دانشگاه صنعتی شریف

دانشکده مهندسی مکانیک

عنوان:

تمرینات سری دوم

مدلسازی یک ربات RRP در محیط simulink و یافتن جدول DH آن

نگارش

محمدسعید صافی زاده

استاد راهنما

دکتر بهزادی پور

آبان ماه ۱۴۰۳

فهرست مطالب

۱ صورت سوالات

به کمک نرم افزار simulink یک ربات RRP را مدل کنید، فریم های مرجع آن را بر روی شکل ۱ نمایش دهید و به سوالات زیر پاسخ دهید.

۱.۱ سوال اول

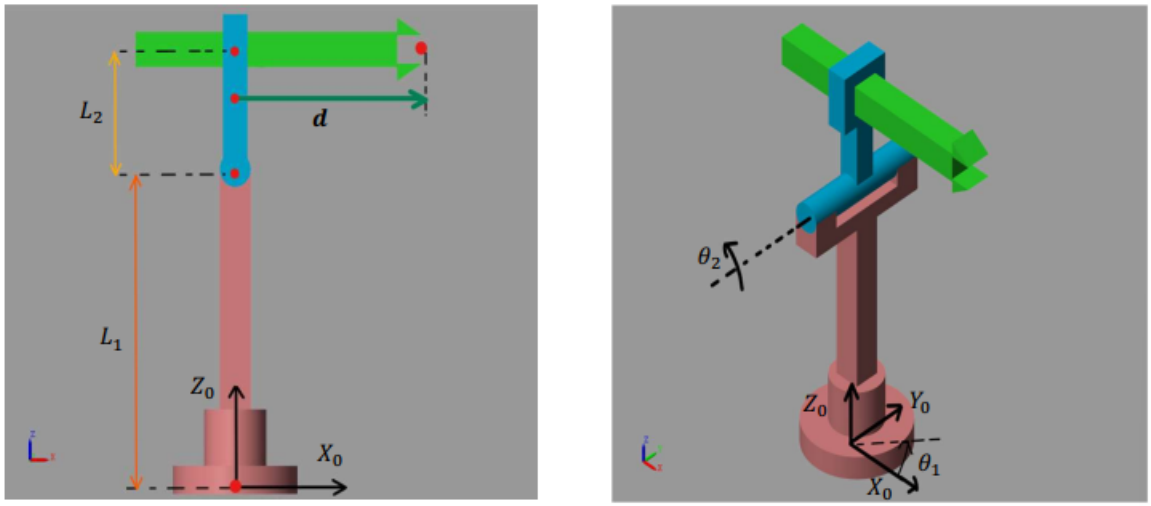
برای هر یک از مفصل های ربات مقادیر داده شده در جدول ۱ را اعمال کرده و موقعیت end-effector را بیابید.

θ_1(deg)	35	15	150	-18	40	-90
θ_2(deg)	40	-20	10	37	90	70
$d(mm)$	300	250	200	150	350	130

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

۲.۱ سوال دوم

با استفاده از قرارداد D-H [1] جدول را برای این ربات تکمیل کنید و به کمک تمرین سری اول مسئله ی forward kinematics را حل کنید و نهایتاً موقعیت end-effector را بیابید.جدول D-H را به همراه موقعیت نهایی end-effector گزارش و نتایج را با قسمت قبل مقایسه کنید. $(L_1 = 46cm, L_2 = 17cm)$



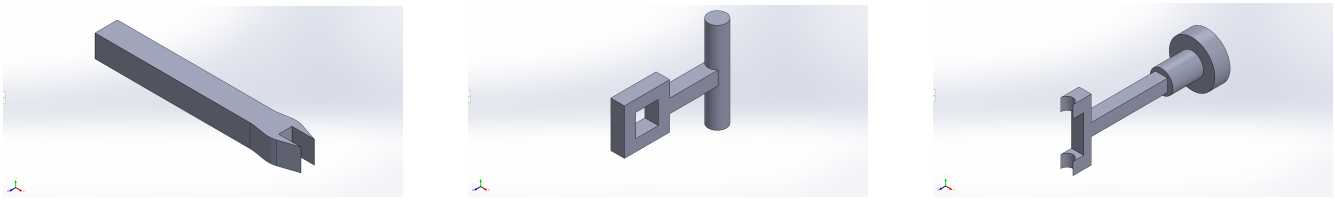
شکل ۱: در زمان نشان داده شده $\theta_1 = 0, \theta_2 = 0$ و $d = 300mm$ هستند .

[1] Denavit-Hartenberg

۲ پاسخ سوال اول

۱.۲ مدلسازی ربات در SolidWorks

ابتدا باید ربات را در نرم افزار SoliWorks مدل کرد. مطابق با شکل 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.



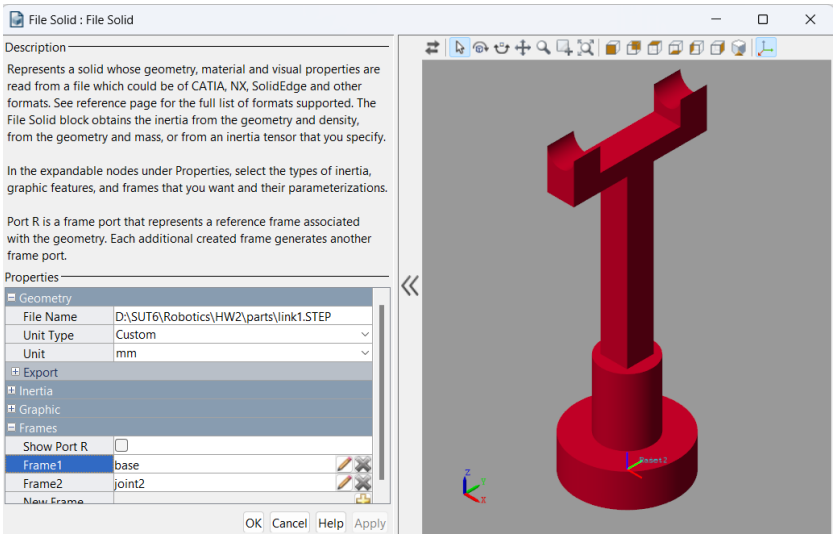
(آ) لینک ۱ (ب) لینک ۲ (ج) لینک ۳

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 اینکار 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 است.

۲.۲ مدلسازی در Simulink

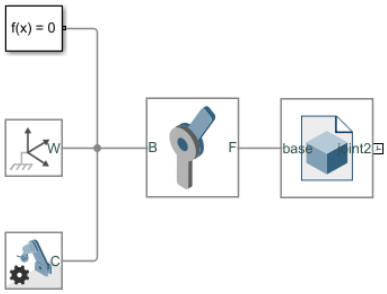
برای مدلسازی در سیمولینک باید یک محیط multibody ایجاد کرده و سپس به کمک بلوک های file solid و rigid transform به مدلسازی پرداخت. ابتدا لینک یک را به کمک بلوک file solid وارد کرده و مطابق با شکل ۳ یک فریم برای آن تعریف می کنیم تا به کمک مفصل revolute به زمین متصل شود.(تیک گزینه ی show port R را حتماً غیرفعال کنید.)



شکل ۳: نمایش فریم مفصل اول

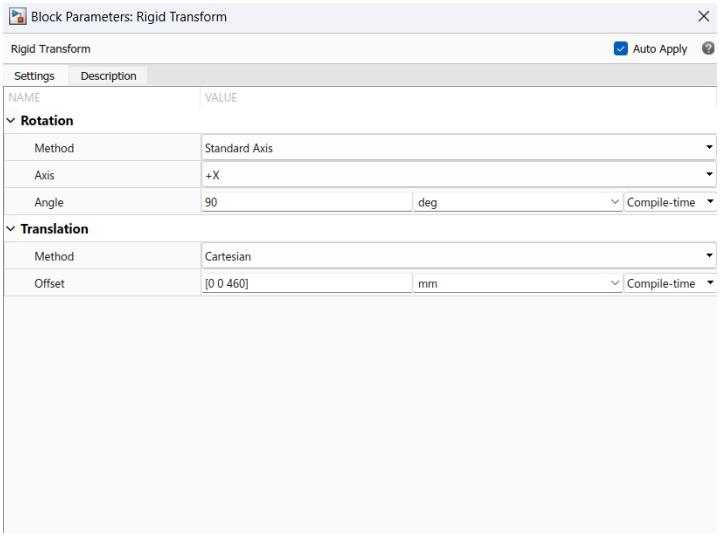
اکنون فریم تعیین شده را به عنوان یک follower [۲] به فریم world [۳] به وسیله ی یک revolute joint [۴] مطابق با شکل ۴ وصل می کنیم.

[۲] دنبال کننده
[۳] دستگاه مرجع
[۴] مفصل دورانی



شکل ۴: نحوه اتصال لینک ۱ به زمین

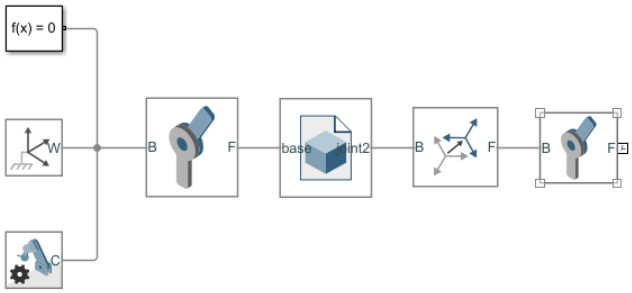
اکنون برای اتصال لینک دو به لینک یک نیاز به تعریف یک دستگاه مختصات جدید داریم. می دانیم که مفصل revolute حول محور z دوران می کنند بنابراین z مختصات جدید ما باید در راستای y کنونی باشد. برای این کار کافی است یک مختصات جدید منطبق بر فریم قبلی که در شکل ۳ داشتیم تعریف کرده و با استفاده از بلوک Rigid Transform این دستگاه را به نقطه ی دلخواهمان ببریم که در اینجا حرکت به اندازه 460 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 ۵ وارد می کنیم و این مختصات جدید را به یک مفصل دورانی متصل می کنیم.



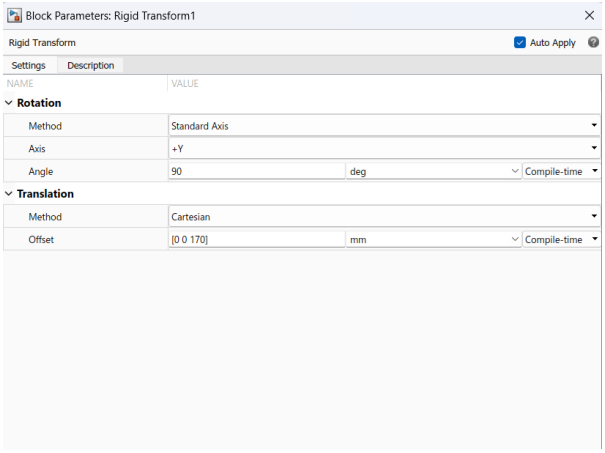
شکل ۵: تنظیمات rigid transform block برای ساختن مختصات مفصل دو

تا اینجا بلوک دیاگرام ما به شکل ۶ در آمده است.
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 ۷(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 joint3 می گذاریم و با Rigid Transform که تنظیمات آن در شکل ۷(ب) آمده است؛ آن را به محل مناسب برای اتصال مفصل سوم منتقل می کنیم.
توجه داشته باشید که مفصل prismatic[۵] 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 ۷(ب) به چشم می خورند.
حالا مفصل انتقالی را به این فریم متصل می کنیم و بلوک دیاگرام ما به صورت شکل ۸ در خواهد آمد.

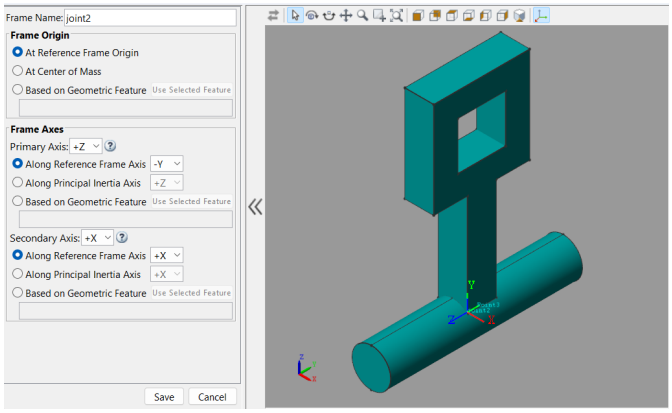
[۵] انتقالی



شکل ۶: بلوک دیاگرام تا این مرحله

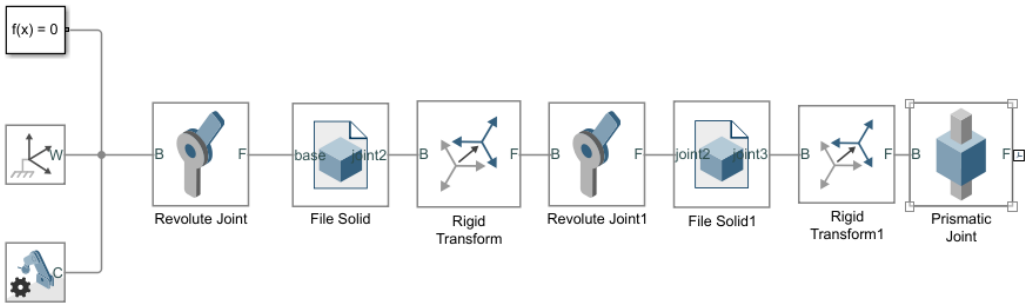


(ب) تنظیمات rigid transform block برای ساختن مختصات مفصل سه



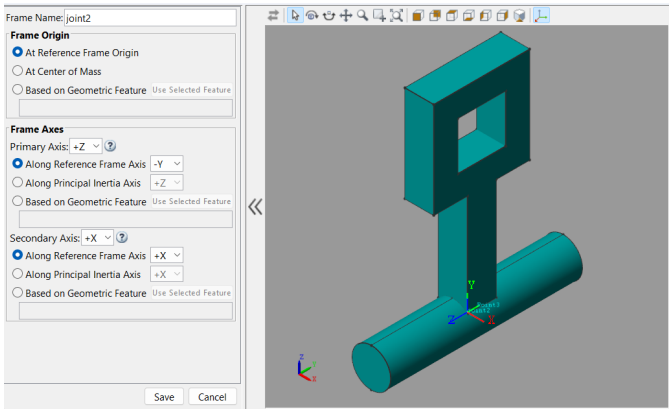
(آ) نمایش فریم لینک دوم

شکل ۷: نمایش فریم لینک دوم و تنظیمات Rigid Trnsform

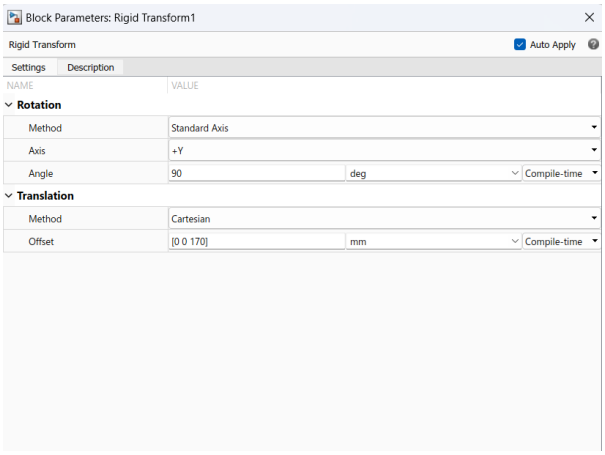


شکل ۸: بلوک دیاگرام تا این مرحله

در این مرحله لینک سوم را وارد می کنیم و فریم خودش را به عنوان فریم جدید تعریف می کنیم(شکل ۹(آ)) و مطابق با تمام توضیحات قسمت قبل این فریم را با توجه به تظیمات شکل ۹(ب) تعیین می کنیم.
در ادامه تنها تفاوت این خواهد بود که برای انجام این انتقال خود لینک سه باید base باشد و یا به بیانی دیگر این انتقال نسبت به لینک سوم انجام



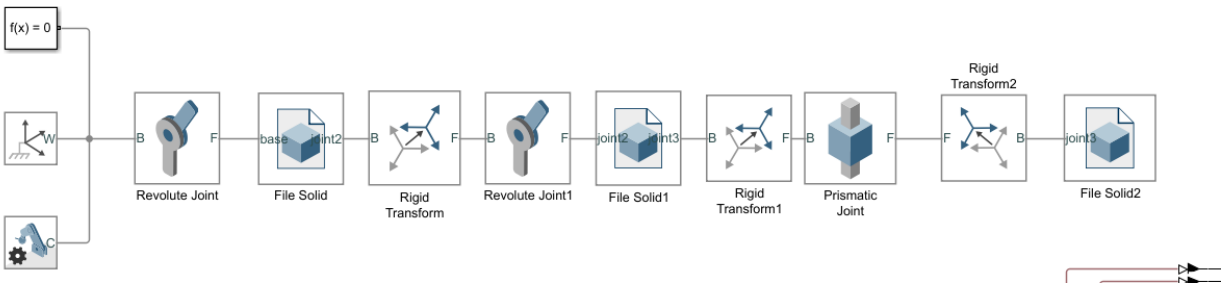
(آ) نمایش فریم لینک سوم



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

شکل ۹: نمایش فریم لینک سوم و تنظیمات Rigid Trnsform

شود. بلوک دیاگرام ما تا این مرحله مطابق شکل ۱۰ می شود.



شکل ۱۰: 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. با توجه به اینکه ورودی های مفاصل دورانی باید به رادیان باشند، درایه های زاویه را به رادیان تبدیل می کنیم.

```
% first column is the time at whih the variables are read
% second column is theta 1
```

```
% third column is theta 2
% forth column is theta d
jointData = [0, 35*pi/180, 40*pi/180, 0.3;
 2, 15*pi/180, -20*pi/180, 0.25;
 4, 150*pi/180, 10*pi/180, 0.2;
    6, -18*pi/180, 37*pi/180, 0.15;
 8, 40*pi/180, 90*pi/180, 0.35;
 10, -90*pi/180, 70*pi/180, 0.13; ];
```

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 دبل 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.

۳.۳.۲ بلوک transform sensor

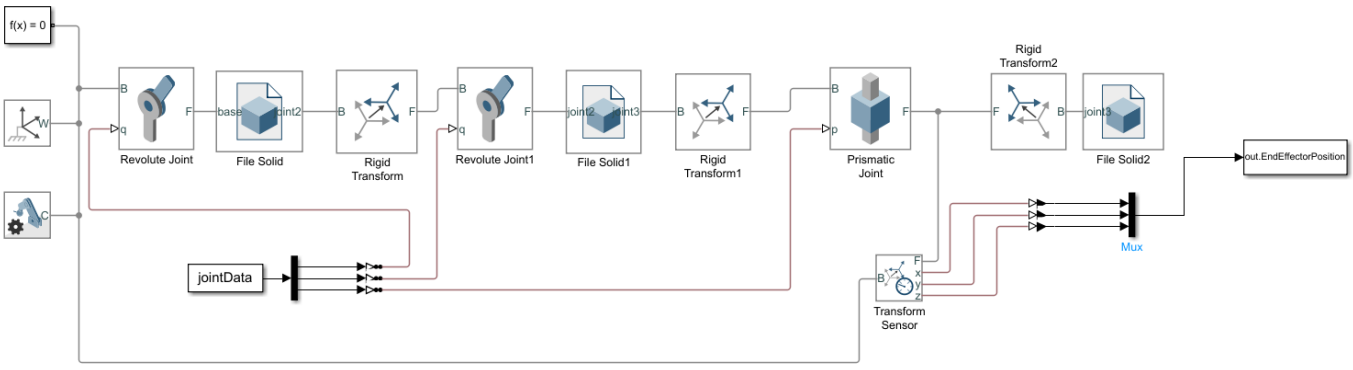
از این بلوک برای یافتن موقعیت end-effector استفاده می کنیم. با دبل کلیک بر روی آن و تیک زدن گزینه های X, Y, Z از منوی Translation و دادن follower و Base مناسب موقعیت end-effector را خواهیم یافت.

٤.۳.۲ 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.

۵.۳.۲ 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 ۱۱ در خواهد آمد.



شکل ۱۱: بلوک دیاگرام تا مرحله نهایی

بعد از اجرای شبیه سازی در سیمولینک داده های خروجی به صورت شکل ۱۲ خواهند بود. ستون اول موقعیت X ، ستون دوم موقعیت Y و ستون سوم موقعیت Z در end-effector است که همگی به mm هستند.

	X	Y	Z
1	98.74012811	69.139	783.06
2	283.0805827	75.851	534.24
3	-145.008473	83.721	662.15
4	16.63089737	-5.4037	686.04
5	-130.22758	-109.27	810
6	2.37E-05	115.29	640.3

شکل ۱۲: 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.

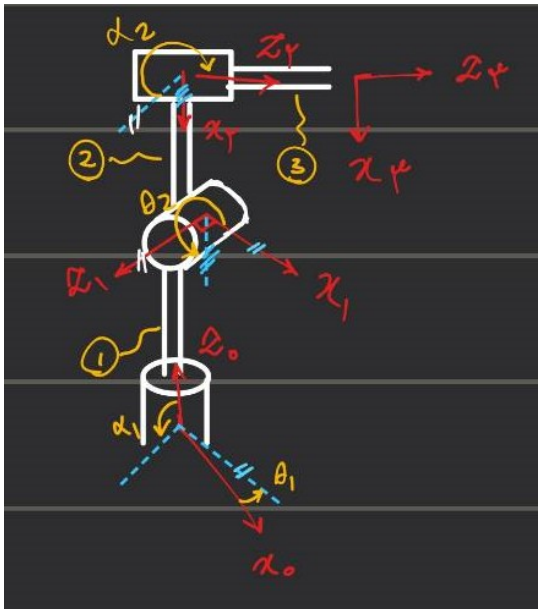
۳ پاسخ سوال دوم

۱.۳ تعریف دستگاه ها و زوایا

در شکل ۱۳ ، 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_2متنافر هستند بنابراین X_2 به شیوه ای که عمود مشترک این دو باشد 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.
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 محورZ_1 تبدیل می کند.
- ◇ زاویه θ_2 زاویه است که محور X_1 را با محور X_2 موازی می کند.
- ◇ $-l_2$ اندازه ای است که باید در راستای X_1 طی کنیم تا به دستگاه دو برسیم.
- ◇ α_2 زاویه ای است که محور Z_1 را به محورZ_2 تبدیل می کند.
- ◇ d اندازه ای است که باید در راستای Z_2 طی کنیم تا به دستگاه سه برسیم.

تمامی این متغیر های در جدول ۲ به صورت خلاصه آورده شده اند.



شکل ۱۳: نمایش دستگاه های تعریف شده برای یافتن جدول DH

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

جدول ۲: $\star$ 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 ۲ بدست می آید. بنابراین داریم:

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

$$H_2^1 = R_{z,\theta_2} T_{x,l_2} R_{x,\frac{-\pi}{2}}$$

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

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

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

اکنون این ماتریس را به کمک تمرین سری اول در متلب وارد می کنیم. فقط باید به یک نکته توجه کرد که θ_2 یک اندازه ی اولیه به اندازه ۲۷۰ درجه یا $\frac{3\pi}{2}$ دارد که این را در کد باید لحاظ کنیم.(تا در موقعیت اولیه خواسته شده سوال قرار بگیرد.)
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، کد ماتریس همگن دستگاه سه به صفر به صورت زیر خواهد شد:

```
%Variables
theta1 = 35;
```

```
theta2 = 270 + 40;
l1 = 460;
l2 = 170;
d = 300;

H1_0 = Rot('z',theta1) * Trans('z', l1) * Rot('x', 90);
H2_1 = Rot('z', theta2) * Trans('x',-1*l2) * Rot('x',-90);
H3_2 = Trans('z', d);
H3_0 = H1_0 * H2_1 * H3_2;
disp(H3_0*[0;0;0;1]);
```

که در این کد مقادیر مفصل ها مطابق با ردیف اول جدول داده شده است. اکنون می توانیم برای دریافت خروجی هر یک از متغیر های برای مفاصل یک حلقه ی for تعریف کنیم . اعداد را جایگذاری کنیم. سه سطر اول ستون آخر ماتریس همگن موقعیت end-effector را به ما می دهند یا می توانیم بردار همگن $P_3^3 = [0001]^T$ را در H_3^0 ضرب کنیم تا مقدار P_3^0 را بیابیم. در این جا راه حل دوم انتخاب شده است. کد مربوط به خواندن دیتا های خروجی در زیر آمده است:

```
jointData = [ 35, 40, 300;
  15, -20, 250;
  150, 10, 200;
  -18, 37, 150;
  40, 90, 350;
 -90, 70, 130;
];

l1 = 460;
l2 = 170;

z = zeros(6,4);

for i=1:1:length(jointData)
theta1 = jointData(i,1);
theta2 = 270 + jointData(i,2);
d = jointData(i,3);
H1_0 = Rot('z',theta1) * Trans('z', l1) * Rot('x', 90);
H2_1 = Rot('z', theta2) * Trans('x',-1*l2) * Rot('x',-90);
H3_2 = Trans('z', d);
H3_0 = H1_0 * H2_1 * H3_2;
z(i,:) = H3_0*[0;0;0;1];
end
```

در کد بالا ابتدا متغیر ها را درون یک ماتریس تعریف کرده و سپس دو مقدار ثابت l_1 و l_2 را می دهیم. برای خروجی نیز در خط بعد یک ماتریس 6x4 صفر تعریف می کنیم. در هر لوپ دیتا های هر سطر را می گیریم، جایگزین می کنیم و نهایتاً در هر سطر ماترس z وارد می کنیم. ماتریس خروجی به صورت شکل ۱۴ خواهد شد.

۳.۳ مقایسه موقعیت نهایی End-Effector در هر دو حالت

در هر دو روش جواب های نهایی یکسان شده اند ولی در قسمت ششم مقدار X در سیمولینک صفر مطلق نشده است. دلیل این امر می تواند محدودیت دقت عددی سیمولینک در شبیه سازی باشد. با توجه به اینکه محاسبات از نوع floating-point هستند، گاهی حافظه رم برای

z

6x4 double

	1	2	3	4	5	6
1	98.7401	69.1386	783.0638	1		
2	283.0806	75.8512	534.2427	1		
3	-145.0085	83.7207	662.1470	1		
4	16.6309	-5.4037	686.0403	1		
5	-130.2276	-109.2739	810	1		
6	0	115.2851	640.3035	1		
7						
8						

شکل ۱۴: موقعیت end-effector برای هر دیتای جدول – ستون اول موقعیت X، ستون دوم Y و ستون سوم موقعیت Z می باشد.

	X	Y	Z
1	98.74012811	69.139	783.06
2	283.0805827	75.851	534.24
3	-145.008473	83.721	662.15
4	16.63089737	-5.4037	686.04
5	-130.22758	-109.27	810
6	2.37E-05	115.29	640.3

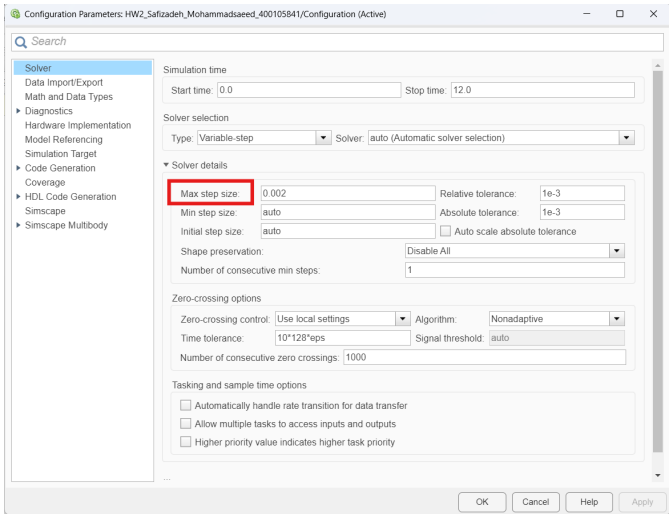
(آ) موقعیت گزارش شده در Simulink

DH	X	Y	Z
1	98.74	69.139	783.06
2	283.08	75.851	534.24
3	-145.01	83.721	662.15
4	16.631	-5.4037	686.04
5	-130.23	-109.27	810
6	0	115.29	640.3

(ب) موقعیت گزارش شده با DH

شکل ۱۵: موقعیت نهایی عملگر گزارش شده در نرم افزار simulink و بر اساس الگوریتم DH

نگهداری تعداد ارقام اعشار کافی نیست که خود منجر به خطا می شود. هم چنین با توجه به حل عددی متلب دقت solver نیز تاثیر زیادی در ایجاد خطا دارد. برای رفع این مورد می توانیم دقت و یا نوع solver را تغییر دهیم ولی از آن جایی که مقدار ما نسبت به بقیه متغیر ها به اندازه ای کوچک است که می توان از آن صرف نظر کرد؛ بنابراین راه حل پیشنهادی، ایجاد یک threshold است که مقادیر کوچک تر از 10^{-4} را صفر کند. همینطور در بخش solver نیز max step size را از روی 0.24 به 0.02 تغییر دادم که افزایش دقت solver را می توانید در شکل ۱۶ مشاهده کنید. تشابه پاسخ هر دو روش نشان دهنده دقت بالای شبیه سازی و هم چنین کارآمد بود روش D-H است. بنابراین هر دو روش در حوزه Forward Kinematics قابل اطمینان هستند.



(آ) تنظیمات مربوط به افزایش دقت solver

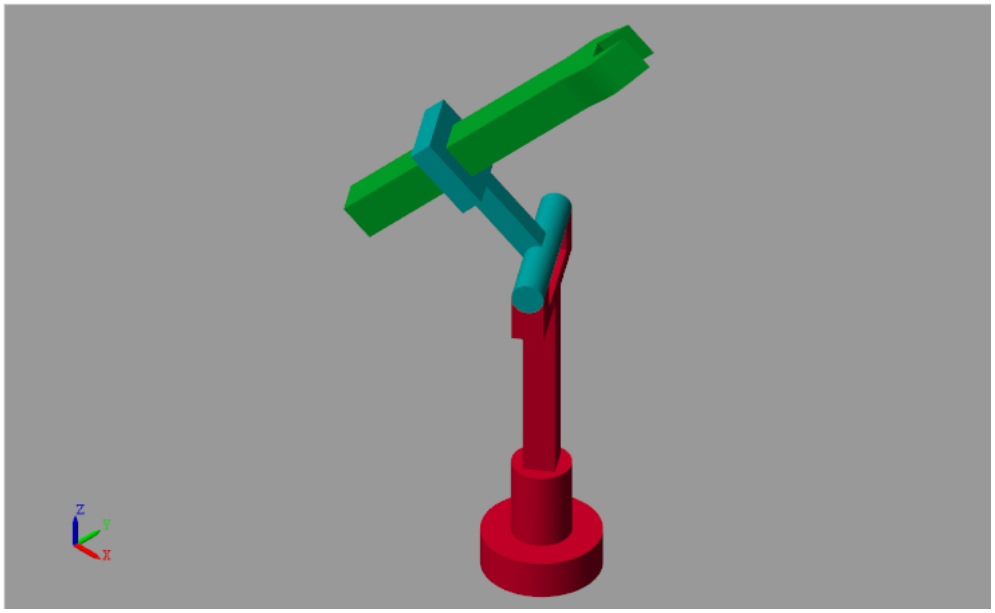
	X	Y	Z
1	98.7401	69.139	783.06
2	283.081	75.851	534.24
3	-145.01	83.721	662.15
4	16.6309	-5.4037	686.04
5	-130.23	-109.27	810
6	2.98E-12	115.29	640.3

(ب) پاسخ نهایی بعد از افزایش دقت

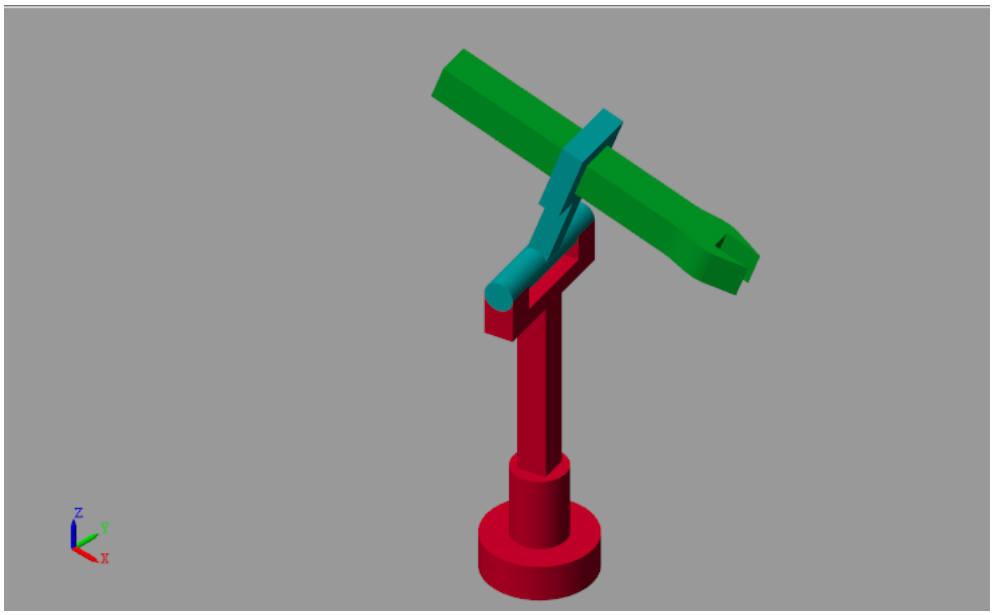
شکل ۱۶: تاثیر افزایش دقت بر پاسخ نهایی

٤ ضمایم

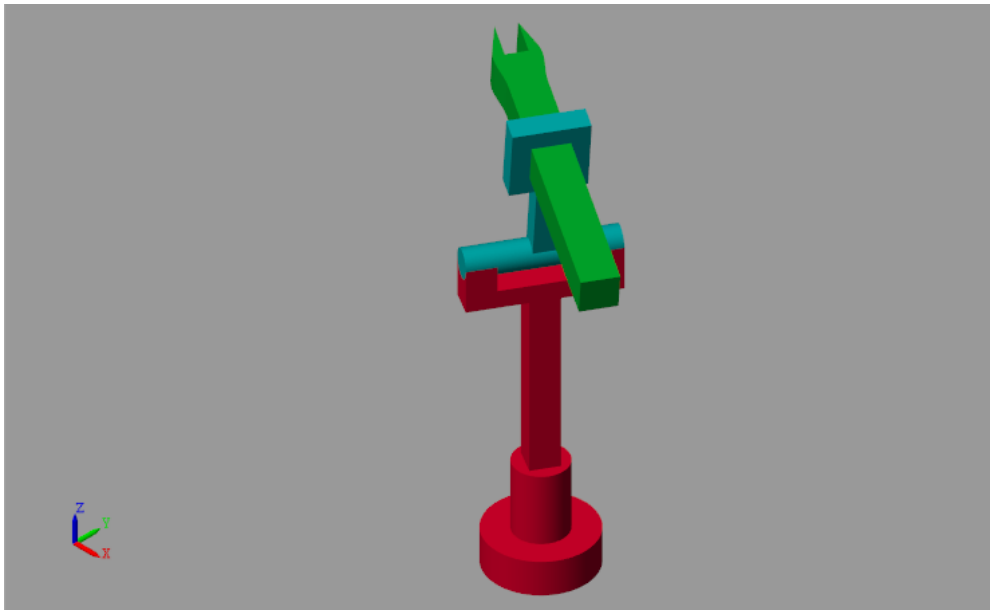
۱.٤ تصاویر شبیه سازی شده ربات برای ٦ حالت جدول ۱



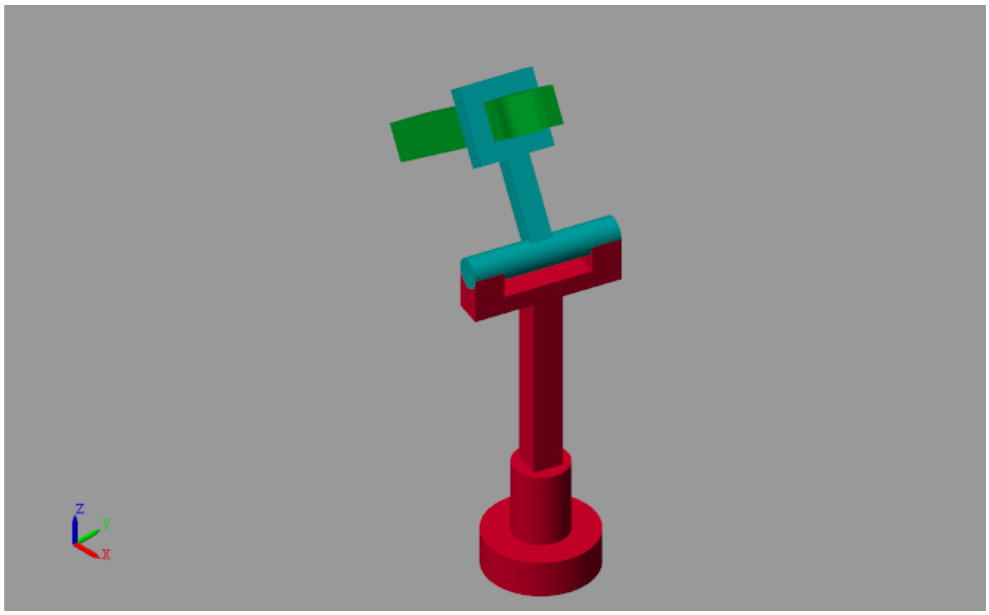
شکل ۱۷: تصویر شبیه سازی شده زبات برای ستون اول جدول۱



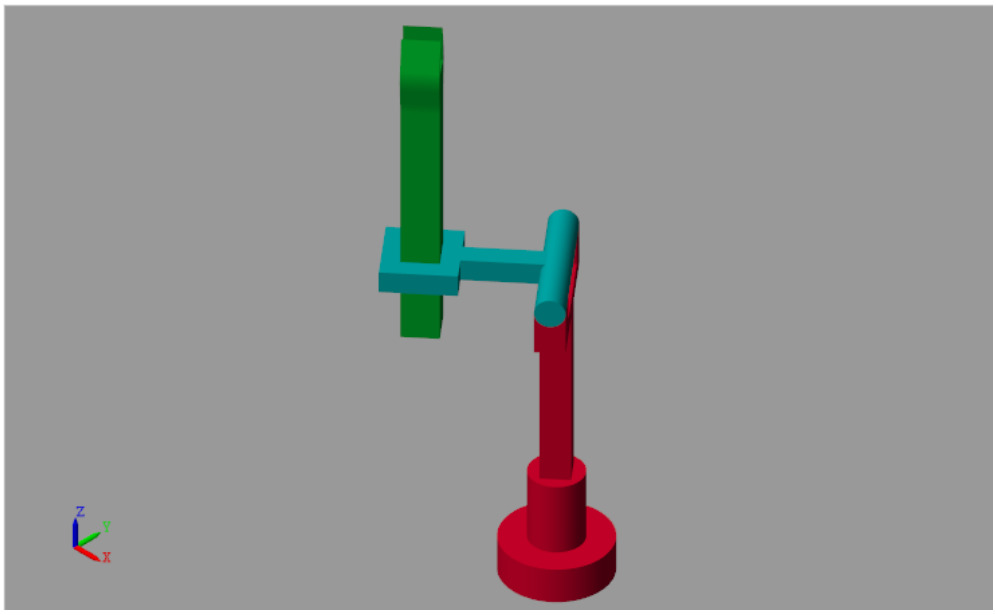
شکل ۱۸: تصویر شبیه سازی شده زبات برای ستون دوم جدول۱



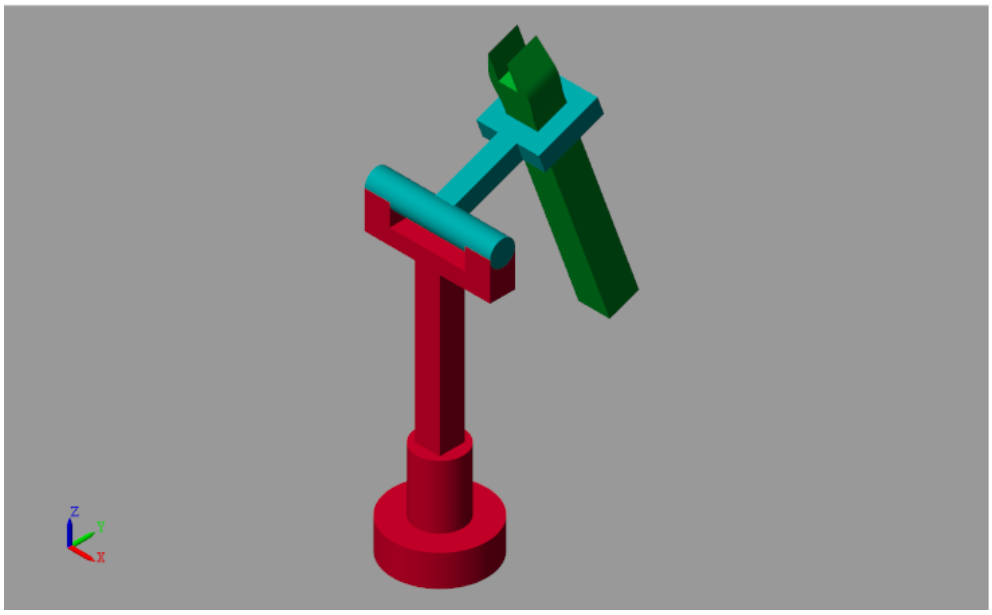
شکل ۱۹: تصویر شبیه سازی شده زبات برای ستون سوم جدول۱



شکل ۲۰: تصویر شبیه سازی شده زبات برای ستون چهارم جدول۱



شکل ۲۱: تصویر شبیه سازی شده زبات برای ستون پنجم جدول۱



شکل ۲۲: تصویر شبیه سازی شده زبات برای ستون ششم جدول۱