_3^*)$$

When $x \geq 0$, the function *atan(y/x)* does not flip the original θ_1^*, shown as the pink one in Figure 2-6. We get

$$\beta = \text{Atan2}(z - d_1, \sqrt{x^2 + y^2})$$

$$\theta_2^* = \frac{\pi}{2} - \beta - \alpha = \text{Atan2}(z - d_1, \sqrt{x^2 + y^2}) - \text{Atan2}(a_3 cos\theta_3^*, a_2 - a_3 sin\theta_3^*)$$

When $x \geq 0$, the function *atan(y/x)* does not flip the original θ_1^*, shown as the yellow one in Figure 2-6. We get

$$\beta' = \text{Atan2}(z - d_1, -\sqrt{x^2 + y^2})$$

$$\theta_2^* = \frac{\pi}{2} - \beta' - \alpha = \text{Atan2}(z - d_1, -\sqrt{x^2 + y^2}) - \text{Atan2}(a_3 cos\theta_3^*, a_2 - a_3 sin\theta_3^*)$$

(5) Judge if the target's orientation is achievable

The orientation transformation R_3^0 is depend only on the first three joint variables, which we have already determined. The way to determine R_3^0 is stated in the lab1's report of Xiaozhou Zhang. We can now determine the orientation of the end effector relative to the frame $o_3x_3y_3z_3$ from the expression (SHV p97)

$$R_e^3 = (R_3^0)^T R = (R_0^1 R_1^2 R_2^3)^T R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

As what is state in the Method part, there are some orientations the end effector cannot achieve because it has only 5 DOF.

In order to make sure z_e axis is vertical to z_3, we need to check the z_3 component of z_e, which is r_{33}. If $r_{33} = 0$, then there is no z_3 component of z_e, which means z_e axis is vertical to z_3, thus the orientation is reachable.

When the robot is in the singular configuration, the z_e axis should collinear with link 2, not only just lie in the plane formed by link 2 and link 3. However, when the end effector's position is in the edge of the workspace and its orientation is not reachable, we would be excluded already in the workspace judgement part, therefore the method of judging the orientation is universal.

The choice of two correspondent solutions for θ_2^* and θ_3^* does not affect the judgement, so once the condition is not satisfied, we can just return a false Boolean value to indicate that the target is not achievable even by the simulated robot.

The process of the judgement is shown as Figure 2-7.

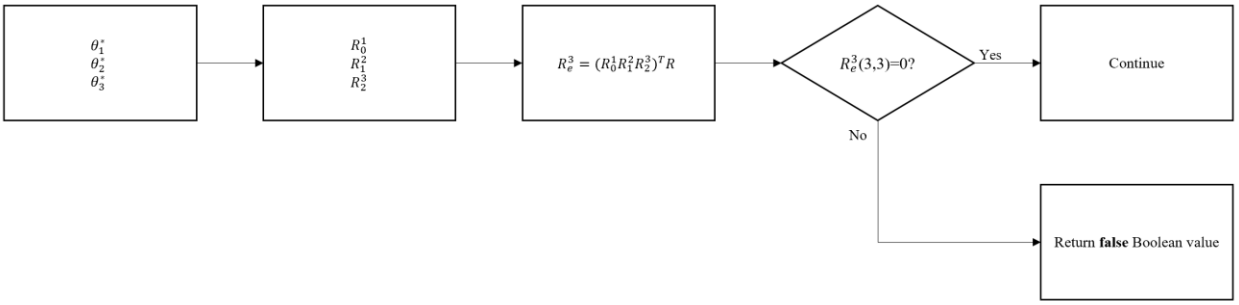

Figure 2-7 Flow chat of judging the orientation existence

(6) Compute θ_4^* and θ_5^*

As what is stated in section 1.2, we have

$$\theta_4^* = \text{Atan2}(r_{23}, r_{13})$$
$$\theta_5^* = \text{Atan2}(-r_{31}, -r_{32})$$

(7) Judge if the joint variables lie within joint limits

If the solutions above lie within in joint limits, we just output them and return a true Boolean value. If not, we repeat step (4) and step (6), then we judge again. If the result this time is satisfied, we output them and return a true Boolean value. Otherwise, we just return a true Boolean value to indicate that there are simulated solutions, but it cannot be performed by the physical robot due to the joint limits. Thus, complete the inverse kinematics.

There is a MATLAB code *IK_lynx_xzzhangxinlongz.m* listed in Appendix 1 and attached to the file.

2.2 Test the code

Write a function *test*, which takes q, 1×6 vector of joint inputs, as the input. Call the function *calculateFK* to generate a homogenous matrix T_e^0 as the input of the function *IK_lynx_xzzhangxinlongz*. Call the function to compute inverse kinematics.

There is a MABLAB code *test.m* is listed in the Appendix 1 and attached to the file.

(1) Test the input within joint limits
We choose two sets of input to call the *test* function, one set is negative and the other is positive. The commands are shown as followed.

```
>> q = [-pi/3 -pi/4 -pi/5 -pi/6 -pi/4 0];
>> test(q)

q =   -1.0472   -0.7854   -0.6283   -0.5236   -0.7854
is_possible =  logical   1
```

```
>> q = [pi/5 pi/3 pi/6 pi/4 pi/3 0];
>> test(q)

q =    0.6283    1.0472    0.5236    0.7854    1.0472
is_possible =  logical   1
```

Both the test returns satisfied results.

(2) Test the input outside of joint limits
We choose five sets of input to call the *test* function, each of them contains a out-of-range joint angles, from joint 1 to joint 5. The commands are shown as followed.

```
>> q = [pi/2 pi/3 pi/6 pi/4 pi/3 0];
>> test(q)

is_possible =  logical   1
no such joint variable values exist, due to the joint limits
```

```
>> q = [pi/4 pi/2 pi/6 pi/4 pi/3 0];
>> test(q)

is_possible =  logical   1
no such joint variable values exist, due to the joint limits
```

```
>> q = [pi/4 pi/2 2*pi/3 pi/4 pi/3 0];
>> test(q)

is_possible =  logical   1
no such joint variable values exist, due to the joint limits
```

```
>> q = [pi/4 pi/3 pi/6 2*pi/3 pi/3 0];
>> test(q)

is_possible =  logical   1
no such joint variable values exist, due to the joint limits
```

```
>> q = [pi/4 pi/3 pi/6 pi/4 pi/2 0];
>> test(q)

is_possible =  logical   1
no such joint variable values exist, due to the joint limits
```

All the test returns satisfied results.

(3) Test the input beyond the workspace

Choose the homogenous matrix T, which is obviously beyond the workspace (the value of position vector's components are relatively big), as the input to call the *IK_lynx_xzzhangxinlongz* function. The matrix and the commands are shown as followed.

$$T = \begin{bmatrix} 1 & 0 & 0 & 300 \\ 0 & 1 & 0 & 500 \\ 0 & 0 & 1 & 500 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
>> T = [1  0  0  300;
        0  1  0  300;
        0  0  1  500;
        0  0  0    1];
>> IK_lynx_xzzhangxinlongz(T)

no such joint variable values exist, because the target is beyond the
workspace
is_possible =  logical   0
```

The outputted result is satisfied.

(4) Test the input where orientation cannot be achieved

Choose the homogenous matrix T, in which the orientation cannot be achieved obviously (the position vector suggests the plane where z_e axis should lie in, but the orientation matrix suggest a different plane where z_e lies in. It will be discussed in detail in 2.3), as the input to call the *IK_lynx_xzzhangxinlongz* function. The matrix and the commands are shown as followed.

$$T = \begin{bmatrix} 1 & 0 & 0 & 90 \\ 0 & 0 & 1 & 90 \\ 0 & 1 & 0 & 20 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
>> T = [1    0    0    90;
        0    0    1    90;
        0    1    0    20;
        0    0    0     1];
>> IK_lynx_xzzhangxinlongz(T)

no such joint variable values exist, because such orientation is not
achievable
is_possible =  logical   0
```

The outputted result is satisfied.

2.3 Try to hit the target

We did this task with the original targets. We attached some hand-made coordinate frames to the targets, base frame, and end effector, to make the measurement a little bit easier, shown as Figure 2-8.

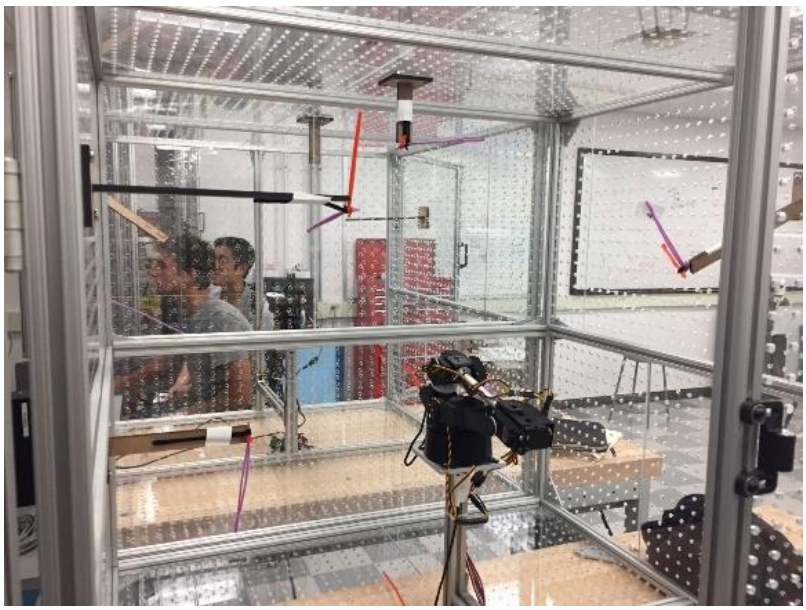

Figure 2-8 Hand-made coordinate frames

(1) Target 1

It seems that target 1 is achievable, since the z_e axis lies in the plane formed by link 2 and link 3. In order to obtain the homogenous transformation matrix T, we need to calculate the rotation matrix R first.

Then rotation from the base frame to the end effector's frame can be broken into three parts, which are rotation about the x_0 axis, rotation about the current z axis, and rotation about the current x axis, as

$$R = R_1R_2R_3 = \begin{bmatrix}1 & 0 & 0\\ 0 & cos\alpha & -sin\alpha\\ 0 & sin\alpha & cos\alpha\end{bmatrix}\begin{bmatrix}cos\beta & -sin\beta & 0\\ sin\beta & cos\beta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & cos\gamma & -sin\gamma\\ 0 & sin\gamma & cos\gamma\end{bmatrix} = \begin{bmatrix}r_{11} & r_{12} & r_{13}\\ r_{21} & r_{22} & r_{23}\\ r_{31} & r_{32} & r_{33}\end{bmatrix}$$

in which

$$\alpha = 90°, \beta = 29.86°, \gamma = -35.21°$$

The value of β and γ can be obtained from the drawing attached to the handout.

Then we manage to assign the translation vector $o = [x \quad y \quad z]^T$. To make sure the z_e axis lies in the plane formed by link 2 and link 3, we only measured x component and z component of the vector, and take y component as

$$y = \frac{r_{23}}{r_{13}}x$$

since the r_{13} and r_{23} are the components of z_e projecting on the x_0 and y_0 axes.
Thus, we obtain the homogeneous transformation matrix T, as

$$T = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix}$$

The commands are shown as followed.

```
>> p1 = pi/2;
   p2 = 29.86/180*pi;
   p3 = -35.21/180*pi;
   R1 = [1        0        0;
         0  cos(p1)  -sin(p1);
         0  sin(p1)   cos(p1)];
   R2 = [cos(p2)  -sin(p2)  0;
         sin(p2)   cos(p2)  0;
               0         0  1];
   R3 = [1        0         0;
         0  cos(p3)  -sin(p3);
         0  sin(p3)   cos(p3)];
   R = R1*R2*R3;
   ze = R(:,3);
   T = [R(1,1:3),-120;
        R(2,1:3),-120*ze(2)/ze(1);
        R(3,1:3), 305;
        0 0 0 1];
>> IK_lynx_xzzhangxinlongz(T)

is_possible =  logical   1
no such joint variable values exist, due to the joint limits
```

The result indicates that there is a solution in the simulated robot. But the target is not achievable by the physically robot due to joint limits.

(2) Target 2

Like target 1, it seems that target 2 is achievable. Rotation from the base frame to the end effector's frame can also be broken into three parts, which are rotation about the x_0 axis, rotation about the current z axis, and rotation about the current x axis, as

$$R = R_1R_2R_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix}\begin{bmatrix} cos\beta & -sin\beta & 0 \\ sin\beta & cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

in which

$$\alpha = -90°, \beta = -90°, \gamma = 19.36°$$

The value of γ can be obtained from the drawing attached to the handout.
Likewise, when we assign the translation vector $o = [x \quad y \quad z]^T$, we only measured x component and z component of the vector, and take y component as

$$y = \frac{r_{23}}{r_{13}}x$$

Thus, we obtain the homogeneous transformation matrix T, as

$$T = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix}$$

The commands are shown as followed.

```
>> p1 = -pi/2;
   p2 = -pi/2;
   p3 = 19.36/180*pi;
   R1 = [1        0        0;
         0  cos(p1)  -sin(p1);
         0  sin(p1)   cos(p1)];
   R2 = [cos(p2)  -sin(p2)  0;
         sin(p2)   cos(p2)  0;
               0         0  1];
   R3 = [1        0         0;
         0  cos(p3)  -sin(p3);
         0  sin(p3)   cos(p3)];
   R = R1*R2*R3;
   ze = R(:,3);
   T = [R(1,1:3),-30;
        R(2,1:3),-30*ze(2)/ze(1);
        R(3,1:3), 380;
        0 0 0 1];
>> IK_lynx_xzzhangxinlongz(T)

is_possible =  logical   1
no such joint variable values exist, due to the joint limits
```

The result indicates that there is a solution in the simulated robot. But the target is not achievable by the physically robot due to joint limits.

(3) Target 3

Like target 1 and target 2, it seems that target 3 is achievable, since the z_e axis lies in the plane formed by link 2 and link 3.

Then rotation from the base frame to the end effector's frame is just rotation about the z_0 axis, as

$$R = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

in which

$$\alpha = 180°$$

When we assign the translation vector $o = [x \quad y \quad z]^T$, we only measured x component and z component of the vector, since the target lies in the plane $x_0 z_0$ thus y component is 0.

Thus, we obtain the homogeneous transformation matrix T, as

$$T = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix}$$

The commands are shown as followed.

```
>> T = [-1  0  0  230;
         0 -1  0    0;
         0  0  1  460;
         0  0  0    1];
>> IK_lynx_xzzhangxinlongz(T)

no such joint variable values exist, because the target is beyond the
workspace
is_possible =  logical   0
```

The result indicates that there is no solution in the simulated robot and physical robot, since the target is not within the workspace.

(4) Target 4

Unlike the former targets, target is not likely be achievable since the z_e axis does not lie in the plane formed by link 2 and link 3.

Then rotation from the base frame to the end effector's frame is just rotation about the x_0 axis, as

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix}$$

in which

$$\alpha = -90°$$

Then we measured the components of the translation vector $o = [x \quad y \quad z]^T$ to get the homogenous transformation matrix T, as

$$T = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix}$$

The commands are shown as followed.

```
>> T = [ 1  0  0   90.5;
         0  0  1  287.0;
         0 -1  0   20.0;
         0  0  0      1];
>> IK_lynx_xzzhangxinlongz(T)

no such joint variable values exist, because such orientation is not
achievable
is_possible =  logical   0
```

The result indicates that there is no solution in the simulated robot and physical robot, since the orientation is not achievable, which agrees with our suppose above.

Since none of the targets is achievable by the physical robot, we made another two targets which are theoretically achievable by the physical robot, named Target 5 and Target 6. Target 5 is just the modification of target 1 where the y_e axis is flipped. Target 6 is the modification of target 3, where the z component of the translation vector is smaller to make sure it lies within the workspace.

(5) Target 5

The rotation from the base frame to the end effector's frame is which from the target 1 followed by a rotation of π about the z_e axis. Everything else about the homogenous transformation matrix is the same as what in the target 1.

The commands are shown as followed.

```
>> p1 = pi/2;
   p2 = 29.86/180*pi;
   p3 = -35.21/180*pi;
   p4 = pi;
   R1 = [1        0        0;
         0  cos(p1)  -sin(p1);
         0  sin(p1)   cos(p1)];
   R2 = [cos(p2)  -sin(p2)  0;
         sin(p2)   cos(p2)  0;
               0         0  1];
   R3 = [1        0         0;
         0  cos(p3)  -sin(p3);
         0  sin(p3)   cos(p3)];
   R4 = [cos(p4)  -sin(p4)  0;
         sin(p4)   cos(p4)  0;
               0         0  1];
   R = R1*R2*R3*R4;
   ze = R(:,3);
   T = [R(1,1:3),-120;
        R(2,1:3),-120*ze(2)/ze(1);
        R(3,1:3), 305;
        0 0 0 1];
>> IK_lynx_xzzhangxinlongz(T)

q =    1.2329   -1.1985   -1.1789   -0.2406   -0.9583
is_possible =  logical   1
```

Write a function *hit*, which take the outputted vector q and the target homogenous transformation matrix *T* as the input. Put the vector into forward kinematics, and compare the computed result with the input target in a 3D plot. There is a MATLAB code *hit.m* listed in Appendix 1 and attached to the file.

When do so, comment on the line 18 (the line mentioned pennkey) of the function *plotLynx*.

The commands are shown as followed.

```
>> q=[1.2329   -1.1985   -1.1789   -0.2406   -0.9583];
>> hit(q,T)
```

The result is shown as Figure 2-9.

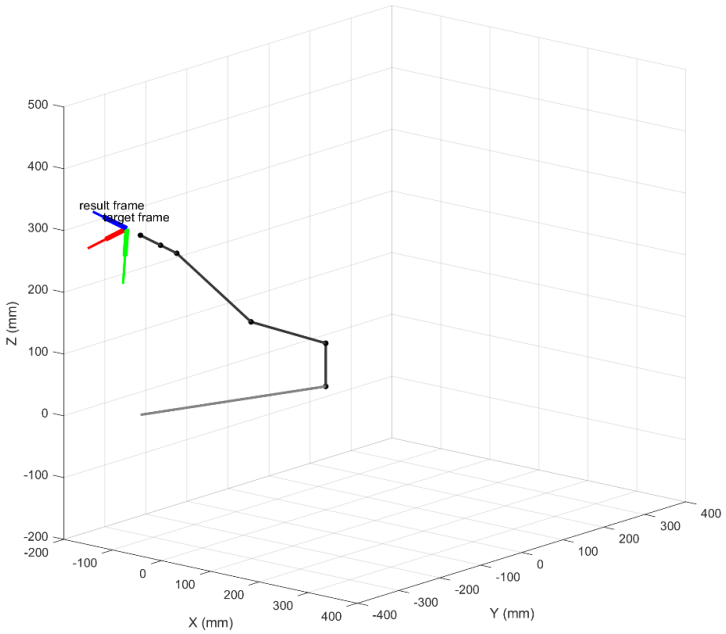

Figure 2-9 IK result of Target 5

Test the inverse kinematics result on the physical robot, the commands are shown as followed.

```
>> q=[1.2329   -1.1985   -1.1789   -0.2406   -0.9583    0];
>> lynxServo(q)
```

The result is shown as Figure 2-10 and there is a video (Video 1) showing the process listed in Appendix 2.

Figure 2-10 Experiment of Target 5

(6) Target 6

The input transformation matrix of target 6 is the same to which of target 2, except that the *z* component of the translation vector is smaller to make sure it lies within the workspace.
The commands are shown as followed.

```
>> T = [-1  0  0  230;
         0 -1  0    0;
         0  0  1  300;
         0  0  0    1];
>> IK_lynx_xzzhangxinlongz(T)

q =         0    0.3003   -0.2248   -1.6463         0
is_possible =  logical   1
```

Like target 5, call the function *hit* to compare the computed result with the input target in a 3D plot.
The commands are shown as followed.

```
>> q=[0    0.3003   -0.2248   -1.6463         0];
>> hit(q,T)
```

The result is shown as Figure 2-11.

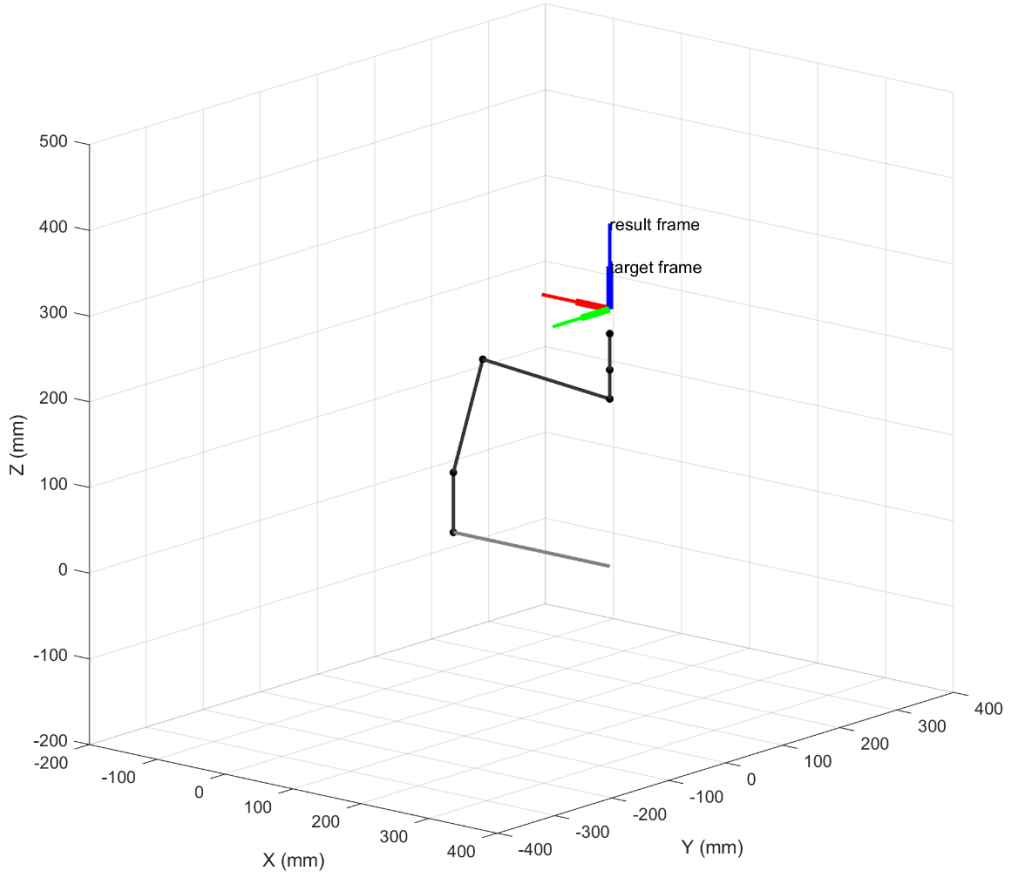

Figure 2-11 IK result of Target 6

Test the inverse kinematics result on the physical robot, the commands are shown as followed.

```
>> q=[0    0.3003   -0.2248   -1.6463     0    0];
>> lynxServo(q)
```

The result is shown as Figure 2-12 and there is a video (Video 2) showing the process listed in Appendix 2.

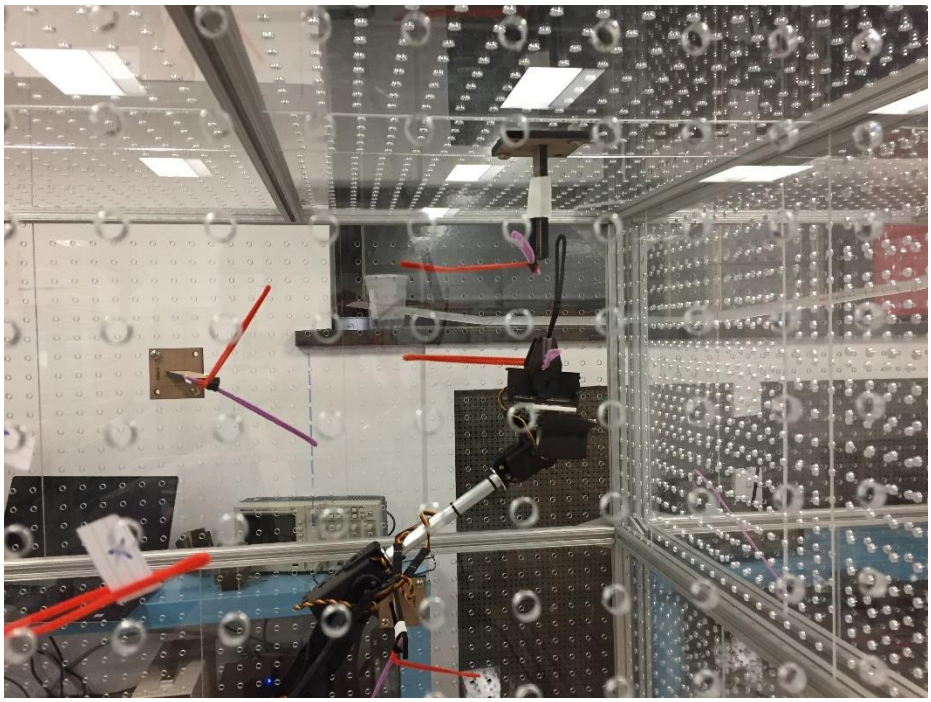

Figure 2-12 Experiment of Target 6

2.4 Compare the simulations and the experiments

During the simulations, put the inverse kinematic results, the outputted joint variables, into forward kinematics, and it will return the same homogenous transformation matrix with the target one. But during the experiments, the result is not that accurate compared to the simulation, due to some system error and random error.

Take Target 5 as an example, projecting the simulation robot and the physical robot on the plane x_0y_0, shown as Figure 2-13. There is a slight difference between these two projections. The reason for which can be the dimension error during the measurement, and some system error as the low accuracy in the servos.

Projecting the simulation robot and the physical robot on the plane formed by link 2 and link 3, shown as Figure 2-14. The difference between these two projections is obvious. The main reason for which is due to the gravity on the servos, dragging the arm down to below the supposed position.

Projecting the simulation end effector's frame and the physical one on the plane x_ey_e, shown as Figure 2-15. There is a slight difference between these two projections. The reason for which can be some system error as frictions in the servos.

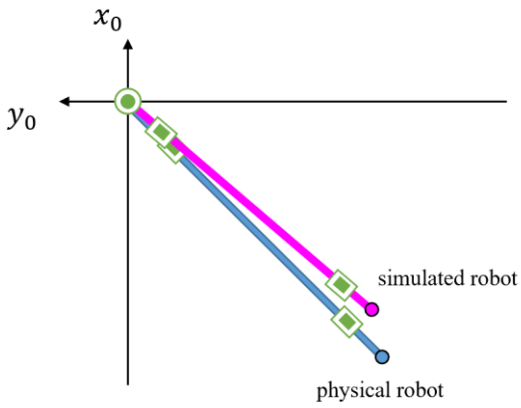

Figure 2-13 Projecting the simulated the physical robots on the plane x_0y_0

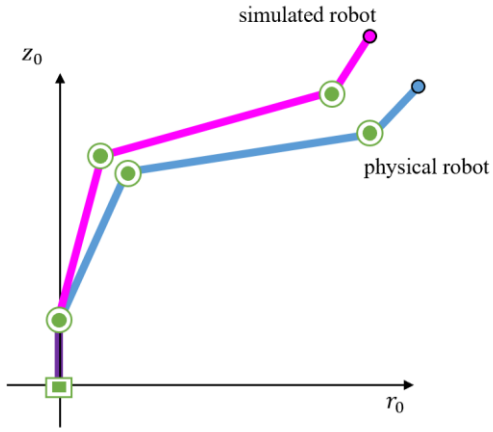

Figure 2-14 Projecting the simulated the physical robots on the plane formed by link 2 and link 3

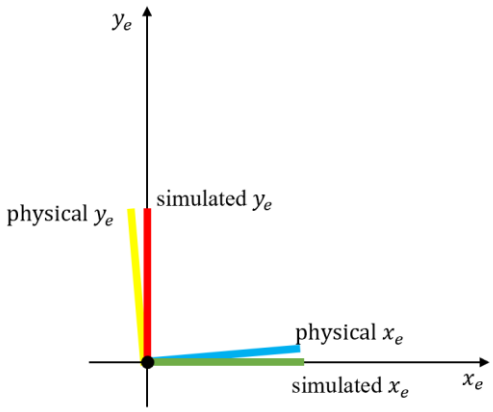

Figure 2-15 Projecting the simulated the physical tool frames on the plane x_ey_e

3 Analysis

3.1 The major factors that affect the accuracy

As what is stated in lab1's report, there are some system error (i.e. error of the servo) and random error (i.e. the measurement error) which affect the accuracy of physical robot's running. The measurement error is indeed a huge factor which affect the accuracy, since it is hard to measure the target position in a cubic box with rules and it would always exist.

In terms of system error, there are torques limits for motor and exist of friction will cause some error. The gravity is also a factor, because the robot is not ideally rigid, its body weight will cause a torque from joint to joint and cause a small curve to the link.

3.2 The impact of the factors on the joint variables

In terms of measurement error, it is hard to tell the specific impact of the error on the joint variables, since it is random.

In terms of system error, firstly, frictions and torques limits have a negative effect when the robot trying to reach a certain target. Thus, we need to give a bigger (absolute value) input than the theoretical one on joint variables to reach the target.

Secondly, the existence of gravity mainly affects the joint variables q_2 and q_3. The reason that we don't consider about q_4 is that the length of link 4 is small and it can be treated as the rigid body. When the robot is trying to reach the target, the existence of gravity would cause a negative effect on which, considering the defined direction of θ_2^* and θ_3^*, we shall give a smaller (absolute value) input on q_2 and q_3 to reach the target.

3.3 Approach to improve the accuracy

After several tests, we find the main difference of the position of the end effect between the real robot and simulated one is that position of the real robot is lower than the simulated one. It is due to the gravity, as what is stated in the section 2.4 and 3.1 above. Therefore, we take the gravity as the major factor which affect the accuracy and try to find an approach to make some compensate for such effect.

Consider that the gravity only comes from the weights of servos and the gripper, since the link part is relatively light. Assume that the gravity just has an effect on θ_2^* and θ_3^*, as what is stated in the section 3.1. Assume that the weights lie on the center of joint 3 (the weight of the servo) and joint 4 (the weights of last two servos and the gripper), which are noted as m_1 and m_2. Consider the plane formed by link 2 and link 3, shown as Figure 3-1, in which θ_2^* and θ_3^* are the theoretical result from the inverse kinematics.

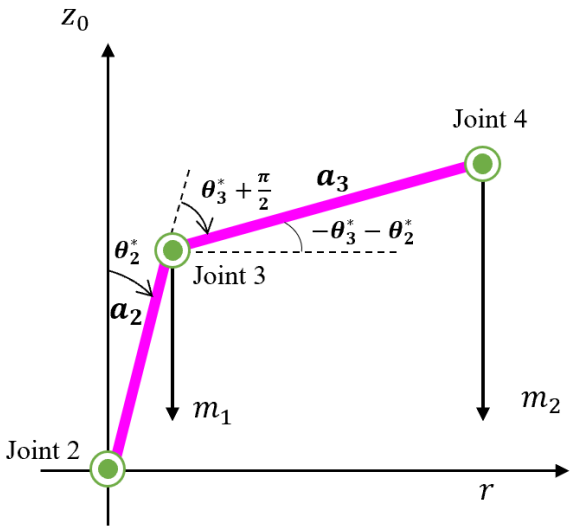

Figure 3-1 Projecting onto the plane formed link 2 and link 3

We have torque caused for joint 2 as

$$\text{Torque}_2 = m_1 a_2 \sin\theta_2^* + m_2(a_2 \sin\theta_2^* + a_3 \cos(\theta_2^* + \theta_3^*))$$

and torque caused for joint 3 as

$$\text{Torque}_3 = m_2 a_3 \cos(\theta_2^* + \theta_3^*)$$

Assuming that the angular error is linear to torque caused for that joint, we have

$$\text{error}_2 = k_2 \text{Torque}_2 = k_2 m_1 a_2 \sin\theta_2^* + k_2 m_2(a_2 \sin\theta_2^* + a_3 \cos(\theta_2^* + \theta_3^*))$$

$$\text{error}_3 = k_3 \text{Torque}_3 = k_3 m_2 a_3 \cos(\theta_2^* + \theta_3^*)$$

in which k_2 and k_3 are error coefficients we defined.

Therefore, our modified input joint variables q_2 and q_3 would be

$$q_2 = \theta_2^* - \text{error}_2$$

$$q_3 = \theta_3^* - \text{error}_3$$

3.4 Data we need and how we use them

As what is stated in section 3.3, there are three coefficients, , $k_2 m_2$, $k_3 m_2$, which we need to obtain in order to implement the proposed approach. Therefore, we need at least three equations to compute these four coefficients.

Choose two set of inputs, (θ_2^1, θ_3^1) and (θ_2^2, θ_3^2). Measure the angular errors under these two inputs noted as $(\text{error}_2^1, \text{error}_3^1)$ and $(\text{error}_2^2, \text{error}_3^2)$, respectively. We have

$$\text{error}_2^1 = k_2 m_1 a_2 \sin\theta_2^1 + k_2 m_2 (a_2 \sin\theta_2^1 + a_3 \cos(\theta_2^1 + \theta_3^1))$$

$$\text{error}_3^1 = k_3 m_2 a_3 \cos(\theta_2^1 + \theta_3^1)$$

$$\text{error}_2^2 = k_2 m_1 a_2 \sin\theta_2^2 + k_2 m_2 (a_2 \sin\theta_2^2 + a_3 \cos(\theta_2^2 + \theta_3^2))$$

$$\text{error}_3^2 = k_3 m_2 a_3 \cos(\theta_2^2 + \theta_3^2)$$

which can be written as

$$\begin{bmatrix} a_2 \sin\theta_2^1 & a_2 \sin\theta_2^1 + a_3 \cos(\theta_2^1 + \theta_3^1) & 0 \\ 0 & 0 & a_3 \cos(\theta_2^1 + \theta_3^1) \\ a_2 \sin\theta_2^2 & (a_2 \sin\theta_2^2 + a_3 \cos(\theta_2^2 + \theta_3^2)) & 0 \\ 0 & 0 & a_3 \cos(\theta_2^2 + \theta_3^2) \end{bmatrix} \begin{bmatrix} k_2 m_1 \\ k_2 m_2 \\ k_3 m_2 \end{bmatrix} = \begin{bmatrix} \text{error}_2^1 \\ \text{error}_3^1 \\ \text{error}_2^2 \\ \text{error}_3^2 \end{bmatrix}$$

In computing, we ignore the input which causes the smallest error in error_3, to make the equation solvable and relatively more accurate. For example, if error_3^2 is the smallest one, the equation would be

$$\begin{bmatrix} a_2 \sin\theta_2^1 & a_2 \sin\theta_2^1 + a_3 \cos(\theta_2^1 + \theta_3^1) & 0 \\ 0 & 0 & a_3 \cos(\theta_2^1 + \theta_3^1) \\ a_2 \sin\theta_2^2 & (a_2 \sin\theta_2^2 + a_3 \cos(\theta_2^2 + \theta_3^2)) & 0 \end{bmatrix} \begin{bmatrix} k_2 m_1 \\ k_2 m_2 \\ k_3 m_2 \end{bmatrix} = \begin{bmatrix} \text{error}_2^1 \\ \text{error}_3^1 \\ \text{error}_2^2 \end{bmatrix}$$

Thus,

$$\begin{bmatrix} k_2 m_1 \\ k_2 m_2 \\ k_3 m_2 \end{bmatrix} = \begin{bmatrix} a_2 \sin\theta_2^1 & a_2 \sin\theta_2^1 + a_3 \cos(\theta_2^1 + \theta_3^1) & 0 \\ 0 & 0 & a_3 \cos(\theta_2^1 + \theta_3^1) \\ a_2 \sin\theta_2^2 & (a_2 \sin\theta_2^2 + a_3 \cos(\theta_2^2 + \theta_3^2)) & 0 \end{bmatrix}' \begin{bmatrix} \text{error}_2^1 \\ \text{error}_3^1 \\ \text{error}_2^2 \end{bmatrix}$$

4 Extra Credit: Calibration

4.1 Compute the coefficients

We choose two set of inputs, in which

$$(\theta_2^1, \theta_3^1) = (0,0)$$

$$(\theta_2^2, \theta_3^2) = (1.4, -\frac{\pi}{2})$$

Then we measure the angular errors of under these two inputs, with a protractor and a long stick for extending the links in order to measure more precisely, shown as Figure 4-1.

Figure 4-1 Measuring the errors

The errors are

$$(\text{error}_2^1, \text{error}_3^1) = (0.01,0.08)$$

$$(\text{error}_2^2, \text{error}_3^2) = (0.14,0.079)$$

Ignoring the smallest error_3 ((error_3^2)) in order to solve the equation, we have

$$\begin{bmatrix} k_2 m_1 \\ k_2 m_2 \\ k_3 m_2 \end{bmatrix} = \begin{bmatrix} a_2 \sin\theta_2^1 & a_2 \sin\theta_2^1 + a_3 \cos(\theta_2^1 + \theta_3^1) & 0 \\ 0 & 0 & a_3 \cos(\theta_2^1 + \theta_3^1) \\ a_2 \sin\theta_2^2 & (a_2 \sin\theta_2^2 + a_3 \cos(\theta_2^2 + \theta_3^2)) & 0 \end{bmatrix}' \begin{bmatrix} \text{error}_2^1 \\ \text{error}_3^1 \\ \text{error}_2^2 \end{bmatrix} = 10^{-3} \times \begin{bmatrix} 0.8509 \\ 0.0534 \\ 0.4271 \end{bmatrix}$$

4.2 Test the approach

Choose target 5 as the test object. Write a function *calibrate*, which takes the outputted q vector from inverse kinematics as the input, outputted the modified version of q. Call the function *lynxServo* with two different version of q, then compare the result. There is a MATLAB code *calibrate.m* listed in the Appendix 1 and attached to the file.

The commands are shown as followed.

```
>> q=[1.2329   -1.1985   -1.1789   -0.2406   -0.9583];
>> calibrate(q)

q_feedback =    1.2329   -1.0683   -1.1211   -0.2406   -0.9583
```

Then we call the *lynxServo* to test the result. The commands are shown as followed.

```
>> q=[1.2329   -1.1985   -1.1789   -0.2406   -0.9583   0];
>> lynxServo(q)
>> q=[1.2329   -1.0683   -1.1211   -0.2406   -0.9583   0];
>> lynxServo(q)
```

There is a video (Video 3) showing the result of the calibration which proves our approach is successful listed in Appendix 2.

Appendix 1 Code Overview

1.	*IK_lynx_xzzhangxinlongz.m*	compute inverse kinematics
2.	*calculateFK.m*	predefined compute forward kinematics
3.	*lynxInitializeHardware.m*	predefined lynx function
4.	*lynxServo.m*	predefined lynx function
5.	*lynxServoSim.m*	predefined lynx function
6.	*lynxStart.m*	predefined lynx function
7.	*lynxVelocityPhysical.m*	predefined lynx function
8.	*lynxServoPhysical.m*	predefined lynx function
9.	*plotLynx.m*	plot simulated configuration
10.	*test.m*	test the method of inverse kinematics
11.	*hit.m*	simulate the inverse kinematics result
12.	*calibrate.m*	calibrate for the feedback control

Appendix 2 Video Overview

1.	https://youtu.be/5vefOq3-cYY	hit target 5
2.	https://youtu.be/1XOnZTJ7aE8	hit target 6
3.	https://youtu.be/akVVOt6z5Tc	calibrate for the feedback control