

Kocaeli Üniversitesi  
Mühendislik Fakültesi Mekatronik Mühendisliği Bölümü  
Robotik ve Otomasyon Uygulamaları  
Laboratuvar Proje Raporu  
Enes GÜNGÖR, 170223046

Arayüzün yerleşimini belirleyerek başladım. İlk sekmede 8 satırlı değişken eklem tipleri ile çalıştırabildiğimiz bir dh tablosu hazırladım. Hesapla ve çizdir butonları ve grafiği ekledim. Dh tablosu oluşturulduğunda sliderlar aktif olacak artık ve slider ile oynandığında yada çizdir butonuna basıldığında robotun üç boyutlu çizgisel grafiği yenileniyor.

Hesapla butonuna bastığımızda birinci satırdan başlayıp sekizinci satıra doğru eklem tipi seçilmiş mi yada seçildiyse hangi türde olduğuna göre çalışan bir script geliştirdim.

```
% Hesapla butonu içerisinde ilk satırdaki eklem tipinin revolute seçildiği
durum için çalışan kod parçası
switch app.lineType1.Value
    case 1
        a = app.a1.Value;
        alfa = app.alp1.Value;
        d = app.d1.Value;
        teta = app.q1.Value;
        syms sym1;
        teta = teta + sym1;

        matris1 = [
            cosd(teta) -sind(teta)*cosd(alfa)
            sind(teta)*sind(alfa) a*cosd(teta);

            sind(teta) cosd(teta)*cosd(alfa) -
            cosd(teta)*sind(alfa) a*sind(teta);

            0 sind(alfa) cosd(alfa) d;

            0 0 0 1
        ];

        matris1 = vpa(matris1);

        guiDosyaYolu = fileparts(mfilename('fullpath'));
        dosyaAdi = fullfile(guiDosyaYolu, 'R-T01.txt');
        fid = fopen(dosyaAdi, 'w');
        fprintf(fid, '%s\n', char(matris1));
        fclose(fid);

        result = matris1;
```

Bu kod parçasında listbox parametreleri ayarlandığı için birinci satırdaki eklem tipi dönerse 1 dönüyor, prizmatikse 2, sabitse 3. Alıntıladığım kod parçasında sadece döner olduğunda çalışan kod mevcuttur. Öncelikle birinci satırdaki dört parametre okunduktan sonra döner eklem tipinde olduğumuz için teta parametresinde sembolik bir teta daha ekleniyor böylece tek bir sembolik değere bağlı transformasyon matrisi elde edilmiş oluyor. Burada matris Variable Precision Arithmetic (VPA) fonksiyonundan geçirilerek yuvarlama vb. durumlardan arındırılıp matris olabildiğince duyarlı hale getiriliyor. Sonrasında arayüzümün çalıştığı dosya yolunu öğreniyorum ve o dosya uzantısına “.txt” dosyamı oluşturup elimdeki matrisi char olarak bu dosyaya kaydedip kapatıyorum.

```
% lineOn fonksiyonundan alıntı
function lineOn(app,i)

    switch i
    case 1
        app.a1.Enable="on";
        app.a1.Editable="on";
        app.alp1.Enable="on";
        app.alp1.Editable="on";
        app.d1.Enable="on";
        app.d1.Editable="on";
        app.q1.Enable="on";
        app.q1.Editable="on";
        app.line1Slider.Enable="on";
        app.line1Degree.Enable="on";
        app.line1Degree.Editable="on";
        app.cizdir.Enable="on";

% lineOff fonksiyonundan alıntı
function lineOff(app,i)

    switch i
    case 1
        app.a1.Enable="off";
        app.a1.Editable="off";
        app.alp1.Enable="off";
        app.alp1.Editable="off";
        app.d1.Enable="off";
        app.d1.Editable="off";
        app.q1.Enable="off";
        app.q1.Editable="off";
        app.line1Slider.Enable="off";
        app.line1Slider.Value=0;
        app.line1Degree.Enable="off";
        app.line1Degree.Editable="off";
        app.line1Degree.Value=0;
        app.lineType1.Value = 0;
        app.cizdir.Enable="off";
```

Burada sliderların ve diğer arayüz elemanlarının kontrollü olarak açılıp kapatılabilmesi için lineOn ve lineOff fonksiyonlarını yazdım. Ayrıca (x1,y1,z1) noktasından başlayan ve (x2,y2,z2) noktasında biten bir çizgi çizen lineDraw fonksiyonu hazırladım. Böylece transformasyon matrisinin 4. Sütunundan Px-Py-Pz 'yi kullanarak çizgisel grafikte bir bağı çizebiliyorum.

```
%lineDraw fonksiyonunun tamamı
function lineDraw(app, x1, y1, z1, x2, y2, z2,type)
    hold(app.UIAxes, 'on');
    plot3(app.UIAxes, [x1, x2], [y1, y2], [z1, z2], 'LineWidth', 6);
    switch type
        case 'R'
            scatter3(app.UIAxes, x1, y1, z1, 'Marker', 'o', 'SizeData',
100, 'MarkerFaceColor', 'flat');
        case 'P'
            scatter3(app.UIAxes, x1, y1, z1, 'Marker', 's', 'SizeData',
100, 'MarkerFaceColor', 'flat');
        case 'S'
            scatter3(app.UIAxes, x1, y1, z1, 'Marker', '^', 'SizeData',
100, 'MarkerFaceColor', 'flat');
    end
    view(app.UIAxes,45,45);
    hold(app.UIAxes, 'off');
end
```

Grafiğin belirlediğim anlarda otomatik güncellenmesini sağlamak için updateGraph fonksiyonu geliştirdim. Bu fonksiyon ilk başta grafiği komple temizledikten sonra hesaplama butonuna bastığımızda hesaplanan “.txt” dosyalarını okuyup dosyanın ismine göre eklem tipini anlıyor ve örneğin “R-T01.txt” dosyası için aşağıdaki kod parçasını çalıştırıyor.

```
% updateGraph Fonksiyonundan bir alıntı
dosyaAdi = 'R-T01.txt';
if exist(dosyaAdi, 'file') == 2
    dosya = fopen(dosyaAdi, 'r');
    sembolikIfade = fgetl(dosya);
    symbolicFunction1 = str2func(['@(sym1) ' sembolikIfade]);
    sym1Val = app.line1Degree.Value;
    matris = symbolicFunction1(sym1Val);
    fclose(dosya);

    app.xValue.Text = num2str(matris(1,4));
```

```

app.yValue.Text = num2str(matris(2,4));
app.zValue.Text = num2str(matris(3,4));

R = matris(1:3, 1:3);
eulerAngles = rotm2eul(R, 'XYZ');
app.rollValue.Text = num2str(eulerAngles(1,1));
app.pitchValue.Text = num2str(eulerAngles(1,2));
app.yawValue.Text = num2str(eulerAngles(1,3));

lineDraw(app,0,0,0,matris(1,4),matris(2,4),matris(3,4), 'R');

end

```

Öncelikle txt dosyasının içindeki sembolik değer içeren matris string olarak direk alındıktan sonra, içindeki sembolik değere göre bu matrisi bir fonksiyon haline getirmek için symbolicFunction1 fonksiyonu yazılıyor. Sembolik değer yerine yazmamız gereken değer ilgili editBox dan alındıktan sonra symbolicFunction1 kullanılarak 4x4'lük bir transformasyon matrisi elde etmiş oluyorum. Bu matristen Px-Py-Pz noktalarını ve 3x3 rotasyon matrisiyle hesapladığım roll-pitch-yaw değerlerini sayfadaki ilgili kısma yazıyorum.

Aslında ikinci sekmedeki updateSim updateGraph fonksiyonu ile, simLineDraw lineDraw fonksiyonu ile aynı amaca hizmet ediyor. Sadece hedef grafik alanı farklı ve birkaç özelleştirme mevcut.

```

% updateSim fonksiyonundan alıntı
function updateSim(app)
    cla(app.sim_UIAxes);

    %line01
    a = app.sim_a1.Value;
    alfa = app.sim_alp1.Value;
    d = app.sim_d1.Value;
    teta = app.sim_q1.Value + app.sim_line1Degree.Value;

    result = [
                                cosd(teta) -sind(teta)*cosd(alfa)
sind(teta)*sind(alfa) a*cosd(teta);

                                sind(teta) cosd(teta)*cosd(alfa) -
cosd(teta)*sind(alfa) a*sind(teta);

                                0 sind(alfa) cosd(alfa) d;

                                0 0 0 1
    ];

    simLineDraw(app,0,0,0,result(1,4),result(2,4),result(3,4));

```

Burada zaten her eklemdeki başlangıç açısı sıfır derece ve satır sayısında sabit olduğundan sembolik değer oluşturmaya gerek duymayıp direk arayüzdeki ilgili kısımlardan parametreleri alarak transformasyon matrisi çözümlerini elde ettim.

```
% Grafiğin eksen limitlerini ayarlamak için kullandığım kod
axis(app.sim_UIAxes, 'auto');

xLim = xlim(app.sim_UIAxes);
yLim = ylim(app.sim_UIAxes);
zLim = zlim(app.sim_UIAxes);

maxLim = max([xLim(2), yLim(2), zLim(2)]);
minLim = min([xLim(1), yLim(1), zLim(1)]);

xlim(app.sim_UIAxes, [minLim,maxLim]);
ylim(app.sim_UIAxes, [minLim,maxLim]);
zlim(app.sim_UIAxes, [minLim,maxLim]);
```

updateSim fonksiyonunun sonunda grafiğin eksen limitlerini karşılaştırıp en geniş aralığa göre hepsini eşitleyen bir kod yazdım. Böylece görsel yanılgılar azaldı.

```
% Simulasyon sayfasındaki çalıştır butonu fonksiyonundan alıntı
open_system('kawasakiRobot.slx');
sim('kawasakiRobot.slx');
close_system('kawasakiRobot.slx', 0);
```

Çalıştır butonuna bastığımızda simulink/simscape modelinin açılması ve başlatılması için bu 3 satırı ekledim.

```
% Birinci satırdaki açı değiştiğinde çalışan fonksiyondan alıntı
blockname = 'kawasakiRobot/j1Degree';

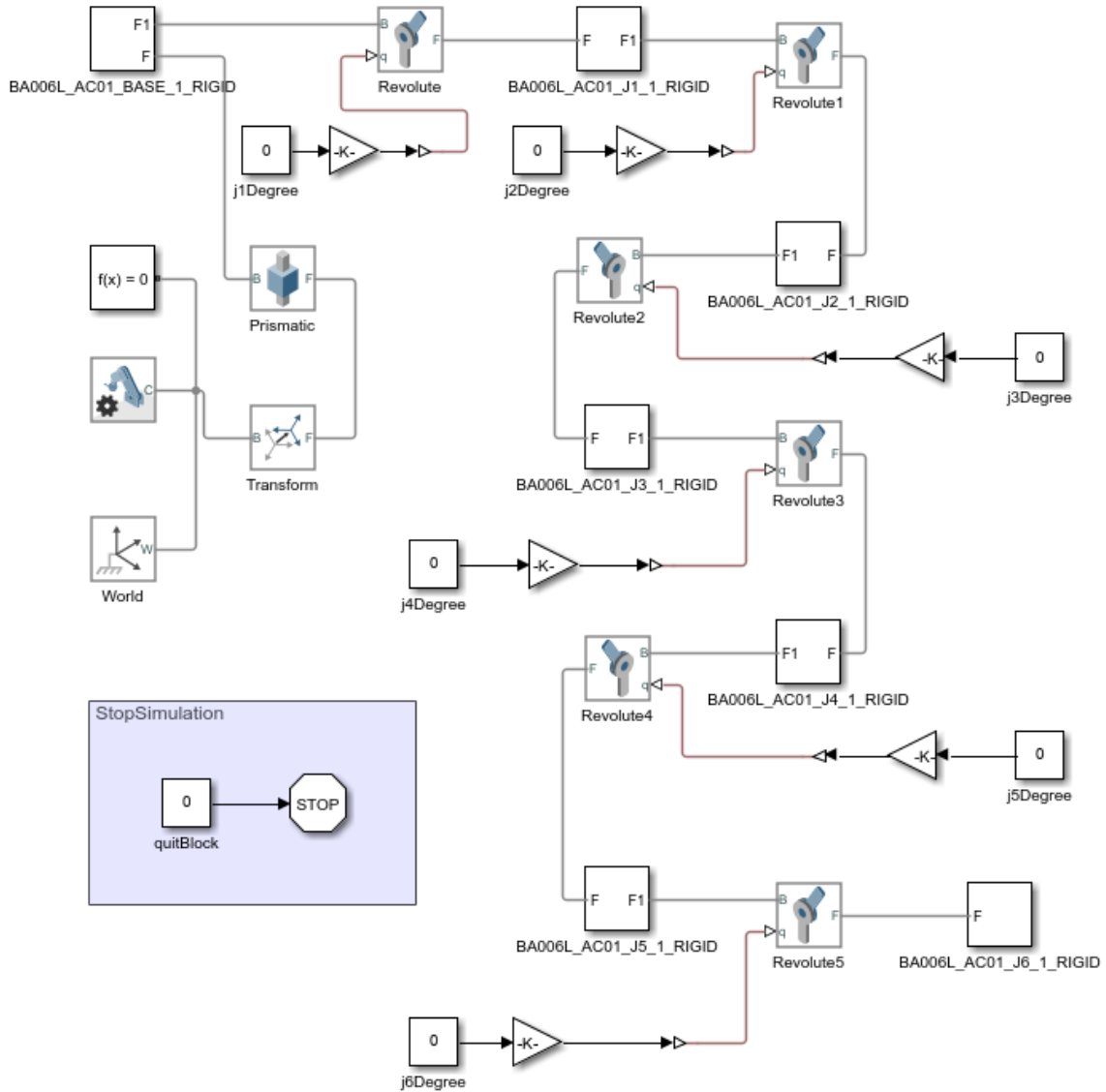
set_param(blockname, 'Value', num2str(app.sim_line1Degree.Value));
```

Simulasyon çalışırken simülasyon içindeki constrain bloklarınının değerlerini değiştirmek için ise yukarıdaki 2 satırlık kod parçasını kullanıyorum. Koddaki “kawasakiRobot” simulink dosyasının adı, “j1Degree” ise değiştirmek istediğimiz bloğun ismidir.

Robotun websitesi üzerinden ulaştığım “.step” dosyalarının solidworks programı üzerinde koordinat eksenlerini düzenleyip, eklentiler kısmından “SimScape Multibody Link” i aktifleştirdince yukarı Araçlar sekmesinden SimScape Multibody Link altındaki Export->SimScapeMultibody’e tıklayarak gerekli step dosyaları ve xml dosyasını elde etmiş oluyoruz.

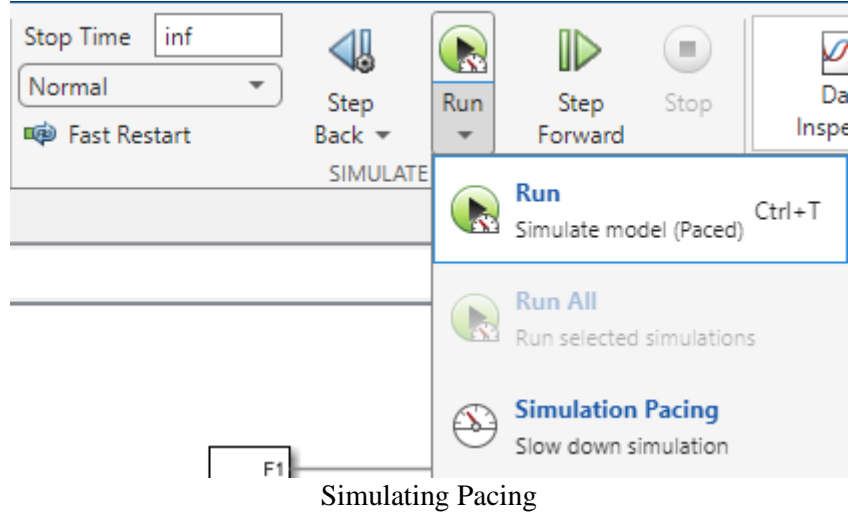
Sonrasında MatlabCommandWindow ‘u kullanarak cd komutlarıyla xml dosyasının olduğu dosya konumuna getirdim matlab’i.

smimport(‘kawasakiRobot.xml’) komutu ile xml ve step dosyalarını kullanarak bir simulink-simscape modeli oluşturdum. Eklemlerdeki torque değerlerini automated computing, motion değerlerini ise provided by input seçerek eklem bloklarına radyan olarak açılış değeri bağlayabileceğim hale getirdim.



Simulink-SimScape Modeli

Açı girişlerine açı-radyan dönüşümü için ( $\pi/180$ ) değerine sahip gain blokları ekledim. Sonrasında açı değerlerini arayüzden yazdırabilmek için constrain ekledim ve bu blokların adını düzenledim.



Ayrıca simülasyonun devamlı olarak çalışması için stop time 'ı inf olarak değiştirip. Simulasyon cycle'ının yavaşlatılması için run butonu altındaki simulation pacing in enable olarak ayarlanması gerekmektedir.

StopSimulation alanında ise simülasyon infite yani devamlı çalıştığı için durdurma koşulu olarak stopSimulation bloğu ekledim ve bu blok 1 değeri aldığında tüm simülasyonu durduruyor. Böylece durdur butonu ile simülasyonu kontrol edebiliyorum.

Genel Simulasyon

DH	a	alfa (Deg)	d	teta (Deg)	Ekleme Tipi	Ekleme Değişkeni
1	0	0	0	0	Kapalı	0.00
2	0	0	0	0	Kapalı	0.00
3	0	0	0	0	Kapalı	0.00
4	0	0	0	0	Kapalı	0.00
5	0	0	0	0	Kapalı	0.00
6	0	0	0	0	Kapalı	0.00
7	0	0	0	0	Kapalı	0.00
8	0	0	0	0	Kapalı	0.00

HESAPLA

3B Çizgisel Robot



X: 0, Y: 0, Z: 0

Yaw: 0, Pitch: 0, Roll: 0

2023-2024 Robotik ve Otomasyon Sistemleri  
Mekatronik Mühendisliği Bölümü



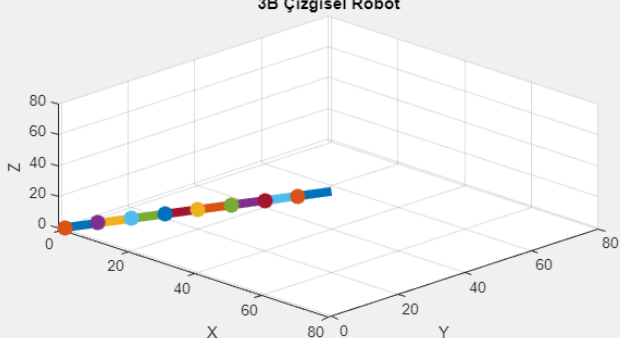
Sekme-1 “Genel” Açılış

Genel Simulasyon

DH	a	alfa (Deg)	d	teta (Deg)	Ekleme Tipi	Ekleme Değişkeni
1	10	0	10	0	Döner(R)	0.00
2	10	0	10	0	Döner(R)	0.00
3	10	0	10	0	Döner(R)	0.00
4	10	0	10	0	Döner(R)	0.00
5	10	0	10	0	Döner(R)	0.00
6	10	0	10	0	Döner(R)	0.00
7	10	0	10	0	Döner(R)	0.00
8	10	0	10	0	Döner(R)	0.00

HESAPLA


3B Çizgisel Robot



X: 80, Y: 0, Z: 80

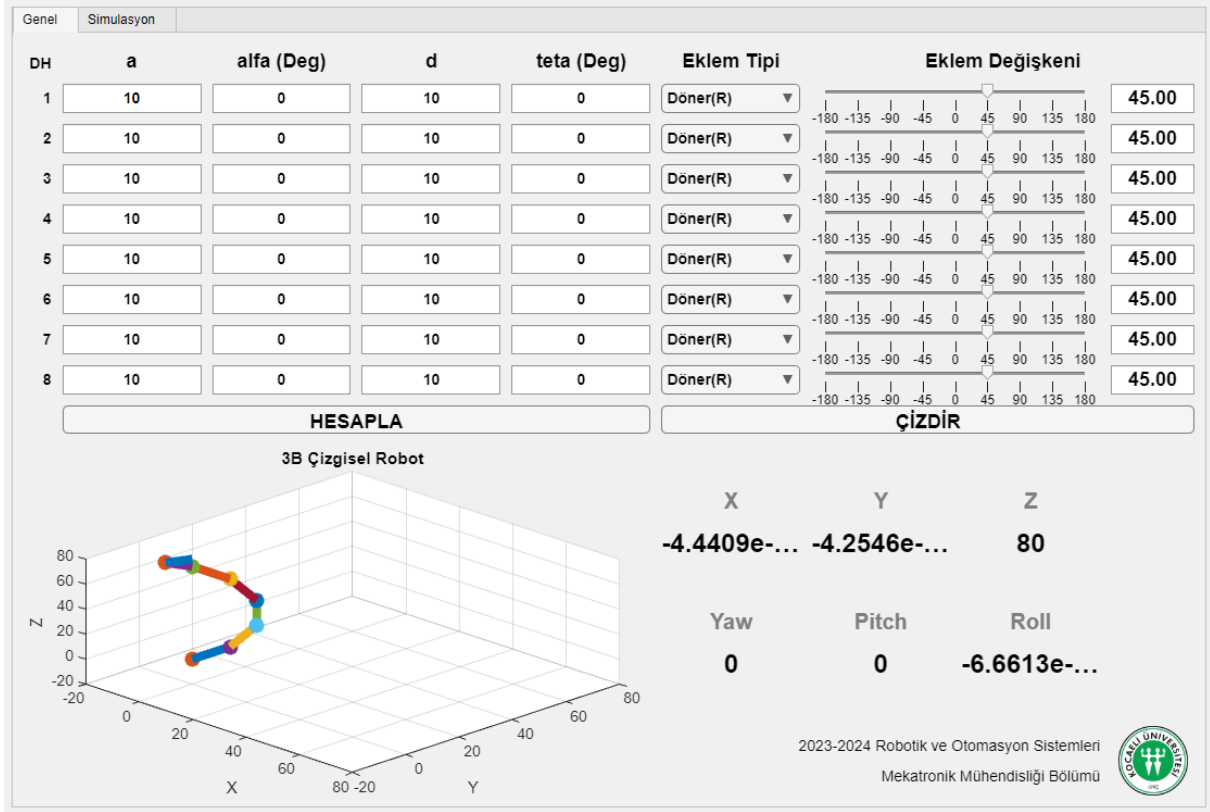
Yaw: 0, Pitch: 0, Roll: 0

2023-2024 Robotik ve Otomasyon Sistemleri  
Mekatronik Mühendisliği Bölümü

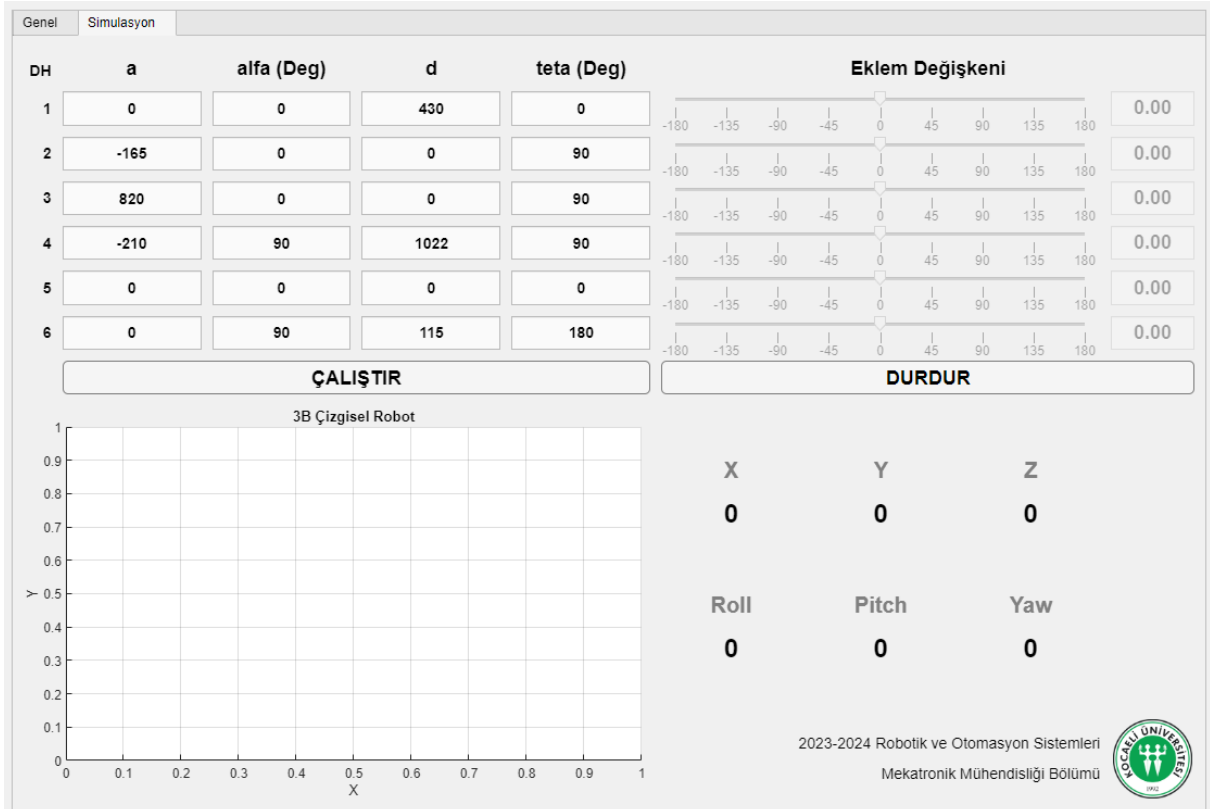


Sekme-1 “Genel” DH tablosu doldurulunca

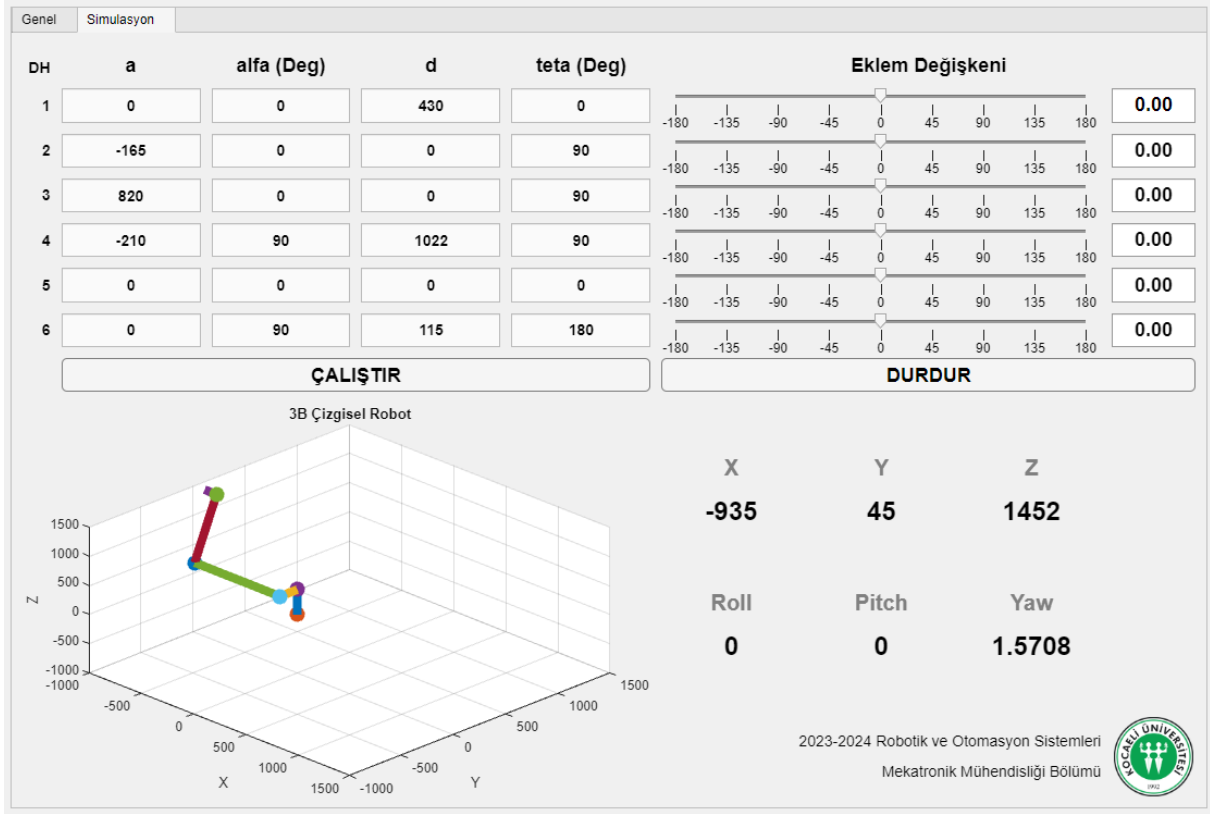




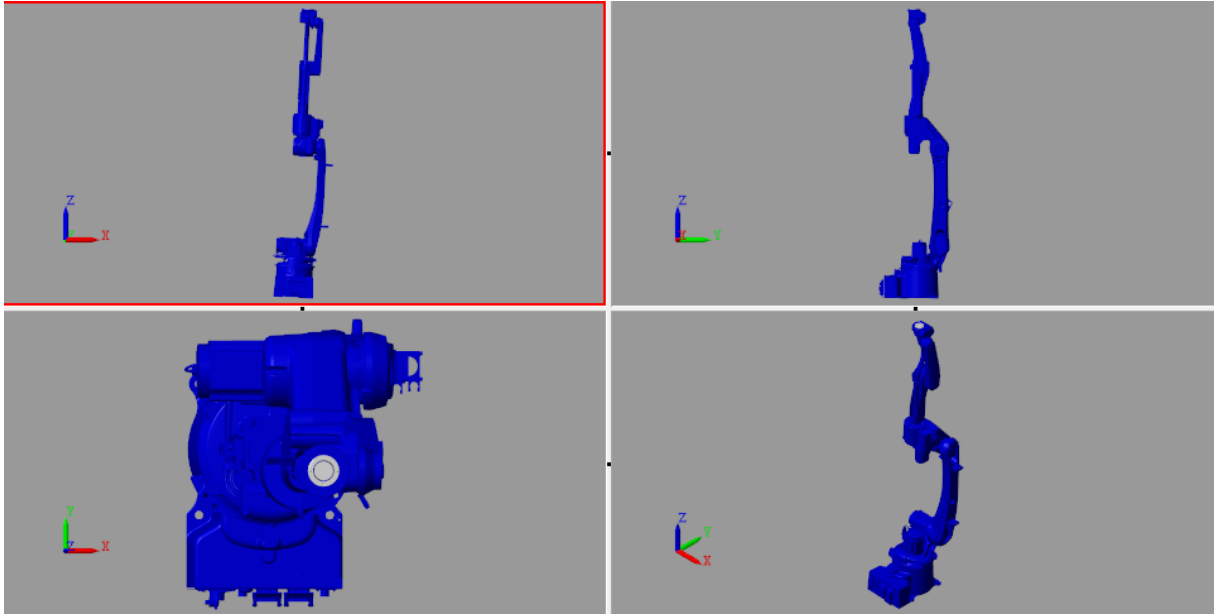
Sekme-1 “Genel” Açılar değiştirilince



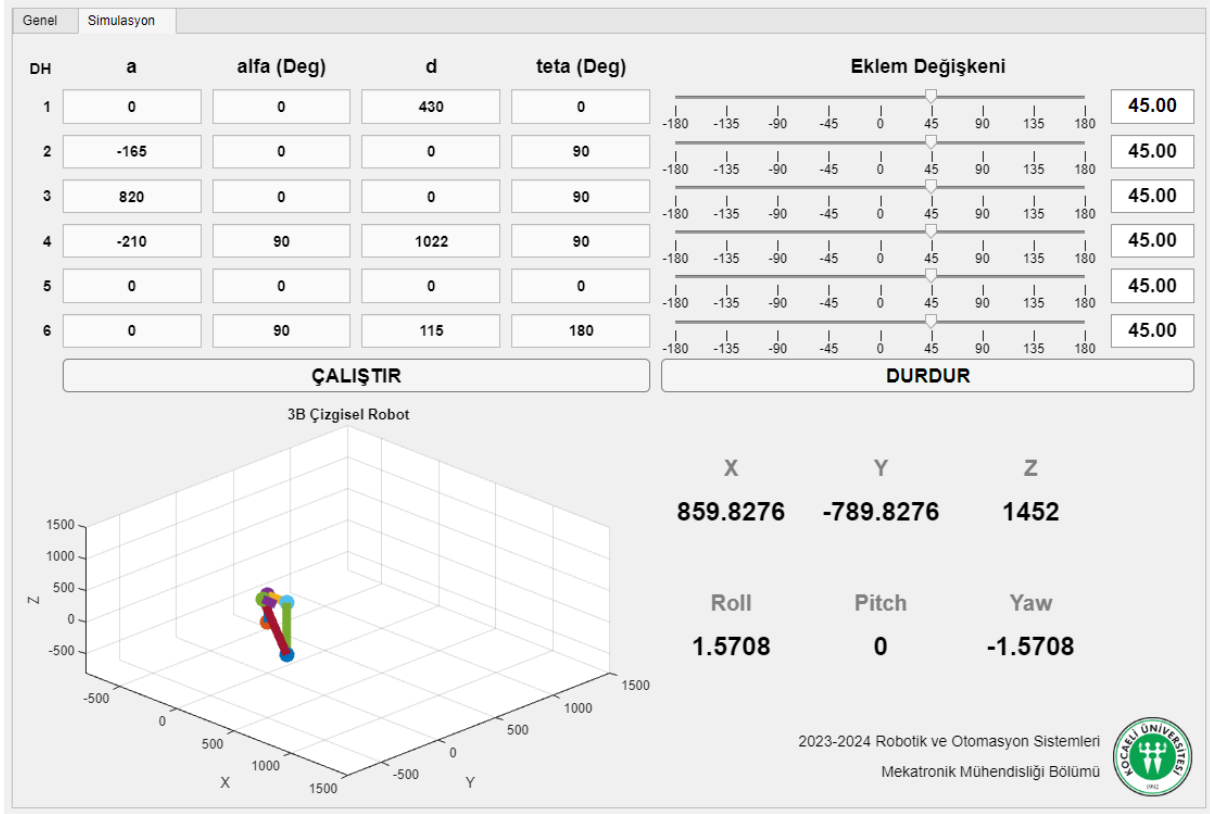
Sekme-2 “Simulasyon” Açılış



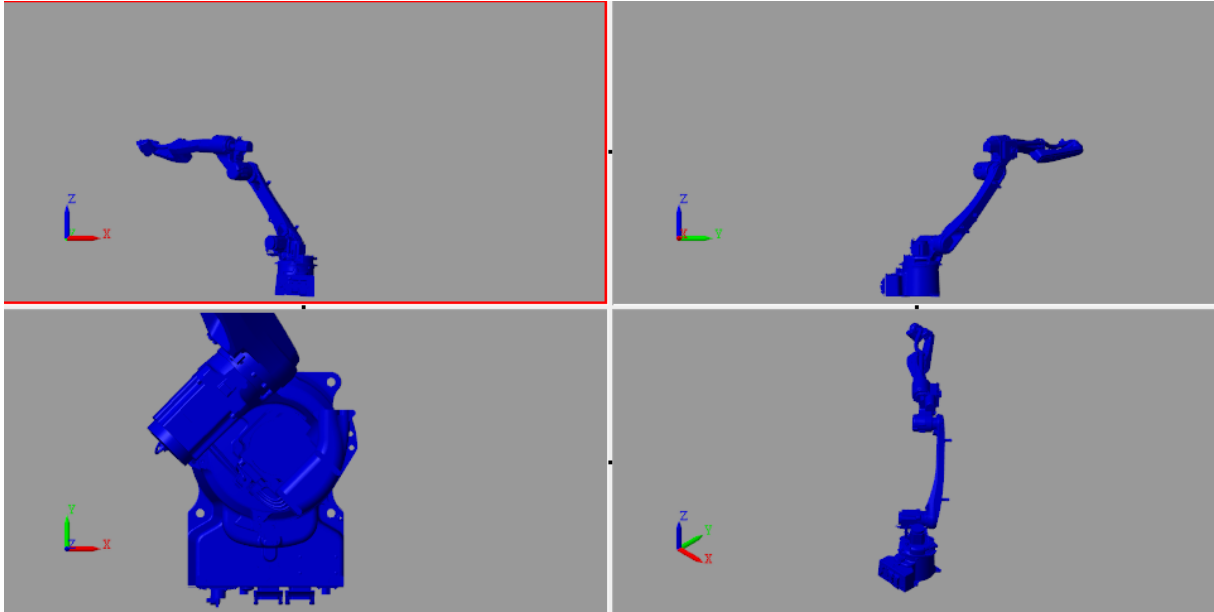
Sekme-2 “Simulasyon” Çalıştırılınca



Sekme-2 “Simulasyon” Çalıştırılınca SimScape



Sekme-2 “Simulasyon” Açılar Değiştirilince



Sekme-2 “Simulasyon” Açılar Değiştirilince SimScape