ROBOTİK PROJE

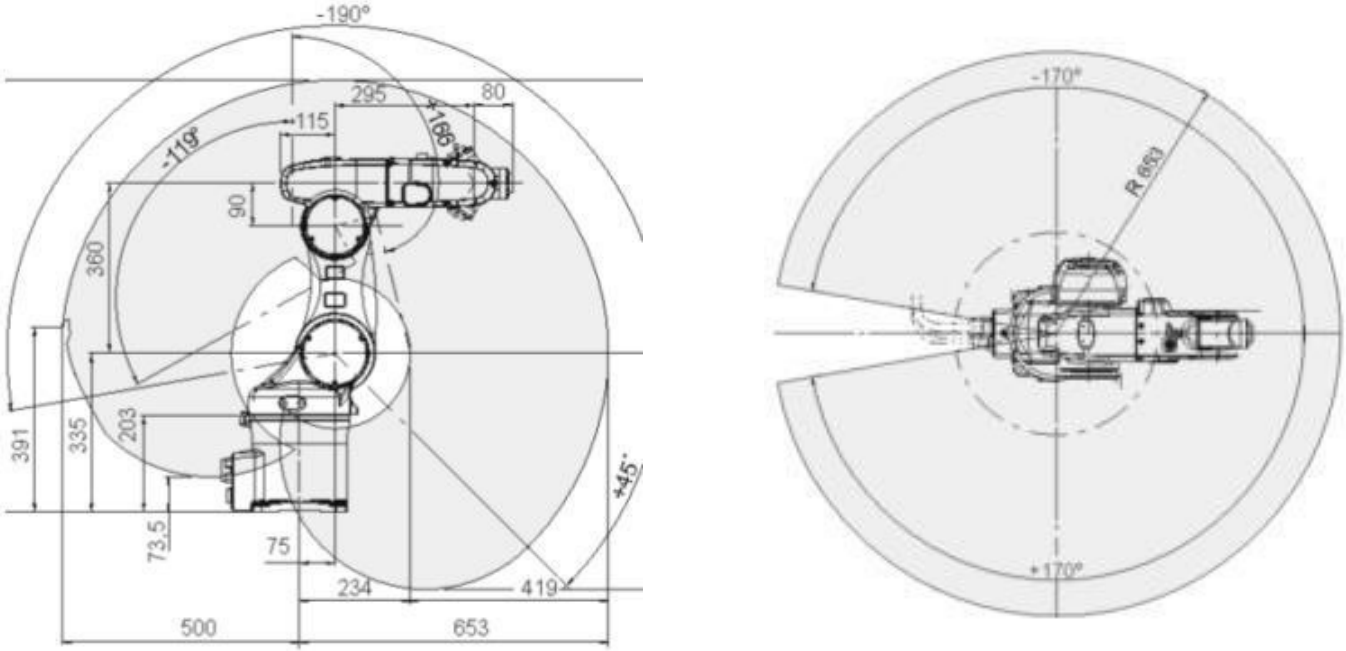
KUKA KR 5 SİXX R650

Yiğit OĞLAKCI-222811010

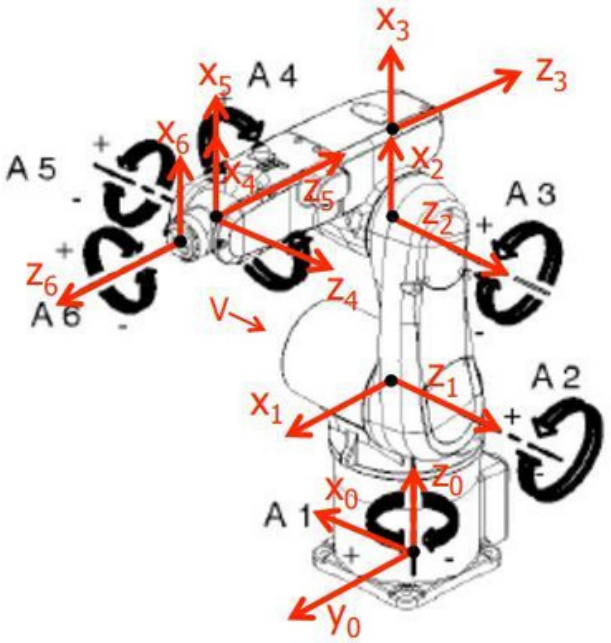
KUKA KR5 SIXX R650 MODEL ROBOTUN İLERİ KİNNEMATİK DENKLEMİNİN ÇIKARILMASI

1.Adım: Eksen Yerleşimlerinin Yapılması:

Robotun teknik dokümanından (şekil 1) yararlanarak basit bir eksen yerleşimi gösterilmiştir (şekil 2).



Şekil 1



Şekil 2

2.Adım: Denavit-Hartenberg (DH) Tablosunun Çıkarılması :

i	αi-1	ai-1	di	θi
1	0	0	335	0
2	75	-90	0	0
3	270	0	0	-90
4	90	-90	295	0
5	0	90	0	0
6	0	-90	80	180

3.Adım: Transformasyon Matrislerinin Bulunması

Her bir ekleme ait transformasyon matrisi için genel formülden yararlanılır.Matlab üzerinde döngü yardımıyla her i değeri için matrisler çözülür

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

```
% DH Matrisleri
for i=1:length(a)
    A(:,:,i) = [cosd(t(i)) -sind(t(i)) 0 a(i);
        sind(t(i))*cosd(alf(i)) cosd(t(i))*cosd(alf(i)) -sind(alf(i)) -sind(alf(i))*d(i);
        sind(t(i))*sind(alf(i)) cosd(t(i))*sind(alf(i)) cosd(alf(i)) cosd(alf(i))*d(i);
        0 0 0 1];
end
```

4.Adım:Transformasyon Matrislerin Çarpımı

Bulmuş olduğumuz bu transformasyon matrislerini ileri kinematik denklem elde etmek için çarpıyoruz.

$$ {}^{0}_{6}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T $$

Matris çarpımını elde edebilmek için Matlab programından yararlanıyoruz.
Fakat matrislerin çarpım sonucu çok uzun ve karmaşık olduğundan burada gösterilememektedir. Elde ettiğimiz çarpım sonucundaki 4x4 matrisin 1.satır 4.sütunundaki değer X, 2.satır 4.sütunundaki değer Y, 3.satır 4.sütunundaki değer ise Z konumunu veren eşitlikleri vermektedir.

```
T = eye(4);
for i=1:length(a)
    T = T * A(:,:,i);
end
```

5.Adım: Test Kodu ve Karşılaştırma

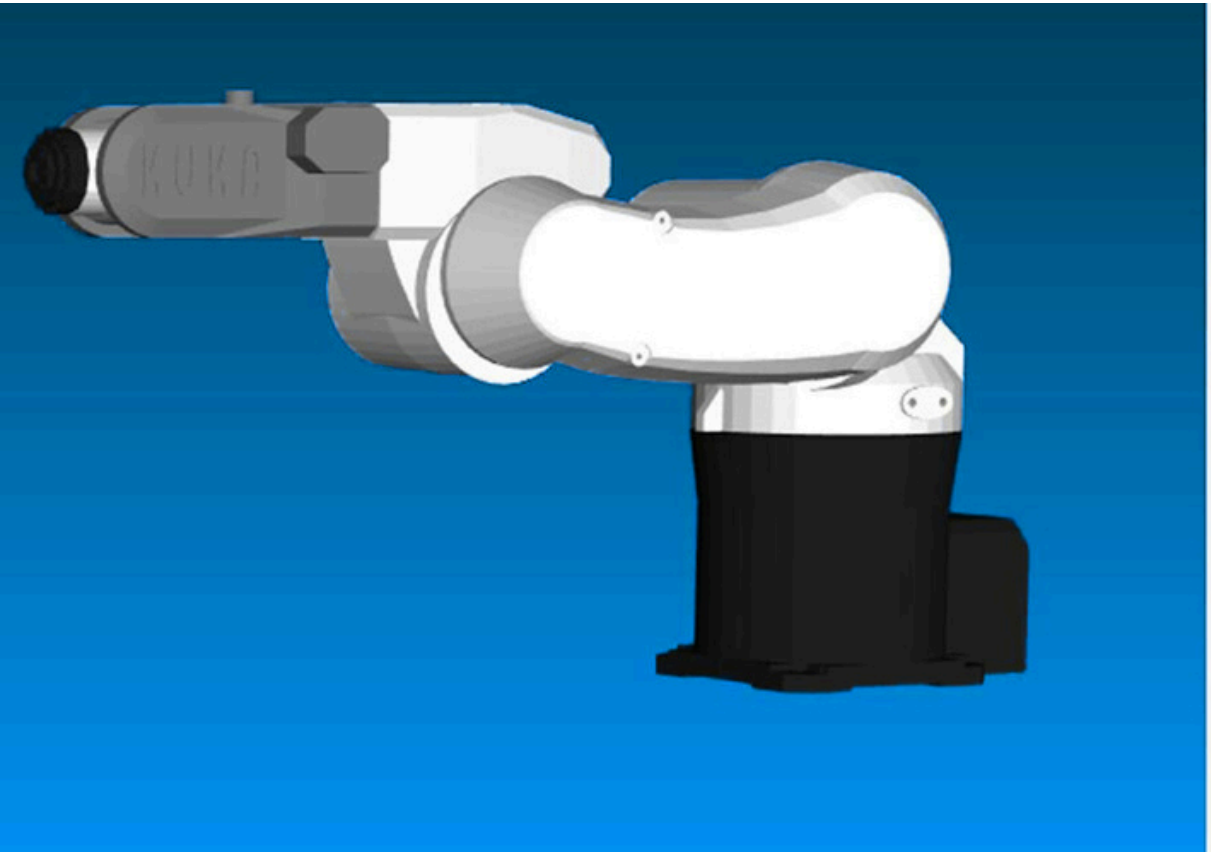
Çarpım işlemleri ve konumu göstermek için Matlab'da daha efektif bir kod yazıyoruz. Birkaç açı değeri test ederek bulduğumuz konum değerlerini "RoKiSim" programındaki konum değerleri ile karşılaştırıp aynı sonucu elde ettiğimizi görüyoruz.

θ1=0, θ2=0, θ3=0, θ4=0, θ5=0, θ6=0 için test:

```
% DH Parametreleri
a = [0 75 270 90 0 0]; % Eklemlerin uzunlukları
alf = [0 -90 0 -90 90 -90]; % Eklemlerin dönüş açıları (derece cinsinden)
d = [335 0 0 295 0 80]; % Eklemlerin offsetleri
t = [0 0 -90 0 0 180]; % Eklemlerin açıları (derece cinsinden)

% DH Matrisleri
for i=1:length(a)
    A(:,:,i) = [cosd(t(i)) -sind(t(i)) 0 a(i);
        sind(t(i))*cosd(alf(i)) cosd(t(i))*cosd(alf(i)) -sind(alf(i)) -sind(alf(i))*d(i);
        sind(t(i))*sind(alf(i)) cosd(t(i))*sind(alf(i)) cosd(alf(i)) cosd(alf(i))*d(i);
        0 0 0 1];
end

% İleri kinematik hesaplaması
T = eye(4);
for i=1:length(a)
    T = T * A(:,:,i);
end

% Uç noktanın konumunu göster (derece cinsinden)
x = T(1,4);
y = T(2,4);
z = T(3,4);
disp(['Uç noktanın konumu: (' num2str(x) ', ' num2str(y) ', ' num2str(z) ')']);
```

Command Window

```
Uç noktanın konumu: (720, 0, 425)
>>
```



Joint Jog

Init.

θ_1:	-170°	170°	0.00	°
θ_2:	-190°	45°	0.00	°
θ_3:	-119°	165°	0.00	°
θ_4:	-190°	190°	0.00	°
θ_5:	-120°	120°	0.00	°
θ_6:	-358°	358°	0.00	°

Cartesian Jog

Position (tool frame w.r.t. world frame)

733.000 mm, 0.000 mm, 412.000 mm

Orientation (tool frame w.r.t. world frame)

Euler angles: 0.000°, 90.000°, 0.000°

Quaternions: 0.70711, 0.00000, 0.70711, 0.00000

Tool frame

	X	Y	Z
Translation along	●	○	○
Rotation about	○	○	○

θ1=0, θ2=-90, θ3=90, θ4=0, θ5=0, θ6=0 için test:

```
% DH Parametreleri
a = [0 75 270 90 0 0]; % Eklemlerin uzunlukları
alf = [0 -90 0 -90 90 -90]; % Eklemlerin dönüş açıları (derece cinsinden)
d = [335 0 0 295 0 80]; % Eklemlerin offsetleri
t = [0 -90 90-90 0 0 180]; % Eklemlerin açıları (derece cinsinden)

% DH Matrisleri
for i=1:length(a)
    A(:,:,i) = [cosd(t(i)) -sind(t(i)) 0 a(i);
        sind(t(i))*cosd(alf(i)) cosd(t(i))*cosd(alf(i)) -sind(alf(i)) -sind(alf(i))*d(i);
        sind(t(i))*sind(alf(i)) cosd(t(i))*sind(alf(i)) cosd(alf(i)) cosd(alf(i))*d(i);
        0 0 0 1];
end

% İleri kinematik hesaplaması
T = eye(4);
for i=1:length(a)
    T = T * A(:,:,i);
end

% Uç noktanın konumunu göster (derece cinsinden)
x = T(1,4);
y = T(2,4);
z = T(3,4);
disp(['Uç noktanın konumu: (' num2str(x) ', ' num2str(y) ', ' num2str(z) ')']);
```

Command Window

```
Uç noktanın konumu: (450, 0, 695)
>>
```

Joint Jog

Init.

θ_1:	-170°	—	170°	0.00	°
θ_2:	-190°	—	45°	-90.00	°
θ_3:	-119°	—	165°	90.00	°
θ_4:	-190°	—	190°	0.00	°
θ_5:	-120°	—	120°	0.00	°
θ_6:	-358°	—	358°	0.00	°

Cartesian Jog

Position (tool frame w.r.t. world frame)

450.000 mm, 0.000 mm, 695.000 mm

Orientation (tool frame w.r.t. world frame)

Euler angles: 0.000°, 90.000°, 0.000°

Quaternions: 0.70711, 0.00000, 0.70711, 0.00000

Tool frame

	X	Y	Z
Translation along	●	○	○
Rotation about	○	○	○

Ters Kinematiğin Oluşturulması

Uç efektörün X,Y,Z konumlarını girip eklemlerin açısını bulmaya yarar.

```
function thetas = inverse_kinematics(goalPos, initialGuess)
    % Tolerans
    epsilon = 1e-3;
    maxIter = 1000;
    alpha = 0.5;  % öğrenme oranı

    thetas = initialGuess;

    for i = 1:maxIter
        % Mevcut uç pozisyonunu al
        positions = forward_kinematics_positions(thetas);
        currentPos = positions(:, end);

        % Hata vektörü
        error = goalPos - currentPos;

        if norm(error) < epsilon
            return;
        end

        % Sayısal türevle Jacobian tahmini
        J = numerical_jacobian(@forward_kinematics_positions, thetas);

        % Güncelleme (dampingli çözüm)
        deltaTheta = alpha * pinv(J) * error;

        % Açılar güncelleniyor
        thetas = thetas + deltaTheta';
    end

    warning('Geri kinematik çözüm bulunamadı, yakınsama sağlanamadı.');
end
```

Jakobiyen Matrisi

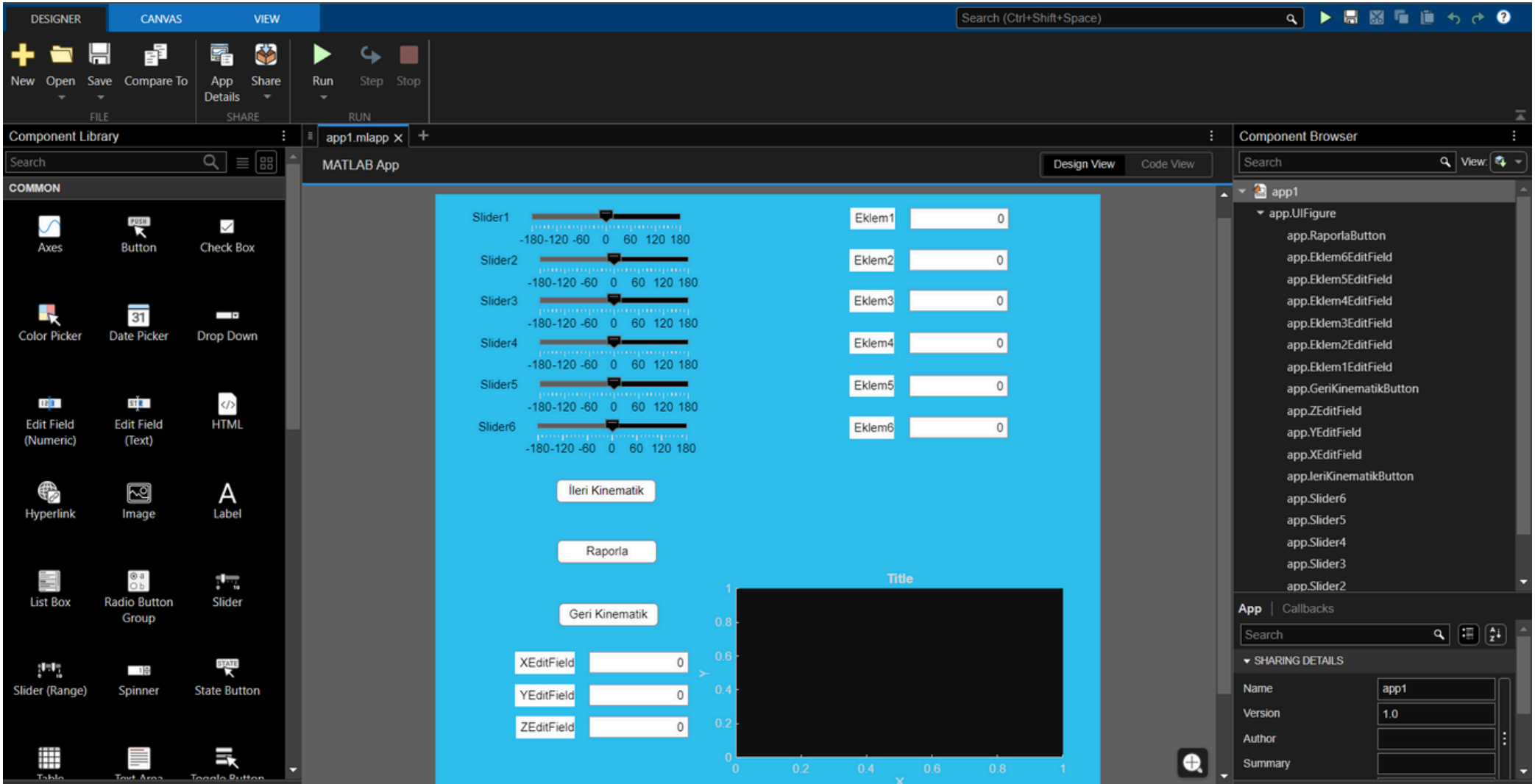
Analitik ters kinematik formülleri çoğu endüstriyel robota uygulanamaz bu yüzden jakobiyen matrisine ihtiyacımız var.

```
function J = numerical_jacobian(f, thetas)
    n = length(thetas);
    f0 = f(thetas);
    p0 = f0(:, end);  % uç efektör konumu
    J = zeros(3, n);
    delta = 1e-5;

    for i = 1:n
        thetas_ = thetas;
        thetas_(i) = thetas_(i) + delta;
        fi = f(thetas_);
        pi = fi(:, end);
        J(:, i) = (pi - p0) / delta;
    end
end
```

UYGULAMANIN GELİŞTİRİLMESİ

1.Adım: Tasarımın Yapılması

Matlab – App Designer'de"Design View" penceresinde, bize uygun tasarımı "COMPONENT LİBRARY" kısmından komponent ekleyerek yapıyoruz.
Kullandığımız komponentler: Slider, Image, Edit Field (numeric), State Button, Axes ve Lable.



2.Adım: Kod Yazımı ve Test

“Code View” kısmına kodlarımızı yazıyoruz:
İlk olarak değişkenlerimizi tanımlıyoruz

```
properties (Access = private
    t1=0;
    t2=0;
    t3=0;
    t4=0;
    t5=0;
    t6=0;
    X=0;
    Y=0;
    Z=0;
    T=0;

end
```

Daha sonra “transformasyon” adında bir fonksiyon tanımlayıp bulduğumuz ileri kinematik denklemleri bu fonksiyona yazıyoruz.
Not: Eklenen resime tüm denklemler sığmamaktadır.

```
methods (Access = private)

    function transformasyon(app)
        app.X=75*cos((pi*app.t1)/180) - 80*cos((pi*app.t5)/180)*(c
        app.Y=75*sin((pi*app.t1)/180) - 80*sin((pi*app.t5)/180)*(c
        app.Z=295*sin((pi*app.t2)/180)*sin((pi*(app.t3 - 90))/180)
    end
end
```

Ardından her Slider komponenti için yandaki kodları yazıyoruz:

Bu kod sayesinde Slider komponentinde girdiğimiz açı değerleri transformasyon fonksiyonunda hesaplanarak X, Y ve Z konumlarını gerçek zamanlı biçimde XEditField ekranında göstermektedir.

```
function Q1SliderValueChanged(app, event)
    app.EditFieldQ1.Value=app.Q1Slider.Value; % EditField ekranını Slider değeri ile eşitle

    app.t1=app.Q1Slider.Value; %Q1Slider değerini t1 açı değerine aktar
    app.t2=app.Q2Slider.Value; %Q2Slider değerini t2 açı değerine aktar
    app.t3=app.Q3Slider.Value; %Q3Slider değerini t3 açı değerine aktar
    app.t4=app.Q4Slider.Value; %Q4Slider değerini t4 açı değerine aktar
    app.t5=app.Q5Slider.Value; %Q5Slider değerini t5 açı değerine aktar
    app.t6=app.Q6Slider.Value; %Q6Slider değerini t6 açı değerine aktar

    app.transformasyon; %transformasyon fonksiyonunu çağır

    app.Xkonum.Value=app.X; % X konum bilgisini UCX ekranına yaz
    app.Ykonum.Value=-1*app.Y; % Y konum bilgisini UCY ekranına yaz
    app.Zkonum.Value=app.Z; % Z konum bilgisini UCZ ekranına yaz
```

Axes komponentinde robotun konumunu kinematik zincir (iskeleti) olarak göstermek için ise kodun devamına şunları ekliyoruz

```
function draw_end_effector_axes(app, T)
    origin = T(1:3, 4);       % Pozisyon
    R = T(1:3, 1:3);          % Yönelim (rotation matrix)
    L = 200;                  % Ok uzunluğu

    % X ekseni (k)
    quiver3(app.UIAxes, origin(1), origin(2), origin(3), ...
        R(1,1)*L, R(2,1)*L, R(3,1)*L, 'Color', 'r', 'LineWidth', 2, 'MaxHeadSize', 2);

    % Y ekseni (y)
    quiver3(app.UIAxes, origin(1), origin(2), origin(3), ...
        R(1,2)*L, R(2,2)*L, R(3,2)*L, 'Color', 'g', 'LineWidth', 2, 'MaxHeadSize', 2);

    % Z ekseni (m)
    quiver3(app.UIAxes, origin(1), origin(2), origin(3), ...
        R(1,3)*L, R(2,3)*L, R(3,3)*L, 'Color', 'b', 'LineWidth', 2, 'MaxHeadSize', 2);
end
```

Son olarak projemizin testlerini gerçekleştirip projemizi bitiriyoruz