Proračun aktivne kontrole vibracija za građevinu

Smolčić, Marko

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij strojarstva

Završni rad

PRORAČUN AKTIVNE KONTROLE VIBRACIJA ZA GRAĐEVINU

Rijeka, rujan 2022.

Marko Smolčić

0069077807

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Preddiplomski sveučilišni studij strojarstva

Završni rad

PRORAČUN AKTIVNE KONTROLE VIBRACIJA ZA GRAĐEVINU

Mentor: prof. dr. sc. Zlatan Car

Rijeka, rujan 2022.

Marko Smolčić

0069077807

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 8. ožujka 2021.

Zavod: **Zavod za automatiku i elektroniku**
Predmet: **Automatizacija**
Grana: **2.11.01 opće strojarstvo (konstrukcije)**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Marko Smolčić (0069077807)**
Studij: Preddiplomski sveučilišni studij strojarstva

Zadatak: **Proračun aktivne kontrole vibracija za građevinu / Calculation of active vibration control for a building**

Opis zadatka:

Dati pregled sustava za aktivnu kontrolu vibracija. Razviti model aktivne kontrole vibracija građevine u Python programskom jeziku. Podešavati parametre Active Mass Driver (AMD) upravljačkog sustava za kontrolu vibracija sa ciljem poboljšanja performansi. Komentirati primjenu modela u realnim sustavima.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Marko Smolčić

Zadatak uručen pristupniku: 15. ožujka 2021.

Mentor:

Prof. dr. sc. Zlatan Car

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Kristian Lenić

IZJAVA

Izjava kojom ja, Marko Smolčić, JMBAG: 0069077807, student Tehničkog fakulteta u Rijeci, smjera preddiplomskog sveučilišnog studija strojarstva, izjavljujem da sam završni rad naslova „Proračun aktivne kontrole vibracija za građevinu“ iz kolegija Automatizacija, izradio samostalno uz mentorstvo prof. dr. sc. Zlatana Cara, a pri izradi mi je pomogao asistent dr. sc. Nikola Anđelić, mag. ing. mech. U završnom radu sam primijenio metodologiju znanstveno istraživačkog rada i koristio navedenu literaturu.

ZAHVALA

Zahvaljujem se svojoj majci Meliti koja me je cijeli život odgajala i podupirala, poticala moju radoznalost kroz odrastanje tako što mi je kupovala enciklopedije i bila mi potpora u onome što sam htio raditi i postati, i što se nije uplitala u moj izbor zanimanja već je poštovala moju slobodnu volju.

Zahvaljujem se svojoj starijoj sestri Martini i njenom mužu Osricu koji su mi bili kao drugi roditelji, podupirali me financijski i moralno tijekom studija i razumjeli me kada sam bio psihički nestabilan i čak i kada sam se ponašao nezahvalno prema njima.

Zahvalan sam svojoj zaručnici Luciji što mi je uljepšala studentske dane, bila mi podrška i donosila obroke na posao kada sam radio kao noćni čuvar i uz to učio za svoj studij pa tako i pisao ovaj rad. Hvala joj što je sa mnom dijelila radost za svaki ispit koji sam prošao.

Zahvaljujem se odgajatelju Anti Marjanoviću iz učeničkog doma Kvarner i razredniku Franji Rimpfu iz srednje škole koju sam pohađao, Prometne škole Rijeka, što su me bodrili i razumjeli moju teško obiteljsku situaciju u mojim adolescentskim godinama života.

Zahvalan sam Riječkoj nadbiskupiji, njezinim svećenicima Ivanu Dominiku Iličiću i Josipu Pendeu koji su me duhovno jačali i animirali, te ponudili financijsku pomoć. Veoma sam zahvalan svojoj braći u duhu iz Bratstva Blaženog Miroslava Bulešića, a posebno Mateju Pedišiću, mom krizmanom kumu i studentu medicine, što mi je pomogao pronaći posao kojim se mogu uzdržavati uz studiranje.

Hvala tvrtki GP Krk što mi je omogućila radno mjesto na kojem sam mogao obavljati svoje obaveze na fakultetu i biti financijski neovisan u isto vrijeme, kao i tvrtki Elcon Geraetebau u kojoj sam odradio studentsku praksu i dobio prvi uvid u primjene strojarstva.

Zahvaljujem se Republici Hrvatskoj i Europskoj Uniji, što su mi omogućili stipendiju, smještaj u studentskom domu i školovanje.

Zahvaljujem svim kolegama i prijateljima sa svog studijskog smjera, a posebno Igoru Šipeku, Matiji Tothu, Luki Vrbaniću i Karmelu Škodi s kojima sam zajedno učio i koji su mi pomagali kad god je trebalo, kao i svim profesorima i asistentima Tehničkog fakulteta Sveučilišta u Rijeci na njihovom trudu kod izvođenja nastave.

Što se tiče samog ovog završnog rada, najviše mogu zahvaliti profesoru dr. sc. Zlatanu Caru što je prihvatio biti moj mentor, asistentu dr. sc. Nikoli Anđeliću, mag. ing. mech, koji mi je pomagao svojim smjernicama i uputama, te asistentu Sebastijanu Blaževiću, mag. ing. mech, mag. ing. el., koji je davao kvalitetne poduke iz kolegija Automatizacija.

Zahvaljujem svom prijatelju Darku Palaiću, mag. ing. el., što mi je davao instrukcije iz automatizacije i podučavao me programiranju u Pythonu te mi predložio da problem ovog rada riješim genetskim algoritmom, što se pokazalo korisnim.

Zahvaljujem Anti Šegi, studentu preddiplomskog sveučilišnog studija elektrotehnike Tehničkog fakulteta u Rijeci, na pomoći kod lektoriranja ovog rada.

SADRŽAJ

1. UVOD

Ovaj rad se temelji na postojećem istraživanju [1] i bavi se redukcijom vibracija uzrokovanih potresom u trokatnoj zgradi pomoću genetskog algoritma.

Kako se temelji uslijed potresa pomiču, a vrh zgrade je zbog tromosti mase u zaostatku u odnosu na gibanje temelja, dolazi do iskrivljenja zgrade i nastanka pukotina koje u najgorem slučaju mogu uzrokovati urušavanje zgrade. Kako bi krov i temelji bili usklađeni, odnosno jednako se gibali u istom smjeru i istim intenzitetom, na vrhu zgrade postavljen je uteg koji se pomiče pomoću hidrauličkih klipova aktuatora. Cilj je dakle koordiniranim kretanjem utega na vrhu zgrade prigušiti vibracije i skratiti njihovo trajanje.

Za simulaciju potresa koriste se podatci o potresu u El centro, Meksiko, iz 1940. godine, te o potresu u Japanu 1968. godine koji je nanio značajnu štetu trokatnoj zgradi Tehničkog fakulteta u mjestu Hachinohe. Podatci o potresu u El Centru iz 1940. godine se uzimaju kao baza za mnoga istraživanja jer je to prvi potres koji je bio praćen modernim seizmografima, te prvi za koji su nam dostupni podatci o pomacima i ubrzanju tla.

Za analizu učinaka potresa na građevinu ne koriste se prave građevine u punoj veličini, već umanjeni model trokatnice zbog praktičnosti i ekonomičnosti takvoga pristupa.

Za podešavanje parametara kontrolera koristi se tzv. „genetski algoritam“ s više različitih postavki parametara, te se dobiveni rezultati međusobno uspoređuju.

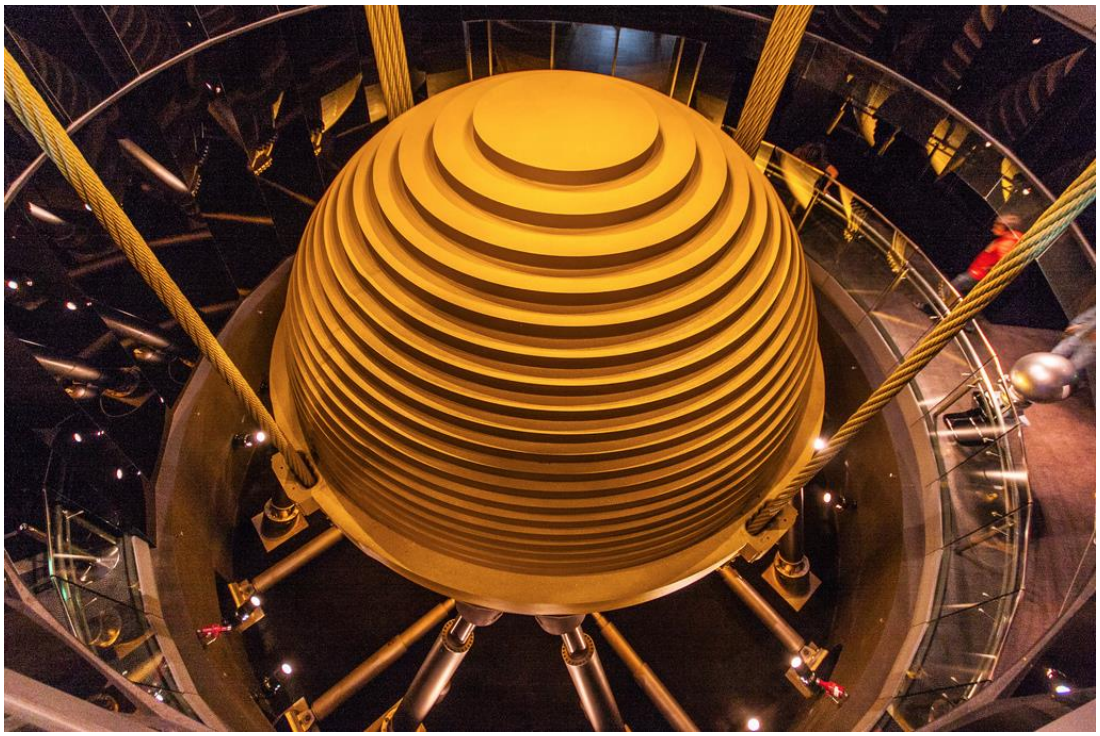
Ovim radom nastoji se evaluirati relativnu efektivnost i praktičnost raznih kontrolnih algoritama kako bi se stvorili temelji istraživanja za buduća rješenja problema, među kojima su istovjetnost rezultata dobivenih na umanjenim modelima i građevina u prirodnoj veličini, ispitivanje interakcije konstrukcija i kontrolnih uređaja, ograničenja u pogledu kontrole, šum senzora, zakašnjenje signala, a ponajviše efektivnost genetskog algoritma u rješavanju ovog i sličnih problema.

2. KLASIFIKACIJA KONTROLE VIBRACIJA

Razlikuju se dvije glavne vrste kontrole vibracija:

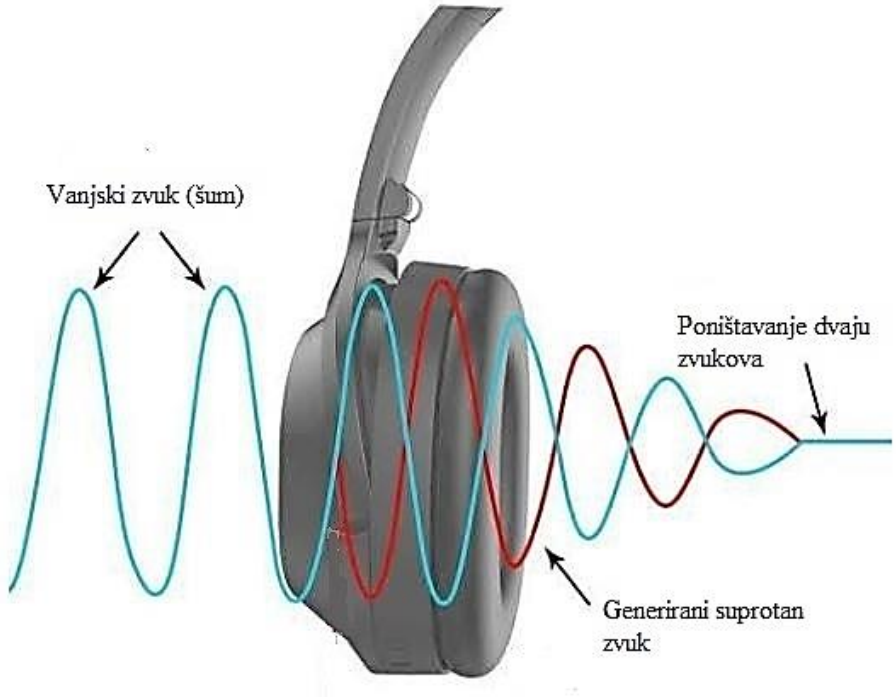
- pasivna kontrola vibracija i
- aktivna kontrola vibracija.

Pasivna kontrola vibracija je kontrola pomoću pasivnog prigušivača, to jest prigušivača kojem nije potreban vanjski izvor energije da bi funkcionirao. Pasivni prigušivači se izvode kao spojevi na kritičnim mjestima obično izrađenim od gume i drugih materijala, koje odlikuje laka pretvorba mehaničke energiju gibanja i titranja nekog objekta u toplinsku energiju. Sustavi pasivne kontrole vibracija su češće upotrebljavani u građevinskoj industriji od aktivne, zbog toga što velika masa nije problem za nepokretne građevine i zbog pouzdanosti jer ne zahtjeva vanjski izvor energije. Iz tog razloga mnogi neboderi imaju pasivne sustave kontrole vibracija koji se koriste uslijed potresa i jakog vjetra. Kod vrlo visokih nebodera, ponekad se koriste prigušivači prilagođene mase. Takvi prigušivači sastoje se od velikog utega, čija je masa podešena s obzirom na rezonantnu frekvenciju nebodera, koja je pak funkcija mase, krutosti konstrukcije i njenih dimenzija. Uteg je elastičnim užetima i klipnim prigušivačima pričvršćen za nosivu konstrukciju zgrade, te svojih asinkronim gibanjem u odnosu na zgradu smanjuje vibracije.



Slika 2.1. Prigušivač prilagođene mase u neboderu Taipei 101, s utegom kuglastog oblika teškim 660 tona [2]

Sustavi aktivne kontrole vibracija su pak vrlo rijetki u građevinarstvu te se najviše koriste kod pokretnih objekata. Najpoznatiji i najrašireniji primjer aktivne kontrole vibracija je *aktivno stišavanje šuma* (engl. *Active Noise Cancelation*). To je naziv za tehnologiju reduciranja šuma koji dolazi izvana kod korištenja slušalica.



Slika 2.2. Princip rada tehnologije aktivnog stišavanja zvuka [3]

Djelovanje sustava je ostvareno putem mikrofona sa vanjske strane slušalica, koji mjeri zvuk vanjskog šuma i prema izmjerenom zvuku kontroler unutar slušalica daje upute zvučniku da uz zvuk koji reproducira (govor, glazba), proizvodi i zvuk suprotan vanjskom šumu, koji je jednake frekvencije i amplitude kao i vanjski šum ali suprotne orijentacije odnosno negativnog predznaka. Iz razloga što su ta dva zvuka jednaka, ali suprotno orijentirana, odnosno u protufazi, se poništavaju i u idealnom slučaju šum nestaje ili dolazi do značajnog smanjenja istog. Često se dodaje još jedan mikrofon s unutarnje strane slušalice, okrenut prema uhu, koji snima dobiveni zvuk, te ga šalje kontrolnom sustavu koji ga uspoređuje sa željenim referentnim zvukom (govor, glazba) kako bi proračunao pogrešku.

Osnovni princip aktivne kontrole vibracija jednak je u svim primjenama. Svodi se na detektiranje poremećaja, u ovom slučaju mehaničkih vibracija, i proračuna najekonomičnijeg načina proizvodnje suprotnih vibracija koje će u potpunosti pobiti neželjene vibracije ili ih svesti na prihvatljivu razinu.

3. UTJECAJ RAZLIČITIH TIPOVA POTRESA NA GRAĐEVINE

Najpoznatija podjela potresa je po njihovoj jačini, i za nju se koriste dobro poznate Richterova i Mercallijeva ljestvica.

Tablica 3.1. Mercallijeva ljestvica [4]

Stupanj	Podrhtavanje	Opis štete
I.	Nezamjetljiv	Bilježe ga isključivo seizmografi.
II.	Jedva osjetan	U višim katovima stambenih zgrada osjete ga vrlo senzibilni ljudi.
III.	Lagan	Osjeti se podrhtavanje tla kao pri prolazu automobila, u unutrašnjosti zgrada primijeti ga više ljudi.
IV.	Umjeren	U zgradama ga osjeti više ljudi, a na otvorenome samo pojedinci, tresu se vrata, prozori, pokućstvo, staklenina i posude zveče kao pri prolazu teških kamiona.
V.	Prilično jak	Osjeti ga više ljudi na otvorenom prostoru, njišu se predmeti koji slobodno vise, zaustavljaju se ure njihalice.
VI.	Jak	Sa zidova padaju slike, ruše se predmeti, razbija se posuđe, pomiče ili prevrće pokućstvo, zvone manja crkvena zvona, oštećuju se pojedine dobro građene kuće.
VII.	Vrlo jak	Crjepovi se lome i kližu s krova, ruše se dimnjaci, oštećuje se pokućstvo u zgradama, pucaju zidovi, ruše se slabije građene zgrade, a na jačima nastaju oštećenja.
VIII.	Razoran	Znatno oštećuje do 25% zgrada, pojedine se kuće ruše do temelja, a velik ih je broj neprikladan za stanovanje, u tlu nastaju pukotine, a na padinama klizišta.
IX.	Pustošan	Oštećuje 50% zgrada, mnoge se ruše, a većina ih je neuporabljiva, u tlu nastaju velike pukotine, a na padinama klizišta i odroni.
X.	Uništavajući	Teško oštećuje 75% zgrada, velik broj dobro građenih kuća ruši se do temelja, ruše se mostovi, pucaju brane, savijaju željezničke tračnice, oštećuju putevi, urušavaju se špilje, izbija podzemna voda.

Mercallijeva ljestvica, prikazana u tablici 3.1., nema matematičku podlogu i njen stupanj utvrđuje se isključivo na temelju opažanja vidljive štete uzrokovane potresom. Za razliku od Mercallijeve, Richterova ljestvica ima matematičku podlogu i označava energiju oslobođenu potresom. Richterova je ljestvica logaritamska, s bazom 10, što znači da će npr. potres jačine 5 po Richterovoj ljestvici biti 10 puta jači od potresa jačine 4 po Richterovoj ljestvici.

Postoje vertikalni i horizontalni potresi [5]. Kod vertikalnih potresa tlo se uglavnom pomiče u pravcu okomice na tlo dok se kod horizontalnih tlo pomiče u ravnini tla. Većina potresa ipak nije strogo vertikalna niti strogo horizontalna, već vibracije uglavnom dolaze iz svih smjerova. Za istu vrijednost po Richterovoj ljestvici horizontalni potres razorniji je od vertikalnog. Ta činjenica je prilično intuitivna kada se u obzir uzmu mehaničke karakteristike betona, posebice betona koji nije ojačan željezom ili nekim drugim ojačanjem. Kod horizontalnih potresa zgrada podrhtava u smjeru širine, uzrokujući tako velika savijanja zidova zgrade, što na konveksnoj strani savijanja zida stvara velika vlačna naprezanja na koja je beton posebno osjetljiv. Kod vertikalnog potresa vlačnih naprezanja gotovo da ni nema, jer je konstrukcija zgrade svojom težinom ionako najviše opterećena tlačno, što materijali korišteni u graditeljstvu vrlo dobro podnose.

U ovom radu u obzir su uzete samo horizontalne vibracije, pod pretpostavkom da vertikalne vibracije imaju vrlo slab ili nikakav utjecaj na stabilnost građevine.

3.1. Prirodna frekvencija zgrade

Naziv „rezonantna frekvencija“ se često koristi umjesto pojma prirodne frekvencije jer se rezonancija javlja kada pobuda frekvencije jednake prirodnoj frekvenciji nekog objekta djeluje na taj objekt. Rezonancija podrazumijeva drastično povećanje amplitude vibracija, što može dovesti do nestabilnosti sustava, odnosno u slučaju mehaničkih vibracija, do pucanja objekta.

Razina oštećenja nakon potresa ne ovisi samo o snazi vibracija, već i o frekvencijskom spektru vibracija i veličini građevina na pogođenom području.

Na veće građevine više djeluju niže frekvencije i veće amplitude vibracija dok na male građevine više utječu više frekvencije i manje amplitude. Iz tog razloga se u sljedećem poglavlju za umanjeni model zgrade frekvencije stvarnih potresa koji su znatno oštetili trokatne zgrade povećavaju pet puta, budući da je umanjeni model otprilike pet puta manje visine od građevine koju se proučava u prirodnoj veličini. Građevine koje imaju manje od pet katova se smatraju malim građevinama. Srednje velikim se smatraju građevine od pet do deset katova, a sve zgrade sa više od deset katova se smatraju velikim građevinama. Iz toga slijedi da su trokatnice smatrane malim građevinama, te

na njih najviše djeluju više frekvencije podrhtavanja tla, no ipak ne tako visoke kao na još manje građevine, kao što su npr. prizemnice i jednokatnice.

Pravi je razlog zašto veća visina građevine podrazumijeva nižu prirodnu frekvenciju zgrade to što je visina približno proporcionalna masi zgrade, a obrnuto proporcionalna krutosti zgrade [6].

Kako je formula za prirodnu frekvenciju sljedeća:

$$f = \frac{1}{2\pi}\sqrt{\frac{k}{m}}, \tag{3.1}$$

prirodna frekvencija je to niža što je masa *m* veća, a krutost *k* manja.

Krutost visokih zgrada je obično manja jer visoke zgrade često imaju duguljasti oblik, odnosno njihov odnos visine je puno veći naspram širine temelja, a često se i namjerno grade kako bi bile savitljive u cilju smanjenja mogućnosti razaranja tokom potresa i jakih vjetrova. To međutim ima za posljedicu jako ljuljanje viših katova, što iako nije nužno prijetnja strukturnoj stabilnosti zgrade, može loše utjecati na udobnost stanara i radnika u takvoj zgradi.

Samo tlo na kojem se zgrada nalazi također ima određenu prirodnu frekvenciju pri kojoj se javlja rezonantno pojačanje vibracija. Ako se temeljne prirodne frekvencije tla i zgrade koja se na tom tlu nalazi poklapaju, to će ishoditi k tomu da vibracije na koje je zgrada najosjetljivija budu dodatno pojačane i time će potencijalna šteta na građevini prilikom potresa biti uvećana [7].

4. MODEL GRAĐEVINE

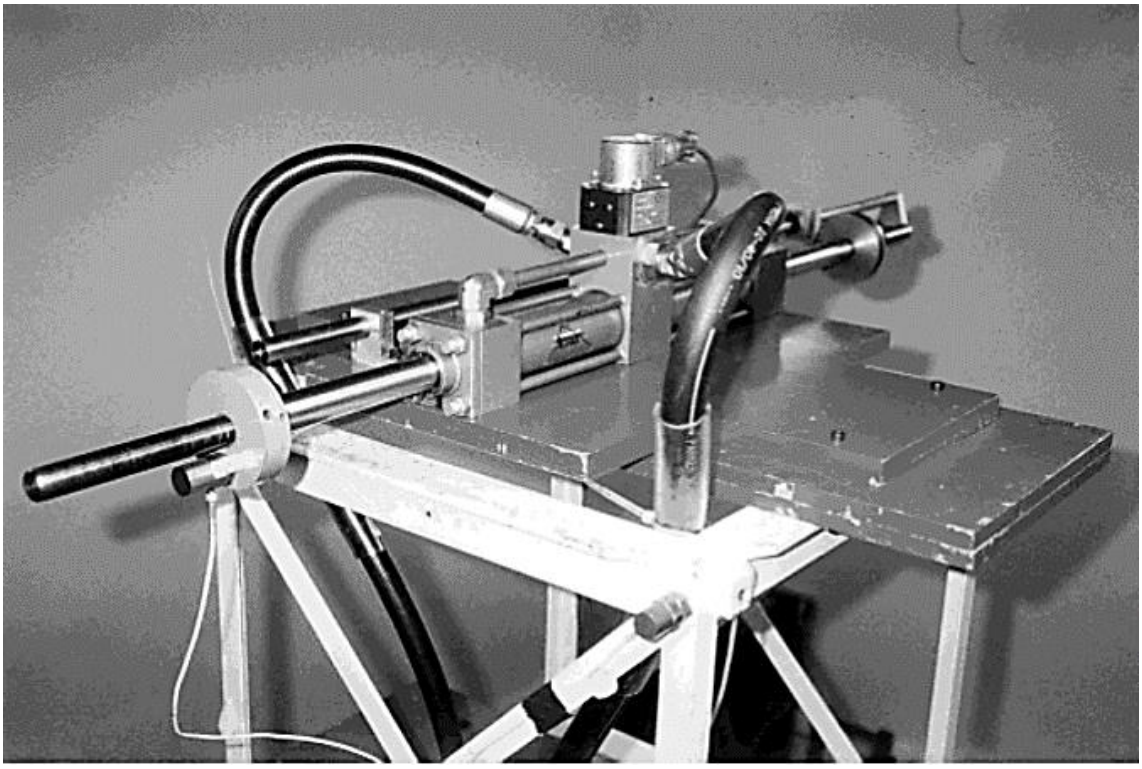
Ovo istraživanje nadovezuje se na istraživanje [1] u kojem su autori izradili i istraživali umanjeni model građevine i razvili matematički model sustava u obliku prijenosne funkcije u prostoru stanja.

4.1. Eksperimentalni model

Eksperimentalni model odnosno prototip građevine je umanjeni model trokatnice. Visok je 158 [cm] i izrađen od čelika. Masa je ravnomjerno raspoređena na sva tri kata i ukupno iznosi 227 [kg]. Sam okvir modela mase je 77 [kg].

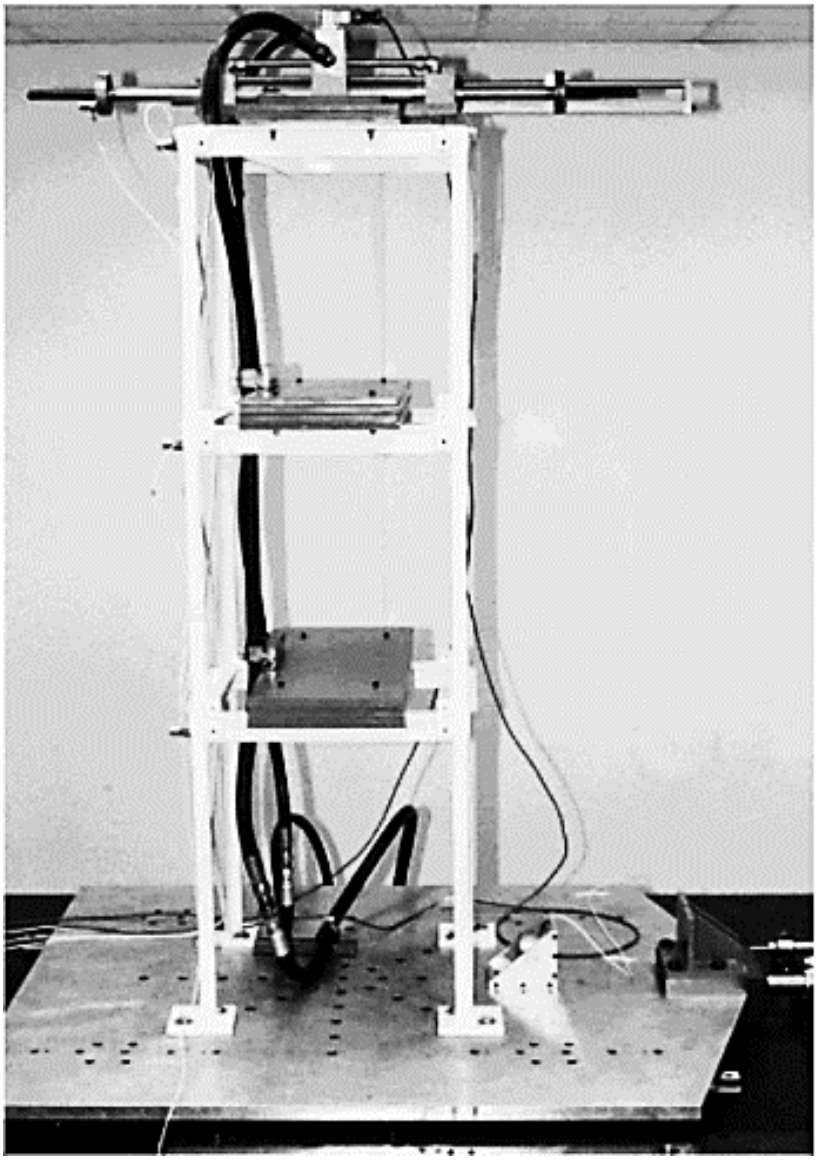
Budući da se radi o umanjenom modelu, vrijeme trajanja potresa je umanjeno množenjem s koeficijentom 0,2, te time uvećava prirodnu frekvenciju modela za otprilike pet puta.

Ukupna masa strukture, uključujući okvir i pokretač mase utega (AMD)(engl. *Active Mass Driver*) [1], je 309 [kg].



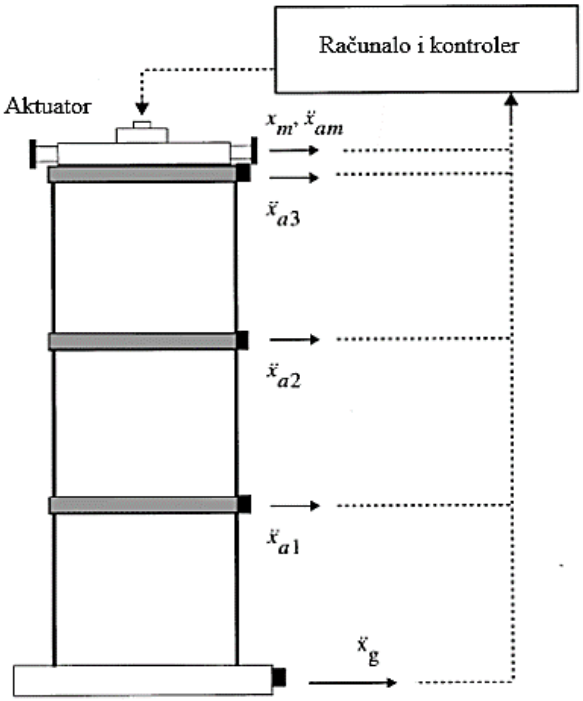
Slika 4.1. Pokretač mase [1]

AMD je smješten na stropu trećeg kata strukture. Sastoji se od jednog hidrauličkog aktuatora, hidrauličkog cilindra i utega pričvršćenih na klipove. Hidraulički cilindar promjera je 3,8 [cm] i dužine je 30,5 [cm]. Ukupna masa koja se pokreće je 5,2 [kg], i sastoji se od klipa, šipke i čeličnih diskova vijcima pričvršćenim na krajeve šipki. Iz toga proizlazi da je udio mase AMD-a u ukupnoj masi strukture tek oko 1,7%. AMD je prikazan na slici 4.1., a cijeli umanjeni model zgrade na slici 4.2.



Slika 4.2. Umanjeni model zgrade [1]

Na modelu su na više mjesta postavljeni akcelerometri. Po jedan akcelerometar je postavljen na temeljima strukture, na svim katovima i na AMD-u, kao što je prikazano na slici 4.3.



Slika 4.3. Shema katova i lokacija akcelerometara [1]

4.1. Evaluacijski model

Za matematički prikaz, evaluaciju i podešavanje parametara sustava opisanog u prethodnom poglavlju koristi se predstavljanje diferencijalne jednadžbe gibanja zgrade metodom prostora stanja. Metoda prostora stanja pogodna je jer olakšava proračun digitalnim računalima [8], te omogućava jednostavno i intuitivno korištenje sustava sa više ulaznih i izlaznih vrijednosti (MIMO)(engl. *Multiple Input, Multiple Output*). Metodom prostora stanja moguće je jednu diferencijalnu jednadžbu *n*-tog reda prikazati sa *n* diferencijalnih jednadžbi prvog reda.

Prostorno i vremenski se eksperimentalni model može predočiti i jednadžbama:

$$\dot{x} = Ax + Bu + E\ddot{x}_g, \tag{4.1}$$

$$y = C_y x + D_y u + F_y \ddot{x}_g + v, \tag{4.2}$$

$$z = C_z x + D_z u + F_z \ddot{x}_g\ , \tag{4.3}$$

gdje *x* predstavlja vektor stanja, $\ddot{x}_g$ skalar ubrzanja temelja, *u* skalar kontrolnog ulaza, a *y* vektor pobuda koje se javljaju kao rezultat vibracija tla i mogu se direktno izmjeriti;

$$y = [x_m \quad \ddot{x}_{a1} \quad \ddot{x}_{a2} \quad \ddot{x}_{a3} \quad \ddot{x}_{am} \quad \ddot{x}_g], \tag{4.4}$$

dok je *z* vektor pobuda koje se mogu regulirati:

$$z = [x_1 \quad x_2 \quad x_3 \quad x_m \quad \dot{x}_1 \quad \dot{x}_2 \quad \dot{x}_3 \quad \dot{x}_m \quad \ddot{x}_{a1} \quad \ddot{x}_{a2} \quad \ddot{x}_{a3} \quad \ddot{x}_{am}]. \tag{4.5}$$

Ovdje je x_i pomak *i*-tog kata u odnasu na tlo, x_m je pomak pokretača mase u odnosu na treći kat, $\ddot{x}_{ai}$ je apsolutno ubrzanje *i*-tog kata.

No zbog lakšeg računanja zanemaruje se u daljnjem razmatranju i proračunu jednadžba (4.3), u jednadžbama (4.1) i (4.2) koriste se samo prva dva člana, te su član koji označava pogrešku v i član $\ddot{x}_g$ koji označava akceleraciju tla kao i pripadna mu matrica izostavljeni. Tako preostaje samo osnovni oblik jednadžbe prostora stanja:

$$\dot{x} = Ax + Bu, \tag{4.6}$$

$$y = C_y x + D_y u. \tag{4.7}$$

Ulazna vrijednost sustava je ubrzanje zgrade $\ddot{x}_g$ koja je definirana kao stacionarni slučajni proces sa spektralnom gustoćom filtriranom *Kanai-Tajimi* spektrom:

$$S_{\ddot{x}_g} = \frac{S_0(4\zeta_g^2\omega_g^2\omega^2 + \omega_g^2}{\left(\omega^2 - \omega_g^2\right)^2 + 4\zeta_g^2\omega_g^2\omega^2}, \tag{4.8}$$

gdje su ω_g i ζ_g nepoznati ali se po pretpostavci vrijednost ω_g nalazi u rasponu od 20 do 120 radijana po sekundi, a ζ_g u rasponu od 0,3 do 0,75. Spektralni intezitet je odabran tako da $\sigma_{\ddot{x}_g}$ iznosi 0,12 g.

$$S_0 = \frac{0{,}03\zeta_g}{\pi\omega_g(4\zeta_g^2 + 1)}\,\mathrm{g}^2[\mathrm{s}] \tag{4.9}$$

Kako se sustav sastoji samo od akcelerometara a ne i brzinomjera, sve brzine su dobivene integriranjem izmjerenih ubrzanja prolaskom kroz filter drugog reda sa sljedećom jednadžbom prijenosne funkcije:

$$H_{\tilde{\dot{x}}\ddot{x}} = \frac{39{,}5\ s}{39{,}5\ s^2 + 8{,}89\ s + 1}, \tag{4.10}$$

gdje je $\tilde{\ddot{x}}$ pseudo ubrzanje jer se odnosi samo na odzive apsolutne brzine >1[Hz]. Filter ustvari propušta niske frekvencije, ali je ovdje upotrebljen kao integrator.

4.2. Osobine modela sustava

Navedeni sustav je linearan i vremenski nepromjenjiv (LTI) (engl. *Linear Time Invariant system*). Vremenski nepromjenjiv sustav ima konstantne parametre što znači da se parametri sustava ne mijenjaju tijekom vremena [9]. Pritom valja odvojiti pojam parametra i varijable. Parametri su koeficijenti članova u jednadžbi koji se ne mijenjaju dok varijable predstavljaju ulazne i izlazne veličine koje se mijenjaju s vremenom.

Linearnost sustava podrazumijeva da sustav zadovoljava uvjete aditivnosti i homogenosti. Za sustav za koji vrijedi:

$$f(x_1 + x_2) = f(x_1) + f(x_2), \tag{4.11}$$

kaže se da je aditivan.

Kada pak vrijedi sljedeća tvrdnja:

$$f(\alpha x) = \alpha f(x), \tag{4.12}$$

kaže se da je sustav homogen.

U ovome slučaju vrijede obje tvrdnje budući da se radi reprezentaciji pomoću prostora stanja.

Sustav koristi koncentrirane parametre jer su samo na sjecišta svakog kata i vanjskog zida te na temelj i krov građevine postavljeni mjerni uređaji, akcelerometri. Time se sustav idealizira na samo tih pet parametara ubrzanja iz čega se dalje integriranjem po vremenu dobivaju brzina i pomak tih istih točaka. Koncentriranje parametara je važno jer se njime model pojednostavljuje kako bi računanje bilo olakšano, istovremeno pružajući dovoljno točne rezultate. Sustavi s koncentriranim parametrima se još nazivaju konačno dimenzijski sustavi. Realni sustavi uvijek u stvarnosti imaju raspodijeljene parametre.

Sustav nije kontinuiran već diskretan jer kao pobudu dobiva diskretan niz varijabli umjesto kontinuirane funkcije, te prema tome računa diskretan niz izlaznih varijabli. Podatci gibanja tla za trajanja potresa spomenutih u uvodu su diskretizirani na određenim vremenskim intervalima.

Sustav ima memoriju jer mu odziv ovisi o prošlim pobudama. Memorija se u ovom slučaju sastoji od položaja katova uslijed djelovanja podrhtavanja tla. Različiti položaji katova uzrokuju različite

količine energije u spremištima energije. Spremišta energije su ovdje kinetička energija gibanja katova i elastična potencijalna energija koja je sadržana u zidovima zgrade. Što je pomak katova veći od pozicije u ravnotežnom stanju, to će potencijalna elastična energija biti veća, dok je brzina gibanja katova obično najveća u trenutku prolaza kata kroz točku ravnotežnog stanja. Tako se na ovaj sustav može gledati kao na kompliciraniji mehanički sustav mase, opruge i prigušenja (engl. *Mass Spring Damper system*).

Prigušenje se poput elastične energije javlja u zidovima zgrade, pretvarajući dio kinetičke energije putem trenja u toplinu koja zatim isijava iz materijala zgrade i time smanjuje vibracije i čini cijeli sustav stabilnijim. Zbog toga što ovaj sustav ima više od jednog spremnika energije, energija se prelijeva iz kinetičke u potencijalnu i obrnuto, uzrokujući oscilacije koje se manifestiraju kao vibracije, ali također i kao odgovor na svaku novu pobudu.

I kao posljednju osobinu ovog sustava može se navesti kauzalnost. Ovaj sustav je kauzalan jer njegov odziv nikad ne dolazi prije pobude, odnosno potreban je uzrok da bi se dogodila posljedica.

4.3. Kontrolni problem i ograničenja

Za riješavanje problema optimizacije potrebno je odrediti vrijednosti koje je potrebno smanjiti i ograničenja kontrolnog napora (engl. *Control effort*) koji djeluje u cilju smanjenja željenih veličina. Kao cilj najbolje je zadati smanjenje pomaka između katova i ubrzanja katova, uz što manje kontrolnog napora. Pomaci između katova dobivaju se kao razlika pomaka pojedinačnih katova:

$$d_1 = x_1, \tag{4.13}$$

$$d_2 = x_2 - x_1, \tag{4.14}$$

$$d_3 = x_3 - x_2. \tag{4.15}$$

Uspjeh kontrolnog algoritma mjeri se prema najvećem pomaku trećeg kata u odnosu na tlo, zato što se u trokatnoj zgradi najviši treći kat najviše pomiće u odnosu na tlo. Najveći pomak trećeg kata se kod zgrade bez zaštite od potresa javlja kod frekvencije $\omega_g = 37{,}3$ radijana po sekundi, te za bezdimenzijsku vrijednost $\zeta_g = 0{,}3$ iznosi:

$$\sigma_{x_{3_0}} = 1{,}31 \text{ [cm]}. \tag{4.16}$$

Stoga je prvi evaluacijski kriterij koeficijent pomaka između katova i najgoreg mogućeg pomaka trećeg kata kod građevine bez zaštite od potresa:

$$J_1 = \max_{\omega_g,\zeta_g} \left\{ \frac{\sigma_{d_1}}{\sigma_{x_{3_0}}}, \frac{\sigma_{d_2}}{\sigma_{x_{3_0}}}, \frac{\sigma_{d_3}}{\sigma_{x_{3_0}}} \right\}. \tag{4.17}$$

Najveće je ubrzanje, pod istim uvjetima kao i za najveći pomak, jednako:

$$\sigma_{\ddot{x}_{a3_0}} = 1{,}79 \text{ [g]}. \tag{4.18}$$

Analogno prvom kriteriju postavlja se i drugi kriterij, po kojem se ocijenjuje uspješnost kontrolnog algoritma u smanjenju ubrzanja katova kao koeficijent ubrzanja svakog kata i maksimalnog ubrzanja trećeg kata bez zaštite od potresa:

$$J_2 = \max_{\omega_g,\zeta_g} \left\{ \frac{\sigma_{\ddot{x}_{a1}}}{\sigma_{\ddot{x}_{a3_0}}}, \frac{\sigma_{\ddot{x}_{a2}}}{\sigma_{\ddot{x}_{a3_0}}}, \frac{\sigma_{\ddot{x}_{a3}}}{\sigma_{\ddot{x}_{a3_0}}} \right\}. \tag{4.19}$$

Sustav ima stroga ograničenja za maksimalni apsolutni napon koji se šalje aktuatoru, maksimalni apsolutni pomak mase utega aktuatora i maksimalno apsolutno ubrzanje utega na aktuatoru, respektivno izraženo jednadžbama 4.20 - 4.22:

$$max|u(t)| \leq 3 \text{ [V]}, \tag{4.20}$$

$$max|x_m(t)| \leq 9 \text{ [cm]}, \tag{4.21}$$

$$max|\ddot{x}_{am}(t)| \leq 6 \text{ [g]}. \tag{4.22}$$

Također su u cilju smanjenja kontrolnog napora postavljena ograničenja vršnih korijena srednjih kvadratnih vrijednosti (RMS) (engl. *Root Mean Square*) napora, no kako su aritmetičke sredine svih vektora odziva jednake nuli, te kako je velik broj uzoraka u svakom od odziva, može se vjerno aproksimirati navedena ograničenja sa ograničenjima standardnih devijacija napona i standardnih devijacija pomaka i ubrzanja mase utega aktuatora:

$$\sigma_u \leq 1 \text{ [V]} \tag{4.23}$$

$$\sigma_{x_m} \leq 3 \text{ [cm]} \tag{4.24}$$

$$\sigma_{\ddot{x}_{am}} \leq 2 \text{ [g]}. \tag{4.25}$$

5. IZRADA KONTROLERA I PODEŠAVANJE PARAMETARA KONTROLERA GENETSKIM ALGORITMOM

U ovom radu razvijen je i primijenjen genetski algoritam zbog svoje jednostavnosti izrade i primjene. Upravo jednostavnost primjene čak i za najzamršenije probleme je jedna od ključnih prednosti svih evolucijskih algoritama, pa tako i genetskog. Jedina metoda koja je još jednostavnija od evolucijskih algoritama je nasumično pretraživanje prostora rješenja, ali ona zahtjeva još više računalnih kapaciteta i mnogo sporije, iako robusnije, konvergira ka rješenju. Zapravo se i genetski algoritmi također oslanjaju na nasumično pretraživanje rješenja, ali to poglavito vrijedi samo za početni stadij, odnosno pripremu za sam proces genetskog algoritma, kada se vrši inicijalizacija populacije. Upravo zato, uz malo igre riječi, se može reći da je među optimizacijskim metodama genetski algoritam za mnoge probleme optimalan.

Jedan od presudnih čimbenika za široku primjenu genetskih algoritama su i sve jača računala, koja istraživačima i znanstvenicima dopuštaju korištenje „računalno skupih“ metoda. Nekada su računala bila veoma ograničena pa su se više primjenjivale računalno manje zahtjevne metode. Taj trend korištenja sve više računalnih resursa vjerojatno će se nastaviti u dogledno vrijeme, sve dok neka nova metoda ne preuzme vodstvo.

Izbor jedinki koje preživljavaju smjenu generacija je na osnovu podobnosti a ne na osnovu dobi [10]. Od ukupno 200 jedinki u svakoj generaciji, odabire se polovica, odnosno 100 jedinki za kreiranje novih jedinki sljedeće generacije.

Svi ostali parametri su unaprijed zadane vrijednosti Python knjižnice *Pygad*. Zadane postavke u toj su knjižnici kako slijedi:

- prikaz broja s pomičnom točkom [11],
- model je stacionarnog stanja, odnosno neće se cijela populacija prethodne generacije izmijeniti istovremeno, nego roditelji ulaze u sljedeću generaciju, pa kako je broj potomaka odnosno novih jedinki koje zamjenjuju stare duplo manji od ukupnog broja jedinki u populaciji, generacijski jaz je 0,5 (100/200) [10],
- broj roditelja u sljedećoj generaciji: svi roditelji prelaze u sljedeću generaciju,
- selekcija je turnirska,
- križanje je s jednom točkom prekida,
- mutacija je slučajna,
- kriterija zaustavljanja nema, odnosno algoritam se zaustavlja nakon određenog broja generacija.

5.1. Strojna i programska oprema računala

Sve računske operacije navedene u nastavku su obavljane na računalu sljedećih specifikacija:

- procesor: AMD Ryzen 5 3600, šesterojezgreni,
- grafička kartica: AMD Radeon R7 200 Series,
- radna memorija: 16 GB i
- pohrana: Samsung SSD 980 250 GB.

Strojna oprema za izvođenje programa se može razlikovati ali to može utjecati, bolje ili lošije, na trajanje izvođenja algoritma i točnost rješenja.

Pogodan je jak procesor sa mnogo jezgri, jer što je više jezgri, to se više paralelnih procesa može obavljati istovremeno.

Programski paketi korišteni u procesu izrade računalnog modela i podešavanja parametara uključuje Microsoft Excel i uređivač programskog koda Spyder, verzije 5.2.2., koji je dio Anaconda 3 okruženja. Korištena je verzija Python programskog jezika 3.8.

Za uspješan rad programskog koda potrebne su sljedeće Python knjižnice: Numpy, Scipy, Matplotlib, Math, Pandas, Control Systems Library, Itertools, Pygad. U nastavku su navedene odgovarajuće naredbe za pozivanje prethodno spomenutih knjižnica.

```
import math
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
import control
from control import matlab as mtb
import pandas as pd
import itertools as it
import pygad
```

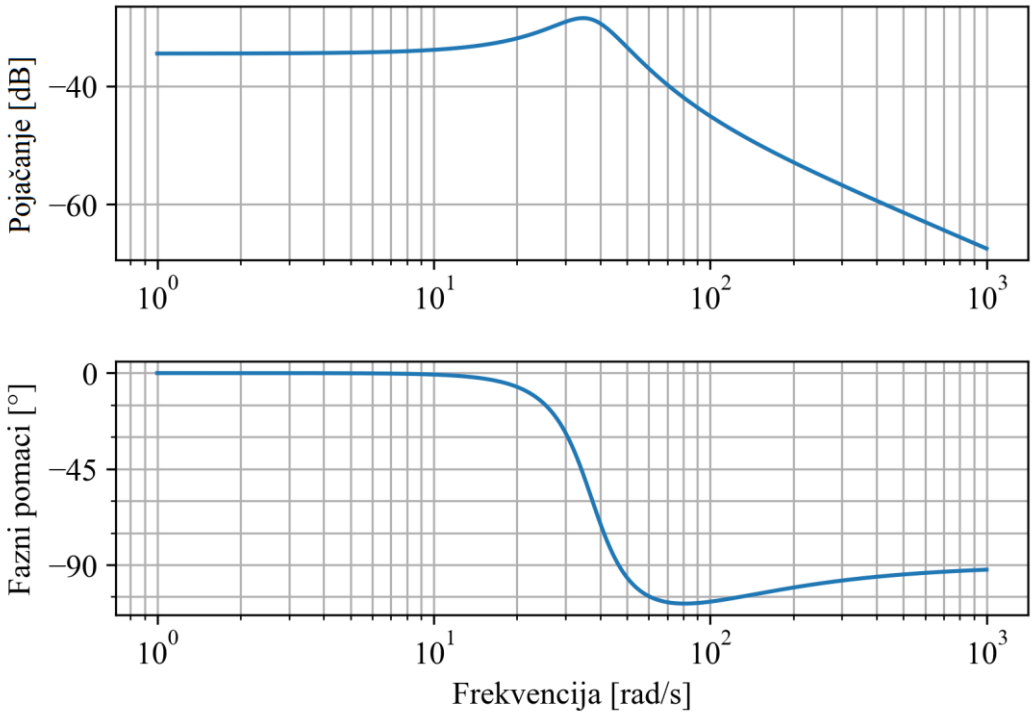
5.2. Stvaranje računalnog modela

Za razvoj računalnog modela potrebno je učitati četiri matrice prostora stanja koje su preuzete sa službene stranice tvrtke Mathworks [12]. Matrice *A, B, C* i *D* su kopirane i spremljene u MS Excel datoteke iz kojih su pomoću knjižnice Pandas učitane u program za uređivanje programskog koda Spyder. Te se matrice zatim komponiraju u sustav prostora stanja *P* pomoću naredbe control.matlab.ss() koja stvara novi sustav iz unesenih matrica.

```
PA = 'A.xlsx'
PB = 'B.xlsx'
PC = 'C.xlsx'
PD = 'D.xlsx'

PA = np.array(pd.read_excel(PA, header = None ))
PB = np.array(pd.read_excel(PB, header = None ))
PC = np.array(pd.read_excel(PC, header = None ))
PD = np.array(pd.read_excel(PD, header = None ))

P = mtb.ss(PA, PB, PC, PD)
```

Kao ulaz sustava koristi se bijeli šum koji prolaskom kroz *Kanai-Tajimi* filter daje nasumične vrijednosti akceleracija tla koje oponašaju vibracije potresa. Potrebno je izraditi *Kanai-Tajimi* filter i pomnožiti ga sa sustavom *P*, iz čega nastaje kombinacija sustava i filtra *PF*, te također izraditi Bodeov dijagram za prikaz gustoće frekvencija kao rezultat prolaska bijelog šuma kroz filter:

```
zg = 0.3
wg = 37.3
S0 = 0.03*zg/(math.pi*wg*(4*zg**2+1))
Numerator = [math.sqrt(S0)*2*zg*wg, math.sqrt(S0)*wg**2]
Denominator = [1,  2*zg*wg,  wg**2]

F = math.sqrt(2*math.pi)*mtb.tf(Numerator, Denominator)
plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
mtb.bode(F)
plt.xlabel('Frekvencija (rad/sec)')
plt.ylabel('Fazni pomaci (°)')
plt.show()

F1 = control.append(F,1)
PF = P*F1
```



Slika 5.1. Bodeov dijagram Kanai-Tajimi filtra

Iz slike 5.1. može se iščitati pojačanje izlaznih vrijednosti za pojedine frekvencije ulaza. U ovome primjeru na model trokatnice ponajviše utječu frekvencije između 30 i 40 [Hz], zato što su najbliže rezonantnoj frekvenciji zgrade. Međutim, kako se radi o umanjenom modelu, potrebno je podijeliti rezonantnu frekvenciju modela sa 5, čime se dobiva rezonantna frekvencija zgrade u prirodnoj veličini u rasponu od 6 do 8 [Hz].

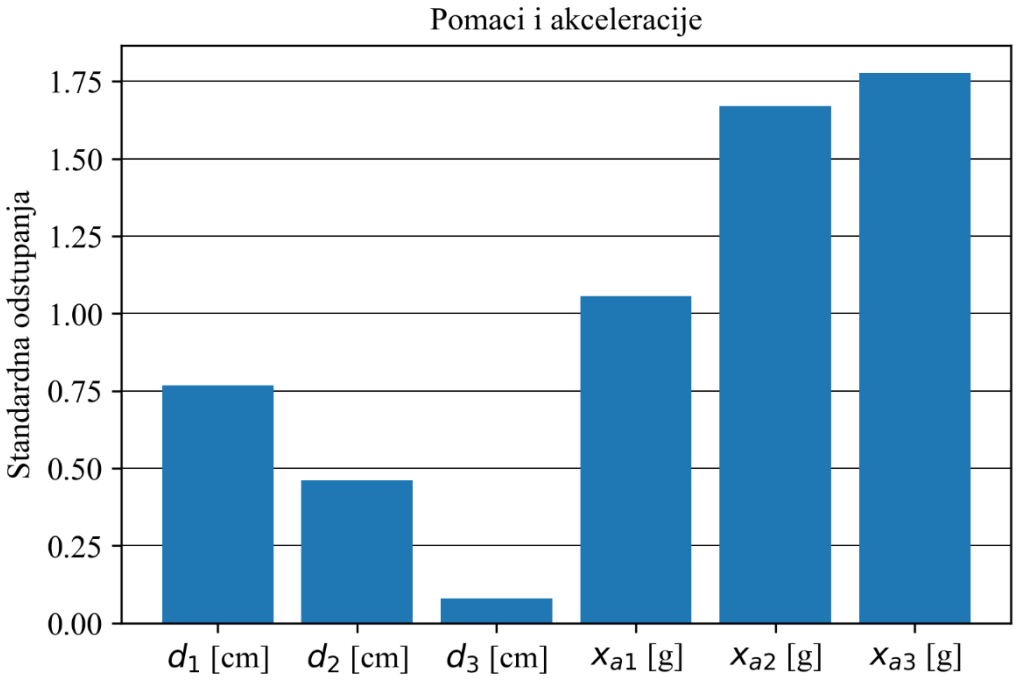
5.3. Simulacija otvorene petlje

Za simulaciju sustava potrebno je uvesti vrijednosti bijelog šuma *n* koji služi kao ulazna vrijednost i odgovarajućih vremenskih intervala *t*. Obje vrijednosti su nizovi brojeva od kojih svaki ima po 500 000 ulaznih vrijednosti odnosno intervala.

```
n = sio.loadmat('n.mat')
n = n['n']
n = np.transpose(n)

t = np.arange(1, 500002, dtype =float)
t = t/1000
```

Sustav *PF* se preoblikuje u otvorenu petlju i zajedno s bijelim šumom i intervalima uvrštava u naredbu matlab.control.lsim() koja vraća simulaciju sustava. Nadalje se iz simuliranih izlaznih vrijednosti pomaka i ubrzanja katova računaju varijance i standardne devijacije koje se zatim grafički prikazuju na slici 5.2.

```
PF_otv = mtb.ss(PF.A, PF.B[:,0], PF.C, PF.D[:,0] )
simulacija_otv = mtb.lsim(PF_otv, n, t)

d0_1 = np.var(simulacija_otv[0][:,12])
d0_2 = np.var(simulacija_otv[0][:,13])
d0_3 = np.var(simulacija_otv[0][:,14])

xa0_1 = np.var(simulacija_otv[0][:,8])
xa0_2 = np.var(simulacija_otv[0][:,9])
xa0_3 = np.var(simulacija_otv[0][:,10])

d0_1_std = np.sqrt(d0_1)
d0_2_std = np.sqrt(d0_2)
d0_3_std = np.sqrt(d0_3)

xa0_1_std = np.sqrt(xa0_1)
xa0_2_std = np.sqrt(xa0_2)
xa0_3_std = np.sqrt(xa0_3)

plt.figure()
fig, ax = plt.subplots()
ax.set_axisbelow(True)
plt.grid(axis = 'y', lw=0.5, color = 'black')
Pomaci_Akceleracije = ['d(1)','d(2)','d(3)','xa(1)','xa(2)','xa(3)']
STDEV_otv = [d0_1_std, d0_2_std, d0_3_std, xa0_1_std, xa0_2_std, xa0_3_std]
plt.bar(Pomaci_Akceleracije, STDEV_otv)
plt.title('Pomaci i akceleracije')
plt.ylabel('Standardna odstupanja')
plt.show()
```

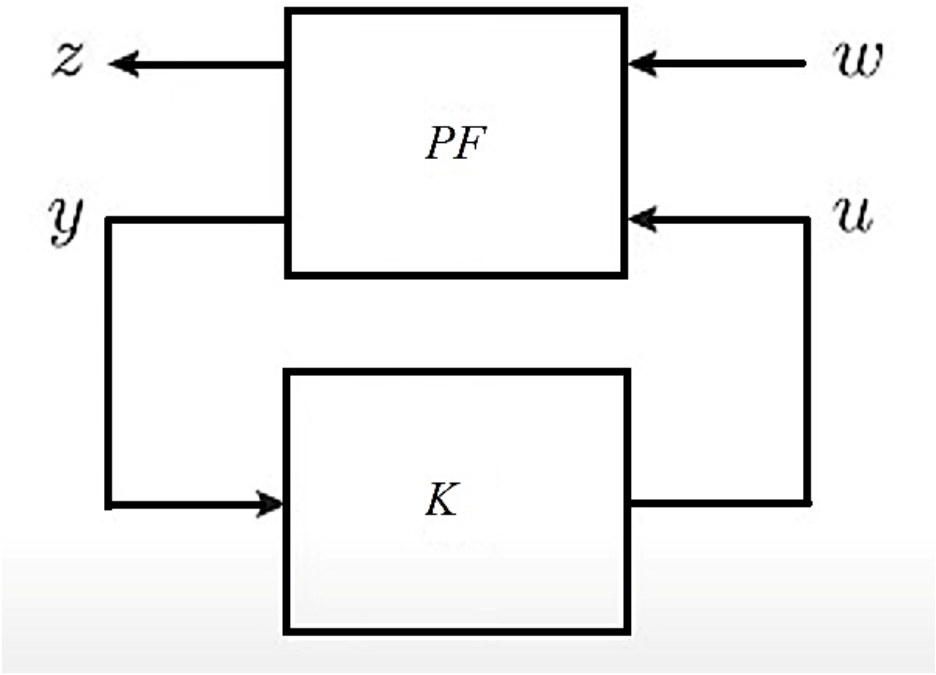


Slika 5.2. Standardna odstupanja pomaka i ubrzanja otvorene petlje

5.4. Kreiranje i spajanje kontrolera

Kontroler je zamišljen kao prostor stanja petog reda koji se spaja sa prostorom stanja kombinacije modela zgrade i *Kanai-Tajimi* filtera. Za kontroler se može odabrati i sustav nekog drugog reda, no nastoji se odabrati što je moguće niži red prostora stanja radi jednostavnijeg računanja zbog manjeg broja parametara koje je potrebno podešavati, ali da pritom nisu kompromitirani performansi kontrolera. Spoj kontrolera *K* i sustava *PF* izvodi se linearnom frakcijskom transformacijom, naredbom control.LinearIOSystem.lft(). Linearna frakcijska transformacija spaja izlaz jednog sustava sa ulazom drugoga i obrnuto (zvjezdasti produkt, zvjezdasti spoj). Shema spoja sustava prikazana je na slici 5.3., na kojoj *z* i ω predstavlja sve ostale ulazne i izlazne vrijednosti iz sustava *PF* koje nisu vezane za kontroler.

```
A = jedinka[0:25].reshape(5,5)
B = jedinka[25:45].reshape(5,4)
C = jedinka[45:50].reshape(1,5)
D = np.zeros(4)

K = mtb.ss(A, B, C, D)

T0 = control.LinearIOSystem.lft(PF, K)
```



Slika 5.3. Linearni frakcijski spoj sustava PF i kontrolera K

5.5. Inicijalizacija populacije za genetski algoritam

Populaciju u ovom slučaju predstavlja skup različitih jedinki kontrolera. Parametri kontrolera su geni jedinke. Za početak je poželjno smanjiti veličinu ulazne vrijednosti, bijelog šuma *n,* na desetinu prvotne vrijednosti, dakle sa početnih 500 000 uzoraka na 50 000, što značajno smanjuje vrijeme izvršavanja svih potrebnih simulacija na jednu desetinu vremena koja je potrebna za računanje sa cijelim ulaznim šumom. Pritom su rezultati simulacija približni onima koji se dobivaju sa cijelim ulaznim vektorom bijelog šuma.

```
size_data = 50000
n = sio.loadmat('n.mat')
n = n['n']
n = np.transpose(n)
n = n[:size_data]
t = np.arange(1, 500002, dtype =float)
t = t/1000
t = t[:size_data]
```

Problem kod inicijalizacije populacije je što nisu sve kombinacije kontrolera i sustava *PF* stabilne. Zato je potrebno osigurati stabilnost svih jedinki prije pokretanja genetskog algoritma. To se provodi posebnom manipulacijom pojedinih gena, gdje se razlikuje elemente matrice *A* prostora

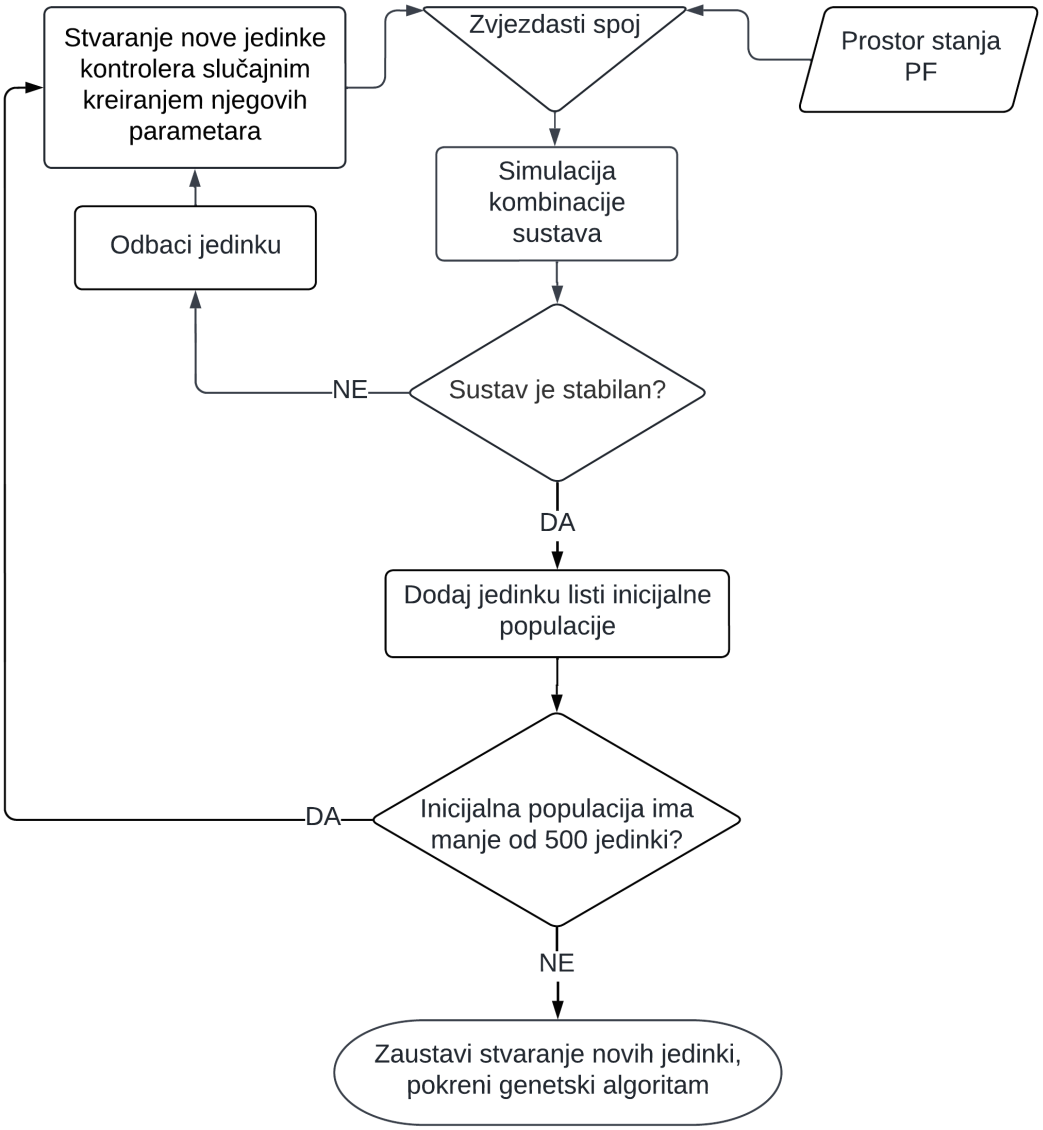
stanja *K*, te matrice *B* i *C* istog sustava. Matricu *D* se ne uzima u obzir jer su kod nje po definiciji svi elementi nula pošto nema izravne veze između kontrolera i mjerenih signala. Za gene matrice *A* odabrane su negativne vrijednosti od -10 do 0 jer ako sadrži vrijednosti izvan tog raspona dolazi do nestabilnosti sustava. Za matrice *B* i *C* vrijednosti se kreću od 0 do 10 jer bi u suprotnom često dolazilo do nestabilnosti sustava.

```
lista_jedinki = []
con_stop = 0
while len(lista_jedinki) <= 499:
    uvjet1 = {'low': -100, 'high': 0}
    uvjet2 = {'low': 0   , 'high': 10}
    geni_A = (np.random.rand(25) - 1 ) * 10
    geni_BC =np.random.rand(25) * 10
    jedinka = np.zeros((50))
    jedinka[:25] = geni_A
    jedinka[25:] = geni_BC

    A = jedinka[0:25].reshape(5,5)
    B = jedinka[25:45].reshape(5,4)
    C = jedinka[45:50].reshape(1,5)
    D = np.zeros(4)

    K = mtb.ss(A, B, C, D)
```

Zatim se i nakon kreiranja jedinke još jednom provjerava je li sustav stabilan tako da se provjerava vrijednost polova. Ako je bilo koji od 35 polova sustava pozitivan broj, to znači da je sustav nestabilan. Sustav ima 35 polova jer ima 35 jednadžbi prostora stanja. To je zbroj od 30 jednadžbi koje ima sustav *PF* i 5 jednadžbi koje ima kontroler *K*, što znači da je sustav ukupno tridesetpetog reda. Dakle ako svi polovi sustava zadovolje uvjet da su manji od nule, biti će pridodani listi jedinki, kojih je u ovom primjeru 500, te će se nakon ispunjenja liste s 500 jedinki *for* petlja zaustaviti.

```
    T0 = control.LinearIOSystem.lft(PF, K)
    pole = T0.pole()

    flag = False
    for p in pole:
        if p.real > 0:
            flag = True
```

```
            print('problem    ', con_stop)
            break
    if flag == False :
        print('Dobri geni ', con_stop)
        lista_jedinki.append(jedinka)
    con_stop = con_stop+1
    if con_stop >= 100000:
        break
```



Slika 5.4. Dijagram toka inicijalizacije populacije

Nakon što se lista jedinki popuni, od svake jedinke se kreira dinamički sustav i prolazi još jednu provjeru. Naime, metodom pokušaja i pogreške može se naslutiti da ako varijanca nekog od odziva

prelazi 100, to vrlo vjerojatno indicira nestabilnost sustava. Zato je ukoliko varijanca pomaka prvog kata prelazi 100 u programskom kodu predviđeno da se ispiše „problem“.

```
for jedinka in lista_jedinki:
    A = jedinka[0:25].reshape(5,5)
    B = jedinka[25:45].reshape(5,4)
    C = jedinka[45:50].reshape(1,5)
    D = np.zeros(4)

    CON = mtb.ss(A, B, C, D)

    T0 = control.LinearIOSystem.lft(PF, CON)
    odziv = mtb.lsim(T0, n, t)[0][:,12]

    mjera = np.var(odziv)
    print(mjera)
    if mjera >= 100:
        print('problem')
```

5.6. Definiranje ograničenja i funkcije podobnosti genetskog algoritma

Genetski algoritam biti će primijenjen korištenjem Python knjižnice *Pygad* (engl. *Python Genetic Algorithm*). Ulogu genotipa imaju elementi matrica prostora stanja kontrolera, dok fenotip predstavljaju standardne devijacije pomaka i akceleracija katova, te uloženi kontrolni napor za smanjenje vibracija.

```
import itertools as it
import pygad
```

Parametar *gene_space* označava sve dopuštene brojčane vrijednosti koje određeni geni mogu poprimiti, odnosno označava ograničenja u domeni genotipa. Tako nije dopušteno da geni matrice *A* kontrolera budu manji od -100, niti veći od 10, sve u cilju očuvanja stabilnosti sustava. To znači da genetski algoritam svojim mutacijama ne smije prijeći zadana ograničenja vrijednosti gena.

```
number = it.count()
gene_space = [{'low': -100, 'high': 0} for _ in range(25)]
gene_space = gene_space + [{'low': 0, 'high': 10} for _ in range(25)]
```

Parametar *fitness_func* označava funkciju podobnosti koju genetski algoritam nastoji maksimizirati.

```
def fitness_func(solution, solution_idx):
    print('####################')
    print('Započni raditi na ', next(number))
    # print('SOLUTION ', solution)

    # matrice kontrolera
    A = solution[0:25].reshape(5,5)
    B = solution[25:45].reshape(5,4)
    C = solution[45:50].reshape(1,5)
    D = np.zeros(4)
```

Stvara se kontroler za svaku jedinku roditelja iz svake generacije te se definira da će funkcija podobnosti biti nula ako je vrijednost polova pozitivan broj, što ukazuje na nestabilnost sustava.

```
    # izrada kontrolera
    kon = mtb.ss(A, B, C, D)

    # IZRADA SUSTAVA S KONTROLEROM
    T = control.LinearIOSystem.lft(PF, kon)
    #PROVIJERA STABILNOSTI SUSTAVA
    #POLOVI SUSTAVA
    pole = T.pole()
    for p in pole:
        if p.real > 0:
            print('problem sa polovima [nestabilnost]')
            return 0
            break
```

Za evaluaciju rješenja i uvrštavanje u funkciju podobnosti potrebno je izračunati standardna odstupanja pomaka i akceleracija svakog kata.

```
    #SIMULACIJA
    odziv = mtb.lsim(T, n, t)[0]

    #varijance pomaka i akcelracija
    d1 = np.std(odziv[:,12])
    d2 = np.std(odziv[:,13])
    d3 = np.std(odziv[:,14])
    xa1 =np.std(odziv[:,8])
```

```
xa2 =np.std(odziv[:,9])
xa3 =np.std(odziv[:,10])

xm = np.std(odziv[:,3])
xam = np.std(odziv[:,11])
u = np.std(odziv[:,15])

XM = odziv[3]
XAM = odziv[11]
U = odziv[15]
```

Ako se dogodi da je sustav nestabilan, tada će se u odzivu pojaviti oznaka *nan* te je predviđeno da funkcija podobnosti tada bude nula. Isto tako će i za beskonačne vrijednosti odziva funkcija podobnosti biti nula.

```
print('SOLUTION ',d1,'\t', d2, '\t', d3)
if np.isnan(d1) or np.isinf(d2) : # nevaljano riješenje
    return 0
```

Za jedinke koje zadovoljavaju uvjet stabilnosti funkcija podobnosti biti će inverzna vrijednost zbroja svih standardnih odstupanja pomaka i ubrzanja katova.

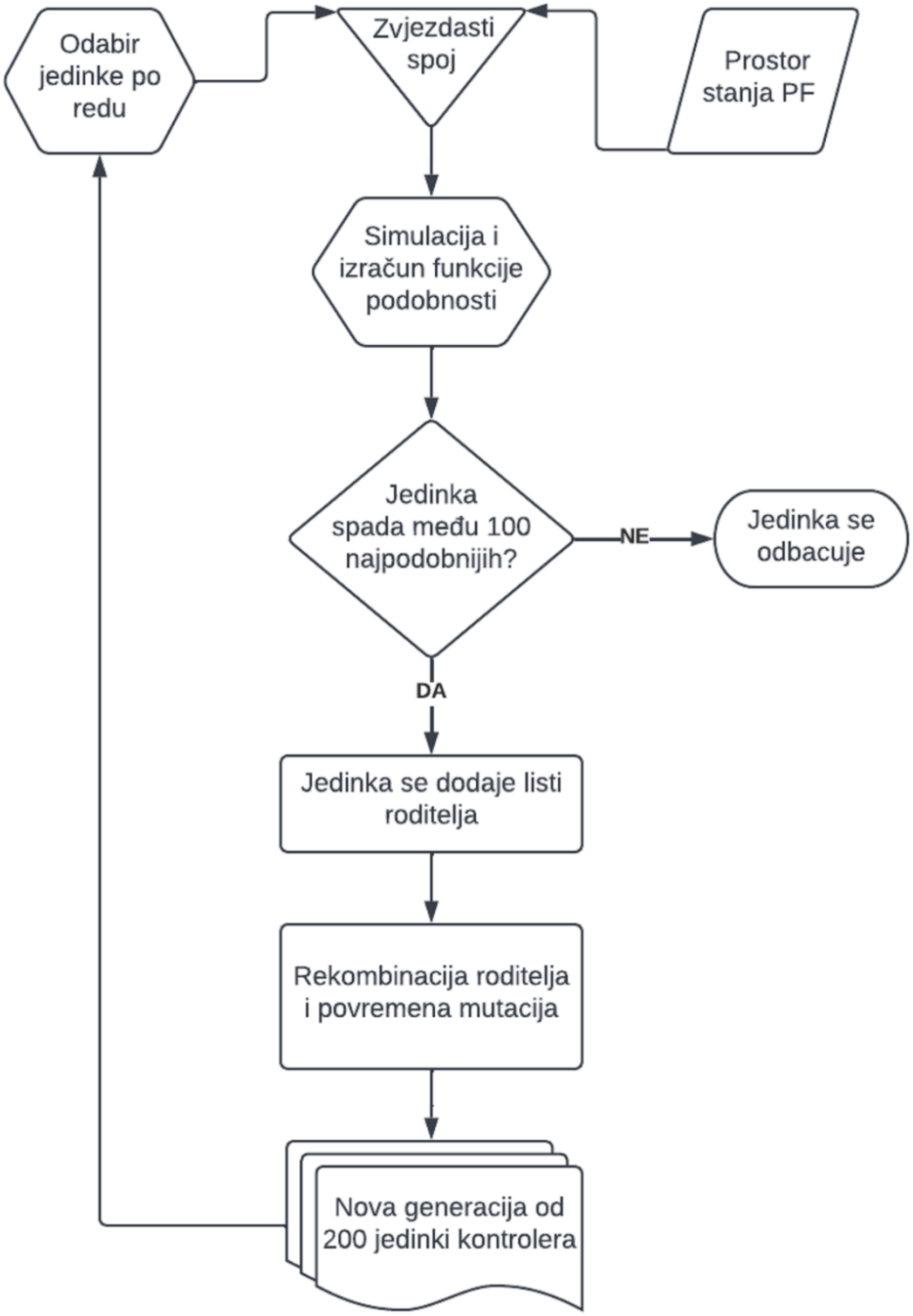
```
fitness = 1/(d1 + d2 + d3 + xa1 + xa2 + xa3)
```

Za funkciju podobnosti se još definira da će biti jednaka nuli ako vrijednosti odziva kontrolnih napora prelaze tvrda ograničenja specificirana u potpoglavlju 3.4.

```
if max(abs(XM))>9 or max(abs(XAM))>6 or max(abs(U))>3:
    fitness = 0

if xm>3 or xam>2 or u>1:
    fitness = 0

print('fitness   ', fitness)
return fitness
```

5.7. Primjena genetskog algoritma

Kako bi se uspješno primijenio genetski algoritam potrebno je podesiti parametre algoritma kako bi polučili dovoljno dobre rezultate uz što kraće trajanje izvođenja algoritma i što manji utrošak računalnih resursa. Na slici 5.5. prikazan je tijek izvođenja genetskog algoritma.



Slika 5.5. Dijagram toka rada genetskog algoritma

Parametar *sol_per_pop* označava koliko će jedinki nastati u svakoj novoj populaciji, dok *num_parent_mating* označava koliko će najpogodnijih roditelja iz prethodne generacije stvarati potomke sljedeće generacije. Broj gena je 50 zato što je broj parametara svih matrica kontrolera ukupno jednak 50.

```
num_generations = 10 #Broj generacija.
num_parents_mating = 100 #Broj roditelja za stvaranje budućih generacija.

sol_per_pop = 200 #Broj različitih rješenja u populaciji.
num_genes = 50
```

Opcija *on_generation* označava da je dozvoljeno pozivati funkciju u toku izvođenja genetskog algoritma, u ovom slučaju pozvana je samo funkcija koja ispisuje broj generacije, vrijednost njene podobnosti i postotak promjene podobnosti u odnosu na prijašnju generaciju.

```
last_fitness = 0
def on_generation(ga_instance):
    global last_fitness
    print("Generation = {generation}".format(generation=ga_instance.generations_completed))
    print("fitness    = {fitness}".format(fitness=ga_instance.best_solution(pop_fitness=ga_instance.last_generation_fitness)[1]))
    print("Change     = {change}".format(change=ga_instance.best_solution(pop_fitness=ga_instance.last_generation_fitness)[1] - last_fitness))
    last_fitness = ga_instance.best_solution(pop_fitness=ga_instance.last_generation_fitness)[1]

ga_instance = pygad.GA(num_generations=num_generations,
                       num_parents_mating=num_parents_mating,
                       sol_per_pop=sol_per_pop,
                       num_genes=num_genes,
                       fitness_func=fitness_func,
                       on_generation=on_generation,
                       gene_space=gene_space,
                       parallel_processing=1,
                       initial_population = lista_jedinki[:sol_per_pop])
```

Nakon definiranja argumenata funkcije, pokreće se genetski algoritam i daje naredba za crtanje grafa funkcije podobnosti. Također dodane su naredbe za ispisivanje najboljeg rješenja, broja

generacije u kojoj je najbolje rješenje postignuto, ispis parametara najboljeg rješenja i za prikaz trajanja računalne operacije.

```
ga_instance.run()
ga_instance.plot_fitness()


# Returning the details of the best solution.
solution, solution_fitness, solution_idx =
ga_instance.best_solution(ga_instance.last_generation_fitness)
print("Parametri najboljeg riješenja : {solution}".format(solution=solution))
print("fitness najboljeg riješenja =
{solution_fitness}".format(solution_fitness=solution_fitness))
print("Redni broj najboljeg riješenja :
{solution_idx}".format(solution_idx=solution_idx))


if ga_instance.best_solution_generation != -1:
    print("Najbolja fitness funkcija postignuta nakon
{best_solution_generation}
generacija.".format(best_solution_generation=ga_instance.best_solution_genera
tion))


end = time.time()
print('Trajanje proračuna', end-start, 's')
```

6. REZULTATI I DISKUSIJA

Za svaki broj generacija bili su zadovoljeni uvjeti strogih ograničenja u pogledu maksimalnog napona signala, ubrzanja i pomaka utega aktuatora. Rezultati su evaluirani za 5 i 50 generacija.

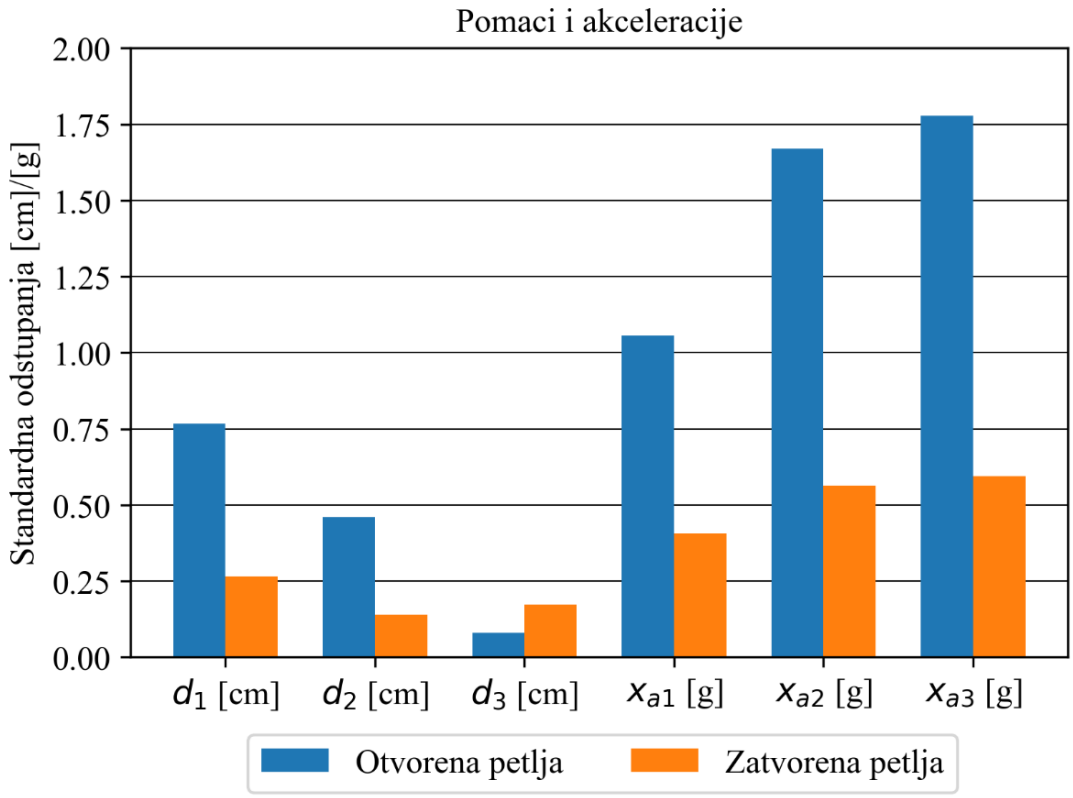
Nakon prvih pet i nakon 50 generacija ostvarena su smanjenja standardnih odstupanja pomaka i vibracija kao što je prikazano u tablici 6.1. za svaki pojedinačni pomak i akceleraciju.

Tablica 6.1. Usporedba standardnih devijacija pomaka i akceleracija između sustava bez kontrole vibracija, kontrole nakon 5 i nakon 50 generacija, crvena polja označavaju vršne pomake/ubrzanja

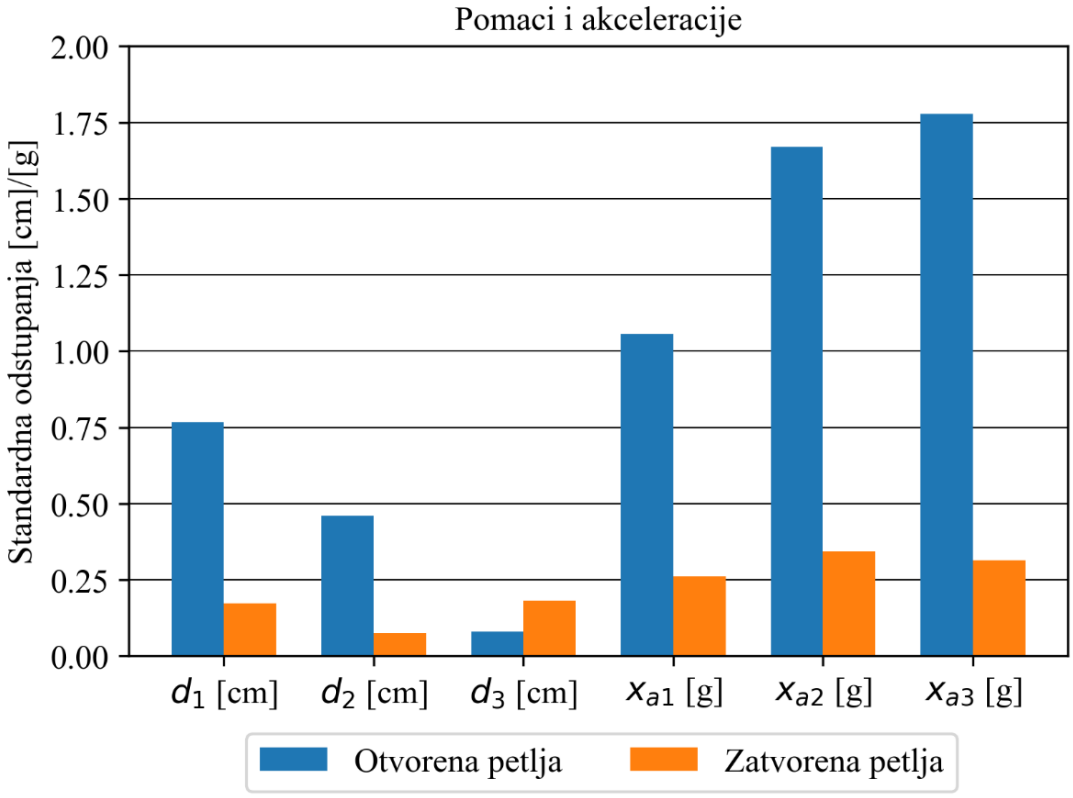
Pomaci/Ubrzanja	Bez zaštite	Nakon 5 generacija	Postotak smanjenja	Nakon 50 generacija	Postotak smanjenja
Pomak prvog kata [cm]	0,77	0,27	65,35%	0,17	77,59%
Pomak drugog kata [cm]	0,46	0,14	69,8%	0,08	83,42%
Pomak trećeg kata [cm]	0,08	0,17	-118,53%	0,18	-129,94%
Ubrzanje prvog kata [g]	1,06	0,41	61,55%	0,26	75,22%
Ubrzanje drugog kata [g]	1,67	0,56	66,25%	0,34	79,4%
Ubrzanje trećeg kata [g]	1,78	0,6	66,52%	0,31	82,37%

Genetski algoritam uspio je značajno smanjiti vršne pomake i akceleracije. Tako su već nakon pet generacija sve akceleracije i svi pomaci osim za treći kat smanjeni za više od 60%.

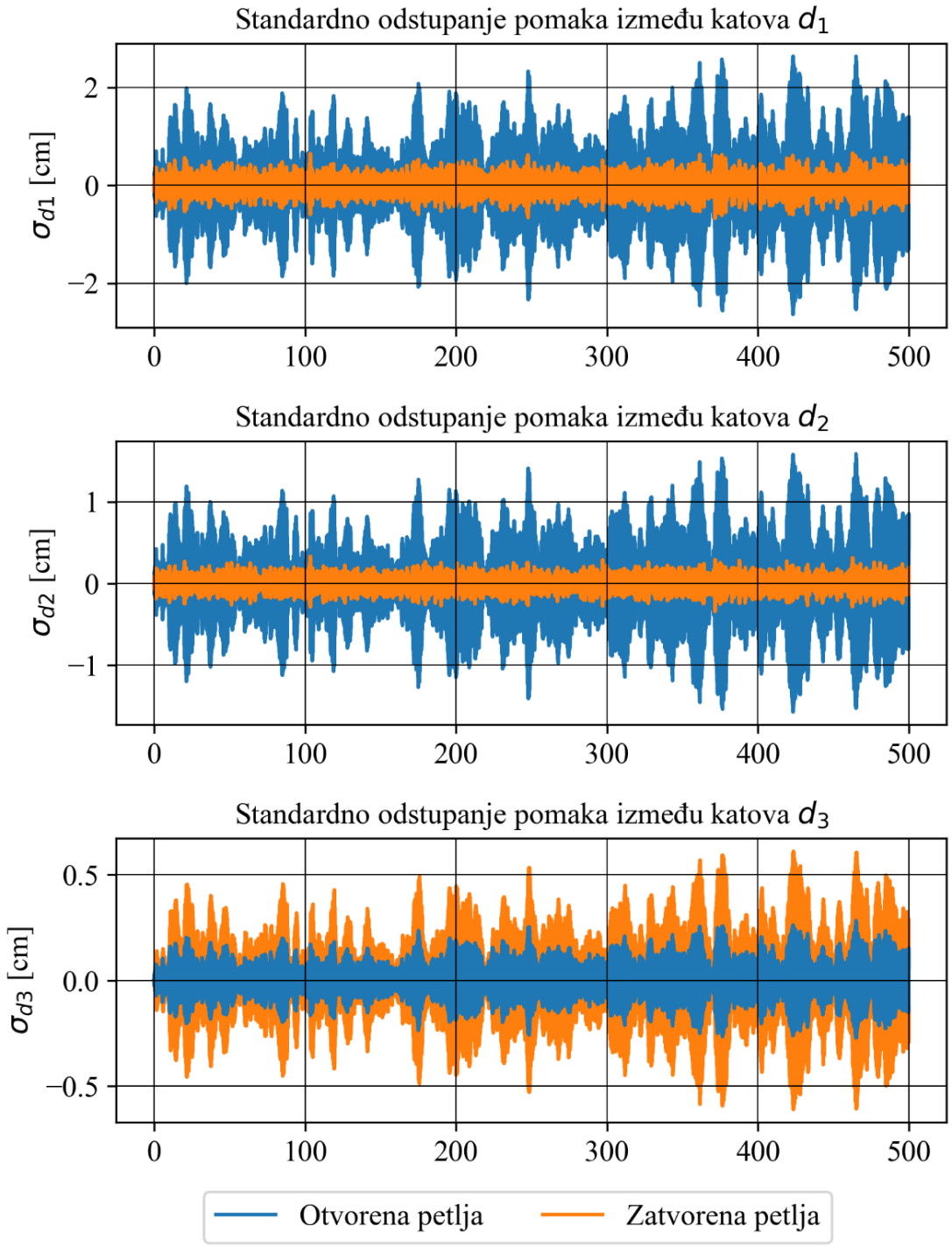
Problem je međutim u tome što se je pomak trećeg kata povećao za čak 118,53%, ali kako je to ionako bio najmanji pomak, ispada da je najveći pomak između katova znatno smanjen. Nije cilj smanjiti sve pomake i akceleracije za jednak postotak, već smanjiti vršne akceleracije što je više moguće. Tako je na primjer, zaštićenija zgrada gdje svaki kat ima ubrzanje 2 [g], nego zgrada koja na jednom katu ima ubrzanje 4 [g], a na ostalima približno nuli, jer su tada sva naprezanja koncentrirana na jednom katu, te dolazi do urušavanja tog kata što lančanom reakcijom može dovesti do urušavanja cijele zgrade. Zato je pogodnije raspodijeliti opterećenje ravnomjerno po katovima.



Slika 6.1. Postignuće kontrolnog algoritma nakon pet generacija



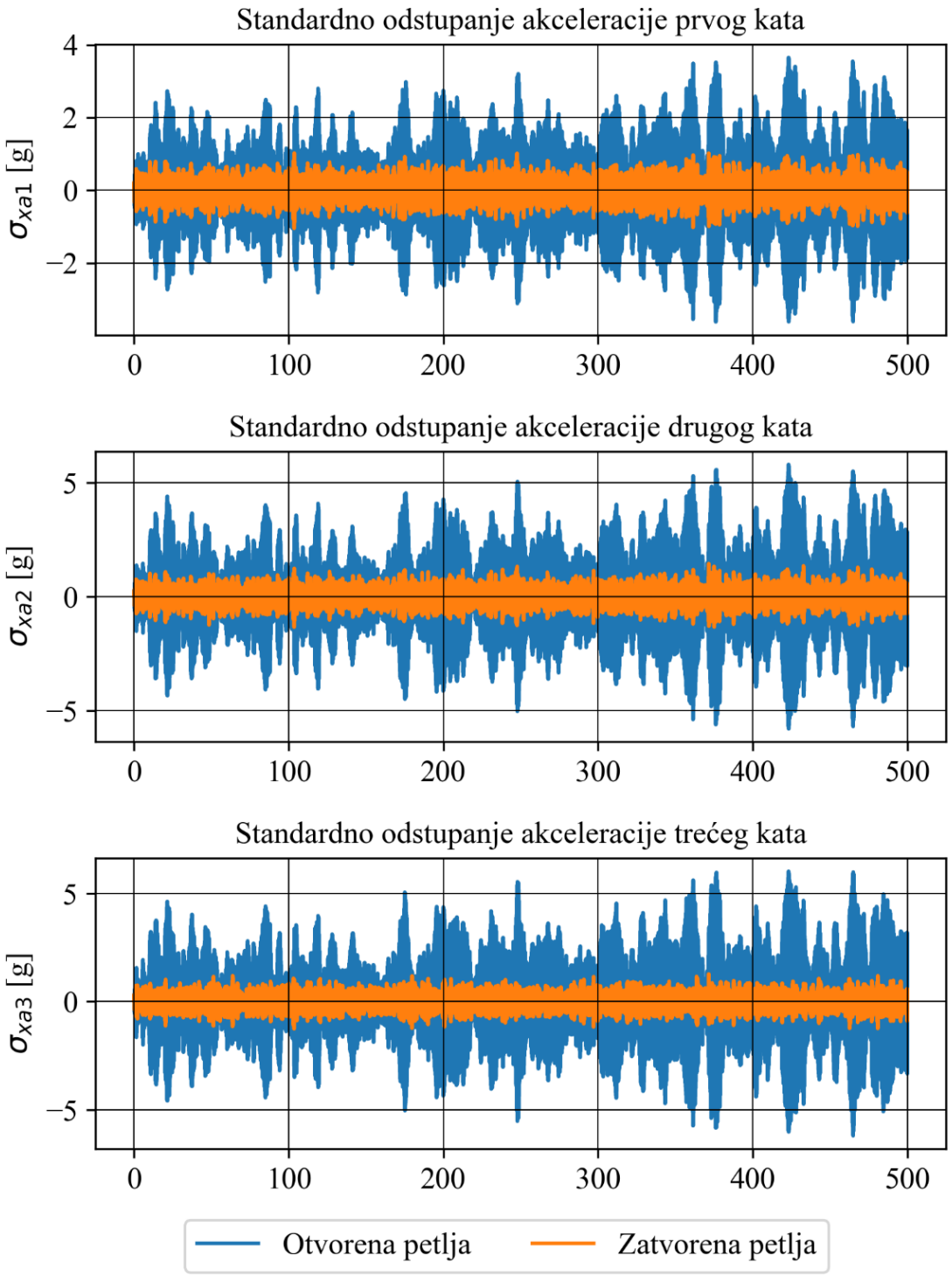
Slika 6.2. Postignuće kontrolnog algoritma nakon 50 generacija



Slika 6.3. Usporedba odziva pomaka kroz vrijeme simulacije nakon 50 generacija podešavanja i bez kontrole

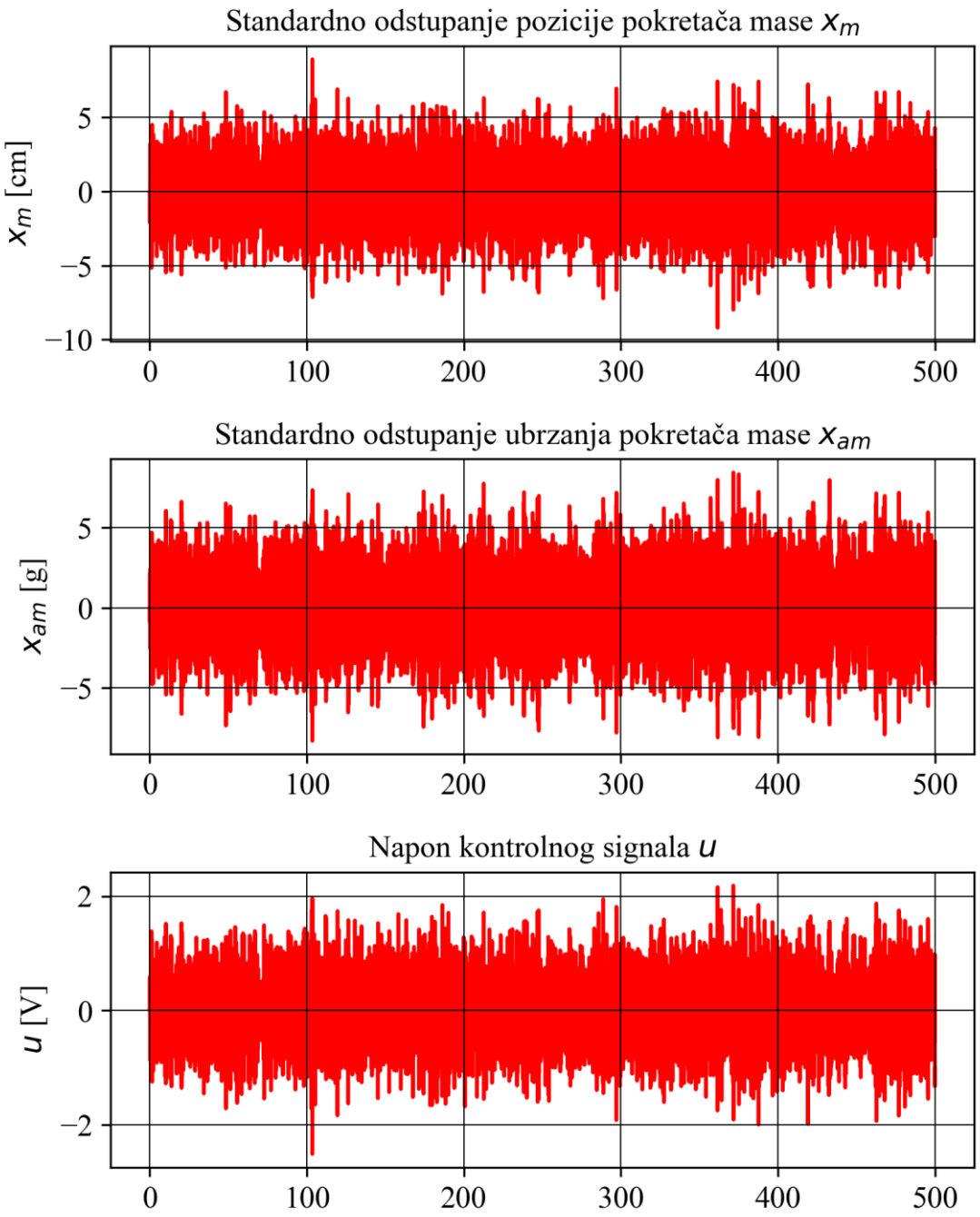
Na slikama 6.3.-6.6. prikazane su samo vrijednosti za rezultate postignute nakon 50 generacija, budući da su rezultati za pet generacija lošiji.

Na dnu slike 6.3. vidi se kako kontrolni algoritam nije uspio smanjiti pomake najvišeg kata, iako se iz slike 6.4. vidi da je za isti kat ubrzanje smanjeno.



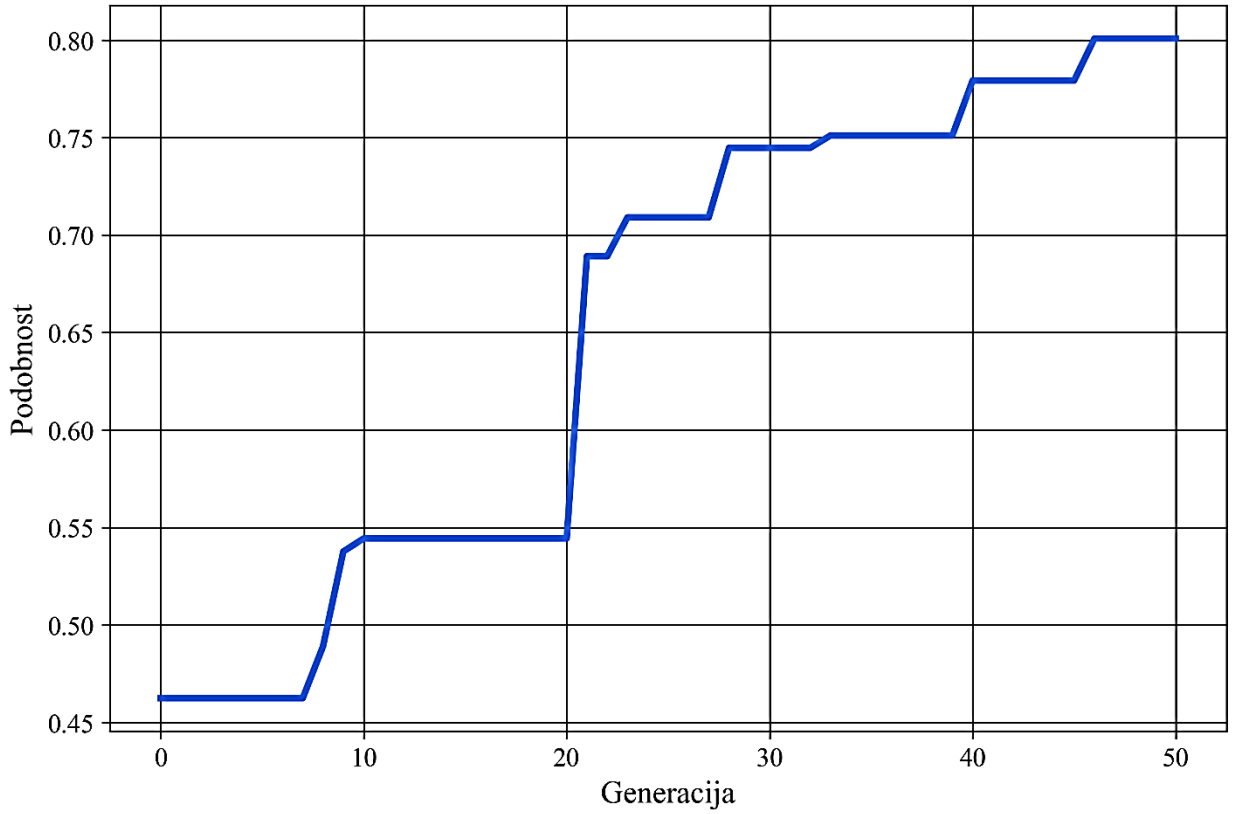
Slika 6.4. Usporedba odziva simulacije ubrzanja kroz vrijeme bez kontrole i nakon 50 generacija podešavanja

Slika 6.5. prikazuje kontrolni napor aktuatora. Poželjno je da kontrolni napori ne prelaze stroge granice kako se mehanizam ne bi oštetio. Svaki sustav također ima ograničenja u pogledu snage koju može razviti kao i fizička ograničenja određena gabaritima objekata.



Slika 6.5. Kontrolni napori sustava dobivenog nakon 50 generacija

Kao što je uobičajeno kod evolucijskih algoritama, brzina napretka ka ostvarenju cilja se drastično smanjuje kroz vrijeme. Većina smanjenja standardnih odstupanja pomaka i ubrzanja je ostvareno već u prvih 5 generacija, dok su ostale generacije pojedinačno znatno manje doprinijele smanjenju pomaka i ubrzanja, sve do trenutka kada napretka uopće više nema ili je toliko malen da nije vrijedan trošenja računalnih resursa.



Slika 6.6. Napredak u povećanju funkcije podobnosti kroz generacije

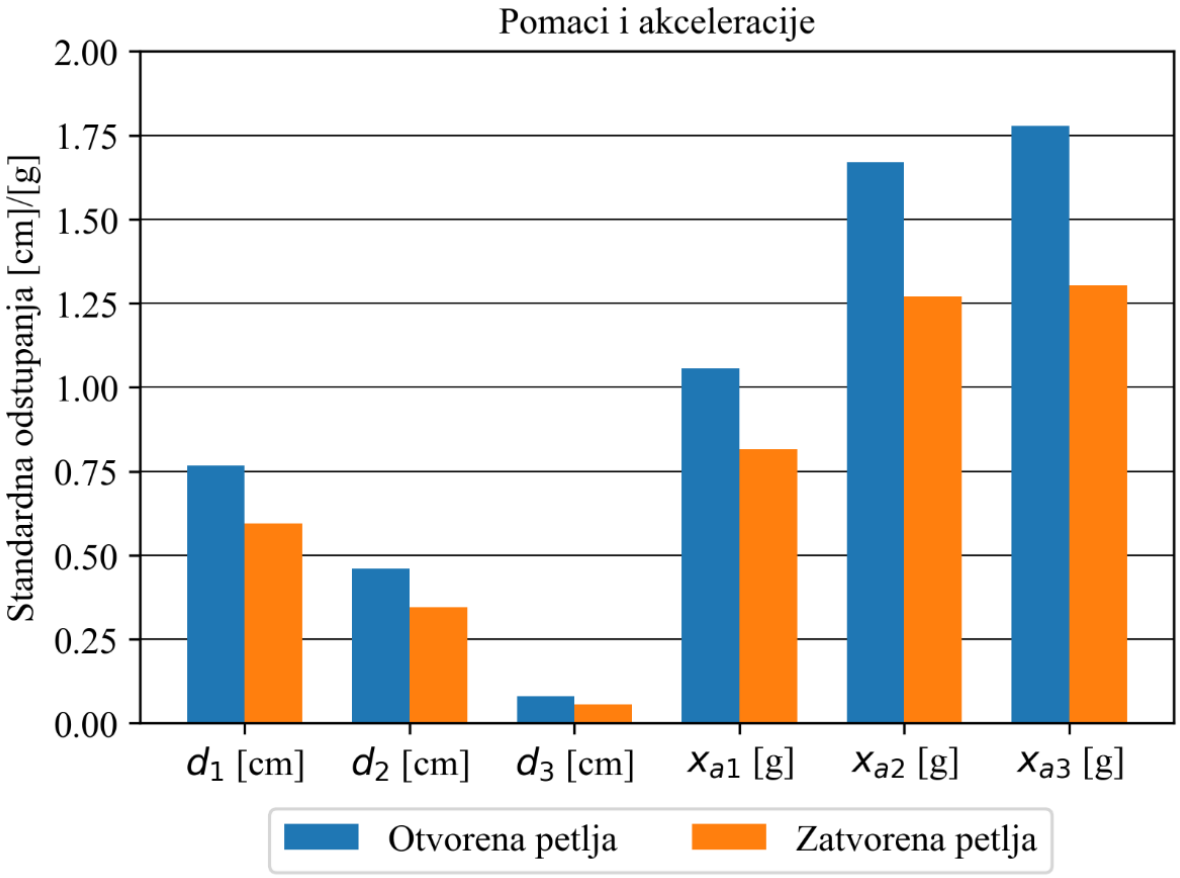
Iz slike 6.6. je uočljivo kako je algoritam počeo značajno stagnirati nakon dvadeset osme generacije. Iako se kod heurističkih metoda nikad sa sigurnošću ne može utvrditi nakon koje generacije bi se ponovno pojavio značajan skok funkcije podobnosti, razumljivo je očekivati da će od pedesete generacije vjerojatnost postizanja značajnih poboljšanja biti vrlo mala.

Trajanje proračuna s pet generacija je 4 [min] i 53,75 [s], dok je za model nakon 50 generacija bilo potrebno 12 [min] i 39,67 [s].

6.1. Mogučnosti poboljšanja kontrolnog sustava

Ukoliko se želi postići smanjenje pomaka trećeg kata, moguće je napraviti jednostavnu modifikaciju funkcije podobnosti, na način da se pomak trećeg kata u toj funkciji pomnoži koeficijentom 50, te mu se tako pridoda veća težina.

```
fitness = 1/(d1 + d2 + 50*d3 + xa1 + xa2 + xa3)
```



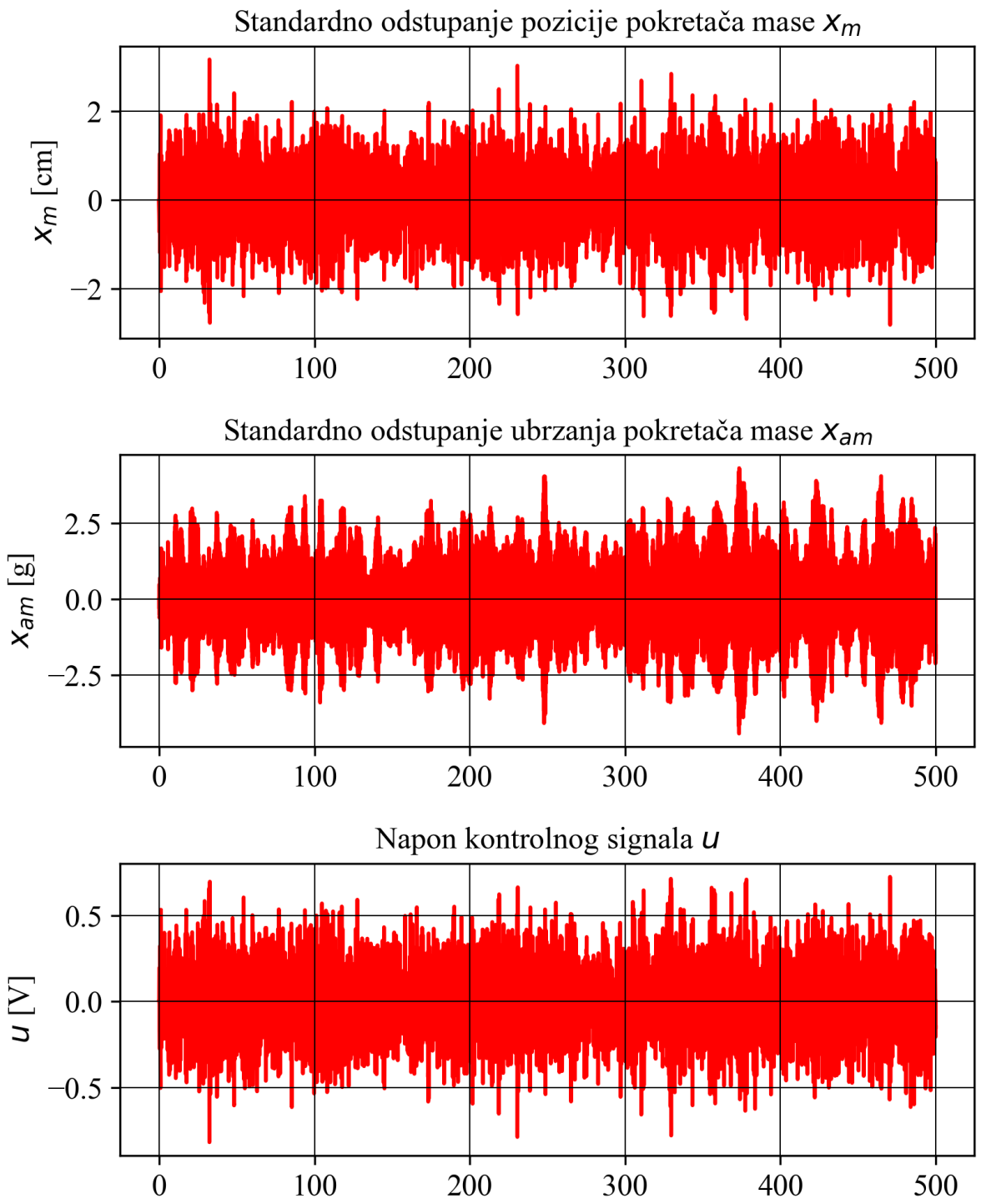
Slika 6.7. Usporedba pomaka i akceleracija otvorene i zatvorene petlje nakon povećanja koeficijenta ispred pomaka trećeg kata 50 puta, nakon 50 generacija

Još jedna korist od takvog pristupa je to što se kontrolni napor, prikazan na slici 6.8. značajno smanjuje u odnosu na kontrolni napor sustava bez te modifikacije, prikazan na slici 6.5.

Ali zato se kompromitiraju sva ostala ubrzanja i pomaci u zgradi. Smanjenja ostalih značajki padaju na manje od 30%, kao što je prikazano na slici 6.7., pa se može ustvrditi da je bolje koristiti početnu funkciju podobnosti bez ove izmjene. Jedini valjani razlog korištenja ove modifikacije je dodatno ograničenje kontrolnog napora.

Rad genetskog algoritma se neznatno može poboljšati odabirom jednolikog (uniformnog) križanja.

Vrijeme računanja bi se smanjilo kada bi se dimenzija ulaznog vektora *n* potrebnog za simulaciju potresa dodatno reducirala sa 50 000 na 5000 titraja ulaznog bijelog šuma. Ta modifikacija je isprobana ali nije dala dobre rezultate. Ulazni bijeli šum je u tom slučaju previše reduciran da bi davao rezultate istovjetne onima koji nastaju simulacijom sa cijelim ulaznim vektorom *n*.



Slika 6.8. Kontrolni napor nakon množenja pomaka trećeg kata u funkciji podobnosti sa koeficijentom 50, nakon 50 generacija

7. ZAKLJUČAK

Genetski algoritam se pokazao se kao korisna metoda za podešavanje parametara ovog mehaničkog sustava, odnosno kombinacije mehaničkog i električnog sustava ako se u obzir uzme kontrolni napor *u* koji upravlja ubrzanjem mase utega aktuatora.

U svojoj osnovnoj formi genetski algoritam značajno smanjuje vršne pomake i ubrzanja, što je vrlo pogodno za očuvanje zgrade i udobnost korisnika prostora. Vršni pomaci i akceleracije su najveća prijetnja statičkoj stabilnosti zgrade jer se puknuća uvijek javljaju na mjestima najvećih naprezanja, a najveća naprezanja se javljaju na mjestima najvećih pomaka i ubrzanja.

Nedostatak algoritma je kompromis koji se javlja između smanjenja pomaka trećeg kata i pomaka i ubrzanja svih drugih katova. Treći kat je jedini kat čiji pomak je bio veći kod kontroliranog modela u odnosu na model otvorene petlje, u slučaju da nisu uvedene nikakve modifikacije algoritma.

Još jedan nedostatak algoritma je spora konvergencija funkcije podobnosti. Tako bi se u programu *Matlab* sličan problem riješio već za nekoliko sekundi, a genetskom algoritmu je potrebno nekoliko minuta. Razlog spore konvergencije je potreba za simulacijom sustava za svaku jedinku u svakoj generaciji, što zauzima više od 90% vremena potrebnog za izvođenja algoritma.

Simulacija sustava je često vrlo spora kod digitalnih računala jer za razliku od analognih ne obrađuju kontinuirane signale pa je potrebna diskretizacija signala. Analogna računala su se dugo smatrala zastarjelom tehnologijom pa je njihov razvoj bio zapostavljen. Analogna računala uistinu jesu lošija od digitalnih kada je u pitanju pouzdanost, te se ne mogu koristiti za toliko široke primjene kao digitalno računalo, ali znatno brže izvode računalne simulacije mehaničkih i električnih sustava. Upravo iz tog razloga se u novije vrijeme ponovno razmatra razvoj analognih računala za primjene u inženjerstvu [13].

Jedna od mogućnosti je da se genetski algoritam izvodi na digitalnom računalu, a za potrebu simulacije sustava za svaku jedinku se može upotrijebiti analogno računalo.

Razvoj suvremenih analognih računala je još uvijek u povojima. Analogna računala su za sada još uvijek vrlo rijetka i skupa. Imaju potencijalnu budućnost samo kao specijalizirana računala za rješavanje diferencijalnih jednadžbi, za razliku od digitalnih računala koja su univerzalna. Zato bi kombinacije analognih i digitalnih računala u budućnosti mogle prevladati za rješavanje raznih diferencijalnih jednadžbi mehaničkih sustava poput ovoga koji je opisan u ovom radu.

LITERATURA

[1] B. F. Spencer, S. J. Dyke i H. S. Deoskar, »Benchmark problems in structural control: Part I—Active mass driver system,« *Earthquake Engineering & Structural Dynamics*, pp. 1127-1139, 1998.

[2] M. Sun, »Quora,« [Mrežno]. Available: https://www.quora.com/Why-did-Taiwan-decide-to-build-the-Taipei-101-when-Taiwan-is-prone-to-lots-of-earthquakes. [Pokušaj pristupa 4. srpnja 2022.].

[3] »Samsung Community,« Samsung Electronics, 21. 7. 2020. [Mrežno]. Available: https://r2.community.samsung.com/t5/Others/What-is-Active-Noise-Cancelation/td-p/4955008. [Pokušaj pristupa 28. 7. 2022.].

[4] »Hrvatska enciklopedija,« [Mrežno]. Available: https://www.enciklopedija.hr/natuknica.aspx?ID=40179. [Pokušaj pristupa 6. 9. 2022.].

[5] B. S. Taranath, Structural Analysis and Design of Tall Buildings: Steel and Composite Construction, Boca Raton: Taylor & Francis Group, 2012.

[6] D. Daniell, »IDEERS - University of Bristol,« University of Bristol, [Mrežno]. Available: http://www.ideers.bris.ac.uk/resistant/vibrating_build_natfreq.html. [Pokušaj pristupa 9. 9. 2022.].

[7] S. Castellaro, R. B. Raykova i M. Tsekov, »Resonance Frequencies of Soil and Buildings — Some Measurements in Sofia and Its Vicinity,« Sofia, 2016.

[8] T. Šurina, »Metoda prostora stanja,« u *Automatska regulacija*, Zagreb, Školska knjiga, 1991., pp. 362 - 398.

[9] Z. Vukić i L. Kuljača, »Klasifikacija sustava,« u *Automatsko upravljanje - analiza linearnih sustava*, Zagreb, Kigen, 2005., pp. 3 - 13.

[10] Z. Car, N. Anđelić i S. Blažević, *Predavanja iz kolegija "Evolucijska robotika"*, Rijeka, 2021.

[11] M. Golub, *Genetski algoritam - prvi i drugi dio,* Zagreb: Fakultet elektrotehnike i računarstva, 2004.

[12] MathWorks Help Center, »mathworks.com,« MathWorks, [Mrežno]. Available: https://www.mathworks.com/help/control/ug/active-vibration-control-in-three-story-building.html. [Pokušaj pristupa 23. 8. 2022.].

[13] S. Köppel, B. Ulmann, L. Heimann i D. Killat, »Using analog computers in today's largest computational challenges,« Berlin, 2021.

SAŽETAK

Ovaj rad bavi se podešavanjem parametara sustava za kontrolu vibracija u trokatnoj zgradi. Cilj rada je smanjiti pomake i akceleracije svih katova zgrade uz što manji utrošak energije i zadovoljavajući sva stroga ograničenja upravljačkog mehanizma. Kao metoda podešavanja parametara korišten je genetski algoritam, čijom primjenom je najviše smanjen pomak drugog kata, za 83,42%, dok je najveće smanjenje ubrzanja od 82,37% ostvareno na trećem katu.

Ključne riječi: genetski algoritam, redukcija vibracija, zaštita od potresa, podešavanje parametara mehaničkih sustava, upravljački sklopovi.

SUMMARY

This paper dealt with parameter adjustment of the vibration control system in a three-storey building. The goal of the work was to reduce the displacements and accelerations of all floors of the building with as little energy consumption as possible while satisfying the constraints of the control mechanism. A genetic algorithm was used as a parameter adjustment method, with the application of which the displacement of the second floor was reduced the most, by 83.42%, and peak acceleration reduction was that of the third floor, which was reduced by 82.37%.

Keywords: genetic algorithm, vibration reduction, earthquake protection, parameter adjustment of mechanical systems, control circuits.

POPIS SLIKA

POPIS TABLICA

DODATAK A - Python kod za podešavanje parametara upravljačkog sustava i prikaz rezultata rješenja

```
import math
import scipy.io as sio
import numpy as np
import matplotlib.pyplot as plt
import control
from control import matlab as mtb
import pandas as pd
import time

zg = 0.3
wg = 37.3
S0 = 0.03*zg/(math.pi*wg*(4*zg**2+1))
Numerator = [math.sqrt(S0)*2*zg*wg, math.sqrt(S0)*wg**2]
Denominator = [1,  2*zg*wg,  wg**2]

F = math.sqrt(2*math.pi)*mtb.tf(Numerator, Denominator)
plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
mtb.bode(F)
plt.xlabel('Frekvencija (rad/sec)')
plt.ylabel('Fazni pomaci (°)')
#plt.savefig('bode_plot', dpi=300, bbox_inches='tight')
plt.show()


F1 = control.append(F,1)

PA = 'A.xlsx'
PB = 'B.xlsx'
PC = 'C.xlsx'
PD = 'D.xlsx'

PA = np.array(pd.read_excel(PA, header = None ))
PB = np.array(pd.read_excel(PB, header = None ))
```

```
PC = np.array(pd.read_excel(PC, header = None ))
PD = np.array(pd.read_excel(PD, header = None ))

P = mtb.ss(PA, PB, PC, PD)
PF = P*F1

n = sio.loadmat('n.mat')
n = n['n']
n = np.transpose(n)

t = np.arange(1, 500002, dtype =float)
t = t/1000

PF_otv = mtb.ss(PF.A, PF.B[:,0], PF.C, PF.D[:,0] )
simulacija_otv = mtb.lsim(PF_otv, n, t)

d0_1 = np.var(simulacija_otv[0][:,12])
d0_2 = np.var(simulacija_otv[0][:,13])
d0_3 = np.var(simulacija_otv[0][:,14])

xa0_1 = np.var(simulacija_otv[0][:,8])
xa0_2 = np.var(simulacija_otv[0][:,9])
xa0_3 = np.var(simulacija_otv[0][:,10])

d0_1_std = np.sqrt(d0_1)
d0_2_std = np.sqrt(d0_2)
d0_3_std = np.sqrt(d0_3)

xa0_1_std = np.sqrt(xa0_1)
xa0_2_std = np.sqrt(xa0_2)
xa0_3_std = np.sqrt(xa0_3)

plt.rcParams["font.family"] = "Times New Roman"

plt.rc('font', size=12)          # controls default text sizes
plt.rc('axes', titlesize=12)     # fontsize of the axes title
plt.rc('axes', labelsize=12)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=12)    # fontsize of the tick labels
plt.rc('ytick', labelsize=12)    # fontsize of the tick labels
plt.rc('legend', fontsize=12)    # legend fontsize
plt.rc('figure', titlesize=12)  # fontsize of the figure title
```

```
plt.figure()
fig, ax = plt.subplots()
ax.set_axisbelow(True)

plt.grid(axis = 'y', lw=0.5, color = 'black')
Pomaci_Akceleracije = ['d(1)','d(2)','d(3)','xa(1)','xa(2)','xa(3)']
STDEV_otv = [d0_1_std, d0_2_std, d0_3_std, xa0_1_std, xa0_2_std, xa0_3_std]
plt.bar(Pomaci_Akceleracije, STDEV_otv)
plt.title('Pomaci i akceleracije')
plt.ylabel('Standardna odstupanja')
#plt.savefig('Otvorena', dpi=300, bbox_inches='tight')
plt.show()

#%%
# količina podataka je smanjenja pa radi brže
size_data = 50_000
n = sio.loadmat('n.mat')
n = n['n']
n = np.transpose(n)
n = n[:size_data]
t = np.arange(1, 500002, dtype =float)
t = t/1000
t = t[:size_data]

"""
INICIJALIZCAIJA POPULACIJE

Kod kreiranja jedinke - postoji problem da je riješenje nestabilno
"""
start = time.time()
lista_jedinki = []
con_stop = 0
while len(lista_jedinki) <= 499:
    uvjet1 = {'low': -100, 'high': 0}
    uvjet2 = {'low': 0   , 'high': 10}
    geni_A = (np.random.rand(25) - 1 ) * 10
    geni_BC =np.random.rand(25) * 10
    jedinka = np.zeros((50))
    jedinka[:25] = geni_A
    jedinka[25:] = geni_BC
```

```
    # jedinka = np.array([np.random.rand() for i in range(50)])

    A = jedinka[0:25].reshape(5,5)
    B = jedinka[25:45].reshape(5,4)
    C = jedinka[45:50].reshape(1,5)
    D = np.zeros(4)

    K = mtb.ss(A, B, C, D)

    T0 = control.LinearIOSystem.lft(PF, K)
    pole = T0.pole()

    flag = False
    for p in pole:
        if p.real > 0:
            flag = True
            print('problem   ', con_stop)
            break
    if flag == False :
        print('Good genes ', con_stop)
        lista_jedinki.append(jedinka)

    con_stop = con_stop+1
    if con_stop >= 100000:
        break

for jedinka in lista_jedinki:
    A = jedinka[0:25].reshape(5,5)
    B = jedinka[25:45].reshape(5,4)
    C = jedinka[45:50].reshape(1,5)
    D = np.zeros(4)

    CON = mtb.ss(A, B, C, D)

    T0 = control.LinearIOSystem.lft(PF, CON)
    odziv = mtb.lsim(T0, n, t)[0][:,12]

    mjera = np.var(odziv)
    print(mjera)
    if mjera >= 100:
```

```
        print('problem')


#%%
"""
GENETSKI ALGORITAM
"""
import itertools as it
import pygad

number = it.count()
gene_space = [{'low': -100, 'high': 0} for _ in range(25)]
gene_space = gene_space + [{'low': 0, 'high': 10} for _ in range(25)]

def fitness_func(solution, solution_idx):
    print('######################')
    print('Započni raditi na ', next(number))
    # print('SOLUTION ', solution)

    # matreice kontrolera
    A = solution[0:25].reshape(5,5)
    B = solution[25:45].reshape(5,4)
    C = solution[45:50].reshape(1,5)
    D = np.zeros(4)

    # izrada kontrolera

    kon = mtb.ss(A, B, C, D)

    # IZRADA SUSTAVA S KONTROLEROM
    T = control.LinearIOSystem.lft(PF, kon)
    #PROVIJERA STABILNOSTI SUSTAVA
    #POLOVI SUSTAVA
    pole = T.pole()
    for p in pole:
        if p.real > 0:
            print('problem sa polovima [nestabilnost]')
            return 0
            break

    #SIMULACIJA
```

```
    odziv = mtb.lsim(T, n, t)[0]

    #varijance pomaka i akcelracija
    d1 = np.std(odziv[:,12])
    d2 = np.std(odziv[:,13])
    d3 = np.std(odziv[:,14])
    xa1 =np.std(odziv[:,8])
    xa2 =np.std(odziv[:,9])
    xa3 =np.std(odziv[:,10])

    xm = np.std(odziv[:,3])
    xam = np.std(odziv[:,11])
    u = np.std(odziv[:,15])

    XM = odziv[3]
    XAM = odziv[11]
    U = odziv[15]


    print('SOLUTION ',d1,'\t', d2, '\t', d3)
    if np.isnan(d1) or np.isinf(d2) : # nevaljano riješenje
        return 0

    fitness = 1/(d1 + d2 + d3 + xa1 + xa2 + xa3)

    if max(abs(XM))>9 or max(abs(XAM))>6 or max(abs(U))>3:
        fitness = 0

    if xm>3 or xam>2 or u>1:
        fitness = 0

    print('fitness   ', fitness)
    return fitness


num_generations = 10 #Broj generacija.
num_parents_mating = 100 #Broj roditelja za stvaranje budućih generacija.

sol_per_pop = 200 #Broj različitih rješenja u populaciji.
num_genes = 50
```

```
last_fitness = 0
def on_generation(ga_instance):
    global last_fitness
    print("Generation = {generation}".format(generation=ga_instance.generations_completed))
    print("fitness    = {fitness}".format(fitness=ga_instance.best_solution(pop_fitness=ga_instance.last_generation_fitness)[1]))
    print("Change     = {change}".format(change=ga_instance.best_solution(pop_fitness=ga_instance.last_generation_fitness)[1] - last_fitness))
    last_fitness = ga_instance.best_solution(pop_fitness=ga_instance.last_generation_fitness)[1]


ga_instance = pygad.GA(num_generations=num_generations,
                       num_parents_mating=num_parents_mating,
                       sol_per_pop=sol_per_pop,
                       num_genes=num_genes,
                       fitness_func=fitness_func,
                       on_generation=on_generation,
                       gene_space=gene_space,
                       parallel_processing=1,
                       initial_population = lista_jedinki[:sol_per_pop])



# Running the GA to optimize the parameters of the function.
ga_instance.run()


ga_instance.plot_fitness()


# Returning the details of the best solution.
solution, solution_fitness, solution_idx = ga_instance.best_solution(ga_instance.last_generation_fitness)
print("Parametri najboljeg riješenja : {solution}".format(solution=solution))
print("fitness najboljeg riješenja = {solution_fitness}".format(solution_fitness=solution_fitness))
print("Redni broj najboljeg riješenja : {solution_idx}".format(solution_idx=solution_idx))



if ga_instance.best_solution_generation != -1:
    print("Najbolja fitness funkcija postignuta nakon {best_solution_generation} generacija.".format(best_solution_generation=ga_instance.best_solution_generation))
```

```
end = time.time()
print('Trajanje proračuna', end-start, 's')
#%%
"""
RIJEŠENJE
"""


A = solution[0:25].reshape(5,5)
B = solution[25:45].reshape(5,4)
C = solution[45:50].reshape(1,5)
D = np.zeros(4)


# izrada kontrolera
kon = mtb.ss(A, B, C, D)


T = control.LinearIOSystem.lft(PF, kon)

n = sio.loadmat('n.mat')
n = n['n']
n = np.transpose(n)

t = np.arange(1, 500002, dtype =float)
t = t/1000

Odziv_zat = mtb.lsim(T, n, t)[0]

d_1 = np.std(Odziv_zat[:,12])
d_2 = np.std(Odziv_zat[:,13])
d_3 = np.std(Odziv_zat[:,14])

xa_1 = np.std(Odziv_zat[:,8])
xa_2 = np.std(Odziv_zat[:,9])
xa_3 = np.std(Odziv_zat[:,10])

#koeficijenti smanjenja standardnih devijacija pomaka i akceleracija
KS_d1 = d_1/d0_1_std
KS_d2 = d_2/d0_2_std
KS_d3 = d_3/d0_3_std
KS_xa1 = xa_1/xa0_1_std
KS_xa2 = xa_2/xa0_2_std
KS_xa3 = xa_3/xa0_3_std

print('Standardna devijacija pomaka prvog kata bez kontrole
vibracija',d0_1_std)
print('Standardna devijacija pomaka prvog kata nakon kontrole vibracija',d_1)
print('Pomak prvog kata smanjen je za', 100*(1-KS_d1),'%')

print('Standardna devijacija pomaka drugog kata bez kontrole
vibracija',d0_2_std)
print('Standardna devijacija pomaka drugog kata nakon kontrole
vibracija',d_2)
print('Pomak drugog kata smanjen je za', 100*(1-KS_d2),'%')
```

```
print('Standardna devijacija pomaka trećeg kata bez kontrole
vibracija',d0_3_std)
print('Standardna devijacija pomaka trećeg kata nakon kontrole
vibracija',d_3)
print('Pomak trećeg kata smanjen je za', 100*(1-KS_d3),'%')

print('Standardna devijacija ubrzanja prvog kata bez kontrole
vibracija',xa0_1_std)
print('Standardna devijacija ubrzanja prvog kata nakon kontrole
vibracija',xa_1)
print('Ubrzanje prvog kata smanjeno je za', 100*(1-KS_xa1),'%')

print('Standardna devijacija ubrzanja drugog kata bez kontrole
vibracija',xa0_2_std)
print('Standardna devijacija ubrzanja drugog kata nakon kontrole
vibracija',xa_2)
print('Ubrzanje drugog kata smanjeno je za', 100*(1-KS_xa2),'%')

print('Standardna devijacija ubrzanja trećeg kata bez kontrole
vibracija',xa0_3_std)
print('Standardna devijacija ubrzanja trećeg kata nakon kontrole
vibracija',xa_3)
print('Ubrzanje trećeg kata smanjeno je za', 100*(1-KS_xa3),'%')

plt.figure()
STDEV_zat = [d_1, d_2, d_3, xa_1, xa_2, xa_3]

x = np.arange(len(Pomaci_Akceleracije))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, STDEV_otv, width, label='Otvorena petlja')
rects2 = ax.bar(x + width/2, STDEV_zat, width, label='Zatvorena petlja')
plt.grid(axis = 'y')
ax.set_axisbelow(True)
ax.yaxis.grid(color='black', lw = 0.5)
ax.set_ylabel('Standardna odstupanja [cm]/[g]')
ax.set_title('Pomaci i akceleracije')
ax.set_xticks(x, ['$d_1$ [cm]', '$d_2$ [cm]', '$d_3$ [cm]', '$x_{a1}$ [g]',
'$x_{a2}$ [g]', '$x_{a3}$ [g]'])
ax.legend(ncol = 2, bbox_to_anchor=(0.9, -0.1))
plt.ylim([0,2])
plt.savefig('Zatvorena_d3x50', dpi=300, bbox_inches='tight')
plt.show()

plt.rcParams["figure.figsize"] = (6,8)

fig, ax = plt.subplots(nrows=3)
fig.subplots_adjust(hspace=0.4)
ax[0].plot(t, simulacija_otv[0][:,12], label='Otvorena petlja', zorder=0)
ax[0].plot(t, Odziv_zat[:,12], label='Zatvorena petlja', zorder=0)
ax[0].set_title(r'Standardno odstupanje pomaka između katova $d_1$')
ax[0].set_ylabel(r'$\sigma_{d1}$ [cm]')
ax[0].grid(which = 'both', lw=0.5, c='black', zorder=10)

ax[1].plot(t, simulacija_otv[0][:,13], zorder=0)
ax[1].plot(t, Odziv_zat[:,13], zorder=0)
ax[1].set_title(r'Standardno odstupanje pomaka između katova $d_2$')
ax[1].set_ylabel(r'$\sigma_{d2}$ [cm]')
ax[1].grid(which = 'both', lw=0.5, c='black', zorder=10)

ax[2].plot(t, simulacija_otv[0][:,14], zorder=0)
```

```
ax[2].plot(t, Odziv_zat[:,14], zorder = 0)
ax[2].set_title(r'Standardno odstupanje pomaka između katova $d_3$')
ax[2].set_ylabel(r'$\sigma_{d3}$ [cm]')
ax[2].grid(which = 'both', lw=0.5, c='black', zorder=10)

fig.legend(ncol = 2, loc = 'lower center', bbox_to_anchor=(0.5, 0.03))
plt.savefig('Titraji_pomaka_d3x50', dpi=300, bbox_inches='tight')
plt.show()


fig, ax = plt.subplots(nrows=3)
fig.subplots_adjust(hspace=0.4)
ax[0].plot(t, simulacija_otv[0][:,8], label='Otvorena petlja', zorder=0)
ax[0].plot(t, Odziv_zat[:,8], label='Zatvorena petlja', zorder=0)
ax[0].set_title('Standardno odstupanje akceleracije prvog kata')
ax[0].set_ylabel(r'$\sigma_{xa1}$ [g]')
ax[0].grid(which = 'both', lw=0.5, c='black', zorder=10)

ax[1].plot(t, simulacija_otv[0][:,9], zorder=0)
ax[1].plot(t, Odziv_zat[:,9], zorder=0)
ax[1].set_title('Standardno odstupanje akceleracije drugog kata')
ax[1].set_ylabel(r'$\sigma_{xa2}$ [g]')
ax[1].grid(which = 'both', lw=0.5, c='black', zorder=10)

ax[2].plot(t, simulacija_otv[0][:,10], zorder=0)
ax[2].plot(t, Odziv_zat[:,10], zorder=0)
ax[2].set_title('Standardno odstupanje akceleracije trećeg kata')
ax[2].set_ylabel(r'$\sigma_{xa3}$ [g]')
ax[2].grid(which = 'both', lw=0.5, c='black', zorder=10)

fig.legend(ncol = 2, loc = 'lower center', bbox_to_anchor=(0.5, 0.03))
plt.savefig('Titraji_akceleracija_d3x50', dpi=300, bbox_inches='tight')
plt.show()


fig, ax = plt.subplots(nrows=3)
fig.subplots_adjust(hspace=0.4)

ax[0].plot(t, Odziv_zat[:,3], c='r', zorder=0)
ax[0].set_title('Standardno odstupanje pozicije pokretača mase $x_m$')
ax[0].set_ylabel('$x_m$ [cm]')
ax[0].grid(which = 'both', lw=0.5, c='black', zorder=10)

ax[1].plot(t, Odziv_zat[:,11], c='r', zorder=0)
ax[1].set_title('Standardno odstupanje ubrzanja pokretača mase $x_{am}$')
ax[1].set_ylabel('$x_{am}$ [g]')
ax[1].grid(which = 'both', lw=0.5, c='black', zorder=10)

ax[2].plot(t, Odziv_zat[:,15], c='r', zorder=0)
ax[2].set_title('Napon kontrolnog signala $u$')
ax[2].set_ylabel('$u$ [V]')
ax[2].grid(which = 'both', lw=0.5, c='black', zorder=10)


plt.savefig('Kontrolni napor_d3x50', dpi=300, bbox_inches='tight')

plt.show()
```