Definitions

In this document "real robot" means the actual physical Dobot robot. "Model" means the simulated version that is created and plotted in Matlab. There are some differences.

For joint limits, "actual" means that the robot can physically be moved (not necessarily move itself) to that value. However, since the IMUs fail on the vertical I have create "suggested" limits, which you should use. As shown latter the "suggested" limits will reduce where the robot can move to, but it should be more successful.

Model vs Real Robot

For the model robot the joint angles are q = [q1,q2,q3,q4,q5].

For the real robot they are named as follows

Model joint name	Suggested Offset (i.e. 'zero' position)	Joint name on Real Dobot robot	Relationship (real to model)	Actual vs Suggested limits
q1	0	BaseAngle	BaseAngle = q1	[-135,135]
q2	-pi/2	RearArmAngle	RearArmAngle = q2	Actual = [-5 85], suggested = [5,80]
q3	pi/2 – q2 (I cannot see how to do this in toolbox, so 0)	ForeArmAngle	ForeArmAngle= q3 - pi/2+q2 q3 = pi/2-q2+ ForeArmAngle	Actual (Real) = [-10,95], (model)=[-5,190] Suggested (Real) = [-5,85], (model)=[15,170] **Note: Depends upon value of q2**
q4	0	N/A (cannot be controlled)	N/A	[-pi/2,pi/2]
q5	0	ServoAngle	ServoAngle = q5	Actual [-pi/2,pi/2] Suggested deg2rad([-85,85])

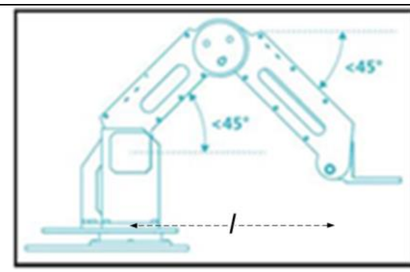
On the real robot, actual joint limits less than 0 or greater than 90 cause problems with IMU and the robot tends to freeze if started (or moved forcefully) into or nearby one of these poses. Therefore the suggested joint limits are shown above

Dobot Model Geometry

In the Dobot manual the figure shown. However, when controlling the second joint, zero position is point straight up and this changes to positive as the joint rotates clockwise (from this view point).

So theta (θ) is used for the angles inside the 5 sided shape (angles add to 540 = 3pi), also for joint 3 (q3) there is a "real" and a "model" value and they are not the same. It looks a like this

Note that in this document the l value is only to the center of joint 4, not to the end effector



Relationships

Using $q_{3,real}$

$$3\pi = \theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5$$

$$3\pi = \frac{\pi}{2} + \pi - q_2 + \frac{\pi}{2} + q_2 - q_{3,real} + \pi - q_4 + \frac{\pi}{2}$$

$$0 = \frac{\pi}{2} - q_2 + q_2 - q_{3,real} - q_4$$

$$q_4 = \frac{\pi}{2} - q_{3,real}$$

Note how $q_{3,model} = \frac{\pi}{2} - q_2 + q_{3,real}$

thus $q_{3,real} = q_{3,model} - \frac{\pi}{2} + q_2$

So $q_4 = \frac{\pi}{2} - (q_{3,model} - \frac{\pi}{2} + q_2)$

And $q_4 = \pi - q_2 - q_{3,model}$

Alternatively,

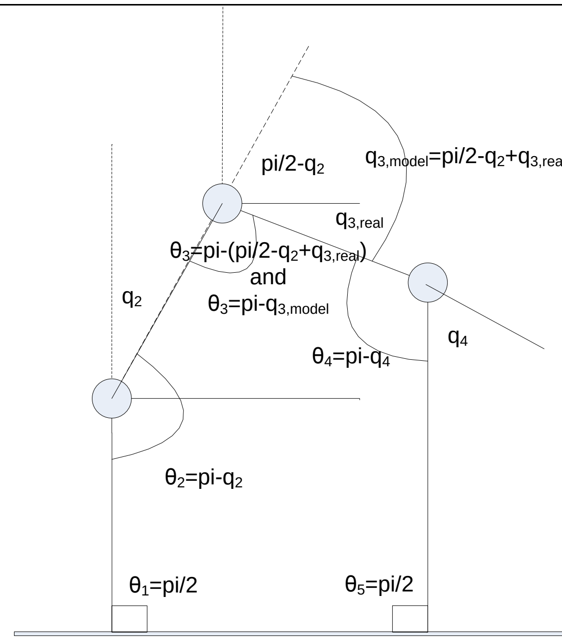
with $q_{3,model}$

$$3\pi = \theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5$$

$$3\pi = \frac{\pi}{2} + \pi - q_2 + \pi - q_{3,model} + \pi - q_4 + \frac{\pi}{2}$$

$$0 = \pi - q_2 - q_{3,model} - q_4$$

$$q_4 = \pi - q_2 - q_{3,model}$$



PlotLimits

Give the following constants

```
properties (Constant)
  %> Joint 2: Actual vs Suggested real joint limits actual joint limits less than 0 cause
problems with IMU (officially
  qlimDegs = [-5,85], suggest it is better to be [5,80] )
  actualRealQ2lim = deg2rad([-5,85]);
  suggestedRealQ2lim = deg2rad([5,80]);

  %> Joint 3: Actual vs Suggested real joint limits actual joint limits less than 0 (or more than
90) cause problems with IMU (officially q2limDegs = -10 to 95 )
  actualRealQ3lim = deg2rad([-10,95]);
  suggestedRealQ3lim = deg2rad([5,85]);
end
```

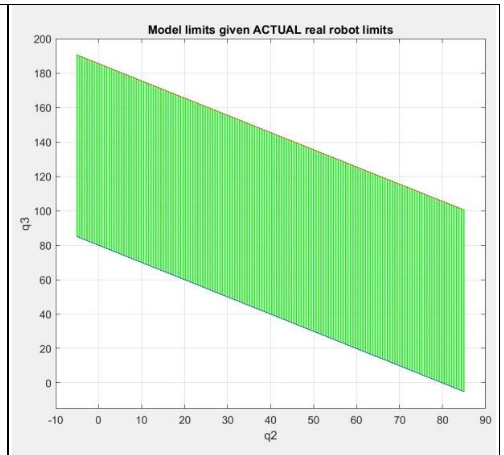
We can plot the joint limits of q3 as q2 changes for both the actual and suggested limits of the real joints 2 and 3 (i.e. RearArmAngle and ForeArmAngle)

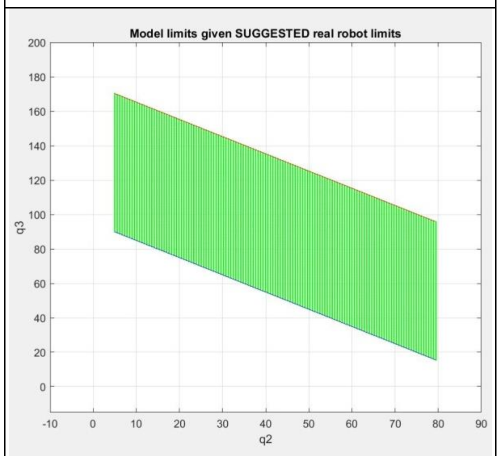
```
function PlotLimits(self)
  titleStr = {'Model limits given ACTUAL real robot limits','Model
limits given SUGGESTED real robot limits'};
  qlimActualAndSuggested = {self.model.qlim,self.model.qlim};
  qlimActualAndSuggested{1}(2,:) = self.actualRealQ2lim;
  qlimActualAndSuggested{1}(3,:) = self.actualRealQ3lim;
  qlimActualAndSuggested{2}(2,:) = self.suggestedRealQ2lim;
  qlimActualAndSuggested{2}(3,:) = self.suggestedRealQ3lim;

  fig_h = figure;
  for limitsIndex = 1:size(titleStr,2)
    clf(fig_h);
    data = [];
    lowerLimit = [];
    upperLimit = [];

    qlim = qlimActualAndSuggested{ limitsIndex };

    for q2 = qlim(2,1):0.01:qlim(2,2)
        for theta3 = qlim(3,1):0.01:qlim(3,2)+0.01
            q3 = pi/2 - q2 + theta3;
            data = [data;q2,q3]; %#ok<AGROW>
            if theta3 <= qlim(3,1)
                lowerLimit = [lowerLimit;q2,q3]; %#ok<AGROW>
            elseif qlim(3,2) <= theta3
                upperLimit = [upperLimit;q2,q3]; %#ok<AGROW>
            end
        end
    end
    plot(rad2deg(data(:,1)),rad2deg(data(:,2)),'g.');
    hold on;
    plot(rad2deg(lowerLimit(:,1)),rad2deg(lowerLimit(:,2)),'b');
    plot(rad2deg(upperLimit(:,1)),rad2deg(upperLimit(:,2)),'r');
    title(titleStr{limitsIndex})
    set(gca,'fontSize',15)
    xlabel('q2')
    ylabel('q3')
    grid on;
    axis([-10,90,-15,200]);
    drawnow();
    img = getframe(gcf);
    imwrite(img.cdata, [titleStr{limitsIndex}, '.jpg']);
  end
  close(fig_h);
end
```



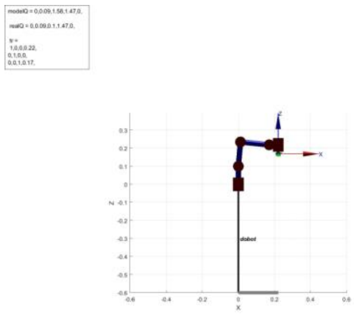


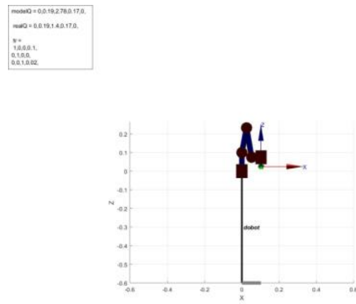
Robot model in various poses (in configuration space)

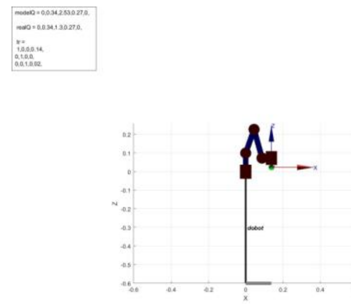
With a side-on view here are some pictures of the Dobot robot model

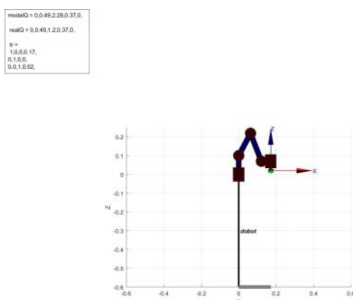
Dobot moving through Cartesian space with ikcon https://goo.gl/ZLQXaX

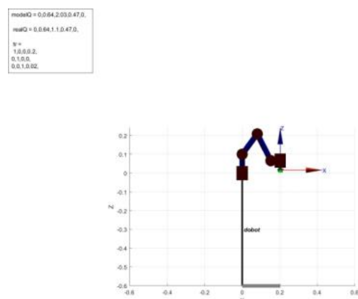
Here are a few samples

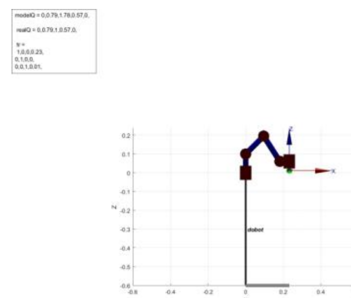


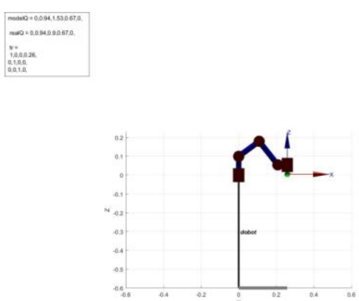


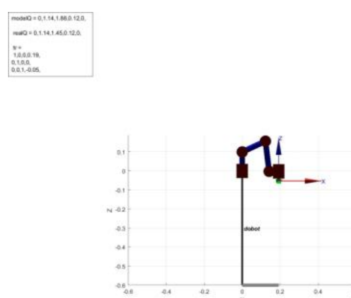


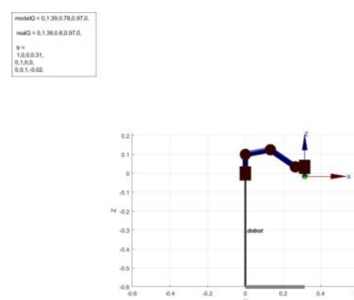










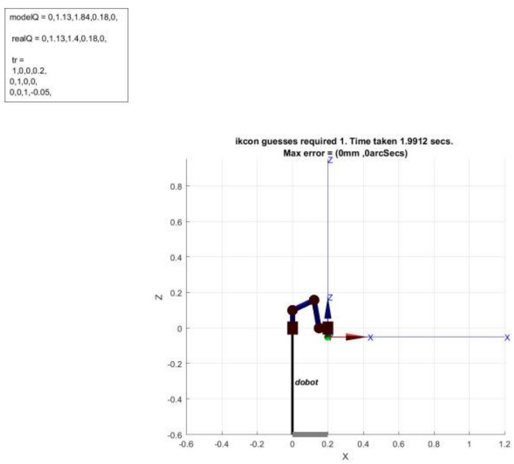


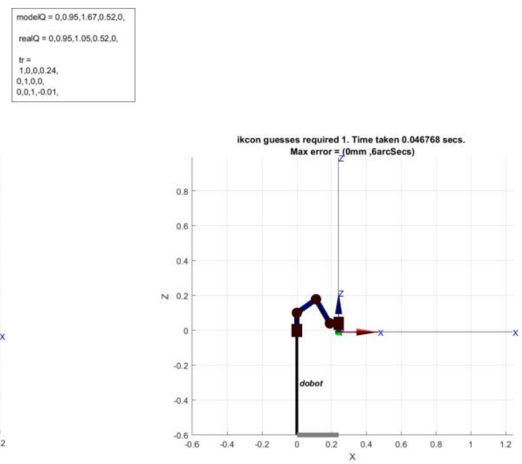
Robot model in various poses (in Cartesian space)

Dobot moving through Cartesian space with ikcon https://goo.gl/ZLQXaX

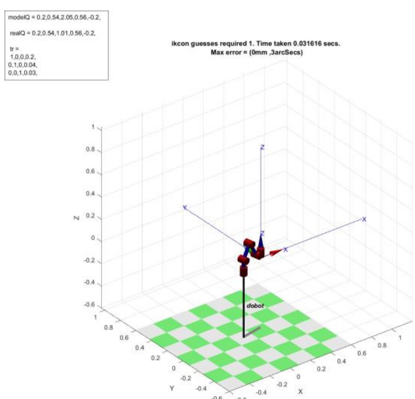
In these cases the x,y, and z are varied between some reasonable bounds, and the roll pitch yaw is kept as the identity matrix. Ikcon is used to solve given an initial guess of q_{model} = [0,pi/4,pi/2,pi/4,0]

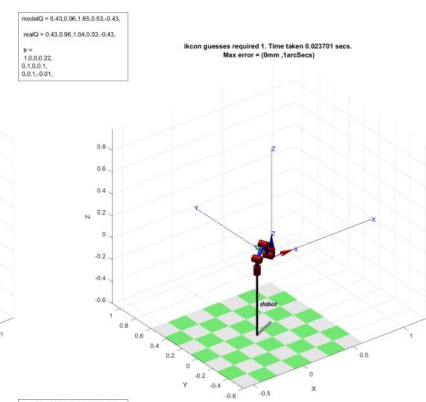
Here are a few samples from side on

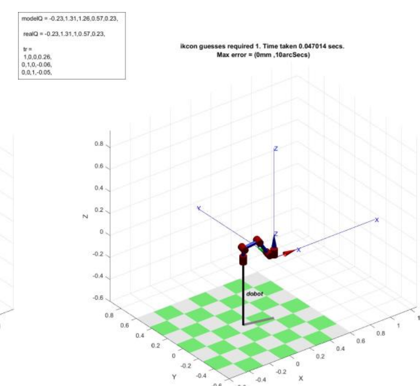


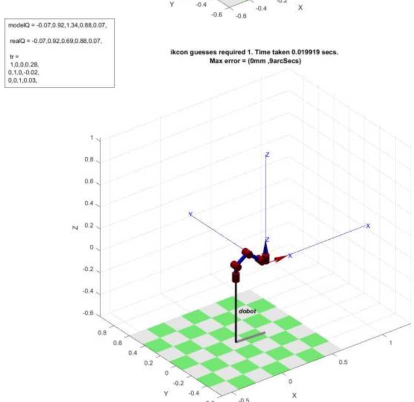


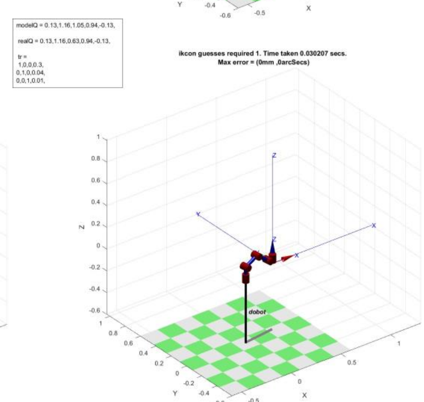
Here are a few samples that change y as well and view from another viewpoint

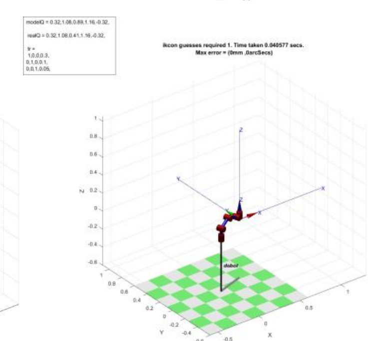






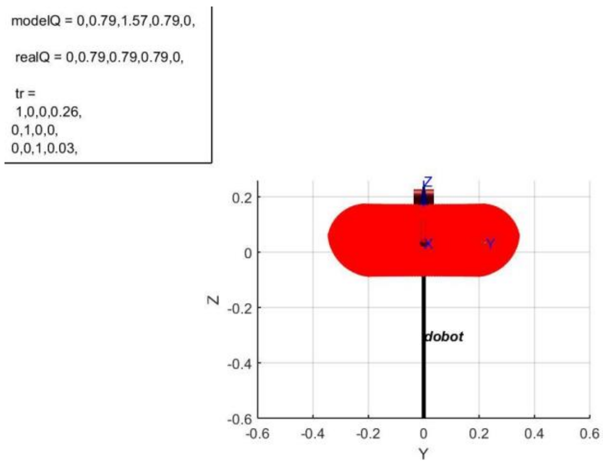


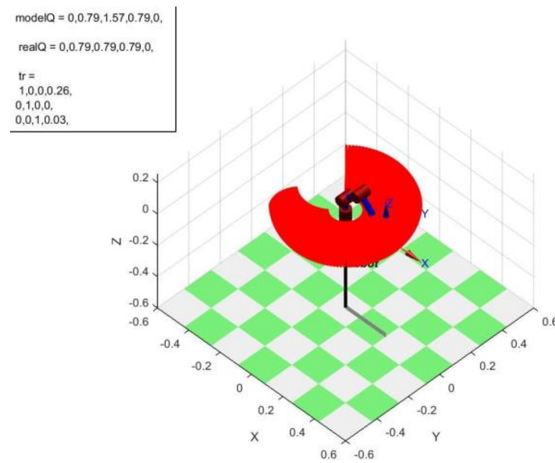


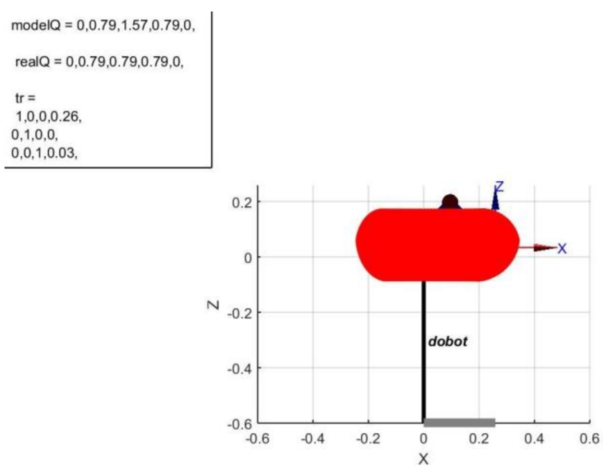


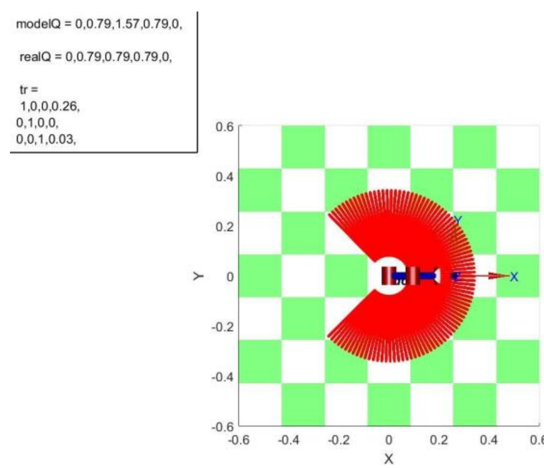
Workspace of Dobot

Using the suggested joint limits of the Dobot the following is the approx workspace of the Dobot



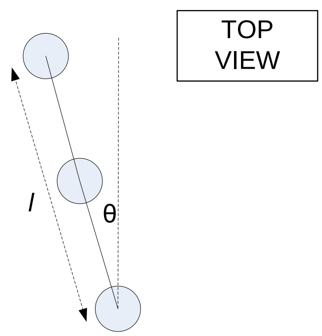


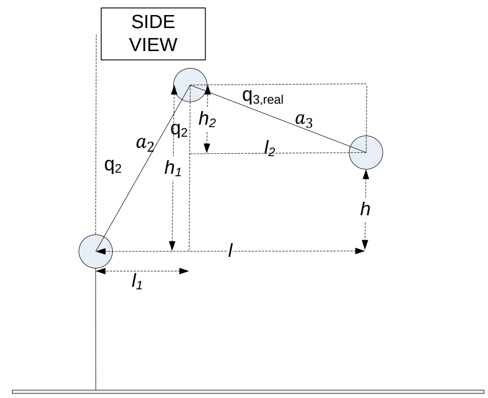




Ikine f(θ,l,h)={q_1,q_2,$q_{3,real}$}

When given θ,*l h* we want to determine the joint angles q_1,q_2,$q_{3,real}$ to the 4th joint.
Note that the 4th joint is constant offset distance from the end effector in both z_e and polar *l* (if using transl(x_e, y_e, z_e) to get to the end effector then the x_e and y_e offset will depend upon q_1)

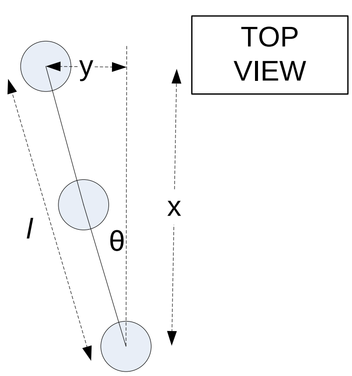


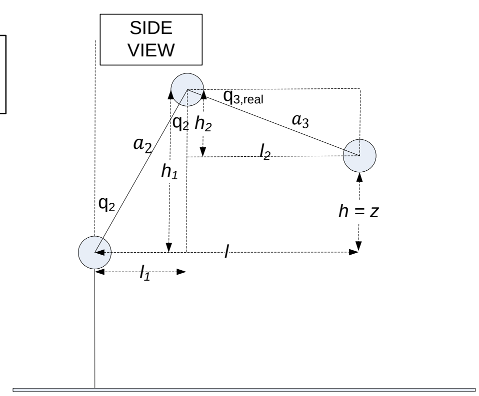


Given link lengths a_2 and a_3 Polar $q_1 = \theta$	Relationships we know $l = l_1 + l_2 = a_2 \sin(q_2) + a_3 \cos(q_{3,real})$ $h = h_1 - h_2 = a_2 \cos(q_2) - a_3 \sin(q_{3,real})$

Ikine f(x,y,z)={θ,*l h*}

When given an [x,y,z] position, we want to determine the polar coordinates [θ,*l h].* Note that the length, l is to the centre of the 4th joint, not the centre of the end effector. For the suction cap, the distance is another $c = 0.05$ to the Dobot's end-effector/head.





Given link lengths a_2 and a_3 and polar $\theta = q_1$	Relationships we know $l^2 = x^2 + y^2$ $\tan(\theta) = \frac{y}{x}$ $h = z$

So $l = \sqrt{x^2 + y^2}$ and $\theta = \tan^{-1}\frac{y}{x}$

Ikine f(x,y,z)={q_1,q_2,q_3}

When given an [x,y,z] position, we want to determine the joint angles q_1,q_2,q_3 to the 4th joint.
Note that the 4th joint is constant offset distance from the end effector in both z_e and polar *l* (if using transl(x_e, y_e, z_e) the x_e and y_e offset will depend upon q_1). For the suction cap, the distance is another $c = 0.05$ to the Dobot's end-effector/head.

From above we know

$l = a_2 \sin(q_2) + a_3 \cos(q_{3,real})$ $h = a_2 \cos(q_2) - a_3 \sin(q_{3,real})$ $\theta = q_1 = \tan^{-1}\left(\frac{y}{x}\right)$	$l = \sqrt{x^2 + y^2}$ $\theta = \tan^{-1}\left(\frac{y}{x}\right)$ $h = z$

$$a_2 \sin(q_2) + a_3 \cos(q_{3,real}) = \sqrt{x^2 + y^2}$$

$$a_2 \cos(q_2) - a_3 \sin(q_{3,real}) = z$$

Solving in Matlab could be done in the following way

```
syms x y z q2 q3 a2 a3;

f1 = a2*sin(q2)+a3*cos(q3) == sqrt(x^2+y^2)
f2 = a2*cos(q2)-a3*sin(q3) == z

solution = solve([f1,f2],[q2,q3]);
solution.q2
solution.q3
```

Which will give 2 solutions in terms of $[x, y, z, a_2, a_3]$. Actually q_2 and q_3 are constrained such that only one solution is possible. If you thought the second solution is correct and determined that $[x, y, z, a_2, a_3]$ are all 1 units (<u>note that they are not really</u>) you could substitute in using Matlab

```
subs(solution.q2(2),{x,y,z,a2,a3},{1,1,1,1,1})
```

which would give

```
-2*atan((3*(3^(1/2)-2))/(5*(2*2^(1/2)+3))-(2*2^(1/2))/5+(2*2^(1/2)*(3^(1/2)-2))/(5*(2*2^(1/2)+3))+2/5)
= 0.4317rads
```

Then to solve for $q_{3,real}$

$$a_2 \cos(q_2) - a_3 \sin(q_{3,real}) = z \qquad \text{from above}$$

$$1 \times \cos(0.4317) - 1 \times \sin(q_{3,real}) = 1$$

$$\sin(q_{3,real}) = \cos(0.4317) - 1$$

$$q_{3,real} = \sin^{-1}(\cos(0.4317) - 1) = -0.0919$$

Or in this case solution 2 and solved in Matlab with

```
subs(solution.q3(2),{x,y,z,a2,a3},{1,1,1,1,1})
```

which gives the same result

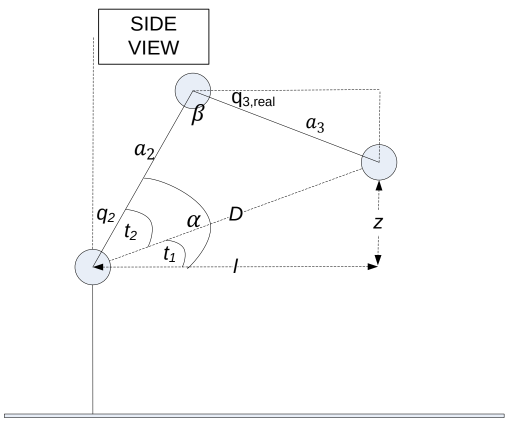
`2*atan((3^(1/2) - 2)/(2*2^(1/2) + 3))` $= -0.0919 rads$

Note: the inputs were nonsense and just for the sake of example, so these outputs values are also nonsense.

Method 2: Ikine f(x,*y,z*)=*{q_1,q_2,q_3}*

Thanks to James Poon for this method.

When given an [x,y,z] position of the 4th joint, we want to determine the joint angles q_1, q_2, q_3 .



$$l = \sqrt{x^2 + y^2}\ ,\ D = \sqrt{l^2 + z^2}$$

$$t_1 = \tan^{-1}\left(\frac{z}{l}\right),\ \ t_2 = \cos^{-1}\left(\frac{a_2^2 + D^2 - a_3^2}{2a_2 D}\right)$$

$$\alpha = t_1 + t_2\ ,\ \beta = \cos^{-1}\left(\frac{a_2^2 + a_3^2 - D^2}{2a_2 a_3}\right)$$

$$q_2 = \frac{\pi}{2} - \alpha\ ,\ q_{3,real} = \pi - \beta - \alpha$$