Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za robotiko

Predmet

Kinematika in dinamika robotov

Navodila za laboratorijske vaje

as. dr. Sebastjan Šlajpah
sebastjan.slajpah@fe.uni-lj.si
tel: (01) 4768 229

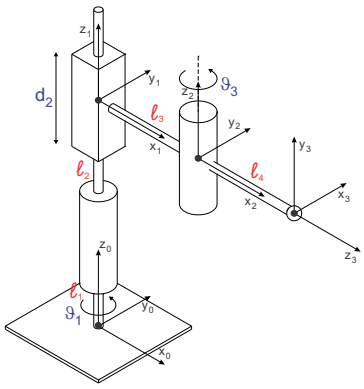
as. dr. Jure Rejc
jure.rejc@gmail.com
tel: 031 317 239

prof. dr. Matjaž Mihelj
matjaz.mihelj@fe.uni-lj.si
tel: (01) 4768 373

Kazalo

GEOMETRIJSKI MODEL ROBOTOV

Vsak robotski mehanizem je sestavljen iz robotskih segmentov, ki so povezani s sklepi. Pri serijskih (antropomorfnih) robotskih mehanizmih si robotski segmenti sledijo v zaporedju eden na drugega, poznamo pa tudi paralelne robotske mehanizme, vendar se bomo pri predmetu Kinematika in dinamika robotov posvetili le serijskim mehanizmom.



Slika 1.1: Preprost robotski mehanizem konfiguracije RTR

Robotski mehanizmi imajo lahko cilindrične in sferične sklepe, najpogosteje pa imajo rotacijske in translacijske sklepe, kot recimo robotski mehanizem na sliki 1.1. Oba najpogostejša tipa sklepov omogočata premikanje sosednjih robotskih segmentov po eni osi koordinatnega sistema, ki je postavljen v robotski sklep. Pri translacijskih sklepih je to translacija (oznaka d) po osi koordinatnega sistema, pri rotacijskih pa rotacija

(oznaka ϑ) okrog osi koordinatnega sistema. Sklepi so med seboj lahko tudi oddaljeni, kar je na sliki označeno s črko ℓ.

Omenili in pokazali (Slika 1.1) smo že, da je v vsak sklep robotskega mehanizma predstavljen s pripadajočim koordinatnim sistemom. Tudi povsem na koncu robotskega mehanizma je postavljen koordinatni sistem, katerega lega glede na bazni koordinatni sistem (x_0, y_0, z_0) je odvisna od sklepnih spremenljivk (ϑ in d) in razdalj med posameznimi sklepi označenimi s črko ℓ.

Običajno se razdalje med sklepi ne spreminjajo, spreminjajo pa se vrednosti sklepnih spremenljivk, in to takrat, ko se sklep premika (translira ali rotira). S premikanjem sklepov se spreminja tudi lega zadnjega koordinatnega sistema robotskega mehanizma glede na osi baznega koordinatnega sistema. Premikanje robotskega mehanizma, ne da bi se spraševali o vzrokih za premik in brez upoštevanja delovanja zunanjih sil imenujemo **kinematika**.

Za opis lege koordinatnega sistema v sklepu glede na prejšnji koordinatni sistem uporabljamo homogene transformacijske matrike (dimenzije 4×4), ki jih običajno označimo s črko **A**. V povezavi s skico robotskega mehanizma je mogoče določiti **direktni geometrijski model** robota s homogenimi transformacijskimi matrikami $\mathbf{A}_1 \ldots \mathbf{A}_n$, ki opisujejo medsebojno lego segmentov robota.

Za zapis geometrijskega modela robotskega mehanizma je poznanih več različnih metod, vendar se na področju robotike najpogosteje uporablja Denavit-Hartenbergova metoda oz. notacija, ki predpisuje pravila za postavitev koordinatnih sistemov v sklepe robotskega mehanizma. Metoda uporablja le štiri skalarne parametre (ϑ, d, a in α) in je iz stališča numeričnega računanja izredno kompaktna. Skalarne parametre dobljene po Denavit-Hartenbergovi metodi zapišemo v obliki tabele, ki je izhodišče za zapis homogenih transformacijskih matrik **A**. Omenjena metoda za zapis posamezne homogene transformacijske matrike **A** uporablja zaporedje, ki je opisano s transformacijami v enačbi 1.1.

$$A_i = Rot(\vartheta_i) \cdot Trans(d_i) \cdot Trans(a_i) \cdot Rot(\alpha_i) \tag{1.1}$$

Končno lego zadnjega koordinatnega sistema glede na osi baznega koordinatnega sistema opisuje homogena transformacijska matrika, ki jo označujemo s črko $\mathbf{T}_n$, kjer n predstavlja število sklepov robotskega mehanizma. Izračun matrike $\mathbf{T}_n$ ponazarja enačba 1.2.

$$T_n = A_1 \cdot A_2 \cdot ... \cdot A_n \tag{1.2}$$

1 Direktni kinematični model za robota Motoman MH5

Slika 1.2: Robot Motoman MH5

Robot Motoman MH5 (Slika 1.2) je robot namenjen delu v industriji. Je serijski mehanizem s šestimi prostostnimi stopnjami oz. aktivnimi rotacijskimi sklepi. Robotski mehanizem omogoča dva načina vodenja. Z originalnim industrijskim krmilnikom omogoča premikanje po poziciji, z namenskim odprtokodnim (razvitim v Laboratoriju za robotiko) pa po sili. Za namen vaj pri predmetu Kinematika in dinamika robotov ga bomo uporabljali za spoznavanje temeljnih znanj s področja kinematike in ob koncu vaj tudi dinamike robotskih mehanizmov.

2 Naloga

- Za narisan robotski mehanizem s pripadajočimi koordinatnimi sistemi (Slika 1.2 in 1.3) in glede na podano predlogo tabel z Denavit-Hartenbergovimi skalarnimi parametri (Tabela 1.1), na list papirja oz. zvezek prerišite tabelo in jo ustrezno zapolnite.
- Glede na zapisano tabelo v razvojnem okolju Matlab dopolnite spodaj navedene funkcije.

  `function A = hdh(`ϑ`,` d`,` a`,` α`)`, ki na podlagi metode Denavit-Hartenbergovih skalarnih parametrov ϑ, d, a in α izračuna homogeno transformacijsko matriko `A` z ustreznim zaporedjem množenja (Enačba 1.1).

  `function A = dirkinA(q)`, ki za vrednosti kotov v sklepih robota zapisanih v vektorju **q**, izračuna listo matrik `A`. Pri tem uporabite za izračun ene matrike `A` funkcijo `hdh(`ϑ`,` d`,` a`,` α`)`.

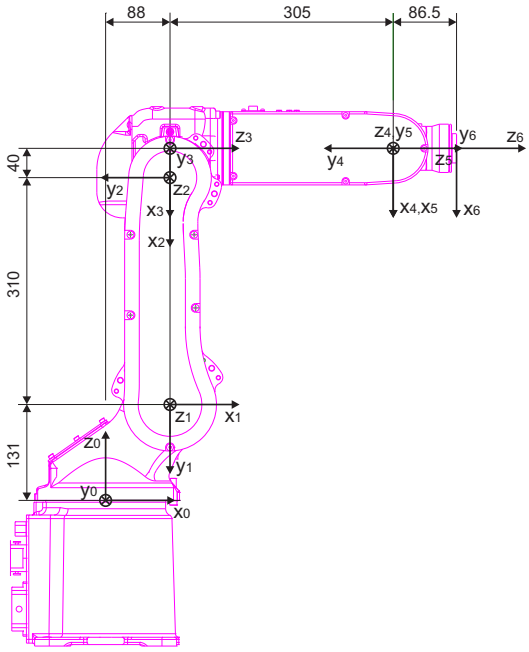
  `function T6 = dirkinT6(A)`, ki iz liste matrik `A` izračuna lego vrha robota zapisano z matriko `T6`.

  V nadaljevanju so podane predloge vseh zgoraj omenjenih funkcij. Vaša naloga je, da pravilno dopolnite funkcije na mestih, kjer je označeno s komentarjem `%%% STUDENT %%%`.

3 Potrebno predznanje, cilj in namen naloge

Cilj naloge je, da s pomočjo Denavit-Hartenbergove metode določite kinematični model robota Motoman MH5. Za izpeljavo naloge potrebujete znanje o pomenu homogenih transformacijskih matrik ter poznavanje Denavit-Hartenbergove metode. Vmesni rezultati v obliki matrik A in končne lege zadnjega koordinatnega sistema robota v obliki homogene transformacijske matrike T, boste uporabljali pri vseh nadaljnjih vajah, saj je direktna kinematika osnova za nadaljni razvoj ter poglobitev znanja s področja kinematike ter na koncu dinamike robotskih mehanizmov.

Slika 1.3 prikazuje robot Motoman MH5 v referenčni legi z označenimi koordinatnimi sistemi posameznih segmentov. Na podlagi skice je mogoče določiti direktni geometrijski model robota s homogenimi transformacijskimi matrikami $\mathbf{A}_1 \dots \mathbf{A}_6$, ki opisujejo medsebojno lego segmentov robota in jih uporabimo za zapis homogene transformacijske matrike $\mathbf{T}_6$.



Slika 1.3: Robot Motoman MH5 v referenčni legi in postavitev koordinatnih sistemov posameznih segmentov glede na Denavit-Hartenbergovo notacijo

i	ϑ_i	d_i	a_i	α_i
1				
2				
3				
4				
5				
6				

Tabela 1.1: Vzorčna tabela za zapis skalarnih parametrov po Denavit-Hartenbergovi metodi za robot Motoman MH5

4 Matlab predloge

Predloga funkcije **function A = hdh(ϑ, d, a, α)**

```
% th, d, a, al - Denavit-Hartenbergovi parametri robota (vhod) - po vrsti:
% Theta(rotz), d(transz), a(transx), Alpha(rotx).
% A - homogena transformacijska matrika (izhod).

% Kosinus kota THETA.
    ct = cos(th);
% Sinus kota THETA.
    st = sin(th);

% Matrika rotacije okrog osi z.
    rz = ;                                          %%% STUDENT %%%

% Matrika translacije vzdolz osi z in x.
    trans = ;                                       %%% STUDENT %%%

% Kosinus kota ALPHA.
    cal = cos(al);
% Sinus kota ALPHA.
    sal = sin(al);

% Matrika rotacije okrog osi x.
    rx = ;                                          %%% STUDENT %%%

% Izracun homogene transformacijske matrike.
    A = ;                                           %%% STUDENT %%%
```

Testni rezultat:

$$hdh(1,2,3,4) = \begin{bmatrix} 0.5403 & 0.5500 & -0.6368 & 1.6209 \\ 0.8415 & -0.3532 & 0.4089 & 2.5244 \\ 0 & -0.7568 & -0.6536 & 2.0000 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Predloga funkcije **function A = dirkinA(q)**

```
% q - Vektor kotnih spremenljivk za 6 sklepov (vhod).
% A(:,:,1) ... A(:,:,6) - Lista matrik A (izhod).

% Parameteri robota.
    a1 = 0.088;
    a2 = -0.310;
    a3 = -0.040;
    d1 = 0.131;
    d4 = 0.305;
    d6 = 0.0865;

% Definiranje matrik A.
    A = zeros(4,4,6);
    A(:,:,1) = ;    %%% STUDENT %%%
    A(:,:,2) = ;    %%% STUDENT %%%
    A(:,:,3) = ;    %%% STUDENT %%%
    A(:,:,4) = ;    %%% STUDENT %%%
    A(:,:,5) = ;    %%% STUDENT %%%
    A(:,:,6) = ;    %%% STUDENT %%%
```

Predloga funkcije **function T6 = dirkinT6(A)**

```
% A(:,:,1) ... A(:,:,6) - Lista matrik A (vhod).
% T6 - Matrika lege vrha robota (izhod).

T6 = ;   %%% STUDENT %%%
```

Testni rezultat:

$$dirkinT6(dirkinA([1,2,3,4,5,6])) = \begin{bmatrix} -0.0409 & 0.9693 & -0.2424 & 0.2049 \\ -0.1072 & 0.2370 & 0.9656 & 0.4353 \\ 0.9934 & 0.0655 & 0.0942 & 0.3140 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

GEOMETRIJSKA JACOBIJEVA MATRIKA

V prejšnjem poglavju oz. vaji smo izpeljali kinematični model za dva robota. Rezultat je matrika $\mathbf{T}$, ki opisuje lego koordinatnega sistema vrha robota glede na osi baznega koordinatnega sistema. Na tem mestu se je potrebno vprašati, kakšna je sprememba v $\mathbf{T}$ (sprememba pozicije $\mathbf{p}$ in orientacije $\boldsymbol{\omega}$), če se sprememba zgodi v poljubnem sklepu q_j.

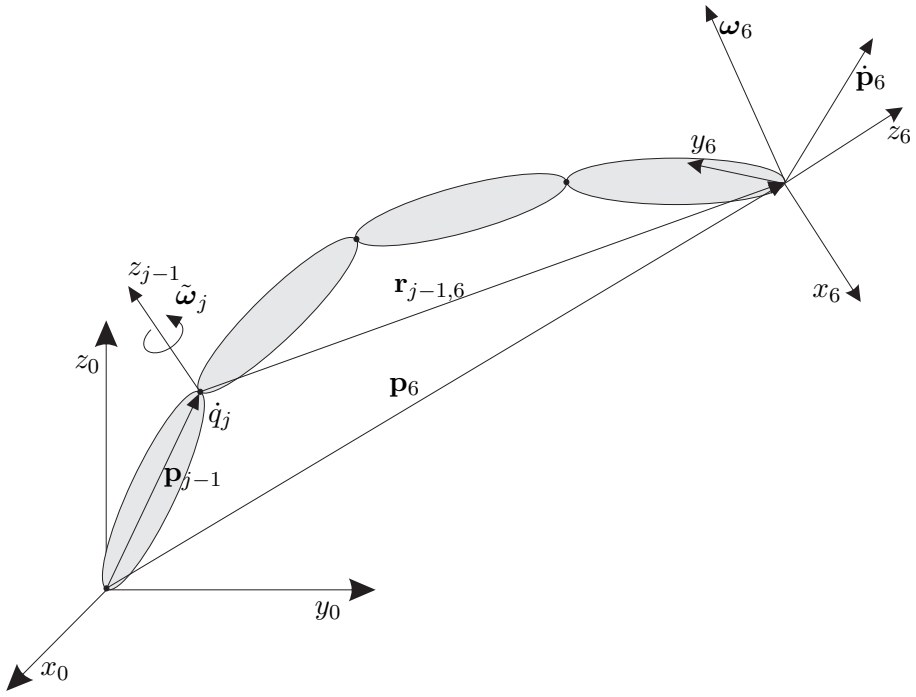
Če je časovno trajanje spremembe izredno kratko, potem lahko na situacijo gledamo kot na spremembo hitrosti v poljubnem sklepu ($\dot{q}_j$), ki se odraža v hitrosti vrha robota v obliki translacijske hitrosti $\dot{p}$ ali rotacijske hitrosti ω izražena glede na osi baznega koordinatnega sistema. Omenjeno relacijo med hitrostmi v sklepih $\dot{\mathbf{q}}$ in hitrostmi vrha robota $\dot{\mathbf{p}}$ ($\mathbf{v}$) opisuje **Jacobijeva matrika**, ki jo označujemo s črko $\mathbf{J}$ in zapišemo z enačbo 2.3.

$$\mathbf{v} = \dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{2.3}$$

1 Vpliv sklepnih hitrosti na gibanje vrha robota

Vpliv sklepne hitrosti $\dot{q}_j$ na hitrost vrha robota analiziramo glede na sliko 2.1, ki predstavlja robotski mehanizem s šestimi prostostnimi stopnjami oz. sklepi. Rotacijsko hitrost sklepa j izraženo glede na bazni koordinatni sistem zapišemo kot

$$\tilde{\boldsymbol{\omega}}_j = \mathbf{z}_{j-1}\dot{q}_j, \tag{2.4}$$



Slika 2.1: Vpliv hitrosti v sklepu j na hitrost gibanja vrha robota. Za konkretne razmere prikazane na sliki velja $j = 2$.

kjer vektor $\mathbf{z}_{j-1}$ določa os sklepa glede na bazni koordinatni sistem in $\dot{q}_j$ predstavlja skalarno hitrost vrtenja sklepa. Hitrost vrha robota, kot posledico sklepne hitrosti $\dot{q}_j$, določimo z vektorskim produktom

$$\begin{aligned}\dot{\mathbf{p}}_6(\dot{q}_j) = \tilde{\boldsymbol{\omega}}_j \times \mathbf{r}_{j-1,6} = \mathbf{z}_{j-1}\dot{q}_j \times (\mathbf{p}_6 - \mathbf{p}_{j-1}) = \\ = \mathbf{z}_{j-1} \times (\mathbf{p}_6 - \mathbf{p}_{j-1})\dot{q}_j = \mathbf{j}_{P_j}\dot{q}_j,\end{aligned} \tag{2.5}$$

kjer je $\mathbf{p}_6$ vektor, ki povezuje bazni koordinatni sistem z vrhom robota, $\mathbf{p}_{j-1}$ vektor, ki povezuje bazni koordinatni sistem s sklepom j in $\mathbf{j}_{P_j} = \mathbf{z}_{j-1} \times (\mathbf{p}_6 - \mathbf{p}_{j-1})$.

Če je sklep translacijski, je prispevek sklepa k celotni hitrosti gibanja vrha robota kar enak translacijski hitrosti sklepa

$$\dot{\mathbf{p}}_6(\dot{q}_j) = \mathbf{z}_{j-1}\dot{q}_j = \mathbf{j}_{P_j}\dot{q}_j, \tag{2.6}$$

kjer je $\mathbf{j}_{P_j} = \mathbf{z}_{j-1}$.

Rotacijska hitrost vrha robota, kot posledica sklepne hitrosti q_j, je kar enaka kotni hitrosti v sklepu, izraženi glede na bazni koordinatni sistem. Torej

$$\boldsymbol{\omega}_6(\dot{q}_j) = \tilde{\boldsymbol{\omega}}_j = \mathbf{z}_{j-1}\dot{q}_j = \mathbf{j}_{O_j}\dot{q}_j, \tag{2.7}$$

kjer je $\mathbf{j}_{O_j} = \mathbf{z}_{j-1}$.

Translacijski sklep ne spreminja orientacije vrha robota in je zato rotacijska hitrost vrha kot posledica delovanja translacijskega sklepa enaka nič

$$\boldsymbol{\omega}_6(\dot{q}_j) = \mathbf{0} = \mathbf{j}_{O_j}\dot{q}_j, \tag{2.8}$$

kjer je $\mathbf{j}_{O_j} = \mathbf{0}$.

Celotna hitrost gibanja vrha robota je določena z gibanjem v posameznih sklepih. Zato moramo prispevke gibanja posameznih sklepov na hitrost vrha robota sešteti med seboj, da bi dobili celotno hitrost vrha. Poglejmo najprej razmere za translacijske hitrosti. Ker so vse parcialne hitrosti $\dot{\mathbf{p}}_6(\dot{q}_j)$ izražene glede na osnovni koordinatni sistem, jih lahko preprosto seštejemo med seboj

$$\begin{aligned}\dot{\mathbf{p}}_6(\dot{\mathbf{q}}) &= \dot{\mathbf{p}}_6(\dot{q}_1) + \ldots + \dot{\mathbf{p}}_6(\dot{q}_j) + \ldots + \dot{\mathbf{p}}_6(\dot{q}_6) = \\ &= \mathbf{j}_{P_1}\dot{q}_1 + \ldots + \mathbf{j}_{P_j}\dot{q}_j + \ldots + \mathbf{j}_{P_6}\dot{q}_6 = \\ &= \begin{bmatrix}\mathbf{j}_{P_1} & \cdots & \mathbf{j}_{P_6}\end{bmatrix}\begin{bmatrix}\dot{q}_1 \\ \vdots \\ \dot{q}_6\end{bmatrix} = \begin{bmatrix}\mathbf{j}_{P_1} & \cdots & \mathbf{j}_{P_6}\end{bmatrix}\dot{\mathbf{q}}.\end{aligned} \tag{2.9}$$

Vidimo, da je translacijska hitrost vrha robota določena s produktom matrike, ki jo sestavljajo stolpci $\mathbf{j}_{P_j}$, in vektorja sklepnih hitrosti $\dot{\mathbf{q}}$.

Podobno lahko seštejemo tudi prispevke, ki določajo rotacijsko hitrost vrha robota

$$\begin{aligned}\boldsymbol{\omega}_6(\dot{\mathbf{q}}) &= \boldsymbol{\omega}_6(\dot{q}_1) + \ldots + \boldsymbol{\omega}_6(\dot{q}_j) + \ldots + \boldsymbol{\omega}_6(\dot{q}_6) = \\ &= \mathbf{j}_{O_1}\dot{q}_1 + \ldots + \mathbf{j}_{O_j}\dot{q}_j + \ldots + \mathbf{j}_{O_6}\dot{q}_6 = \\ &= \begin{bmatrix}\mathbf{j}_{O_1} & \cdots & \mathbf{j}_{O_6}\end{bmatrix}\begin{bmatrix}\dot{q}_1 \\ \vdots \\ \dot{q}_6\end{bmatrix} = \begin{bmatrix}\mathbf{j}_{O_1} & \cdots & \mathbf{j}_{O_6}\end{bmatrix}\dot{\mathbf{q}}.\end{aligned} \tag{2.10}$$

Rotacijska hitrost vrha robota je določena s produktom matrike, ki jo sestavljajo stolpci $\mathbf{j}_{O_j}$, in vektorja sklepnih hitrosti $\dot{\mathbf{q}}$.

2 Jacobijeva matrika

Kot je že bilo omenjeno v uvodu poglavja, Jacobijeva matrika določa relacijo med hitrostmi vrha robota $\mathbf{v} = \begin{bmatrix}\dot{\mathbf{p}}_6 & \boldsymbol{\omega}_6\end{bmatrix}^T$ in hitrostmi v sklepih $\dot{\mathbf{q}}$, kar smo zapisali z enačbo 2.3 oziroma lahko zapišemo kot

$$\begin{bmatrix}\dot{\mathbf{p}}_6 \\ \boldsymbol{\omega}_6\end{bmatrix} = \begin{bmatrix}\mathbf{J}_P \\ \mathbf{J}_O\end{bmatrix} \dot{\mathbf{q}}. \tag{2.11}$$

V zgornjem izrazu smo Jacobijevo matriko robota smo **razdelili na translacijski in rotacijski del**

$$\mathbf{J}^{6\times 6} = \begin{bmatrix}\mathbf{J}_P^{3\times 6} \\ \mathbf{J}_O^{3\times 6}\end{bmatrix}. \tag{2.12}$$

Enačbi (2.9) in (2.10) uporabimo za izračun geometrijske Jacobijeve matrike robota. Posamezen stolpec v Jacobijevi matriki določa kako posamezna sklepna hitrost vpliva na hitrost premikanja vrha robota.

3 Translacijska Jacobijeva podmatrika

Najprej iz izraza (2.9) določimo Jacobijevo podmatriko za izračun translacijskih hitrosti vrha robota

$$\mathbf{J}_P = \begin{bmatrix}\mathbf{j}_{P_1} & \dots & \mathbf{j}_{P_j} & \dots & \mathbf{j}_{P_6}\end{bmatrix}^{3\times 6}. \tag{2.13}$$

Stolpec Jacobijeve matrike $\mathbf{j}_{P_j}$ opisuje kako hitrost v sklepu j vpliva na translacijsko hitrost vrha robota.

Torej lahko zapišemo

$$\mathbf{j}_{P_j} = \begin{cases}\mathbf{z}_{j-1} & \text{za translacijski sklep} \\ \\ \mathbf{z}_{j-1} \times (\mathbf{p}_6 - \mathbf{p}_{j-1}) & \text{za rotacijski sklep}\end{cases}, \tag{2.14}$$

kar izhaja iz enačb (2.5) in (2.6). Vektor $\mathbf{z}_{j-1}$ je določen kot **tretji stolpec rotacijske matrike** $\mathbf{R}_{j-1}$ (podmatrika matrike $\mathbf{A}_{j-1}$), torej

$$\mathbf{z}_{j-1} = \mathbf{R}_1(q_1) \dots \mathbf{R}_{j-1}(q_{j-1})\mathbf{z}_0\,, \tag{2.15}$$

kjer je $\mathbf{z}_0 = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. Vektor $\mathbf{p}_6$ določa položaj vrha robota glede na bazni koordinatni sistem in je torej določen s prvimi tremi elementi

četrtega stolpca matrike

$$\mathbf{T}_6 = \mathbf{A}_1(q_1) \dots \mathbf{A}_6(q_6). \tag{2.16}$$

Vektor $\mathbf{p}_{j-1}$ je določen s prvimi tremi elementi četrtega stolpca transformacijske matrike $\mathbf{T}_{j-1} = \mathbf{A}_1(q_1) \dots \mathbf{A}_{j-1}(q_{j-1})$; $\mathbf{p}_0 = \mathbf{0}^{3\times 1}$.

4 Rotacijska Jacobijeva podmatrika

$\mathbf{J}_O$ je Jacobijeva podmatrika za izračun rotacijskih hitrosti vrha robota izraženih glede na bazni koordinatni sistem. Iz izraza (2.10) dobimo

$$\mathbf{J}_O = \begin{bmatrix} \mathbf{j}_{O_1} & \cdots & \mathbf{j}_{O_j} & \cdots & \mathbf{j}_{O_6} \end{bmatrix}^{3\times 6}, \tag{2.17}$$

kjer je

$$\mathbf{j}_{O_j} = \begin{cases} \mathbf{0} & \text{za translacijski sklep} \\ \\ \mathbf{z}_{j-1} & \text{za rotacijski sklep.} \end{cases} \tag{2.18}$$

Pri tem smo upoštevali, da translacijski sklep ne spreminja orientacije vrha robota in je zato rotacijska hitrost vrha kot posledica delovanja translacijskega sklepa enaka nič. Rotacijski sklep pa povzroča rotacijo vrha robota, kot je določeno v enačbi (2.7). Vektor $\mathbf{z}_{j-1}$ je določen v enačbi (2.15).

5 Naloga

- V Matlabu dopolnite funkcijo `function J = jacobi0(q)` za izračun geometrijske Jacobijeve matrike za robot Motoman MH5. Uporabite funkcije iz prvega poglavja. Vhodni parameter je vektor spremenljivk kotov sklepov robota **q**, izhod pa geometrijska Jacobijeva matrika robota **J**. Vaša naloga je, da pravilno dopolnite funkcijo na mestih, kjer je označeno s komentarjem `%%% STUDENT %%%`.
- V simulacijskem okolju Matlab Simulink zgradite shemo vodenja z uporabo geometrijske Jacobijeve matrike. Shema in detajlna razlaga sta podani na koncu poglavja.

6 Potrebno predznanje, cilj in namen naloge

Cilj naloge je, da se seznanite z vlogo geometrijske Jacobijeve matrike v robotiki. Izpolnjeno Matlab funkcijo boste v simulacijskem okolju Matlab Simulink uporabili in rezultat testirali na realnem robotu Motoman MH5. Rezultat poizkusa je, kako definirana hitrost vrha robot glede na osi baznega koordinatnega sistema, vpliva na hitrosti v sklepih. Dejansko se bo robotska roka premikala glede na podano hitrost vrha, in to v smeri definirane hitrosti.

Potrebno znanje pri nalogi zajema pisanje oz. dopolnjevanje Matlab predloge v .m datoteki in poznavanje simulacijskega okolja Matlab Simulink.

7 Matlab predloga

Predloga funkcije **function J = jacobi0(q)**

```
% q - Vrednost kotov v sklepih robota (vhod).
% J - Geometrijska Jacobijeva matrika (izhod).

% Izracun transformacijskih matrik A.
    A = ;                                                   %%% STUDENT %%%

% Transformacijska matrika lege vrha T6.
    T6 = ;                                                  %%% STUDENT %%%

% Inicializacija prazne Jacobijeve matrike.
    Jp = zeros(3,6);    % Pozicijska podmatrika.
    Jo = zeros(3,6);    % Orientacijska podmatrika.

% Inicializacija spremenljivk.
    z_0 = [0,0,1]';     % Vektor v smeri osi z_0.
    p_0 = [0,0,0]';     % Vektor polozaja prvega sklepa glede na ref. k.s.
    Tn = eye(4);        % Zacetna vrednost delne matrike Tn; n = 1,2...

% 1. STOLPEC JACOBIJEVE PODMATRIKE
    % Izracun 1. stolpca pozicijske Jacobijeve podmatrike.
    Jp(:,1) =                                               %%% STUDENT %%%
    % Izracun 1. stolpca rotacijske Jacobijeve podmatrike.
    Jo(:,1) =                                               %%% STUDENT %%%

% 2. STOLPEC JACOBIJEVE PODMATRIKE
    % Izracun matrike Tn, ki vsebuje informacijo Zj-1 in Pj-1.
    Tn =                                                    %%% STUDENT %%%
    % Izracun 2. stolpca pozicijske Jacobijeve podmatrike.
    Jp(:,2) =                                               %%% STUDENT %%%
    % Izracun 2. stolpca rotacijske Jacobijeve podmatrike.
    Jo(:,2) =                                               %%% STUDENT %%%
```

```
    % 3. STOLPEC JACOBIJEVE PODMATRIKE
        % Izracun matrike Tn, ki vsebuje informacijo Zj-1 in Pj-1.
        Tn =                                                %%% STUDENT %%%
        % Izracun 3. stolpca pozicijske Jacobijeve podmatrike.
        Jp(:,3) =                                               %%% STUDENT %%%
        % Izracun 3. stolpca rotacijske Jacobijeve podmatrike.
        Jo(:,3) =                                           %%% STUDENT %%%

    % 4. STOLPEC JACOBIJEVE PODMATRIKE
        % Izracun matrike Tn, ki vsebuje informacijo Zj-1 in Pj-1.
        Tn =                                                %%% STUDENT %%%
        % Izracun 4. stolpca pozicijske Jacobijeve podmatrike.
        Jp(:,4) =                                           %%% STUDENT %%%
        % Izracun 4. stolpca rotacijske Jacobijeve podmatrike.
        Jo(:,4) =                                           %%% STUDENT %%%

    % 5. STOLPEC JACOBIJEVE PODMATRIKE
        % Izracun matrike Tn, ki vsebuje informacijo Zj-1 in Pj-1.
        Tn =                                                %%% STUDENT %%%
        % Izracun 5. stolpca pozicijske Jacobijeve podmatrike.
        Jp(:,5) =                                           %%% STUDENT %%%
        % Izracun 5. stolpca rotacijske Jacobijeve podmatrike.
        Jo(:,5) =                                           %%% STUDENT %%%

    % 6. STOLPEC JACOBIJEVE PODMATRIKE
        % Izracun matrike Tn, ki vsebuje informacijo Zj-1 in Pj-1.
        Tn =                                                %%% STUDENT %%%
        % Izracun 6. stolpca pozicijske Jacobijeve podmatrike.
        Jp(:,6) =                                           %%% STUDENT %%%
        % Izracun 6. stolpca rotacijske Jacobijeve podmatrike.
        Jo(:,6) =                                           %%% STUDENT %%%

    % Celotna Jacobijeva matrika je sestavljena iz obeh podmatrik.
        J = [Jp; Jo];
```

Testni rezultat:

$$jacobi0([1,2,3,4,5,6]) = \begin{bmatrix} -0.4353 & 0.0989 & 0.1686 & -0.0781 & 0.0200 & -0.0000 \\ 0.2049 & 0.1540 & 0.2625 & -0.0214 & -0.0032 & -0.0000 \\ 0 & -0.3890 & -0.1071 & 0.0178 & 0.0841 & 0.0000 \\ 0 & -0.8415 & -0.8415 & 0.1533 & 0.9421 & -0.2424 \\ 0 & 0.5403 & 0.5403 & 0.2387 & 0.2575 & 0.9656 \\ 1.0000 & 0.0000 & 0.0000 & 0.9589 & -0.2147 & 0.0942 \end{bmatrix}$$

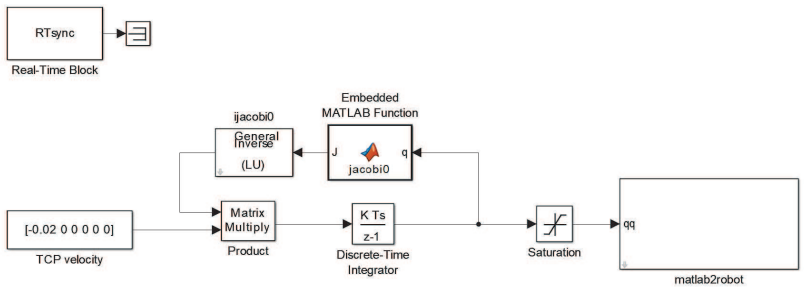
8 Simulacijska shema v Matlab Simulink okolju

V simulacijskem okolju Matlab Simulinku zgradite simulacijsko shemo s slike 2.2. Blok *TCP velocity* predstavlja željeno hitrost vrha v zunanjih koordinatah $\dot{\mathbf{x}} = \begin{bmatrix} -0.02 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$. Blok *MATLAB Function - Jacobi0* glede na vektor notranjih spremenljivk ($\mathtt{q}$) poda izhod v obliki geometrijske Jacobijeve matrike $\mathbf{J}(\mathbf{q})$. V ta blok je potrebno prekopirati celotno funkcijo *jacobi0* s prejšnje strani in tudi vse funkcije odvisne funkcije (*hdh, dirkinA, dirkinT6*) oz. znotraj *Embedded funkcije* klicati funkcijo znotraj .m datoteke. Blok *GeneralInverse* invertira geometrijsko Jacobijevo matriko, nato pa sledi matrično množenje željeno hitrostjo vrha robota ($\dot{\mathbf{x}}$); enote so m/s. Simulacijska shema se preračunava s kon-

stantnim časovnim korakom $dT = 2$ ms. Simulacijsko shemo ponazarja enačba 2.19.

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{x}} \tag{2.19}$$

Za odpiranje predloge Simulink sheme v Matlab ukazni vrstici zaženite program *initVR.m*!
`>> initVR`
Spremenjeno shemo shranite pod drugim imenom!

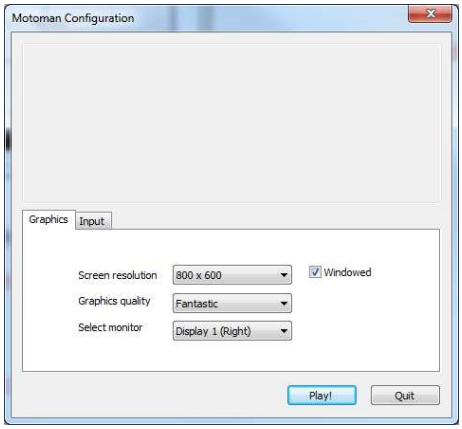


Slika 2.2: Simulacijska shema v Simulinku.

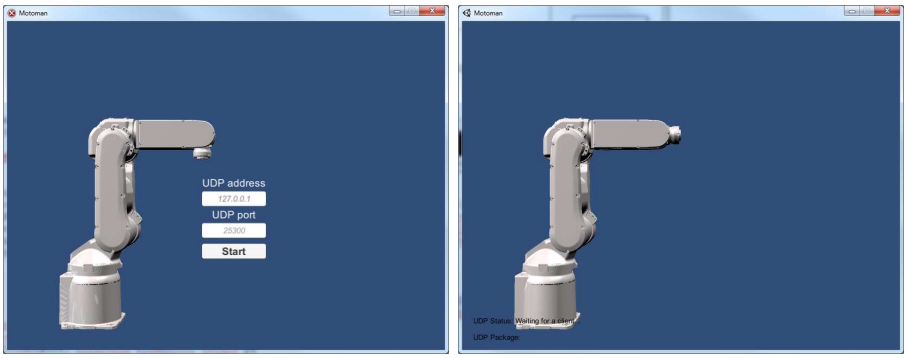
Kje se nahajajo posamezni Simulink bloki?
Descrete-Time Integrator ⇒ *Simulink* → *Discrete*
Gain value: 1.0; Initial condition: q0; Sample time: dT
Matrix Multiply ⇒ *Simulink* → *Math Operations*
Embadded MATLAB Function ⇒ *Simulink* → *User-Defined Func.*
LU Inverse ⇒ *DSP System Toolbox* → *Math Functions*
→ *Matrices and Linear Algebra* → *Matrix Inverse*

Poleg Simulink sheme se odpre tudi okno za vizualizacijo premika robota, vendar najprej okno za določanje nastavitev (Slika 2.3). Priporočljiva resolucija je **800×600** slikovnih elementov, ostalo pa ne potrebuje spremembe. Nadaljujemo s klikom na gumb **Play!**.

Odpre se okno (Slika 2.4 levo). Nastavitev ne spreminjamo. S klikom na gumb **Start** nadaljujemo naprej do zadnjega okna, kjer se bo izvajala vizualizacija premika robota (Slika 2.4 desno).



Slika 2.3: Vizualizacijsko okno za določanje nastavitev.

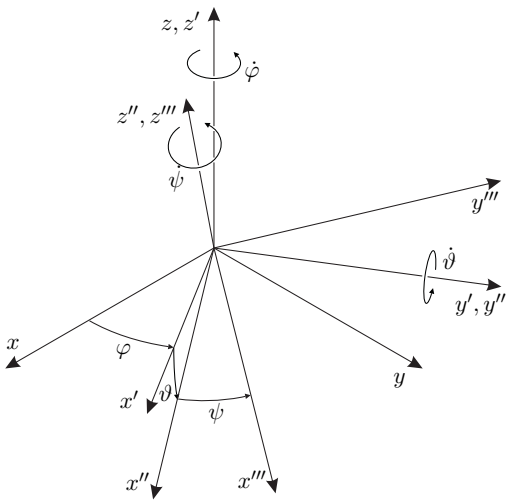


Slika 2.4: Vizualizacijsko okno za določanje nastavitev mrežne povezave.

RELACIJA MED GEOMETRIJSKO IN ANALITIČNO JACOBIJEVO MATRIKO TER INVERZNA KINEMATIKA Z UPORABO ANALITIČNE JACOBIJEVE MATRIKE

Rotacijska matrika oz. 3×3 podmatrika homogene transformacijske matrike (lahko matrika z oznako T) podaja redundanten opis orientacije nekega koordinatnega sistema glede na referenčni koordinatni sistem. Orientacija objekta v prostoru je določena s tremi parametri, rotacijska matrika pa jih vsebuje devet.

Minimalno predstavitev orientacije s tremi parametri lahko izvedemo z uporabo treh kotov $\boldsymbol{\phi} = \begin{bmatrix} \varphi & \vartheta & \psi \end{bmatrix}$.



Slika 3.1: Predstavitev orientacije z ZYZ Eulerjevimi koti

Predpostavimo rotacijsko matriko $\mathbf{R}_z(\varphi)$, ki opisuje rotacijo okrog ene osi (recimo os z) kot funkcijo enega izmed kotov (recimo φ). Splošno rotacijo okrog treh osi torej dobimo s kombinacijo treh zaporednih rotacij, pri čemer moramo biti pozorni na to, da dveh zaporednih rotacij ne izvedemo okrog dveh paralelnih osi. Da opišemo orientacijo objekta v prostoru, lahko uporabimo 12 različnih kombinacij treh osnovnih rotacij okrog posameznih osi (na primer kombinacija ZYZ pomeni, da najprej izvedemo rotacijo okrog osi z, nato okrog osi y že rotiranega koordinatnega sistema in na koncu še okrog osi z predhodno dvakrat rotiranega koordinatnega sistema; razmere so prikazane na sliki 3.1). Vsako takšno zaporedje rotacij predstavlja trojček Eulerjevih kotov.

1 *ZYX* Eulerjevi koti

Za demonstracijo trojčka Eulerjevih kotov bomo uporabili kombinacijo rotacij ZYX, ki je definirana z zaporedjem naslednjih osnovnih rotacij:

- Rotacija osnovnega (bazni, referenčni) koordinatnega sistema za kot φ okrog osi z. Rotacijo opišemo z rotacijsko matriko $\mathbf{R}_z(\varphi)$.

- Rotacija trenutnega (lokalni, relativni) koordinatnega sistema za kot ϑ okrog osi y'. Rotacijo opišemo z rotacijsko matriko $\mathbf{R}_{y'}(\vartheta)$.

- Rotacija trenutnega (lokalni, relativni) koordinatnega sistema za kot ψ okrog osi x''. Rotacijo opišemo z rotacijsko matriko $\mathbf{R}_{x''}(\psi)$.

Rezultirajočo celotno rotacijo dobimo z zaporedjem rotacij okrog trenutnega koordinatnega sistema, torej pri množenju matrik uporabimo postmultiplikacijo (Enačba 3.25).

$$\begin{aligned}\mathbf{R}(\boldsymbol{\phi}) &= \mathbf{R}_z(\varphi)\mathbf{R}_{y'}(\vartheta)\mathbf{R}_{x''}(\psi) \\ &= \begin{bmatrix} c_\varphi c_\vartheta & c_\varphi s_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta c_\psi + s_\varphi s_\psi \\ s_\varphi c_\vartheta & s_\varphi s_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta c_\psi - c_\varphi s_\psi \\ -s_\vartheta & c_\vartheta s_\psi & c_\vartheta c_\psi \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\end{aligned} \tag{3.20}$$

Če poznamo elemente (r_{xy}) matrike $\mathbf{R}(\boldsymbol{\phi})$, lahko iz nje izračunamo Eulerjeve kote. S primerjavo elementov matrik v enačbi (3.25) lahko

izračunamo tri Eulerjeve kote. Ob predpostavki, da $r_{11} \neq 0$ in $r_{21} \neq 0$ velja

$$\varphi = atan2(r_{21}, r_{11}), \tag{3.21}$$

kjer je $atan2$ štiri-kvadrantna arkustangens funkcija. S kvadriranjem in seštevanjem elementov r_{32} in r_{33} ter upoštevaje r_{31} dobimo

$$\vartheta = atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}). \tag{3.22}$$

Izbira pozitivnega predznaka za izraz $\sqrt{r_{32}^2 + r_{33}^2}$ omejuje vrednosti kota ϑ na $(0, \pi)$. Kot ψ je mogoče določiti na osnovi enačbe

$$\psi = atan2(r_{32}, r_{33}). \tag{3.23}$$

Iz zapisanega sledi, da lahko trojček ZYX Eulerjevih kotov iz splošne rotacijske matrike oz. 3×3 podmatrike homogene transformacijske matrike, ki opisuje neko lego, določimo na podlagi sledečih enačb

$$\begin{aligned} \varphi &= atan2(r_{21}, r_{11}) \\ \vartheta &= atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \psi &= atan2(r_{32}, r_{33}). \end{aligned} \tag{3.24}$$

2 *ZYZ* Eulerjevi koti

Podobno razmišljanje lahko uporabimo tudi za izpeljavo enačb za kombinacijo rotacij ZYZ. Rezultirajoča celotna rotacija je tako

$$\begin{aligned} \mathbf{R}(\boldsymbol{\phi}) &= \mathbf{R}_z(\varphi)\mathbf{R}_{y'}(\vartheta)\mathbf{R}_{z''}(\psi) \\ &= \begin{bmatrix} c_\vartheta c_\varphi c_\psi - s_\varphi s_\psi & -c_\psi s_\varphi - c_\vartheta c_\varphi s_\psi & c_\varphi s_\vartheta \\ c_\varphi s_\psi + c_\vartheta c_\psi s_\varphi & c_\varphi c_\psi - c_\vartheta s_\varphi s_\psi & s_\vartheta s_\varphi \\ -c_\psi s_\vartheta & s_\vartheta s_\psi & c_\vartheta \end{bmatrix} \end{aligned} \tag{3.25}$$

Trojček ZYZ Eulerjevih kotov iz splošne rotacijske matrike določimo na podlagi sledečih enačb

$$\begin{aligned} \varphi &= atan2(r_{23}, r_{13}) \\ \vartheta &= atan2(\sqrt{r_{13}^2 + r_{23}^2}, r_{33}) \\ \psi &= atan2(r_{32}, -r_{31}). \end{aligned} \tag{3.26}$$

3 Transformacijska matrika

Spomnimo, da lahko lego objekta v prostoru z minimalnim številom parametrov zapišemo s šestimi spremenljivkami, kjer prve tri določajo pozicijo in tvorijo vektor $\mathbf{p}$, druge tri pa določajo orientacijo in tvorijo vektor $\boldsymbol{\phi}$

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \boldsymbol{\phi} \end{bmatrix}. \tag{3.27}$$

Geometrijska Jacobijeva matrika določa geometrijske relacije med hitrostmi v sklepih in hitrostjo vrha robota glede na osnovni oz. bazni koordinatni sistem. Pri opisu lege vrha robota z minimalnim številom parametrov $\mathbf{x}$ pa obstaja možnost izračuna Jacobijeve matrike z odvajanjem elementov vektorja $\mathbf{x}$ po sklepnih spremenljivkah

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}_A(\mathbf{q}) \dot{\mathbf{q}}, \tag{3.28}$$

torej lahko izračunamo translacijsko hitrost vrha robota kot

$$\dot{\mathbf{p}} = \frac{\partial \mathbf{p}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}_p(\mathbf{q}) \dot{\mathbf{q}} \tag{3.29}$$

in rotacijsko hitrost vrha robota kot

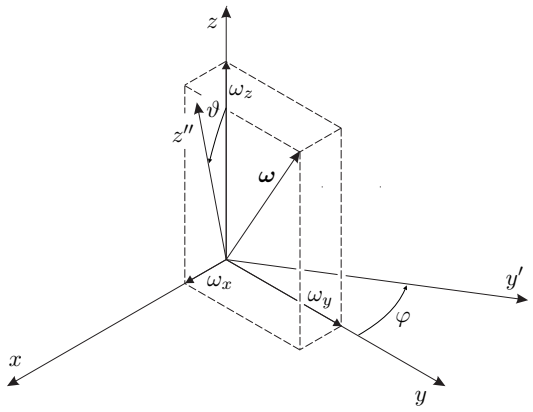
$$\dot{\boldsymbol{\phi}} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}_\phi(\mathbf{q}) \dot{\mathbf{q}}, \tag{3.30}$$

pri čemer vektor $\boldsymbol{\phi}$ določa trojček Eulerjevih kotov. Upoštevaje translacijo in rotacijo lahko torej zapišemo

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\boldsymbol{\phi}} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_\phi(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}_A(\mathbf{q}) \dot{\mathbf{q}}. \tag{3.31}$$

V nadaljevanju bomo poiskali transformacijsko matriko, ki povezuje geometrijsko Jacobijevo matriko z analitično Jacobijevo matriko določeno na osnovi ZYX Eulerjevih kotov. V ta namen je potrebno poiskati relacije med rotacijskimi hitrostmi $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ in hitrostmi ZYX Eulerjevih kotov $\dot{\boldsymbol{\phi}}$ (Slika 3.2). Na sliki 3.3 je primer hitrosti ZYZ Eulerjevih kotov, kjer so hitrosti $\dot{\varphi}$, $\dot{\vartheta}$ in $\dot{\psi}$ predstavljene glede na osi trenutnega koordinatnega sistema. Slika tudi prikazuje postopek izračuna doprinosa posamezne hitrosti k hitrostim izraženim glede na osi osnovnega koordinatnega sistema. Za hitrosti ZYX Eulerjevih kotov:

- kot rezultat hitrosti $\dot{\varphi}$: $\begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T = \dot{\varphi} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$; upoštevali smo, da je na začetku os z trenutnega koordinatnega sistema poravnana z osjo z osnovnega koordinatnega sistema,



Slika 3.2: Razstavitev vektorja hitrosti v rotacijske hitrosti v osnovnem koordinatnem sistemu.

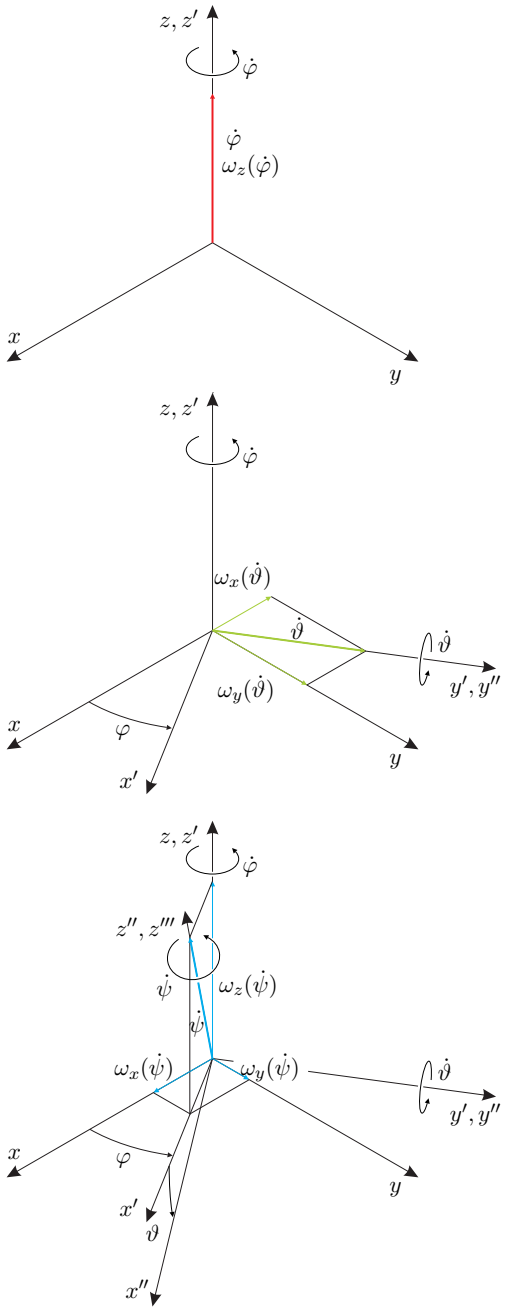
- kot rezultat hitrosti $\dot{\vartheta}$: $\begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T = \dot{\vartheta}\begin{bmatrix}-s_\varphi & c_\varphi & 0\end{bmatrix}^T$; upoštevali smo, da je med osnovnim in trenutnim koordinatnim sistemom zasuk φ okrog osi z osnovnega koordinatnega sistema,

- kot rezultat hitrosti $\dot{\psi}$: $\begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T = \dot{\psi}\begin{bmatrix}c_\varphi c_\vartheta & s_\varphi c_\vartheta & -s_\vartheta\end{bmatrix}^T$; upoštevali smo, da je med osnovnim in trenutnim koordinatnim sistemom zasuk φ okrog osi z osnovnega koordinatnega sistema in ϑ zasuk okrog osi y trenutnega koordinatnega sistema.

Postopek izračuna hitrosti izražene glede na osi osnovnega koordinatnega sistema:

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\ 0\\ \dot{\varphi}\end{bmatrix} + \begin{bmatrix}c_\varphi & -s_\varphi & 0\\ s_\varphi & c_\varphi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\ \dot{\vartheta}\\ 0\end{bmatrix} + \begin{bmatrix}c_\varphi & -s_\varphi & 0\\ s_\varphi & c_\varphi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}c_\vartheta & 0 & s_\vartheta\\ 0 & 1 & 0\\ -s_\vartheta & 0 & c_\vartheta\end{bmatrix}\begin{bmatrix}\dot{\psi}\\ 0\\ 0\end{bmatrix}, \quad (3.32)$$

Vplive vseh treh hitrosti Eulerjevih kotov na kotne hitrosti okrog osnovnega koordinatnega sistema dobimo, če zgornje izraze združimo v matrično obliko



Slika 3.3: Dekompozicija ZYZ Eulerjevih hitrosti v komponente referenčnega koordinatnega sistema.

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}0 & -s_\varphi & c_\varphi c_\vartheta\\ 0 & c_\varphi & s_\varphi c_\vartheta\\ 1 & 0 & -s_\vartheta\end{bmatrix} \begin{bmatrix}\dot{\varphi}\\ \dot{\vartheta}\\ \dot{\psi}\end{bmatrix}_{XYZ}. \tag{3.33}$$

Podobno velja tudi za ZYZ kombinacijo Eulerjevih kotov, kjer je vpliv vseh treh hitrosti Eulerjevnih kotov na kotne hitrosti okrog osnovnega koordinatnega sistema podan z

$$\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} = \begin{bmatrix}0 & -s_\varphi & c_\varphi s_\vartheta\\ 0 & c_\varphi & s_\varphi s_\vartheta\\ 1 & 0 & c_\vartheta\end{bmatrix} \begin{bmatrix}\dot{\varphi}\\ \dot{\vartheta}\\ \dot{\psi}\end{bmatrix}_{ZYZ}. \tag{3.34}$$

Zgornja izraza lahko krajše zapišemo kot

$$\boldsymbol{\omega} = \mathbf{T}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}. \tag{3.35}$$

Preostane nam še določitev transformacijske matrike za transformacijo med geometrijsko in analitično Jacobijevo matriko. Pri tem bomo upoštevali, da se translacijske hitrosti vrha robota ne spremenijo, če uporabimo različne zapise za orientacijo vrha. To pomeni, da je podmatrika Jacobijeve matrike, ki se nanaša na translacijske hitrosti enaka $\mathbf{J}_p(\mathbf{q})$, ne glede na izbiro analitične ali geometrijske Jacobijeve matrike. Transformacija rotacijskih hitrosti pa je določena z matriko $\mathbf{T}(\boldsymbol{\phi})$, iz česar lahko izpeljemo transformacijo med $\mathbf{J}_o(\mathbf{q})$ in $\mathbf{J}_\phi(\mathbf{q})$.

Kot izhodišče vzemimo sledeče tri relacije

$$\begin{aligned}\boldsymbol{\omega} &= \mathbf{J}_o(\mathbf{q})\dot{\mathbf{q}},\\ \dot{\boldsymbol{\phi}} &= \mathbf{J}_\phi(\mathbf{q})\dot{\mathbf{q}},\\ \boldsymbol{\omega} &= \mathbf{T}(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}.\end{aligned} \tag{3.36}$$

Iz zgornjih relacij lahko zapišemo

$$\mathbf{T}(\boldsymbol{\phi})\mathbf{J}_\phi(\mathbf{q})\dot{\mathbf{q}} = \mathbf{J}_o(\mathbf{q})\dot{\mathbf{q}}, \tag{3.37}$$

ker pomeni

$$\mathbf{T}(\boldsymbol{\phi})\mathbf{J}_\phi(\mathbf{q}) = \mathbf{J}_o(\mathbf{q}). \tag{3.38}$$

Celotno transformacijo lahko torej zapišemo

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_o(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}(\boldsymbol{\phi}) \end{bmatrix} \begin{bmatrix} \mathbf{J}_p(\mathbf{q}) \\ \mathbf{J}_\phi(\mathbf{q}) \end{bmatrix} = \mathbf{T}_A(\boldsymbol{\phi})\mathbf{J}_A(\mathbf{q}), \tag{3.39}$$

kjer je $\mathbf{I}$ enotska matrika dimenzij 3×3 in $\mathbf{0}$ matrika ničel dimenzij 3×3. Velja tudi obratno

$$\mathbf{J}_A(\mathbf{q}) = \mathbf{T}_A^{-1}(\boldsymbol{\phi})\mathbf{J}(\mathbf{q}). \tag{3.40}$$

4 Izvedba izračuna inverzne kinematike

Analitično Jacobijevo matriko robota $\mathbf{J}_A(\mathbf{q})$ lahko uporabimo za izračun hitrosti vrha robota $\dot{\mathbf{x}}$ (koordinate vrha robota so predstavljene z minimalnim številom parametrov – pozicija in orientacija z Eulerjevimi koti), če poznamo vektor sklepnih hitrosti $\dot{\mathbf{q}}$

$$\dot{\mathbf{x}} = \mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}}. \tag{3.41}$$

Seveda lahko uporabimo tudi obratno relacijo, če Jacobijeva matrika ni singularna

$$\dot{\mathbf{q}} = \mathbf{J}_A^{-1}(\mathbf{q})\dot{\mathbf{x}}. \tag{3.42}$$

Za izračun inverzne kinematike robota na osnovi inverzne analitične Jacobijeve matrike bomo kot izhodišče uporabili preprosto Eulerjevo integracijsko metodo. Ocenjeno vrednosti kotov sklepov $\hat{\mathbf{q}}(t_{k+1})$ v trenutku t_{k+1} lahko določimo na osnovi ocenjene vrednosti kotov sklepov $\hat{\mathbf{q}}(t_k)$ v trenutku t_k in hitrosti gibanja robota $\dot{\mathbf{q}}(t_k)$

$$\hat{\mathbf{q}}(t_{k+1}) = \hat{\mathbf{q}}(t_k) + \dot{\mathbf{q}}(t_k)\Delta t, \tag{3.43}$$

pri čemer je Δt računski korak regulatorja robota. Upoštevaje enačbo (3.42) lahko zgornjo enačbo zapišemo kot

$$\hat{\mathbf{q}}(t_{k+1}) = \hat{\mathbf{q}}(t_k) + \mathbf{J}_A^{-1}(\hat{\mathbf{q}}(t_k))\dot{\mathbf{x}}(t_k)\Delta t. \tag{3.44}$$

Na osnovi enačbe (3.44) bi bilo mogoče izračunavati inverzno kinematiko robota ob poznavanju začetne lege in hitrosti gibanja vrha robota. Ker numerična integracija vnaša lezenje, je takšen rezultat neuporaben. Rešitev problema lezenja je mogoče najti v vpeljavi povratne zanke, ki odpravlja lezenje s primerjavo prave lege vrha robota $\mathbf{x}$ z ocenjeno lego vrha robota $\hat{\mathbf{x}}$, izračunano na osnovi kotov določenih v enačbi (3.44). Najprej definirajmo napako izračunane lege vrha robota

$$\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - k(\hat{\mathbf{q}}), \tag{3.45}$$

pri čemer $k(\hat{\mathbf{q}})$ označuje direktni geometrijski model robota. Enačbo (3.45) odvajamo po času in dobimo

$$\dot{\tilde{\mathbf{x}}} = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}} = \dot{\mathbf{x}} - \mathbf{J}_A(\hat{\mathbf{q}})\dot{\hat{\mathbf{q}}}. \tag{3.46}$$

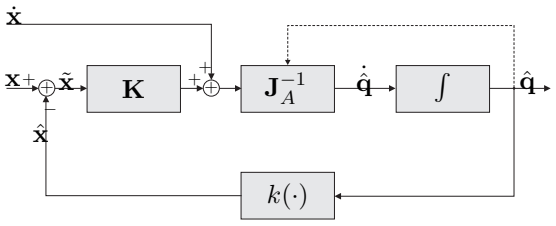
Napako $\tilde{\mathbf{x}}$ je potrebno vključiti v oceno sklepnih hitrosti $\dot{\hat{\mathbf{q}}}$, tako da bo zagotovljena konvergenca napake proti nič. Ena od možnosti je izbira sledečega regulacijskega algoritma

$$\dot{\hat{\mathbf{q}}} = \mathbf{J}_A^{-1}(\hat{\mathbf{q}})(\dot{\mathbf{x}} + \mathbf{K}\tilde{\mathbf{x}}) = \mathbf{J}_A^{-1}(\hat{\mathbf{q}})\dot{\mathbf{x}} + \mathbf{J}_A^{-1}(\hat{\mathbf{q}})\mathbf{K}\tilde{\mathbf{x}}, \tag{3.47}$$

kjer je $\mathbf{K}$ matrika proporcionalnih ojačanj zaprtozančnega sistema. Z vstavitvijo enačbe (3.47) v enačbo (3.46) dobimo

$$\dot{\tilde{\mathbf{x}}} + \mathbf{K}\tilde{\mathbf{x}} = \mathbf{0}. \tag{3.48}$$

Če je matrika ojačanj regulatorja $\mathbf{K}$ pozitivno-definitna (običajno diagonalna), je sistem (3.48) asimptotično stabilen. Shema izračuna inverzne kinematike robota na osnovi inverzne analitične Jacobijeve matrike je prikazana na sliki 3.4.



Slika 3.4: Inverzna kinematika z uporabo inverzne analitične Jacobijeve matrike. $\mathbf{K}$ predstavlja diagonalno matriko ojačanj, $\mathbf{J}_A$ analitično Jacobijevo matriko, $\int$ integrator, $k(\cdot)$ direktno kinematiko, $\mathbf{x}$ lega vrha robota, ki jo želimo pretvoriti v notranje koordinate, $\dot{\mathbf{x}}$ želene hitrost vrha robota in $\hat{\mathbf{x}}$ ocenjena lega vrha na podlagi izračunanih vrednosti kotov. Vse lege vrha robota so izražene z minimalnim številom parametrov (pozicija in orientacija z Eulerjevimi koti).

5 Odpravljanje skokov pri izračunu Eulerjevih kotov

V primeru, da je kateri izmed Eulerjevih kotov $\vartheta = [\varphi, \vartheta, \psi]^{\mathrm{T}}$ enak π, potem obstaja možnost, da ta vrednost pri izračunavanju variira med

π in $-\pi$, zato je potrebno ustrezno spremljati razliko med trenutno in prejšnjo vrednostjo.

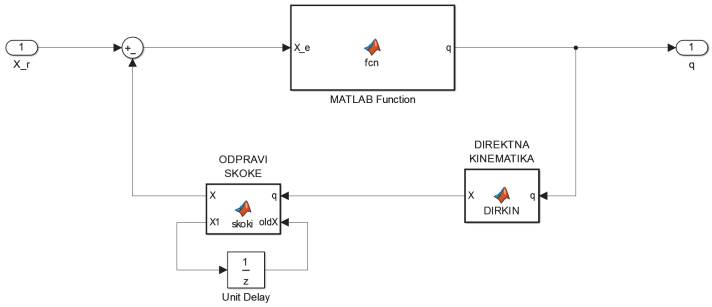
Če je absolutna razlika med prejšnjo $\mathbf{x}_{k-1}$ in trenutno vrednostjo $\mathbf{x}_k$ večja ali enaka π,

$$\Delta\mathbf{x} = \mathbf{x}_k - \mathbf{x}_{k-1}, \tag{3.49}$$

potem je potrebno trenutni vrednosti ustrezno odšteti vrednost 2π po sledeči enačbi

$$\mathbf{x}' = \mathbf{x}_k - sign(\Delta\mathbf{x})2\pi. \tag{3.50}$$

Ta koda pride takoj za izračunom direktne kinematike z Eulerjevimi koti v povratni zanki – primer uporabe je podan na sliki 3.5. V bloku `Unit Delay` se kot `Initial condition` nastavi začetne zunanje koordinate, ki se jih dobi kot rezultat funkcije `q2ZYXeul(q0)`.



Slika 3.5: Primer umestitve kode za odpravljanje skokov pri izračunu Eulerjevih kotov

6 Naloga

- V Matlabu dopolnite že zapisane predloge funkcij `function X = q2ZYXeul(q)` in `function invJ = ijacZYXeul(q)`.

  Funkcija `q2ZYXeul(q)` za vrednosti kotov v sklepih robota zapisanih v vektorju **q**, izračuna lego vrha robota v smislu pozicije XYZ in orientacije predstavljene z ZYX Eulerjevimi koti. Pozicija in Eulerjevi koti tvorijo vektor lege vrha robota. Pri tem uporabite funkcijo za izračun direktne kinematike robota.

  Druga funkcija `ijacZYXeul(q)` za vrednosti kotov v sklepih robota zapisanih v vektorju **q**, izračuna inverzno analitično Jacobijevo matriko. Uporabite funkcije za izračun geometrijske Jacobijeve matrike in lege robota predstavljene s pozicijo ter Eulerjevimi koti.

  Vaša naloga je, da pravilno dopolnite funkcijo na mestih, kjer je označeno s komentarjem `%%% STUDENT %%%`.

- Na osnovi slike 3.4 zgradite v Simulinku simulacijsko shemo za izračun inverzne kinematike na osnovi inverzne Jacobijeve matrike, kot je prikazano na sliki 3.6. Delovanje sistema preizkusite s testnimi podatki, ki omogočajo premikanje robota po premici v ravnini YZ, kot je prikazano na sliki 3.7. Testni podatki so shranjeni v datoteki `xTrajec.mat`.

7 Potrebno predznanje, cilj in namen naloge

Pri izpeljavi naloge v okolju Matlab in Matlab Simulink dodatno znanje glede na prejšnje naloge ni potrebno. Cilj naloge je izvedba simulacijske sheme inverzne kinematike z uporabo inverzne analitične Jacobijeve matrike z uporabo minimalnega števila parametrov za opis robotskega mehanizma.

8 Matlab predlogi

Predloga funkcije **x = q2ZYXeul(q)**

```
% q - Vektor kotnih spremenljivk robota (vhod).
% x - Lega vrha robota XYZ in orientacija z ZYX Eulerjevimi koti (izhod).

% Matrika lege vrha robota.
    T6 = ;                                              %%% STUDENT %%%

% Izluscimo ustrezne komponente rotacijske matrike.
    r21 = ;                                             %%% STUDENT %%%
    r31 = ;                                             %%% STUDENT %%%
    r32 = ;                                             %%% STUDENT %%%
    r33 = ;                                             %%% STUDENT %%%
    r11 = ;                                             %%% STUDENT %%%

% Izracunamo ustrezne Eulerjeve kote.
    fi = ;                                              %%% STUDENT %%%
    theta = ;                                           %%% STUDENT %%%
    psi = ;                                             %%% STUDENT %%%

% Orientacija vrha predstavljena z vektorjem treh ZYX Eulerjevih kotov.
    ea = [fi; theta; psi];

% Lega vrha robota v smislu polozaja XYZ in orientacije predstavljene z
% ZYX Eulerjevimi koti.
    X = ;                                               %%% STUDENT %%%
```

Testni rezultat:
$q2ZYXeul([1,2,...,6]) = \begin{bmatrix} 0.2049 & 0.4353 & 0.3140 & -1.9354 & -1.4558 & 0.6075 \end{bmatrix}^T$

Predloga funkcije **function invJ = ijacZYXeul(q)**

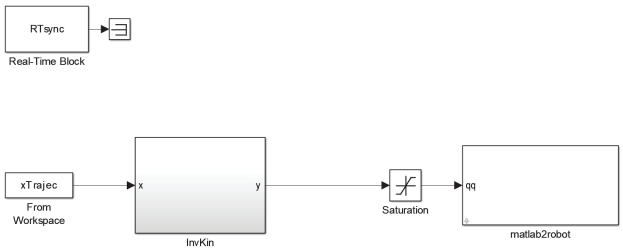
```
% q - Vrednost kotov v sklepih robota (vhod).
% invJ - Inverzna analiticna Jacobijeva matrika, kjer je orientacija
%        predstavljena z ZYX Eulerjevimi koti (izhod).

% Izracun geometrijske Jacobijeve matrike (Jg).
    Jg = ;                                              %%% STUDENT %%%

% Izracun lege vrha robota z ZYX Eulerjevimi koti za opis orientacije.
    X = ;                                               %%% STUDENT %%%

% Iz vektorja X izluscimo ustrezne komponente Eulerjevih kotov.
    fi      = ;                                         %%% STUDENT %%%
    theta   = ;                                         %%% STUDENT %%%
    psi     = ;                                         %%% STUDENT %%%

% Izracun transformacijske matrike za pretvorbo iz geometrijske v
% analiticno Jacobijevo matriko.
    Tr = ;                                              %%% STUDENT %%%

% Izracun inverzne analiticne Jacobijeve matrike.
    invJ = ;                                            %%% STUDENT %%%
```

Testni rezultat:

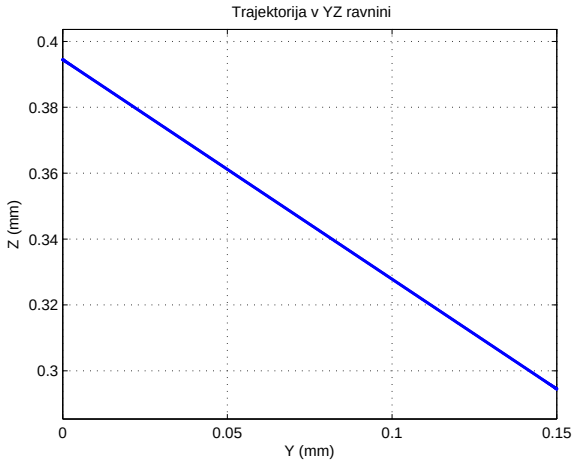
$$ijacZYXeul([1,...]) = \begin{bmatrix} -2.0129 & 1.2925 & -0.0000 & -0.1410 & 0.0040 & -0.1423 \\ -0.2833 & -0.4412 & -3.3076 & -0.0329 & 0.2292 & -0.0515 \\ 1.9414 & 3.0235 & 1.9032 & 0.2256 & -0.1056 & 0.2355 \\ 2.4118 & -0.7322 & -0.3144 & 1.1999 & 0.2629 & 1.1867 \\ 0.6517 & 1.9654 & -0.9180 & -0.1190 & 0.8700 & -0.1897 \\ -1.6978 & -1.7881 & 1.1084 & -0.3727 & -0.7355 & -0.4568 \end{bmatrix}$$

Za identiteto (I) uporabite eye(N,M), za ničelno matriko pa zeros(N,M)!

9 Simulacijska shema v Matlab Simulink okolju



Slika 3.6: Simulacijska shema v Simulinku; blok InvKin vsebuje shemo s slike 3.4.



Slika 3.7: Preizkusna trajektorija za testiranje izračuna inverzne kinematike robota.

Za odpiranje predloge Simulink sheme v Matlab ukazni vrstici zaženite program *initVR.m*!

Spremenjeno shemo shranite pod drugim imenom!

Kje se nahajajo posamezni Simulink bloki:

Discrete Derivative ⇒ *Simulink* → *Discrete*
Gain value: 1.0; Inital condition ...: 0.0
Saturation ⇒ *Simulink* → *Discontinuities*
Upper limit: 0.5; Lower limit: -0.5
Gain ⇒ *Simulink* → *Math Operations*
Gain: fbGain
MATLAB Function ⇒ *Simulink* → *User-Defined Func.*
Matrix Multiply ⇒ *Simulink* → *Math Operations*
Descrete-Time Integrator ⇒ *Simulink* → *Discrete*
Gain value: 1.0; Initial condition: q0; Sample time: dT

TRANSFORMACIJE SIL IN NAVOROV

Na robotski mehanizem lahko med samim delovanjem delujejo zunanje sile, ki povzročajo prenos sil in navorov na sam robotski mehanizem. Industrijski roboti običajno niso v neposrednem stiku z okoljem, vendar pa se v raziskovalni sferi vedno bolj pogosto robotski mehanizmi uporabljajo ravno za interakcijo z okoljem in z leti bodo ta koncept prenesli tudi na industrijske robote. Za ta namen imajo robotski mehanizmi na vrhu nameščen senzor sil in navorov s katerim lahko izmerimo velikost fizikalnih veličin in glede na te veličine vplivamo preko posebnih shem vodenja na obnašanje mehanizma. Da pa je premikanje mehanizma glede na zunanje vplive ustrezno, je potrebno poznavanje oz. znanje preračuna sil in navorov glede na posamezne segmente robotskega mehanizma oz. v posamezne koordinatne sisteme mehanizma.

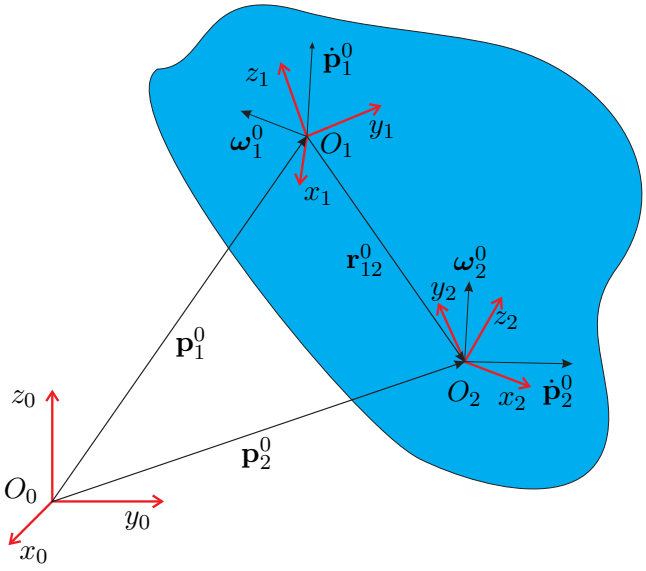
Za ta namen bomo izpeljali in uporabili koncept kineto-statične dualnosti, ki jo najlažje izpeljemo na primeru transformacij hitrosti med koordinatnimi sistemi.

1 Transformacije hitrosti

Slika 4.1 prikazuje togo telo s pripetima dvema koordinatnima sistemoma ($O_1 - x_1y_1z_1$ in $O_2 - x_2y_2z_2$), med seboj razmaknjena za razdaljo, ki jo predstavlja vektor $\mathbf{r}_{12}^0$. Poleg teh dveh koordinatnih sistemov je narisan tudi zunanji referenčni koordinatni sistem označen z $O_0 - x_0y_0z_0$. Premikanje togega telesa obravnavamo glede na referenčni koordinatni sistem. Povezava med translacijskima in rotacijskima hitrostima obeh koordinatnih sistemov je določena z

$$\begin{aligned} \dot{\mathbf{p}}_2^0 &= \dot{\mathbf{p}}_1^0 + \boldsymbol{\omega}_1^0 \times \mathbf{r}_{12}^0 \\ \boldsymbol{\omega}_2^0 &= \boldsymbol{\omega}_1^0, \end{aligned} \tag{4.51}$$

kjer so vsi vektorji izraženi glede na referenčni koordinatni sistem O_0.



Slika 4.1: Predstavitev linearnih in kotnih hitrosti v različnih koordinatnih sistemih na istem togem telesu.

Vektorski produkt, ki nastopa v enačbi (4.51), lahko zapišemo kot produkt poševno-simetrične matrike $\mathbf{S}(\mathbf{r}_{12}^0)$ in vektorja $\boldsymbol{\omega}_1^0$, torej

$$\boldsymbol{\omega}_1^0 \times \mathbf{r}_{12}^0 = -\mathbf{r}_{12}^0 \times \boldsymbol{\omega}_1^0 = -\mathbf{S}(\mathbf{r}_{12}^0)\boldsymbol{\omega}_1^0, \tag{4.52}$$

kjer je poševno-simetrična matrika

$$\mathbf{S}(\mathbf{r}_{12}^0) = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}. \tag{4.53}$$

Matrika vsebuje komponente vektorja $\mathbf{r}$, ki povezuje koordinatna sistema O_1 in O_2. Za poševno-simetrično matriko lahko zapišemo sledeči lastnosti

$$\mathbf{S}(\mathbf{R}\mathbf{r}) = \mathbf{R}\mathbf{S}(\mathbf{r})\mathbf{R}^T \tag{4.54}$$

in

$$\mathbf{S} + \mathbf{S}^T = \mathbf{0}. \tag{4.55}$$

Z uporabo poševno-simetrične matrike enačbi (4.51) zapišemo kot

$$\begin{aligned} \dot{\mathbf{p}}_2^0 &= \dot{\mathbf{p}}_1^0 - \mathbf{S}(\mathbf{r}_{12}^0)\boldsymbol{\omega}_1^0 \\ \boldsymbol{\omega}_2^0 &= \boldsymbol{\omega}_1^0, \end{aligned} \tag{4.56}$$

oziroma v matrični obliki

$$\begin{bmatrix} \dot{\mathbf{p}}_2^0 \\ \boldsymbol{\omega}_2^0 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{S}(\mathbf{r}_{12}^0) \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}}_1^0 \\ \boldsymbol{\omega}_1^0 \end{bmatrix}, \tag{4.57}$$

kjer je $\mathbf{I}$ enotska matrika ustreznih dimenzij.

Vektorji v enačbi (4.57) so izraženi glede na referenčni koordinatni sistem $O_0 - x_0y_0z_0$. Če pa so vektorji izraženi v lokalnih koordinatnih sistemih, je potrebno upoštevati ustrezne transformacije. Vektor $\mathbf{r}_{12}^1$ izražen v koordinatnem sistemu $O_1 - x_1y_1z_1$ lahko zapišemo v osnovnem koordinatnem sistemu $O_0 - x_0y_0z_0$ kot

$$\mathbf{r}_{12}^0 = \mathbf{R}_1^0\mathbf{r}_{12}^1, \tag{4.58}$$

kjer matrika $\mathbf{R}_1^0$ določa orientacijo koordinatnega sistema $O_1 - x_1y_1z_1$ glede na $O_0 - x_0y_0z_0$. Podobno lahko transformiramo tudi vektorja $\dot{\mathbf{p}}_1^1$ in $\boldsymbol{\omega}_1^1$, ki sta izražena v koordinatnem sistemu $O_1 - x_1y_1z_1$

$$\begin{aligned} \dot{\mathbf{p}}_1^0 &= \mathbf{R}_1^0\dot{\mathbf{p}}_1^1 \\ \boldsymbol{\omega}_1^0 &= \mathbf{R}_1^0\boldsymbol{\omega}_1^1. \end{aligned} \tag{4.59}$$

Vektorja $\dot{\mathbf{p}}_2^2$ in $\boldsymbol{\omega}_2^2$ sta izražena v koordinatnem sistemu $O_2 - x_2y_2z_2$, zato je transformacija v osnovni koordinatni sistem sledeča

$$\begin{aligned} \dot{\mathbf{p}}_2^0 &= \mathbf{R}_2^0\dot{\mathbf{p}}_2^2 = \mathbf{R}_1^0\mathbf{R}_2^1\dot{\mathbf{p}}_2^2 \\ \boldsymbol{\omega}_2^0 &= \mathbf{R}_2^0\boldsymbol{\omega}_2^2 = \mathbf{R}_1^0\mathbf{R}_2^1\boldsymbol{\omega}_2^2, \end{aligned} \tag{4.60}$$

kjer matrika $\mathbf{R}_2^0$ določa orientacijo koordinatnega sistema $O_2 - x_2y_2z_2$ glede na $O_0 - x_0y_0z_0$ in matrika $\mathbf{R}_2^1$ orientacijo koordinatnega sistema $O_2 - x_2y_2z_2$ glede na $O_1 - x_1y_1z_1$.

Upoštevaje zapise (4.58), (4.59) in (4.60) ter lastnost poševno-simetrične matrike (4.54) v enačbah (4.56) dobimo

$$\begin{aligned} \mathbf{R}_1^0\mathbf{R}_2^1\dot{\mathbf{p}}_2^2 &= \mathbf{R}_1^0\dot{\mathbf{p}}_1^1 - \mathbf{R}_1^0\mathbf{S}(\mathbf{r}_{12}^1)\mathbf{R}_1^{0\,T}\mathbf{R}_1^0\boldsymbol{\omega}_1^1 \\ \mathbf{R}_1^0\mathbf{R}_2^1\boldsymbol{\omega}_2^2 &= \mathbf{R}_1^0\boldsymbol{\omega}_1^1. \end{aligned} \tag{4.61}$$

Enačbi pomnožimo na obeh straneh z $\mathbf{R}_1^{-1}$ in $\mathbf{R}_2^{1\,-1} = \mathbf{R}_1^2$ ter rezultat zapišemo v matrični obliki

$$\begin{bmatrix} \dot{\mathbf{p}}_2^2 \\ \boldsymbol{\omega}_2^2 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1^2 & -\mathbf{R}_1^2\mathbf{S}(\mathbf{r}_{12}^1) \\ \mathbf{0} & \mathbf{R}_1^2 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}}_1^1 \\ \boldsymbol{\omega}_1^1 \end{bmatrix}. \tag{4.62}$$

Enačba (4.62) predstavlja splošno transformacijo hitrosti iz začetnega (koordinatni sistem z oznako 1 glede na sliko 4.1) v nov koordinatni sistem (koordinatni sistem z oznako 2 glede na sliko 4.1). Ker zapis transformira hitrosti, lahko matriki v enačbi (4.62) pripišemo vlogo Jacobijeve matrike. Tako lahko enačbo zapišemo tudi v obliki

$$\mathbf{v}_2^2 = \mathbf{J}_1^2 \mathbf{v}_1^1. \tag{4.63}$$

Enačba 4.63 ponazarja relacijo za pretvorbo hitrosti med dvema koordinatnima sistemoma, pri čemer je $\mathbf{v}_1^1 = \begin{bmatrix} \dot{\mathbf{p}}_1^1 & \boldsymbol{\omega}_1^1 \end{bmatrix}^T$, $\mathbf{v}_2^2 = \begin{bmatrix} \dot{\mathbf{p}}_2^2 & \boldsymbol{\omega}_2^2 \end{bmatrix}^T$ in

$$\mathbf{J}_1^2 = \begin{bmatrix} \mathbf{R}_1^2 & -\mathbf{R}_1^2 \mathbf{S}(\mathbf{r}_{12}^1) \\ \mathbf{0} & \mathbf{R}_1^2 \end{bmatrix}. \tag{4.64}$$

2 Transformacije sil in navorov

Ob začetku poglavja smo navedli pojem kineto-statične dualnosti. Enačbi 4.65 in 4.66 predstavljata kineto-statično dualnost v smislu transformacije hitrosti in sil med dvema koordinatnima sistemoma.

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}, \tag{4.65}$$

$$\boldsymbol{\tau} = \mathbf{J}^T \mathbf{F}. \tag{4.66}$$

Enačba (4.65) določa pretvorbo med hitrostmi izraženimi v sklepnem koordinatnem sistemu v hitrosti vrha manipulatorja. Enačba (4.66) pa opisuje pretvorbo sil na vrhu manipulatorja v navore v sklepih. Torej gre v obeh enačbah za transformacije iz sklepnega v koordinatni sistem vrha oziroma obratno. Relacija, ki določa transformacijo, je Jacobijeva oziroma transponirana Jacobijeva matrika. Dualnost predstavljeno z enačbama (4.65) in (4.66) bomo uporabili za določitev transformacij sil in navorov.

V ta namen je potrebno poiskati transformacijo dualno transformaciji (4.63). Glede na dualnost (4.66) lahko zapišemo

$$\boldsymbol{\gamma}_1^1 = {\mathbf{J}_1^2}^T \boldsymbol{\gamma}_2^2, \tag{4.67}$$

kjer vektor $\boldsymbol{\gamma}_1^1 = \begin{bmatrix} \mathbf{f}_1^1 & \boldsymbol{\mu}_1^1 \end{bmatrix}^T$ predstavlja sile in navore izražene v koordinatnem sistemu $O_1 - x_1 y_1 z_1$ in vektor $\boldsymbol{\gamma}_2^2 = \begin{bmatrix} \mathbf{f}_2^2 & \boldsymbol{\mu}_2^2 \end{bmatrix}^T$ sile in navore

izražene v koordinatnem sistemu $O_2 - x_2y_2z_2$. Matrika $\mathbf{J}_1^{2^T}$ je enaka

$$\mathbf{J}_1^{2^T} = \begin{bmatrix} \mathbf{R}_1^{2^T} & \mathbf{0} \\ (-\mathbf{R}_1^2\mathbf{S}(\mathbf{r}_{12}^1))^T & \mathbf{R}_1^{2^T} \end{bmatrix}. \tag{4.68}$$

Upoštevaje lastnost poševno-simetrične matrike (4.55) lahko transformacijo $\left(-\mathbf{R}_1^2\mathbf{S}(\mathbf{r}_{12}^1)\right)^T$, z upoštevanjem nekaterih pravil

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T, \quad \mathbf{R}_1^{2^T} = \mathbf{R}_1^{2^{-1}} = \mathbf{R}_2^1, \quad \mathbf{S} = -\mathbf{S}^T, \tag{4.69}$$

zapišemo kot

$$\left(-\mathbf{R}_1^2\mathbf{S}(\mathbf{r}_{12}^1)\right)^T = -\mathbf{S}^T(\mathbf{r}_{12}^1)\mathbf{R}_2^1 = \mathbf{S}(\mathbf{r}_{12}^1)\mathbf{R}_2^1. \tag{4.70}$$

S tem dobimo končno enačbo, ki omogoča transformacijo sil in navorov med dvema koordinatnima sistemoma

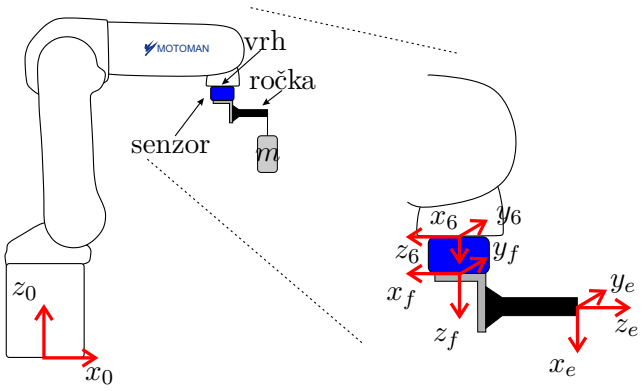
$$\begin{bmatrix} \mathbf{f}_1^1 \\ \boldsymbol{\mu}_1^1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{0} \\ \mathbf{S}(\mathbf{r}_{12}^1)\mathbf{R}_2^1 & \mathbf{R}_2^1 \end{bmatrix} \begin{bmatrix} \mathbf{f}_2^2 \\ \boldsymbol{\mu}_2^2 \end{bmatrix}. \tag{4.71}$$

3 Naloga

Na koncu ročke pritrjene na robotu Motoman je obešena utež kot prikazuje slika 4.2. Silo, ki jo povzroča utež merimo s senzorjem sile pritrjenim na vrh robota. Senzor sile podaja izmerjeno silo v lastnem koordinatnem sistemu. Izračunati morate kakšna sta dejanska sila in navor v koordinatnem sistemu ročke in v osnovnem koordinatnem sistemu robota.

V prvi vaji ste za robota Motoman že izračunali direktno kinematiko. Vektor izmerjenih vrednosti kotov v sklepih robota lahko uporabite v funkciji za izračun direktne kinematike robota. Sile, ki jih izmeri senzor sil in navorov odčitajte z ukazom `[w, ft, q] = read_motoman`, kjer `w` določa lego koordinatnega sistema vrha robota, `ft` izmerjene sile in navore senzorja sil in navorov ter `q` trenutne kote v sklepih robota. Masa obešena na koncu ročke je 2 kg.

V nadaljevanju sta podani predlogi funkcij za preračun sil in navorov v koordinatni sistem ročke oziroma koordinatni sistem baze robota. Vaša naloga je, da pravilno dopolnite funkcije na mestih, kjer je označeno s komentarjem `%%% STUDENT %%%`.



Slika 4.2: Koordinatni sistemi baze, vrha, senzorja ter ročke na robotu.

Zapis 4.72 predstavlja transformacijo med koordinatnim sistemom vrha robota in koordinatnim sistemom senzorja sile.

$$T_{6f} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.0395 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.72}$$

Transformacijska matrika 4.73 pa predstavlja transformacijo med koordinatnim sistemom senzorja sile ter koordinatnim sistemom konca ročke, kjer je obešena utež.

$$T_{fe} = \begin{bmatrix} 0 & 0 & -1 & -0.170 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0.0355 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.73}$$

4 Potrebno predznanje, cilj in namen naloge

Cilj naloge je, da na praktičnem primeru preizkusite koncept kineto-statične dualnosti za tranformacijo sil in navorov med koordinatnimi sistemi. Potrebno predznanje obsega razumevanje pomena transformacijskih matrik in uporaba okolja Matlab.

5 Matlab predlogi

Predloga funkcije **fte = ft2endeffector(q, ft)**

```
% Funkcija preracuna sile in navore, ki jih izmeri senzor v koordinatni
% sistem, ki je pritrjen na vrhu robota v rocki.
% Vhodna vektorja sta polozaji v sklepih q in sile ter navori, ki jih
% izmeri senzor ft.
% Izhod funkcije je vektor sil in navorov izrazen v koordinatnem sistemu
% rocke fte.

% Transformacija od k.s. vrha robota do k.s. senzorja.
    t6F = [eye(3),[0; 0; 0.0395]; [0 0 0 1]];

% Transformacija od k.s. senzorja do k.s. rocke.
    tFE = [0 0 -1 -0.170;0 1 0 0;1 0 0 0.0355;0 0 0 1];

% Izracunajte transformacijsko matriko T za transformacijo sil in
% navorov med koordinatnim sistemom senzorja in rocke.
    T = ;                                              %%% STUDENT %%%

% Izhodni vektor sil in navorov.
    fte = ;                                            %%% STUDENT %%%
```

Testni rezultat:

$ft2endeffector([1,2,...,6]', [1,2,...,6]') = \begin{bmatrix} 3 & 2 & -1 & 6.3400 & 4.4545 & -4.0710 \end{bmatrix}^T$

Predloga funkcije **ftb = ft2base(q, ft)**

```
% Funkcija preracuna sile in navore, ki jih izmeri senzor v koordinatni
% sistem, ki je pritrjen v bazi robota.
% Vhodna vektorja sta polozaji v sklepih q in sile ter navori, ki jih
% izmeri senzor ft.
% Izhod funkcije je vektor sil in navorov izrazen v koordinatnem sistemu
% baze robota ftb.

% Transformacija od k.s. vrha robota do k.s. senzorja.
    t6F = [eye(3),[0; 0; 0.0395]; [0 0 0 1]];

% Transformacija od k.s. senzorja do k.s. rocke.
    tFE = [0 0 -1 -0.170;0 1 0 0;1 0 0 0.0355;0 0 0 1];

% Lista transformacijskih matrik A.
    A = ;                                              %%% STUDENT %%%

% Transformacijska matrika lege vrha.
    T6 = ;                                             %%% STUDENT %%%

% Izracunajte transformacijsko matriko T za transformacijo sil in
% navorov med koordinatnim sistemom senzorja in baze robota.
    T = ;                                              %%% STUDENT %%%

% Izhodni vektor sil in navorov.
    ftb = ;                                            %%% STUDENT %%%
```

Za identiteto (I) uporabite eye(N,M), za ničelno matriko pa zeros(N,M)!

Testni rezultat:

$ft2base([1,...,6]', [1,...,6]') = \begin{bmatrix} 1.1703 & 3.2635 & 1.4070 & 2.8575 & 6.6466 & 4.9497 \end{bmatrix}^T$

NAČRTOVANJE TRAJEKTORIJE

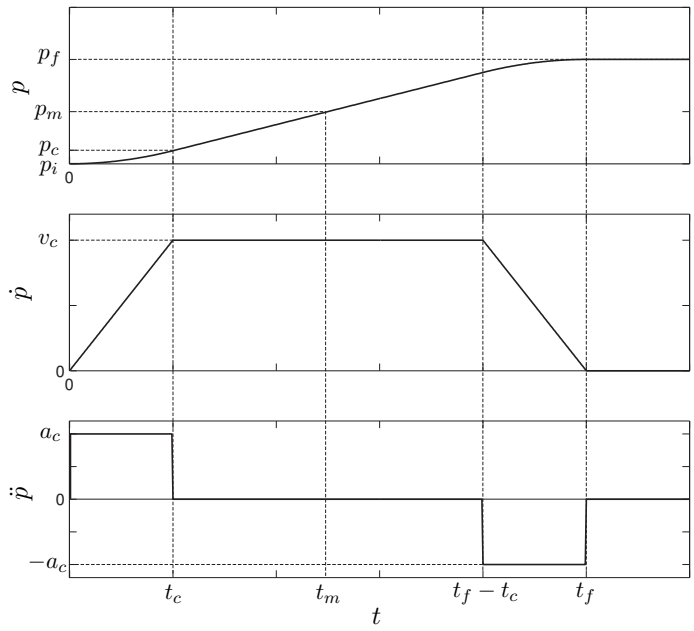
Pri prejšnjih vajah smo se ukvarjali z matematičnimi modeli robotskih mehanizmov oz. kinematiko mehanizmov. Preden pridobljeno znanje uporabimo za študij vodenja robotov, moramo spoznati načrtovanje robotskega gibanja oz. načrtovanje robotskih trajektorij. Cilj načrtovanja trajektorij je generiranje referenčnih vhodov regulacijskim sistemom, ki bodo zagotovili, da se bo vrh robota gibal po željeni trajektoriji oz. poti. Gibanje robotskega manipulatorja je najpogosteje določeno v zunanjih kartezičnih koordinatah, vendar srečamo tudi premikanje robotskega mehanizma po sklepih oz. notranjih koordinatah.

Na področju robotike srečamo različne metode za interpolacijo in generiranje trajektorij. Za generiranje trajektorij od točke do točke (*point-to-point*) se najpogosteje uporablja linearna interpolacija in interpolacije s pomočjo polinomov tretjega ter petega reda. Obstajajo tudi bolj neobičajni postopki interpolacije, kot recimo *on-line* generiranje trajektorije s pomočjo A* algoritma in baznih zlepkov oz. B-zlepkov (*B-splines*).

Za demonstracijo generiranja trajektorije bomo uporabili najpreprostejši pristop, kjer uporabnik določi le začetno in končno točko vrha robota, trajektorija med točkama pa je linearna.

1 Interpolacija trajektorije med dvema točkama

Včasih nas trajektorija vrha robota niti ne zanima, zato je dovolj interpolacija notranjih spremenljivk sklepov. Nemalokrat pa želimo gibanje vrha robota med dvema točkama po premici, pri čemer sodeluje



Slika 5.1: Časovni poteki spremenljivk pri trapeznem hitrostnem profilu.

več sklepov robota. Spremenljivko trajektorije vrha bomo posplošeno označevali s črko p. Pri gibanju od točke do točke, se mora robotski manipulator premakniti iz začetne (p_i) v končno (p_f) točko v času t_f.

Izmed mnogih profilov je najbolj razširjen tako imenovanem *trapezni hitrostni profil*. Gib se začne ob času $t = 0$ s fazo konstantnega pospeška oz. enakomernega povečevanja hitrosti, nadaljuje s fazo konstantne hitrosti in konča s fazo konstantnega pojemka (Slika 5.1) oz. enakomernega zmanjševanja hitrosti. Vsi trije deli predstavljajo trapez in od tu tudi ime. Rezultirajoči časovni potek premika vrha je sestavljen iz osrednjega linearnega dela, ki se začne in konča s paraboličnim segmentom. Začetna in končna hitrost giba sta enaki nič. Faza konstantnega pospeška traja enako dolgo kot faza konstantnega pojemka. V obeh fazah je velikost pospeška enaka a_c. Na ta način imamo opravka s simetrično trajektorijo, kjer velja

$$p_m = \frac{p_f + p_i}{2} \quad \text{ob času} \quad t_m = \frac{t_f}{2}. \tag{5.74}$$

Vendar je za dosego linearnega giba vrha robota potrebno upoštevati, da željena hitrost giba in željen pospešek ter pojemek veljata samo za najdaljšo pot, gledano na koordinatne osi baznega koordinatnega sistema. Za vse ostale smeri premika vrha, pa podane zaheteve po hitrosti

in pospešku ne veljajo, ampak jih je potrebno izračunati glede na parametre najdaljše poti.

Trajektorija $p(t)$ mora zadovoljiti nekatere omejitve, tako da bomo prišli iz začetne točke p_i v končno točko p_f v času t_f. Hitrost na koncu parabolične faze mora biti enaka konstantni hitrosti linearnega dela. Prvo hitrost dobimo iz enačbe, ki opisuje gibanje s konstantnim pospeškom

$$v = a_c t. \tag{5.75}$$

Na koncu prve faze velja

$$v_c = a_c t_c. \tag{5.76}$$

Hitrost v drugi fazi dobimo s pomočjo slike 5.1

$$v_c = \frac{p_m - p_c}{t_m - t_c}, \tag{5.77}$$

kjer pomeni p_c vrednost premika ob koncu faze paraboličnega gibanja, to je v času t_c. V tem času je potekalo gibanje s konstantnim pospeškom a_c in hitrost je določena z enačbo (5.75). Potek poti dobimo z integriranjem enačbe (5.75)

$$p = \int v dt = a_c \int t dt = a_c \frac{t^2}{2} + p_i, \tag{5.78}$$

kjer smo kot integracijsko konstanto upoštevali začetni položaj p_i. Ob koncu prve faze velja

$$p_c = p_i + \frac{1}{2} a_c t_c^2. \tag{5.79}$$

Hitrost na koncu prve faze (5.76) izenačimo s konstantno hitrostjo druge faze (5.77)

$$a_c t_c = \frac{p_m - p_c}{t_m - t_c}. \tag{5.80}$$

Če v enačbo (5.80) vstavimo enačbo (5.79) in upoštevamo izraz (5.74), po preureditvi dobimo naslednjo kvadratično enačbo

$$a_c t_c^2 - a_c t_f t_c + p_f - p_i = 0. \tag{5.81}$$

Izrazimo še t_c:

$$t_c = \frac{t_f}{2} - \frac{1}{2} \sqrt{\frac{t_f^2 a_c - 4(p_f - p_i)}{a_c}}. \tag{5.82}$$

Da generiramo gib med začetno lego p_i in končno lego p_f, moramo v prvi fazi generirati naslednji polinom

$$p_1(t) = p_i + \frac{1}{2} a_c t^2 \qquad 0 \leq t \leq t_c. \tag{5.83}$$

V drugem delu generiramo premico z začetkom v točki $[t_c, p_c]$ in ima naklon v_c:

$$(p - p_c) = v_c(t - t_c). \tag{5.84}$$

Po preureditvi dobimo naslednji potek

$$p_2(t) = p_i + a_c t_c(t - \frac{t_c}{2}) \qquad t_c < t \leq (t_f - t_c). \tag{5.85}$$

V zadnjem delu ponovno generiramo parabolo, le da ima sedaj ekstrem v točki $[t_f, p_f]$ in je obrnjena navzdol

$$p_3 = p_f - \frac{1}{2}a_c(t - t_f)^2 \qquad (t_f - t_c) < t \leq t_f \tag{5.86}$$

Tako smo dobili analitični potek giba vrha robota med dvema točkama.

2 Naloga

V Matlabu napišite funkcije za izvedbo sledečih nalog:

- `function [time, p] = interpolate(t_c, t_f, a_c, dT, p_i, p_f)`, ki generira interpolirano pot med dvema podanima tičkama (p_i, p_f) pri vhodnih pogojih (t_c, t_f, a_c, dT).
- `function trajectory()`, ki izračunava potrebne vhodne parametre za funkcijo *interpolate* in izriše potek interpolirane poti za posamezno koordinato ter skupen premik.
- Odprite simulacijski model iz poglavja 3 in testirajte interpolirano trajektorijo, tako da pošljete robotu Motoman interpolirano trajektorijo.

Vaša naloga je, da pravilno dopolnite mesta s `%%% STUDENT %%%`.

3 Potrebno predznanje, cilj in namen naloge

Zahtevano predznanje je poznavanje osnovnih fizikalnih enačb enakomernega gibanja (hitrost in pospešek) ter poznavanje okolja Matlab. Z uporabo predznanja in podanih enačb ter razlage boste pri vaji na praktičnem primeru generirali trajektorijo (Slika 3.7) za premik vrha robota Motoman z uporabo trapeznega hitrostnega profila.

4 Matlab predlogi

Predloga funkcije **[time, p] = interpolate(t_c, t_f, a_c, dT, p_i, p_f)**

```
% t_c - Cas pospesevanja in zaviranja.
% t_f - Celoten cas gibanja.
% a_c - Pospesek/pojemek.
% dT  - Vzrocni cas interpolatorja.
% p_i - Zacetni polozaj.
% p_f - Koncni polozaj.

% Definiramo vektor casa za posamezne odseke interpolacije.
    t1 = ;                                          %%% STUDENT %%%
    t2 = ;                                          %%% STUDENT %%%
    t3 = ;                                          %%% STUDENT %%%

% Izracun interpolirane poti za pospesevanje, enakomerno
% hitrost in zaviranje.
    p1 = ;                                          %%% STUDENT %%%
    p2 = ;                                          %%% STUDENT %%%
    p3 = ;                                          %%% STUDENT %%%

% Sestavimo vektorje v samostojni vektor.
    time = [t1,t2,t3];
    p = [p1,p2,p3];
```

Predloga datoteke **trajectory.m**

```
% Zbrisemo stare podatke.
    clear all; close all; clc;

% Vzrocni cas.
    dT = 1/4000;

% Zacetna lega robota.
    q0 = [0 0 0 0 90 0]'*pi/180;

% Zapis lege robota po komponentah.
    p0 = q2XYZeul(q0);
    pX0 = p0(2);
    pY0 = p0(3);

% Sprememba koncne pozicije robota (v metrih).
    dpY = 0.15;
    dpZ = -0.10;

% Zeljena hitrost gibanja (m/s).
    v_c = 0.05;

% Pospesek in pojemek giba (m/s2).
    a_c = 0.25;
```

```
% Odlociti se je potrebno kateri premik je daljsi.
% Lp_i ... Long initial
% Lp_f ... Long final
% Sp_i ... Short initial
% Sp_f ... Short final
    if(dpY >= dpZ)
        p_i_Dolg = pY0;
        p_f_Dolg = pY0 + dpY;
        p_i_Kratek = pZ0;
        p_f_Kratek = pZ0 + dpZ;
    else
        p_i_Dolg = pZ0;
        p_f_Dolg = pZ0 + dpZ;
        p_i_Kratek = pY0;
        p_f_Kratek = pY0 + dpY;
    end

% *** IZRACUN ZA DALSI GIB ***
% Izracun casa pospesevanja.
    t_c = ;                                             %%% STUDENTI %%%

% Izracun polozaja ob koncu pospesevanja.
    p_c = ;                                             %%% STUDENTI %%%

%  Izracun dolzine giba pri konstantni hitrosti.
    d_l = ;                                             %%% STUDENTI %%%

% Ce je dolzina linearnega dela negativna, je pospesek prevelik.
    if(d_l < 0 )
        sprintf('Pospesek je prevelik!')
        break
    end

% Izracun casa giba pri konstantni hitrosti.
    t_l = ;                                             %%% STUDENTI %%%

% Celoten cas giba - cas pospesevanje je enak casu zaviranja.
    t_f = ;                                             %%% STUDENTI %%%

% Izracun interpolirane poti za koordinato daljse poti.
    [time1,p1] = ;                                      %%% STUDENTI %%%

% *** IZRACUN ZA KRAJSI GIB ***
% Izracun pospeska za koordinato kratke poti.
    a_c = ;                                             %%% STUDENTI %%%

% Izracun interpolirane poti za koordinato kratke poti.
    [time2,p2] = ;                                      %%% STUDENTI %%%

% Izris grafov ...
```

NEWTON-EULERJEVA DINAMIKA

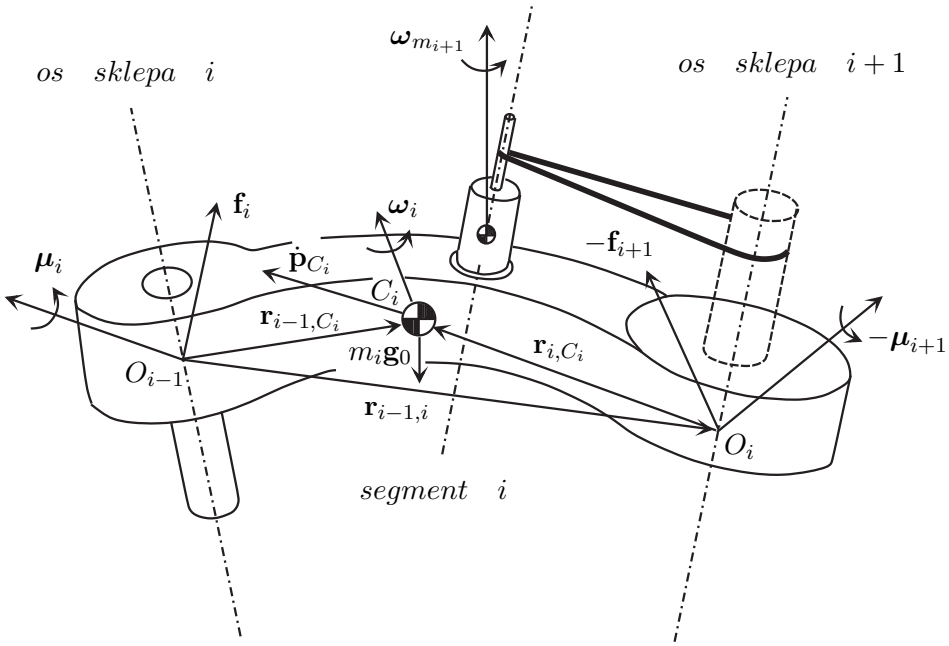
Vse dosedanje laboratorijske vaje (razen transformacij sil in navorov) so bile namenjene obravnavi kinematičnih veličin, torej pozicij (položajev), hitrosti in pospeškov robotskega mehanizma. Vendar na vsak robotski mehanizem vplivajo sile in navori, ki dejansko povzročajo gibanje in tej tematiki je namenjena zadnja vaja. Pri dinamiki robotov nas zanima premikanje oz. gibanje robotskega mehanizma kot posledica delovanja sil in navorov v posameznem sklepu robotskega mehanizma. Še bolj zanimiv je obratni oz. inverzni problem, to je kakšne sile in navore potrebujemo v sklepih robotskega mehanizma, da realiziramo željeno gibanje robota.

O dinamiki robota govorimo, ko obravnavamo sile in navore v njem in samo gibanje mehanizma, ki ga te sile in navori povzročajo. Za matematični opis dinamike se v robotiki običajno uporabljata dve metodi: Langrange-va in Newton-Eulerjeva metoda. Slednjo bomo podrobneje spoznali na laboratorijskih vajah.

1 Enačbe za izračun dinamičnega modela robota

Izhajajoč iz baze robota, najprej določimo kotne hitrosti in kotne pospeške v posameznih sklepih ter translacijske pospeške izhodišč koordinatnih sistemov pritrjenih na posamezen segment ter translacijske pospeške masnih središč posameznih segmentov. Izhajamo iz predpostavke, da sta kotna hitrost ter kotni pospešek segmenta nič (baze robota) enaka nič ($\boldsymbol{\omega}_0^0 = 0, \dot{\boldsymbol{\omega}}_0^0 = 0$). Spremenljivke sklepov so označene s

črko q in velja $q \equiv \vartheta$ za rotacijski sklep ter $q \equiv d$ za translacijski sklep. Skica 6.1 prikazuje sile in navore, ki delujejo na prosti segment.



Slika 6.1: Prosti segment.

Ker velja, da je vsota vseh sil, ki delujejo na segment enaka spremembi gibalne količine težišča segmenta, lahko zapišemo sledečo relacijo

$$\mathbf{f}_i - \mathbf{f}_{i+1} + m_i \mathbf{g}_0 = m_i \ddot{\mathbf{p}}_{C_i} \,, \tag{6.87}$$

prav tako velja, da je vsota vseh navorov, ki delujejo na segment enaka spremembi vrtilne količine segmenta okrog težišča, torej

$$\boldsymbol{\mu}_i + \mathbf{f}_i \times \mathbf{r}_{i-1,C_i} - \boldsymbol{\mu}_{i+1} - \mathbf{f}_{i+1} \times \mathbf{r}_{i,C_i} = \frac{d}{dt} \left(\bar{\mathbf{I}}_i \boldsymbol{\omega}_i \right) \,. \tag{6.88}$$

V nadaljevanju bomo določili vse kinematične veličine, ki jih potrebujemo za izračun sil in navorov. Vse izračunane veličine se nanašajo na lokalni koordinatni sistem i, zaradi česar je izračunavanje enostavnejše in preglednejše.

2 Izračun inverznega dinamičnega modela

Kotno hitrost segmenta i določimo kot

$$\textcircled{1} \quad \boldsymbol{\omega}_i^i = \begin{cases} \mathbf{R}_i^{i-1^T} \boldsymbol{\omega}_{i-1}^{i-1} & \text{za translacijski sklep} \\ \mathbf{R}_i^{i-1^T} \left(\boldsymbol{\omega}_{i-1}^{i-1} + \dot{q}_i \mathbf{z}_0 \right) & \text{za rotacijski sklep} \end{cases} \tag{6.89}$$

Kotni pospešek segmenta i določimo kot

$$\textcircled{2} \quad \dot{\boldsymbol{\omega}}_i^i = \begin{cases} \mathbf{R}_i^{i-1^T} \dot{\boldsymbol{\omega}}_{i-1}^{i-1} & \text{za translacijski sklep} \\ \mathbf{R}_i^{i-1^T} \left(\dot{\boldsymbol{\omega}}_{i-1}^{i-1} + \ddot{q}_i \mathbf{z}_0 + \dot{q}_i \boldsymbol{\omega}_{i-1}^{i-1} \times \mathbf{z}_0 \right) & \text{za rotacijski sklep} \end{cases} \tag{6.90}$$

Translacijska hitrost izhodišča koordinatnega sistema i je

$$\dot{\mathbf{p}}_i^i = \begin{cases} \mathbf{R}_i^{i-1^T} \left(\dot{\mathbf{p}}_{i-1}^{i-1} + \dot{q}_i \mathbf{z}_0 \right) + \boldsymbol{\omega}_i^i \times \mathbf{r}_{i-1,i}^i & \text{za translacijski sklep} \\ \mathbf{R}_i^{i-1^T} \dot{\mathbf{p}}_{i-1}^{i-1} + \boldsymbol{\omega}_i^i \times \mathbf{r}_{i-1,i}^i & \text{za rotacijski sklep} \end{cases} \tag{6.91}$$

Translacijski pospešek izhodišča koordinatnega sistema i je

$$\textcircled{3} \quad \ddot{\mathbf{p}}_i^i = \begin{cases} \mathbf{R}_i^{i-1^T} \left(\ddot{\mathbf{p}}_{i-1}^{i-1} + \ddot{q}_i \mathbf{z}_0 \right) + 2\dot{q}_i \boldsymbol{\omega}_i^i \times \mathbf{R}_i^{i-1^T} \mathbf{z}_0 \\ + \dot{\boldsymbol{\omega}}_i^i \times \mathbf{r}_{i-1,i}^i + \boldsymbol{\omega}_i^i \times \left(\boldsymbol{\omega}_i^i \times \mathbf{r}_{i-1,i}^i \right) & \text{za translacijski sklep} \\ \mathbf{R}_i^{i-1^T} \ddot{\mathbf{p}}_{i-1}^{i-1} + \dot{\boldsymbol{\omega}}_i^i \times \mathbf{r}_{i-1,i}^i + \boldsymbol{\omega}_i^i \times \left(\boldsymbol{\omega}_i^i \times \mathbf{r}_{i-1,i}^i \right) & \text{za rotacijski sklep} \end{cases} \tag{6.92}$$

Translacijski pospešek masnega središča segmenta je

$$\textcircled{4} \quad \ddot{\mathbf{p}}_{C_i}^i = \ddot{\mathbf{p}}_i^i + \dot{\boldsymbol{\omega}}_i^i \times \mathbf{r}_{i,C_i}^i + \boldsymbol{\omega}_i^i \times \left(\boldsymbol{\omega}_i^i \times \mathbf{r}_{i,C_i}^i \right) \tag{6.93}$$

Izračun sil in momentov v sklepih začnemo na vrhu manipulatorja, ob predpostavki, da poznamo silo in moment, ki deluje na vrh manipulatorja. Če robot ne deluje v dotiku z okolico, potem lahko predpostavimo, da velja $\mathbf{f}_{n+1}^{n+1} = 0$ ter $\mu_{n+1}^{n+1} = 0$, kjer je n število sklepov manipulatorja. Sila, ki jo izvaja segment $i+1$ na segment i (prejšnji segment izračuna na trenutno obravnavani segment – pomnite, da je smer izračuna od vrha proti bazi!) je tako enaka

$$\textcircled{5} \quad \mathbf{f}_i^i = \mathbf{R}_{i+1}^i \mathbf{f}_{i+1}^{i+1} + m_i \ddot{\mathbf{p}}_{C_i}^i \tag{6.94}$$

Moment, ki ga izvaja segment $i+1$ na segment i je

$$\textcircled{6} \quad \mu_i^i = -\mathbf{f}_i^i \times \left(\mathbf{r}_{i-1,i}^i + \mathbf{r}_{i,C_i}^i\right) + \mathbf{R}_{i+1}^i \mu_{i+1}^{i+1} + \left(\mathbf{R}_{i+1}^i \mathbf{f}_{i+1}^{i+1}\right) \times \mathbf{r}_{i,C_i}^i + \mathbf{I}_i^i \dot{\boldsymbol{\omega}}_i^i + \boldsymbol{\omega}_i^i \times \left(\mathbf{I}_i^i \boldsymbol{\omega}_i^i\right) \tag{6.95}$$

Posplošene sile, ki delujejo v posameznem sklepu so določene z enačbo

$$\textcircled{7} \quad \tau_i = \begin{cases} \mathbf{f}_i^{i\,T} \mathbf{R}_i^{i-1\,T} \mathbf{z}_0 + F_{vi}\dot{q}_i + F_{si} sgn(\dot{q}_i) & \text{za translacijski sklep} \\ \\ \mu_i^{i\,T} \mathbf{R}_i^{i-1\,T} \mathbf{z}_0 + F_{vi}\dot{q}_i + F_{si} sgn(\dot{q}_i) & \text{za rotacijski sklep} \end{cases} \tag{6.96}$$

3 Izračun direktnega dinamičnega modela

Na podlagi enačb (6.96) je mogoče ob poznavanju gibanja robota izračunati navore, ki delujejo v sklepih.

$$\boldsymbol{\tau} = \mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s \mathrm{sgn}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}). \tag{6.97}$$

Za simulacijo dinamike robota običajno potrebujemo **direktni dinamični model**, ki iz navorov, ki jih generirajo motorji robota, izračunava gibanje robota.

$$\ddot{\mathbf{q}} = \mathbf{B}^{-1}(\mathbf{q}) \left(\boldsymbol{\tau} - \left(\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s \mathrm{sgn}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q})\right)\right). \tag{6.98}$$

Z uporabo Newton-Eulerjeve formulacije je do direktnega dinamičnega modela relativno enostavno priti. Kot podlaga za izračun naj bo enačba (6.97). Stolpec $\mathbf{b}_i$ vztrajnostne matrike robota $\mathbf{B}$ lahko izračunamo pri $\mathbf{g}_0 = \mathbf{0}$, $\dot{\mathbf{q}} = \mathbf{0}$, $\ddot{q}_i = 1$ in $\ddot{q}_j = 0$, če $j \neq i$. S tem dosežemo, da so $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = 0, \mathbf{F}_v\dot{\mathbf{q}} = 0$ in $\mathbf{g}(\mathbf{q}) = 0$. Parameter statičnega trenja $\mathbf{F}_s$ ne upoštevamo. Stolpec $\mathbf{b}_i$ vztrajnostne matrike robota $\mathbf{B}$ izračunamo po naslednjem postopku.

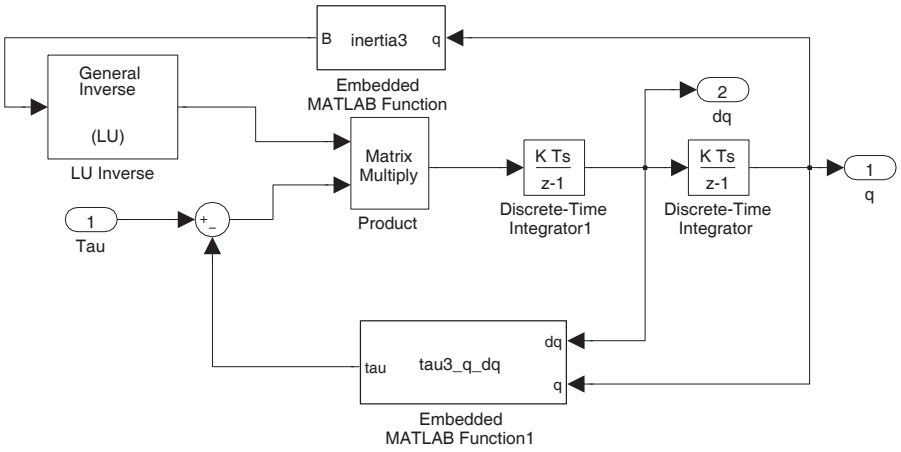
$$\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33}\end{bmatrix} \begin{bmatrix}1\\0\\0\end{bmatrix} = \begin{bmatrix}b_{11}\\b_{21}\\b_{31}\end{bmatrix} \Rightarrow \cdot \begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}b_{12}\\b_{22}\\b_{32}\end{bmatrix} \Rightarrow \cdot \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}b_{13}\\b_{23}\\b_{33}\end{bmatrix} \tag{6.99}$$

$$\text{pri } \ddot{q}_1 = 1 \qquad \text{pri } \ddot{q}_2 = 1 \qquad \text{pri } \ddot{q}_3 = 1$$

Izraz v oklepaju $(\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s + \mathbf{g}(\mathbf{q}))$ je mogoče preprosto izračunati upoštevaje $\ddot{\mathbf{q}} = \mathbf{0}$ v enačbi (6.97) ter tako dobimo:

$$\tau = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s + \mathbf{g}(\mathbf{q}). \tag{6.100}$$

Izvedbo direktnega dinamičnega modela v Simulinku prikazuje shema na sliki 6.2.



Slika 6.2: Izvedba direktnega dinamičnega modela v okolju Simulink.

4 Naloga

Izračun naj bo izveden samo za prve tri segmente robota.

- V Matlabu dopolnite funkcije `function tau = torque(ddq, dq, q, ddp0)`, `function B = inertia3(q)` in `function tau = tau3_q_dq(dq, q)`.

  Funkcija `torque(ddq, dq, q, ddp0)` za vrednosti kotov v sklepih robota zapisanih v vektorju **q**, vrednosti sklepnih hitrosti zapisanih v vektorju **dq**, vrednosti sklepnih pospeškov zapisanih v vektorju **ddqq** in vektorja pospeška, ki deluje na bazo robota **ddp0**, izračuna inverzni dinamični model robota, torej vektor sklepnih navorov **tau**.

  Funkcija `inertia3(q)` za vrednosti kotov v sklepih robota zapisanih v vektorju **q**, izračuna vztrajnostno matriko robota; pri tem uporabite kot osnovo funkcijo za izračun inverznega dinamičnega modela robota.

  Funkcija `tau3_q_dq(dq, q)` za vrednosti kotov v sklepih robota zapisanih v vektorju **q** in vrednosti sklepnih hitrosti zapisanih v vektorju **dq**, izračuna navore v sklepih robota, ki so posledica delovanja Coriolisovih in gravitacijskih komponent ter posledica trenja; pri tem uporabite kot osnovo funkcijo za izračun inverznega dinamičnega modela robota.

Vaša naloga je, da pravilno dopolnite Matlab funkcije na mestih, kjer je označeno s komentarjem `%%% STUDENT %%%`.

- V Simulinku zgradite simulacijsko shemo, ki vključuje sledeče komponente: interpolirano trajektorijo, inverzno kinematiko robota na osnovi inverzne Jacobijeve matrike, direktni dinamični model robota ter enostavni PD regulator (PD regulator je prikazan na sliki 6.3). Celotna simulacijska shema je prikazana na sliki 6.4. Simulacijsko shemo testirajte na realnem robotu Motoman.

5 Potrebno predznanje, cilj in namen naloge

Pri nalogi je zahtevano predznanje Matlab programiranja in uporaba Matlab Simulink okolja. Potrebno je tudi znanje teorije direktnega kinematičnega modela in znanje dinamike po Newton-Euler metodi.

6 Matlab predloge

Predloga funkcije **tau = torque(ddq, dq, q, ddp0)**

```
% Funkcija izracuna inverzni dinamicni model za manipulator s tremi
% rotacijskimi prostostnimi stopnjami.
% Vhodni vektorji so polozaji v sklepih q, sklepne hitrosti dq,
% sklepni pospeski ddq in vektor ddp0, ki doloca pospesek baze robota
% Izhod funkcije je vektor sklepnih navorov tau.
% Mase segmentov robota.
    m = [2.867 11.150 3.887];
% Vektorji od sklepa i-1 do sklepa i.
    r = [[0.088 0 0.131]' [0 -0.310 0]' [-0.040 0 0]'];
% Vektorji tezisc segmentov glede na lokalni koordinatni sistem.
    rC = [[-0.044 0.066 0]' [0.2 0.02 0]' [0.05 0 0]'];
% Vztrajnostne matrike segmentov.
    I = zeros(3,3,3);
    I(:,:,1) = [0.018 0 0; 0 0.023 0; 0 0 0.031];
    I(:,:,2) = [0.056 0 0; 0 0.429 0; 0 0 0.394];
    I(:,:,3) = [0.015 0 0; 0 0.012 0; 0 0 0.020];

% Parametri viskoznega trenja za posamezne sklepe.
    fv = [18.2 0.42 2.8];
% Struktura z vsemi robotskimi parametri (npr. masa prvega segmenta
% je rob.m(1)).
    rob = struct('m', m, 'rC', rC, 'r', r, 'I', I, 'fv', fv);

% Izracun transformacijskih matrik za prve tri segmente.
    a1 = 0.088;
    a2 = -0.310;
```

```
        a3 = -0.040;
        d1 = 0.131;

        A = zeros(4,4,3);
        A(:,:,1) = ;                                        %%% STUDENT %%%
        A(:,:,2) = ;                                        %%% STUDENT %%%
        A(:,:,3) = ;                                        %%% STUDENT %%%

    % Vektor v smeri osi z0.
        z0 = [0 0 1]';
    % Inicializacija vektorja navora.
        tau = zeros(3,1);
    % Dolocitev rotacijskih matrik kot podmatrik matrik A.
        R = zeros(3,3,3);
        R(:,:,1) = ;                                        %%% STUDENT %%%
        R(:,:,2) = ;                                        %%% STUDENT %%%
        R(:,:,3) = ;                                        %%% STUDENT %%%
        ...
```

```
        % Izracun kinematicnih velicin.
        for ii = 1:3
            if (ii == 1) % Robni pogoj za prvi sklep.
                % Kotna hitrost segmenta.
(1)                 omega(:,ii) = ;                         %%% STUDENT %%%
                % Kotni pospesek segmenta.
(2)                 domega(:,ii) = ;                        %%% STUDENT %%%
                % Translacijski pospesek segmenta.
(3)                 ddp(:,ii) = ;                           %%% STUDENT %%%
            else
                % Kotna hitrost segmenta.
(1)                 omega(:,ii) = ;                         %%% STUDENT %%%
                % Kotni pospesek segmenta.
(2)                 domega(:,ii) = ;                        %%% STUDENT %%%
                % Translacijski pospesek segmenta.
(3)                 ddp(:,ii) = ;                           %%% STUDENT %%%
            end
            % Translacijski pospesek tezisca segmenta.
(4)              ddpC(:,ii) = ;                             %%% STUDENT %%%
        end

        % Izracun dinamicnih velicin.
        for ii = 3:-1:1
            if (ii==3) % Robni pogoj za silo na vrhu.
                % Sila, ki deluje na sklep.
(5)                 f(:,ii) = ;                             %%% STUDENT %%%
                % Navor, ki deluje na sklep.
(6)                 mi(:,ii) = ;                            %%% STUDENT %%%
            else
                % Sila, ki deluje na sklep.
(5)                 f(:,ii) = ;                             %%% STUDENT %%%
                % Navor, ki deluje na sklep.
(6)                 mi(:,ii) = ;                            %%% STUDENT %%%
            end
            % Navor, ki deluje v osi sklepa (obremenjuje motor).
(7)              tau(ii) = ;                                %%% STUDENT %%%
        end
```

Testni rezultat:

$$torque([1,2,3]', [4,5,6]', [7,8,9]', [0,0,9.81]') = \begin{bmatrix} 79.1549 & 30.4636 & 16.1794 \end{bmatrix}^T$$

Predloga funkcije **B = inertia3(q)**

```
% Funkcija izracuna vztrajnostno matriko za robot s tremi segmenti.
% Vhod je vektor kotov v sklepih q.

% Inicializacijs vztrajnostne matrike.
    B = zeros(3,3);

% Upostevamo pogoj za pospesek, ki deluje na bazo robota.
    ddp0 = ;                                          %%% STUDENT %%%

% Upostevamo pogoj za kotne pospeske v sklepih.
    dq = ;                                            %%% STUDENT %%%


% Izracunamo prvi stolpec vztrajnostne matrike B, pri cemer uporabimo
% funkcijo za izracun inverzne dinamike robota za izracun navorov,
% ki predstavljajo stolpec matrike vztrajnosti.
    ddq = ;                                           %%% STUDENT %%%
    B(:,1) =;                                         %%% STUDENT %%%

% Izracunamo drugi stolpec vztrajnostne matrike B.
    ddq = ;                                           %%% STUDENT %%%
    B(:,2) = ;                                        %%% STUDENT %%%

% Izracunamo tretji stolpec vztrajnostne matrike B.
    ddq = ;                                           %%% STUDENT %%%
    B(:,3) = ;                                        %%% STUDENT %%%
```

Testni rezultat:

$$inertia3([1,2,3]') = \begin{bmatrix} 1.2369 & 0.4086 & 0.0014 \\ 0.4086 & 2.1602 & 0.0107 \\ 0.0014 & 0.0107 & 0.0124 \end{bmatrix}$$

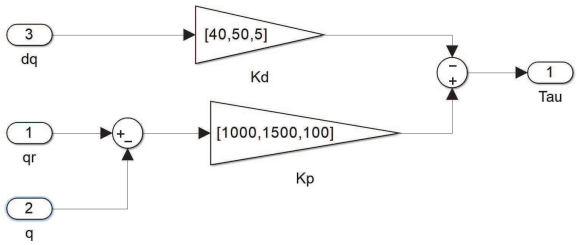
Predloga funkcije **tau = tau3_q_dq(dq, q)**

```
function tau = tau3_q_dq(dq, q)

    % Funkcija izracuna navore v sklepih robota, ki so posledica delovanja
    % Coriolisovih in gravitacijskih komponent ter posledica trenja.
    % Vhodni vektorji so polozaji v sklepih q in sklepne hitrosti dq.
    % Izhod funkcije je vektor sklepnih navorov tau.

    % Upostevamo pogoj za pospesek, ki deluje na bazo robota.
        ddp0 = ;                                      %%% STUDENT %%%

    % Upostevamo pogoj za kotne pospeske v sklepih.
        ddq = ;                                       %%% STUDENT %%%

    % Uporabimo funkcijo za izracun inverzne dinamike robota
    % za izracun navorov.
        tau = ;                                       %%% STUDENT %%%
```

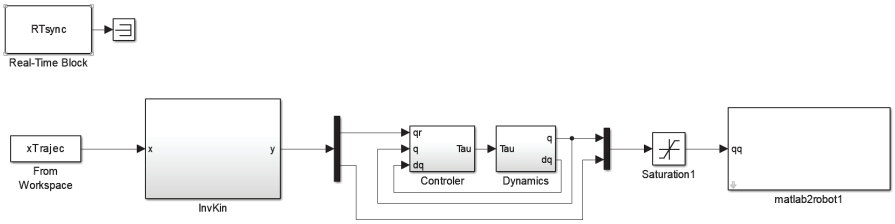
Testni rezultat:

$$tau3\_q\_dq([1,2,3]',[4,5,6]') = \begin{bmatrix} 24.1348 & -34.1199 & 8.0650 \end{bmatrix}^T$$

7 Simulacijske sheme v Matlab Simulink okolju



Slika 6.3: Enostavni PD regulator. qr predstavlja referenčni položaj, q dejanski položaj in dq dejansko hitrost robota. Kp in Kd sta ojačenji položaja in hitrosti.



Slika 6.4: Simulacijska shema, ki vključuje vse komponente: interpolirano trajektorijo, inverzno kinematiko robota na osnovi inverzne Jacobijeve matrike, direktni dinamični model robota ter enostavni PD regulator.

Kje se nahajajo posamezni Simulink bloki:

Embadded MATLAB Func. ⇒ *Simulink* → *User-Defined Func.*
Matrix Multiply ⇒ *Simulink* → *Math Operations*
Descrete-Time Integrator ⇒ *Simulink* → *Discrete*
Gain value: 1.0
Initial condition: [0 0 0] in q0(1:3)
Sample time: dT
LU Inverse ⇒ *Signal Processing Blockset* → *Math Functions*

Za odpiranje predloge Simulink sheme v Matlab ukazni vrstici zaženite program *initVR.m*!

Spremenjeno shemo shranite pod drugim imenom!

DOMAČE NALOGE

Domače naloge so namenjene sprotni pripravi študenta na laboratorijske vaje, saj so vse tako zasnovane. Asistent bo naključno pregledal domače naloge študentov in tudi zastavil kakšno vprašanje povezano z rezultatom. Domača naloga se šteje kot sprotno delo in se vključi v ocenjevanje nalog *za trening*, torej v 15 %. Ocene so znakovne: $+$, $+\bigcirc$, $\bigcirc$, $\bigcirc-$, $-$.

Če študent:

- nima domače naloge,
- asistent ugotovi, da jo je prepisal,
- znanje snovi ni zadovoljivo,

potem študent ni dovolj pripravljen za delo na laboratorijskih vajah in mora le-te zapustiti. Tako je trenuten termin vaj ocenjen negativno.

Naloge lahko rešujete z ali brez uporabe okolja Matlab. Postopek izračuna in rezultate domače naloge zapišite v prazen prostor pod besedilo naloge; pod Rezultati.

1 Geometrijski model robotov

Besedilo naloge

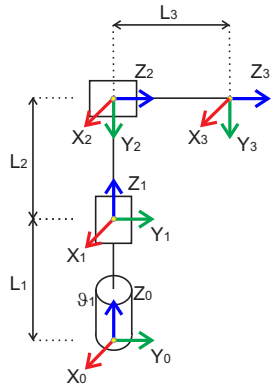
Naloga, ki jo morate narediti doma pred 1. laboratorijskimi vajami, je študij izdelave kinematičnega modela robota po metodi Denavit-Hartenberg. Glede na sliko 1.3 zapolnite tabelo 1.1, ki je predpogoj za uspešno izpeljavo 1. laboratorijske vaje z naslovom *Geometrijski model robotov*.

Rezultati

Tabelo lahko prerišete na to stran ali pa jo zapolnite kar na začetku navodil.

2 Geometrijska Jacobijeva matrika

Besedilo naloge



Slika 7.1: Robot v referenčni legi s koordinatnimi sistemi

Glede na robot na sliki 7.1 zapišite matrike A_1 (lega 1. k.s. glede na bazni k.s.), A_2 (lega 2. k.s. glede na 1. k.s.), A_3 (lega 3. k.s. glede na 2. k.s.) in T_3 (lega k.s. na vrhu glede na bazni k.s.), pri tem pa upoštevajte naslednje parametre $L_1 = 0.1$, $L_2 = 0.4$, $L_3 = 0.2$, $\vartheta_1 = 0$, $d_2 = 0$ in $d_3 = 0$. Iz zapisanih podatkov v matrikah zapišite še geometrijsko Jacobijevo matriko (J) za predstavljen manipulator v podani legi.

Rezultati

3 Relacija med geometrijsko in analitično Jacobijevo matriko ter inverzna kinematika z uporabo analitične Jacobijeve matrike

Besedilo naloge

- Za robot na sliki 7.1 izračunajte ZYX Eulerjeve kote $(\varphi, \vartheta, \psi)$ pri začetnih pogojih iz prejšnje domače naloge.
- Izračunajte oz. zapišite matriko $T(\phi)$, ki je podmatrika matrike $T_A(\phi)$, in to za Eulerjeve kote izračunane v točki 1 te domače naloge. Matrika $T_A(\phi)$ je transformacijska matrika med analitično in geometrijsko Jacobijevo matriko.
- V navodilih za laboratorijske vaje je na sliki 3.4 skica inverzne kinematike. Za lažje in hitrejše delo na laboratorijskih vajah skicirajte dejansko Simulink shemo z vsemi pravimi povezavami, bloki pa so lahko samo ustrezno označeni in jih boste na laboratorijskih vajah ustrezno izbrali iz Simulink knjižnice.

Rezultati

4 Transformacija sil in navorov

Besedilo naloge

Na robot s slike 7.1 deluje sila v smeri osi Z koordinatnega sistema vrha z velikostjo 10 N. S teorijo opisano v navodilih za laboratorijske vaje izračunajte silo izraženo glede na bazni koordinatni sistem. Rezultat naj bo podan v obliki vektorja sil in navorov. Uporabite parametre iz 2. domače naloge!

Rezultati

5 Generiranje trajektorij

Besedilo naloge

Opraviti imamo z mehanizmom z eno aktivno translacijsko prostostno stopnjo. Pogon v obliki polža je voden po trapeznem hitrostnem profilu. Začetna pozicija polža je 0,0 m. Ob pričetku premikanja polža je njegov pospešek 0,1 m/s^2 in traja do željene končne hitrosti 0,5 m/s. Polž doseže polovico željenega premika v času 10 sekund.

Izračunajte naslednje parametre premika:

- V kolikšnem času polž doseže 1/3 poti pospeševanja?
- Kakšno pot opravi polž po končani fazi enakomernega gibanja?
- Po kolikšnem času se polž ustavi in kakšen premik je naredil?

Rezultati

6 Newton-Eulerjeva dinamika

Besedilo naloge

- Izračunajte kotno hitrost 1., 2. in 3. sklepa (ω_1^1, ω_2^2 in ω_3^3) pri čemer je hitrost 1. sklepa ($\dot{q}_1$) enaka 0,3. Parametri robota so podani v 2. domači nalogi.
- Izračunajte silo (f_3^3), ki deluje na sklep 3, če je masa segmenta 3 enaka 0,3 in translacijski pospešek masnega središča segmenta 3 znaša [0,3 0,8 0,06]. Sila na vrhu robota je enaka 0.

Rezultati