Upravljanje robotskom rukom

Stjepanović, Josip

Sveučilište J.J. Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Josip Stjepanović

Upravljanje robotskom rukom

Završni rad

Osijek, 2023

Sveučilište J.J. Strossmayera u Osijeku
Fakultet primijenjene matematike i informatike
Sveučilišni prijediplomski studij Matematika i računarstvo

Josip Stjepanović

Upravljanje robotskom rukom

Završni rad

Mentor: izv.prof.dr.sc. Zoran Tomljanović

Osijek, 2023

Sažetak

Ovaj rad analizira i testira na primjeru kontrole ruke robota teorijsku pozadinu, svojstva i ponašanje proporcionalno-integracijsko-derivacijskog regulatora (PID regulatora). Kako bi potvrdili teorijske zaključke implementiran je PID regulator za upravljanje kretanja ruke u programskom softveru Choregraphe gdje nam je omogućena kontrola nad kretnjama NAO robota. Nakon implementacije regulator je testiran na simuliranom i pravom robotu. Uz to tijekom pokreta ruke praćene su razne vrijednosti o kretanju ruke i elemenata regulatora te je sve grafički prikazano. Testovi pokreta na robotu potvrđuju efikasnost teorije PID regulatora, a njegova robustnost i jednostavnost implementacije pokazuju zašto ima tako široku uporabu.

Ključne riječi: PID regulator, humanoidni robot, NAO, Choregraphe

Control of robotic arm

Abstract

This paper analyzes and tests the theoretical background, properties and behavior of the Proportional-Integral-Derivative controller (PID controller) using the example of robot arm control. To confirm the theoretical conclusions, in Choregraphe software, a PID controller is implemented to control the motion of the robot arm. Choregraphe software allows us to control the movements of the NAO robot. After the implementation, the controller is tested on both simulated and real robot. Additionally, during arm movement, various motion-related values and controller elements are tracked and graphically represented. The motion tests on the robot confirm the efficiency of the PID controller theory. Its robustness and ease of implementation demonstrate why it has such broad applicability.

Keywords: PID controller, humanoid robot, NAO, Choregraphe

Sadržaj

1. Uvod

PID regulator je vrsta sustava koji se koristi kada zahtjevamo kontinuiranu kontrolu nad određenom varijablom kao što su temperatura sobe ili brzina automobila u slučaju upaljenoga tempomata. Cilj regulatora je održavanje određene ciljane vrijednosti. To se ostvaruje računanjem pogreške odnosno razlike između trenutne i ciljane vrijednosti varijable kojom sustav upravlja. Konačna vrijednost djelovanja na trenutno stanje izračunata je zbrajanjem proporcionalnog, integralnog i derivacijskog doprinosa pogreške odakle dolazi njegov naziv.

Radi punog razumijevanja PID regulatora prvo ćemo se dotaknuti njegove teorijske pozadine, kako je definiran te kako svaki od tri doprinosa utječe na konačno djelovanje regulatora. Kada opišemo PID regulator i njegovo ciljano ponašanje ukratko ćemo se dotaknuti SoftBank Robotics NAO humanoidnog robota.[9] Konkretno njegove ruke za čije kretanje implementiramo PID regulator kojim potkrepljujemo teorijska razmatranja. Zatim u idućim poglavljima dolazimo do implementacije PID regulatora za ruku robota i analize prikupljenih podataka tijekom pokreta ruke. Pomoću tih podataka ćemo analizirati uspješnost implementiranog regulatora te ilustrirati promjene ponašanja regulatora pri izmjeni početnih parametara.

2. PID regulator

U uvodu smo spomenuli kako kratica PID dolazi od tri različita izraza koji određuju promjene unutar sustava (proporcionalnog, integracijskog, derivacijskog). PID regulator i njegovo ponašanje ovise o kontrolnoj funkciji $u(t) : \mathbb{R} \to \mathbb{R}$ oblika

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \ , \ K_p, K_i, K_d > 0 \tag{2.1}$$

gdje je $e(t)$ trenutna pogreška specifičnije razlika između ciljane i trenutne vrijednosti kontrolirane varijable odnosno

$$e(t) = r(t) - y(t) \tag{2.2}$$

gdje smo s $r(t)$ označili ciljanu vrijednost, a s $y(t)$ trenutnu vrijednost varijable. K_p, K_i, K_d su realni nenegativni koeficijenti svakog od izraza. A funkcija prijenosa PID regulatora $u(s)$ [5] dana je izrazom

$$u(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \tag{2.3}$$

O samom pojmu funkcije prijenosa možemo pronaći u [3, str. 25-26].

Osim prikazanog oblika kontrolne funkcije koji je poznat kao paralelni oblik i uobičajeno je korišten u akademske svrhe poznat je još i takozvani serijski oblik koji je preferiran u inženjerstvu zbog lakšeg podešavanja regulatora. Kontrolna funkcija u serijskom obliku jest

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt}\right)$$

gdje imamo koeficijent K koji ima utjecaj na sva tri doprinosa, a vrijedi $T_i = \frac{1}{K_i}$, a $T_d = K_d$. Inženjeri često nemaju dovoljno vremena za izračun matematičkog modela pa je podešavanje koeficijenata lakše u serijskom obliku s obzirom da je moguće utjecati na cijeli regulator samo promijenom koeficijenta K. Ako poznajemo matematički model sustava oblik kontrolne funkcije nam ne pravi razliku. Više o serijskom obliku, njegovim koeficijentima i doprinosima možemo pronaći u [2].

Regulator pokušava smanjiti trenutnu vrijednost pogreške dodavanjem vrijednosti funkcije (2.1) na $y(t)$ trenutnu vrijednost kontrolirane varijable.

U nastavku ovog poglavlja dotaknut ćemo se svakog dijela PID regulatora to jest funkcije (2.1) i kako svaki dio doprinosi kretanju vrijednosti pogreške (2.2) te koje su posljedice promjene njihovih koeficijenata.

2.1. Proporcionalni doprinos

Proporcionalni doprinos odnosno izraz

$$P = K_p e(t) \tag{2.4}$$

je fundamentalan dio regulatora, određen je umnoškom pogreške (2.2) i proporcionalnog koeficijenta K_p. Njegov doprinos dovodi do trenutne reakcije na pogrešku. Izraz nam donosi linearnu vezu između pogreške i kontrole. Jasno je kako je pogreška veća tako je i vrijednost doprinosa veća i regulator ima jaču reakciju. Kako imamo proporcionalnu linearnu vezu korist ovoga izraza jest što će jako brzo reagirati na pogrešku i ubrzati dolazak do ciljane vrijednosti. No kada bismo u funkciji (2.1) imali samo proporcionalni doprinos došlo bi do toga da nakon stabilizacije postoji razlika između ciljane vrijednosti i vrijednosti na kojoj se regulator stabilizirao, detaljnije u [8, str 5]. Tu razliku nazivamo steady-state pogreška. Jedan od nedostataka proporcionalnog doprinosa je činjenica da sam po sebi ne može upotpunosti eliminirati steady-state pogrešku iako ju smanjuje. To je razlog dodavanja ostalih doprinosa ponajviše integracijskog. Proporcionalni doprinos mora biti dobro prilagođen da bi se ponašao u skladu s očekivanjem.

Njegovo ponašanje kontroliramo podešavanjem proporcionalnog koeficijenta K_p. Promjena koeficijenta ima veliki utjecaj na ponašanje regulatora. Povećanje vrijednosti koeficijenta dovodi do veće osjetljivosti regulatora i ubrzava proces stabiliziranja te može smanjiti steady-state pogrešku. Ali prevelika vrijednost može uzročiti preveliko premašivanje ciljane vrijednosti zbog nemogućnosti regulatora da se zaustavi na vrijeme s obzirom na svoju agresivnu reakciju na pogrešku. Isto tako može dovesti do nestabilnosti konstantnim premašivanjem. Dok premala vrijednost koeficijenta može značajno povećati steady-state pogrešku i usporiti korekcije regulatora te tako povećati vrijeme konvergencije ka ciljanoj vrijednosti.

2.2. Integracijski doprinos

Izraz kojim definiramo integracijski doprinos je

$$I = K_i \int_0^t e(\tau) d\tau \tag{2.5}$$

Najvažnija uloga ovoga izraza kontrolne funkcije jest eliminacija steady-state pogreške. Doprinos kroz vrijeme akumulira i popravlja pogrešku integrirajući funkciju pogreške (2.2) i tako eliminira steady-state pogrešku. Njega kontroliramo prilagodbom integracijskog koeficijenta K_i. Ako je vrijednost integracijskog koeficijenta loše odabrana može dovesti do premašivanja ciljane vrijednosti i oscilacija.

Kod integracijskog koeficijenta njegovim povećanjem regulator agresivnije akumulira i reagira na pogrešku te brže smanjuje steady-state error što osigurava brže dostizanje i održavanje ciljane vrijednosti. Veći integracijski koeficijent može usporiti prvotnu reakciju na promjene u pogrešci s obzirom da je fokusiran na smanjenje ukupne pogreške. Isto tako prevelika

vrijednost koeficijenta lako može uzročiti nestabilnosti na isti način kao i prevelika vrijednost proporcionalnog koeficijenta gdje dolazi do premašivanja i oscilacija. Premala vrijednost dovodi do povećanja steady-state pogreške i dok će vjerojatno osigurati stabilnost sustava neće doći do eliminacije steady-state pogreške, što je važan dio regulatora.

2.3. Derivacijski doprinos

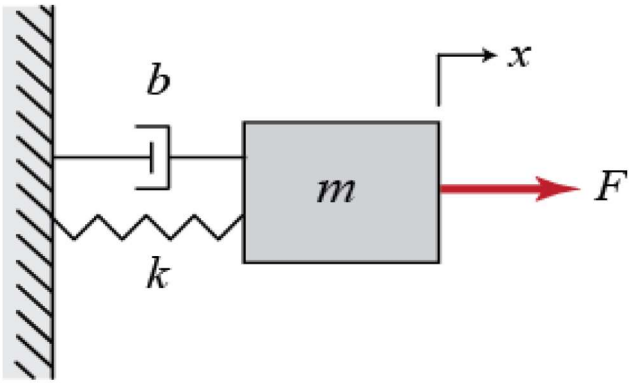
Derivacijski doprinos je definiran izrazom

$$D = K_d \frac{de(t)}{dt} \tag{2.6}$$

Svrha derivacijskog doprinosa je predviđanje ponašanja funkcije pogreške. Računa se kao derivacija funkcije pogreške s obzirom na vrijeme. Doprinos ovisi o brzini kretanja funkcije (2.2) u danom trenutku t. Brzinu računamo izračunom njene derivacije te ju množimo derivacijskim koeficijentom kako bi dobili konačnu vrijednost doprinosa. Možemo reći da kompenzira za pogrešku unaprijed i tako smanjuje oscilacije i premašivanje. Utjecaj doprinosa na regulator kontroliramo prilagođavanjem derivacijskog koeficijenta K_d.

Kada povećavamo derivacijski koeficijent regulator postaje osjetljiviji na značajne promjene u pogrešci i tako uzrokuje smanjenje premašivanja ciljane varijable i smiruje oscilacije. Preveliko povećanje koeficijenta će rezultirati pretjeranim reagiranjem na promjene u pogrešci te čak može dovesti i do nestabilnosti. Ali ako je vrijednost koeficijenta premala regulator će slabo reagirati na smetnje poput bijega od ciljane vrijednosti ili promjene same ciljane vrijednosti.

Ulogu svakog od PID doprinosa i način rada regulatora ćemo u nastavku ilustrirati na primjeru jednostavnog sustava mase s oprugom i prigušivač koji je prikazan na slici (Slika 1).



Slika 1: Sustav mase s oprugom i prigušivačem[11]

Objekt mase m povlačimo od opruge koeficijenta k uz prigušivač viskoznosti b silom F, x nam označava pomak objekta od početne pozicije. Cilj nam je postići optimalnu kontrolu sustava kojeg kontroliramo određivanjem vrijednosti sile PID regulatorom kako bismo stabilizirali sustav oko ciljane vrijednosti pomaka. Jednadžba ovoga sustava jest

$$F = mx'' + bx' + kx \tag{2.7}$$

iz jednadžbe (2.7) primjenom Laplaceove transformacije slijedi

$$F(s) = ms^2X(s) + bsX(s) + kX(s)$$

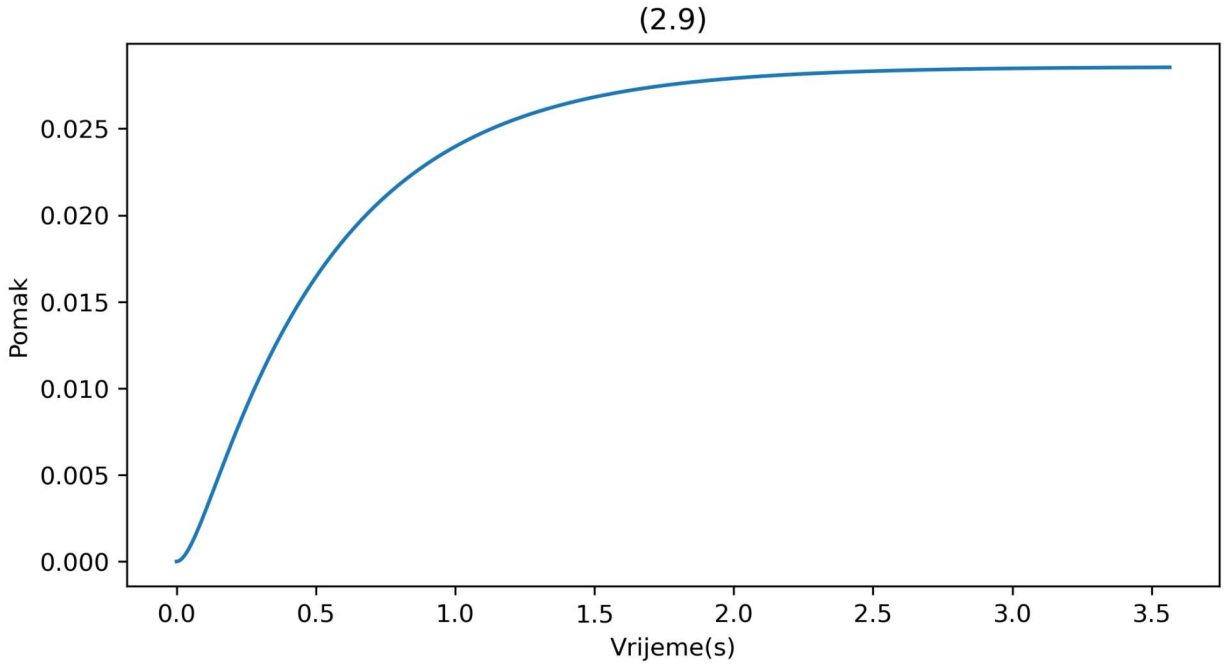
odakle možemo vidjeti kako je funkcija prijenosa sustava

$$h(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k} \ . \tag{2.8}$$

Uzeti ćemo da je $m = 1$ kg, $b = 20$ N s/m, $k = 35$ N/m $F = 1$ N. Tada substitucijom tih vrijednosti u (2.8) dobivamo

$$h(s) = \frac{1}{s^2 + 20s + 35} \ . \tag{2.9}$$

Sustav možemo simulirati korištenjem Python biblioteke Control [12]. Funkcijama dostupnim unutar biblioteke simulirati ćemo sustav kroz vrijeme pomoću njegove funkcije prijenosa. Prvo ćemo simulirati ponašanje sustava bez utjecaja regulatora (Slika 2). Primjećujemo kako se vrijednost varijable pomaka x stabilizira oko 0.05, a mi ćemo za ciljanu vrijednost uzeti 1. Dakle steady-state pogreška je jako velika. Također vrijeme stabilizacije iznosi nešto više od 1.5 sekunde. To ćemo pokušati poboljšati dodavanjem PID regulatora.



Slika 2: Sustav bez regulatora

Funkciju prijenosa nakon dodavanja regulatora možemo izračunati pomoću funkcije prijenosa sustava $h(s)$ (2.9) i funkcije prijenosa regulatora $u(s)$ (2.3). Naš izlaz je jednak umnošku tih dvaju funkcija prijenosa i funkcije pogreške gdje je r ciljana vrijednost odnosno

$$X(s) = u(s)h(s)(r(s) - X(s)) \ .$$

Iz te jednadžbe lako vidimo da

$$X(s)(1 + u(s)h(s)) = u(s)h(s)r(s)$$

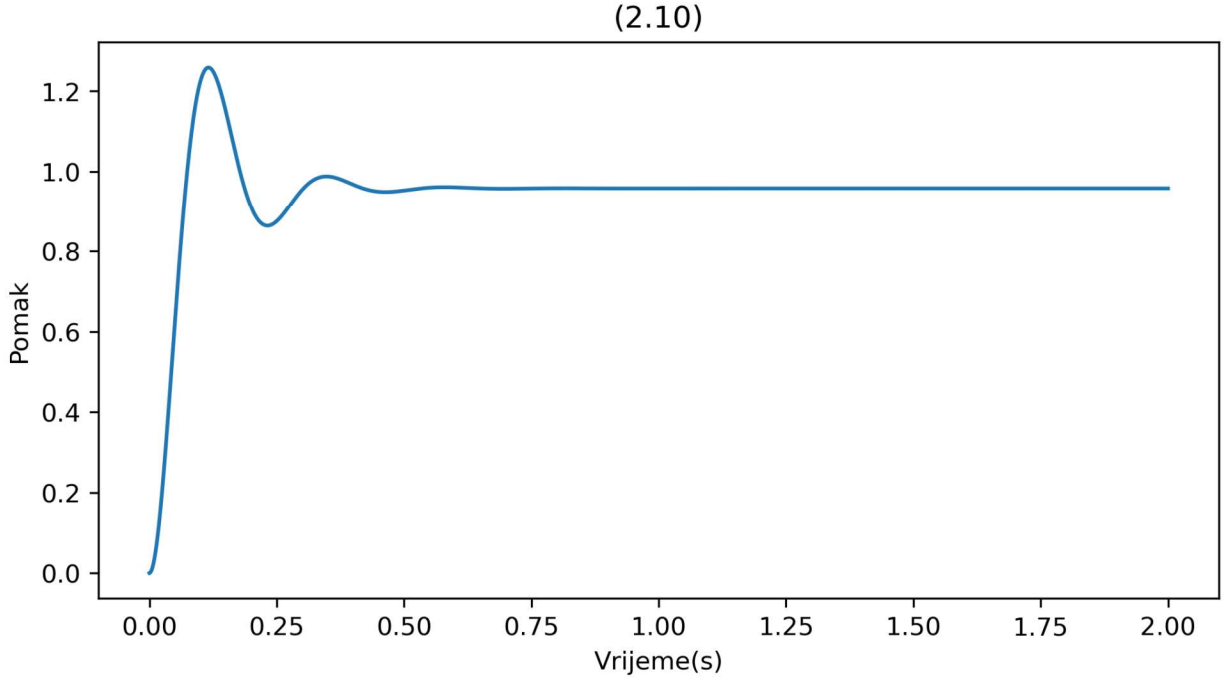
nadalje jasno je da je funkcija prijenosa nakon dodavanja kontrole PID regulatora $g(s)$ jednaka

$$g(s) = \frac{X(s)}{r(s)} = \frac{u(s)h(s)}{1 + u(s)h(s)} \ . \tag{2.10}$$

Prvo ćemo pogledati što se događa uvedemo li samo proporcionalni doprinos. Ako radimo samo s proporcionalnim doprinosom funkcija prijenosa regulatora (2.3) iznosi $u(s) = K_p$. Ako tu funkciju i funkciju (2.9) substituiramo u (2.10) nakon sređivanja dobivamo funkciju prijenosa

$$g(s) = \frac{K_p}{s^2 + 20s + (35 + K_p)} \ . \tag{2.11}$$

Simulacijom sustava gdje je $K_p = 800$ (Slika 3) možemo primjetiti poboljšanje preciznosti sustava, brži uspon i bržu stabilizaciju, ali isto tako primjećujemo povećanje premašivanja i pojavu oscilacija. Utjecaj regulatora možemo poboljšati dodavanjem ostalih doprinosa.



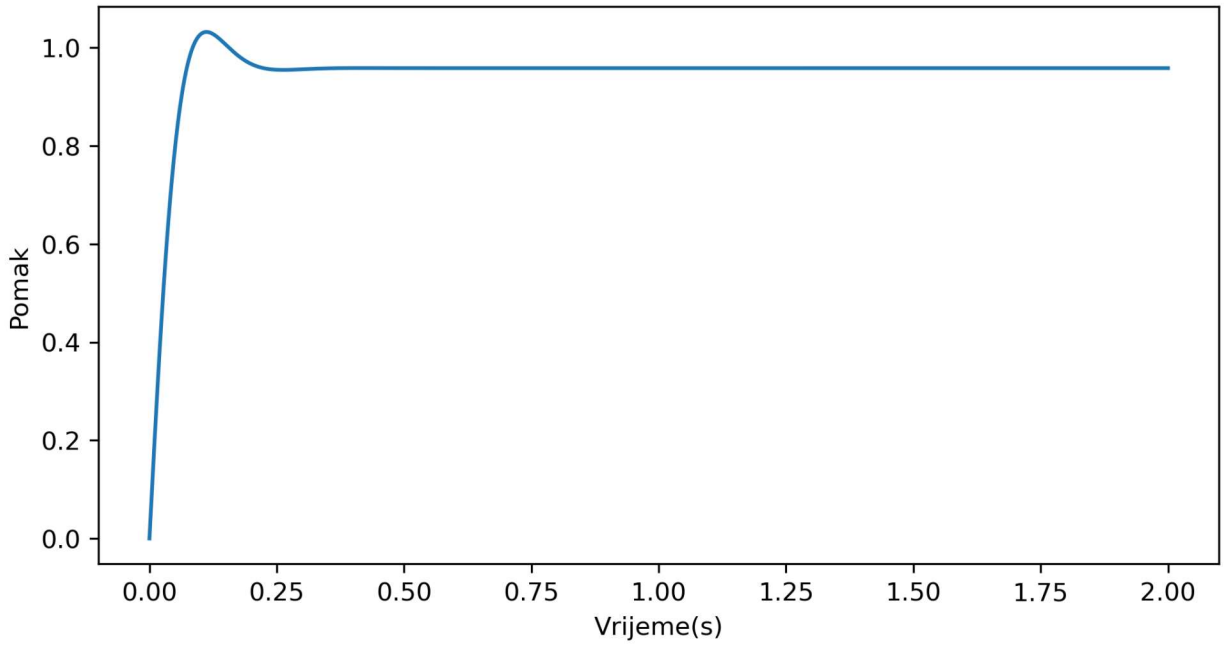
Slika 3: Sustav s P doprinosom

Dodavanjem derivacijskog doprinosa od funkcije prijenosa (2.9) i funkcije regulatora koja u ovom slučaju iznosi $u(s) = K_p + K_d s$ substitucijom u (2.10) dolazimo do funkcije prijenosa

$$g(s) = \frac{K_d s + K_p}{s^2 + (20 + K_d)s + (35 + K_p)} \ . \tag{2.12}$$

Postavimo li derivacijski koeficijent $K_d = 20$ vidjet ćemo kako je dodavanjem derivacijskog doprinosa umanjeno premašivanje i vrijeme potrebno za stabilizaciju (Slika 4). Međutim derivacijski doprinos nije imao nikakav utjecaj na steady-state pogrešku koja je upotpunosti ista. Odnosno sustav se stabilizira na istoj vrijednosti kao i u prošlom slučaju.

(2.11)

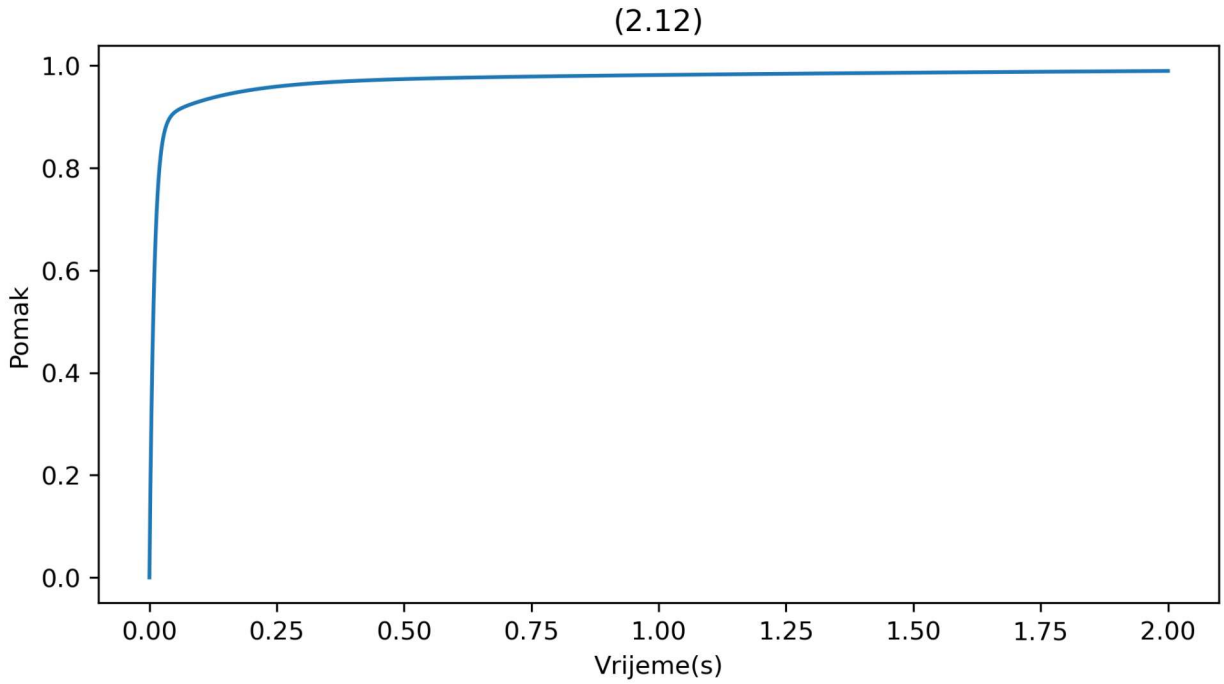


Slika 4: Sustav s PD doprinosima

Kako bismo upotpuno eliminirali steady-state pogrešku potreban nam je dodatak integracijskog doprinosa. Dodavanjem integracijskog doprinosa dolazimo do konačne funkcije prijenosa tako što substituiramo funkcije (2.3) i (2.9) u formulu (2.10) i tada je funkcija prijenosa $g(s)$ jednaka

$$g(s) = \frac{K_d s^2 + K_p s + K_i}{s^3 + (20 + K_d)s^2 + (35 + K_p)s + K_i} \ . \tag{2.13}$$

Konačno simularimo li sustav gdje su nam koeficijenti jednaki $K_p = 850, K_i = 450, K_d = 100$ (Slika 5) dobivamo sustav bez premašivanja, brzim vremenom stabilizacije i nepostojećom steady-state pogreškom.



Slika 5: Sustav s PID doprinosima

Primjerom smo ilustrirali zašto nam je potreban svaki od doprinosa i kakav je utjecaj njegova dodavanja na sustav, a u sljedećem poglavlju ćemo opisati robota na kojem ćemo implementirati PID regulator i alate koji će nam poslužiti u tu svrhu.

3. NAO robot

NAO robot je humanoidni robot kojeg je razvila tvrtka SoftBank Robotics u razne edukacijske i istraživačke svrhe. Robot je dizajniran kako bi imitirao ljudske pokrete i ponašanje. Robot posjeduje 25 stupnjeva slobode među kojima su i 5 njih u ruci, po dva za rame i lakat i jedan za zapešće. Detaljno o specifikacijama robota i kinematici svakog od njegovih dijelova možemo pronaći u [4]. Također uz robota postoji i softver Choregraphe koji omogućuje programiranje kretnji i ponašanja robota u jezicima C++ i Python.

Korištenjem programskog jezika Python implementirat ćemo PID regulator koji će kontrolirati kretanje ruke robota. Uz Choregraphe dolaze i Python moduli u kojima postoje razne funkcije za kontrolu kretanja svakoga stupnja slobode. To će nam omogućiti implementaciju PID regulatora nad pojedinim aktuatorima robota.

Robot pri pokretanju ustaje i dolazi u svoju početnu poziciju (Slika 6).

Slika 6: NAO robot

4. Implementacija

Implementacija PID regulatora za ruku NAO robota odrađena je u programskom jeziku Python. Ideja je simulirati kontinuiranost pomoću petlje. Petlja koja je postavljena izvršava određeni broj ponavljanja, u našem slučaju 500 i čije je trajanje izvršavanja jednog ponavljanja unaprijed zadano vrijeme, u našem slučaju 0.1s. Trajanje izvršavanja petlje kontroliramo pomoću funkcije time.sleep koja zaustavlja program na određeno vrijeme, vrijeme izvršavanja ostalog koda unutar petlje je zanemarivo. Varijabla koju kontroliramo je kut aktuatora ramena robota. Unutar petlje izračunavamo vrijednost kontrolne funkcije (2.1), dobivenu vrijednost množimo s vremenom izvršavanja petlje koje iznosi 0.1, tako dobivenu vrijednost zatim dodajemo na trenutnu vrijednost kontrolirane varijable. Zbrojem trenutne vrijednosti kuta i vrijednosti kontrolne funkcije dobivamo novu vrijednost kuta. Nakon izračuna pozivamo funkciju SetAngles [10] koja za dane parametre koji su novi kut i postotak maksimalne brzine aktuatora mijenja kut aktuatora. Prije izvršavanja skripte potrebno je zadati ciljanu vrijednost, a trenutnoj vrijednosti kuta možemo pristupiti funkcijom GetAngles [10].

4.1. Proporcionalni doprinos

Implementacija proporcionalnog doprinosa je najjednostavniji dio implementacije PID regulatora. Prije samoga izvršavanja petlje definiramo vrijednost proporcionalnog koeficijenta K_p i vrijednost početne pogreške koju nam je lagano izračunati oduzimanjem trenutne vrijednosti kuta od ciljane vrijednosti. Zatim unutar petlje u svakom ponavljanju izračunavamo novu pogrešku kao razliku ciljanog i trenutnog kuta, a vrijednost proporcionalnog dijela kontrolne funkcije nam je jednaka proporcionalnom koeficijentu i pogrešci odnosno funkciji (2.4).

4.2. Integracijski doprinos

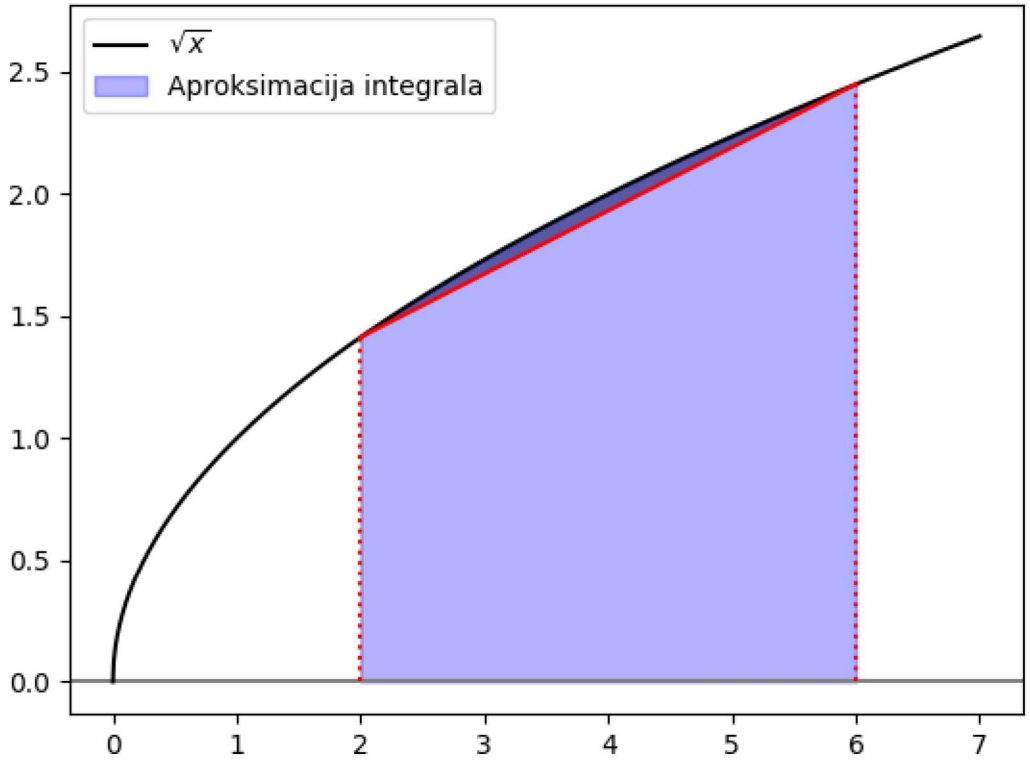
Izračun vrijednosti integracijskog izraza kontrolne funkcije zahtjeva od nas aproksimaciju integrala (2.5). Kako bi aproksimirali vrijednost integrala koristiti ćemo trapezno pravilo zbog jednostavnosti implementacije. Još neki od mogućih načina aproksimacije su Newton-Cotesova formula kod koje se interval dijeli n jednakih podintervala, a podintegralnu funkciju aproksimiramo interpolacijskim polinomom n-tog reda u Lagrangeovom obliku. Specijalno za $n = 2$ iz Newton-Cotesove formule dobivamo Simpsonovo pravilo koje bi mogli koristiti za aproksimaciju integrala. [7, str. 196-202]

Trapezno pravilo aproksimira površinu ispod grafa funkcije kao trapez i zatim računa njegovu površinu (Slika 7). Pravilo tvrdi da za funkciju $f : [a, b] \to \mathbb{R}$ vrijedi

$$\int_a^b f(x)dx \approx \frac{(b-a)}{2}(f(a) + f(b)) \ . \tag{4.1}$$

Razliku između vrijednosti integrala i aproksimacije integrala nazivamo pogreškom aproksimacije integrala. Što je pogreška manja kažemo da je naša aproksimacija preciznija. U [7, str. 192-193] možemo pronaći kako postoji $c \in \langle a, b\rangle$ takav da pogreška iznosi

$$E = -\frac{(b-a)^3}{12} f''(c) \ .$$



Slika 7: Trapezno pravilo

Možemo zaključiti da ako je $[a, b]$ velik i pogreška će biti velika. Integral možemo bolje izračunati ako podijelimo interval $[a, b]$ na podintervale i zatim izračunamo trapezno pravilo (4.1) za svaki podinterval posebno te rezultate zbrojimo. Taj pristup nazivamo generalizirano trapezno pravilo (Slika 8). Na grafu možemo primjetiti znatno smanjenje pogreške unatoč tome što je interval podijeljen samo na četiri dijela.

Kod podijele neka je x_k takav da $a = x_0 < x_1 < ... < x_{n-1} < x_n = b$ i neka je $\Delta x_k = x_k - x_{k-1}$ duljina k-tog podintervala, tada vrijedi

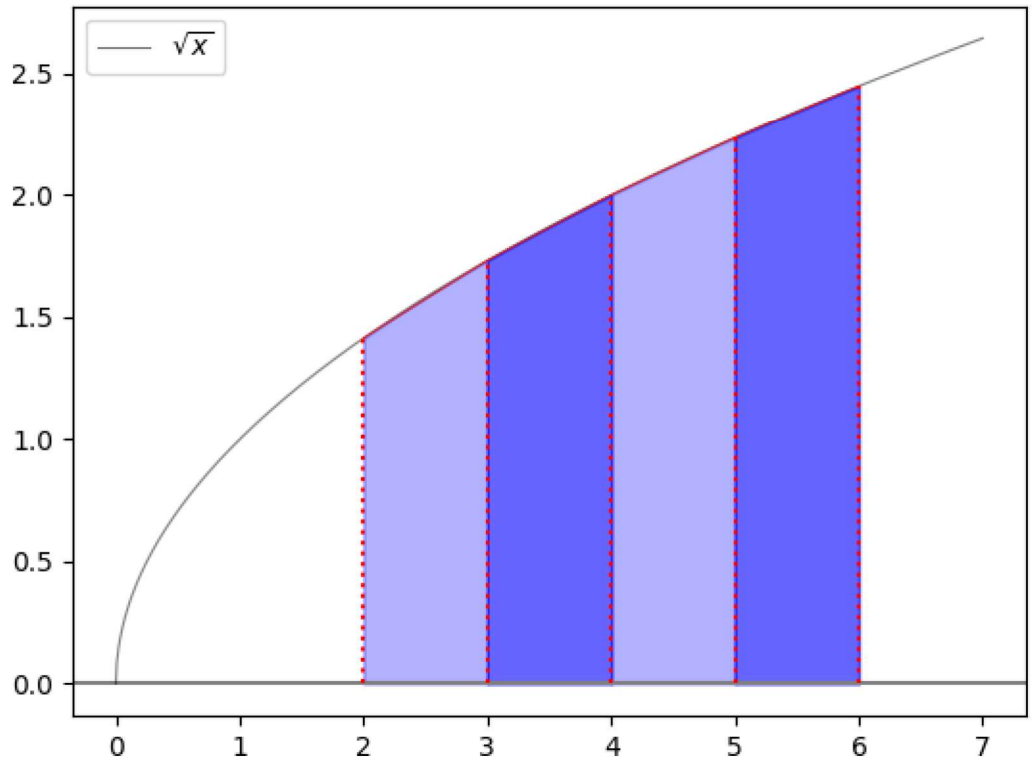
$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k \ . \tag{4.2}$$

Kada primjenjujemo generalizirano trapezno pravilo (4.2) u našem slučaju na integral $\int_0^t e(\tau)d\tau$ x-os nam je vrijeme od početka petlje, a y-os je vrijednost funkcije $e(t)$. Možemo na svako izvršavanje petlje gledati kao na jedan podinterval intervala $[0, t]$ i kako znamo da svako izvršavanje petlje traje 0.1s vrijedi da je $\Delta x_k = 0.1$, a kako znamo da će biti 500 ponavljanja petlje $n = 500$. Tako da kod nas generalizirano trapezno pravilo jest

$$\int_0^t e(\tau)d\tau \approx \sum_{k=1}^{500} \frac{e(x_{k-1}) + e(x_k)}{2} 0.1 \tag{4.3}$$

gdje je $e(x_{k-1})$ vrijednost pogreške u predhodnom izvršavanju petlje, a $e(x_k)$ pogreška u trenutnom izvršavanju petlje. Kada govorimo o implementaciji ovo je oblik koji ćemo koristiti i koji nam najbolje odgovara s obzirom na to da aproksimiramo integral u trenutku kroz vrijeme, odnosno kako se petlja izvršava. Računamo vrijednost funkcije $e(t)$ u točki t svaki prolazak kroz petlju, a kako bi izračunali integral u trenutnoj točki na vrijednost integrala u prošloj točki dodamo izraz

$$\frac{e(x_{k-1}) + e(x_k)}{2} 0.1 \ .$$



Slika 8: Generalizirano trapezno pravilo

Isto tako pošto je trajanje svakog izvršavanja petlje jednako naša podjela na podintervale je ekvidistantna pa prema [7, str. 194] jednostavno možemo izračunati pogrešku aproksimacije integrala generaliziranog trapeznog pravila kao

$$E = -\frac{b-a}{12} \Delta x_k f''(c) \ .$$

Podijelimo li $[a, b]$ na više dijelova smanjujemo Δx_k i tako smanjujemo pogrešku. Ukoliko bismo imali problema s prevelikom pogreškom aproksimacije možemo smanjiti trajanje jednog izvršavanja petlje.

Dakle prije prvoga izvršavanja petlje postavimo vrijednost varijable integral na 0 i definiramo vrijednost integracijskog koeficijenta K_i. Zatim tijekom svakog izvršavanja petlje varijabli dodajemo vrijednost dobivenu računanjem izraza pod sumom u tvrdnji (4.3), a varijablu integral dodajemo vrijednosti kontrolne funkcije.

4.3. Derivacijski doprinos

Za računanje ukupne vrijednosti kontrolne funkcije još nam je potrebna vrijednost derivacije funkcije $e(t)$ to jest izraz (2.6). Kako bismo izračunali vrijednost derivacije moramo pronaći način na koji bismo aproksimirali vrijednost derivacije koji je primjenjiv u našem slučaju kao što je trapezno pravilo kod integrala. S obzirom da računamo derivaciju kako vrijeme prolazi to jest tijekom izvršavanja pokreta potrebna je metoda kojom možemo računati trenutnu derivaciju u svakom prolasku kroz petlju. Metoda koja nam je prikladna za aproksimaciju derivacije u svakom prolasku petlje jest metoda konačnih diferencija o kojoj možemo pronaći u [6, str. 244-247].

Ako po definiciji pogledamo derivaciju funkcije $f(x)$ u točki x ona jest

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

ako limes postoji. Pa prema Taylorovom teoremu ako je funkcija dva puta neprekidno derivabilna onda slijedi

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(c)$$

gdje je $x < c < x + h$. Iz te jednadžbe jednostavno slijedi

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h^2}{2} f''(c) \ . \tag{4.4}$$

Jednažba (4.4) implicira kako za jako mali h vrijednost razlomka će jako dobro aproksimirati vrijednost derivacije te uzimamo

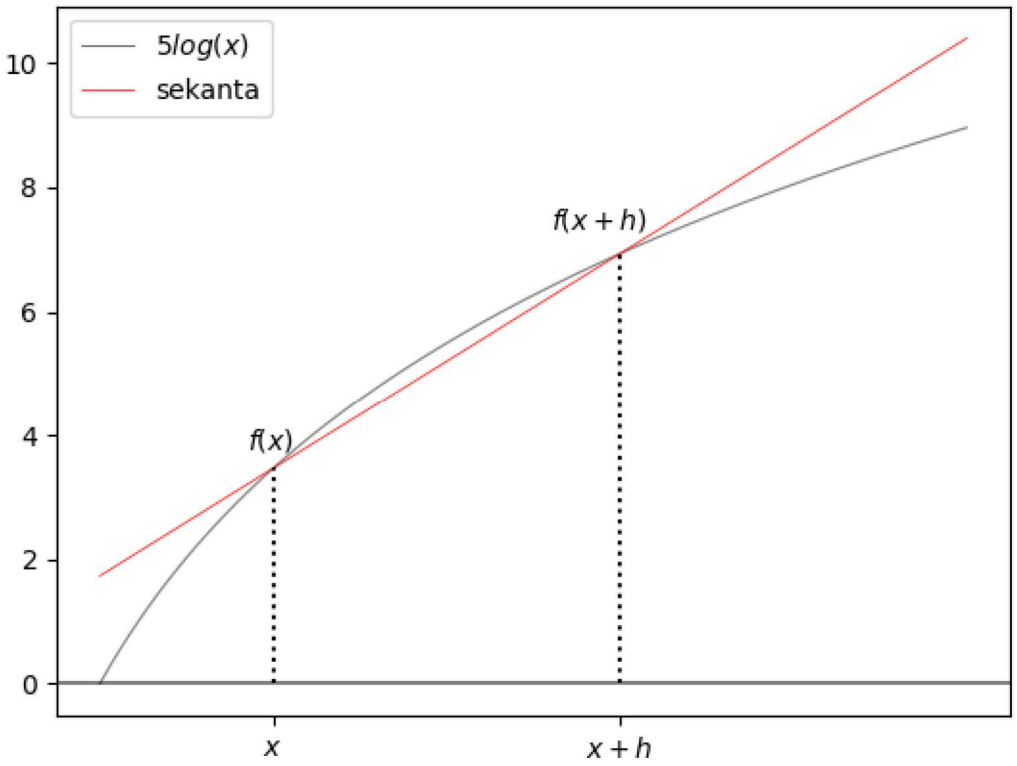
$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

kao aproksimaciju prve derivacije funkcije f u točki x, a iz (4.4) izraz

$$E = \frac{h^2}{2} f''(c)$$

možemo uzeti kao pogrešku aproksimacije. Kako je izraz pogreške proporcionalan vrijednosti h možemo utjecati na preciznost aproksimacije promjenom h. Ukoliko želimo smanjiti pogrešku smanjili bismo h.

Metodu također možemo protumačiti kao računanje vrijednosti koeficijenta rasta sekante između dviju točaka kako bi aproksimirali derivaciju (Slika 9). Isto tako možemo primjetiti kako smanjenjem razlike između točaka h smanjujemo pogrešku aproksimacije.



Slika 9: Metoda konačnih diferencija

Kada govorimo o implementaciji u našem slučaju možemo primjetiti kako je funkcija pogreške (2.2) funkcija čiji derivaciju aproksimiramo te da je proteklo vrijeme t varijabla na x-osi. Razlika između točaka je jednaka vremenu potrebnom za jedno izvršavanje petlje odnosno $h = 0.1$. Derivaciju u točki t tijekom izvršavanja petlje možemo aproksimirati kao

$$\frac{e(t + 0.1) - e(t)}{0.1} \tag{4.5}$$

gdje će nam $e(t)$ biti vrijednost pogreške iz prethodnog izvršavanja petlje, a $e(t + 0.1)$ vrijednost pogreške u trenutnom izvršavanju petlje.

Kako bismo dobili konačnu vrijednost derivacijskog doprinosa moramo aproksimaciju derivacije pomnožiti s derivacijskim koeficijentom K_d čiju vrijednost ćemo postaviti prije samoga

izvršavanja petlje. Tijekom svakoga izvršavanja petlje aproksimiramo vrijednost derivacije u trenutnoj točki t koristeći metodu konačnih diferencija prema izrazu (4.5) koji smo izveli te nakon što dobivenu vrijednost pomnožimo s derivacijskim koeficijentom kako bismo izračunali (2.6) imamo sve što nam je potrebno za izračun vrijednosti kontrolne funkcije $u(t)$.

Kako smo izračunali vrijednost svakog od doprinosa u točki t u mogućnosti smo izračunati vrijednost kontrolne funkcije $u(t)$ u točki t. Nakon što pomnožimo dobivenu vrijednost s trajanjem izvršavanja petlje dobivamo vrijednost kontrole. Tu vrijednost dodajemo na trenutnu vrijednost kuta kako bismo dobili novu vrijednost kuta na koju robot dolazi. Sve ovo se događa tijekom svakog izvršavanja petlje.

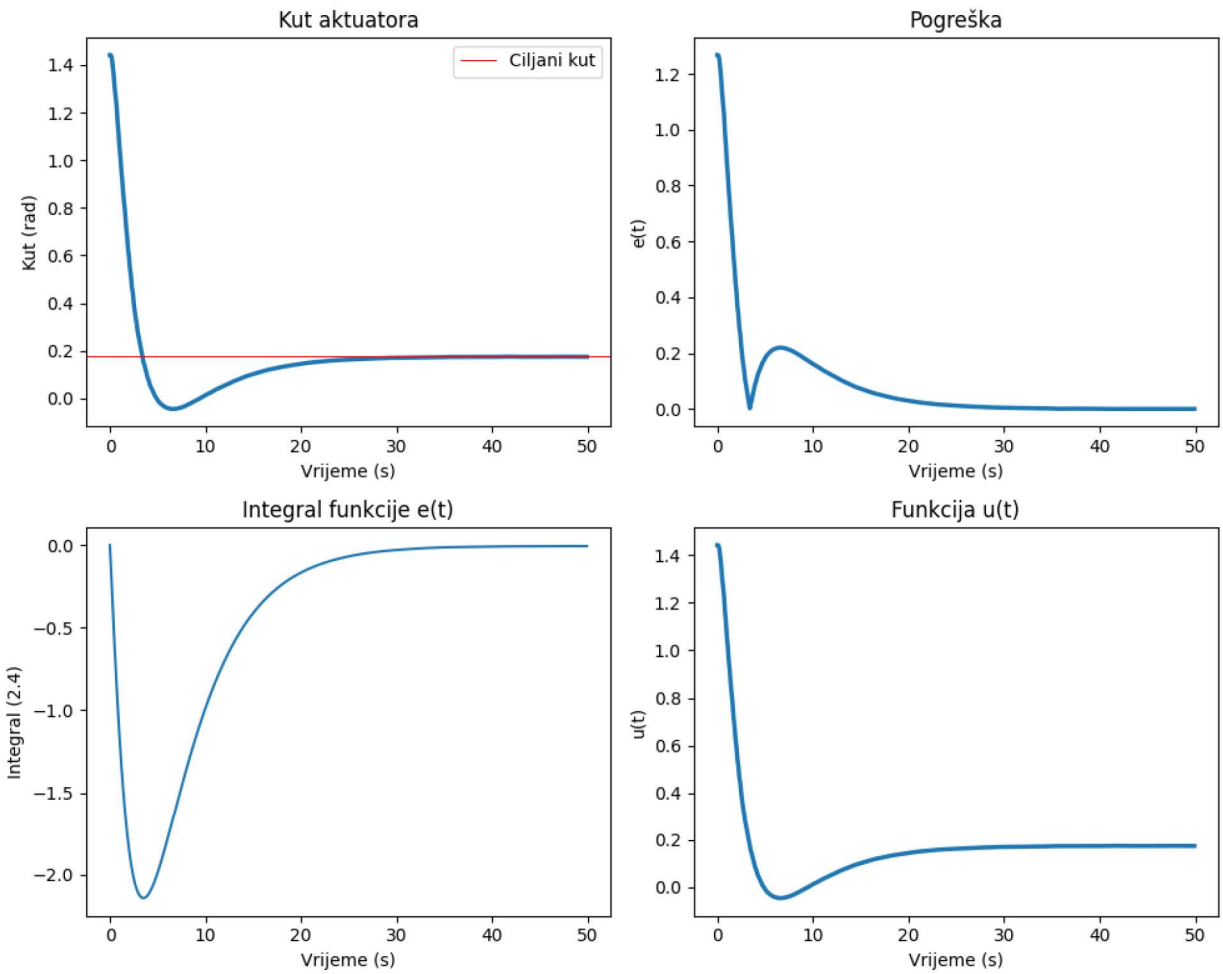
U primjeru izvršavanja PID regulatora na pokretu ruke robota pokrenuli smo robota u početnoj poziciji (Slika 6). Moguće je regulator pokrenuti iz bilo koje pozicije koju robot može dostići. Vrijednost ciljane varijable postavili smo tako da ruka dođe u poziciju paralelnu s tlom. Regulator je tijekom izvršavanja stabilizirao ruku robota i postavio ju na ciljanu vrijednost kuta (Slika 10). U sljedećem poglavlju ćemo proučiti rezultat implementacije PID regulatora na pokretu ruke robota i ilustrirati njegovo ponašanje s obzirom na promjene u koeficijentima.

Slika 10: Robot nakon izvršavanja petlje

5. Analiza podataka

Tijekom izvođenja petlje unutar implementacije u svakom prolasku kroz petlju prikupljani i zapisani u zasebnu csv datoteku su podaci o radu PID regulatora. Podaci su vrijednost kontrolirane varijable specifičnije kuta, vrijednost funkcije pogreške, aproksimacija vrijednosti integrala i vrijednost kontrolne funkcije. Podaci su grafički prikazani pomoću Python-a tako da svaka od zapisanih informacija ima prateći graf. Na slikama imamo četiri grafa gdje prvi graf je graf kuta aktuatora koji prati kretanje vertikalnog kuta ramena u radianima, a crvena linija na grafu označava ciljanu vrijednost kuta u radianima. Također imamo graf pogreške koji prati vrijednost funkcije $e(t)$ kroz vrijeme (2.2). Treći graf prikazuje vrijednost integrala (2.4). Na kraju dolazi graf kontrolne funkcije $u(t)$ koji nam prikazuje kako se ponaša vrijednost kontrolne funkcije kroz vrijeme. S tim nam je omogućena usporedba ponašanja regulatora s očekivanjem i pregled promjena u ponašanju kao posljedica različitih PID koeficijenata.

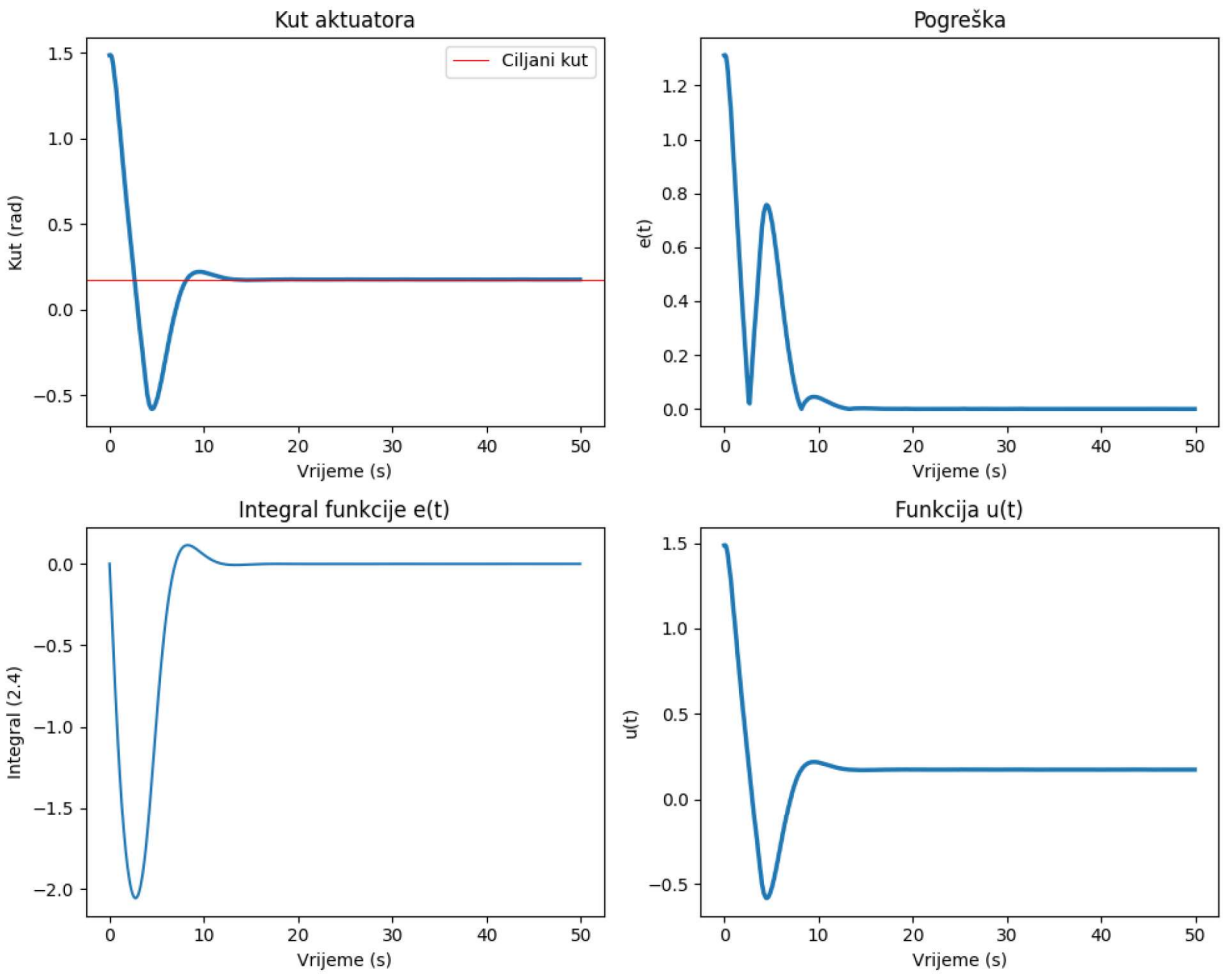
Kroz testiranje otkriveno je kako vrijednosti koeficijenata postavljene na $K_p = 1.5, K_i = 0.2, K_d = 0.3$ ukazuju na vrlo dobre rezultate. Prvo ćemo se dotaknuti podataka dobivenih ovim koeficijentima (Slika 11) zatim ćemo ih usporediti s promjenjenim koeficijentima i vidjeti imaju li promjena na koeficijente očekivani utjecaj iskazan u teorijskim razmatranjima. Postoje razne tehnike određivanja idealnih koeficijenata. Kao što je naprimjer Ziegler-Nichols metoda kod koje postoje jednostavne formule određivanja koeficijenta te Chien-Hrones-Reswick metoda koja je modifikacija Ziegler-Nichols metode. [1, str. 134-150]. U ovom slučaju korištena je najjednostavnija metoda testiranja raznih vrijednosti i usporedbe rezultata.



Slika 11: Koeficijenti $K_p = 1.5, K_i = 0.2, K_d = 0.3$

Možemo primjetiti kako se vrijednost kuta stabilizira dosta brzo nakon samo 20 sekundi. Kretanje kuta je očekivano, imamo zadovoljavajuću inicijalnu brzinu i malo premašivanje ciljane vrijednosti. Ako pogledamo graf koji prati pogrešku kroz vrijeme nakon stabilizacije regulatora pogreška je gotovo jednaka nuli što znači da smo praktički eliminirali steady-state pogrešku. Također iz grafova se može vidjeti kako nema oscilacija te da su koeficijenti odgovarajući za ovaj regulator.

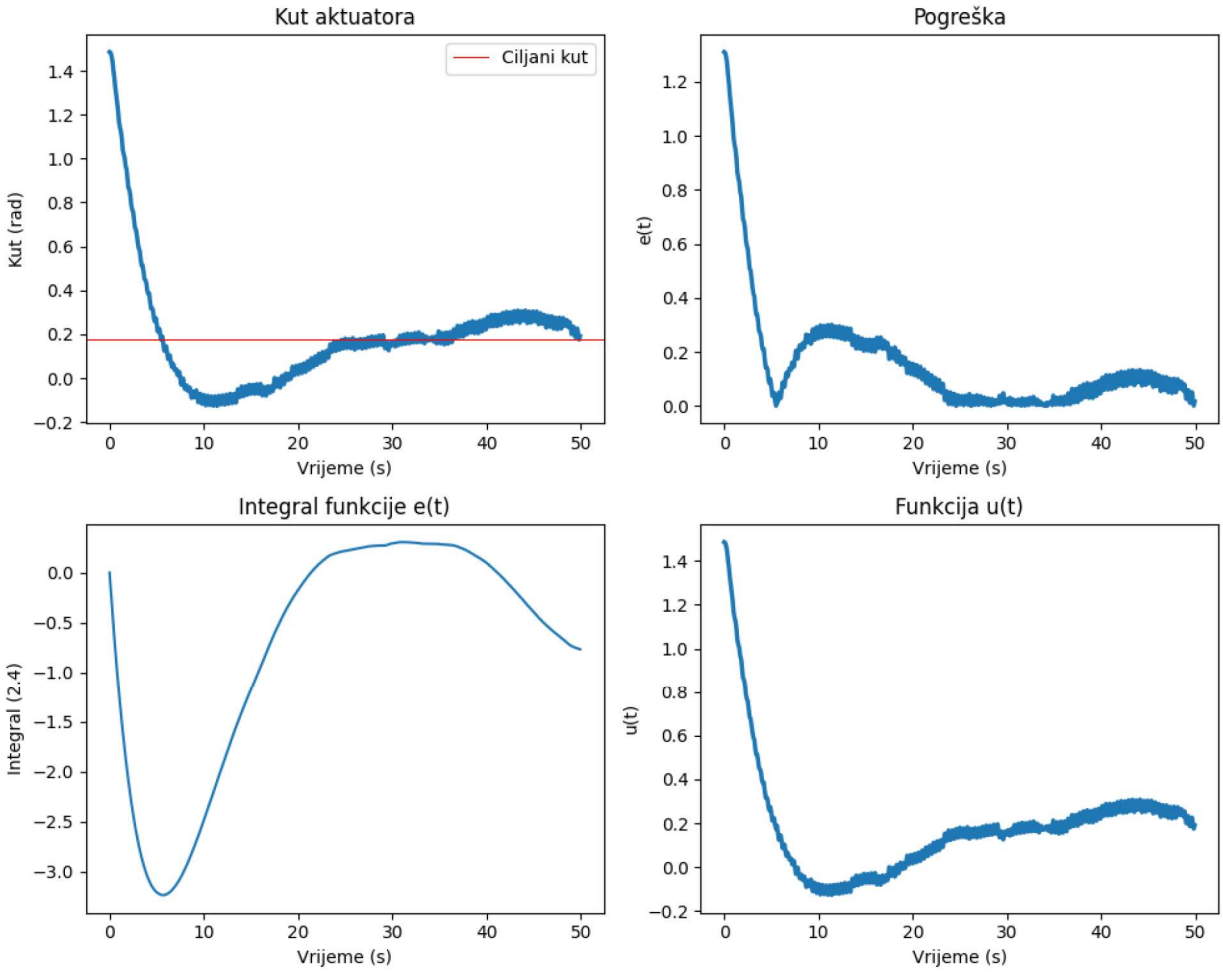
Ako povećamo proporcionalni i integracijski koeficijent tako da su sada vrijednosti koeficijenata $K_p = 2.5, K_i = 1.5, K_d = 0.3$. Na grafičkom prikazu (Slika 12) primjećuju se promjene u ponašanju regulatora.



Slika 12: Koeficijenti $K_p = 2.5, K_i = 1.5, K_d = 0.3$

Kada pogledamo graf kuta možemo primjetiti kao što je očekivano povećanjem proporcionalnog i integracijskog koeficijenata došlo do ubrzanja vremena dolaska do ciljane vrijednosti koje je sada malo iznad 10 sekundi. Isto tako prema očekivanjima primjećujemo veće premašivanje ciljane vrijednosti te također oštrije kretanje grafa pogreške i dolaska do većih oscilacija prije stabiliziranja regulatora. Graf kontrole nam pokazuje kako je kod ovih koeficijenata kontrola poprimala veće vrijednosti što je u skladu s agresivnijim i bržim dolaskom do ciljane vrijednosti kuta i povećanjem oscilacija i premašivanja prilikom dolaska.

Kada bismo povećali derivacijski koeficijent tako da naši koeficijenti iznose $K_p = 1.5, K_i = 0.2, K_d = 2.5$ ponašanje regulatora će se drastično promijeniti (Slika 13).



Slika 13: Koeficijenti $K_p = 1.5, K_i = 0.2, K_d = 2.5$

Prema očekivanjima povećanje derivacijskog koeficijenta dovelo je smanjenja premašivanja ciljane vrijednosti. Međutim na grafovima se jasno može vidjeti kako je prevelika vrijednost koeficijenta dovela do nestabilnosti regulatora. Regulator pretjerano reagira na promjene u pogrešci i vrijednost kuta se ne stabilizira oko ciljane vrijednosti.

6. Zaključak

Tijekom rada upoznali smo se s idejom i teorijskom pozadinom PID regulatora. Došli smo do očekivanja kako bi se ispravno implementirani regulator trebao ponašati i što predstavlja svaki dio akronima PID te kako izmjena vrijednosti njegovim koeficijenata utječe na njegovo ponašanje. Zatim smo se ukratko upoznali s robotom nad kojim je implementiran regulator.

Tijekom implementacije PID regulatora na primjeru upravljanja rukom robota upravljali smo jednim zglobom ruke. Na isti način na koji smo upravljali tim zglobom koji kontrolira kut ramena možemo upravljati svakim zglobom robota i dizajnirati kompleksnije pokrete poput mahanja ili udarca rukom robota. Mogućnosti upravljanja robotom pomoću PID regulatora su razne i otvaraju prostor istraživanjima i optimizaciji upravljanja.

Implementirali smo regulator korištenjem metoda trapeznog pravila za aproksimaciju integrala i metode konačnih diferencija za aproksimaciju derivacija funkcije pogreške. Tijekom rada regulatora pratili smo ponašanje regulatora spremanjem različitih podataka o kontroliranoj varijabli i kontrolnoj funkciji.

Prikupljeni podaci potvrđuju naša očekivanja o ponašanju regulatora pri dobro odabranim koeficijentima. Isto tako regulator reagira prema očekivanjima pri promjenama koeficijenata. Implementacija PID regulatora i njegova teorijska pozadina su jako intuativni, ali jednako toliko i efektivni.

7. Literatura

[1] K.J.Astrom, T.Hagglund, PID Controllers, Theory, Design and Tuning (2nd Edition), Instrument Society of America, 1995

[2] K.J.Astrom, Control System Design, Department of Mechanical and Environmental Engineering University of California Santa Barbara, 2002, https://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/astrom-ch6.pdf

[3] J. P. Hespanha, Linear Systems Theory, 2nd Edition, Princeton University Press, 2018

[4] N. Kofinas, Forward and Inverse Kinematics for the NAO Humanoid Robot, Technical University of Crete, Greece Department of Electronic and Computer Engineering, 2012

[5] Xh. Mehmeti, Adaptive PID controller design for joints of Humanoid Robot, IFAC-PapersOnLine 52(2019),
https://www.sciencedirect.com/science/article/pii/S240589631932364X

[6] T. Sauer, Numerical Analysis, Pearson, 2012

[7] R.Scitovski, Numerička matematika, 3. izmijenjeno i dopunjeno izdanje, Odjel za matematiku, Sveučilište u Osijeku, 2012

[8] I. Tejado, B. M. Vinagre, J. E. Traver, J. Prieto-Arranz, C. Nuevo-Gallardo, Back to Basics: Meaning of the Parameters of Fractional Order PID Controllers, Mathematics 7(2019), https://doi.org/10.3390/math7060530

[9] https://unitedrobotics.group/en/robots/nao, 14.9.2023.

[10] http://doc.aldebaran.com/1-14/naoqi/motion/control-joint-api.html, 14.9.2023.

[11] https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID, 14.9.2023.

[12] https://python-control.readthedocs.io/en/0.9.4/, 14.9.2023.