Primjena robotic system toolboxa za simulaciju kretanja robota

Tadić, Tomislav

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Tomislav Tadić

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Doc. dr. sc. Petar Ćurković, dipl. ing. Tomislav Tadić

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu. Zahvaljujem mentoru doc. dr. sc. Petru Ćurkoviću, dipl. ing. Na pruženom znanju i pomoći. Također zahvaljujem svojim roditeljima na pruženoj potpori, savjetima i osloncu, kao i svim kolegama i prijateljima koji su bili uz mene tokom studija i koji su mi na bilo koji način pomogli da uspješno položim preddiplomski studij fakulteta strojarstva i brodogradnje.

Tomislav Tadić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

ZAVRŠNI ZADATAK

Student: **Tomislav Tadić** Mat. br.: 0035212200

Naslov rada na hrvatskom jeziku: **Primjena robotic system toolboxa za simulaciju kretanja robota**

Naslov rada na engleskom jeziku: **Robotic system toolbox in robot motion simulation**

Opis zadatka:

Robotic system toolbox (RST) korisničko je sučelje koje je razvijeno s ciljem izvođenja realnih simulacija robotskih sustava različitih konfiguracija. Moguće je implementirati čitav niz senzora (npr. vizijski, kapacitivni, induktivni), kao i simulirati rad industrijskih mrežnih protokola za komunikaciju različitih komponenti složenih robotskih sustava. Također je moguće povezivanje robota s cijelim nizom upravljačkih algoritama (za planiranje trajektorije, kontrolu kolizije, kinematsku i dinamičku analizu robota i slično).

U radu je potrebno napraviti sljedeće:

- upoznati se s Robotic system toolbox okružjem
- ispitati način oblikovanja fizičkog modela robota i njegovog učitavanja u RST
- upoznati se s načinom generiranja trajektorija
- osigurati slijeđenje trajektorije u Point to Point (PtP) modu i prikazati kretanje robota koji slijedi generiranu trajektoriju

U radu je potrebno navesti literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
30. studenoga 2020.

Datum predaje rada:
1. rok: 18 veljače 2021.
2. rok (izvanredni): 5. srpnja 2021.
3. rok: 23. rujna 2021.

Predviđeni datumi obrane:
1. rok: 22.2. – 26.2.2021.
2. rok (izvanredni): 9.7.2021.
3. rok: 27.9. – 1.10.2021.

Zadatak zadao:

Doc. dr. sc. Petar Ćurković

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

Sadržaj

Popis slika

Popis Tablica

SAŽETAK

U ovom radu dan je opis Robotics System Toolboxa (RST), koji je dio programskog paketa Matlab. Također je dan opis operativnog sustava robota (ROS /Robot Operating System) te ROS mreže i njenih sastavnih dijelova. Osim toga, objašnjen je i Simscape Multibody Toolbox, te je priloženo nekoliko kodova u Matlab-u koji prikazuju različite mogućnosti primjene Robotics System Toolboxa i gore navedenih programskih paketa. Za neke primjere korišten je i Robotics Toolbox for Matlab od Peter-a Corke-a, koji sadrži neke funkcije kojih nema u RST-u. Uz navedene programske pakete, korišteni su i CAD program SolidWorks te Simulink koji je dio programskog paketa Matlab.

Ključne riječi: Robotics System Toolbox, ROS, Matlab, Simscape Multibody Toolbox, Robotics Toolbox for Matlab, SolidWorks, .xml, .urdf, Rigid Body Tree, Simulink

SUMMARY

This thesis gives a description of Robotics System Toolbox (RST), which is a part of Matlab program package. Robot Operating System (ROS) is also described, aswell as ROS network and its components. Besides that, Simscape Multibody Toolbox is described and several Matlab codes, which present different capabilities of Robotics System Toolbox and upper mentioned program packages, are provided. Peter Corke's Robotics Toolbox for Matlab, which provides some functionallities that RST doesen't have, is used for some examples. CAD program package SolidWorks and Simulink program package are used aswell.

Key words: Robotics System Toolbox, ROS, Matlab, Simscape Multibody Toolbox, Robotics Toolbox for Matlab, SolidWorks, .xml, .urdf, Rigid Body Tree, Simulink

1. UVOD

Matlab je računalna platforma za programiranje i numeriku koju koriste milijuni inženjera i znanstvenika za analiziranje podataka, razvijanje algoritama i stvaranje modela. Matlab ima veliki broj dodatnih programskih paketa (toolbox-a) koji mu proširuju mogućnosti. Robotics System Toolbox je programski paket koji se može instalirati kao dio Matlaba i pomoću njega se mogu modelirati složeni robotski sustavi, s robotskim manipulatorima, mobilnim robotima, humanoidnim robotima, senzorima i aktuatorima. RST je stoga pogodan za razne implementacije na području robotike. RST omogućuje povezivanje Matlab-a i Simulink-a sa ROS mrežom preko koje se može ostvariti veza sa fizičkim robotima ili simulatorima kao što je npr. Gazebo. RST pruža izbor od preko 40 modela robota ('abbIrb120', 'abbIrb120T', 'abbIrb1600', 'abbYuMi', 'atlas', 'fanucLRMate200ib', 'fanucM16ib', 'frankaEmikaPanda', 'kinovaGen3', 'kinovaJacoJ2N6S200', 'kinovaJacoJ2N6S300', 'kinovaJacoJ2N7S300', 'kinovaJacoJ2S6S300',
'kinovaJacoJ2S7S300', 'kinovaJacoTwoArmExample', 'kinovaMicoM1N4S200', 'kinovaMicoM1N6S200', 'kinovaMicoM1N6S300',
'kukaIiwa7', 'kukaIiwa14', 'quanserQArm', 'rethinkBaxter', 'rethinkSawyer', 'robotisOP2', 'robotisOpenManipulator',
'universalUR10', 'universalUR3', 'universalUR5', 'valkyrie', 'yaskawaMotomanMH5', 'kinovaMovo', 'amrPioneerLX',
'amrPioneer3AT', 'amrPioneer3DX', 'clearpathHusky', 'clearpathJackal', 'clearpathTurtleBot2', 'robotisTurtleBot3Burger',
'robotisTurtleBot3Waffle', 'robotisTurtleBot3WaffleForOpenManipulator', 'robotisTurtleBot3WafflePi',
'robotisTurtleBot3WafflePiForOpenManipulator', 'quanserQBot2e', 'quanserQCar', 'willowgaragePR2'). Također pruža mogućnost oblikovanja okruženja robota, a okruženje se može oblikovati i u CAD programskim paketima i potom uvesti u RST okruženje preko .urdf datoteka. Robotski sustavi se mogu oblikovati i pomoću Simscape Multibody toolbox-a, gdje se upravljanje robotima odvija pomoću Simulink-a. U ovom radu dan je pregled nekih funkcionalnosti spomenutih programa, među kojima je i generiranje te praćenje trajektorije robotskim manipulatorima.

2. ROBOTICS SYSTEM TOOLBOX

Robotics System Toolbox (RST) pruža mogućnost povezivanja algoritama i hardvera za razvoj autonomnih robotskih aplikacija za zračna i cestovna vozila, robotske manipulatore i humanoidne robote.

Toolbox sadrži algoritme za: planiranje i slijeđenje trajektorija za robote sa diferencijalnim pogonom, podudaranje scan-ova, izbjegavanje prepreka i procjenu stanja robota.

Za robotske manipulatore toolbox sadrži algoritme za inverznu kinematiku, kinematska ograničenja i dinamiku koristeći reprezentaciju stabla krutih tijela ("Rigid body tree representation").

System Toolbox pruža sučelje između Matlaba, Simulinka i ROS-a (Robot Operating System / Operativni sustav robota), što omogućava ispitivanje i verificiranje aplikacija na robotima koji podržavaju ROS i na robotskim simulatorima kao što je npr. Gazebo.

Uključuje primjere koji pokazuju kako raditi s virtualnim robotima u Gazebu i sa stvarnim robotima koji podržavaju ROS.

RST podržava generiranje C++ koda, što omogućuje generiranje ROS čvora iz Simulink modela i automatsko slanje koda na ROS mrežu.

Potpora za vanjski način rada Simulinka dozvoljava korisniku pregled signala i promjenu parametara za vrijeme rada pokrenutog Simulink modela. [1]

2.1 Transformacije koordinatnih sustava u robotici

U robotskim aplikacijama se može koristiti puno različitih koordinatnih sustava za definiranje položaja robota, senzora i drugih objekata.

Općenito, položaj objekta u 3D prostoru se može specificirati pomoću vrijednosti pozicije i orijentacije. Postoji više mogućih reprezentacija ovih vrijednosti, od kojih su neke specifične za određene aplikacije. Translacija i rotacija su alternativni pojmovi za poziciju i orijentaciju.

RST podržava reprezentacije koje su uobičajene u robotici te omogućuje pretvorbe između njih. Dakle, kada se ove reprezentacije primjene na točke u prostoru, moguća je transformacija koordinatnih sustava pomoću RST-a.

Uobičajene reprezentacije su:

Axis-Angle (axang), Euler-Angles (eul), Quaternion (quat), Rotation Matrix (rotm), Homogeneous Transformation (tform) i Translation Vector (trvec)

Axis-Angle (axang) – rotacija u 3D prostoru opisana skalarnom rotacijom oko fiksirane osi, definira se vektorom

Numerička reprezentacija: 1x3 jedinični vektor i skalarni kut ujedinjeni u 1x4 vektoru

Npr., rotacija od $\pi/2$ radijana oko y-osi definira se:

axang = [0 1 0 pi/2]

Euler-Angles (eul) – Eulerovi kutovi su tri kuta koja opisuju orijentaciju krutog tijela (rigid body). Svaki kut predstavlja skalarnu rotaciju oko određene osi koordinatnog sustava. RST podržava dva redoslijeda rotacija. "ZYZ" redoslijed je uobičajen za robotske aplikacije. Podržava se i "ZYX" redoslijed osi koji se također referira kao "Roll Pitch Yaw (rpy)" redoslijed. Važno je znati koji redoslijed osi koristimo za primjenu rotacija na točke i pretvorbu u druge reprezentacije.

Numerička reprezentacija: 1x3 vektor skalarnih kutova

Npr., rotacija od π radijana oko y-osi definira se:

eul = [0 pi 0]

Napomena: redoslijed osi nije pohranjen u transformaciji, pa je važno znati koji redoslijed osi treba primijeniti.

Quaternion (quat) – quaternion je vektor od 4 elementa sa skalarnom rotacijom i troelementnim vektorom. Prednost quaterniona je to što izbjegavaju probleme singulariteta koji su prisutni kod ostalih reprezentacija. Prvi element, "w", je skalar za normalizaciju vektora s ostale tri vrijednosti, [x y z], koji definira os rotacije.

Numerička reprezentacija: 1x4 vektor

Npr., rotacija od $\pi/2$ oko y-osi definira se:

quat = [0.7071 0 0.7071 0]

Rotation Matrix (rotm) – opisuje rotaciju u 3D prostoru. To je kvadratna, ortonormalna matrica s determinantom 1.

Numerička reprezentacija: 3x3 matrica

Npr., rotacija za kut α u stupnjevima oko x-osi definira se:

rotm =

$$\begin{matrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{matrix}$$

Homogeneous Transformation (tform) – matrica homogenih transformacija, kombinira translaciju i rotaciju u jednoj matrici

Numerička reprezentacija: 4x4 matrica

Npr., rotacija za kut α oko y -osi i translacija od 4 jedinice uzduž

y -osi definira se:

tform =

$$\begin{matrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 4 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Translation Vector (trvec) – reprezentira se u 3D Euklidskom prostoru s kartezijskim koordinatama. Uključuje samo translaciju koja se primjenjuje jednako na sve točke. Rotacija nije uključena.

Numerička reprezentacija: 1x3 vektor

Npr., translacija od 3 jedinice uzduž x-osi i 2.5 jedinice uzduž z-osi definira se:

trvec = [3 0 2.5]

Pretvorbene funkcije i transformacije

RST pruža pretvorbene funkcije za prethodno spomenute reprezentacije transformacija.

Međutim, nisu sve pretvorbe podržane. U **Tablici 2.1** ispod prikazane su podržane pretvorbe (u plavom).

Tablica 2.1 Tablica podržanih pretvorbi

Converting From \ Converting To	Axis-Angle (axang)	Euler Angles (eul)	Quaternion (quat)	Rotation Matrix (rotm)	Homogeneous Transformation (tform)	Translation Vector (trvec)
Axis-Angle (axang)						
Euler Angles (eul)						
Quaternion (quat)						
Rotation Matrix (rotm)						
Homogeneous Transformation (tform)						
Translation Vector (trvec)						

Imena svih pretvorbenih funkcija slijede standardni format. Slijede formu **alfa2beta**, gdje je alfa skraćenica za ono iz čega pretvaramo, a beta je skraćenica za ono u što pretvaramo.

Npr. pretvorba iz Euler angles u quaternion bi bila **eul2quat.**

Sve funkcije očekuju ispravne ulaze. Ako specificiramo nevažeće ulaze, izlazi će biti nedefinirani. Postoje i druge pretvorbene funkcije za pretvorbu između radijana i stupnjeva, kartezijskih i homogenih koordinata i za izračunavanje razlika omotanih kutova. [1]

3. ROS

Robot Operating System (ROS) je komunikacijsko sučelje koje omogućuje različitim dijelovima robotskog sustava da otkrivaju, šalju i primaju podatke.

To je " Open source" fleksibilni sustav za razvoj robotskih softvera.

Matlabova potpora za ROS je knjižnica funkcija koja omogućava razmjenu podataka sa fizičkim robotima koji podržavaju ROS ili s robotskim simulatorima kao što je Gazebo. [1]

3.1 ROS terminologija

ROS network sadrži različite dijelove robotskog sustava (npr. sučelje kamere) koji komuniciraju preko ROS-a. Mreža može biti raspodijeljena na nekoliko strojeva

ROS master koordinira različite dijelove ROS mreže. Identificira se pomoću Master URI (Uniform Resource Identifier / Jedinstveni identifikator resursa) koji specificira "hostname" ili IP adresu stroja na kojem je pokrenut master.

ROS node je entitet koji sadrži kolekciju povezanih ROS mogućnosti (kao što su "izdavači", "pretplatnici" i "usluge"). ROS mreža može imati puno ROS čvorova.

Izdavači ("publishers"), pretplatnici ("subscribers") i usluge ("services") su različite vrste ROS entiteta koji obrađuju podatke. Oni razmjenjuju podatke pomoću poruka.

Izdavač šalje poruke vezane za određenu temu ("topic"), a pretplatnici na toj temi primaju poruke od izdavača. Može biti više izdavača i pretplatnika povezanih s jednom temom.

Npr. Matlab, Simulink, senzori, aktuatori- to su sve čvorovi (nodes) u ROS mreži

Za pribavljanje podataka o nekoj ROS temi (topic) od strane robota ili simulatora, potrebno je stvoriti "pretplatnika" (funkcija "rossubscriber").

Ako pak želimo slati podatke robotu, moramo stvoriti "izdavača" (funkcija "rospublisher").

Dakle, ROS je skup alata, knjižnica i algoritama koji pomažu korisnicima pri razvoju robotskih softvera. To je fleksibilni sustav za programiranje robota i upravljanje robotskim platformama.

ROS je razvijen od strane "open-source" suradničke zajednice kako bi pomogao u razvoju svijeta robotike. U njemu su dostupne aplikacije za rad s hardverom, robotskim simulacijskim modelima, planiranjem trajektorija, lokalizacijama, mapiranjem, kao i mnogi drugi algoritmi.

RST omogućuje pristup funkcionalnostima ROS-a u Matlabu, korištenje Matlaba za komuniciranje sa ROS mrežom, interaktivno istraživanje mogućnosti robota i vizualiziranje podataka sa senzora. [1]

3.2 Povezivanje sa ROS mrežom u Matlabu

ROS mreža se sastoji od jednog ROS mastera i više ROS čvorova.

ROS master omogućava komunikaciju u ROS mreži praćenjem svih aktivnih ROS entiteta.

Svaki se čvor mora registrirati kod ROS mastera da bi mogao komunicirati s ostatkom mreže. Matlab može pokrenuti ROS master, a master može biti pokrenut i izvan Matlaba (npr. na drugom računalu). Svi se ROS čvorovi registriraju kod mastera i proglašavaju mrežnu adresu na kojoj se može doći do njih.

Pri radu s ROS-om, obično se slijede ovi koraci:

Povezivanje s ROS mrežom – kako bi se povezali s ROS mrežom, može se kreirati ROS master u Matlabu ili se povezati s postojećim ROS masterom. U oba slučaja Matlab će također kreirati i registrirati vlastiti RS čvor (tzv. Matlab "global node") kod mastera.

Funkcija "rosinit" upravlja ovim procesom.

Razmjena podataka – kada se poveže, Matlab razmjenjuje podatke s ostalim ROS čvorovima putem "izdavača", "pretplatnika" i "usluga".

Isključivanje s ROS mreže – pozivom funkcije "rosshutdown" Matlab se isključuje iz ROS mreže. [1]

3.3 Stvaranje ROS mastera u Matlabu

Kako bi stvorili ROS master u Matlabu, potrebno je pozvati funkciju "rosinit" bez ulaznih argumenata. Time će se također stvoriti i "global node" preko kojeg će Matlab komunicirati s drugim čvorovima unutar ROS mreže. Nakon pozva ove funkcije, ROS čvorovi koji se nalaze izvan Matlaba se mogu povezati sa ROS mrežom. Oni se na ROS master u Matlabu mogu spojiti korištenjem "hostname" ili IP adresu Matlabovog računala "domaćina".

ROS master i global node se isključuju naredbom "rosshutdown". [1]

3.4 Povezivanje na vanjski ROS master

Naredba "rosinit" se može koristiti i za povezivanje s vanjskim ROS masterom (koji se nalazi npr. u robotu ili virtualnom stroju).

Adresa mastera se može specificirati na dva načina: koristeći IP (Internet Protocol) adresu ili pomoću "hostname" računala koji pokreće master.

Nakon svakog poziva naredbe "rosinit" potrebno je pozvati naredbu "rosshutdown" prije poziva naredbe "rosinit" s drugačijom sintaksom.

Primjeri spajanja na vanjski ROS master:

rosinit ('192.168.1.1') – spajanje preko IP adrese računala na kojem je pokrenut ROS master

rosinit ('master_host') – spajanje preko "hostname" računala na kojem je pokrenut ROS master

Oba prethodna poziva pretpostavljaju da će master prihvatiti veze na portu 11311, koji je standardni port za ROS master.

U slučaju da se master pokreće na nekom drugom portu, željeni port se može specificirati kao drugi argument u naredbi "rosinit".

Npr.

rosinit('master_host', 12000) – povezivanje s ROS masterom na portu 12000

rosinit('http://192.168.1.1:12000') – povezivanje na ROS master preko URI mastera

Kao dio registracije u ROS masteru, Matlabov "global node" mora specificirati IP adresu ili "hostname" na kojem će ga ostali ROS čvorovi moći pronaći.

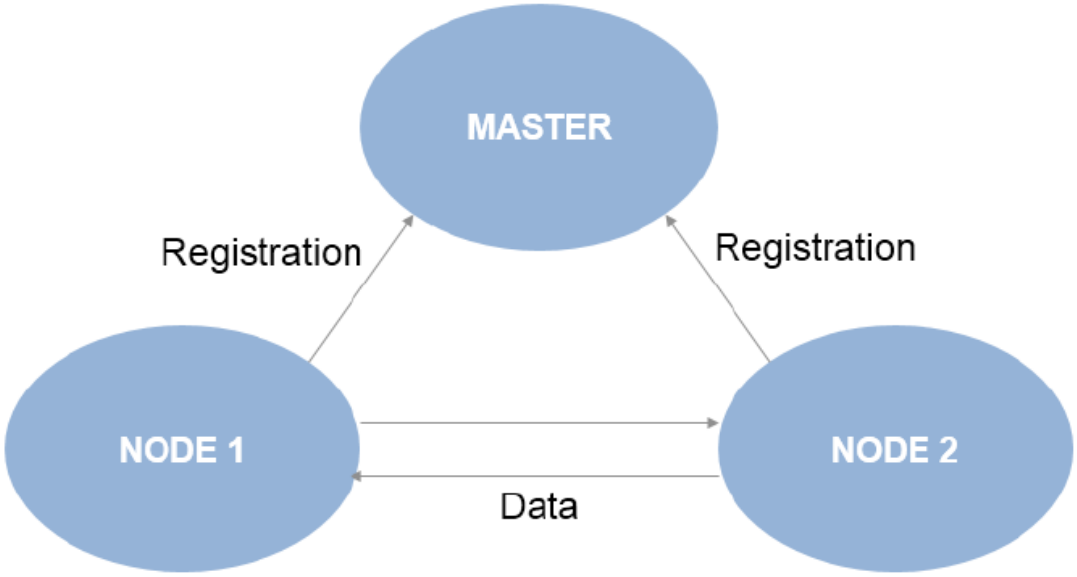
Svi ostali čvorovi registrirani u ROS mreži će preko te adrese slati podatke globalnom čvoru u Matlabu.

Npr. rosinit('master_host', 'Nodelist', '192.168.1.100') – definiranje adrese globalnog čvora u Matlabu kao "192.168.1.100". [1]

3.5 Verifikacija ROS mreže

Kako bi ROS mreža radila ispravno, potrebno je osigurati da svi čvorovi mogu komunicirati s masterom i jedni s drugima. Individualni čvorovi moraju komunicirati s masterom kako bi registrirali "pretplatnike" "izdavače" i "usluge". Čvorovi moraju moći komunicirati i jedni s drugima kako bi slali i primali podatke. Zbog ovakvog ustroja mreže može se dogoditi da se podaci mogu slati a ne mogu primati ili obrnuto ako ROS mreža nije ispravno uspostavljena. Svaki čvor će kontaktirati master kako bi "saznao" adrese ostalih čvorova u ROS mreži. Jednom kada čvorovi "saznaju" međusobne adrese, izmjena podataka može se uspostaviti bez uključivanja mastera. [1]

3.6 Izgled veza u ROS mreži



Slika 3.1 izgled veza u ROS mreži [1]

Svaki čvor registrira različite izdavače, pretplatnike i usluge na ROS master za slanje i primanje informacija među čvorovima. Iako su svi čvorovi u ROS mreži registrirani s masterom, podatci se razmjenjuju izravno između čvorova. Izdavači šalju podatke preko naziva tema (topic names), koje pretplatnici potom primaju preko mreže. Usluge koriste klijente (service clients) pomoću kojih zahtijevaju informacije od poslužitelja (service server).

Gornja slika [**Slika 3.1**] prikazuje raspored ROS mreže sa dva ROS čvora. Važno je da svi čvorovi imaju dvosmjernu povezljivost za dijeljenje podataka kroz mrežu. Prema tome, verifikacija ovih veza je važna prilikom postavljanja mreže. [1]

4. SIMSCAPE

Simscape omogućuje brzu izradu modela fizičkih sustava unutar Simulink okruženja. Pomoću Simscape-a možemo graditi modele fizičkih komponenti na temelju fizičkih veza koje izravno integriraju s blok dijagramima i ostalim paradigmama modeliranja. Modeliramo sustave kao što su električni motori, mostni ispravljači, hidraulički aktuatori i rashladni sustavi, na način da spajamo osnovne komponente u sheme. Simscape add-on proizvodi pružaju složenije komponente i sposobnosti analize.

Simscape pomaže pri razvoju upravljačkih sustava i testiranju performansi na razini sustava. Možemo stvarati vlastite modele komponenti koristeći Simscape jezik na bazi Matlab-a, koji omogućava autorstvo komponenata fizičkog modeliranja, domena i knjižnica (libraries) na bazi teksta. Moguće je parametrizirati modele pomoću Matlab-ovih varijabli i izraza, kao i konstruirati upravljačke sustave za naše fizičke sustave u Simulink-u. Za implementaciju modela u drugim simulacijskim okruženjima, uključujući hardware-in-the-loop (HIL) sustave, Simscape podržava generiranje C-koda. [2]

4.1 Simscape Multibody

Simscape Multibody pruža simulacijsko okruženje s više tijela za 3D mehaničke sustave kao što su roboti, ovjesi vozila, građevinska oprema i podvozja zrakoplova. Omogućuje modeliranje sustava s više tijela koristeći blokove koji predstavljaju tijela, zglobove, ograničenja, elemente sila i senzore. Simscape Multibody formulira i rješava jednadžbe gibanja kompletnog mehaničkog sustava. Moguće je uvesti kompletne CAD (Computer-Aided Design) sklopove, uključujući sve mase, inercije, zglobove, ograničenja i 3D geometrije u Simulink model. Automatski generirana 3D animacija pruža vizualizaciju dinamike sustava. Moguće je integrirati hidrauličke, električne, pneumatske i druge fizičke sustave u model koristeći komponente iz Simscape-ovog skupa proizvoda. Kao što je već spomenuto, implementacija je moguća i na drugim simulacijskim okruženjima preko generiranja C-koda. [2]

4.2 Uvoz modela (Model Import)

Multibody model se može uvesti u Simscape Multibody okruženje. Za to koristimo funkciju **smimport.** Mogu se uvesti CAD, URDF i Robotics System Toolbox modeli. Za stvaranje RST modela potrebna je odgovarajuća licenca. Funkcija **smimport** raščlanjuje model, izvlači potrebne podatke i rekonstruira sklop koristeći Simscape Multibody blokove koji predstavljaju tijela, ograničenja i zglobove sklopa.

URDF modeli moraju biti u **.urdf** datotekama, RST modeli u **rigidBodyTree** objektima, a **SimscapeMultibody** modeli moraju biti u prikladnom **XML** formatu. URDF datoteke otvaraju se pomoću naredbe **importrobot()**, XML datoteke pomoću naredbe **smimport()**, a rigidBodyTree (RBT) objekti pomoću naredbe **loadrobot()**. Sva tri načina uvoze modele koji imaju stablastu strukturu (RBT), ali su različiti formati u kojima je zapisana definicija tog stabla. Za informacije o RBT-u koristi se naredba **showdetails()**.

CAD modeli se mogu iz Autodesk-a, Inventor-a, Creo-a, Parametric-a ili SolidWorks-a izvesti u .XML file pomoću **Simscape Multibody Link** programskog paketa, koji nije standardan pri instalaciji Matlaba, ali se može naknadno instalirati, a potom ga je potrebno povezati sa željenim programom. Povezivanje sa npr. Solidworks-om obavlja se upisom naredbe **smlink_linksw** u Matlabu. [2]

Prikaz naredbe za povezivanje Simscape Multibody Toolboxa i SolidWorksa iz Matlaba

```
%Povezivanje SolidWorks-a sa Simscape Multibody Toolbox-om
smlink_linksw
```

4.3 Uvoz URDF modela

URDF model se uvozi u Simscape Multibody okruženje koristeći funkciju **smimport,** u koju kao glavni argument upisujemo ime URDF datoteke. Funkcija preko nastavka datoteke (.xml ili .urdf) identificira vrstu modela. Ako se izostavi nastavak datoteke, funkcija pretpostavlja da se radi o XML formatu datoteke koji se češće upotrebljava za uvoz CAD modela.

Npr.,

naredba **smimport('sm_humanoid.urdf')** kaže funkciji da uveze URDF model s nazivom **sm_humanoid.**

Naredba **smimport('sm_humanoid')** kaže funkciji da uveze multibody model iz posredničke XML datoteke pod nazivom **sm_humanoid.** Ako funkcija ne pronađe XML datoteku sa specificiranim imenom, vraća pogrešku, čak i ako postoji URDF datoteka s istim imenom u istoj mapi. Prema tome, pravilo je da se mora uključiti .urdf nastavak datoteke kad god se uvozi URDF model. [2]

4.4 URDF entiteti koji se mogu uvoziti

Simscape Multibody softver podržava samo dio od svih dostupnih URDF elemenata i atributa. Moguće je uvesti URDF model koji sadrži nepodržane elemente ili atribute, ali takvi entiteti će biti zanemareni. U kodu koji slijedi prikazani su elementi i atributi koji se mogu i oni koji se ne mogu uvoziti. Elementi su prikazani u podebljanom fontu, a atributi u normalnom fontu. Nepodržani elementi i atributi prikazani su crvenom bojom.

```
<robot name>
        <link name>
                <inertial>
                        <origin xyz rpy />
                        <mass value />
                        <inertia ixx iyy izz ixy ixz iyz />
                </inertial>
                <visual name>
                        <origin xyz rpy />
                        <geometry>
                                <box size />
                                <cylinder radius length />
                                <sphere radius />
                                <mesh filename scale />
                        </geometry>
                        <material name>
                                <color rgba />
                                <texture filename />
                        </material>
                </visual>
                <collision name>
                        <origin xyz rpy />
                        <geometry>
```

```
                        <box size />
                        <cylinder radius length />
                        <sphere radius />
                        <mesh filename scale />
                    </geometry>
                </collision>
        </link>
        <joint name type>
                <origin xyz rpy />
                <parent link />
                <child link />
                <axis xyz />
                <calibration rising />
                <calibration falling />
                <dynamics damping friction />
                <limit lower upper effort velocity />
                <mimic joint multiplier offset />
                <safety_controller soft_lower_limit ...
                ... soft_upper_limit k_position k_velocity />
        </joint>
</robot>
```
[2]

4.5 Mapiranje na Simscape Multibody blokove

URDF element **<robot>** mapira se u Simscape Multibody model. Elementi **<link>** smješteni unutar elementa **<robot>** mapiraju se u Simulink Subsystem blokove predstavljajući veze, odnosno u Simscape Multibody nomenklaturi, tijela. Elementi **<joint>** mapiraju se u ekvivalentne Simscape Multibody zglobne blokove (joint blocks). Atributi imena ovih elemenata mapiraju se u odgovarajuće ime modela, imena Subsystem blokova i imena joint blokova. Subsystem (podsustavni) blokovi obuhvaćaju blokove krutosti, inercije, krute transformacije i referentnih sustava.

Blokovi krutosti sadržavaju geometrije i boje tijela, ovi blokovi odgovaraju **<visual>** oznakama URDF modela i nose naziv **Visual.**

Blok inercije sadrži masu, momente inercije i proizvode inercije tijela, ovaj blok odgovara **<inertial>** elementu URDF modela i naziva se **Inertia.**

Blokovi krute transformacije pružaju translacijske i rotacijske pomake od lokalnog referentnog okvira tijela do inercijskih i vizualnih elemenata. Ove transformacije su izvedene iz **<origin>** elemenata **<inertial>** i **<visual>** elemenata veza, kao i iz **<origin>** i **<axis>** elemenata zglobova.

Blok referentnog okvira (Reference Frame block) identificira lokalni referentni okvir tijela. Vrsta zglobnog bloka koji se koristi ovisi o **<type>** atributu **<joint>** elementa. Mapiranje zglobova između URDF i Simscape Multibody softvera je velikim dijelom intuitivno.

Npr. **<joint>** element vrste "prismatic" mapira se u "Prismatic Joint" blok, **<joint>** element vrste "fixed" mapira se u "Weld Joint" blok. [2]

Tablica 4.1 pokazuje mapiranje ostalih URDF **<joint>** elemenata.

Tablica 4. 1 Mapiranje URDF <joint> elemenata

URDF <joint type> atribut	Simscape Multibody zglobni blok	Stupnjevi slobode
revolute	Revolute Joint	jedna rotacija sa zglobnim ograničenjima
continuous	Revolute Joint	jedna rotacija (bez zglobnih ograničenja)
prismatic	Prismatic Joint	jedna translacija sa zglobnim ograničenjima
fixed	Weld Joint	nula (kruta veza)
floating	6-DOF Joint	3 translacije i 3 rotacije
planar	Planar Joint	2 rotacije i 1 translacija

5. RIGID BODY TREE MODEL ROBOTA (RBT)

Rigid body tree (stablo krutog tijela) model je reprezentacija robotske strukture. Koristi se za prikazivanje robota kao što su manipulatori ili nekih drugih kinematičkih stabala. Za izradu ovih modela koriste se **rigidBodyTree** objekti. RBT se sastoji od krutih tijela (**rigidBody**) koji su spojeni preko zglobova (**rigidBodyJoint**). Svako kruto tijelo ima zglob koji definira kako se to tijelo kreće u odnosu na roditelja ("parent") u stablu. Specificiranje transformacije iz jednog tijela u drugo moguće je pomoću postavljanja fiksne transformacije na svaki zglob (**setFixedTransform**). Tijela se mogu dodavati, zamjenjivati ili uklanjati sa RBT modela. Također se mogu zamjenjivati zglobovi određenih tijela. RBT objekt održava odnose i ažurira svojstva **rigidBody** objekta i na taj način odražava ovaj odnos. Transformacije između različitih okvira tijela mogu se dobiti i pomoću naredbe **getTransform**. [1]

5.1 Komponente stabla krutog tijela (Rigid Body Tree)

5.1.1 Base (baza)

Svako stablo krutog tijela ima bazu. Baza definira globalni koordinatni sustav i predstavlja točku pričvršćivanja za kruto tijelo. Baza se ne može modificirati, samo joj se može promijeniti ime. To se postiže promjenom **BaseName** svojstva RBT-a. [1]

5.1.2 Rigid Body (kruto tijelo)

Kruto tijelo je osnovni građevni blok RBT modela i stvara se pomoću naredbe **rigidBody**. Kruto tijelo, koje se naziva i veza (**link**), predstavlja čvrsto tijelo koje se ne može deformirati. Udaljenost između bilo koje dvije točke istog krutog tijela ostaje konstantna. **Slika 5.1** daje shematski prikaz jednog krutog tijela. [1]

Slika 5. 1 Shematski prikaz krutog tijela [1]

Kada se dodaju u RBT koji se sastoji od više tijela, kruta tijela imaju tijela roditelje ("parent bodies") i tijela djecu ("children bodies") koja su povezana s njima (Parent or Children properties/ svojstva roditelja ili djece). Roditelj je tijelo za koje je vezano ovo kruto tijelo, i to može biti npr. baza robota. Djeca su sva tijela pričvršćena za ovo tijelo nizvodno od baze stabla krutog tijela. Svako kruto tijelo ima koordinatni okvir, odnosno koordinatni sustav, koji je povezan s njim i sadrži **rigidBodyJoint** objekt. [1]

5.1.3 Joint (zglob)

Svako kruto tijelo ima jedan zglob, koji definira kretnju tog krutog tijela u odnosu na njegovog roditelja. To je točka spajanja koja povezuje dva kruta tijela u modelu robota. Za reprezentaciju jednog fizičkog tijela sa više zglobova ili različitim osima kretanja, koristi se više **rigidBody** objekata. Objekt **rigidBodyJoint** podržava fiksni (fixed), rotacijski (revolute) i prizmatični (prismatic) zglob. Sheme tih zglobova prikazuje **Slika 5.2**.

Slika 5. 2 Shematski prikaz zglobova [1]

Ovi zglobovi dozvoljavaju slijedeće kretnje, ovisno o vrsti zgloba:

'fixed' — Bez kretnje. Tijelo je kruto vezano za roditelja

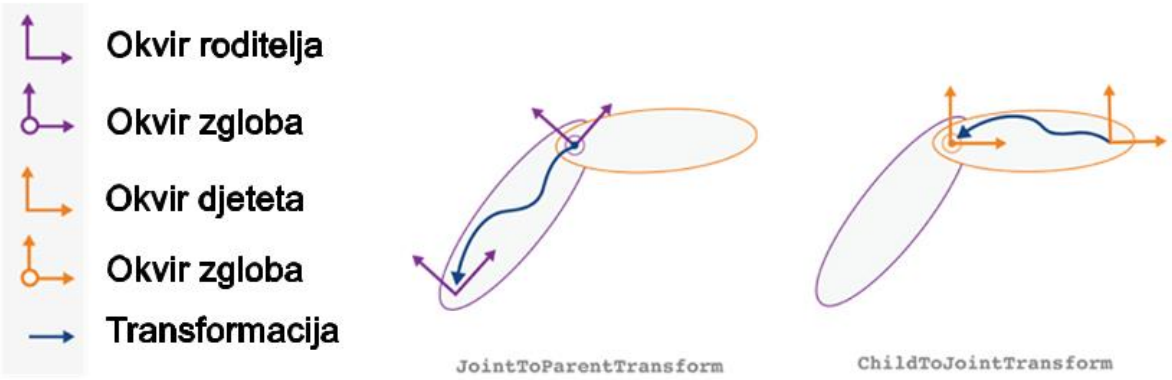
'revolute' — Samo rotacija. Tijelo rotira oko ovog zgloba u odnosu na roditelja. Granice položaja definiraju minimalni i maksimalni kutni položaj u radijanima oko osi rotacije.

'prismatic' — Samo translacija. Tijelo se giba linearno u odnosu na roditelja duž osi kretanja.

Svaki zglob ima os kretanja koja je definirana sa **JointAxis** (os zgloba) svojstvom. Os zgloba je 3-D jedinični vektor koji definira os rotacije (kod rotacijskih zglobova) ili os translacije

(kod prizmatičnih zglobova). **HomePosition** svojstvo definira početni položaj za taj specifični zglob, a taj položaj je točka koja se nalazi unutar granica položaja. Naredba **homeConfiguration** vraća početnu konfiguraciju robota, odnosno skup početnih položaja svih zglobova u modelu.

Zglobovi imaju i svojstva koja definiraju fiksnu transformaciju između koordinatnih sustava roditelja i djece. Ta se svojstva mogu postaviti samo pomoću **setFixedTransform** metode. Ovisno o načinu unosa transformacijskih parametara, pomoću ove metode može se postaviti **JointToParentTransform** ili **ChildToJointTransform** svojstvo. Kada izaberemo željeno svojstvo, drugo svojstvo se automatski postavlja na **identity matrix** (matrica identiteta). **Slika 5.3** prikazuje što svako svojstvo označava.



Slika 5. 3 Prikaz koordinatnih sustava krutih tijela [1]

JointToParentTransform (transformacija iz k.s. zgloba u k.s. tijela roditelja) definira gdje se nalazi k.s. zgloba tijela djeteta u odnosu na k.s. roditelja. Kada je **JointToParentTransform** definiran kao matrica identiteta, koordinatni sustavi zgloba i tijela roditelja se podudaraju.

ChildToJointTransform (transformacija iz k.s. tijela djeteta u k.s. zgloba) definira gdje se nalazi k.s. zgloba tijela djeteta u odnosu na k.s. tijela djeteta. . Kada je **ChildToJointTransform** definiran kao matrica identiteta, koordinatni sustavi zgloba i tijela djeteta se podudaraju.

Napomena: stvarni položaji zglobova nisu dio ovog **Joint** objekta. Model robota nema stanje. Postoji posredna transformacija između koordinatnih sustava zgloba tijela djeteta i tijela roditelja koja definira položaj zgloba duž osi kretanja. Ta transformacija definirana je u konfiguraciji robota. [1]

5.2 Konfiguracija robota

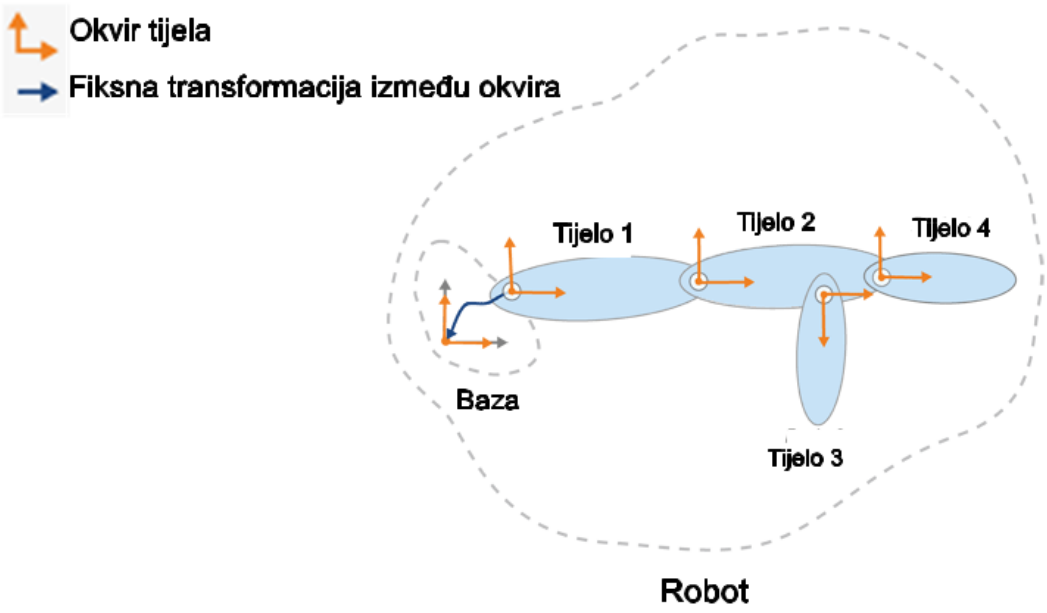
Nakon potpunog sastavljanja modela robota i definiranja transformacija između različitih tijela, mogu se stvarati konfiguracije robota. Konfiguracija definira pozicije svih zglobova robota preko imena zglobova. Pomoću naredbe **homeConfiguration** dobiva se **HomePosition** svojstvo svakog zgloba i stvara početna konfiguracija. Konfiguracije zglobova dane su kao niz struktura.

Primjer iz Matlaba:

```
config = homeConfiguration(robot)

config =

  1×6 struct array with fields:

    JointName
    JointPosition
```

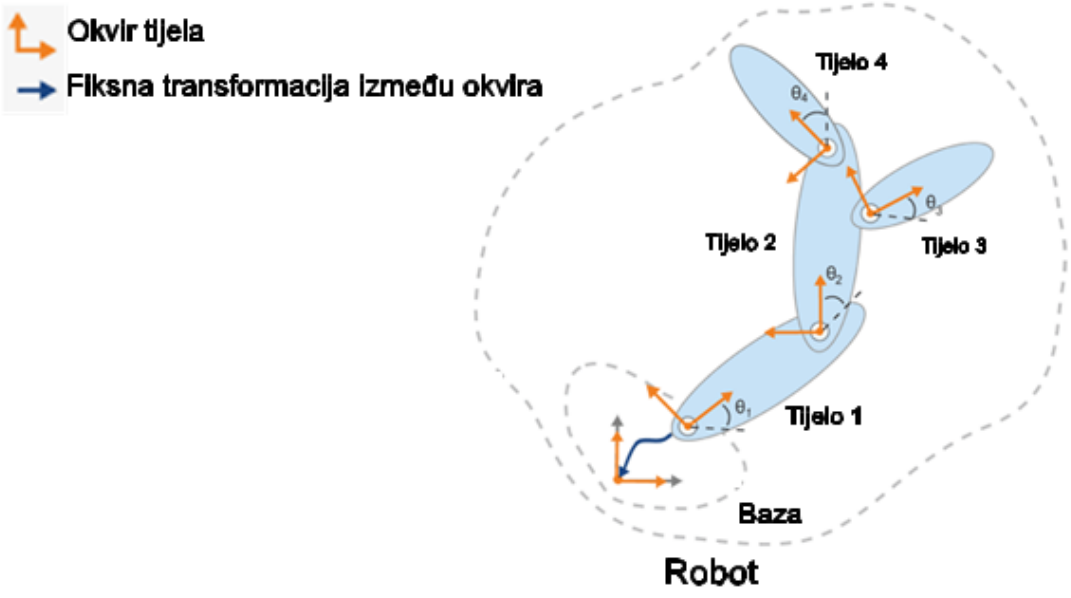
Svaki element u nizu je struktura koja sadrži ime i poziciju jednog od zglobova robota.

```
config(1)

ans =

  struct with fields:

        JointName: 'jnt1'
    JointPosition: 0
```



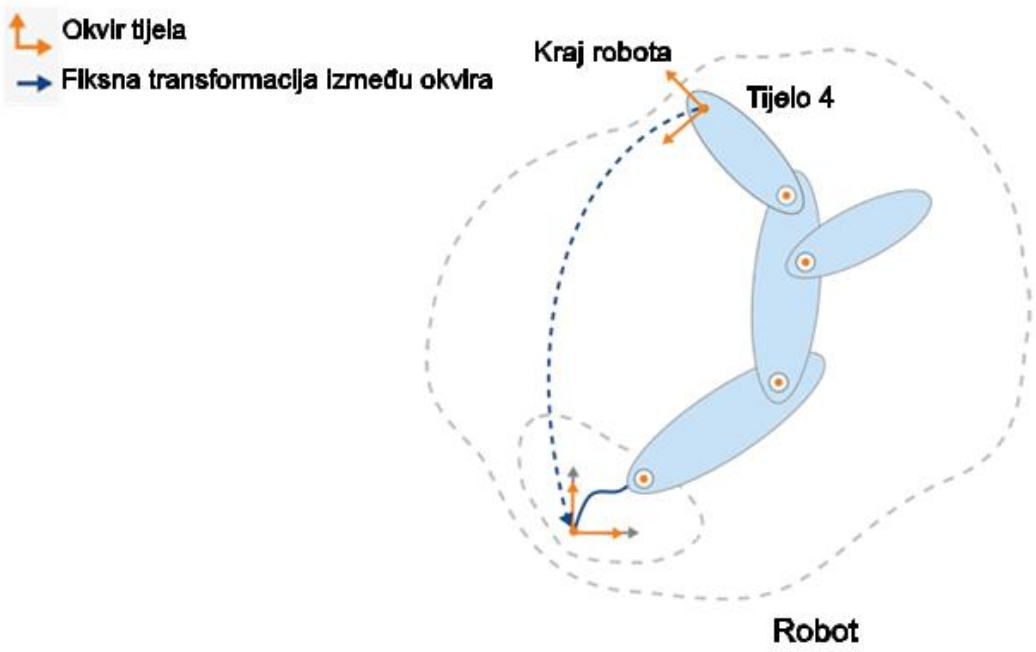
Slika 5. 4 Početna konfiguracija robota [1]

Slika 5.4 shematski prikazuje početnu konfiguraciju robota. Pomoću funkcije **randomConfiguration** može se generirati nasumična konfiguracija robota koja poštuje sva ograničenja zglobova. Njen prikaz dan je na **Slici 5.5**.



Slika 5. 5 Nasumična konfiguracija robota [1]

Konfiguracije robota se koriste kod prikazivanja slike robota pomoću naredbe **show.** Transformaciju između koordinatnih sustava dvaju tijela s određenim konfiguracijama moguće je dobiti pomoću naredbe **getTransform**.



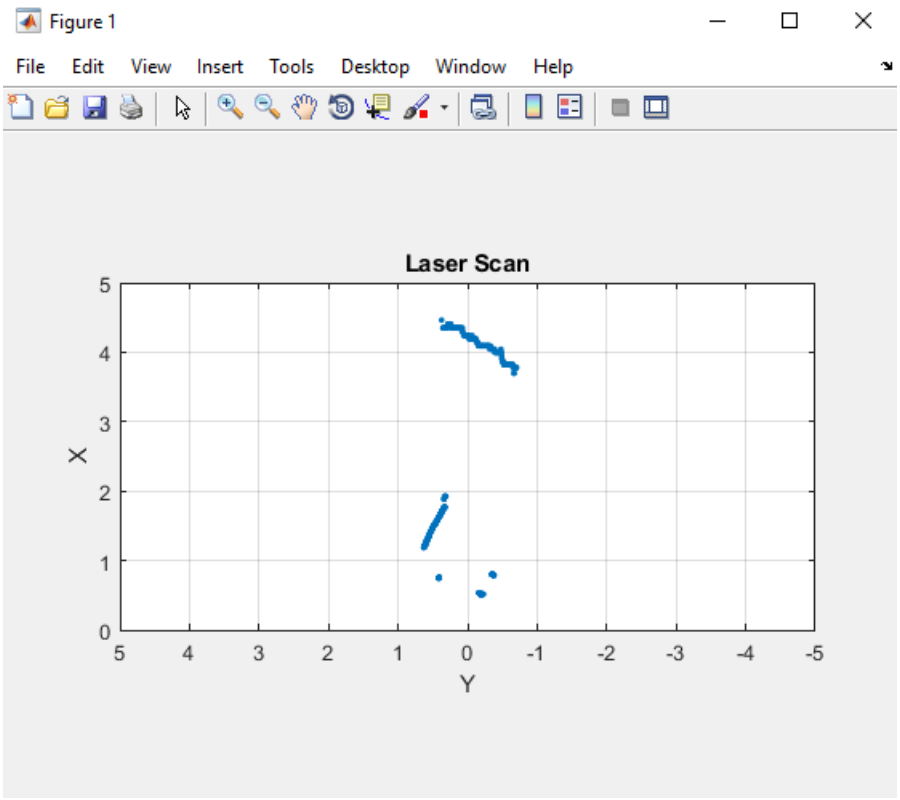
Slika 5. 6 Prikaz koordinatnog sustava kraja robota [1]

Na **Slici 5.6** dan je prikaz koordinatnog sustava kraja robota. Za dobivanje konfiguracije robota sa određenim položajem kraja robota (end effector), koristi se naredba **inverseKinematics**. Ova naredba aktivira algoritam koji određuje potrebne kutove zglobova robota kako bi se postigao željeni položaj određenog krutog tijela, dakle rješava inverzni kinematički problem. [1]

6. PRIMJERI RADA U ROS MREŽI

6.1 Primjer 1

```
%example messages (LaserScan)/ primjer poruka u ROS mreži (lasersko
skeniranje)
exampleHelperROSLoadMessages
emptyscan = rosmessage('sensor_msgs/LaserScan')
scan
xy = readCartesian(scan);
figure
plot(scan,'MaximumRange',5)
```

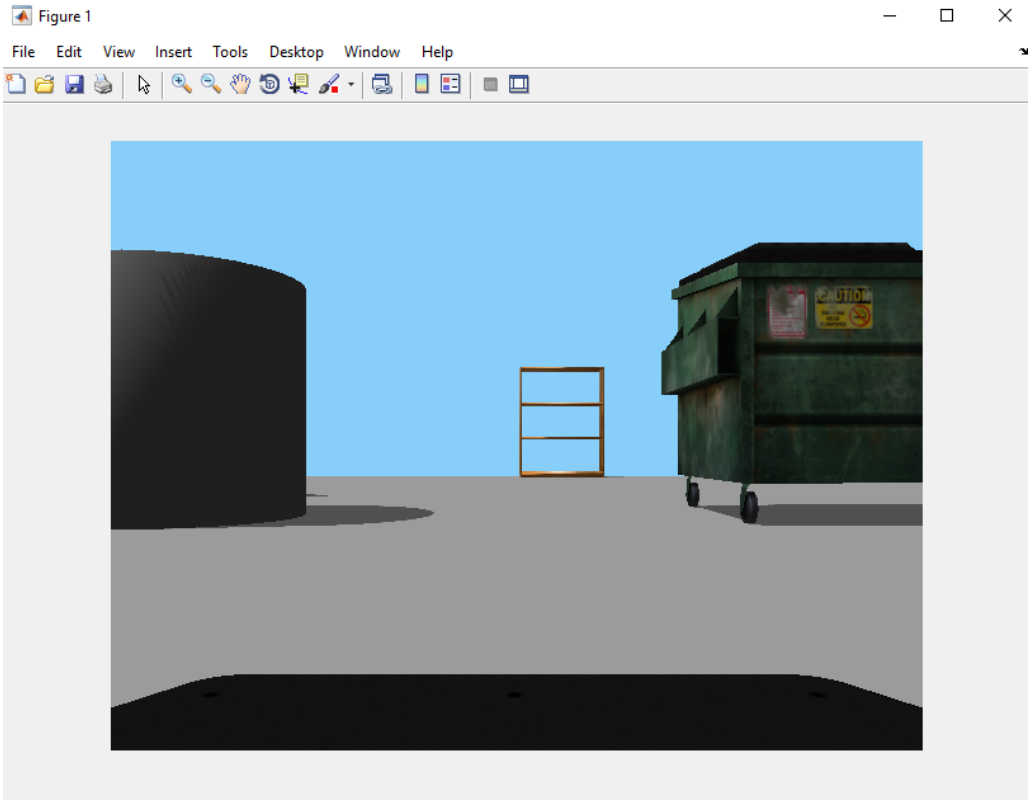


Slika 6. 1 Podatci od skenera

Na **Slici 6.1** prikazani su podatci od skenera preuzeti od robota preko ROS mreže i otvoreni u Matlab-u.

6.2 Primjer 2

```
%example messages (image)/ primjer poruka u ROS mreži (slika)
exampleHelperROSLoadMessages
emptyimg = rosmessage('sensor_msgs/Image')
img
imageFormatted = readImage(img);
figure
imshow(imageFormatted)
```

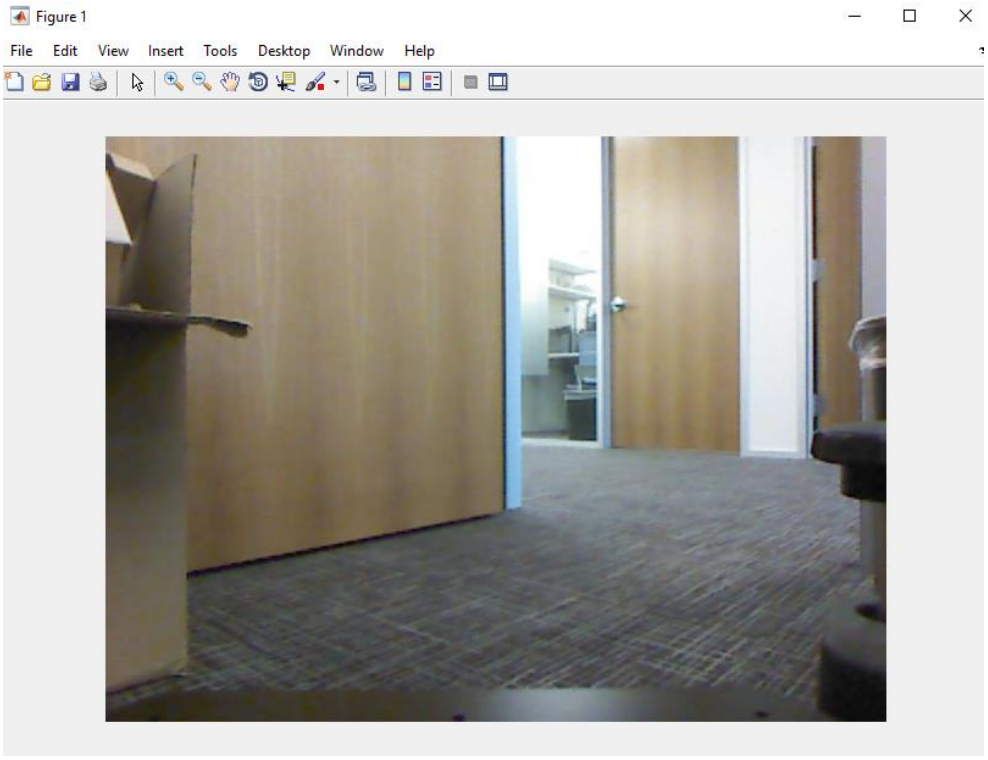


Slika 6. 2 Podatak slika

Slika 6.2 daje prikaz slike koja je u Matlab-u preuzeta od robota preko ROS mreže.

6.3 Primjer 3

```
%example messages (compressed messages)/ primjer poruka u ROS mreži
(stisnute poruke)
exampleHelperROSLoadMessages
emptyimgcomp = rosmessage('sensor_msgs/CompressedImage')
imgcomp
compressedFormatted = readImage(imgcomp);
figure
imshow(compressedFormatted)
```

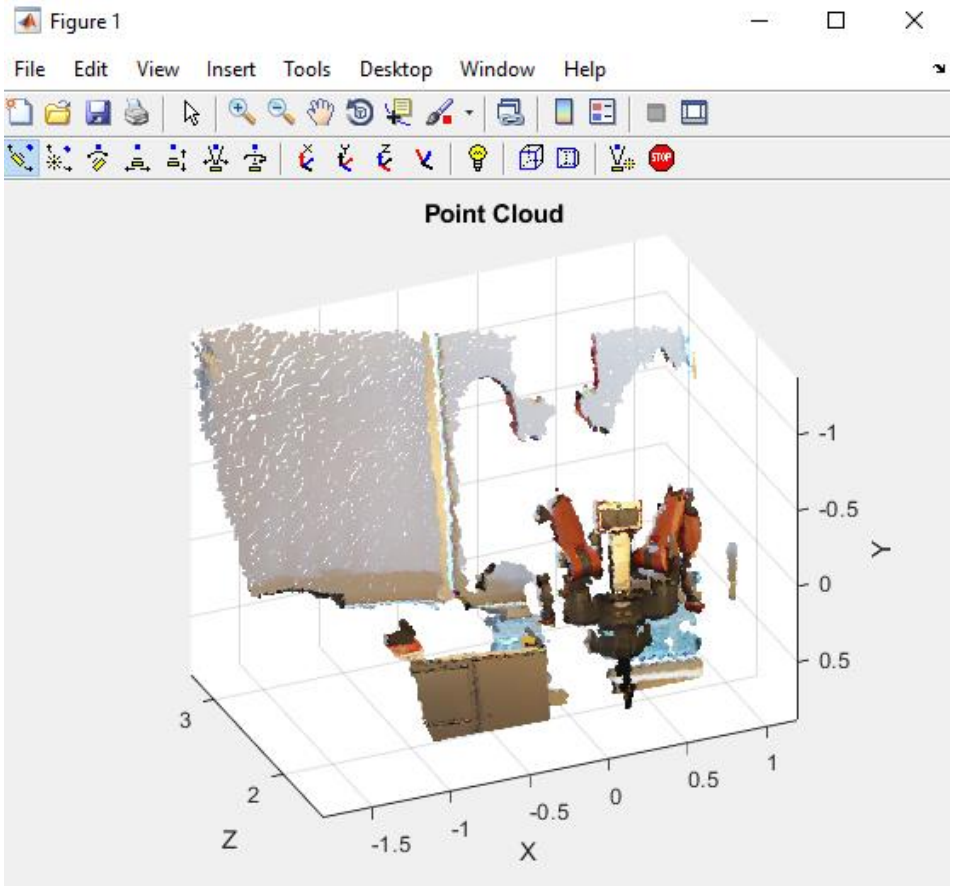


Slika 6. 3 Podatak slika_2

Slika 6.3 također prikazuje sliku koja je u Matlab-u preuzeta od robota preko ROS mreže.

6.4 Primjer 4

```
%example messages (compressed messages)/ primjer poruka u ROS mreži
(stisnute poruke)
exampleHelperROSLoadMessages
emptyptcloud = rosmessage('sensor_msgs/PointCloud2');
ptcloud
xyz = readXYZ(ptcloud);
xyzvalid = xyz(~isnan(xyz(:,1)),:);
rgb = readRGB(ptcloud);
figure
scatter3(ptcloud)
```



Slika 6. 4 Oblak točaka

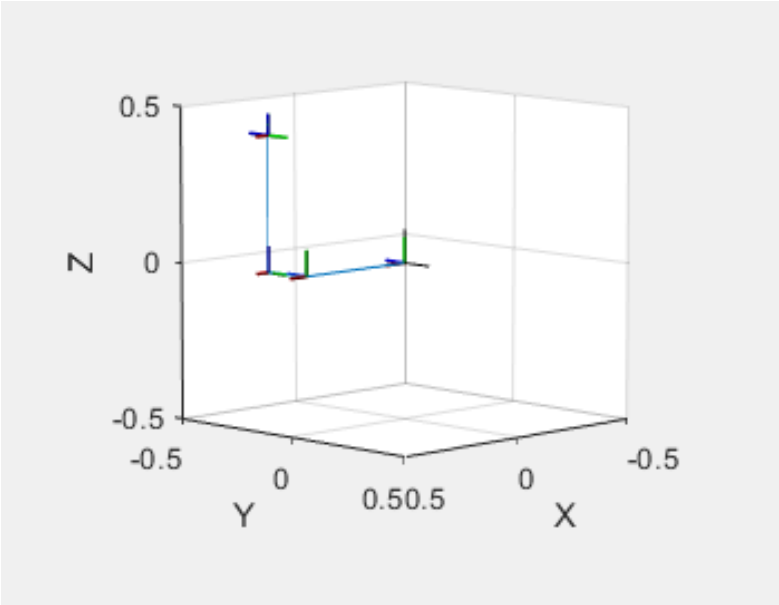
Na **Slici 6.4** prikazan je oblak točaka preuzet od robota preko ROS mreže.

7. IZRADA ROBOTSKOG MANIPULATORA POMOĆU DENAVIT-HARTEMBERG (DH) PARAMETARA

```
%U ovom primjeru korite se DH parametri robota Puma560 za izradu vlastitog
robota. Svako kruto tijelo dodaje se zasebno,
%i određuje mu se transformacija djeteta u roditelja (child-to-parent
transform) koja se specificira preko joint objekta.
%DH parametri definiraju geometriju robota na način da određuju kako je
svako pojedino kruto tijelo vezano za svog roditelja.
%Radi praktičnosti u ovom primjeru su parametri za Puma560 robota zapisani
u matrici. Puma robot je serijski ulančani manipulator.

dhparams = [0       pi/2    0       0;
            0.4318  0       0       0
            0.0203  -pi/2   0.15005 0;
            0       pi/2    0.4318  0;
            0       -pi/2   0       0;
            0       0       0       0];

% Stvaranje RBT objekta za izgradnju robota.
robot = rigidBodyTree;

%Stvaranje prvog krutog tijela i dodavanje istog robotu. Za dodavanje
krutog tijela potrebno je:
% 1)stvoriti rigidBody objekt i dati mu jedinstveno ime
% 2)koristiti naredbu setFixedTransform za specifikaciju transformacije
između tijela pomoću DH parametara.
% Zadnji element DH parametara, theta, se ignorira jer taj kut ovisi o
položaju zgloba.
% 3)pozvati funkciju addBody kako bi se zglob prvog tijela pričvrstio na
okvir baze.

body1 = rigidBody('body1');
jnt1 = rigidBodyJoint('jnt1','revolute');

setFixedTransform(jnt1,dhparams(1,:),'dh');
body1.Joint = jnt1;

addBody(robot,body1,'base')
```

```
% Stvaranje i dodavanje ostalih krutih tijela robotu.
%Kod zvanja funkcije addBody potrebno je specificirati ime prethodnog
tijela na koje se novo tijelo pričvršćuje.
% Svaka fiksna transformacija relativna je koordinatnom sustavu prethodnog
zgloba.

body2 = rigidBody('body2');
jnt2 = rigidBodyJoint('jnt2','revolute');
body3 = rigidBody('body3');
jnt3 = rigidBodyJoint('jnt3','revolute');
body4 = rigidBody('body4');
jnt4 = rigidBodyJoint('jnt4','revolute');
body5 = rigidBody('body5');
jnt5 = rigidBodyJoint('jnt5','revolute');
body6 = rigidBody('body6');
jnt6 = rigidBodyJoint('jnt6','revolute');

setFixedTransform(jnt2,dhparams(2,:),'dh');
setFixedTransform(jnt3,dhparams(3,:),'dh');
setFixedTransform(jnt4,dhparams(4,:),'dh');
setFixedTransform(jnt5,dhparams(5,:),'dh');
setFixedTransform(jnt6,dhparams(6,:),'dh');

body2.Joint = jnt2;
body3.Joint = jnt3;
body4.Joint = jnt4;
body5.Joint = jnt5;
body6.Joint = jnt6;

addBody(robot,body2,'body1')
addBody(robot,body3,'body2')
addBody(robot,body4,'body3')
addBody(robot,body5,'body4')
addBody(robot,body6,'body5')
```

```
% Za verifikaciju ispravnosti robota koristi se funkcija showdetails ili
funkcija show.
% Funkcija showdetails daje listu svih tijela u Matlabovom naredbenom
prozoru (command window).
% Funkcija show daje prikaz robota u definiranoj konfiguraciji (koja je po
difoltu početna ili home konfiguracija).
% Funkcija axis služi za modificiranje granica osi kod prikaza robota.

showdetails(robot)
show(robot);
axis([-0.5,0.5,-0.5,0.5,-0.5,0.5])
```



Slika 7. 1 Prikaz robota

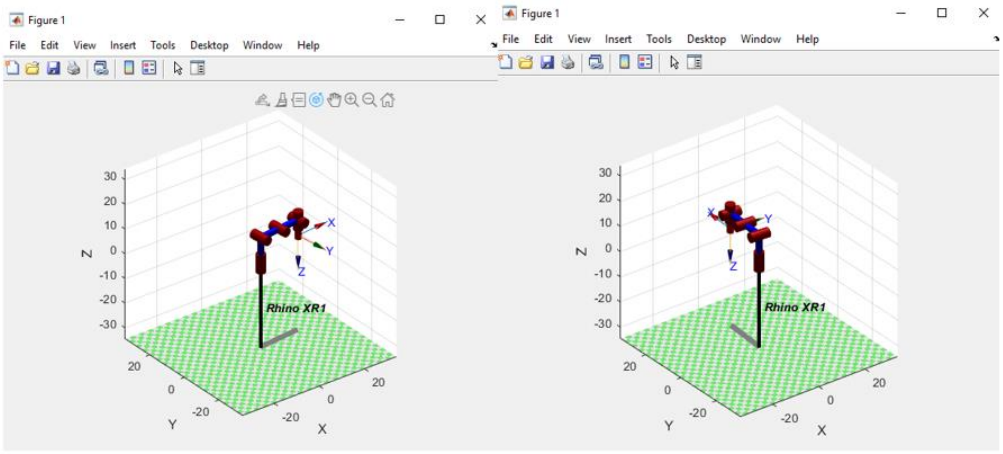
Na **Slici 7.1** prikazana je struktura robota izgrađenog pomoću DH parametara u Robotics Toolbox-u Peter-a Corke-a. [3]

8. SIMULACIJA ROBOTA POMOĆU DH PARAMETARA

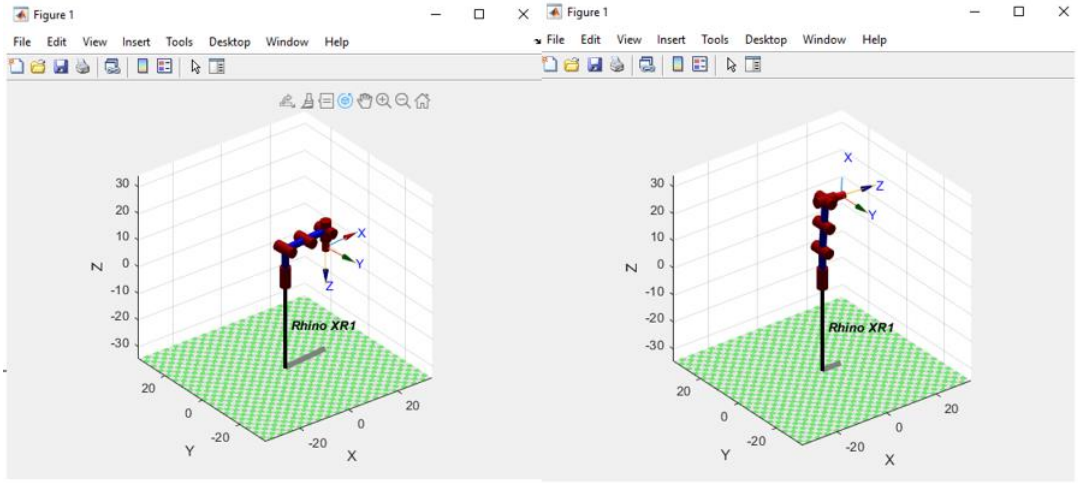
Za ovaj primjer potrebno je instalirati Robotics Toolbox for Matlab od Peter-a Cork-a. [4]

```
L1=10.25; L2=9; L3=9; L5=6.25;
% L=Link([Th d a alph])
L(1)=Link([0 L1 0 pi/2]);
L(2)=Link([0 0 L2 0]);
L(3)=Link([0 0 L3 0]);
L(4)=Link([0 0 0 pi/2]);
L(5)=Link([0 L5 0 0]);
Robot= SerialLink(L);
Robot.name= 'Rhino XR1';
q1=0; q2=0; q3=0; q4=0;q5=0;
Robot
for th1=0:0.1:pi/2
    Robot.plot([th1 0 0 0 0]);
end
for th2=0:0.1:pi/2
    Robot.plot([0 th2 0 0 0]);
end
th2=0;th3=0;
for th1=0:0.1:pi/2
    th2=th2+0.2;
    th3=th3+0.15;
    Robot.plot([th1 th2 0 th3 th3]);
    pause(0.1)
end
```

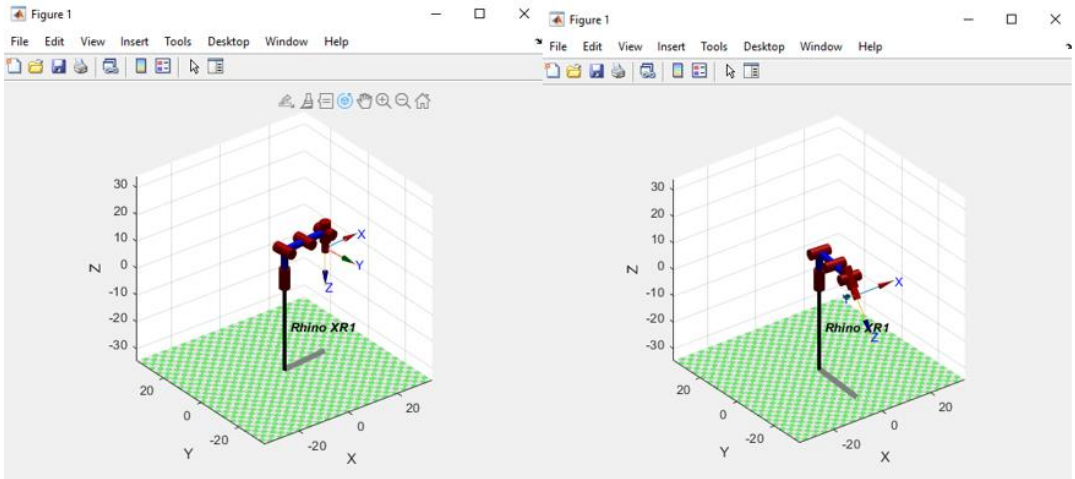
Slike **8.1, 8.2** i **8.3** koje slijede prikazuju početne i krajnje položaje robotskog manipulatora za pokretanje različitih zglobova robota.



Slika 8. 1 Rotiranje zgloba 1 za kut od 90°



Slika 8. 2 Rotiranje zgloba 2 za kut od 90°



Slika 8. 3 Istovremeno rotiranje zglobova 1, 2, 4 i 5

9. GENERIRANJE I SLIJEĐENJE TRAJEKTORIJE ROBOTSKOG MANIPULATORA

Stvaranje uzorka međutočaka za generiranje trajektorije.

Definiranje podataka potrebnih za generiranje trajektorije robotskog manipulatora. Možemo koristiti različite robote i različite međutočke, ovdje je prikazan skup potrebnih podataka koji se mogu mijenjati po želji.

```
%Stvaranje uzorka međutočaka za generiranje trajektorije.
%Podatci o stablu krutih tijela (RBT)
load gen3
load gen3positions % ovdje su smješteni podatci o kutovima zglobova te
orijentaciji i poziciji kraja robota
eeName = 'Gripper';
numJoints = numel(gen3.homeConfiguration);
ikInitGuess = gen3.homeConfiguration;
%Maksimalni broj međutočaka (za Simulink)
maxWaypoints = 20;
% Pozicije (X Y Z)
waypoints = toolPositionHome' + ...
            [0 0 0.2 ; -0.1 0.2 0.4 ; -0.2 0 0.1 ; -0.1 -0.2 0.4 ; 0 0
0.2]';
waypoints %naredba za prikaz međutočaka, svaki stupac predstavlja x, y i z
koordinate pojedine međutočke, ukupno je definirano 5 međutočaka u ovom
primjeru.

%Eulerovi kutovi (Z Y X) u odnosu na početnu (home) orijentaciju.
orientations = [0     0    0;
                pi/8  0    0;
                0    pi/2  0;
               -pi/8  0    0;
                0     0    0]';
```

```
%Niz vremena međutočaka
waypointTimes = 0:4:16;
%Vrijeme uzorkovanja trajektorije
ts = 0.2;
trajTimes = 0:ts:waypointTimes(end);
%Dodatni parametri
%Rubni uvjeti (za polinomske trajektorije)

%Brzine kraja robota u pojedinim međutočkama (za kubičnu i kvintičnu
trajektoriju)
waypointVels = 0.1 *[ 0  1  0;
                     -1  0  0;
                      0 -1  0;
                      1  0  0;
                      0  1  0]';
%Ubrzanja (za kvintičnu trajektoriju)
waypointAccels = zeros(size(waypointVels));
%Vremena ubrzanja (za trapezoidnu trajektoriju)
waypointAccelTimes = diff(waypointTimes)/4;
```

U naredbenom prozoru Matlaba (Command Window) mogu se vidjeti definirane međutočke za trajektoriju.

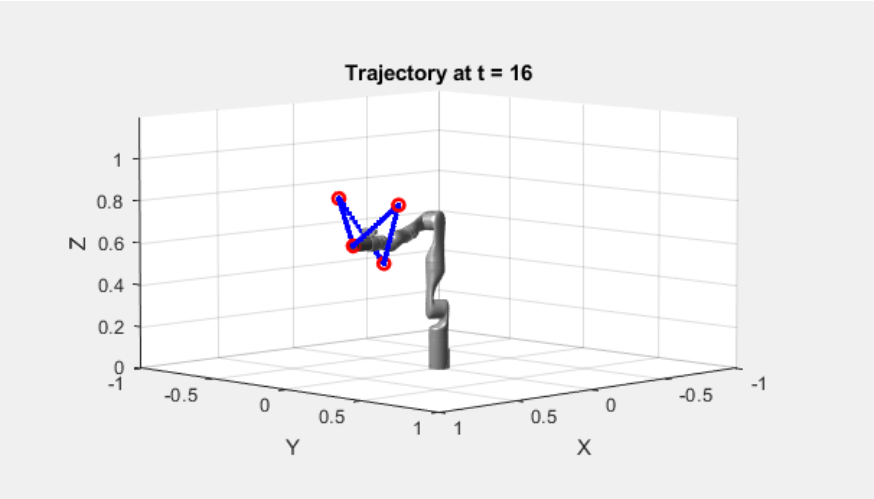
```
waypoints =

    0.5639    0.4639    0.3639    0.4639    0.5639

    0.0013    0.2013    0.0013   -0.1987    0.0013

    0.6336    0.8336    0.5336    0.8336    0.6336
```

9.1 Primjer 1

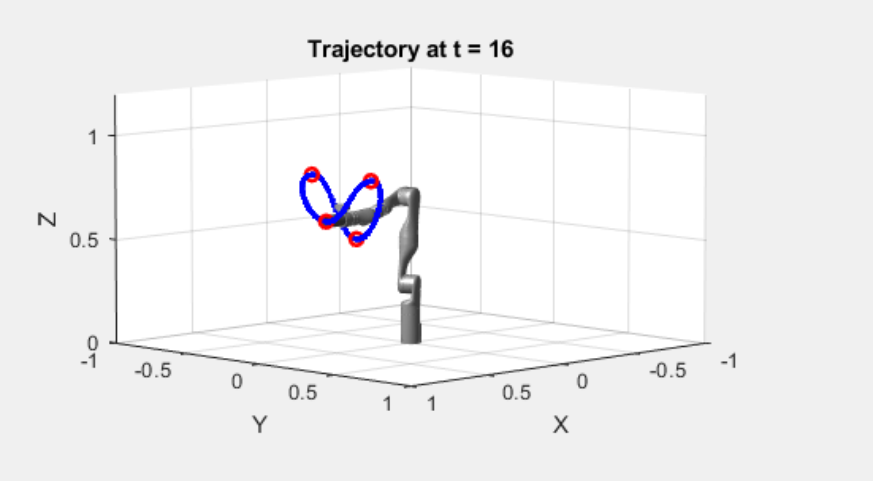
U ovom primjeru koriste se podatci iz Matlabovog koda s prethodne stranice. Koristi se model robota Kinova Gen3 koji je jedan od robotskih modela dostupnih u Robotics System Toolbox-u u Matlabu.

```
%Definiranje podataka međutočaka
createWaypointData; % učitavanje podataka iz koda s prethodne stranice!

%Definiranje podataka potrebnih za algoritam koji rješava inverzni
kinematički problem.
ik = inverseKinematics('RigidBodyTree',gen3);
ikWeights = [1 1 1 1 1 1];
ikInitGuess = gen3.homeConfiguration;
showdetails(gen3)

% Postavljanje plot-a
plotMode = 1; % 0 = None, 1 = Trajectory, 2 = Coordinate Frames
show(gen3,jointAnglesHome','Frames','off','PreservePlot',false);
xlim([-1 1]), ylim([-1 1]), zlim([0 1.2])
hold on
if plotMode == 1
    hTraj = plot3(waypoints(1,1),waypoints(2,1),waypoints(3,1),'b.-');
end
plot3(waypoints(1,:),waypoints(2,:),waypoints(3,:),'ro','LineWidth',2);

%Kod za generiranje trajektorije
% Samo kartezijski pokreti
trajType = 'trap'; %Ovdje se može birati vrsta trajektorije ('trap',
'cubic', 'quintic' ili 'bspline')

switch trajType
    case 'trap'
        [q,qd,qdd] = trapveltraj(waypoints,numel(trajTimes), ...
            'AccelTime',repmat(waypointAccelTimes,[3 1]), ...
            'EndTime',repmat(diff(waypointTimes),[3 1]));

    case 'cubic'
        [q,qd,qdd] = cubicpolytraj(waypoints,waypointTimes,trajTimes, ...
            'VelocityBoundaryCondition',waypointVels);
```

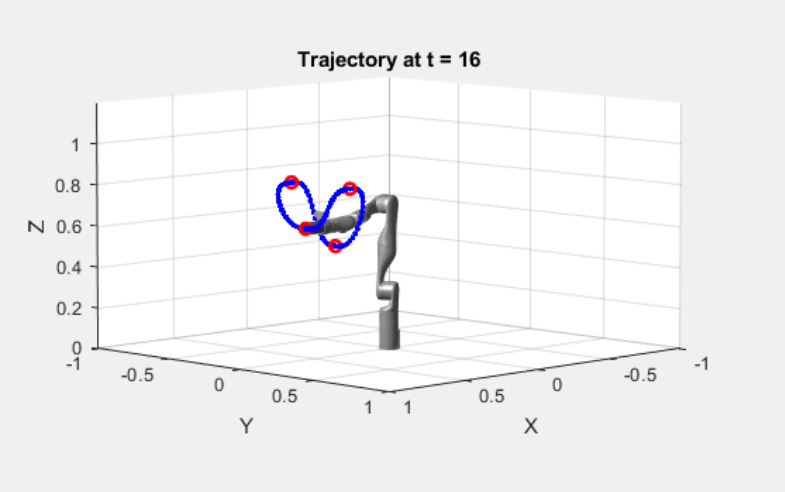
```
    case 'quintic'
        [q,qd,qdd] = quinticpolytraj(waypoints,waypointTimes,trajTimes, ...
            'VelocityBoundaryCondition',waypointVels, ...
            'AccelerationBoundaryCondition',waypointAccels);

    case 'bspline'
        ctrlpoints = waypoints;
        [q,qd,qdd] = bsplinepolytraj(ctrlpoints,waypointTimes([1
end]),trajTimes);

    otherwise
        error('Invalid trajectory type! Use ''trap'', ''cubic'',
''quintic'', or ''bspline''');
end

%Prikazivanje cjelokupne trajektorije zajedno sa stablom krutih tijela
(RBT)
if plotMode == 1
    set(hTraj,'xdata',q(1,:),'ydata',q(2,:),'zdata',q(3,:));
elseif plotMode == 2
    plotTransforms(q',repmat([1 0 0 0],[size(q,2) 1]),'FrameSize',0.05);
end

%Za prikaz trajektorije, pokrenuti slijedeću liniju koda
plotTrajectory(trajTimes,q,qd,qdd,'Names',["X","Y","Z"],'WaypointTimes',way
pointTimes)

% Petlja slijeđenja trajektorije
for idx = 1:numel(trajTimes)
    % Rješavanje inverznog kinematičkog problema
    tgtPose = trvec2tform(q(:,idx)');
    [config,info] = ik(eeName,tgtPose,ikWeights,ikInitGuess);
    ikInitGuess = config;

    % Prikaz robota
    show(gen3,config,'Frames','off','PreservePlot',false);
    title(['Trajectory at t = ', num2str(trajTimes(idx))])
    drawnow
end
```



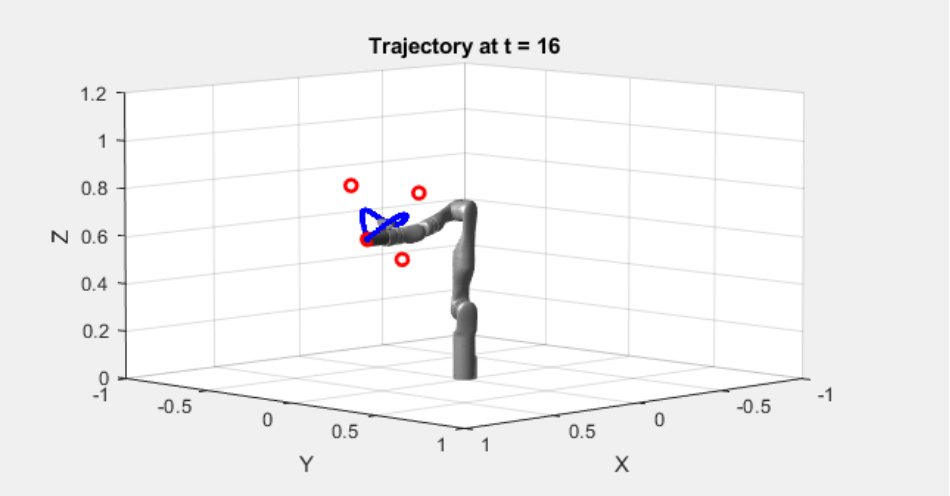
Slika 9. 1 Trap



Slika 9. 2 Cubic



Slika 9. 3 Quintic

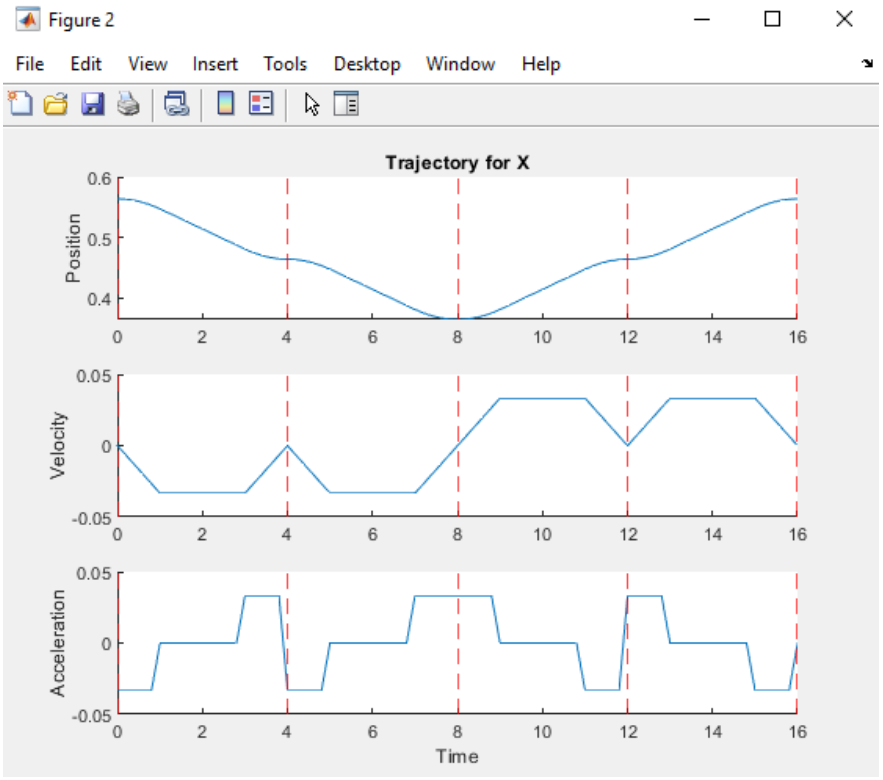


Slika 9. 4 Bspline

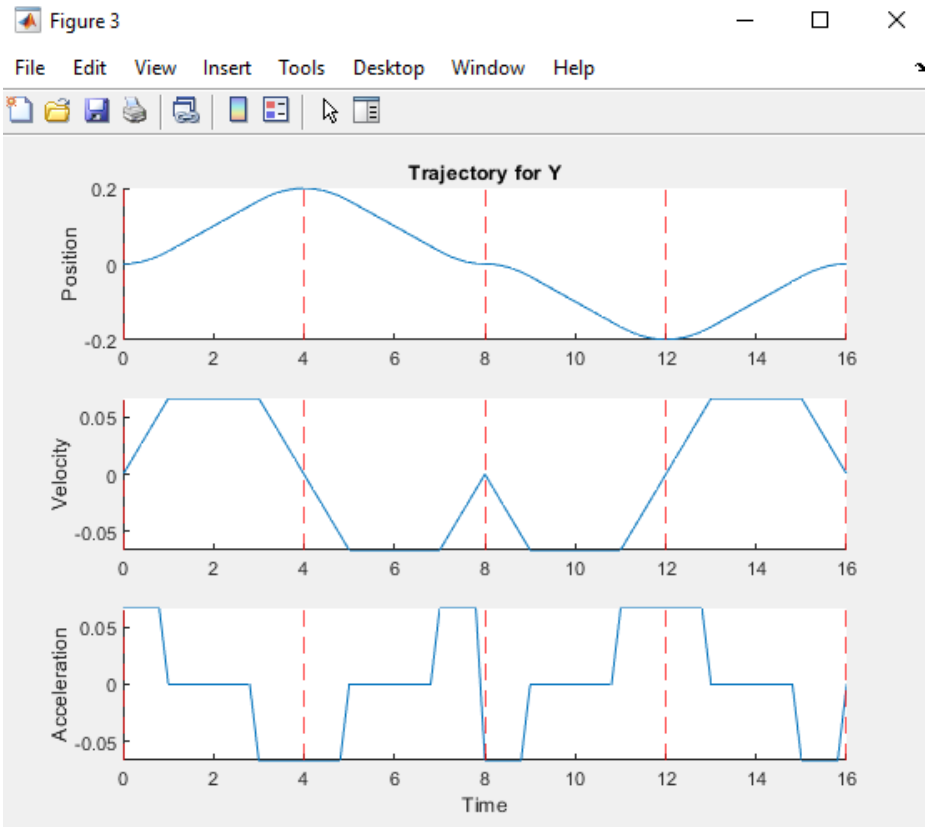
Slike **9.1**, **9.2**, **9.3** i **9.4** prikazuju generiranje trajektorije (redom trapezoidalna, kubična, kvintična I bspline) koje robot Kinova Gen3 slijedi.

```
%Za prikaz trajektorije, pokrenuti slijedeću liniju koda
plotTrajectory(trajTimes,q,qd,qdd,'Names',["X","Y","Z"],'WaypointTimes',way
pointTimes)
```

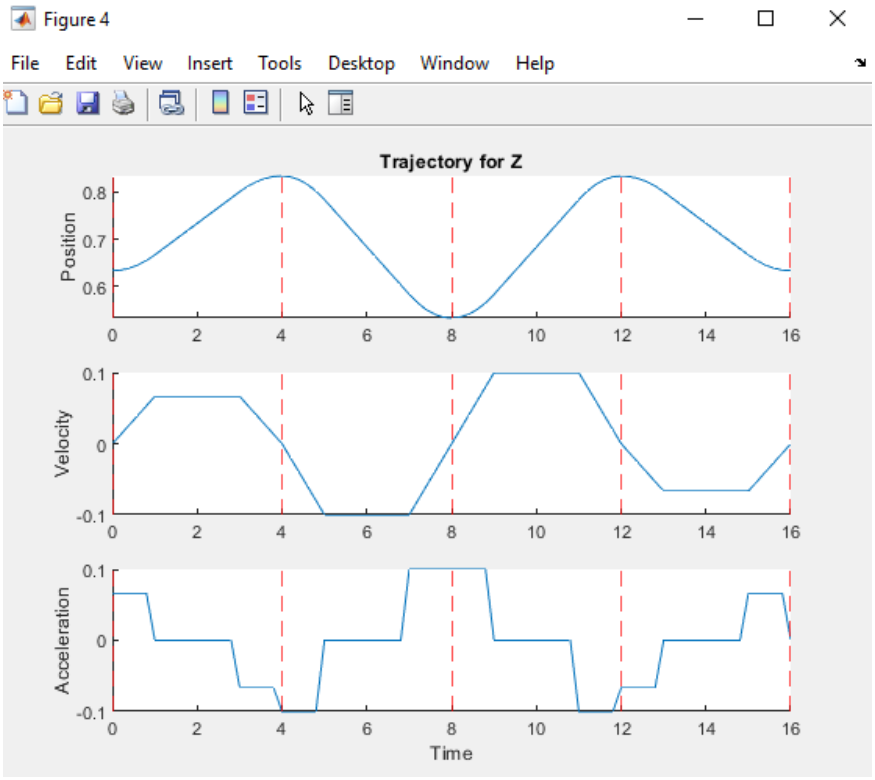
Ovaj dio koda prikazuje podatke o položaju, brzini i ubrzanju kraja robota po x, y i z osima tokom cijele trajektorije. Slike **9.5** do **9.10** prikazuju te podatke za slučajeve trapezoidalne ('trap') i kubične ('cubic') trajektorije.



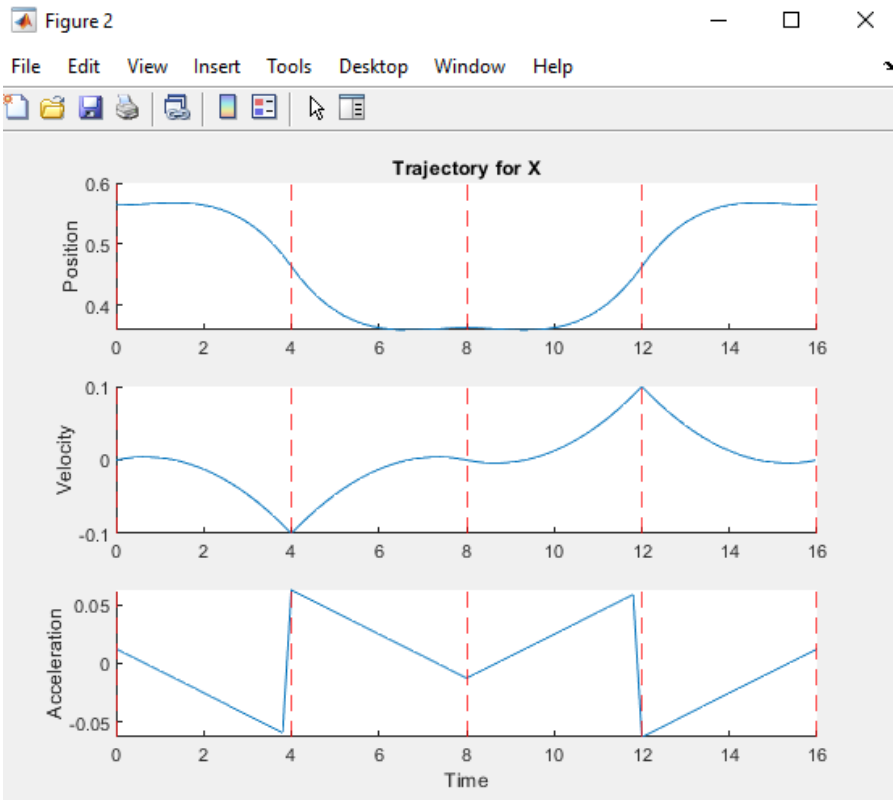
Slika 9. 5 Trapezoidalna podatci za x os



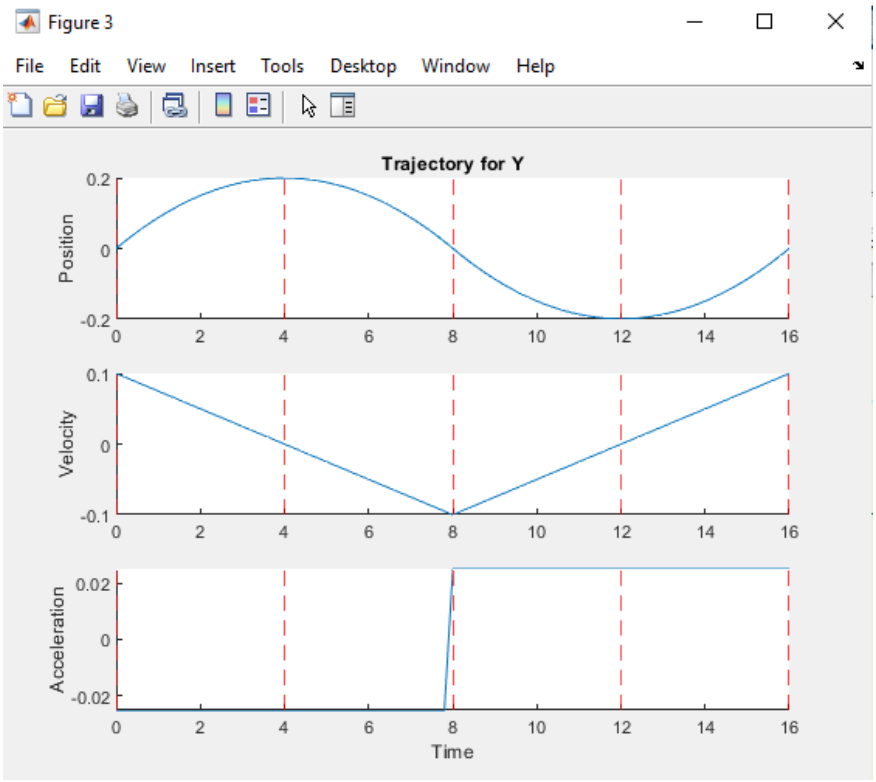
Slika 9. 6 Trapezoidalna podatci za y os



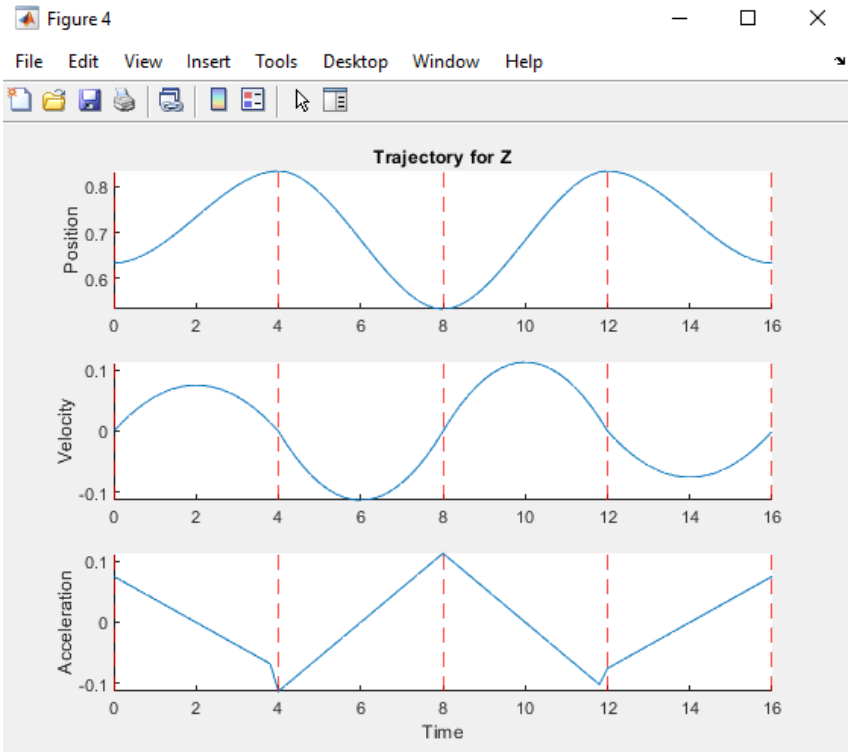
Slika 9. 7 Trapezoidalna podatci za z os



Slika 9. 8 Kubična podatci za x os



Slika 9. 9 Kubična podatci za y os

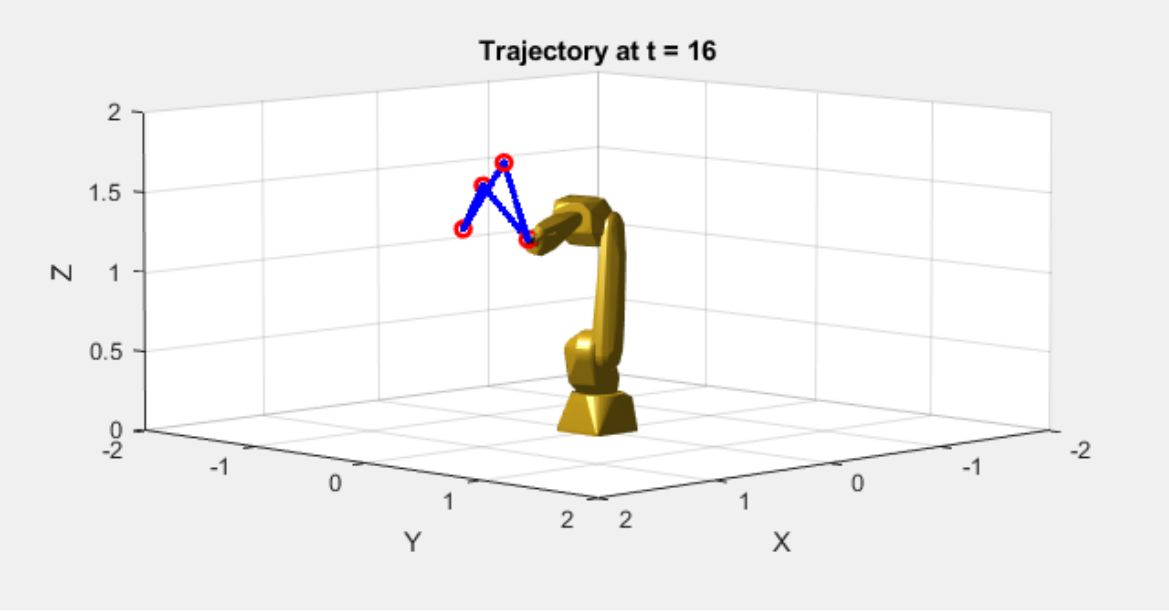


Slika 9. 10 Kubična podatci za z os

9.2 Primjer 2

U ovom primjeru napravljene su neke promjene u prethodnom kodu tako da je korišten model robota FANUC M-16iB i definiran je novi skup međutočaka.

```
%Definiranje modela robota i međutočaka
robot=loadrobot('fanucM16ib');
waypoints=[0.9 0.3 1.3; 1.2 0.4 1.8; 1.4 0.25 1.4; 0.7 -0.3 1.57; 0.9 0.3
1.3]'
jointAnglesHome=[6.4 0.3 3.2 4.5 1.1 2]';
eeName = 'tool0';
showdetails(robot);
Ostatak koda ostao je nepromijenjen.

Nove međutočke su:

 waypoints =

    0.9000    1.2000    1.4000    0.7000    0.9000

    0.3000    0.4000    0.2500   -0.3000    0.3000

    1.3000    1.8000    1.4000    1.5700    1.3000
```



Slika 9. 11 Robot Fanuc M-16iB

Slika **9.11** prikazuje robota FANUC M-16iB i njegovu trajektoriju.

10. INTERAKTIVNO GRAĐENJE TRAJEKTORIJE ZA ABB YUMI ROBOTA

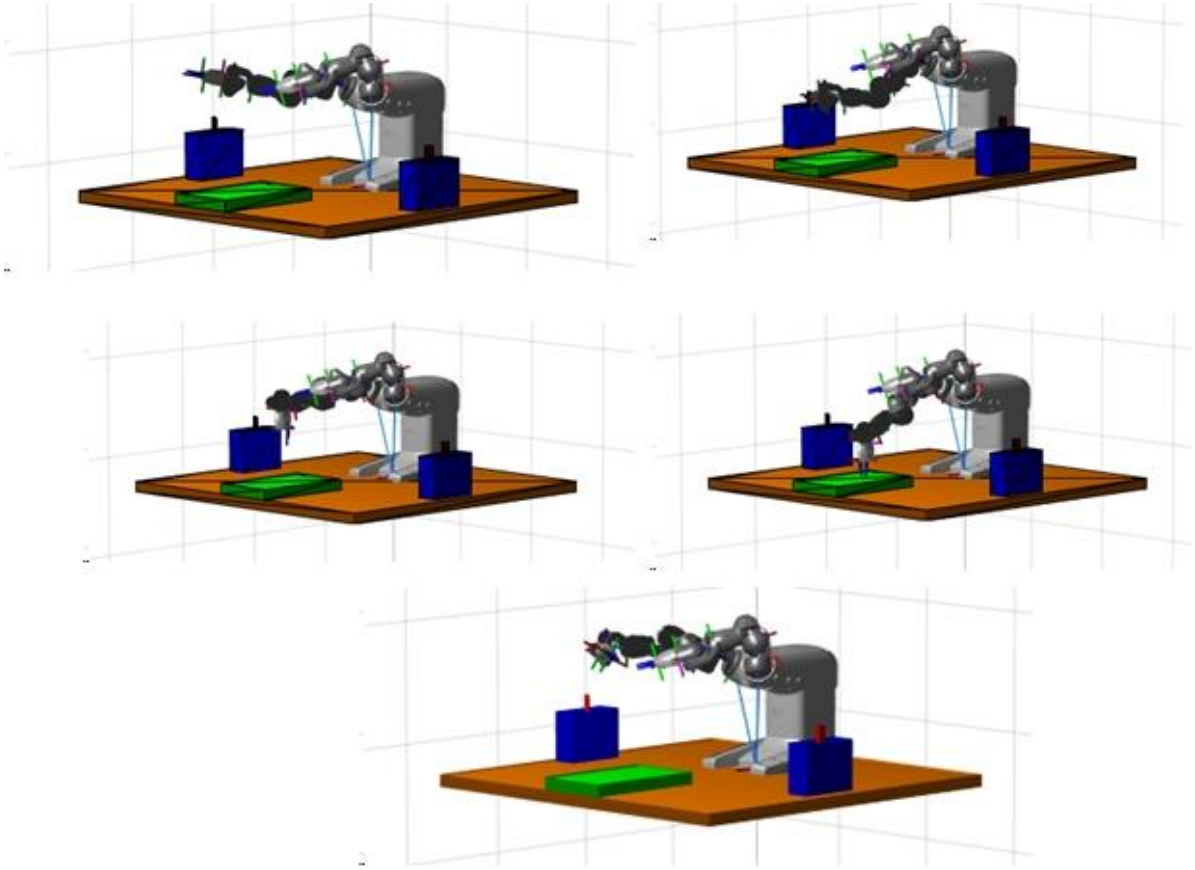
Ovaj primjer pokazuje kako koristiti **interactiveRigidBodyTree** objekt za pomicanje robota, definiranje trajektorije i prikazivanje slijeđenja trajektorije

```
%Učitavanje vizualizacije robota i izgradnja okruženja.

%Učitavanje modela robota 'abbYumi'. Inicijaliziranje interaktivne slike
pomoću naredbe ''interactiveRigidBodyTree''.
%Spremanje trenutnih osi.
robot = loadrobot('abbYumi', 'Gravity', [0 0 -9.81]);
iviz = interactiveRigidBodyTree(robot);
ax = gca;

%Izgradnja okruženja koji se sastoji od sudarnih kutija (collision boxes)
koje predstavljaju pod, dvije police s predmetima i središnji stol.
plane = collisionBox(1.5,1.5,0.05);
plane.Pose = trvec2tform([0.25 0 -0.025]);
show(plane,'Parent', ax);

leftShelf = collisionBox(0.25,0.1,0.2);
leftShelf.Pose = trvec2tform([0.3 -.65 0.1]);
[~, patchObj] = show(leftShelf,'Parent',ax);
patchObj.FaceColor = [0 0 1]; % [0 0 1]--> rgb/red green blue/ crvena
zelena plava/ definiranje boje predmeta

rightShelf = collisionBox(0.25,0.1,0.2);
rightShelf.Pose = trvec2tform([0.3 .65 0.1]);
[~, patchObj] = show(rightShelf,'Parent',ax);
patchObj.FaceColor = [0 0 1];

leftWidget = collisionCylinder(0.01, 0.07);
leftWidget.Pose = trvec2tform([0.3 -0.65 0.225]);
[~, patchObj] = show(leftWidget,'Parent',ax);
patchObj.FaceColor = [1 0 0];
```

```
rightWidget = collisionBox(0.03, 0.02, 0.07);
rightWidget.Pose = trvec2tform([0.3 0.65 0.225]);
[~, patchObj] = show(rightWidget,'Parent',ax);
patchObj.FaceColor = [1 0 0];

centerTable = collisionBox(0.5,0.3,0.05);
centerTable.Pose = trvec2tform([0.75 0 0.025]);
[~, patchObj] = show(centerTable,'Parent',ax);
patchObj.FaceColor = [0 1 0];

%Interaktivno generiranje konfiguracija.
%Korištenje interaktivne vizualizacije za pomicanje robota i postavljanje
željenih konfiguracija.
%Kada se slika pokrene, robot se nalazi u svojoj početnoj konfiguraciji
(home configuration) s prekriženim rukama.
%Zumiranjem i lijevim klikom na određeni dio robota, dobivaju se
informacije o tom
%dijelu, a desnim klikom se taj dio može postaviti kao aktivan dio koji
%pomičemo.

%Aktivno tijelo (MarkerBody) se može postaviti i linijom koda:
iviz.MarkerBodyName = "gripper_r_base";

%Kada je aktivno tijelo odabrano, možemo ga pomicati korištenjem ponuđenih
elemenata markera.
%Povlačenjem središnjeg sivog markera pomičemo marker u kartezijskom
%prostoru. Crvene, zelene i plave osi pomiču marker po x, y i z osima.
%Krugovi rotiraju marker oko osi ekvivalentnih boja. Moguće je pomicati i
pojedinačne zglobove

%desnim klikom na zglob i odabirom ''Toggle marker control
method''(uključivanje/isključivanje metode upravljanja markerom).
%''MarkerControlMethod'' svojstvo objekta postavljeno je na "JointControl"
(upravljanje zglobom).
```

```
%Navedeni koraci mogu se postići i izravnom promjenom svojstava na objektu
odnosno aktivnom tijelu.
iviz.MarkerBodyName = "yumi_link_2_r";
iviz.MarkerControlMethod = "JointControl";

%Promjena u kontrolu zgloba daje žuti marker koji omogućuje izravno
postavljanje položaja odabranog zgloba.
%Iterativno pomicanje robota u željene konfiguracije i spremanje
konfiguracija pomoću naredbe ''addConfiguration''.
%Svaki poziv te naredbe dodaje trenutnu konfiguraciju u
''StoredConfigurations'' svojstvo.

addConfiguration(iviz)

%Definiranje međutočaka za trajektoriju.
%Za ovaj primjer dan je skup konfiguracija u .mat datoteci.
%Učitavanje konfiguracija i specificiranje istih kao skup pohranjenih
konfiguracija (''stored configurations'').
%Prva konfiguracija dodaje se ažuriranjem konfiguracijskog svojstva, što se
%može napraviti i interaktivno, a ostale konfiguracije se jednosavno dodaju
izravnim dodjeljivanjem ''StoredConfigurations'' svojstva.

load abbYumiSaveTrajectoryWaypts.mat
removeConfigurations(iviz) %Uklanjanje pohranjenih konfiguracija

%Postavljanje valjane početne konfiguracije
iviz.Configuration = startingConfig;
addConfiguration(iviz)

%Određivanje cijelog skupa međutočaka
iviz.StoredConfigurations = [startingConfig, ...
                             graspApproachConfig, ...
                             graspPoseConfig, ...
                             graspDepartConfig, ...
                             placeApproachConfig, ...
                             placeConfig, ...
                             placeDepartConfig, ...
                             startingConfig];
```

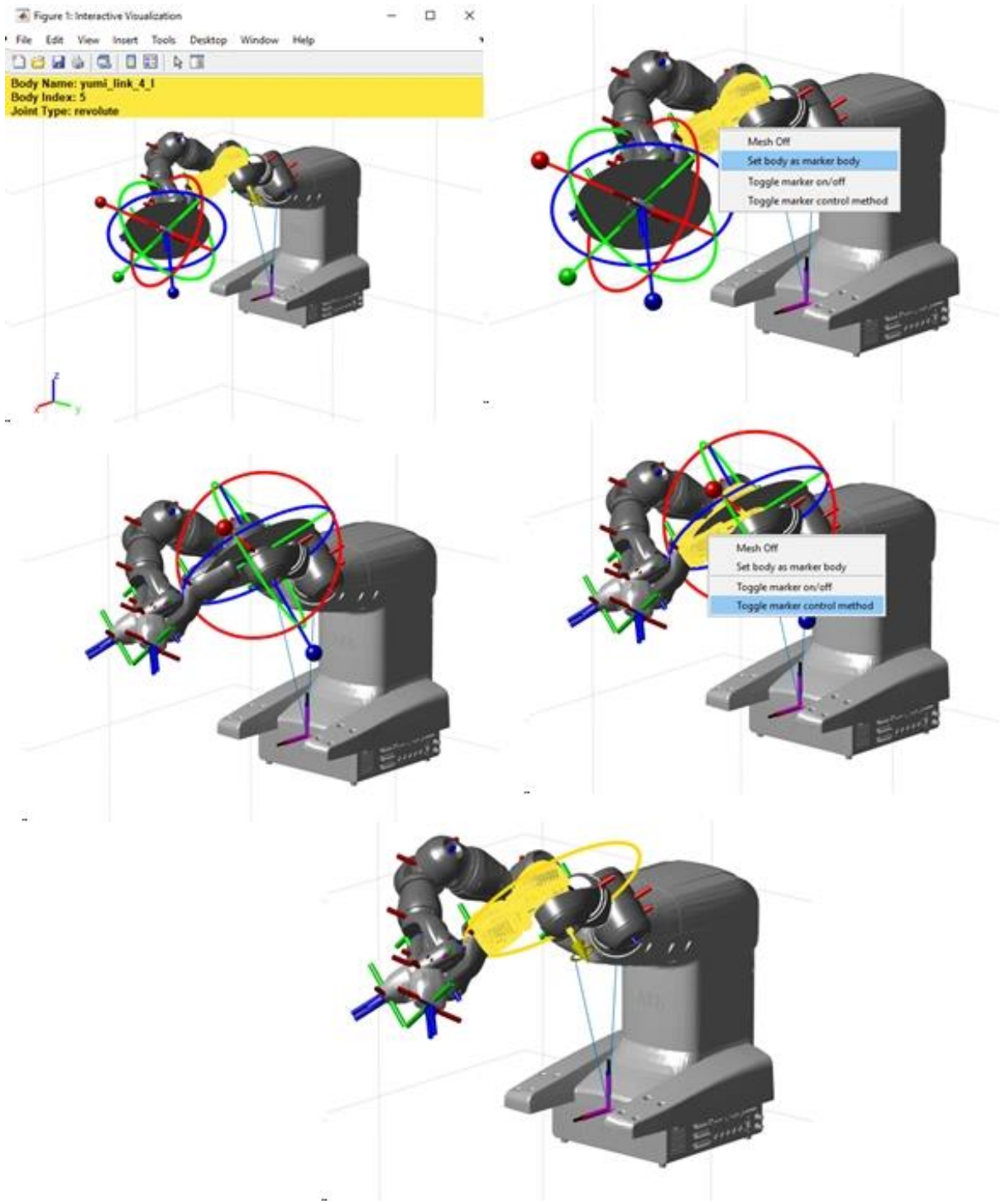
```
%Generiranje i slijeđenje trajektorije
%Nakon što su pohranjene sve međutočke, izgrađuje se trajektorija koju će
%robot slijediti. U ovom primjeru generiran je trapezoidni profil brzina
pomoću naredbe ''trapveltraj''.
%Trapezoidni profil brzina znači da robot glatko zastaje u svakoj
međutočki, ali u gibanju postiže maksimalnu brzinu .
numSamples = 100*size(iviz.StoredConfigurations, 2) + 1;
[q,~,~, tvec] =
trapveltraj(iviz.StoredConfigurations,numSamples,'EndTime',2);

%Prikaz praćenja generirane trajektorije iteriranjem generirane q matrice,
%koja predstavlja niz konfiguracija zglobova koje se gibaju između svake
međutočke. U ovom primjeru,
%korišten je ''rateControl'' objekt kako bi se osiguralo da brzina prikaza
odražava stvarnu izvedbenu brzinu kretanja robota.

iviz.ShowMarker = false;
showFigure(iviz)
rateCtrlObj = rateControl(numSamples/(max(tvec) + tvec(2)));
for i = 1:numSamples
    iviz.Configuration = q(:,i);
    waitfor(rateCtrlObj);
end
```



Slika 10. 1 Prikaz robota u različitim položajima tokom slijeđenja trajektorije

Slika **10.1** daje Prikaz robota ABB YuMi u različitim položajima tokom slijeđenja trajektorije koja je generirana pomoću interaktivnog stabla krutog tijela.

Na **Slici 10.2** je dan prikaz upravljanja interaktivnim modelom robota, odnosno odabir aktivnog tijela (Marker Body) i načina pomicanja aktivnog tijela izborom metode upravljanja markerom (pomicanje markera u kartezijskom prostoru uzduž i oko x, y i z osi, ili rotiranje pojedinog zgloba).



Slika 10. 2 Prikaz načina manipuliranja interaktivnim modelom robota

11. UBACIVANJE CAD SKLOPA (ASSEMBLY-JA) IZ SOLIDWORKSA U MATLAB

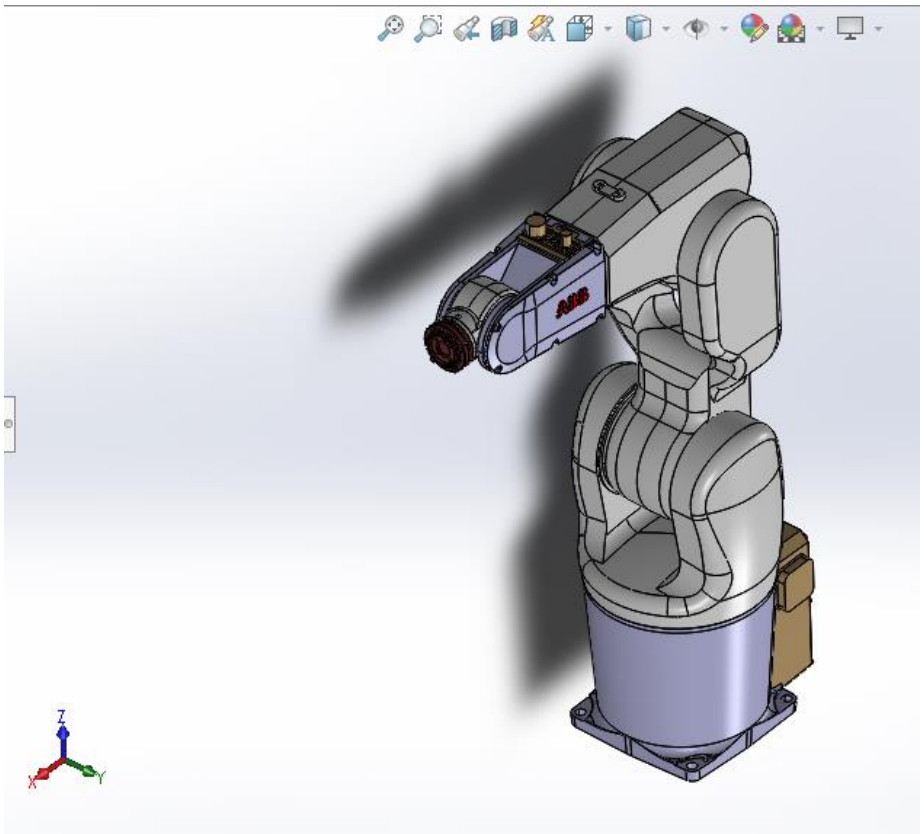
Ubacivanje CAD modela u Matlab može se izvršiti na dva načina.

Prvi način je pretvaranjem CAD sklopa u **.xml** datoteku pomoću programskog paketa **Simscape Multibody Link**, koji je potrebno instalirati za Matlab i potom povezati sa SolidWorks-om pomoću naredbe **smlink_linksw** u Matlabu, što je već ranije spomenuto. Naravno, potrebno je prethodno instalirati **Simscape Multibody** programski paket za Matlab.

Drugi način je izrada **.urdf** datoteke iz CAD sklopa u SolidWorksu, za što je potrebno instalirati **sw2urdf** programski paket za SolidWorks.

11.1 Izrada .xml datoteke iz CAD sklopa u SolidWorksu i otvaranje .xml datoteke u Matlabu

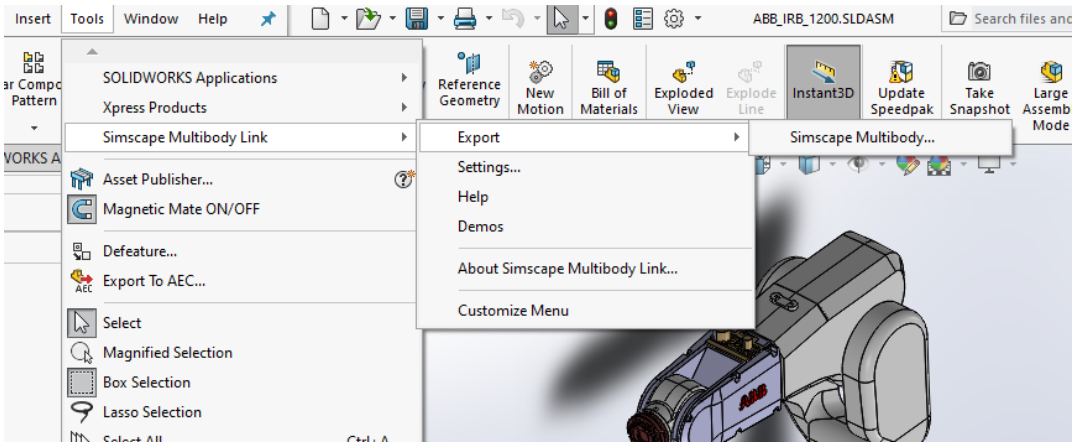
Na **Slici 11.1** je prikazan CAD model robota ABB IRB 1200-5/0.9 u SolidWorks-u, na kojem će biti prikazan i objašnjen postupak izrade .xml datoteke i otvaranje modela u Matlab-u.



Slika 11. 1 Model robota ABB IRB 1200-5/0.9 u SolidWorks-u

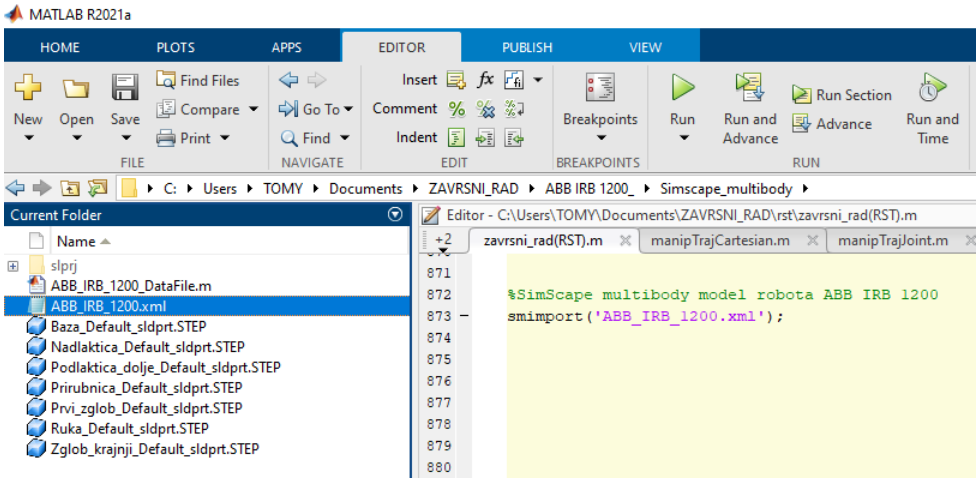
Nakon instalacije programskih paketa **Simscape Multibody** i **Simscape Multibody Link** te upisivanja naredbe **smlink_linksw** u Matlabu, pojavit će se padajući izbornik Simscape Multibody Link u SolidWorksu unutar izbornika **Tools,** kao što je prikazano na **slic**i. To omogućuje pretvorbu CAD sklopa u **.xml** datoteku klikom na Export->Simscape Multibody. Nakon klika odvija se automatizirana konverzija u **.xml** datoteku koju je potrebno spremiti u neku mapu za daljnje korištenje.

Nakon instalacije programskih paketa **Simscape Multibody** i **Simscape Multibody Link** te upisivanja naredbe **smlink_linksw** u Matlabu, pojavit će se padajući izbornik Simscape Multibody Link u SolidWorksu unutar izbornika **Tools,** kao što je prikazano na **slici 11.2**. To omogućuje pretvorbu CAD sklopa u **.xml** datoteku klikom na Export->Simscape Multibody. Nakon klika odvija se automatizirana konverzija u **.xml** datoteku koju je potrebno spremiti u neku mapu za daljnje korištenje.

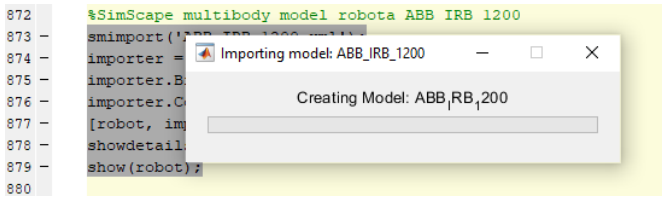


Slika 11. 2 Prikaz načina izrade .xml datoteke u SolidWorks-u

U Matlabu pomoću naredbe **smimport()** otvaramo **.xml** datoteke. Naredbi je potrebno kao argument navesti ime **.xml** datoteke, u ovom slučaju **ABB_IRB_1200.xml**. Da bi se datoteka otvorila, mapa u kojoj se nalazi mora biti postavljena kao trenutna mapa (Current Folder) što se može vidjeti na **Slici 11.3**. Pokretanjem naredbe Matlab-ov paket Simscape Multibody generira **Simulink** model robota, pokretanje je prikazano na **Slici 11.4**.

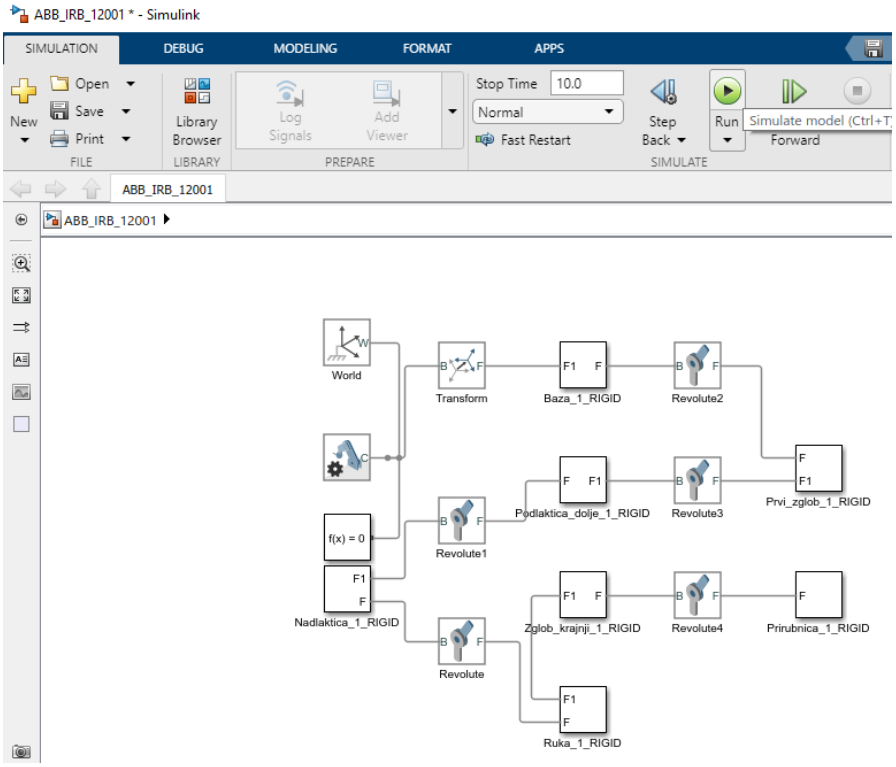


Slika 11. 3 Prikaz načina otvaranja .xml datoteke u Matlab-u



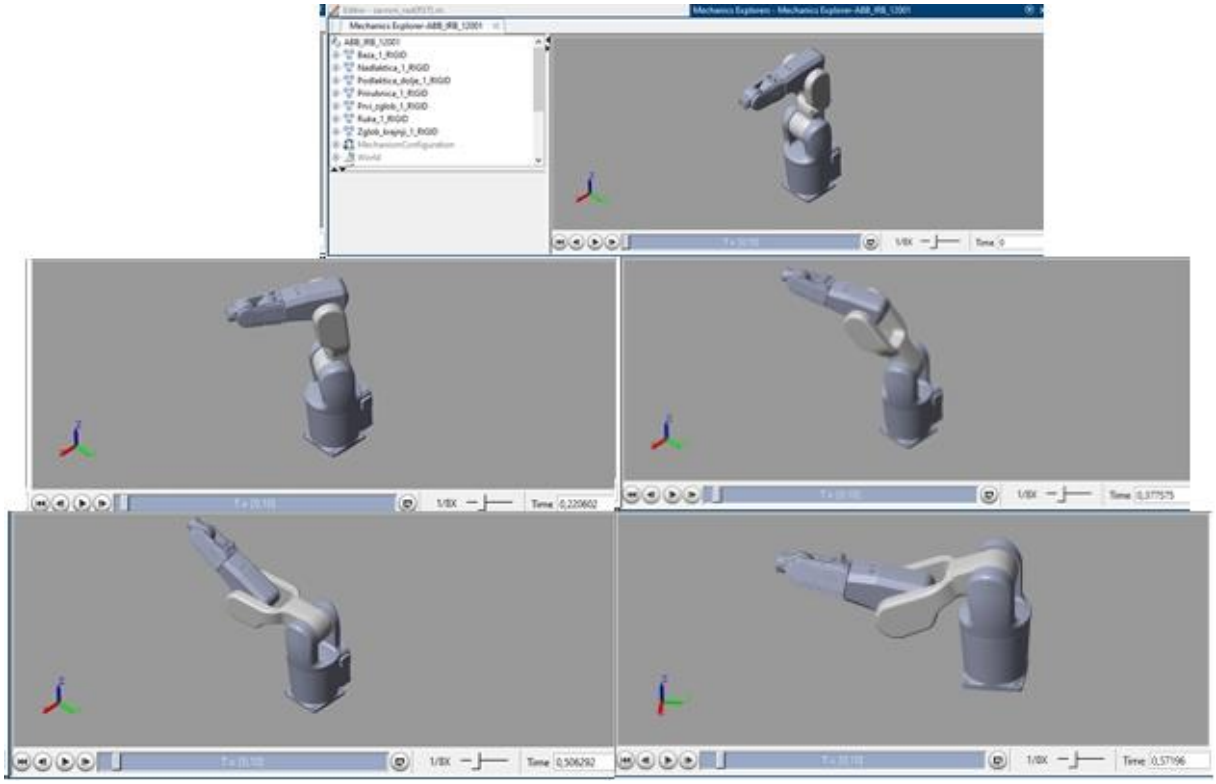
Slika 11. 4 Generiranje Simulink modela iz .xml datoteke

Slika 11.5 prikazuje izgled generiranog **Simulink** modela robota ABB IRB 1200-5/0.9.



Slika 11. 5 Prikaz Simulink modela robota ABB IRB 1200-5/0.9

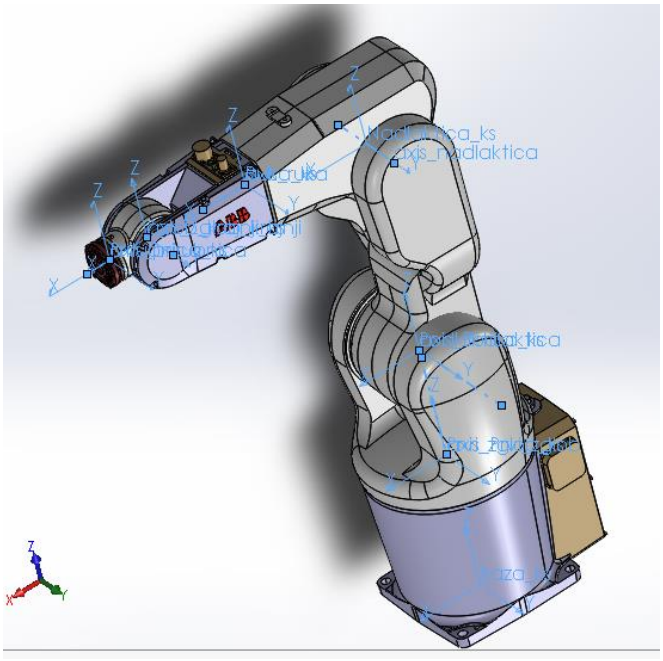
Klikom na **Run** pokrećemo automatski generiranu simulaciju robotske ruke.



Slika 11. 6 Prikaz robota u Simscape simulatoru

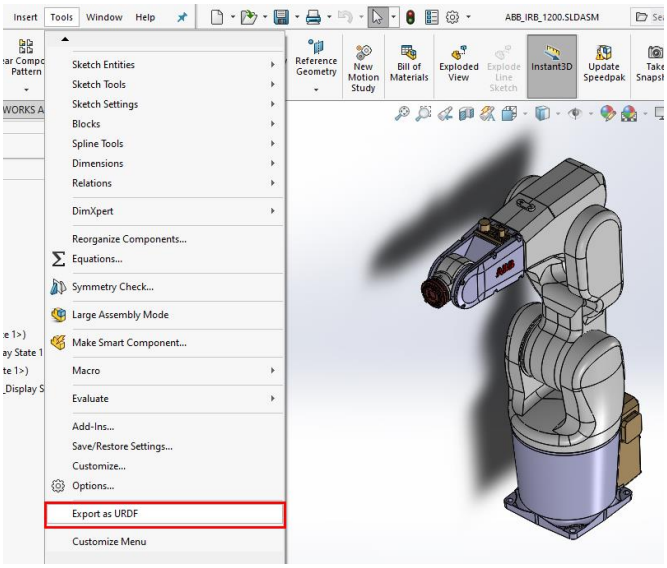
Slika 11.6 prikazuje robotsku ruku u raznim položajima tokom simulacije.

11.2 Izrada .urdf datoteke iz CAD sklopa u SolidWorksu i otvaranje .urdf datoteke u Matlabu



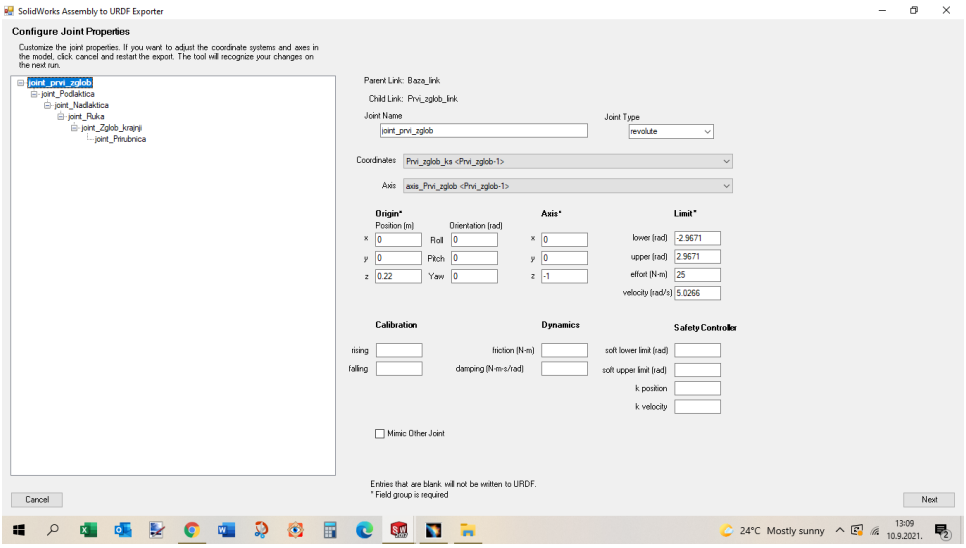
Slika 11. 7 Prikaz koordinatnih sustava i osi rotacija na modelu robota

Prije izrade urdf-a potrebno je uskladiti koordinatne sustave svih dijelova robota, dakle osi x, y i z moraju biti usklađene. Također je potrebno za svaki rotacijski zglob robota definirati os rotacije oko koje se pojedini zglob okreće. **Slika 11.7** prikazuje sve koordinatne sustave i osi rotacija zglobova robota ABB IRB 1200-5/09.

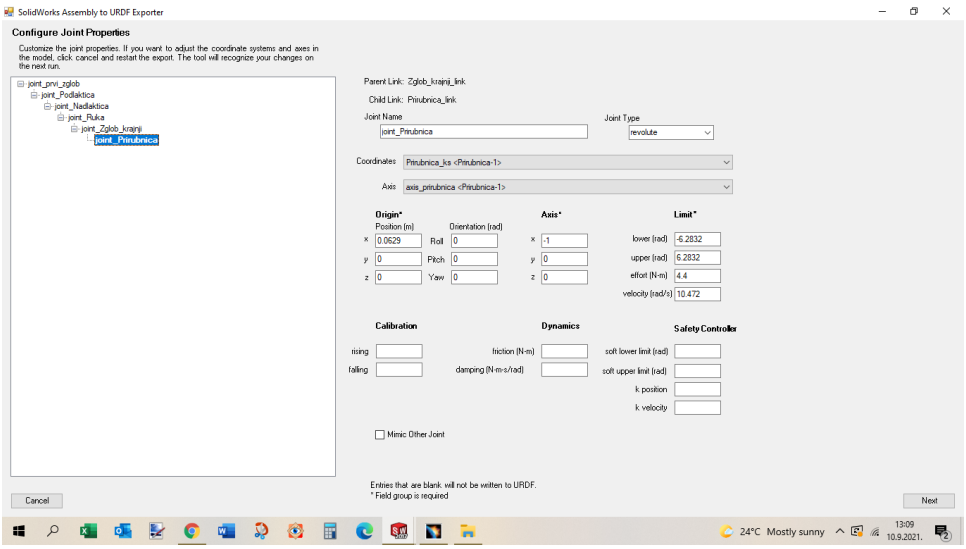


Slika 11. 8 Prikaz načina izrade .urdf datoteke u SolidWorks-u

Nakon instalacije paketa **sw2urdf** u SolidWorksu pojavit će se padajući izbornik **Export as URDF** unutar izbornika **Tools** kao što je prikazano na **Slici 11.8.** Klikom na to, započinje postupak izrade .urdf datoteke iz CAD modela robota.

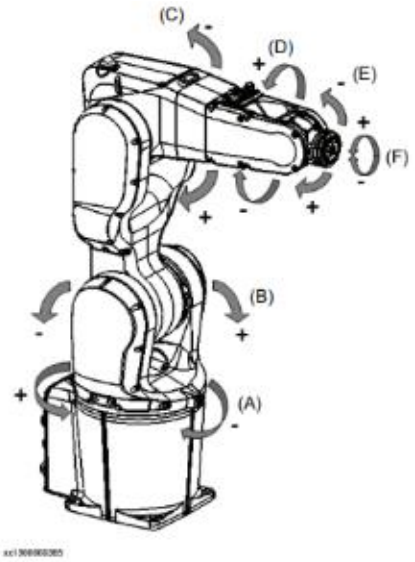


Slika 11. 9 Unošenje podataka prvog zgloba za urdf



Slika 11. 10 Unošenje podataka zadnjeg zgloba za urdf

Slike 11.9 i **11.10** prikazuju podatke koje je potrebno upisati za svaki zglob robota. Upisuju se podatci o brzinama, momentima i dozvoljenim granicama rotacije zglobova, a također se definiraju referentni koordinatni sustavi i osi rotacije za svaki zglob, koji su prikazani na **Slici 11.7.** Podatci za robota ABB IRB 1200-5/09 izvučeni su od proizvođača i prikazani na slikama **11.11** do **11.13**.



Position	Description	Position	Description
A	Axis 1	B	Axis 2
C	Axis 3	D	Axis 4
E	Axis 5	F	Axis 6

Slika 11. 11 Prikaz pojedinih zglobova robota ABB IRB 1200-5/0.9 [5]

1.8.3 Velocity

3-phase power supply

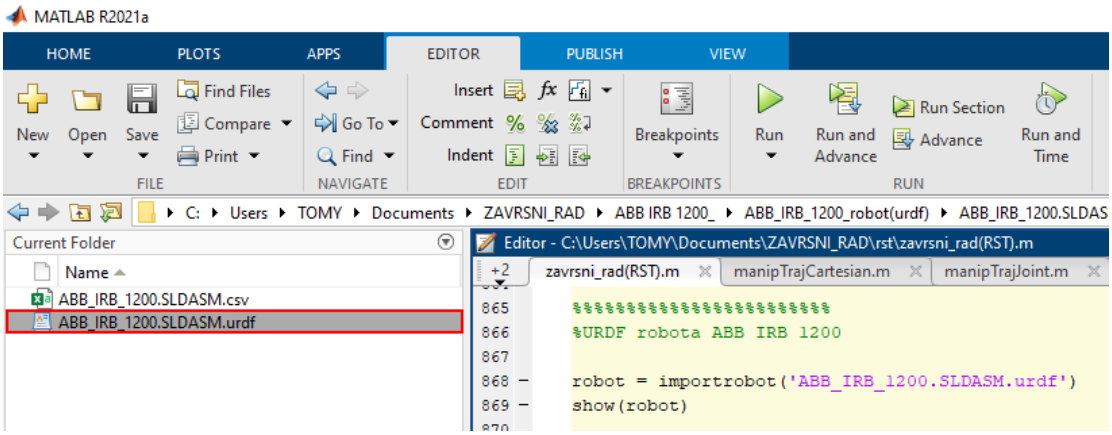
Axis number	1,200-5/0.9	1,200-7/0.7
1	288°/s	288°/s
2	240°/s	240°/s
3	297°/s	297°/s
4	400°/s	400°/s
5	405°/s	405°/s
6	600°/s	600°/s

Slika 11. 12 Prikaz podataka o brzinama zglobova robota ABB IRB 1200-5/0.9 [5]

Robot variant	Max wrist torque axis 4 and 5	Max wrist torque axis 6	Max torque valid at load
IRB 1200-7/0.7	12.5 Nm	6.2 Nm	7 kg
IRB 1200-5/0.9	8.9 Nm	4.4 Nm	5 kg

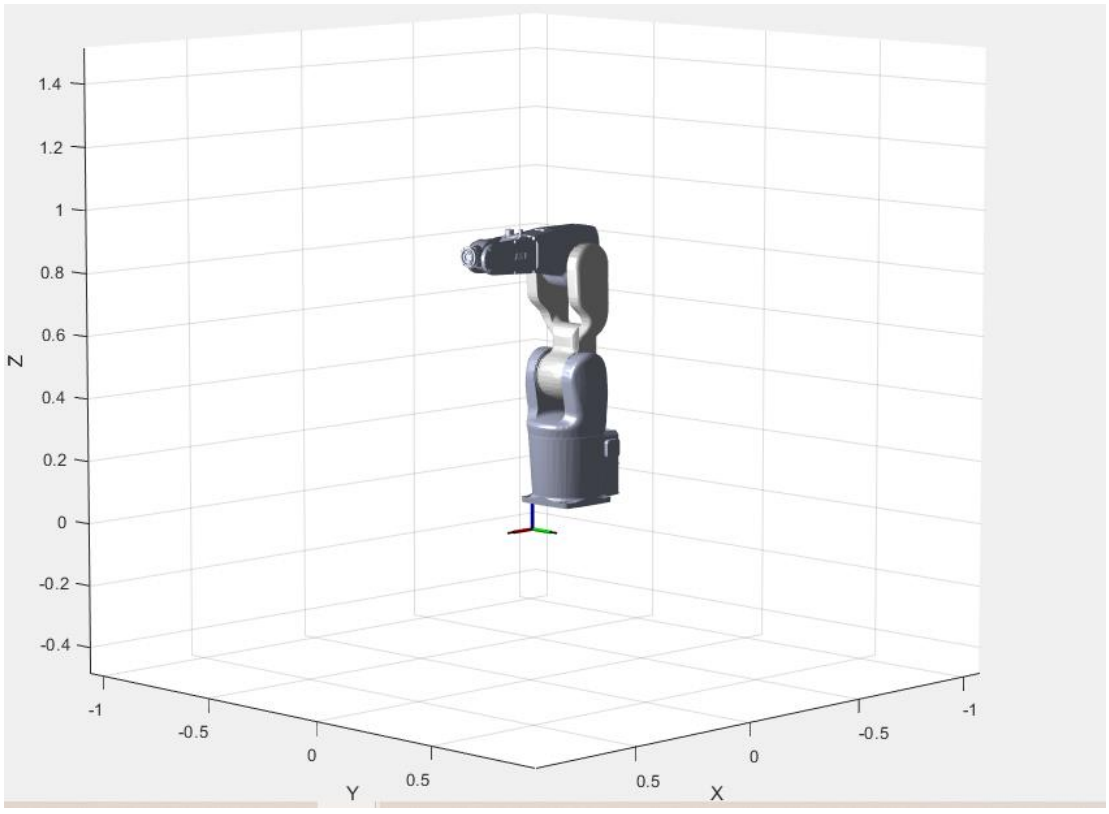
Slika 11. 13 Prikaz podataka o dozvoljenim momentima zadnja tri zgloba robota ABB IRB 1200-5/0.9 [5]

Nakon izrade urdf-a, u Matlabu možemo pomoću naredbe **importrobot()** otvoriti model robota, kao što je prikazano na **Slici 11.14**. Također je potrebno mapu u kojoj se nalazi .urdf datoteka postaviti kao trenutnu mapu kako bi Matlab pronašao urdf.



Slika 11. 14 Prikaz otvaranja .urdf datoteke u Matlab-u

Pomoću naredbe **show()** dobivamo prikaz modela robota, kao što je vidljivo na **Slici 11.15**.



Slika 11. 15 Prikaz robota ABB IRB 1200-5/0.9 iz .urdf datoteke

12. ZAKLJUČAK

Robotics System Toolbox daje korisnicima na raspolaganje veliki broj modela robota, te niz primjera u kojima su prikazane različite primjene funkcionalnosti ovog programskog paketa. RST omogućuje i napredne implementacije kao što su realne simulacije robotskih kretnji u nekom specifičnom okruženju i prijenos koda iz Matlaba preko ROS mreže na fizičke robotske sustave. Mogu se programirati robotski manipulatori, humanoidni roboti ali i mobilni roboti. Također je moguće simulacije izvoditi i s robotima koji se ne nalaze u ponudi kao gotovi modeli, putem uvoza proizvoljnih CAD modela koje korisnik može oblikovati po vlastitim željama i potrebama. To je moguće nakon instalacije programa Simscape Multibody, Simscape Multibody Link, Simulink i sw2urdf, koji čine interakcijski okvir između Matlaba i CAD programa kao što je SolidWorks.

Literatura:

[1.] https://www.mathworks.com/help/robotics/index.html?s_tid=srchtitle_Robotics%20system%20toolbox_1 (datum posjete 9. rujna 2021.)

[2.] Simscape Multibody R2021a by MathWorks (Matlab Add-On)

[3.] https://www.mathworks.com/help/robotics/ref/rigidbodytree.html#bvan8uq-8 (datum posjete 1. rujna 2021.)

[4.] https://youtu.be/HvtD1tgpC3s (datum posjete 5. rujna 2021.)

[5.] https://library.e.abb.com/public/5545a515e1c7454594132dfd715368d0/3HAC046982%20PS%20IRB%201200-en.pdf?x-sign=YaHf7XOLWpmATGlyPOWZYsvo2ynpdapDrLoWshVWoGepf2OD/2e+pnucBmuWILs9 (datum posjete 9.rujna 2021.)