UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

DRUGI DOMAĆI ZADATAK IZ PREDMETA ROBOTIKA I AUTOMATIZACIJA

Studenti:

Uroš Pantelić 2021/0073

Lena Rašević 2021/0370

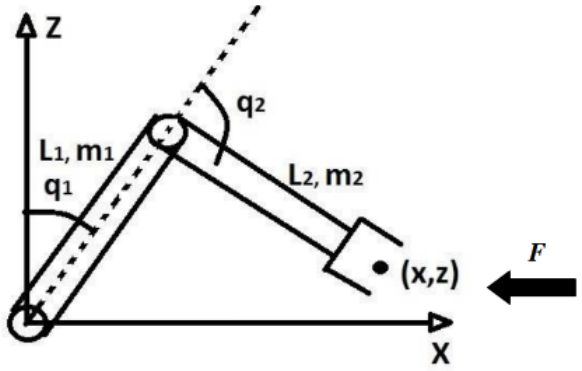
Beograd, jul 2024

Contents

1 Postavka problema

Robot se sastoji od dva rotaciona zgloba i dva segmenta jednakih dužina i masa m. Potrebno je realizovati kretanje robota između 3 tačke u prostoru (A-B-C). Od tačke A do tačke B potrebno je obezbediti kretanje po glatkoj putanji gde se unutrašnje koordinate robota menjaju po profilu koji odgovara polinomu trećeg stepena tokom prvih 0.5T kretanja. Između tačaka B i C potrebno je obezbediti kretanje po principu point-to-point kretanja, tokom ostatka vremena od 0.5T. Pretpostaviti da se u početnom trenutku robot nalazio u fazi mirovanja i da je pozitivni referentni smer kretanja usvojen smer kretanja kazaljke na satu (Slika 1).

Analizirati kretanje robotskog sistema ako na vrh hvataljke tokom kretanja robota deluje spoljna sila F=-2N duž X ose tokom intervala 0.35T-0.4T. Realizovati funkcije koje izračunavaju kinematiku i dinamiku robota, planiranje trajektorije kao i numeričku integraciju opisanog robotskog sistema tokom vremena T koji koristi momente kao upravljačke veličine.



Slika 1. Robot u ravni sa dva stepena slobode i odgovarajućim referentnim smerovima

Parametri za simulaciju

- dužine segmenata – $l = 0.3$ m
- mase segmenata – $m = 2$ kg
- inicijalna pozicija hvataljke je A: $(q_{A,1}, q_{A,2}) = (45, 45)$[deg]
- željena pozicija hvataljke je B: $(X_B, Z_B) = (0.3, 0)$ m
- željena pozicija tačke C je: $(X_C, Z_C) = (0.3\sqrt{3}, 0)$ m
- trajanje simulacije – $T = 1$s
- tip simulacije - decentralizovano upravljanje

2 Rešenje

U ovom delu dokumenta opisujemo naš pristup rešavanju zadatka, uključujući implementaciju forward kinematike i određivanje položaja A, B i C robota.

2.1 Implementacija Forward Kinematike

Prvi korak u rešavanju zadatka bio je implementacija forward kinematike kako bi se odredile pozicije krajnjeg dela robota (hvataljke). Koristeći matematički model robota sa dva rotaciona zgloba i segmentima jednakih dužina, izračunali smo koordinate x i y hvataljke u prostoru u zavisnosti od položaja zglobova q1 i q2.

```
function [X, dX] = forward_kinematics(q, dq, J)
    global l

    % Cartesian position
    x = l * sin(q(1)) + l * sin(q(1) + q(2));
    z = l * cos(q(1)) + l * cos(q(1) + q(2));
    X = [x; z];

    % Cartesian velocity
    dX = J * dq;
end
```

MATLAB kod za forward kinematiku

2.2 Određivanje Položaja A, B i C

Nakon implementacije kinematike, sledeći korak bio je određivanje specifičnih tačaka kretanja robota: A, B i C.

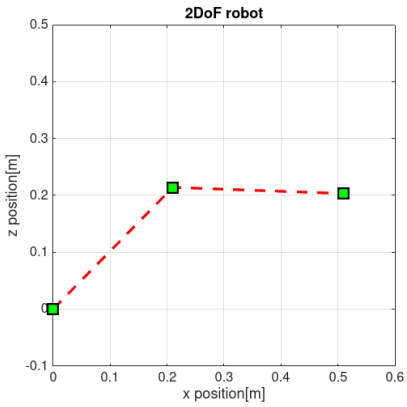
- **Pozicija A:** Početna pozicija robota bila je definisana sa $qA = [\frac{\pi}{4}, \frac{\pi}{4}]$.
- **Pozicija B:** Željena pozicija B bila je postavljena na $qB\_stop = [30°, 120°]$.
- **Pozicija C:** Krajnja pozicija C bila je postavljena na $qC\_stop = [\frac{\pi}{3}, \frac{\pi}{3}]$, što je predstavljalo point-to-point kretanje između B i C tokom drugog dela simulacije.

```matlab
function [q1, q2] = inverse_kinematics(X, Z)
    global l

    q_guess = [0; 0];  % Initial guess for q1 and q2

    % Define function for fsolve
    fun = @(q) forward_kinematics(q, [0; 0], eye(2)) - [X; Z];

    % Solve inverse kinematics using fsolve
    q_sol = fsolve(fun, q_guess);

    % Extract joint angles q1 and q2
    q1 = q_sol(1);
    q2 = q_sol(2);
end
```

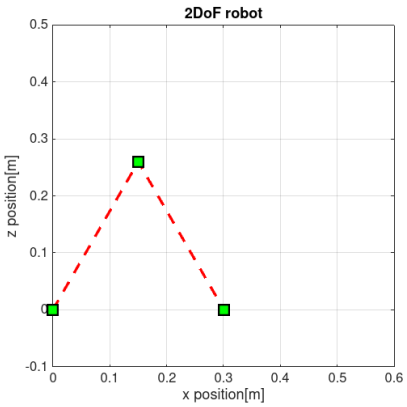
MATLAB kod za inverznu kinematiku

2.3 Vizualizacija Položaja

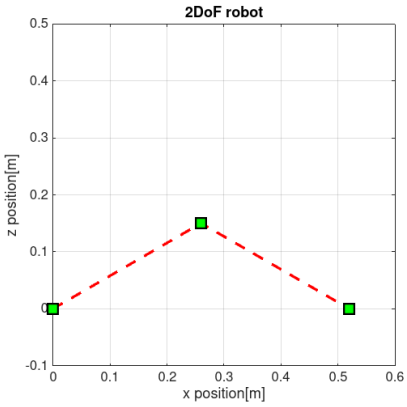
Za vizualizaciju pozicija A, B i C, priložene su slike koje prikazuju položaje robota u prostoru tokom simulacije.



$qA = [45°, 45°]$



$qB\_stop = [30°, 120°]$



$qC\_stop = [60°, 60°]$

2.4 Planiranje trajektorije

2.4.1 Trajektorija od tačke A do tačke B

Polazna osnova za generisanje trajektorije su početna tačka q_A i krajnja tačka q_B putanje, i ukupno trajanje pokreta T. Prema tome, imamo zadate četiri granične uslova, od kojih se dva odnose na početnu tačku putanje:

$$q(0) = q_A, \quad \dot{q}(0) = 0$$

i dva koja se odnose na krajnju tačku putanje:

$$q\left(\frac{T}{2}\right) = q_B, \quad \dot{q}\left(\frac{T}{2}\right) = 0$$

Zadatak je pronaći glatku interpolacionu funkciju koja određuje pozicije tokom vremena između početne i krajnje tačke koristeći interpolaciju polinomom trećeg stepena. Uvedimo normalizaciju po vremenu, uvođenjem zamene:

$$\tau = \frac{2t}{T}$$

Čime se realno vreme t iz intervala $[0, \frac{T}{2}]$ preslikava u normalizovano vreme τ na normalizovanom intervalu $\tau \in [0, 1]$. Uvedimo zatim i normalizovanu poziciju p tako da se interval promene pozicije q preslikava na normalizovani interval:

$$p = \frac{q - q_A}{q_B - q_A}$$

Nakon ovih normalizacija, pozicija zgloba može se odrediti na osnovu:

$$q(t) = q_A + (q_B - q_A)p(\tau)$$

Diferenciranjem izraza možemo odrediti brzinu zgloba u funkciji normalizovane brzine:

$$\dot{q}(t) = \frac{2(q_B - q_A)}{T} p'(\tau)$$

Slično, ubrzanje zgloba u funkciji normalizovanog ubrzanja:

$$\ddot{q}(t) = \frac{4(q_B - q_A)}{T^2} p''(\tau)$$

Obzirom na to da imamo četiri zadate uslove, na osnovu njih možemo odrediti polinom trećeg stepena, koji ima ukupno četiri koeficijenta, u obliku:

$$p(\tau) = a_3\tau^3 + a_2\tau^2 + a_1\tau + a_0$$

U nastavku ide primer implementacije u MATLAB-u za planiranje trajektorije od tačke A do tačke B:

```
function [q_ref, dq_ref, ddq_ref] = planiranje_trajektorije(dt, T,
    startValue1,startValue2, stopValueB1,stopValueB2, stopValueC1,stopValueC2)

% pocetni i krajnji polozaji tokom kretanja
qi = [startValue1; startValue2];    % tacka A
qB = [stopValueB1; stopValueB2];    % tacka B
qC = [stopValueC1; stopValueC2];    % tacka C

% vremenski parametri
Tf = 0.5*T;

% A-B: interpolacija polinomom treceg stepena
time = linspace(0, Tf, Tf/dt);

% koeficijenti polinoma
a0 = qi;
a1 = [0; 0];
a2 = 3 * (qB - qi) / (T/2)^2;
a3 = -2 * (qB - qi) / (T/2)^3;

q_ref_ab = zeros(2, length(time));
dq_ref_ab = zeros(2, length(time));
ddq_ref_ab = zeros(2, length(time));

for j = 1:2
    q_ref_ab(j, :) = a3(j) * time.^3 + a2(j) * time.^2 + a1(j) * time + a0(j);
    dq_ref_ab(j, :) = 3 * a3(j) * time.^2 + 2 * a2(j) * time + a1(j);
    ddq_ref_ab(j, :) = 6 * a3(j) * time + 2 * a2(j);
end

\newpage
\vspace*{-3cm}
```

MATLAB kod za kretanje od A do B

2.4.2 Kretanje od tačke B do tačke C

Point-to-point kretanje predstavlja skokovitu promenu reference pozicije, bez zadavanja brzine i ubrzanja. Specifično je po tome što se definišu samo početna i krajnja tačka putanje, dok se putanja između njih ne precizira unapred.

```
% B-C: point to point
q_ref_bc = repmat(qC, 1, length(time));
dq_ref_bc = zeros(2, length(time));
ddq_ref_bc = zeros(2, length(time));
```

MATLAB kod za kretanje od B do C

2.4.3 Kombinovanje referentnih vrednosti

Kombinovanjem referentnih vrednosti za kretanje od tačke A do tačke B i od tačke B do tačke C dobija se trajektorija cele putanje:

```
% Kombinacija referentnih vrednosti za ukupno kretanje od A do B do C
q_ref = [q_ref_ab q_ref_bc];
dq_ref = [dq_ref_ab dq_ref_bc];
ddq_ref = [ddq_ref_ab ddq_ref_bc];
```

MATLAB kod za kombinovanje kretanja od A do C

2.5 Dinamika robota

U opštem obliku, jednačina dinamike robotskog sistema se može predstaviti preko pojedinih komponenti momenta koje uravnotežuje pogonski moment τ:

$$\tau = H(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q)$$

Onaj deo izraza za pogonski moment koji se sadrži u izrazu $H(q)\ddot{q}$ jeste moment koji potiče od inercijalnih efekata, deo pogonskog momenta $C(q, \dot{q})\dot{q}$ potiče od brzinskih efekata, dok poslednji član $G(q)$ predstavlja gravitaciono opterećenje u zglobovima.

Funkcija 'matrix_dyn' izračunava matricu inercije H, matricu brzinskih efekata C i vektor gravitacionih efekata G na osnovu trenutnih pozicija q i brzina $\dot{q}$.

```
function [H, C, G] = matrix_dyn(q, dq)

global m l I g

% Matrica inercije H
H11 = m*l^2 + m*(l/2)^2 + m*(l/2)^2 + 2*l*(l/2)*m*cos(q(2)) + I;
H12 = (l/2)^2*m + l*(l/2)*m*cos(q(2));
H21 = H12;
H22 = I + 0.25*m*l^2;

H = [H11, H12; H21, H22];

% Matrica brzinskih efekata C
C11 = -m*l^2*sin(q(2))*dq(2);
C12 = -0.5*m*l^2*sin(q(2))*dq(2);
C21 = 0.5*m*l^2*sin(q(2))*dq(1);
C22 = 0;

C = [C11, C12; C21, C22];

% Gravitaciona matrica G
G11 = -1.5*m*l*g*cos(q(1)) - 0.5*m*l*g*cos(q(1) + q(2));
G21 = 0.5*m*l*g*cos(q(1) + q(2));

G = [G11; G21];
```

MATLAB kod za dinamičke matrice robota

2.6 Upravljanje kretanja - Decentralizovano upravljanje

Da bismo ostvarili decentralizovani tip kretanja robota, razvili smo posebne kontrolere za svaki zglob. Svaki od ovih kontrolera koristi PD (Proporcionalno-Derivativnu) kontrolu kako bi pratili zadate reference pozicije i brzine.U ovom pristupu, svaki zglob se kontroliše nezavisno koristeći PD (Proporcionalno-Derivativnu) kontrolu. Matematički izraz za ukupni moment u okviru decentralizovanog upravljanja za svaki zglob je:

$$\tau_j = G_j + K_{p,j}(q_{ref,j} - q_j) + K_{d,j}(\dot{q}_{ref,j} - \dot{q}_j)$$

Za realizaciju decentralizovanog PD kontrolera, koristimo funkciju koja izračunava ukupni moment za svaki zglob na osnovu zadatih referentnih vrednosti i trenutnih stanja.

```
function [Tau, Tau_FF, Tau_FB] = calculate_Tau(q, q_ref, dq, dq_ref, Kp, Kd, G)
    Tau = zeros(2, 1);
    Tau_FF = zeros(2, 1);
    Tau_FB = zeros(2, 1);

    for j = 1:2
        % Feedforward control - gravity compensation
        Tau_FF(j) = G(j);

        % Feedback PD control
        error_q = q_ref(j) - q(j);
        error_dq = dq_ref(j) - dq(j);
        Tau_FB(j) = Kp(j) * error_q + Kd(j) * error_dq;

        % Total control torque
        Tau(j) = Tau_FF(j) + Tau_FB(j);
    end
end
```

MATLAB kod za izračunavanje momenta za svaki zglob

2.7 Glavni program

U glavnom programu simulacije, inicijalizujemo parametre, postavljamo početne uslove, planiramo trajektoriju i implementiramo petlju kontrole. Kod se izvodi sledećim redosledom:

2.7.1 Inicijalizacija parametara simulacije

Prvi deo koda postavlja parametre simulacije, uključujući početne uslove, vremenske korake i gravitacionu konstantu.

```
clear all; close all; clc
global m I l l1 l2 g Fint Tau
%% simulation parameters
t = 0;           % simulation time initialization
dt = 0.001;      % simulation step
T = 10;          % total simulation time
lengthT = T/dt; % number of simulation steps
i = 1;           % simulation step
g = 9.81;        % gravity acceleration
Fint = [-2; 0]; % interaction force - external force
```

MATLAB kod za inicijalizaciju parametara simulacije

2.7.2 Kinematski i dinamički parametri

Sledeći deo koda definiše kinematske i dinamičke parametre robota, kao što su dužine linkova, mase i momenti inercije.

```
%% kinematics and dynamics parameters
l1 = 0.3;        % [m] length of the 1st link
l2 = 0.3;        % [m] length of the 2nd link
m1 = 2;          % [kg] mass of the 1st link
m2 = 2;          % [kg] mass of the 2nd link
I1 = 0.03;       % [kgm2] moment of inertia of the 1st link
I2 = 0.03;       % [kgm2] moment of inertia of the 2nd link
l = l1; m = m1; I = I1;
```

MATLAB kod za kinematske i dinamičke parametre

2.7.3 Početni uslovi

Ovaj deo koda postavlja početne uslove za pozicije, brzine i ubrzanja zglobova robota.

```
%% initial conditions
q = [pi/4; pi/4]; % initial position
dq = [0; 0];      % initial velocity
ddq = [0; 0];     % initial acceleration
```

MATLAB kod za početne uslove

2.7.4 Planiranje trajektorije

U ovom delu se generišu referentne vrednosti pozicija i brzina zglobova.

```
%% trajectory planning
% calculate reference joint positions and velocities - trapezoidal profile
q_init = q;                          % start position
qB_stop = [30*pi/180; 120*pi/180];  % stop position
qC_stop = [pi/3; pi/3];
[q_ref, dq_ref, ddq_ref] = planiranje_trajektorije(dt, T, q_init(1), q_init(2),
    qB_stop(1), qB_stop(2), qC_stop(1), qC_stop(2));
```

MATLAB kod za planiranje trajektorije

2.7.5 Postavljanje parametara kontrolera

Postavljaju se proporcionalni i derivativni koeficijenti za PD kontrolu.

```
%% controller
Kp = [100; 100]; % proportional gain for each joint
Ki = [0; 0];     % integration gain for each joint (not used in PD control)
Kd = [10; 10];   % derivative gain for each joint
```

MATLAB kod za postavljanje parametara kontrolera

2.7.6 Glavna petlja simulacije

Glavna petlja simulacije se izvodi za definisani broj koraka, izračunavajući trenutne momente, pozicije i brzine zglobova.

- **while petlja** se izvršava dok trenutni trenutak t ne dostigne ukupno vreme simulacije T. Ovo omogućava da se simulacija odvija u diskretnim koracima vremena.
- **Vremenski korak** dt je postavljen na 0.001 sekundi. Ovaj vremenski korak određuje koliko se simulacija pomera napred u svakom iteracijskom koraku. Ukupno trajanje simulacije je postavljeno na 1 sekund.
- **Interakcijska sila** $Fint$ se primenjuje na sistem između 35% i 40% ukupnog vremena simulacije ($0.35 \times T$ i $0.4 \times T$). Van ovog intervala, interakcijska sila je nula.
- **Kinematika i dinamika** se izračunavaju za trenutne vrednosti pozicija zglobova q i referentnih vrednosti q_{ref}. Matrica Jacobi J i J_{ref} se koriste za izračunavanje napredne kinematike.
- **Matrice dinamike** $H_kappa, C_kappa, G_kappa$ se izračunavaju za trenutne referentne vrednosti pozicija i brzina q_{ref} i dq_{ref}.
- **Kontrolni moment** τ se izračunava kao suma feedforward momenta τ_{FF} i feedback momenta τ_{FB}. Feedforward moment uzima u obzir gravitacione efekte, dok feedback moment koristi PD kontrolu za praćenje referentnih vrednosti.
- **Numerička integracija** se izvodi korišćenjem funkcije `ode45` koja rešava diferencijalne jednačine sistema za trenutni korak vremena. Ova funkcija vraća nove vrednosti pozicija i brzina zglobova za sledeći vremenski korak.
- **Ažuriranje pozicija i brzina** se vrši na kraju svakog iteracijskog koraka, čime se simulacija pomera napred za jedan vremenski korak dt.

```
while (t<T)
    t = i*dt;

   % Delovanje spoljne sile
   if t > 0.35*T && t < 0.4*T
        Fint=[-2;0];
   else
        Fint=[0;0];
   end
   J = matrix_kin(q,l); % calculates J
   J_ref = matrix_kin(q_ref(:,i),l);
   [X, dX] = forward_kinematics(q, dq, J); % q -> X
   [X_ref, dX_ref] = forward_kinematics(q_ref(:,i), dq_ref(:,i), J_ref); % q_ref
       -> X_ref

   [H_kappa, C_kappa, G_kappa] = matrix_dyn(q_ref(:, i), dq_ref(:, i)); %
       calculates H, C, G
    q_ref_curr = q_ref(:, i);
    dq_ref_curr = dq_ref(:, i);
    ddq_ref_curr = ddq_ref(:, i);
   [Tau, Tau_FF, Tau_FB] = calculate_Tau(q, q_ref_curr, dq, dq_ref_curr, [Kp;Kp],
       [Kd;Kd], G_kappa, i); % Tau = Tau_FF + Tau_FB
   Tau = Tau - J'*Fint;
   Q_4 = [q; dq];
   options = odeset('RelTol',1e-2,'AbsTol',1e-3,'MaxOrder',3);
   [tout,Q_4_out] = ode45(@int_2DoF,[t t+dt], Q_4, options);
   Q_4 = Q_4_out(end,:)';
   % size(Q_4_out);
   q = Q_4(1:2);
   dq = Q_4(3:4);
```

MATLAB kod glavne petlje

Glavna petlja simulacije se izvodi $lengthT$ puta (gde je $lengthT = T/dt$), što znači da će simulacija imati ukupno 1000 iteracija (jer je $T = 1$ sekundi, a $dt = 0.001$ sekundi).

2.7.7 Vizualizacija

Ovaj deo koda prikazuje trenutni položaj robota svakih 20 iteracija.

```
    % visualisation
    if mod(i, 20) == 0
        figure(1)
        plot([0 l1 * sin(q(1)) l1 * sin(q(1)) + l2 * sin(q(1) + q(2))], [0 l1 *
            cos(q(1)) l1 * cos(q(1)) + l2 * cos(q(1) + q(2))],...
             '--rs', 'LineWidth', 2, 'MarkerEdgeColor', 'k', 'MarkerFaceColor',
                 'g', 'MarkerSize', 10)
        axis equal;
        axis([0 0.6 -0.1 0.5]); % defines axes regardless of the number of
            coordinates
        title('2DoF robot')
        ylabel('z position [m]')
        xlabel('x position [m]')
        grid
        pause(0.01)
    end

    write_in_memory;
    i = i + 1;
end
```

MATLAB kod za vizuelizaciju

2.7.8 Prikaz rezultata

Na kraju, prikazuju se rezultati simulacije.

```
display_results
```

MATLAB kod za prikaz rezultata

Kod glavnog programa simulacije uključuje sledeće korake:

- **Inicijalizacija parametara simulacije**: Postavljaju se početni uslovi i trajanje simulacije.
- **Planiranje trajektorije**: Generišu se referentne vrednosti pozicija i brzina zglobova.
- **Kontroler**: Postavljaju se proporcionalni i derivativni koeficijenti za PD kontrolu.
- **Glavna petlja**: U ovoj petlji se izračunavaju trenutni momenti, pozicije i brzine zglobova, dok se simulacija izvodi za definisani broj koraka.
- **Vizualizacija**: Prikaz trenutnog stanja robota.

2.8 Funkcija `int_2DOF`

Funkcija `int_2DoF` definiše dinamički model 2-DoF robota manipulatora za numeričku integraciju tokom simulacije.Izračunava se trenutna dinamika sistema na osnovu pozicija i brzina zglobova, primenjenih momenata i spoljašnjih sila.

```
function dydt_Q_4 = int_2DoF(t, Q_4)
    global l Fint Tau

    q = Q_4(1:2);
    dq = Q_4(3:4);

    J = matrix_kin(q, l);
    [H, C, G] = matrix_dyn(q, dq);

    dydtQ_2 = H \ (Tau + J' * Fint - C * dq - G);

    dydt_Q_4(1:2) = dq;
    dydt_Q_4(3:4) = dydtQ_2;
    dydt_Q_4 = dydt_Q_4';
end
```

MATLAB kod funkcije int_2DOF

2.9 Funkcija `write_in_memory`

Funkcija `write_in_memory` ažurira strukturu `Ps` tokom simulacije, čuvajući trenutna vremena, pozicije, brzine, referentne vrednosti, momente i sile za dalju analizu i vizualizaciju.

```
function write_in_memory
    global Ps t q dq X X_ref q_ref dq_ref Tau Fint Tau_FF Tau_FB

    Ps.t(end+1) = t;
    Ps.q(:,end+1) = q .* (180/pi);
    Ps.dq(:,end+1) = dq .* (180/pi);
    Ps.X(:,end+1) = X;
    Ps.X_ref(:,end+1) = X_ref;
    Ps.q_ref(:,end+1) = q_ref .* (180/pi);
    Ps.dq_ref(:,end+1) = dq_ref .* (180/pi);
    Ps.Tau(:,end+1) = Tau;
    Ps.Fint(:,end+1) = Fint;
    Ps.Tau_FF(:,end+1) = Tau_FF;
    Ps.Tau_FB(:,end+1) = Tau_FB;
end
```

MATLAB kod funkcije write_in_memory

Ove funkcije su ključne za praćenje stanja sistema tokom simulacije i omogućavaju detaljnu analizu rezultata nakon izvršenja simulacije.

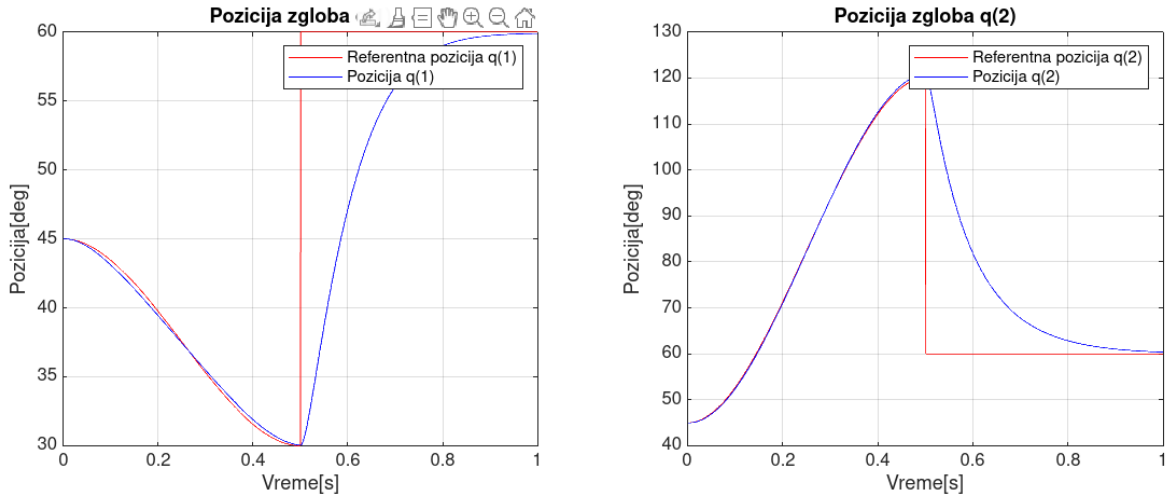
3 Dobijeni rezultati

Na kraju glavnog programa se nalazi komanda `display_results` koja prikazuje sledeće vrednosti:

3.0.1 Unutrašnje koordinate q

```
%% internal coordinates q
figure('Position',[50 400 700 250]);
for m1 = 1:2
    subplot(1,2,m1)
    switch m1
        case 1
            plot(dt:dt:T,Ps.q_ref(1,:),'r',dt:dt:T,Ps.q(1,:),'b')
            grid
            title('Pozicija zgloba q(1)')
            ylabel('Pozicija[deg]')
            xlabel('Vreme[s]')
            legend('Referentna pozicija q(1)','Pozicija
                q(1)','Location','NorthEast')
        case 2
            plot(dt:dt:T,Ps.q_ref(2,:),'r',dt:dt:T,Ps.q(2,:),'b')
            grid
            title('Pozicija zgloba q(2)')
            ylabel('Pozicija[deg]')
            xlabel('Vreme[s]')
            legend('Referentna pozicija q(2)','Pozicija
                q(2)','Location','NorthEast')
    end
end
```

MATLAB kod za prikaz unutrašnjih koordinata

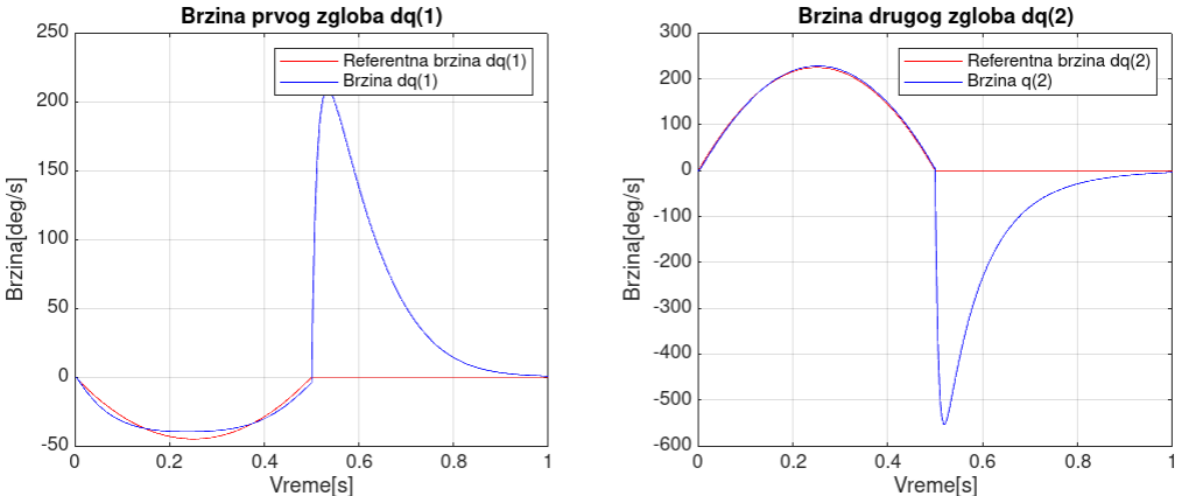


Pozicije zglobova $q(1)$ i $q(2)$

3.0.2 Brzine zglobova $\dot{q}$

```
figure('Position',[50 400 700 250]);
for m1 = 1:2
    subplot(1,2,m1)
    switch m1
        case 1
            plot(dt:dt:T,Ps.dq_ref(1,:),'r',dt:dt:T,Ps.dq(1,:),'b')
            grid
            title('Brzina prvog zgloba dq(1)')
            ylabel('Brzina[deg/s]')
            xlabel('Vreme[s]')
            legend('Referentna brzina dq(1)','Brzina
                dq(1)','Location','NorthEast')
        case 2
            plot(dt:dt:T,Ps.dq_ref(2,:),'r',dt:dt:T,Ps.dq(2,:),'b')
            grid
            title('Brzina drugog zgloba dq(2)')
            ylabel('Brzina[deg/s]')
            xlabel('Vreme[s]')
            legend('Referentna brzina dq(2)','Brzina q(2)','Location','NorthEast')
    end
end
```

MATLAB kod za prikaz brzina zglobova

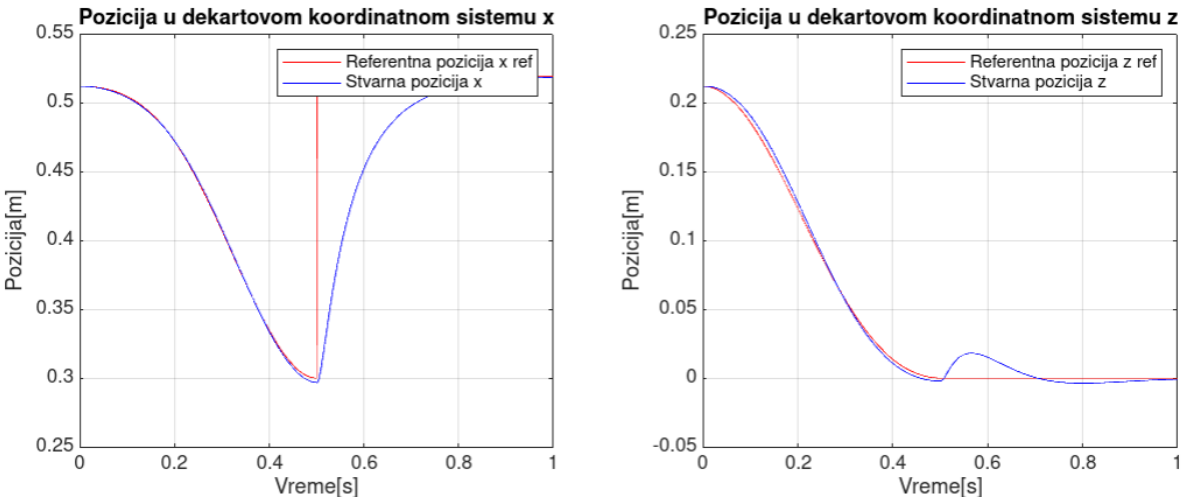


Brzine zglobova $q(1)$ i $q(2)$

3.0.3 Pozicija u dekartovom koordinatnom sistemu

```
%% external coordinates
figure('Position',[50 50 700 250]);
for m1 = 1:2
    subplot(1,2,m1)
    switch m1
        case 1
            plot(dt:dt:T,Ps.X_ref(1,:),'r',dt:dt:T,Ps.X(1,:),'b')
            grid
            title('Pozicija u dekartovom koordinatnom sistemu x')
            ylabel('Pozicija[m]')
            xlabel('Vreme[s]')
            legend('Referentna pozicija x ref','Stvarna pozicija
                x','Location','NorthEast')
        case 2
            plot(dt:dt:T,Ps.X_ref(2,:),'r',dt:dt:T,Ps.X(2,:),'b')
            grid
            title('Pozicija u dekartovom koordinatnom sistemu z')
            ylabel('Pozicija[m]')
            xlabel('Vreme[s]')
            legend('Referentna pozicija z ref','Stvarna pozicija
                z','Location','NorthEast')
    end
end
```

MATLAB kod za prikaz pozicija u dekartovom koordinatnom sistemu

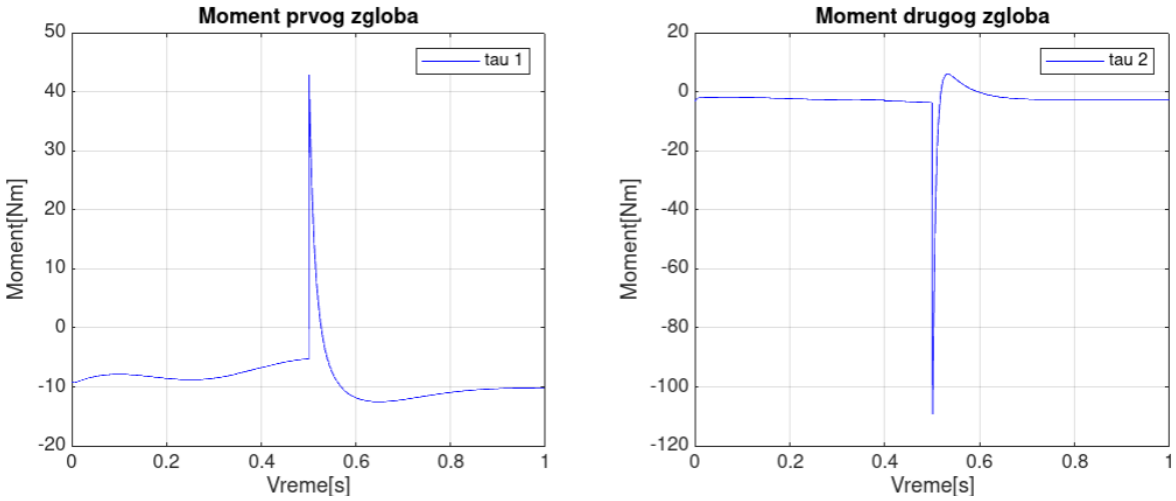


Pozicije x i z u dekartovom koordinatnom sistemu

3.0.4 Moment zglobova

```
%% joint torques
figure('Position',[800 400 700 250]);
for m1 = 1:2
     subplot(1,2,m1)
     switch m1
         case 1
             plot(dt:dt:T,Ps.Tau(1,:),'b')
             grid
             title('Moment prvog zgloba')
             ylabel('Moment[Nm]')
             xlabel('Vreme[s]')
             legend('tau 1')
         case 2
             plot(dt:dt:T,Ps.Tau(2,:),'b')
             grid
             title('Moment drugog zgloba')
             ylabel('Moment[Nm]')
             xlabel('Vreme[s]')
             legend('tau 2')
    end
end
```

MATLAB kod za prikaz momenta zglobova

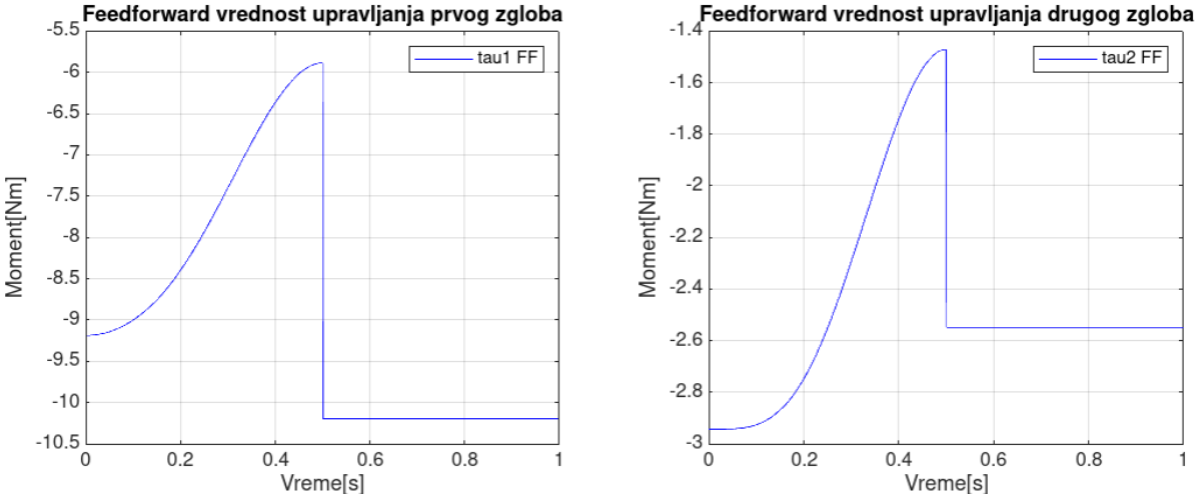


Momenti zglobova $q(1)$ i $q(2)$

3.0.5 Feedforward moment zglobova

```
%% joint torques
figure('Position',[800 400 700 250]);
for m1 = 1:2
     subplot(1,2,m1)
     switch m1
         case 1
             plot(dt:dt:T,Ps.Tau_FF(1,:),'b')
             grid
             title('Feedforward vrednost upravljanja prvog zgloba')
             ylabel('Moment[Nm]')
             xlabel('Vreme[s]')
             legend('tau1 FF')
         case 2
             plot(dt:dt:T,Ps.Tau_FF(2,:),'b')
             grid
             title('Feedforward vrednost upravljanja drugog zgloba')
             ylabel('Moment[Nm]')
             xlabel('Vreme[s]')
             legend('tau2 FF')
     end
end
```

MATLAB kod za prikaz feedforward moment zglobova

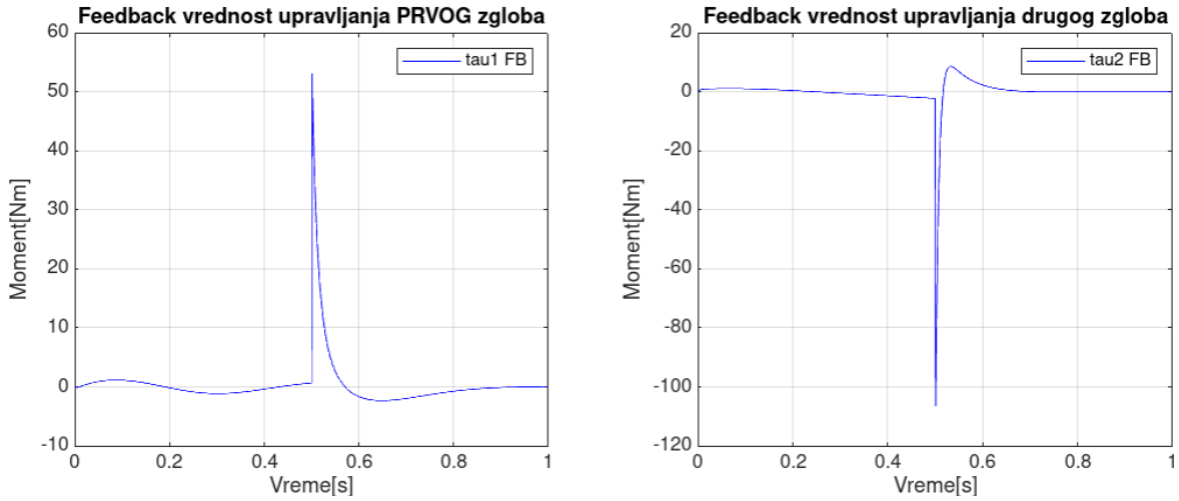


Feedforward momenti zglobova $q(1)$ i $q(2)$

3.0.6 Feedback moment zglobova

```
%% joint torques
figure('Position',[800 400 700 250]);
for m1 = 1:2
     subplot(1,2,m1)
     switch m1
         case 1
             plot(dt:dt:T,Ps.Tau_FB(1,:),'b')
             grid
             title('Feedback vrednost upravljanja prvog zgloba')
             ylabel('Moment[Nm]')
             xlabel('Vreme[s]')
             legend('tau1 FB')
         case 2
             plot(dt:dt:T,Ps.Tau_FB(2,:),'b')
             grid
             title('Feedback vrednost upravljanja drugog zgloba')
             ylabel('Moment[Nm]')
             xlabel('Vreme[s]')
             legend('tau2 FB')
    end
end
```

MATLAB kod za prikaz feedback moment zglobova



Feedback momenti zglobova $q(1)$ i $q(2)$