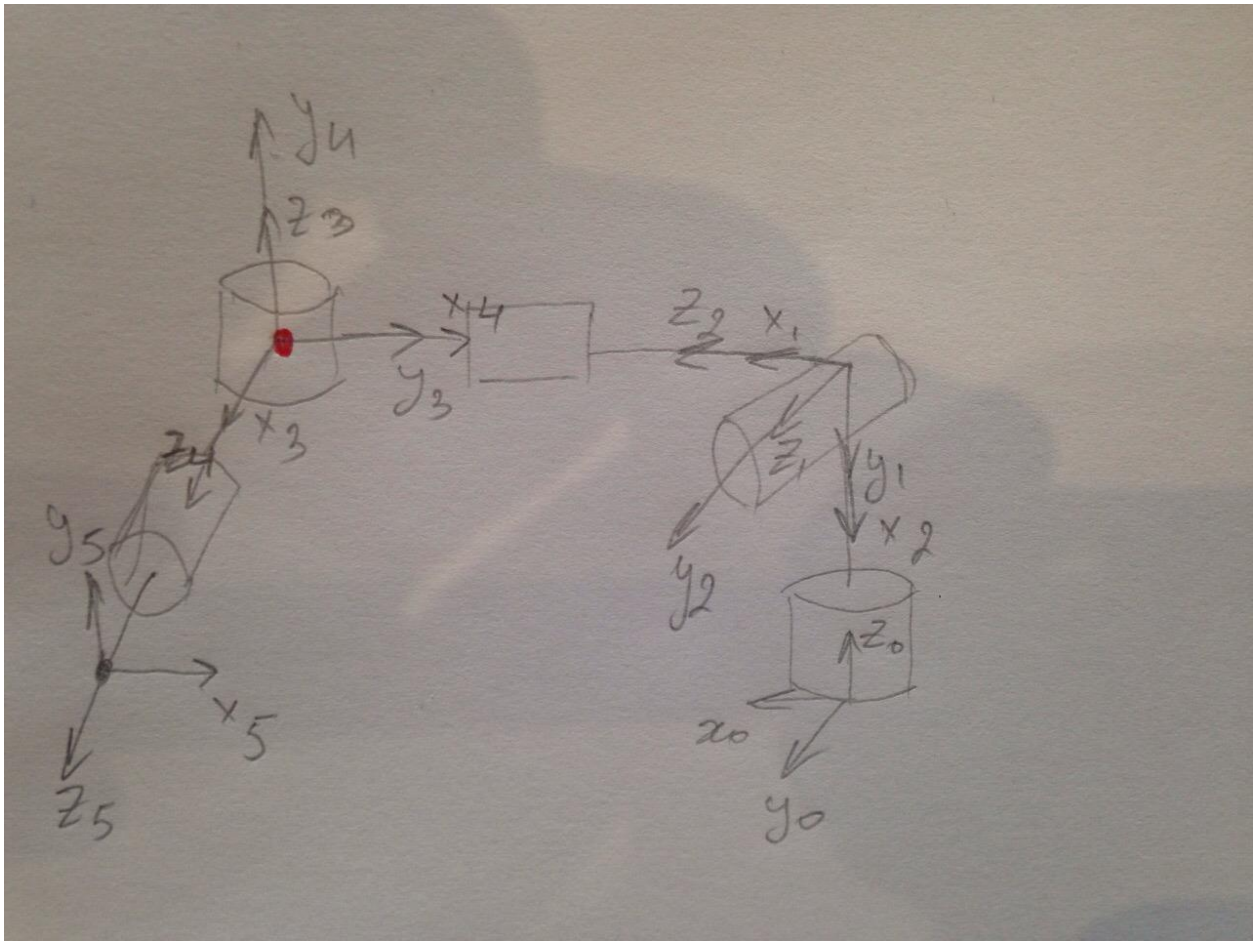
Description of arm of AR-601

Consists of 5 links and 5 joints. All the lengths are equal to 1. Task: to create programs for solving forward and invers kinematics tasks.

Kinematic scheme

The primary orientation of robot was change in case to make easier creation in the MatLab. So Link1 oriented from floor to up, Link 2 and 3 from right to left, Link 4 and 5 towards us.

Finaly, it looks like that:



How were oriented axis:

z – is axis around which rotation is

x – perpendicular and for z_n and for z_{n-1} and intersects the z_{n-1}

y – perpendicular for z and y.

Iteration of joints begins from 0. Zero frame and for joint and for world.

Lengths of links were equaled to 1 for easier understanding.

DH-params:

Joint number	theta	alpha	a	d
1	Theta1	-pi/2	0	L1
2	pi/2+theta2	pi/2	0	0
3	pi/2+theta3	-pi/2	0	L2+L3
4	pi/2+theta4	pi/2	0	0
5	Theta5	0	0	L4+L5

Where theta – angle around z_{n-1} between x_{n-1} and x_n

Alpha - angle around x_n between z_{n-1} and z_n

a – distance between origins along x – axis

d - distance between origins along z – axis

Forward kinematics.

The result should be transformation matrix T_0^6 which connects world frame with the last link.

$T_0^6 = T_0^1 * T_1^2 * T_2^3 * T_3^4 * T_4^5 * T_5^6$ - all these matrixes shows transformation for each joint to previous one.

Let’s show how to count T_0^1 .

Firstly, count rotation matrix: $R_0^1 = Rj_1 * Ra_1$

Rj_1 – rotation of joints

Ra_1 – rotation of axis w.r.t. previous axes

Angle is changing around z – axis, so Rj_1 = [cos(alpha1) -sin(alpha1) 0

sin(alpha1) cos(alpha1) 0

0 0 1]

Axis x_1 oriented the same as x_0, axis y_1 oriented the opposite as z_0, axis z_1 oriented the same as y_0, so Ra_1 = [1 0 0

0 0 1

0 -1 0]

Multiplication is done in Matlab.

The same way we find other rotation matrixes and multiply them to have R_0^6 which can be seen in Matlab in common decision or be calculated in numbers.

Secondly, find translation matrix.

Each row of 3x1 matrix is a translation along x, y, z of link is its coordinate frame w.r.t. previous coordinate frame. For example, Tr_0^1 = [0 0 L1]

In the end we have transformation matrix:

T = [R(1,1) R(1,2) R(1,3) Tr(1)

R(2,1) R(2,2) R(2,3) Tr(2)

R(3,1) R(3,2) R(3,3) Tr(3)

0 0 0 1]

Finally, we multiply all the T to get T_0^6 – that is the solution of forward kinematics.

Inverse kinematics

Here we have final T

$$T = \begin{bmatrix} T(1,1) & T(1,2) & T(1,3) & T(1,4) \\ T(2,1) & T(2,2) & T(2,3) & T(2,4) \\ T(3,1) & T(3,2) & T(3,3) & T(3,4) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and our purpose is to find angles from theta0 to theta5.

Firstly, we should divide arm into 2 parts: 3 first links and 2 last (as far as we have 5 DoF arm).

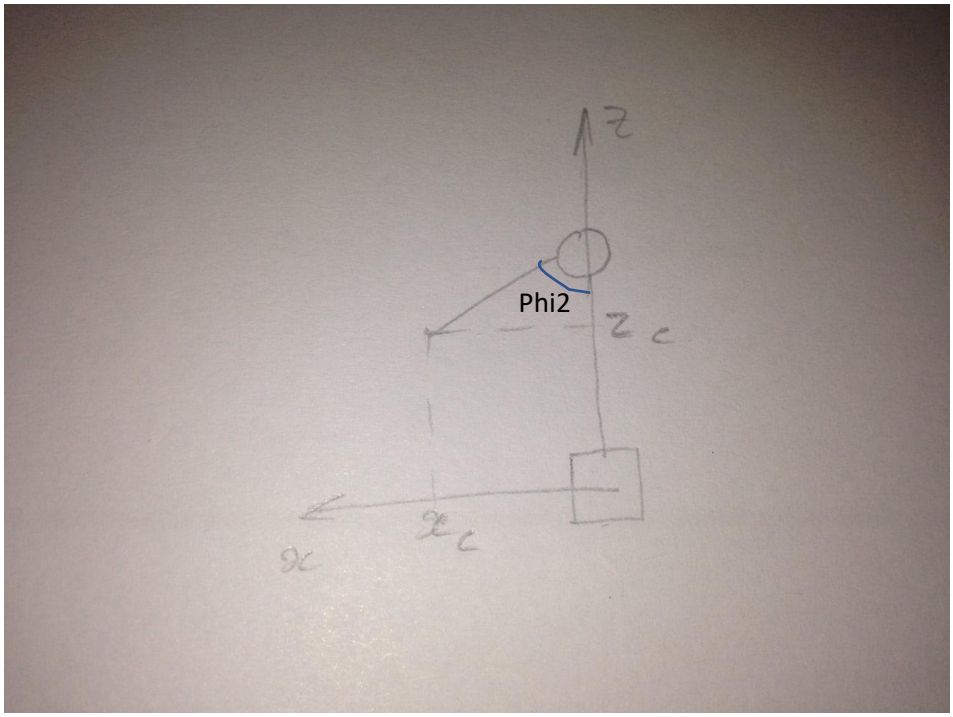
Secondly, find the coordinates of the points, where is intersection of axis from first and last parts. This point is colored red in upper picture. Let it be point C

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} T(1,4) \\ T(1,4) \\ T(1,4) \end{bmatrix} - d_6 * R * \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

where d_6 is the distance from end-effector to C-point.

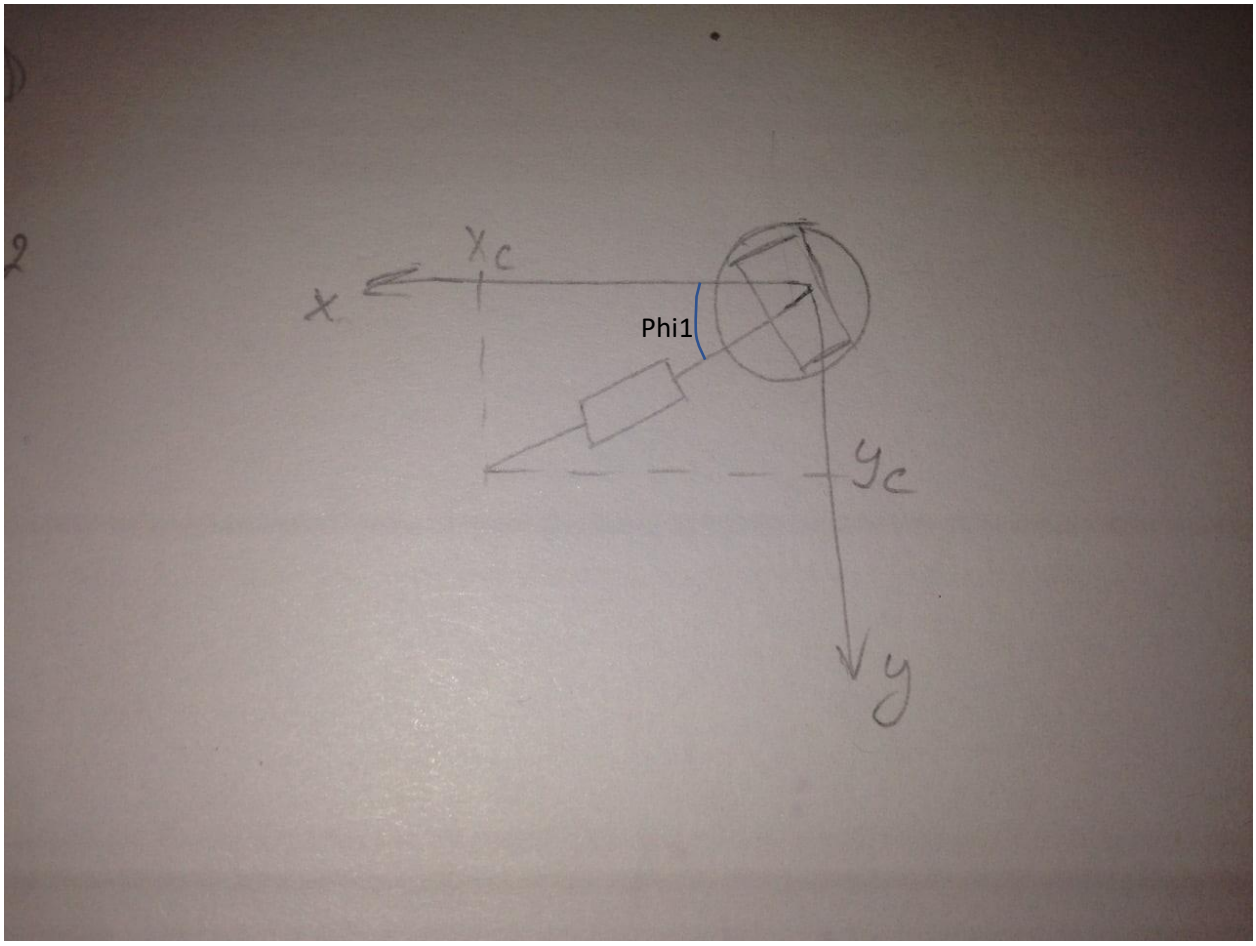
Next, we work with first 3 joints.

Let's watch at the front view



Here it is easy to find phi2 = atan2(L1-z, x)

Now let’s observe view from the top:



Phi1 = atan2(y,x)

Now we need to find R_2^5

$R_2^5 = (R_0^2)^{-1} * R$

R_0^2 can be found the same as in forward task, invertation of it is done in MatLab.

After we found R_2^5 in numerical way is it necessary to find symbolic solution. It founds the same as in forward kinematics.

R_2^5 = [cos(alpha5)*sin(alpha3)*sin(alpha4) - cos(alpha3)*sin(alpha5), - cos(alpha3)*cos(alpha5) - sin(alpha3)*sin(alpha4)*sin(alpha5), - cos(alpha4)*sin(alpha3)]

[- sin(alpha3)*sin(alpha5) - cos(alpha3)*cos(alpha5)*sin(alpha4), cos(alpha3)*sin(alpha4)*sin(alpha5) - cos(alpha5)*sin(alpha3), cos(alpha3)*cos(alpha4)]

[-cos(alpha4)*cos(alpha5), cos(alpha4)*sin(alpha5), -sin(alpha4)]

From here can be found other angels:

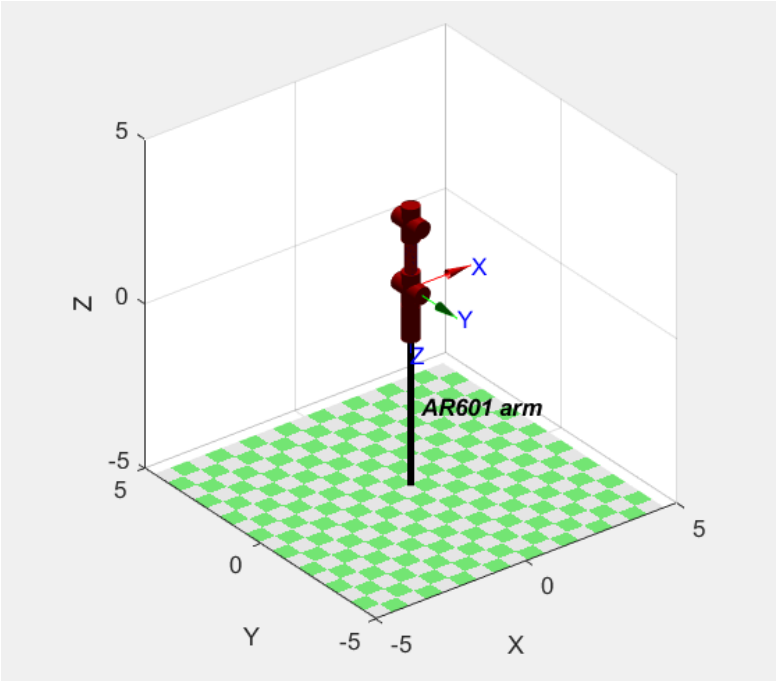
alpha4 = -asin(R25(3,3))

alpha5 = acos(R25(3,2)/sin(alpha4))

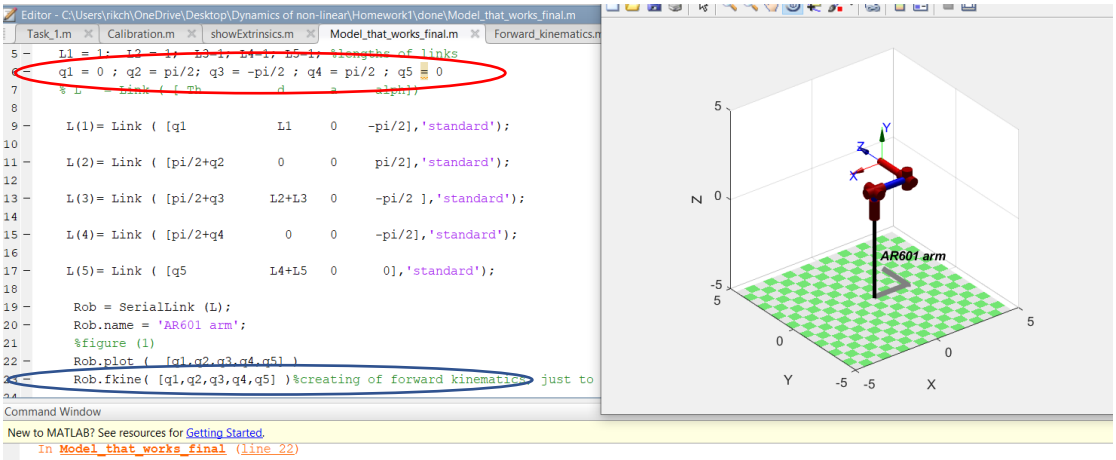
alpha3 = acos(R25(2,3)/cos(alpha4))

Experiments:

Model in Matlab. As I understood, it creates it in the way as it wants, never mind of DH-parameters



For better result we sholud change effort in joints (red circle in the picture). Moreover to check, I introduce built-in function of forward kinematics (in blue circle), here can be shown the results (in green circle



Let’s compare built-in decision and my own:

```
T05 =

    -1     0     0     2
     0     0     1     2
     0     1     0     1
     0     0     0     1

>> builtindecis


builtindecis =
        -1          0          0          2
         0          0          1          2
         0          1          0          1
         0          0          0          1
```

Wow. Not everything is so bad. It is workable! Even if to change angels to comprehensive:

```
5 -    L1 = 1;  L2 = 1;  L3=1; L4=1; L5=1; %lengths of links
6 -    q1 = pi/4 ; q2 = pi/2; q3 = -pi/2+pi/3 ; q4 = pi/2 ; q5 = 0
7      % L    = Link ( [ Th          d       a      alph])
8
9 -     L(1)= Link ( [q1            L1      0     -pi/2],'standard');
10
11 -    L(2)= Link ( [pi/2+q2       0       0      pi/2],'standard');
12
```

Command Window

New to MATLAB? See resources for Getting Started.

```
T05 =

   -0.7071    0.6124   -0.3536    0.7071
   -0.7071   -0.6124    0.3536    2.1213
         0    0.5000    0.8660    2.7321
         0         0         0    1.0000

>> builtindecis


builtindecis =
   -0.7071    0.6124   -0.3536    0.7071
   -0.7071   -0.6124    0.3536     2.121
         0    0.5000    0.8660     2.732
         0         0         0         1
>>
```

Not bad as I think!

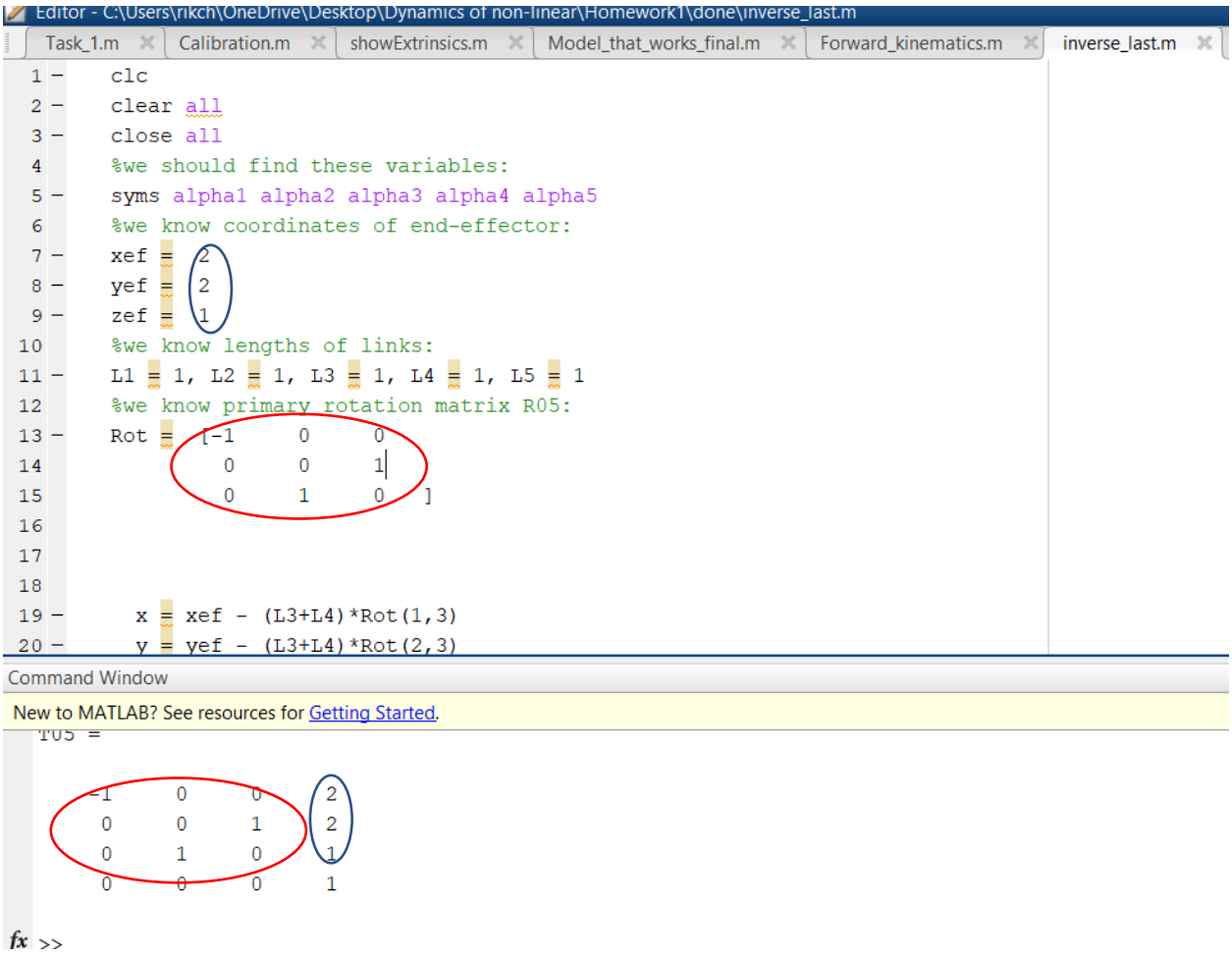
Now inverse kinematics:

Transformation matrix for primary position is

T05 =

-1	0	0	2
0	0	1	2
0	1	0	1
0	0	0	1

So, carefully copy it:



And the result of inverse task:

```
Task_1.m | Calibration.m | showExtrinsics.m | Model_that_works_final.m | Forward_kinematics.m | inverse_last.m

4    %we should find these variables:
5 -  syms alpha1 alpha2 alpha3 alpha4 alpha5
6    %we know coordinates of end-effector:
7 -  xef =  2
8 -  yef =  2
9 -  zef =  1
10   %we know lengths of links:
11 - L1 = 1, L2 = 1, L3 = 1, L4 = 1, L5 = 1
12   %we know primary rotation matrix R05:
13 - Rot =  [-1     0     0
14             0     0     1
15             0     1     0   ]
16
17
18
19 -    x = xef - (L4+L5)*Rot(1,3)
20 -    y = yef - (L4+L5)*Rot(2,3)
21 -    z = zef - (L4+L5)*Rot(3,3)
22   % %from the picture we can find coordinate of 4th joint:
23   % x = xef-sqrt(xef^2+yef^2)*sin(atan(xef/yef))
```

Command Window

New to MATLAB? See resources for Getting Started.

```
     0


alpha =

     0     0     0     0   NaN
```

Yes, that's it. But what is with the last alpha? Actually, we can't know anything about that angle before the wrist will be not just link (hand as an example). For our task this angle plays no role.

Okay, it's time to change configuration. Do you like hardcore? Let's try to do it.

Forward kinematics:

```
% clc
% clear all
% close all
%syms alpha1 L1 alpha2 L2 alpha3 L3 alpha4 L4 alpha5 L5
alpha1 = 0.256, L1 = 1, alpha2 = 0.896, L2 = 1, alpha3 = 0.587, L3 = 1, alpha4 = 0.7845, L4 = 1, alpha5 = 0, L5 = 1

%1 - rotation around 'z' and translation along 'z'
Rj1 = [cos(alpha1) -sin(alpha1) 0
       sin(alpha1) cos(alpha1)  0  %rotation of joints
           0             0      1]

Ra1 = [ 1  0  0
        0  0  1  %rotation of axes
        0 -1  0]

R01 = Rj1*Ra1 %full roatation
T01 = [R01(1,1) R01(1,2) R01(1,3) 0
       R01(2,1) R01(2,2) R01(2,3) 0
       R01(3,1) R01(3,2) R01(3,3) L1
        0        0        0       1]
```

```
T05 =

   -0.5744    0.7692   -0.2801    0.6486
   -0.7584   -0.3712    0.5358    1.3880
    0.3082    0.5202    0.7965    1.0314
         0         0         0    1.0000
```

CAREFULLY copy it to inverse:

```
clc
clear all
close all
%we should find these variables:
syms alpha1 alpha2 alpha3 alpha4 alpha5
%we know coordinates of end-effector:
xef =  0.6486
yef =  1.3880
zef =  1.0314
%we know lengths of links:
L1 = 1, L2 = 1, L3 = 1, L4 = 1, L5 = 1
%we know primary rotation matrix R05:
Rot =  [-1     0   -0.2801
          0     0    0.5358
          0     1    0.7965   ]



    x = xef - (L4+L5)*Rot(1,3)
    y = yef - (L4+L5)*Rot(2,3)
```

```
    0.5735


alpha =

  0.2560 + 0.0000i   0.9121 + 0.0000i   0.5735 + 0.0000i   0.7933 + 0.0000i   3.1416 - 0.4640i
```

It looks like the same. If to say, I don't know, why is it not the same in high precision, but error is about 9%, I think it is enough. Now, it's time to celebrate another one done homework! ^_^

Common decisions and numerical:

To change between symbolic decision and numerical in forward kinematics change commented line:

```
% clc
% clear all
% close all
%syms alpha1 L1 alpha2 L2 alpha3 L3 alpha4 L4 alpha5 L5
alpha1 = 0.256, L1 = 1, alpha2 = 0.896, L2 = 1, alpha3 = 0.587, L3 = 1, alpha4 = 0.7845, L4 = 1, alpha5 = 0, L5 = 1

%1 - rotation around 'z' and translation along 'z'
Rj1 = [cos(alpha1) -sin(alpha1) 0
       sin(alpha1) cos(alpha1)  0  %rotation of joints
           0             0      1]

Ra1 = [ 1   0   0
        0   0   1  %rotation of axes
        0  -1   0]

R01 = Rj1*Ra1 %full roatation
T01 = [R01(1,1) R01(1,2) R01(1,3) 0
       R01(2,1) R01(2,2) R01(2,3) 0
       R01(3,1) R01(3,2) R01(3,3) L1
        0        0        0       1]
```

Now you have numerical solution

```
Editor - C:\Users\rikch\OneDrive\Desktop\Dynamics of non-linear\Homework1\done\Forward_kinematics.m
Task_1.m | Calibration.m | showExtrinsics.m | Model_that_works_final.m | Forward_kinematics.m | inverse_last.m | Inverse_in_syms.m
1   % clc
2   % clear all
3   % close all
4   syms alpha1 L1 alpha2 L2 alpha3 L3 alpha4 L4 alpha5 L5
5   %alpha1 = 0.256, L1 = 1, alpha2 = 0.896, L2 = 1, alpha3 = 0.587, L3 = 1, alpha4 = 0.7845, L4 = 1, alpha5 = 0, L5 = 1
6
```

And now symbolic.

To change between symbolic decision and numerical in inverse kinematics use different files: *inverse* (numerical solution) and *inverse_in_syms* (symbolic solution)

File *model* helps to show the model. To chance angels it q2, q3, q4 type angle you want after symbol “+”. Do not delete “pi/2”

```
clc
clear all
close all

L1 = 1;  L2 = 1;  L3=1; L4=1; L5=1; %lengths of links
q1 = 0 ; q2 = pi/2+0; q3 = -pi/2+0 ; q4 = pi/2+0; q5 = 0
% L    = Link ( [ Th          d       a      alph])

 L(1)= Link ( [q1             L1      0     -pi/2 'standard');
```