Required:

title page, table of contents, figure numbers and captions, conclusions

**MSE 4401 ------ Robotic Manipulators**

**Laboratory #1**

**Kinematics**

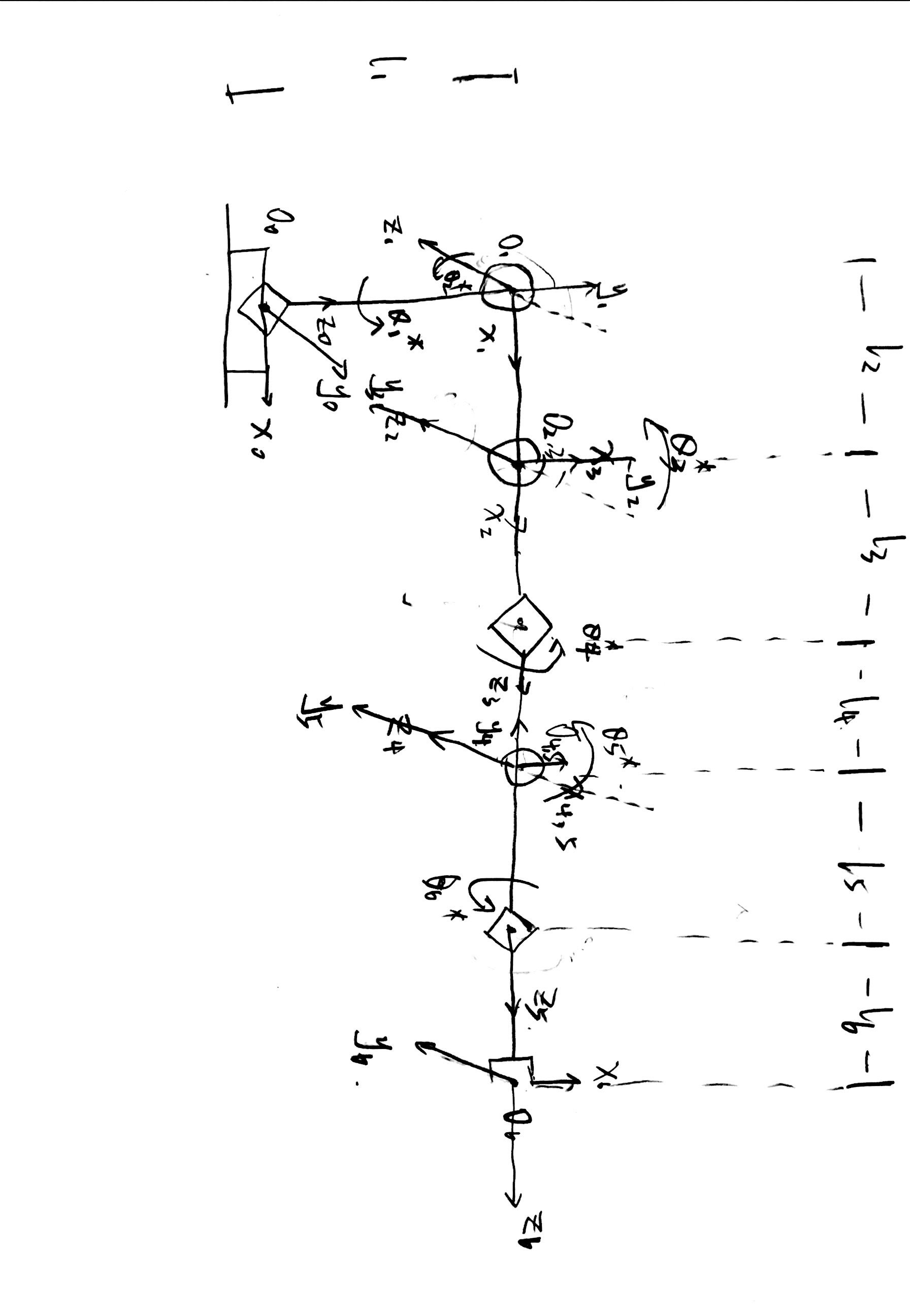
Group 4:

Jianhui Li 250843421

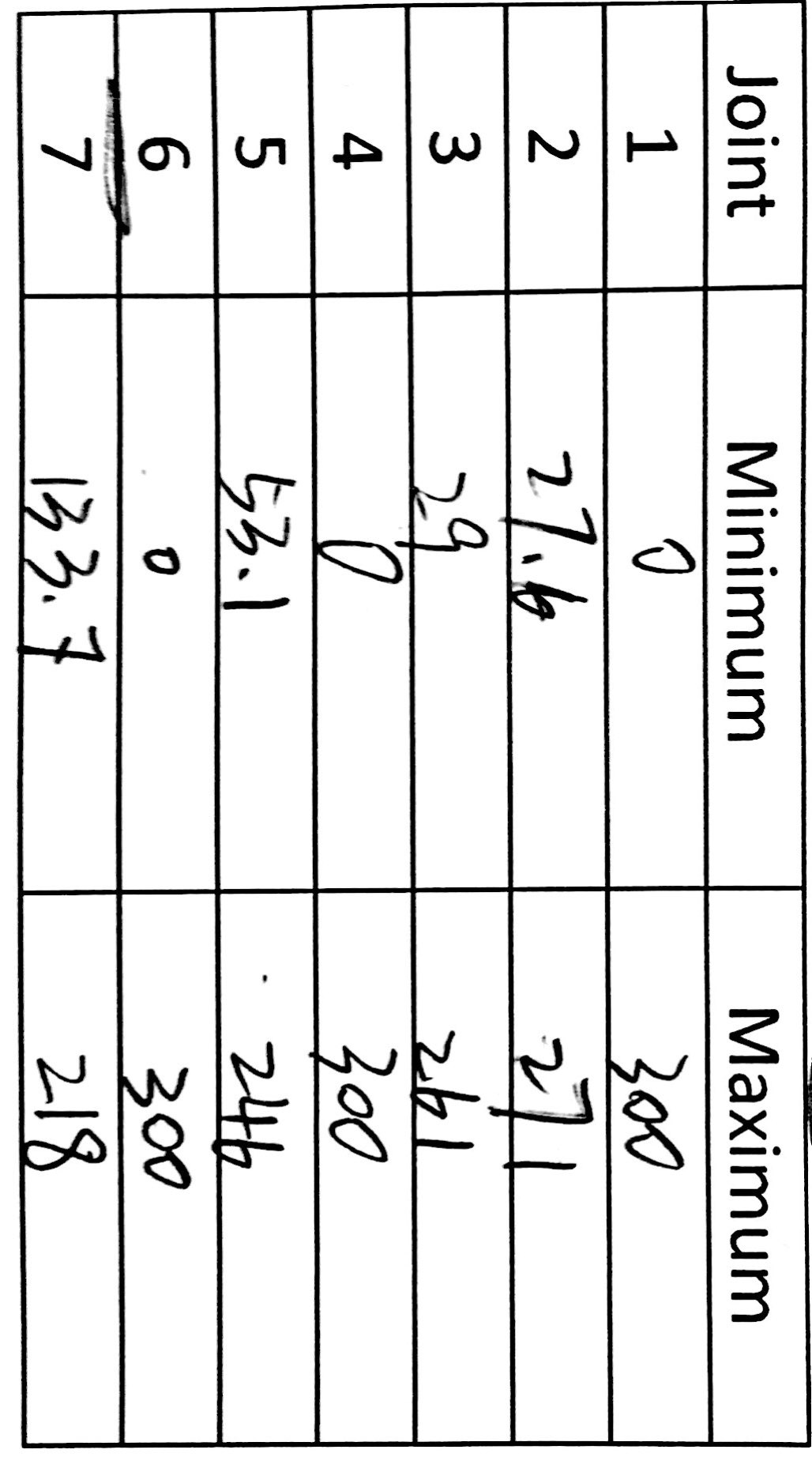
Marioh Richard Lourenco

Part 1:

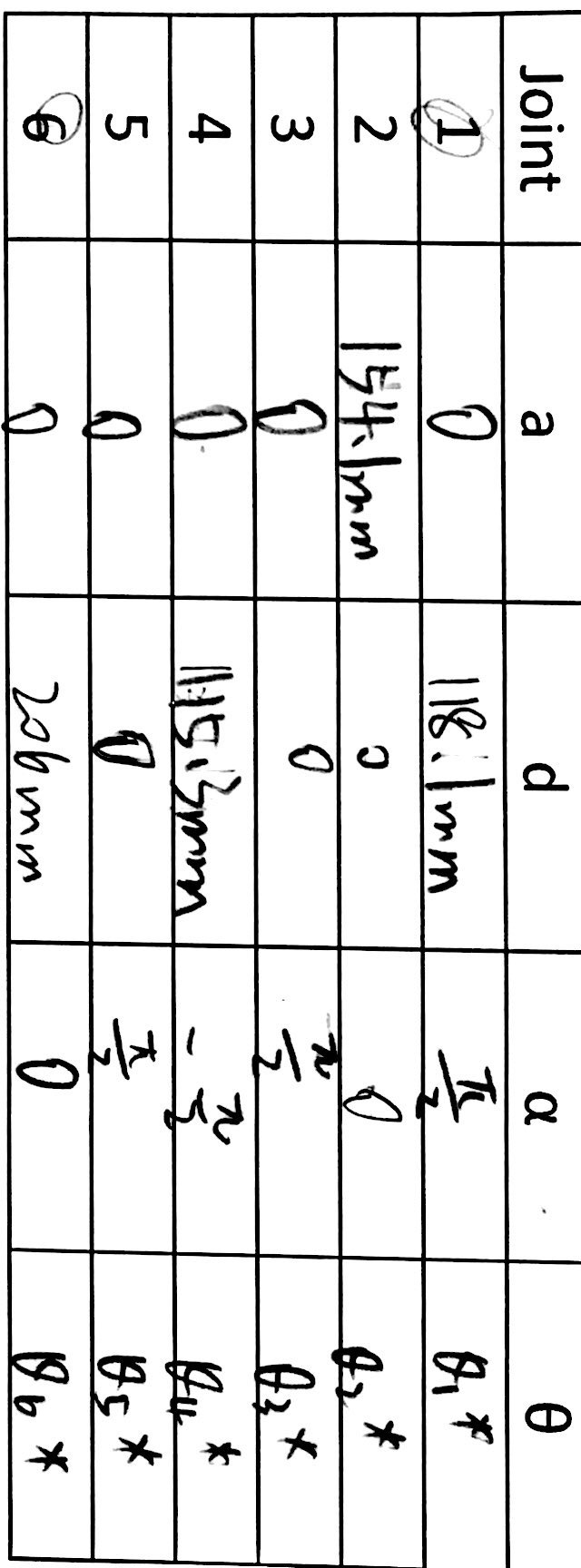
1. 1 Diagram of the configuration of the robot



1.3 Based on the assigned frames, record the estimated joint limits in the table below:



1. DH table.





syms x1 x2 x3 x4 x5 x6;

a = [0,154.1,0,0,0,0];

d =[118.1,0,0,115.3,0,206];

alpha =[pi/2,0,pi/2,-pi/2,pi/2,0];

Theta = [x1,x2,x3,x4,x5,x6];

A1 =[cos(x1) -sin(x1)\*cos(alpha(1)) sin(x1)\*sin(alpha(1)) a(1)\*cos(x1); sin(x1) cos(x1)\*cos(alpha(1)) -cos(x1)\*sin(alpha(1)) a(1)\*sin(x1);0 sin(alpha(1)) cos(alpha(1)) d(1); 0 0 0 1];

A2 =[cos(x2) -sin(x2)\*cos(alpha(2)) sin(x2)\*sin(alpha(2)) a(2)\*cos(x2); sin(x2) cos(x2)\*cos(alpha(2)) -cos(x2)\*sin(alpha(2)) a(2)\*sin(x2);0 sin(alpha(2)) cos(alpha(2)) d(2); 0 0 0 1];

A3 =[cos(x3) -sin(x3)\*cos(alpha(3)) sin(x3)\*sin(alpha(3)) a(3)\*cos(x3); sin(x3) cos(x3)\*cos(alpha(3)) -cos(x3)\*sin(alpha(3)) a(3)\*sin(x3);0 sin(alpha(3)) cos(alpha(3)) d(3); 0 0 0 1];

A4 =[cos(x4) -sin(x4)\*cos(alpha(4)) sin(x4)\*sin(alpha(4)) a(4)\*cos(x4); sin(x4) cos(x4)\*cos(alpha(4)) -cos(x4)\*sin(alpha(4)) a(4)\*sin(x4);0 sin(alpha(4)) cos(alpha(4)) d(4); 0 0 0 1];

A5 =[cos(x5) -sin(x5)\*cos(alpha(5)) sin(x5)\*sin(alpha(5)) a(5)\*cos(x5); sin(x5) cos(x5)\*cos(alpha(5)) -cos(x5)\*sin(alpha(5)) a(5)\*sin(x5);0 sin(alpha(5)) cos(alpha(5)) d(5); 0 0 0 1];

A6 =[cos(x6) -sin(x6)\*cos(alpha(6)) sin(x6)\*sin(alpha(6)) a(6)\*cos(x6); sin(x6) cos(x6)\*cos(alpha(6)) -cos(x6)\*sin(alpha(6)) a(6)\*sin(x6);0 sin(alpha(6)) cos(alpha(6)) d(6); 0 0 0 1];

T=A1\*A2\*A3\*A4\*A5\*A6;

%homogenous transformation from frame 6 to the world frame 0

round(T) % ans

Output matric:

homogenous transformation from frame 6 to the world frame 0

????

Part 2:

**Forward Kinematics**

1. Snippet of code that generates your Link objects for the toolbox

%lab part 2.1 begin

L1 = Link('d', 118.1, 'a', 0, 'alpha', pi/2);

L2 = Link('d', 0, 'a', 154.1, 'alpha', 0);

L3 = Link('d', 0, 'a', 0, 'alpha', pi/2);

L4 = Link('d', 115.3, 'a', 0, 'alpha', -pi/2);

L5 = Link('d', 0, 'a', 0, 'alpha', pi/2);

L6 = Link('d', 206, 'a', 0, 'alpha', 0);

bot = SerialLink([L1 L2 L3 L4 L5 L6], 'name', 'lab1');

%robot's end-effector

1. Snippet of code that generates your T matrix given a configuration

% configuration 1

bot.fkine([0 pi/2 0 0 0 0]);

bot.plot([0 pi/2 0 0 0 0]);

Robot.sendPosition([150 150 60 150 150 150 150]);

ans %use ans to check and show the ans

% configuration2

 bot.fkine([0 pi/2 pi/2 0 0 0]);

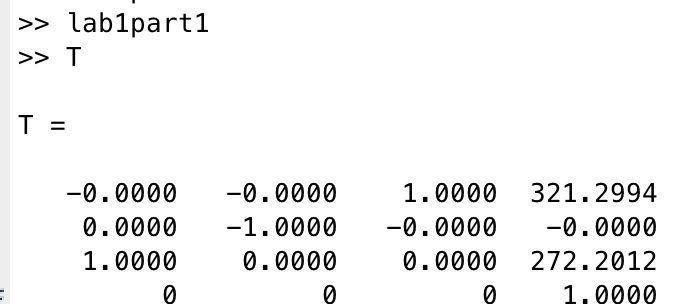
 bot.plot([0 pi/2 pi/2 0 0 0]);

 Robot = MSE4401BOT(1234,4321);

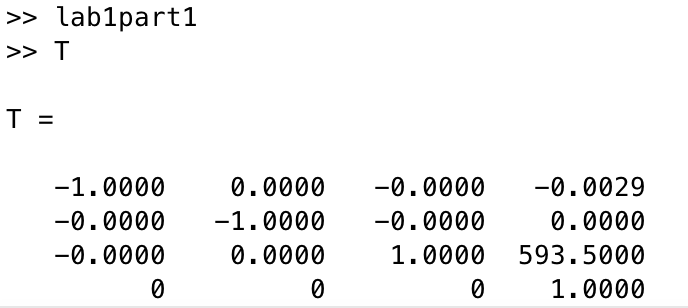
 Robot.sendPosition([150 150 150 150 150 150 150]);

1. By using my forward kinematic code:

Configuration 1:

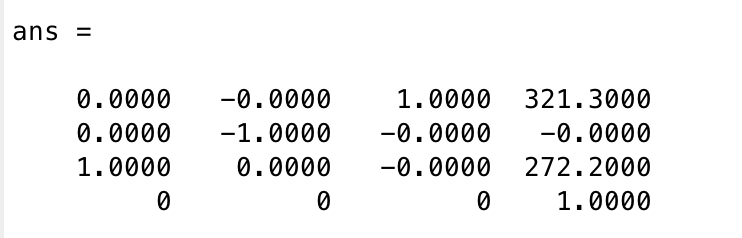


Configuration 2:

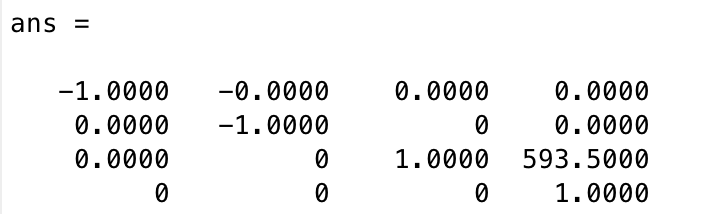


By using bot.fkine(), the get the result below:

Configuration 1:

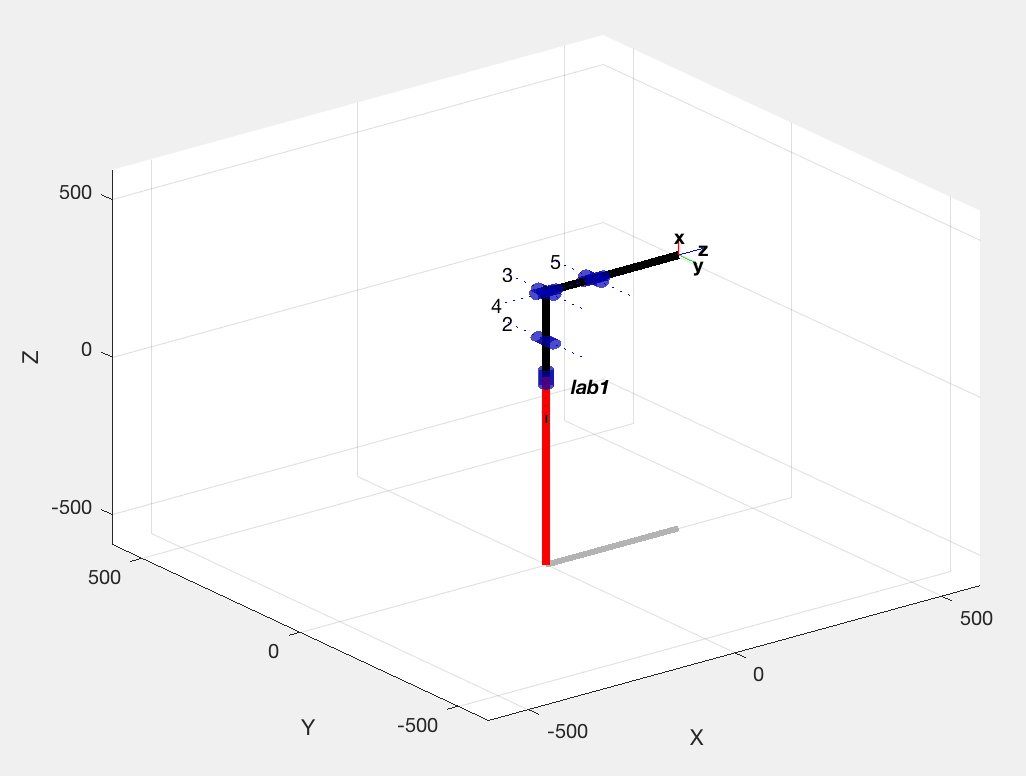


Configuration 2:

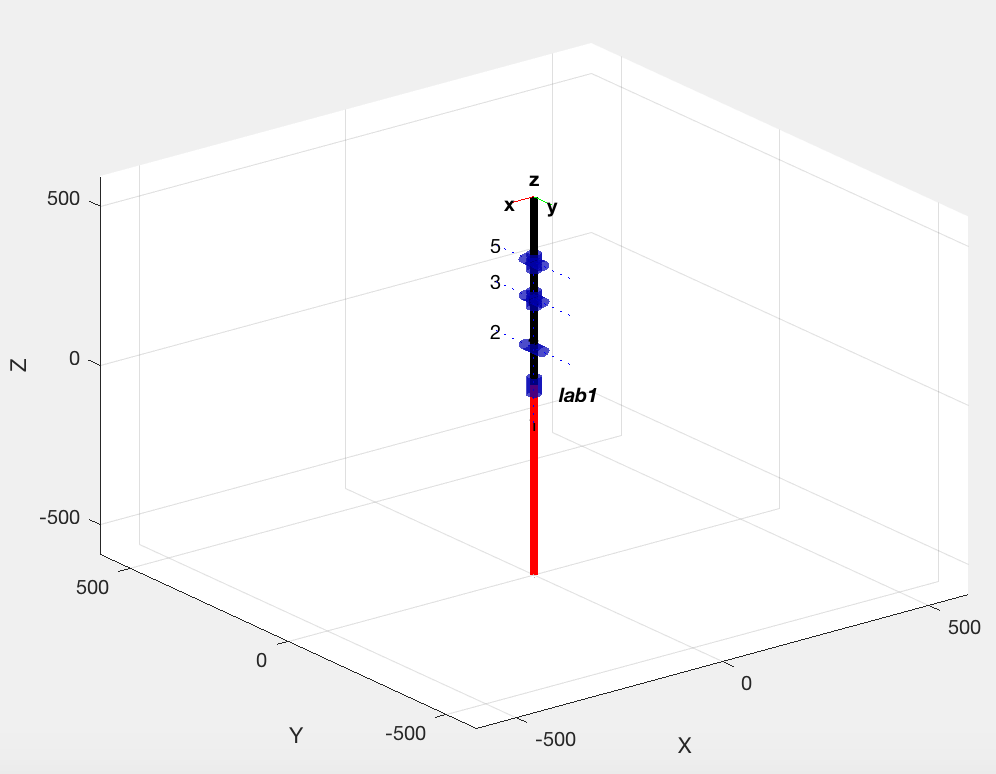


4. both configurations tested plotted using bot.plot().

configuration 1:



configuration 2:



5. Proof that you drove the cruss crawler to such configurations and that the robot pose matches  your bot.plot() – measurements taken which validate the end effector position relative to the base frame and how it differs from your T matrix. MANY PEOPLE FORGOT TO DO THIS.

comment: the offset of each joint (unit degree)

x1: +150 x2: +60 x3: +60 x4: +150 x5: +150 x6: +150

for configuration 1 : Robot.sendPosition([150 150 60 150 150 150 150]);

for configuration 2:  Robot.sendPosition([150 150 150 150 150 150 150]);

insert picture:

**Inverse Kinematics**

1. The first
2. Q\_in -> your forward kinematics code -> T\_in -> invkinCC -> Q\_out -> your forward kinematics  code -> T\_out.

Configuration 1 by using invkinCC

%configuration 1

q\_in=[0 1 0 pi/2 0 0];

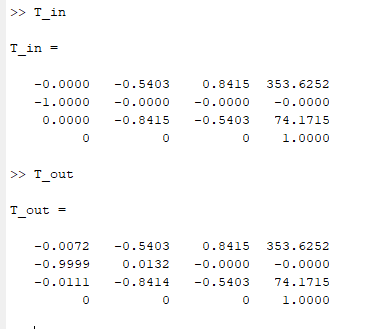
%homogenous transformation from frame 6 to the world frame 0

T\_in = HomoT(q\_in);

q\_out = invkinCC(T\_in);

T\_out=HomoT(q\_out);

bot.plot(q\_out);



Configuration 1 by using bot.ikine and bot.fkine

L1 = Link('d', 118.1, 'a', 0, 'alpha', pi/2);

L2 = Link('d', 0, 'a', 154.1, 'alpha', 0);

L3 = Link('d', 0, 'a', 0, 'alpha', pi/2);

L4 = Link('d', 115.3, 'a', 0, 'alpha', -pi/2);

L5 = Link('d', 0, 'a', 0, 'alpha', pi/2);

L6 = Link('d', 206, 'a', 0, 'alpha', 0);

bot = SerialLink([L1 L2 L3 L4 L5 L6], 'name', 'lab1');

%configuration 1

q\_in=[0 1 0 pi/2 0 0];

T\_in=bot.fkine(q\_in);

disp(T\_in)

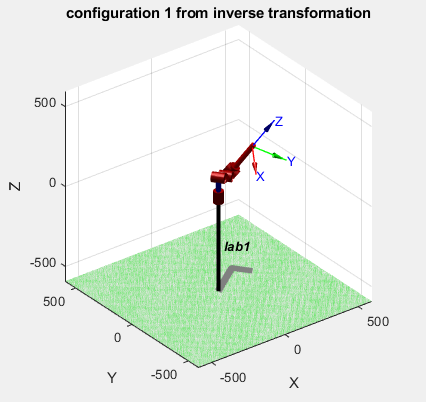
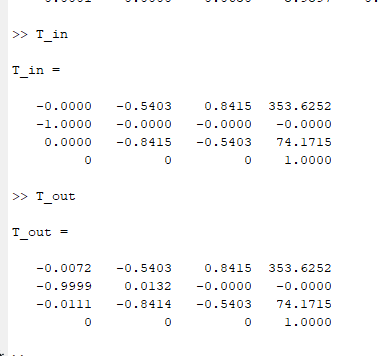
q\_out=bot.ikine(T\_in, 'pinv');

disp(q\_out);

T\_out=bot.fkine(q\_out);

bot.plot(q\_out);

title('configuration 1 from inverse transformation');



Configuration 2 by using invkinCC

% %configuration 2

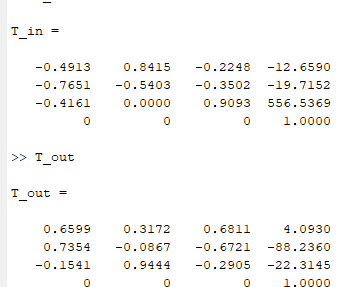
q\_in=[1 pi/2 1 0 1 0];

%homogenous transformation from frame 6 to the world frame 0

T\_in = HomoT(q\_in);

q\_out = invkinCC(T\_in);

T\_out=HomoT(q\_out);



Configuration 2 by using bot.ikine and bot.fkine

L1 = Link('d', 118.1, 'a', 0, 'alpha', pi/2);

L2 = Link('d', 0, 'a', 154.1, 'alpha', 0);

L3 = Link('d', 0, 'a', 0, 'alpha', pi/2);

L4 = Link('d', 115.3, 'a', 0, 'alpha', -pi/2);

L5 = Link('d', 0, 'a', 0, 'alpha', pi/2);

L6 = Link('d', 206, 'a', 0, 'alpha', 0);

bot = SerialLink([L1 L2 L3 L4 L5 L6], 'name', 'lab1');

%

% %configuration 1

% q\_in=[0 1 0 pi/2 0 0];

% %configuration 2

q\_in=[1 pi/2 1 0 1 0];

% %homogenous transformation from frame 6 to the world frame 0

% T\_in = HomoT(q\_in);

%

% q\_out = invkinCC(T\_in);

%

% T\_out=HomoT(q\_out);

T\_in=bot.fkine(q\_in);

disp(T\_in)

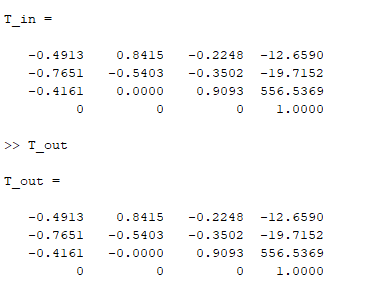
q\_out=bot.ikine(T\_in, 'pinv');

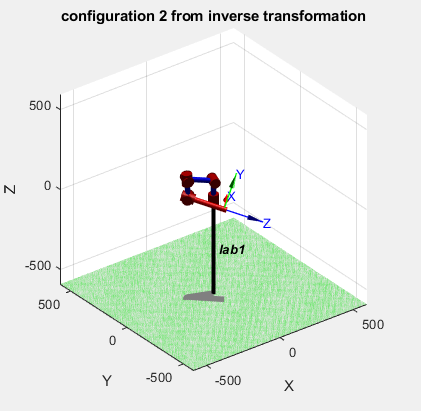
disp(q\_out);

T\_out=bot.fkine(q\_out);

bot.plot(q\_out);

title('configuration 1 from inverse transformation');





1. T\_out and T\_in should always be the same!!! If these are not the same YOU FAIL 😉. But you  must explain if there are any differences between the Q\_out and Q\_in. This should help you explain what some differences are between the invkinCC and bot.fkine() functions.

???

Part 3:

1. Snippet of code that generates your Jacobian matrix and the computation of the determinant.
2. Using the computation to figure out the equation which makes the Jacobian singular (det(J) = 0)
3. Form this I need to see if you can identify many possible joint configurations, Q, which are  singular. I need to see the bot.plot(Q) for each singularity you find.