دانشگاه صنعتی شریف

دانشکده مهندسی مکانیک

عنوان:

تمرینات سری سوم

مدلسازی یک ربات RRP در محیط simulink و حل سینماتیک معکوس آن

نگارش

محمدسعید صافی زاده

استاد راهنما

دکتر بهزادی پور

آبان ماه ۱۴۰۳

فهرست مطالب

۱ صورت سوالات

شکل ۱ربات کروی را نشان میدهد که از ترکیب RRP برای تعیین موقعیت عملگر نهایی استفاده می کند. مقادیر مورد نیاز پارامترهای ربات در شکل مشخص شده اند.

۱.۱ سوال اول

مدل این ربات را در نرم افزار Simmechanic بسازید. مقادیر زوایای مفصلی را مطابق جدول ۱ به مدل خود اعمال کرده و موقعیت دستگاه عملگر نهایی را از نرم افزار خوانده و در جدول زیر وارد کنید.

θ_1(deg)	120	30	90
θ_2(deg)	15	50	20
$d(mm)$	280	160	440
X(mm)			
Y(mm)			
Z(mm)			

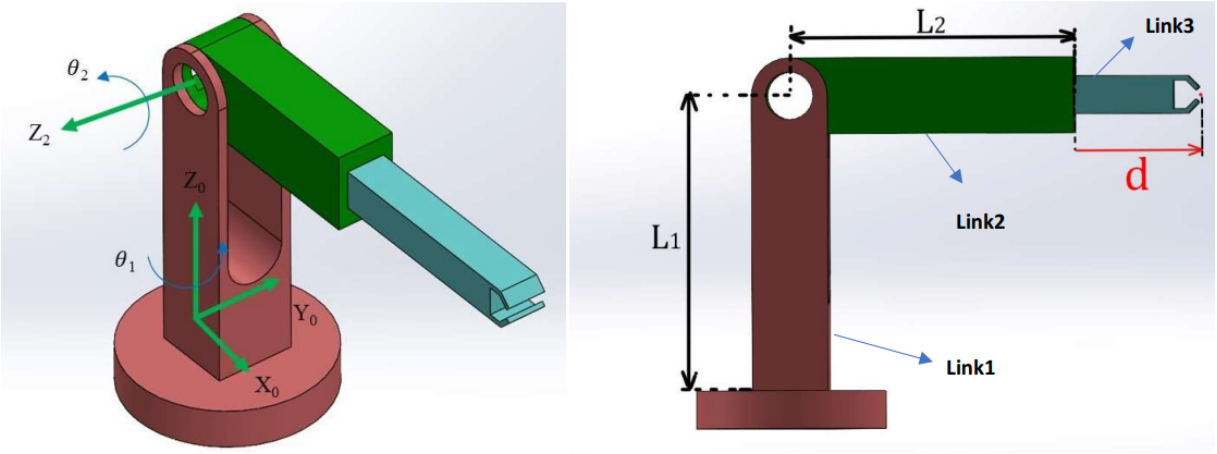
جدول ۱: دیتاهای داده شده برای یافتن موقعیت نهایی end-effector

۲.۱ سوال دوم

در نرم افزار متلب تابعی بسازید که حل سینماتیک معکوس این ربات را انجام دهد. ورودی این تابع مختصات نقطه انتهایی لینک سه و خروجی آن زوایای سه گانه مفاصل (مطابق تعریف قسمت۱) خواهد بود. مقادیر بدست آمده برای مختصات عملگر نهایی را از بخش ۱ به این تابع ارسال کرده و مقادیر زوایای مفاصل را گزارش کنید. تابع خود را با مقادیر جدول ۲ نیز امتحان کنید.

θ_1(deg)			
θ_2(deg)			
$d(mm)$			
X(mm)	294.3	347.9	154.6
Y(mm)	25.7	281.7	-149.3
Z(mm)	297.9	549.3	856.3

جدول ۲: دیتاهای داده شده برای یافتن متغیر های مفصلی به کمک موقعیت نهایی end-effector

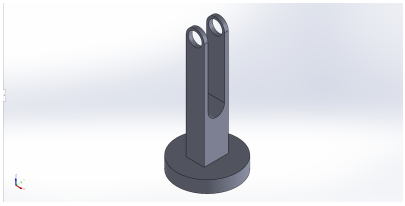


شکل ۱: مفصل چرخشی اول و دوم در این حالت در وضعیت صفر قرار دارند. $(L_1 = 46cm, L_2 = 17cm)$

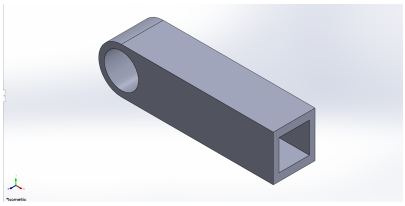
۲ پاسخ سوال اول

۱.۲ مدلسازی ربات در SolidWorks

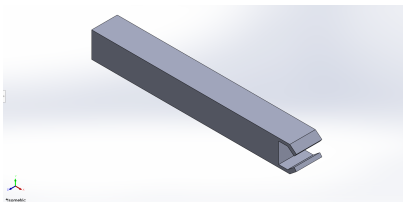
ابتدا باید ربات را در نرم افزار SoliWorks مدل کرد. مطابق با شکل ۲ سه لینک داریم که ابعاد آن ها را به جز L_1, L_2, d به صورت تخمینی وارد می کنیم. نهایتاً سه لینک ما مشابه شکل ۲ خواهند شد.



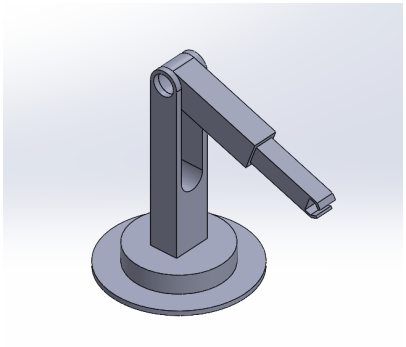
(آ) لینک ۱



(ب) لینک ۲



(ج) لینک ۳



(د) ربات مونتاژ شده

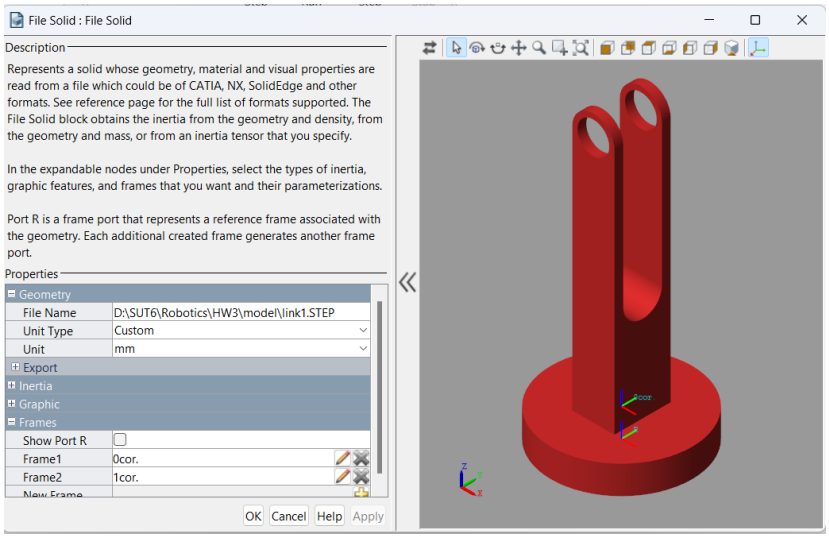
شکل ۲: شکل لینک های مدل شده در SolidWorks

در هر سه شکل، به دستگاه مختصاتی که جسم در آن قراردارد توجه کنید. مدلسازی به صورتی انجام شده است که با محور های محیط simulink هم خوانی داشته باشد. اما لزوماً نیازی به انجام این کار نبود چرا که در محیط simulink می توانستیم با انتخاب دستگاه ها جهت قرارگیری جسم را تعیین کنیم. در حالی که الان کار ما بسیار آسان تر خواهد بود.هم چنین در شکل ۲(د) ربات assemble[1] شده را مشاهده می کنید.

[1] مونتاژ

۲.۲ مدلسازی در Simulink

برای مدلسازی در سیمولینک باید یک محیط multibody ایجاد کرده و سپس به کمک بلوک های file solid و rigid transform به مدلسازی پرداخت. ابتدا لینک یک را به کمک بلوک file solid وارد می کنیم. به این صورت که در سربرگ Geometry و در بخش File Name فایل کد[۲] خود را با پسوند STEP. انتخاب می کنیم. Unit Type[۳] را بر روی Custom قرار داده و Unit را به mm تغییر می دهیم. در سربرگ Inertia گزینه Based On را بر روی Custom Density قرار می دهیم و در بخش Density چگالی مورد نظرمان را وارد می کنیم. در سربرگ Graphics و از زیر بخش Visual Properties از قسمت Color رنگ دلخواهمان را بر می گزینیم و نهایتاً در سربرگ frames گزینه Show Port R را غیرفعال می کنیم. مطابق با شکل ۳ یک فریم جدید را با کلیک بر روی گزینه New Frame تعریف می کنیم تا به کمک مفصل revolute به زمین متصل شود. در ادامه به نحوه ایجاد فریم می پردازیم.

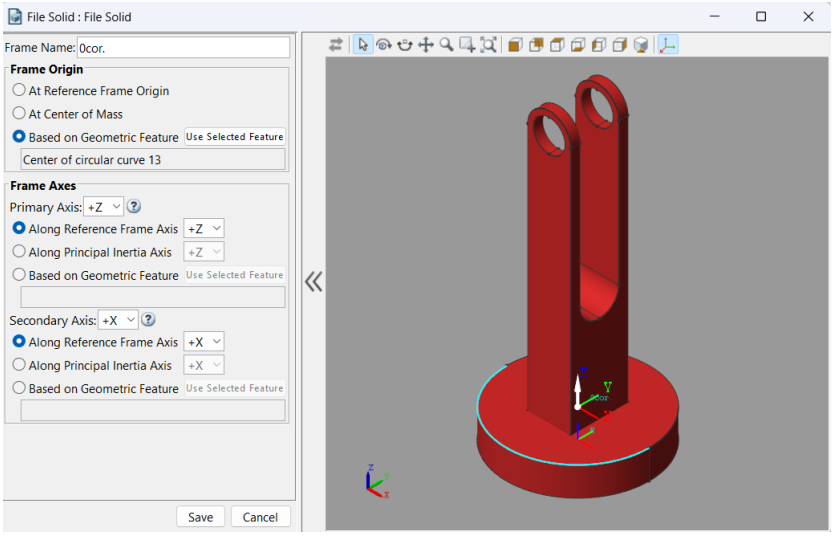


شکل ۳: نمایش فریم مفصل اول

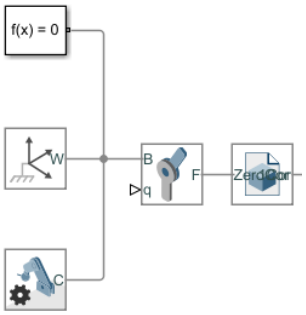
برای ایجاد فریم، با توجه به مبدا مختصات مشخص شده در شکل ۱، فریم لینک یک را با توجه به شکل ۴ انتخاب می کنیم. در بخش frame origin گزینه Based on geometry feature را بر می گزینیم و با انتخاب قطاع دایره مد نظر و سپس کلیک بر روی use selected feature فریم جدید را ایجاد می کنیم.
اکنون فریم تعیین شده را به عنوان یک follower[۴] به فریم world[۵] به وسیله ی یک revolute joint[۶] مطابق با شکل ۵ وصل می کنیم.

[2] CAD

[۳] واحد اندازه گیری
[۴] دنبال کننده
[۵] دستگاه مرجع
[۶] مفصل دورانی

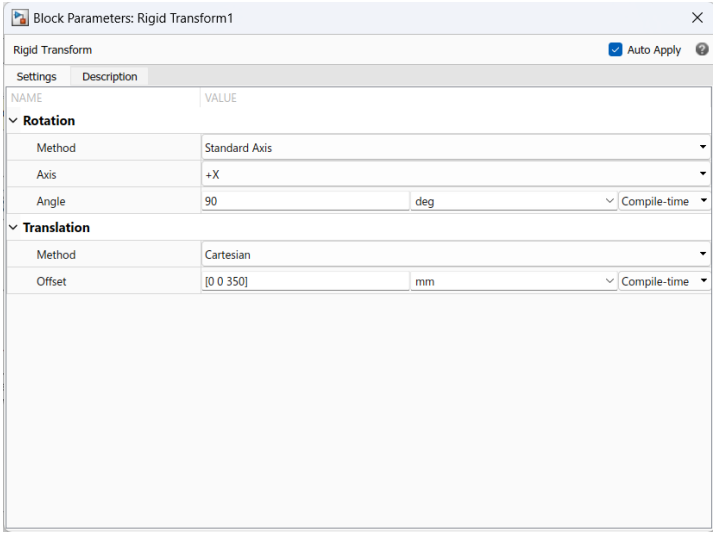


شکل ۴: نحوه ایجاد فریم برای لینک ۱

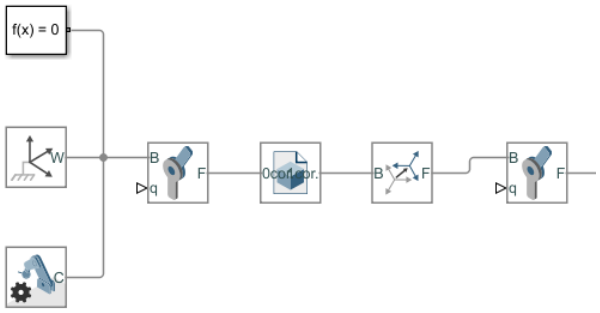


شکل ۵: نحوه اتصال لینک ۱ به زمین

اکنون برای اتصال لینک دو به لینک یک نیاز به تعریف یک دستگاه مختصات جدید داریم. می دانیم که مفصل revolute حول محور z دوران می کنند بنابراین z مختصات جدید ما باید در راستای y کنونی باشد. برای این کار کافی است یک مختصات جدید منطبق بر فریم قبلی که در شکل ۳ داشتیم تعریف کرده و با استفاده از بلوک Rigid Transform این دستگاه را به نقطه ی دلخواهمان ببریم که در اینجا حرکت به اندازه 350 mm در راستای Z و یک دوران به اندازه ۹۰ درجه حول محور $+X$ می باشد که این مقادیر را در بلوک Rigid Transform نیز مطابق با شکل ۶ وارد می کنیم و این مختصات جدید را به یک مفصل دورانی متصل می کنیم. تا اینجا بلوک دیاگرام ما به شکل ۷ در آمده است.



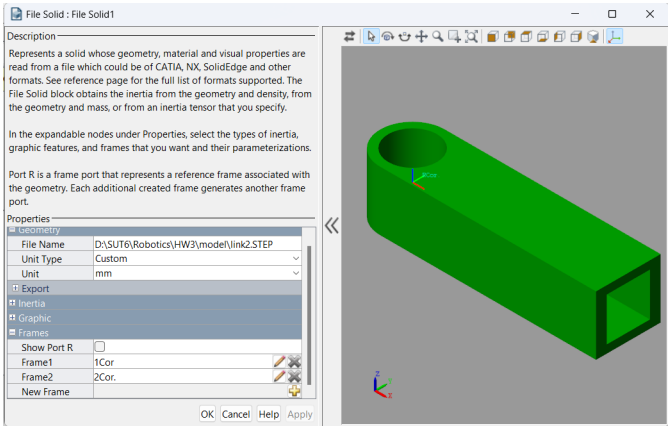
شکل ۶: تنظیمات rigid transform block برای ساختن مختصات مفصل دو



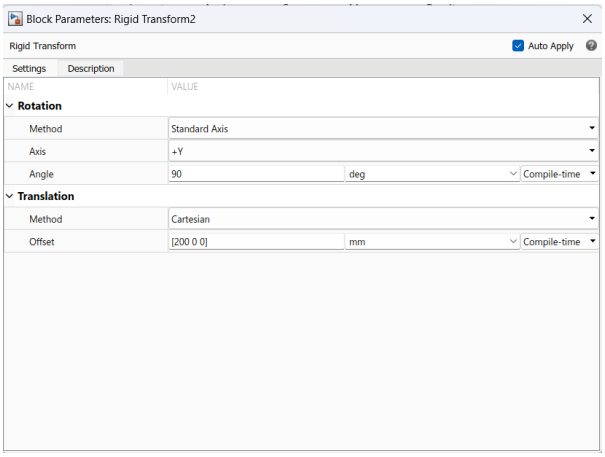
شکل ۷: بلوک دیاگرام تا این مرحله

در این مرحله دوباره به کمک بلوک file solid لینک دوم را وارد می کنیم و مختصات آن را با ایجاد یک فریم جدید به گونه ای تعیین می کنیم که محور Z در راستای محور مفصل بیفتد و این امر به گونه ای که در شکل ۸(آ) مشاهده می کنید رخ می دهد.
لینک دوم را نیز در بلوک دیاگرام به مفصل متصل می کنیم و دوباره مانند قسمت قبل یک فریم روی فریم آخر ایجاد می کنیم و نام آن را 2Cor. می گذاریم و با Rigid Transform که تنظیمات آن در شکل ۸(ب) آمده است؛ آن را به محل مناسب برای اتصال مفصل سوم منتقل می کنیم. توجه داشته باشید که مفصل prismatic[۷] در راستای Z حرکت می کند بنابراین باید فریم را علاوه بر انتقال در جهت Z حول محور Y دوران دهیم تا در راستای مفصل انتقالی باشد. که این توضیحات در شکل ۸(ب) به چشم می خورند. حالا مفصل انتقالی را به این فریم متصل می کنیم و بلوک دیاگرام ما به صورت شکل ۹ در خواهد آمد.

[۷] انتقالی

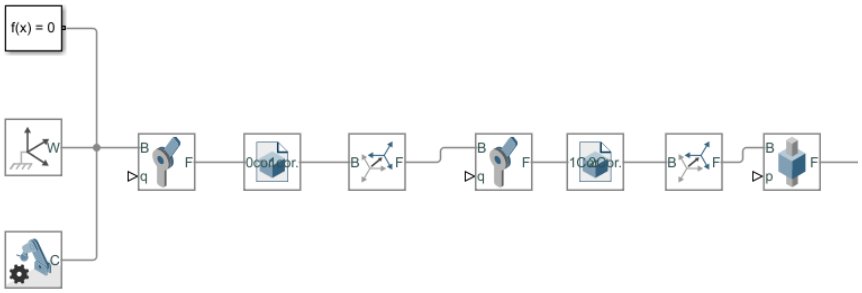


(آ) نمایش فریم لینک دوم



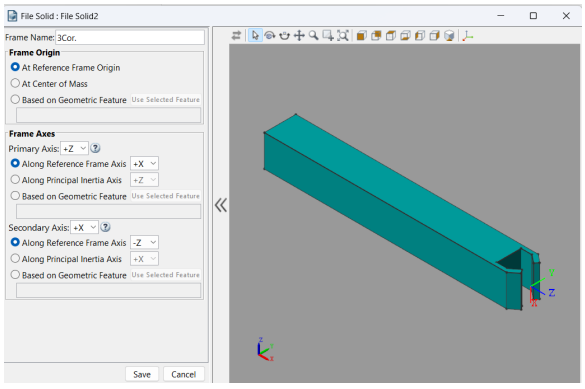
(ب) تنظیمات rigid transform block برای ساختن مختصات مفصل سه

شکل ۸: نمایش فریم لینک دوم و تنظیمات Rigid Trnsform



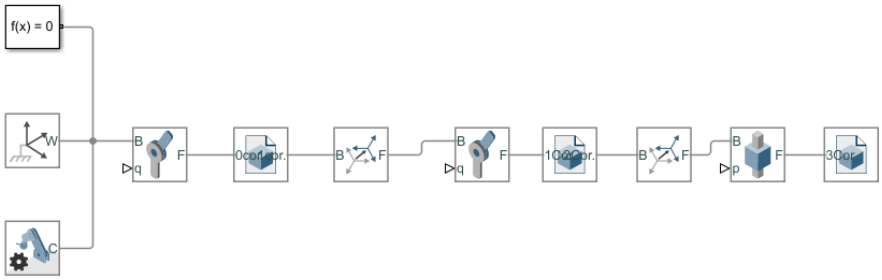
شکل ۹: بلوک دیاگرام تا این مرحله

در این مرحله لینک سوم را وارد می کنیم و فریم خودش را به عنوان فریم جدید تعریف می کنیم و این متختصات را به اندازه 90- درجه حول محور Y دوران می دهیم.(شکل ۱۰) و سپس این فریم را به مفصل انتقالی متصل می کنیم.



شکل ۱۰: نمایش فریم لینک سوم

بلوک دیاگرام ما تا این مرحله مطابق شکل ۱۱ می شود.



شکل ۱۱: بلوک دیاگرام تا این مرحله

۳.۲ تعیین ورودی های مفاصل و خروجی آن ها

در این بخش به نحوه ی دادن زوایا و جابه جایی مفصل ها مطابق با جدول ۱ می پردازیم.
برای این کار راه حل های متفاوتی داریم. برای مثال می توانیم این متغیر ها را یک به یک درون زوایا و جابه جایی های اولیه مفاصل اعمال کنیم و یا از بلوکconstant استفاده کنیم ولی در اینجا برای بهتر کردن عملکرد و ساده شدن کد از بلوک های from workspace ، Demux ، simulink-PS converter ، trsnsform sensor ، PS-simulink converter ، Mux و نهایتاً to workspace استفاده کرده ام که در ادامه به توضیح هرکدام می پردازم.

۱.۳.۲ بلوک from workspace

ابتدا در workspace متلب، ماتریسی به صورت کد زیر برای هر داده تعریف می کنیم به گونه ای که ستون اول این ماتریس نمایان گر زمانی باشد که این مقادیر باید اعمال شوند. اسم این فایل را JointData می گذاریم. ورودی مفاصل می توانند به درجه یا رادیان و میلی متر یا متر باشند که خودمان این واحد ها را در simulinnk-ps converter وارد می کنیم. برای راحتی و ماموس تر بودن در اینجا از میلی متر برای واحد طول و از درجه برای واحد زاویه استفاده شده است.

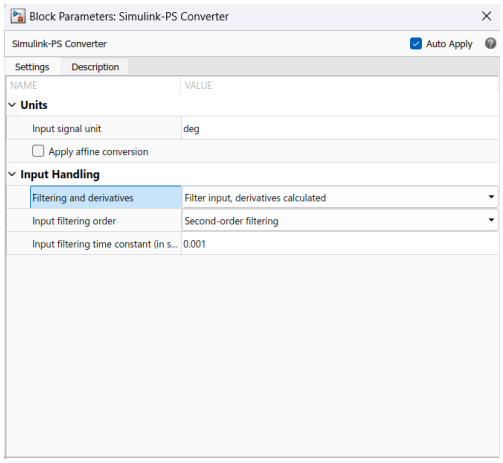
```
% first column is the time at whih the variables are read
% second column is theta 1 in deg
% third column is theta 2 in deg
% forth column is theta d in mm
jointData = [0, 120, 15,280;
2, 30, 50, 160;
4, 90, 20, 440;
];
```

حال وارد محیط سیمولینک می شویم و بلوک from workspace را وارد می کنیم و با دابل کلیک بر روی آن تنظیمات آن را اصلاح می کنیم. در بخش VariableName مقدار JointData را وارد می کنیم و در بخش sample time مقدار ۲ را وارد می کنیم تا هر دوثانیه دیتا دریافت کند و نهایتا holding final value را برمی گزینیم تا داده آخر را نگه دارد.

۲.۳.۲ بلوک Demux

برای اینکه درایه های هر سطر ماتریس را به یک مفصل بدهیم نیاز است که هر سطر را به سه عدد بشکنیم که با بلوک Demux این امر امکان پذیر خواهد بود. فقط کافی است بعد از وارد کردن این بلوک با دیل کلیک بر روی آن مقدار number of outputs را بر روی ۳ قرار دهید. نهایتاً خروجی آن را که از جنس سیگنال سیمولینک است با بلوک simulink-PS converter به مفصل ها می دهیم. البته ذکر دو نکته ضروری است. اول اینکه باید واحد تبدیل این سیگنال را درست تنظیم کنیم. به این منظور با دبل کلیک بر روی آیکون simulink-ps converter

وارد تنظیمات آن می شویم و در بخش input signal unit واحد مورد نظرمان را (deg, mm) وارد می کنیم. دوم اینکه برای دادن ورودی به مفاصل نیازمند مشتق دوم ورودی داریم به این منظور نیز از منوی filtering and derivatives گزینه ی filtr input, derivatives calculated را انتخاب و هم چنین از منوی input filtering order گزینه second order filtering را انتخاب می کنیم.می توانید نحوه اعمال این تغییرات را در شکل ۱۲ مشاهده کنید.



شکل ۱۲: تنظیمات بلوک simulink-ps converter

در انتها نیز، این سیگنال ها را به مفاصل متصل می کنیم. البته قبل از آن باید در مفاصل از بخش actuation ، قسمت torque را به Automat-ically computed و Motion را به provided by input تغییر دهیم.

۳.۳.۲ بلوک transform sensor

از این بلوک برای یافتن موقعیت end-effector استفاده می کنیم. با دبل کلیک بر روی آن و تیک زدن گزینه های X, Y, Z از منوی Translation و دادن follower و Base مناسب موقعیت end-effector را خواهیم یافت.

۴.۳.۲ بلوک Mux

بعد از دریافت موقعیت end-effector این سیگنال ها را به وسیله PS-simulink converter به سیگنال های simulink تبدیل می کنیم. برای اینکه بتوانیم این این سه مقدار را کنار هم داشته باشیم از بلوک Mux استفاده می کنیم.

۴.۲ بلوک matlab fcn

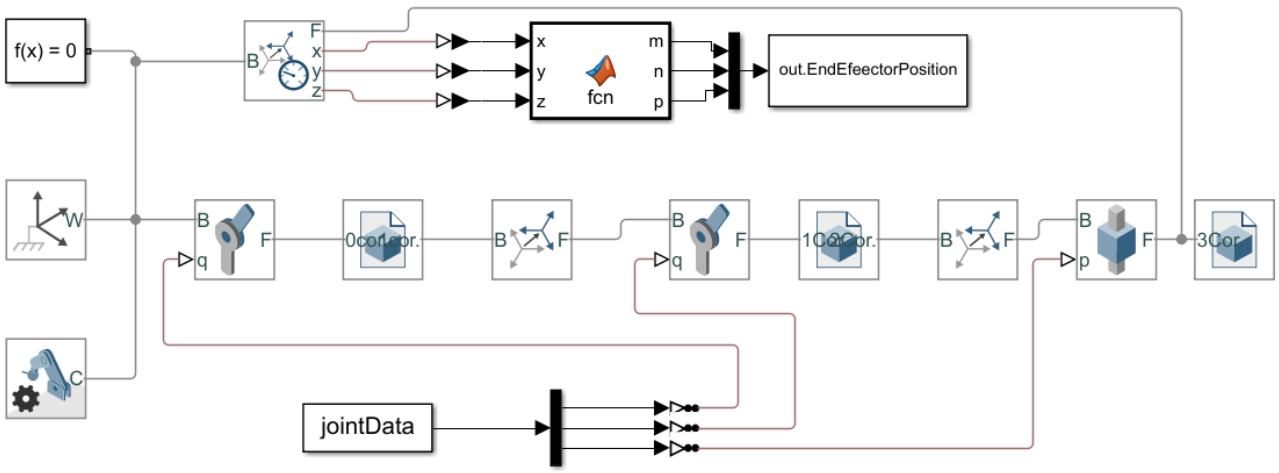
همانطور که در تمرین قبل دیدیم، حل عددی شامل خطاهایی است که در خروجی مقدار صفر مطلق دریافت نمی کنیم برای همین تابعی نوشتم که به عنوان یک threshold عمل کند که کد آن در ادامه آورده شده است. این تابع در صورتی که مقادیر ما از عدد threshold کوچک باشند آن ها را صفر در نظر می گیرد.

```
function [m,n,p]= fcn(x,y,z)
if abs(x) < 10^-3
m = 0;
else
m = x;
end

if abs(y) < 10^-3
n = 0;
else
```

```
n = y;
end

if abs(z) < 10^-3
p = 0;
else
p = z;
end
end
```

۱.٤.۲ بلوک to workspace

برای گرفتن خروجی ها از بلوک to workspace استفاده می کنیم تا داده ها را به صورت یک جدول در خروجی مشاهده کنیم. بلوک دیاگرام نهایی ما به شکل ۱۳ در خواهد آمد.



شکل ۱۳: بلوک دیاگرام تا مرحله نهایی

بعد از اجرای شبیه سازی در سیمولینک داده های خروجی و جدول تکمیل شده به صورت جدول ۳ خواهند بود. ستون اول موقعیت X ، ستون دوم موقعیت Y و ستون سوم موقعیت Z در end-effector است که همگی گزارش شده تا دو رقم اعشار و به mm هستند.

θ_1(deg)	120	30	90
θ_2(deg)	15	50	20
$d(mm)$	280	160	440
X(mm)	-231.82	200.40	0
Y(mm)	401.53	115.70	601.40
Z(mm)	474.23	625.78	568.89

جدول ۳: داده های خروجی برای موقعیت end-effector به ازای داده های ورودی

بنابراین موقعیت end-effector را برای زوایا و جهت های داده شده را پیدا کردیم. اکنون به سراغ بخش دوم سوال می رویم.

θ_1(deg)	120	30	90
θ_2(deg)	15	50	20
$d(mm)$	280	160	440
X(mm)	-231.822198309376	200.401342881761	-3.82104254876537e-05
Y(mm)	401.527825794148	115.701757870167	601.403300220489
Z(mm)	474.233141649210	625.776000863956	568.892878434144

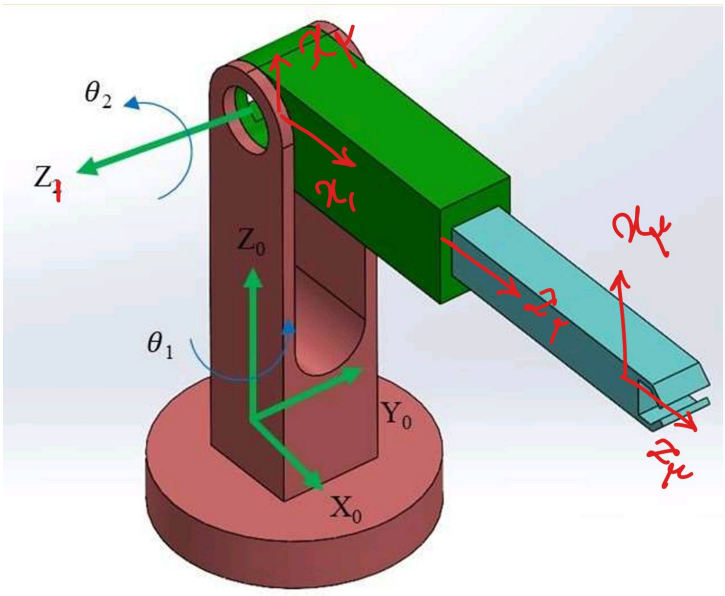
جدول ۴: داده های دقیق خروجی بدون اعمال thresholdبرای موقعیت end-effector به ازای داده های ورودی

۳ پاسخ سوال دوم

در این بخش با روش D-H مقدار ماتریس همگن را به صورت پارامتری به وسیله یک تابع می یابیم و سپس معادلات غیرخطی ای را برای یافتن زوایا و فاصله مفصل ها بر اساس موقعیت end-effector حل می کنیم.

۱.۳ تعریف دستگاه ها و زوایا

در شکل ۱۴ ، دستگاه مختصات های هر لینک آورده شده اند. ابتدا با دستگاه مختصات صفرم شروع می کنیم. راستای Z آن در جهت مفصل و راستای X آن دلخواه است. سپس به سراغ مفصل دوم می رویم و دستگاه دو را به شیوه ای که باز محور Z دستگاه در راستای مفصل دو بیفتد قرار می دهیم اما X_1 عمود مشترک Z_0 و Z_1 می شود که چون این دو متقاطع هستند، X_1 در محل برخورد آن ها و عمود به صفحه دو بردار رسم می شود. دستگاه مختصات بعدی را روی مفصل سوم به شیوه ای که محور Z_2 در راستای محور مفصل باشد قرار می دهیم. در این حالت Z_1 و Z_2 نیز متقاطع هستند بنابراین X_2 را در نقطه تقاطع و عمود به صفحه این دو رسم می کنیم. نهایتا محور Z دستگاه سوم را دلخواه انتخاب می کنیم. با توجه به شکل Z_3 موازی Z_2 می شود که یعنی بی شمار عمود مشترک داریم و یکی را به دلخواه X_3 می نامیم.



شکل ۱۴: نمایش دستگاه های تعریف شده برای یافتن جدول DH

هم چنین برای متغیرهای D-H داریم :

- زاویه θ_1 زاویه است که محور X_0 را با محور X_1 موازی می کند.
- l_1 اندازه ای است که باید در راستای Z_0 طی کنیم تا به دستگاه یک برسیم.

- α_1 زاویه ای است که محور Z_0 را به محور Z_1 تبدیل می کند.
- زاویه θ_2 زاویه است که محور X_1 را با محور X_2 موازی می کند.
- α_2 زاویه ای است که محور Z_1 را به محور Z_2 تبدیل می کند.
- $d + l_2$ اندازه ای است که باید در راستای Z_2 طی کنیم تا به دستگاه سه برسیم.

تمامی این متغیر های در جدول ۵ به صورت خلاصه آورده شده اند.

	θ	d	l	α
1	$\theta_1^\star$	l_1	0	$\frac{\pi}{2}$
2	$\theta_2^\star$	0	0	$\frac{\pi}{2}$
3	0	$d^\star + l_2$	0	0

جدول ۵: $\star$ نمایش دهنده متغیر ها است

۲.۳ نوشتن ماتریس همگن و توضیح کد متلب آن

می دانیم که

$$H_3^0 = H_1^0 H_2^1 H_3^2$$

و هم چنین برای مثال، H_1^0 حاصل چهار تبدیل استاندارد است که از ردیف اول جدول ۵ بدست می آید. بنابراین داریم:

$$H_1^0 = R_{z,\theta_1} T_{z,l_1} R_{x,\frac{\pi}{2}}$$

$$H_2^1 = R_{z,\theta_2} R_{x,\frac{\pi}{2}}$$

$$H_3^2 = T_{z,d+l_2}$$

نهایتاً ماتریس همگن ما به صورت ضرب چند ماتریس همگن استاندارد در می آید که در زیر آمده است:

$$H_3^0 = R_{z,\theta_1} T_{z,l_1} R_{x,\frac{\pi}{2}} R_{z,\theta_2} R_{x,\frac{\pi}{2}} T_{z,d+l_2}$$

اکنون این ماتریس را به کمک تمرین سری اول در متلب وارد می کنیم.
برای ماتریس همگن انتقال استاندارد تابع زیر را داریم:

```
function T=Trans (axis, distance)
%Homogenous transformation along "axis" for the amount of "distance"
axis=upper(axis);
if (axis == 'X')
T= [1 0 0 distance; 0 1 0 0; 0 0 1 0; 0 0 0 1];
end
if (axis == 'Y')
T= [1 0 0 0; 0 1 0 distance; 0 0 1 0; 0 0 0 1];
end
if (axis =='Z')
T= [1 0 0 0; 0 1 0 0; 0 0 1 distance; 0 0 0 1];
end
```

برای ماتریس همگن دوران استاندارد نیز تابع زیر را داریم:

```
function R=Rot(axis,angle)
%Homogenous transfermation around an axis for the amount of "angle"
% input for axis should be in this format -> 'axis'
% angle is in degrees if you like to input angle in radians uncommect
% the below code line
%angle = 180*angle / pi; %converting rad to degree
axis = upper(axis);
if axis == 'X'
R=[1 0           0            0;
0 cosd(angle) -sind(angle) 0;
0 sind(angle) cosd(angle)  0;
0 0           0            1];
end

if axis == 'Y'
R=[cosd(angle)  0 sind(angle) 0;
0            1 0           0;
-sind(angle) 0 cosd(angle) 0;
0            0 0           1];
end

if axis == 'Z'
R=[cosd(angle) -sind(angle) 0 0;
sind(angle) cosd(angle)  0 0;
0           0            1 0;
0           0            0 1];
end
end
```

به کمک این دو تابع، ماتریس همگن را به صورت پارامتری می نویسیم تا در ادامه با مساوی قرار دادن آن با موقعیت نهایی end-effector و حل معادلات به متغیر های مفصلی برسیم.

```
function z=InvKin(x, y, z)
l1 = 350;
l2 = 200;
syms theta1 theta2 d
H1_0 = Rot('z',theta1) * Trans('z',l1) * Rot('x',90);
H2_1 = Rot('z', theta2) * Rot('x',90);
H3_2 = Trans('x',(d+l2));
H = H1_0 * H2_1 * H3_2;
H = H * [0; 0; 0; 1];
P = [x; y; z];
eq1 = H(1) == P(1);
eq2 = H(2) == P(2);
```

```
eq3 = H(3) == P(3);
s = solve([eq1, eq2, eq3]);
z = zeros(length(s.d),3);
for i=1:length(s.d)
theta1 = vpa(s.theta1);
theta2 = vpa(s.theta2);
d = vpa(s.d);
z(i,1) = theta1(i);
z(i,2) = theta2(i);
z(i,3) = d(i);
end
disp(z);
end
```

در ابتدا تابعی به نام InvKin تعریف کردم که موقعیت عملگر نهایی را دریافت می کند.در خط بعد متغیر های ثابت مسئله یعنی l_1, l_2 را تعریف کردیم. متغیر های θ_1, θ_2 و d را به صورت پارامتری تعریف و ماتریس های همگن بین هر دستگاه را مطابق با توضیحات بالا و جدول ۵ می یابیم. سپس با ضرب این ماتریس های همگن در بردار $[0001]^T$ به بردار موقعیت عملگر نهایی به صورت پارامتری از متغیر های مفصلی می رسیم. بردار P همان موقعیت نهایی عملگر می باشد. در ادامه ۳ معادله داریم که از مساوی قرار دادن درایه های بردارهای موقعیت بدست می آیند. حلقه ی for را برای نمایش بهتر نوشته ام تا همه خروجی با هم نمایش داده شوند.

۳.۳ یافتن خواسته های مسئله قسمت اول

در این بخش مختصات هایی را که در جدول ۳ یافته بودیم به تابع نوشته شده در قسمت بالا می دهیم تا خروجی ها را مقایسه کنیم.

```
InvKin(-231.8222,401.5278,474.2331)
InvKin(200.4013,115.7018,625.7760)
InvKin(0,601.4033,568.8929)
```

```
ans =

  -60.0000  -15.0000 -680.0000
  120.0000   15.0000  280.0000
  120.0000 -165.0000 -680.0000
  -60.0000  165.0000  280.0000


ans =

 -150.0000  -50.0000 -560.0000
   30.0000   50.0000  160.0000
   30.0000 -130.0000 -560.0000
 -150.0000  130.0000  160.0000

Warning: Unable to solve symbolically. Returning a numeric solution
using <a href="matlab:web(fullfile(docroot,
'symbolic/vpasolve.html'))">vpasolve</a>.

ans =

   90.0000   20.0000  440.0000
```

شکل ۱۵: خروجی تابع معکوس سینماتیک

۴.۳ تحلیل داده ها

همانطور که در شکل ۱۵ داریم دو جواب اول چهار پاسخ دارند. دلیل این امر وجود توابع مثلثاتی در معادلات غیرخطی ما است چون توابع مثلثاتی دوره تناوب دارند. در واقع این چهار جواب از لحاظ ریاضی قابل قبول هستند اما از نظر فیزیکی، آن دسته از جواب هایی که d منفی دارند مورد قبول نیستد چرا که در آن صورت لینک ۳ با مفصل دو برخورد خواهد داشت. بنابراین در دو جواب اول، سطر دوم و چهارم از نظر فیزیکی هم پذیرفته هستند.(البته باید دقت کرد در این جا فرض شده است تمام نقاط در workspace داده شده اند.) برای حالت سوم، نرم افزار متلب نتواسته است که معادلات را به صورت تحلیلی حل کند برای همین از دستور vpasolve استفاده کرده و معادلات را حل عددی کرده است. همین امر موجب شده است تا یک جواب داشته باشیم. دلیل این اتفاق این است که ما در قسمت های قبل برای اعداد بسیار کوچک threshold تعریف کردیم. اگر این کار را انجام نمی دادیم و مقادیر کوچک را هر چقدر کم به تابع می دادیم، می توانست معادلات را حل تحلیلی کند. یعنی معادله آخر هم چهار جواب دارد که با توجه به حدس اولیه دستور vpasolve ما فقط به یک جواب آن رسیده ایم. اکنون برای یافتن چهار جواب معادله آخر بلوک تابع را حذف می کنم و اعداد دقیق را به تابع InvKin می دهم. در واقع خروجی داده های جدول ۴ را بدست می آورم. این مورد در شکل ۱۶ آورده شده است.

```
InvKin(-231.822198309376,     401.527825794148,     474.233141649210)
InvKin(200.401342881761,      115.701757870167,     625.776000863956)
InvKin(-3.82104254876537e-05, 601.403300220489,     568.892878434144)
```

```
ans =

  -60.0000  -15.0000 -680.0000
  120.0000   15.0000  280.0000
  120.0000 -165.0000 -680.0000
  -60.0000  165.0000  280.0000


ans =

 -150.0000  -50.0000 -560.0000
   30.0000   50.0000  160.0000
   30.0000 -130.0000 -560.0000
 -150.0000  130.0000  160.0000


ans =

  -90.0000  -20.0000 -840.0000
   90.0000   20.0000  440.0000
   90.0000 -160.0000 -840.0000
  -90.0000  160.0000  440.0000
```

Published with MATLAB® R2022a

شکل ۱۶: خروجی تابع معکوس سینماتیک برای مقادیر دقیق

۵.۳ یافتن خواسته های مسئله قسمت دوم

برای یافتن مقادیر متغیر های مفصلی جدول ۲ کافی این مقادیر را در تابع سینماتیک معکوس جایگذاری کنیم. در این صورت شکل ۱۷ حاصل می شود. که مقادیر منفی برای متغیر مفصل سه (d) مورد قبول نیست. چرا که همانطور در بالا توضیح داده شد در آن صورت لینک سه با مفصل دو تداخل خواهند داشت و هم جنین دو حالت باقی مانده دقیقاً زوایای مکمل دارند. پس هر دو از نظر ریاضی و فیزیکی قابل قبول هستند. از نظر من با توجه به موقعیت فعلی ربات آن حرکتی که کمترین میزان جا به جایی را داشته باشد بهتر است زیرا هم سریع تر به آن موقعیت می رسد و هم نیروی کمتری مصرف می کند. بنابراین جواب ها با زاویه دیگر در شکل ۱۷ در جدول ۶ هم مورد قبول هستند.

θ_1(deg)	4.9907	38.9976	-44.0009
θ_2(deg)	-10.0018	23.9994	66.9990
$d(mm)$	99.9790	290.01	350.0285
X(mm)	294.3	347.9	154.6
Y(mm)	25.7	281.7	-149.3
Z(mm)	297.9	549.3	856.3

جدول ۶: دیتاهای داده شده برای یافتن متغیر های مفصلی به کمک موقعیت نهایی end-effector

```
InvKin(294.3,  25.7,  297.9)
InvKin(347.9,281.7,549.3)
InvKin(154.6,-149.3,856.3)
```

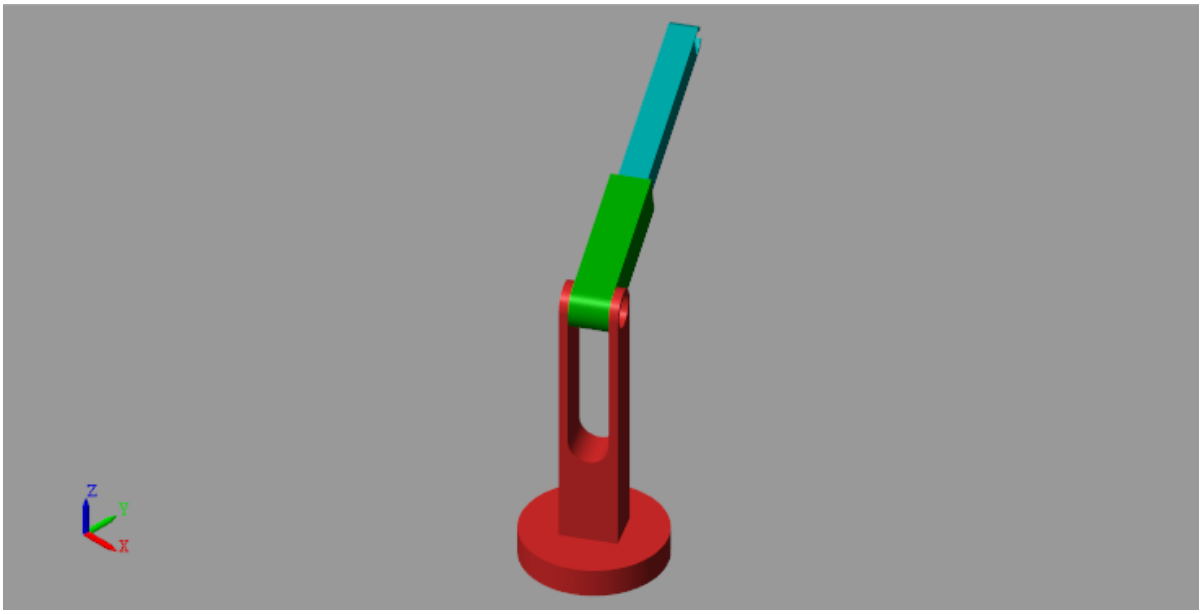
```
ans =

    4.9907  -10.0018   99.9790
 -175.0093   10.0018 -499.9790
 -175.0093 -169.9982   99.9790
    4.9907  169.9982 -499.9790


ans =

 -141.0024  -23.9994 -690.0100
   38.9976   23.9994  290.0100
   38.9976 -156.0006 -690.0100
 -141.0024  156.0006  290.0100


ans =

  135.9991  -66.9990 -750.0285
  -44.0009   66.9990  350.0285
  -44.0009 -113.0010 -750.0285
  135.9991  113.0010  350.0285
```

Published with MATLAB® R2022a

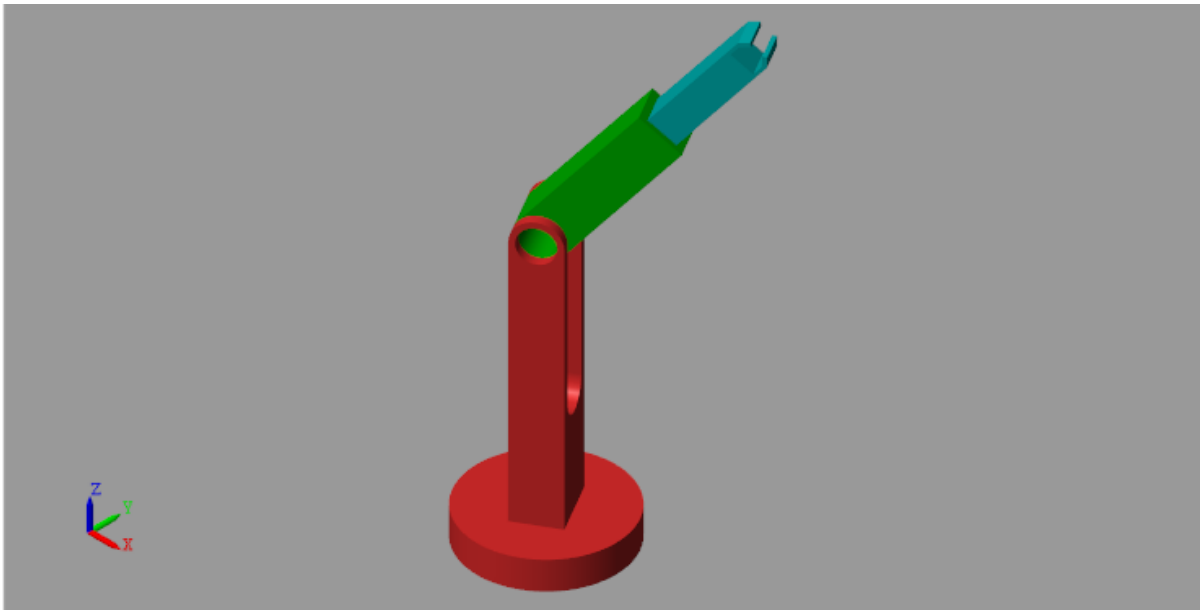
شکل ۱۷: خروجی تابع معکوس سینماتیک برای مقادیر جدول۲

٤ ضمایم

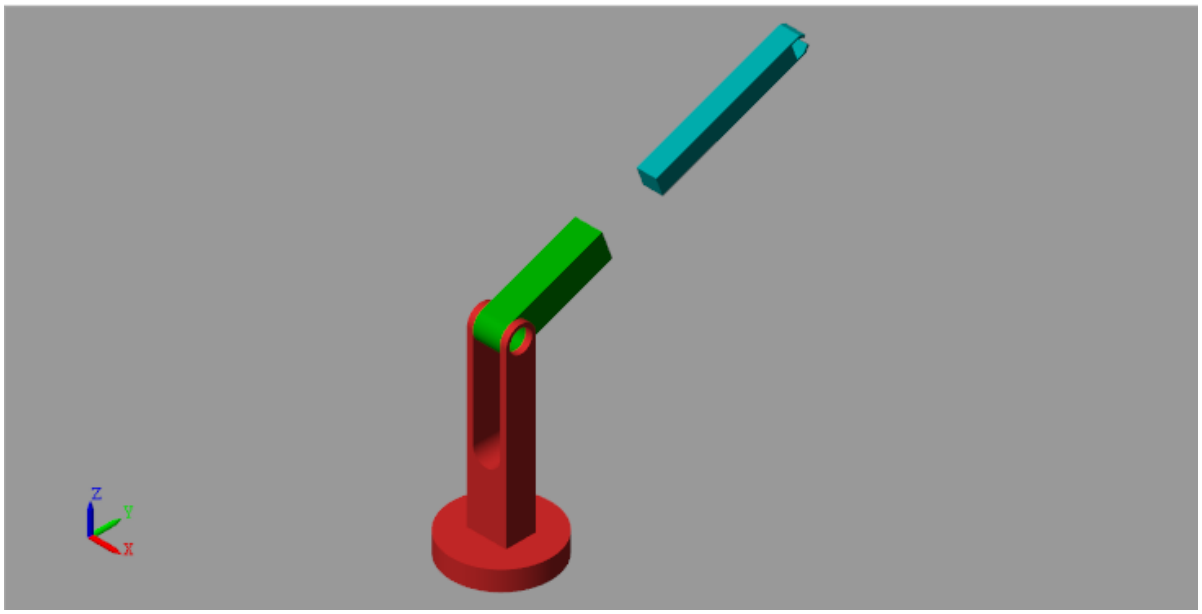
١.٤ تصاویر شبیه سازی شده ربات برای ۳ حالت جدول ۱



شکل ۱۸: تصویر شبیه سازی شده ربات برای ستون اول جدول۱



شکل ۱۹: تصویر شبیه سازی شده ربات برای ستون دوم جدول۱



شکل ۲۰: تصویر شبیه سازی شده ربات برای ستون سوم جدول۱