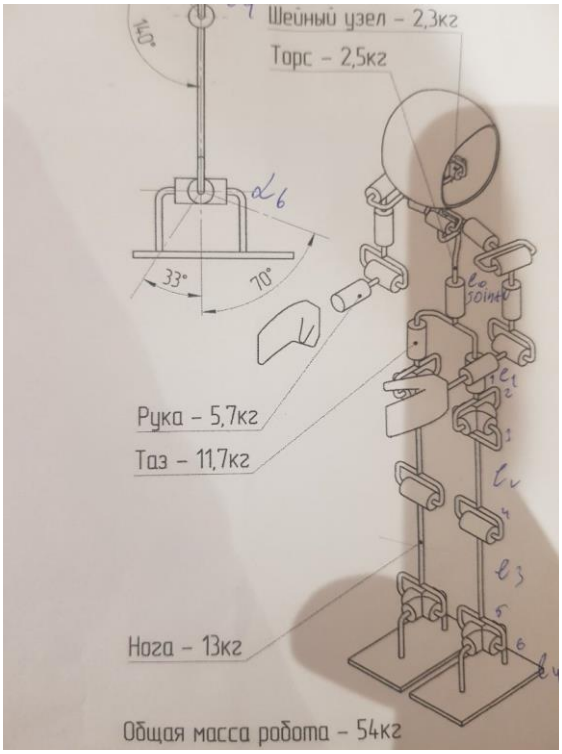
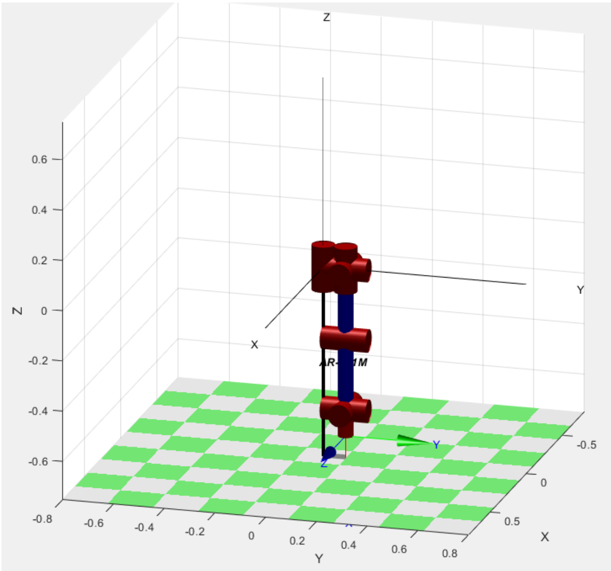
Homework_1

Sevostyanov Nikita

AR601 leg

We have a kinematic scheme of our robot that you can see below(pic.1-2)





Pic.1 AR-601M

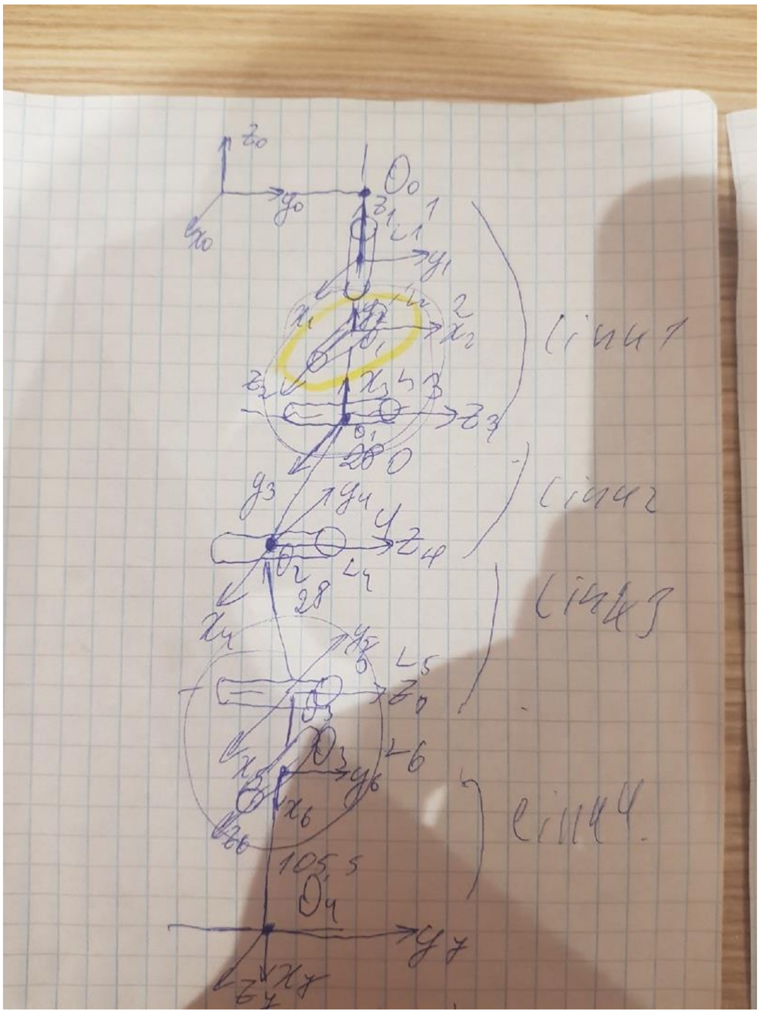
Pic.2 AR-601M in MATLAB

Task that I need to do:

1.Develop kinematic model of the robot

So I used MATLAB for this purpose and robotics Toolbox.

We have 6 joints and 5 links(including link from frame 0 to frame 1) , that you can see in pic.3 below.



Pic.3 Kinematic scheme of the leg

Where base frame is a point at bottom side of torso. Joints 1-3 have the same location of frames and joints 5-6 also have the same location of frames.

I used DH convention for assigning the coordinate frames. And after that we can denote DH parameters for FK.

2. Solve forward kinematics problem

DH- parameters:

q = [fix 0 -pi/2 0 0 0 0];
alpha = [pi -pi/2 -pi/2 0 0 pi/2 0];
d = [0 0 0 0 0 0 0];
a = [0.088 0 0 0.280 0.280 0 0.1055];
Where:
q(theta) - the angle about the previous z to align its x to the new x
d - the depth along the previous joint's z axis to the common normal
a - is the distance along the rotated x axis (radius of rotation about previous z axis)
alpha - angle of rotation about the new x axis to put previous z in its desired orientation.
After that we use formula from DH-convention, that you can see below:

$$
\begin{aligned}
A_i &= Rot_{z,\theta_i}\text{Trans}_{z,d_i}\text{Trans}_{x,a_i}Rot_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

After that I wrote a function in MATLAB, that you can see in file FK.m, which after that is used in AR_601M.m file for calculating forward kinematics. The result of this function is homogeneous matrix.

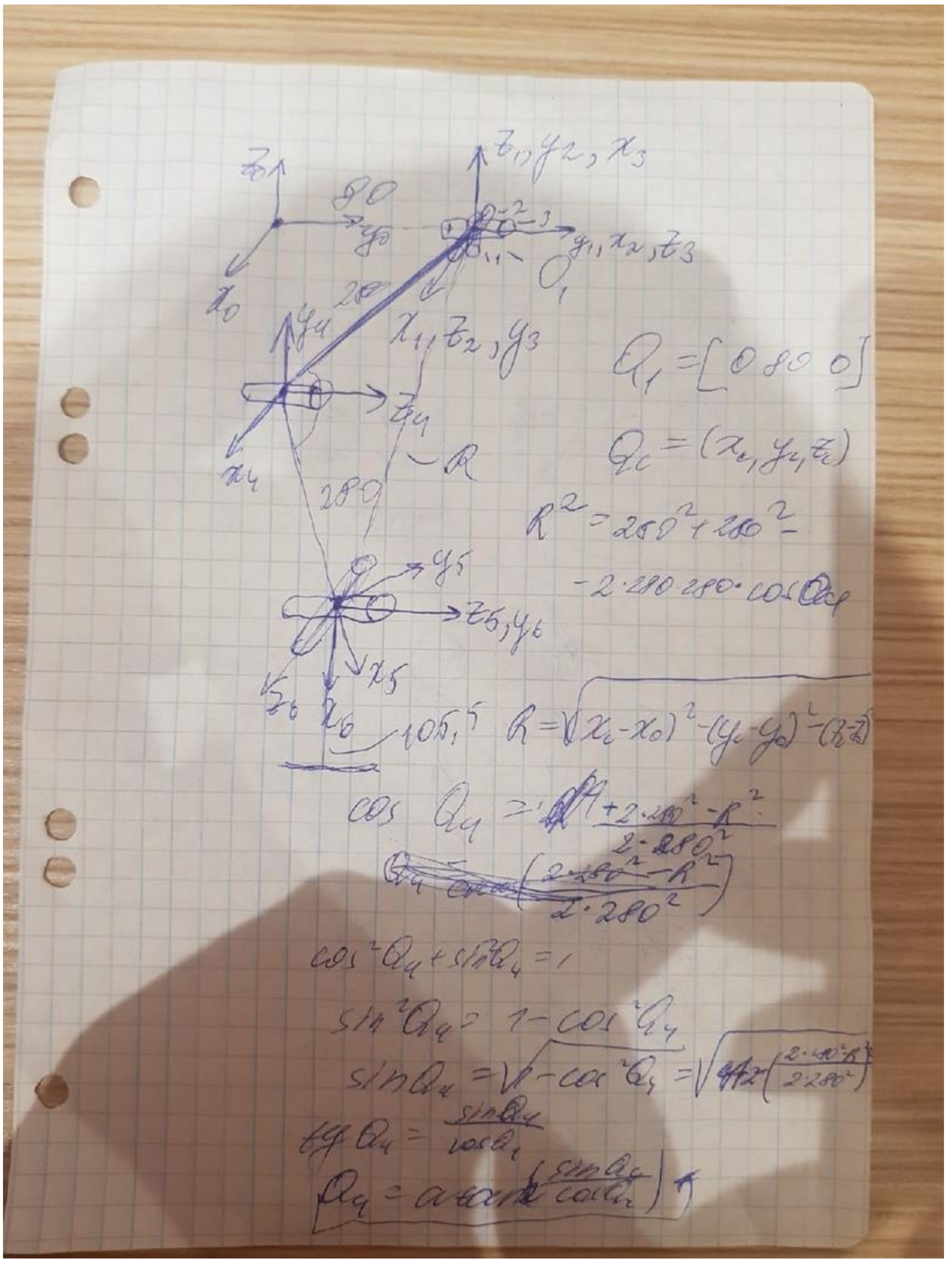
```
%Method of solfing FK
q1 = q0*(180/pi); % converting theta angles from radian to degrees to calculate FK via
%DH convention
alpha1 = alpha*(180/pi); %converting alpha angles
DH1 = [q1(1)  d(1) a(1)   alpha1(1)];
DH2 = [q1(2) d(2) a(2)   alpha1(2)];
DH3 = [q1(3) d(3) a(3)   alpha1(3)];
DH4 = [q1(4) d(4) a(4)   alpha1(4)];
DH5 = [q1(5) d(5) a(5)   alpha1(5)];
DH6 = [q1(6) d(6) a(6)   alpha1(6)];
DH7 = [q1(7) d(7) a(7)   alpha1(7)];
% Forward Kinematics function that you can see in FK.m
T = FK(q1, DH1, DH2, DH3, DH4, DH5, DH6,DH7);
%Exam the Forward Kinematics solution
Direct = R.fkine(q0);
%Received results are equal, so it's a right function
```

Function for FK in FK.m:

```
function T = FK(q, DH1, DH2, DH3, DH4, DH5, DH6, DH7) %Function for implementing FK
%using definition of DH convention, as 2 rotations and 2 translations,
%calculatin T0_7 matrix.
    T = trotz(q(1))*transl(0,0,DH1(2))* transl(DH1(3),0,0)*trotx(DH1(4))* ...
        trotz(q(2))*transl(0,0,DH2(2))* transl(DH2(3),0,0)*trotx(DH2(4))* ...
        trotz(q(3))*transl(0,0,DH3(2))* transl(DH3(3),0,0)*trotx(DH3(4))* ...
        trotz(q(4))*transl(0,0,DH4(2))* transl(DH4(3),0,0)*trotx(DH4(4))* ...
        trotz(q(5))*transl(0,0,DH5(2))* transl(DH5(3),0,0)*trotx(DH5(4))* ...
        trotz(q(6))*transl(0,0,DH6(2))* transl(DH6(3),0,0)*trotx(DH6(4))* ...
        trotz(q(7))*transl(0,0,DH7(2))* transl(DH7(3),0,0)*trotx(DH7(4));
```

3. Solve inverse kinematics problem

I used a geometric approach for finding theta_4. The calculation of this angle you can see below.



I used a formula of cosine rule and formula for distance between two points.

Where (x_c, y_c, z_c) is the desired position of end-effector.

After that I calculate this parameter(theta4) via MATLAB:

```
%Calculating theta4 via geometric approach
dot_c = [0 0 0.560];
dot_0 = [0 0 0];
R = real(sqrt((dot_c(1)-dot_0(1))^2 +(dot_c(2)-dot_0(2))^2 +(dot_c(3)-dot_0(3))^2));
X = (2*0.280^2-R^2)/(2*0.280^2);
Y = real(sqrt(1-cos(X)^2)); %take into consideration only real numbers,
%excluding outliers from workspace
theta_41 =real(acos(X));
```

4. Some examples and test

1) Checking FK via MATLAB , comparing results of function fkine and my method