Prirodom inspirirani algoritmi u optimiranju topologije rešetkastih konstrukcija

Rašperić, Borna

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Borna Rašperić

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Dr. sc. Petar Ćurković, dipl. ing.

Student:

Borna Rašperić

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svom ocu jer me pogurao na fakultet i što sad imam osim šireg uvida u samo strojarstvo i nove spoznaje o svijetu i mogućnostima današnjeg vremena.

Borna Rašperić

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-14/22-6/1	
Ur. broj: 15-1703-22-	

DIPLOMSKI ZADATAK

Student: **BORNA RAŠPERIĆ** Mat. br.: 0035210797

Naslov rada na hrvatskom jeziku: **Prirodom inspirirani algoritmi u optimiranju topologije rešetkastih konstrukcija**

Naslov rada na engleskom jeziku: **Bioinspired algorithms in topology optimization of lattice structures**

Opis zadatka:

Prirodom inspirirani algoritmi koriste se uspješno za pronalazak kvalitetnih rješenja teških optimizacijskih problema. Posebno se ističu populacijski algoritmi koji prirodno pronalaze Pareto frontu kod problema sa suprotstavljenim kriterijima optimiranja poput primjerice istodobnog smanjenja mase i podatljivosti.

U ovom radu potrebno je ispitati mogućnost primjene ovakvih algoritama, s težištem na evolucijskom algoritmu, kojega je potrebno primijeniti za optimiranje topologije rešetkastog nosača s ciljem smanjenja progiba u zadanoj točki. Pri tome položaji čvorova predstavljaju projektne varijable i cilj je algoritma pronaći takav raspored čvorova koji smanjuje podatljivost, a istodobno slijedi zadano ograničenje ukupne mase.

U radu je potrebno napravit sljedeće:

- upoznati se s teorijom prirodom inspiriranih algoritama, s posebnim naglaskom na evolucijskom algoritmu
- u programskom jeziku Python implementirati metodu konačnih elemenata za rešetkaste konstrukcije
- u programskom jeziku Python implementirati evolucijski algoritam kojim se prema zadanim kriterijima optimiraju rešetkaste konstrukcije

U radu je potrebno navesti literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
5. svibnja 2022.	7. srpnja 2022.	18. srpnja do 22. srpnja 2022.

Zadatak zadao:
izv. prof. dr. sc. Petar Ćurković

Predsjednica Povjerenstva:
prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
cos		kosinus
E	Gpa	Youngov modul elastičnosti
F	N	Sila
G	Gpa	Modul smičnosti
h	mm	Horizontalna koordinata
J	$Kgmm^2$	Moment inercije
k	N/mm	Konstanta opruge
K_G		Globalna matrica krutosti
K_L		Lokalna matrica krutosti
K_p		Primarna matrica krutosti
K_S		Matrica krutosti strukture
M	Nmm	Moment sile
sin		sinus
u	mm	Pomak točke
x	mm	Produljenje
v	mm	Vertikalna koordinata
ε	mm/mm	Relativno produljenje
σ	N/mm^2	Naprezanje

SAŽETAK

Tema ovog diplomskog rada je izrada evolucijskog algoritma za optimiranje topologije rešetkastih konstrukcija. Fitnes funkciju evolucijskog algoritma predstavlja samostalno napisani algoritam metode konačnih elemenata sa kojim se proračunava krutost, odnosno deformacija konstrukcije u svakoj generaciji evolucijskog algoritma. Algoritam metode konačnih elemenata i evolucijski algoritam izrađeni su u python programskom jeziku, dok je sama konstrukcija dizajnirana u Blender open source software-u.

Ključne riječi: evolucijski algoritam, metoda konačnih elemenata, Python, Blender

SUMMARY

The topic of this thesis is creation of an evolutionary algorithm for optimizing the topology of lattice structures. The fitness function of the evolutionary algorithm is represented by an independently written algorithm of the finite element method, which is used to calculate stiffness, deformation of the structure in each generation of the evolutionary algorithm. The algorithm of the finite element method and the evolutionary algorithm were created in the Python programming language, while the construction itself was designed in Blender open source software.

Key words: evolutionary algorithm, finite element method, Blender

1. UVOD

Od pamtivijeka inženjeri pokušavaju izraditi više, veće, čvršće konstrukcije koje se trebaju opirati ljudskoj destruktivnoj ruci, elementarnim nepogodama, potresima, a na koncu i zubu vremena. Inženjeri su kroz tisuće godina najosnovnijim načinom evolucije razmišljanja i znanja uspjeli izgraditi i izraditi nebrojene inženjerske pothvate diljem svijeta, a da pritom nisu imali napredna računala i inženjerski software. Ti prvi inženjeri, a tako i inženjeri do prije 100 godina su stvarali čuda koristeći svoj zdrav razum, intuiciju i iskustvo stečeno kroz šegrtovanje s starijim i iskusnijim inženjerima. Čitajući knjige napisne i ilustrirane od tad vrhunskih inženjera imali su uvid u cjeloživotno znanje stečeno učenjem iz pogreška, a ponekad i nagađanja s malo sreće. Što inženjer treba napraviti kad dodavanje materijala ne čini konstrukciju čvršćom? Kako inženjer može osmisliti neku apstraktnu konstrukciju za dani problem, a da pritom minimizira troškove i materijal u nekom realnom vremenu?

U sadšnjosti svaki Inženjer ima na raspolaganju računala i programske jezike s kojima ima slobodu stvarati što god poželi. Metodom konačnih elemenata izbacujemo iz proces razmišljanja učenje temeljeno pokušajem i pogreškom iz stvarnih konstrukcija te simulacijom konstrukcija oblika zamišljenih od strane inženjera, zadanih opterećenja i odnos s okolinom simuliramo realni problem u djeliću vremena koje bi inženjeri potrošili bez uporabe računala. Izrada tih konstrukcija postaje jeftinija, brža i kvalitetnija. Tim pothvatom inežnjeri su došli do granice razmišljanja kako napraviti neku konstrukciju te je gotovo nemoguće izmisliti najbolji mogući ishod za dani problem uobičajenim načinom razmišljanja. Okrenuvši se prirodi jer je priroda imala milijune godina vremena za svoja rješenja, inženjeri su pronašli inspiraciju u evoluciji životinja i biljka oko nas. [1] – [5].

1.1. Cilj i plan rada

Cilj ovog rada je samostalna izrada algoritma Metode konačnih elemenata za 2D konstrukciju napravljene od jednodimenzionalnihkonačnih elemenata i upotreba tog algoritma u kombinaciji s evolucijskim algoritmom za evoluciju osnovne konstrukcije u bolju, čvršću, manje mase s obzirom na postavljena ograničenja dimenzija i odnos s okolinom.

Sami podaci osnovne konstrukcije i njezini čvorovi će biti izvezeni iz programskog paketa Blender, a za algoritam Metode konačnih elemenata i evolucijski algoritam će biti programirani Python programskom jeziku.

1.2. Pregled poglavlja

Rad se sastoji od sljedećih poglavlja:

- Prvo poglavlje ukratko opisuje problem ovog diplomskog rada
- Drugo poglavlje, *Evolucijski algoritmi,* u kojem se pobliže pojašnjava kako radi evolucijski algoritam? Koji su to dijelovi mehanizma evolucijskog algoritma? i poveznice računalne i prirodne evolucije.
- Treće poglavlje, *Metoda konačnih elemenata,* u kojem pojašnjavamo kako funkcionira solver za Metodu konačnih elemenata te kako se generiranju podaci konstrukcija potrebni za simulaciju
- Četvrto poglavlje, *Optimizacija konstrukcije s evolucijskim algoritmom,* u kojem pobliže pojašnjavamo kako evolucijski algoritam funkcionira s metodom konačnih elemenata

2. Evolucijski algoritmi

Evolucijski algoritmi su metaheuristički optimizacijski algoritmi zasnovani na populacijama jedinki. Evolucijski algoritam koristi mehanizme nadahnute biološkom evolucijom poput razmnožavanja, mutacija gena, rekombinacije i prirodne selekcije. Kandidati rješenja problema optimizacije igraju ulogu pojedinih organizama, odnosno jedinki u populaciji, a funkcija sposobnosti preživljavanja ili „fitnes“, određuje kvalitetu svake jedinke u pojedinoj generaciji. Početna generacija jedinki ili potencijalnih rješenja problema je generirana nasumično. Nakon toga sve jedinke populacije prolaze kroz test fitnes na temelju kojeg se odabiru najbolji pojedinci za razmnožavanje. Ciklus opet započinje evaluacijom fitnes populacije, a jedinke najslabijeg fitnesa bivaju odbačene. [6] – [10].

Proces odabira najboljih mogućih rješenja prema fitnesu jedinki je analogan Darwinovoj teoriji preživljavanja najsposobnijih. Rješenja koja imaju najbolji fitnes se razmnožavaju dalje, dok se ona najlošijeg fitnesa ne razmnožavaju, naravno kao i u stvarnosti ponekad se i najlošiji razmožavaju te time čine populaciju još raznovrsnijom! Danas evolucijski algoritmi imaj zaista raznovrsnu primjenu u području inženjerstva [11] – [15].

2.1. Prirodni algoritam

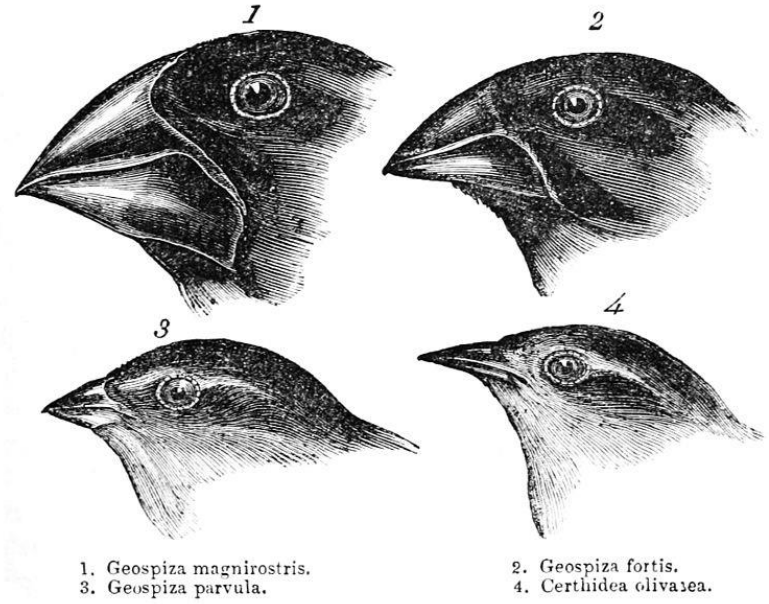
2.1.1. Teorija evolucije

Utemeljitelj moderne teorije o evoluciji je Charles Robert Darwin[6]. Darwin je svoju teoriju evolucije putem prirodnog puta temeljio na opažanju da živi svijet u pravilu proizvodi ukupno više potomstva nego što to potomstvo ima mogućnost preživljavanja.

Proces evolucije prirodnim odabirom određuju poznate činjenice o populaciji:

a) Osobine variraju među jedinkama u smislu izgleda, ponašanja
b) Različitosti uzrokuju različite stope preživljavanja i reprodukcije
c) Osobine se mogu prenositi iz generacije u generaciju

Zbog tih opažanja Darwin je zaključio da u svakoj generaciji populacije neke životinje pronalazimo sve više jedinki potomaka onih roditelja koji su najbolje prilagođeni svojem okolišu. Ti potomci imaju veće šanse za preživljavanje i razmnožavanje i time njihove osobine prevladavaju u budućim generacijama.



Slika 1. Darwinove zebe [5]

Darwinove zebe poznate kao i zebe Galapagosa je skupina od oko 18 vrsta ptica pjevica. Najpoznatije su po svojoj značajnoj razlici u obliku kljuna i njegove svrhe koji je prilagođen hrani pojedine vrste zeba. Sam oblik i raspored otočja Galapagos je znatno utjecalo na evoluciju zeba stanarica tih otoka. Relativna blizina pojedinih otoka daje veliku mogućnost slučajne selidbe ptica s jednog otoka na drugi, a udaljenost ih opet dovoljno razdvaja da se reprodukcija tih ptica odvija unutar populacije na pojedinom otoku. Selidbom na pojedini otok Darwinove zebe su bile suočene s novom okolinom i oskudicom hrane koju su jele do tad. Kao što vidite na slici(Slika 1). zebe imaju značajne razlike u obliku kljuna povezane s hranom koju jedu. Zebe s dugim kljunovima buše rupe u kaktus voću i jedu njegovu pulpu, dok one s kraćim kljunom rastrgaju to isto voće kako bi jeli pulpu, a uz to i pojedu kukce koji bi se našli u tom voću. Ovakve razlike u obliku kljuna ptica značajno povećavaju mogućnost preživljavanja jedinki na pojedinom otoku, a veća mogućnost preživljavanja, veća unesena količina hrane čini te jedinke bolje, brže i spremnije kao partnere za reprodukciju i propagiranje tih gena i osobina u daljnje generacije zeba.

2.1.2. Pojmovi Evolucijskih algoritama

Evolucijski algoritmi su usko povezani s sljedećim pojmovima: Nasumičnost, populacija, mutacija, prijelazna granica i selekcija. Pitanje je što ti pojmovi zapravo predstavljaju? Kako te pojmove uopće možemo predstaviti u računalnom smislu? Zašto imaju smisla? Koja je razlika naspram klasičnih determinističkih optimizacijskih algoritama?

Nasumičnost evolucijskih algoritama je čar i prokletstvo ovakvog rješenja optimizacijskog problema. Nasumičnost nam pruža mogućnost brzog konvergiranja ka dovoljno dobrom rješenju, a ista ta nasumičnost nam nikad neće dati isto rješenje. Za razliku od evolucijskih algoritama kao metaheurističkih algoritama, deterministički algoritmi će uvijek dati isto rješenje ako će početni uvjeti biti isti. Isti početni uvjeti kod evolucijskih algoritama će zbog mehanizma evolucijskih algoritama brzo biti zanemareni te će rezultati na kraju biti drugačiji.

Populacija evolucijske algoritme čini raznovrsnijima jer svaki član te populacije predstavlja potencijalno rješenje problema. Naravno, nisu svi članovi te populacije najbolja rješenja, svega par njih ili čak jedan predstavlja dobro rješenje dok drugi su tu samo zbog varijacije u samom rješenju. Kada bi od početka birali smo najbolje, riskiramo potencijalno zapinjanje u lokalnom optimumu. Križanjem najboljih s lošijima zamršujemo trenutačne najbolje jedinke ali time se povećava šansa pronalaska najboljeg globalnog rješenja.

Mutacija inspirirana ulogom mutacije DNA u živim bićima, evolucijski algoritam mutira potomke djece nasumičnim promjenama u njihovim osobinama iz kojih se određuje fitnes. Svaka mutacija može poboljšati ili pogoršati rezultat najboljih pojedinaca, stoga se primjenjuje elitizam u smom algoritmu koji najbolje pojedince gura nemutirane dalje u generacije i time čuva najbolji mogući rezultat.

Prijelazna granica inspirirana seksualnom reprodukcijom živih bića, svaki potomak ima kombinaciju svojstva pojedinog roditelja. Postotak svojstva pojedinog roditelja u potomku određuje prijelazna granica, a ta granica je nasumična i time još dodaje samoj nasumičnosti

proces evolucijskog algoritma i krajnjem rezultatu koji će uvijek odstupati od prethodnog eksperimenta iako imamo iste početne uvjete.

Selekcija inspirirana prirodnom selekcijom u evoluciji, evolucijski algoritam vrši proces selekcije tako da jedinke ili rješenja najlošijeg fitnes bivaju odbačeni, a oni najboljeg fitnes propagiraju dalje u sljedeće generacije, te se od njih reprodukcijom stvaraju potomci. Naravno ako bi otpočetka uzimali smo najbolje, velika je mogućnost da će rješenje konvergirati u neki lokalni minimum, stoga u evolucijskom algoritmu kod reprodukcije uzimamo i jedinke ili rješenja slabijeg fitnesa jer će možda baš njihov dio gena dodati vjerojatnosti da cijela populacija konvergira u globalni optimum.

2.1.3. Osnovni elementi evolucijskih algoritama u općem smislu

Evolucijski algoritmi iterativno kroz pokušaj i pogrešku konvergiraju ka željenom obliku rješenja. Misao da se problem riješi sam od sebe je isprva kontraintuitivna ali ako se taj algoritam raščlani na dijelove koji čine mehanizam pokretač konvergiranja ka krajnjem rješenju, taj algoritam itekako počinje imati smisla.

Na sljedećoj slici (slika 2) prikazan je pseudokod evolucijskog algoritma čiji će se dijelovi mehanizma evolucije detaljno objasniti u daljnjem dijelu ovog potpoglavlja.

```
POČETAK
    INICIJALIZIRAJ populaciju jedinki nasumičnih svojstva;
    OCIJENI pojedine jedinke u populaciji;
    PONAVLJAJ DO (uvjet prekida zadovoljen)
        1 IZABERI jedinke za potencijalne roditelje;
        2 SPOJI parove roditelja za reprodukciju;
        3 REPRODUKCIJA
        4  MUTIRAJ potomke;
        5  OCIJENI potomke;
    prekid
KRAJ
```

Slika 2. Pseudokod evolucijskog algoritma

INICIJALIZIRAJ – Prva i osnovna stvar svakog evolucijskog algoritma je inicijalizacija početne populacije jedinki od kojih kreče daljnja selekcija za reprodukciju. Svaka jedinka početne populacije je karakteristična sama za sebe svojim podacima iz kojih se računa fitnes jedinke. Poželjno je da su sve različite po nekom parametru ali nije nužno jer će se svakako u daljnjem mehanizmu evolucijskog algoritma uvesti mutacije koje su jedan od pokretača mehanizma evolucije prema globalnom minimumu.

OCIJENI – podaci ili „geni“ su svojstva koja svaku jedinku predstavljaju i na temelju kojih se ocjenjuje njihova sposobnost preživljavanja, sposobnost reprodukcije i pozicija u populaciji naspram drugih. Provlačenjem tih podataka kroz fitnes funkciju dobivamo sliku jedinke, odnosno koliko je to rješenje dobro za naš zadani problem. Jedinke ocijenjene najbolje su najpoželjnije za daljnju reprodukciju stoga njih uvažavamo u daljnjem mehanizmu evolucijskog algoritma. One najlošije ocijenjene naspram ostatka populacije bivaju eliminirane, a u smislu mehanizma evolucijskog algoritma jednostavno nisu uzete u obzir za daljnju reprodukciju.

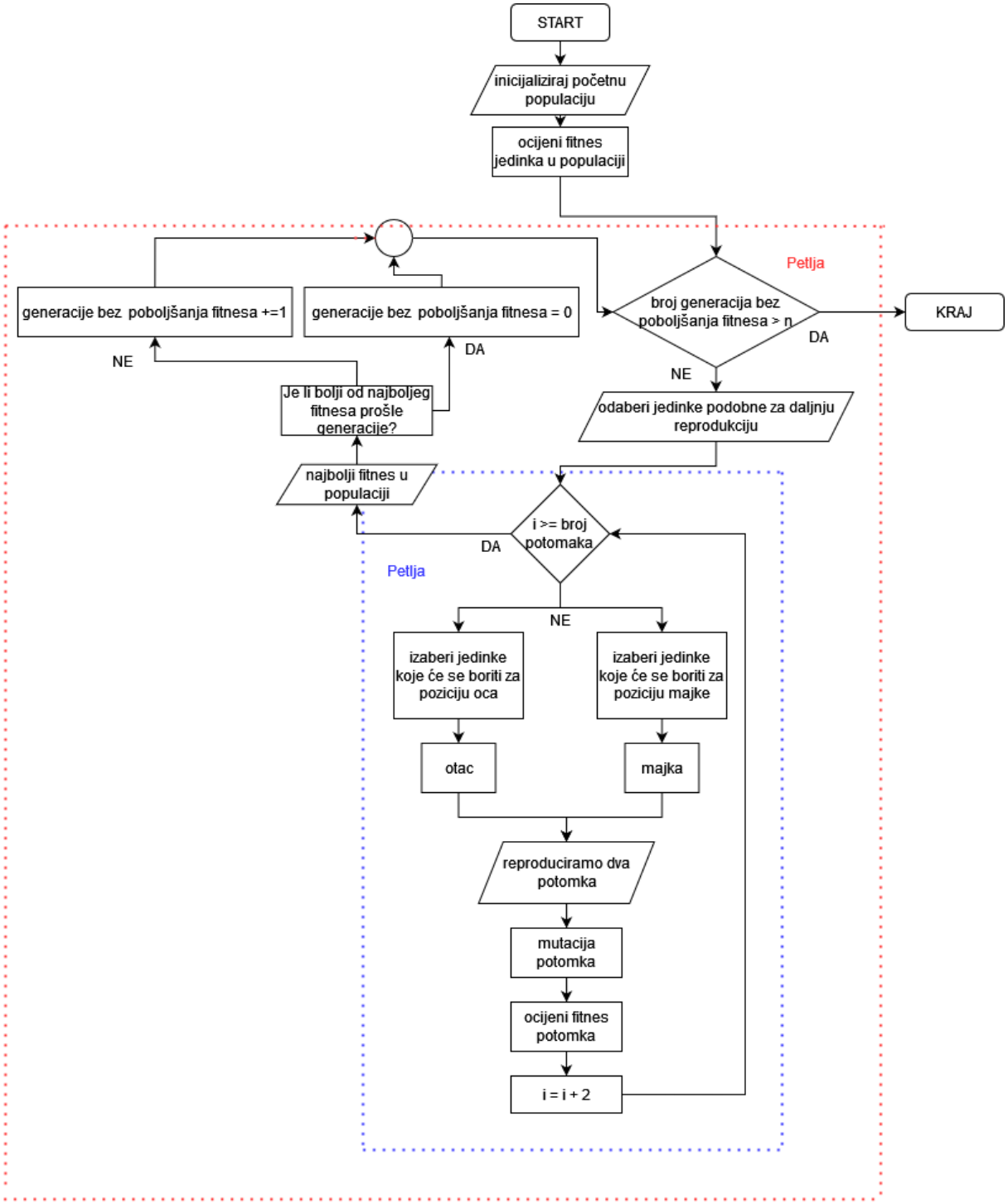
IZABERI – odabirom jedinki najboljeg fitnesa osiguravamo konvergenciju populacije rješenja prema minimumu no odabirom samo najboljih populacije vrlo brzo upada u lokalni minimum. Stoga odabirom jediniki lošijeg fitnesa za daljnju reprodukciju osiguravamo raznolikost rješenja koja će imati veću vjerojatnost pronalaska globalnog minimuma ili barem dovoljno dobrog globalnog minimuma.

SPOJI – postoje razni načini spajanja roditelja za reprodukciju, a u sklopu ovog diplomskog rada osvrnut ćemo se smo na spajanje putem borbe. U borbu za majku ulaze dva nasumično odabrana roditelja te onaj jačeg fitnesa osvaja poziciju majke za reprodukciju, a isto tako i za oca. Borbama osiguravamo bržu konvergenciju prema minimumu, a nasumičnim odabirom boraca osiguravamo unos različitih gena lošijih od najboljih pojedinki koji možda potaknu rješenje populacije prema globalnom minimumu.

REPRODUKCIJA – u ovom koraku nastaje potomstvo miješanjem gena majke i oca. U sklopu ovog diplomskog rada miješanje postoka gena majke i oca se odvija prema nasumičnoj prijelaznoj granici gena. Recimo da je prijelazna granica vrijednosti 0,4 to znači da će jedan potomak imati 40% gena svoje majke, a 60% gena svoga oca. Takvim pravilom možemo dobiti i izgubiti na kvaliteti gena koji propagiraju u daljnje generacije stoga uz tog jednog potomka stvaramo i potomka s 40% gena svog oca i 60% gena svoje majke. Što ako takvim dijeljenjem gena izgubimo roditelja najboljeg fitnes? To rješavamo tako da u sljedeće generacije propagiramo najbolju jedinku bez mutacija, takva jedinka će sigurno svoje dobre gene propagirati u daljnje generacije.

MUTIRAJ – u ovom koraku mehanizma evolucijskog algoritma mutiramo gene potomaka što može i ne mora poboljšati njihov fitnes. Nasumične mutacije su jedan od dijelova mehanizma koji čine evolucijski algoritam metaheuristučan od kojih Loše mutacije neće osigurati preživljavanje jedinki pa njihovi geni neće biti propagirani u daljnje generacije. S druge strane dobre mutacije jedinkama poboljšavaju fitnes jedinke čiji geni će propagirati u buduće generacije.

Sljedećim dijagramom toka(Slika 3.) detaljnije ćemo objasniti mehanizam evolucijskog algoritma koji se primjenjuje za rješavanje općeg problema. Ovo je samo uvod u rad mehanizma evolucijskog algoritma koji će se u potpunosti objasniti u četvrtom poglavlju ovog diplomskog rada, dok ćemo u trećem poglavlju odgovoriti na pitanje što to zapravo predstavlja fitnes funkciju mehanizma? Kako se računa fitnes funkcija? Te koji su to osnovni podaci pokretači rada evolucijskog algoritma u ovom optimizacijskom problemu?



Slika 3. Dijagram toka evolucijskog algoritma

3. METODA KONAČNIH ELEMENATA

U ovom poglavlju objasnit ćemo osnovnu teoriju metode konačnih elemenata primijenjenu u analizi svojstva konstrukcije. Valjalo bi se povezati na drugo poglavlje kako konstrukcija opisana metodom konačnih elemenata predstavlja jedinku u populaciji, odnosno predstavlja rješenje problema koje želimo dovesti do globalnog optimuma ili dovoljno dobrog rješenja.

Želimo ostvariti što čvršću konstrukciju, a da pritom ima što manju masu!

3.1. Teorija metode konačnih elemenata

Metoda konačnih elemenata ima veliki izbor različitih konačnih elemenata kojima se može više i manje vjerno simulirati reakcija konstrukcije na zadano opterećenje. U sklopu ovog diplomskog rada obrađivati ćemo smo jednodimenzionalne konačne elemente korištene u simuliranju 2D i 3D konstrukcija. Kombinacijom različitih tipova elemenata se može dobiti najbolji i najvjerniji rezultat simulacije koji s određenim predznanjem vjerno prikazuje realni problem. Ovaj diplomski rad je više usmjeren istraživačkom smislu i stoga će se zanemariti određeni dijelovi mehanizma metode konačnih elemenata bez kojih simulacija za neki pravi projekt ne bi bila valjana.

3.1.1. Izvod matrice krutosti za jednodimenzionalništapni element

Osnovni koncept zasnovan je na hookovom zakonu koji ćemo opisati s jednodimenzijskim elementom u 2D problemu . Što nam to govori Hookov zakon? [7]

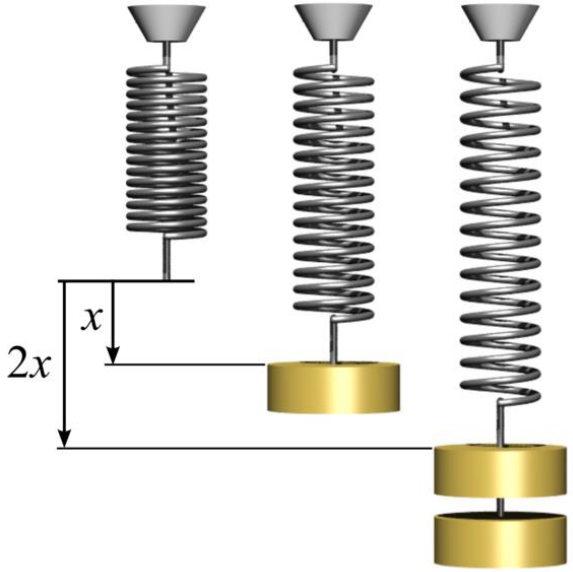
Hookov zakon je zakonitost koja opisuje ovisnost promjene oblika čvrstog tijela u obliku štapa proporcionalno djelovanju vanjske sile. Opterećenjem izazvano naprezanje $\boldsymbol{\sigma}$ razmjerno je deformaciji $\boldsymbol{\varepsilon}$, odnosno:

$$\boldsymbol{\sigma} = \boldsymbol{E} * \boldsymbol{\varepsilon} \tag{1}$$

Ako jednadžbu (1) pomnožimo s površinom presjeka A tog čvrstog tijela, dobivamo izraz:

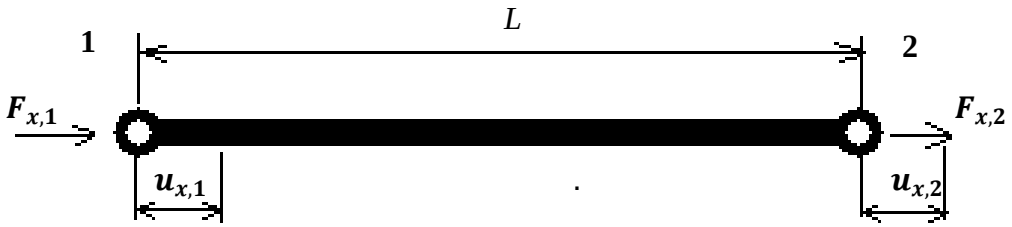
$$\boldsymbol{F} = \boldsymbol{k} * \boldsymbol{x} \tag{2}$$

Gdje *k* predstavlja konstantu krutosti tog tijela, *x* promjenu duljine tog tijela.



Slika 4. Proporcionalnost hookovog zakona prikazana oprugama [9]

Hookov zakon je uglavnom prikazan u kontekstu opruga (Slika 4.) u čijem slučaju se krutost opisuje konstantom opruge. U sklopu ovog diplomskog fokusirati ćemo se na konstrukcije s aksijalno opterećenim elementima.



Slika 4. aksijalno opterećeni element

Aksijalno opterećeni element (Slika 4.) opterećen je dvama sila $\boldsymbol{F_{x,1}}$ $\boldsymbol{i}$ $\boldsymbol{F_{x,2}}$, po jednu na svakom kraju ili čvoru. Iz hookovog zakona znamo da će te sile uzrokovati pomak $\boldsymbol{u_{x,1}}$ $\boldsymbol{i}$ $\boldsymbol{u_{x,2}}$ u čvorovima 1 i 2. Sljedeći korak je dobiti izraz kroz Hookov zakon koji će povezati sile $\boldsymbol{F_{x,1}}$ $\boldsymbol{i}$ $\boldsymbol{F_{x,2}}$ s pomacima $\boldsymbol{u_{x,1}}$ $\boldsymbol{i}$ $\boldsymbol{u_{x,2}}$.

Iz [1] na stranici 178 dobivamo izraz za računanje naprezanja pri vlaku ili tlaku:

$$\boldsymbol{\sigma = \frac{F}{A}} \quad (\mathbf{3})$$

Iz jednadžbe (3) dobivamo izraz za silu *F* :

$$\boldsymbol{F = \sigma * A} \quad (\mathbf{4})$$

Spajanjem izraza (4) , (2) dobivamo izraz:

$$\boldsymbol{k * x = \sigma * A} \quad (\mathbf{5})$$

Spajanjem izraza (5) i (1) dobivamo izraz:

$$\boldsymbol{k * x = E * \varepsilon * A} \quad (\mathbf{6})$$

A iz izraza (6) izvodimo izraz za *k* s kojime ćemo povezati $\boldsymbol{F_{x,1}}$ $\boldsymbol{i}$ $\boldsymbol{F_{x,2}}$ s pomacima $\boldsymbol{u_{x,1}}$ $\boldsymbol{i}$ $\boldsymbol{u_{x,2}}$.

$$\boldsymbol{k = \frac{EA}{L}} \quad (\mathbf{7})$$

Gdje je *E* Youngov modul elastičnosti, A površina poprečnog presjeka elementa, a L duljina smog elementa. Obzirom na čvor 1 možemo izvesti odnos sile u pomaka ovako:

$$\boldsymbol{F_{x,1} = \frac{EA}{L}(u_{x1} - u_{x2})} \quad (\mathbf{8})$$

$$\boldsymbol{F_{x,1} = \frac{EA}{L}u_{x1} - \frac{EA}{L}u_{x2}} \quad (\mathbf{9})$$

Analogno kao za čvor 2 izvodimo odnos sile i pomaka za čvor 1:

$$\boldsymbol{F_{x,2} = \frac{EA}{L}(u_{x2} - u_{x1})} \quad (\mathbf{10})$$

$$\boldsymbol{F_{x,2} = \frac{EA}{L}u_{x2} - \frac{EA}{L}u_{x1}} \quad (\mathbf{11})$$

Sada možemo izraze (9) i (11) spojiti i prikazati u formi matrice:

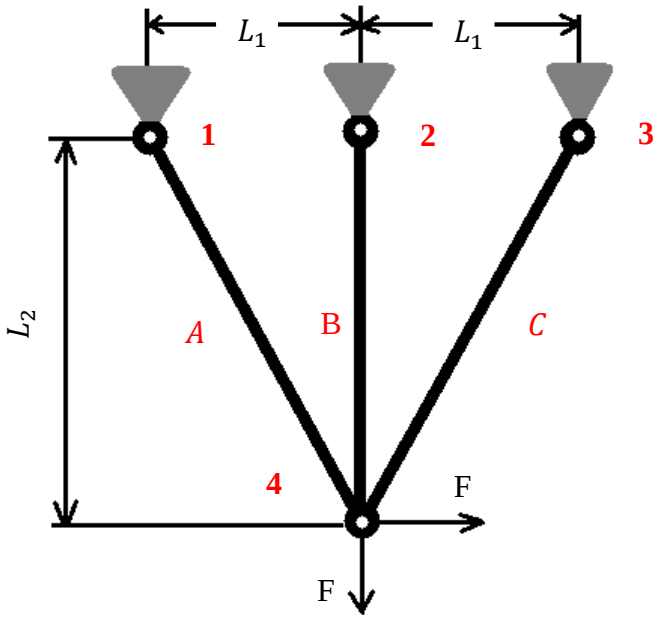
$$\begin{Bmatrix} F_{x,1} \\ F_{x,2} \end{Bmatrix} = \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} u_{x,1} \\ u_{x,2} \end{Bmatrix} \quad (\mathbf{12})$$

Gdje izraz $\frac{EA}{L}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}$ predstavlja matricu krutosti za jednodimenzionalankonačni element odnosno lokalnu matricu krutosti $[\boldsymbol{K_L}]$.

Iz matrice krutosti izraza (12) možemo zaključiti da sile u svakom čvoru imaju isti iznos ali suprotan smjer, time potvrđujući statičku ravnotežu elementa. Lokalna matrica krutosti je osnovni dio metode analize direktnom krutosti, osnovni blok s kojime sklapamo model konstrukcije i računamo otklone točaka, sile u elementima te prikazujemo deformaciju cijele konstrukcije.

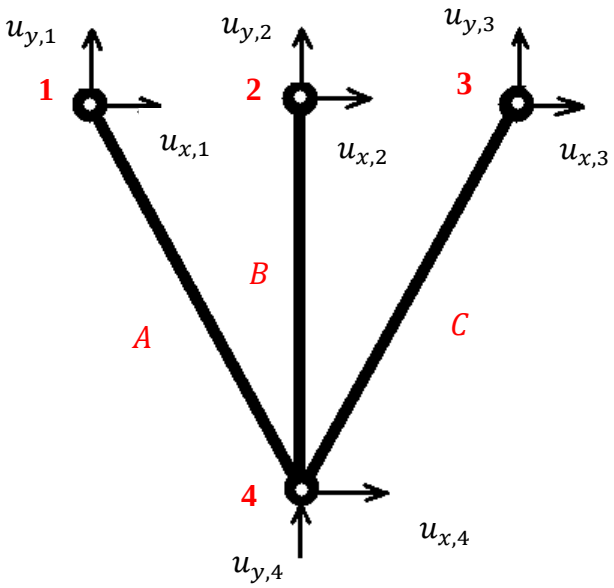
3.1.2. Sklapanje lokalnih matrica krutosti u globalnu matricu krutosti

U ovom potpoglavlju pojasniti ćemo kako se lokalne matrice krutosti spajaju jednu globalnu matricu krutosti pomoću koje računamo pomak ili iznos unutarnjih sila za bilo koji čvor konstrukcije na kojoj vršimo analizu direktnom pretvorbom matrice krutosti. Jednostavnom konstrukcijom s 3 elementa (Slika 5.) odnosno s općim primjerom biti će prikazan mehanizam spajanja lokalnih matrica krutosti u globalnu čiji princip se vrlo lako preslikava na sustave s složenijim konačnim elementima kao što su gredni element s dvanaest stupnjeva slobode.



Slika 5. Jednostavna konstrukcija s tri elementa

Prvi korak u sklapanju globalne matrice krutosti je identifikacija svih mogućih pomaka čvorova u dva moguća smjera, a te pomake zovemo stupnjevi slobode gibanja. Naša konstrukcija (Slika 5.) ima tri elementa i sveukupno osam stupnjeva slobode gibanja.



Slika 6. Jednostavna konstrukcija s stupnjevim slobode

Ako želimo lokalne matrice krutosti spojiti u jednu globalnu matricu krutosti, prvo moramo uzeti u obzir da svaki element konstrukcije ima drugačiju orijentaciju što znači da lokalni koordinatni sustav jednog elementa nije poravnat s lokalnim koordinatnim sustavima ostalih elemenata. U prijevodu nećemo moći zbrojati sile u zajedničkim čvorovima jer to kao prvo nema fizikalnog smisla, a kao drugo neće osigurati statičku ravnotežu elemenata. Stoga moramo izvršiti transformaciju lokalnih pomaka čvorova koje će se poklapati s globalnim referentnimkoordinatnim sustavom, a za to koristimo transformacijsku matricu $[T]$

Sljedeći izraz predstavlja transformacijsku matricu $[T]$:

$$\begin{Bmatrix} u_{x,L,1} \\ u_{x,L,2} \end{Bmatrix} = \begin{bmatrix} cos\,\theta & sin\,\theta & 0 & 0 \\ 0 & 0 & cos\,\theta & sin\,\theta \end{bmatrix} \begin{Bmatrix} u_{x,G,1} \\ u_{y,,G,1} \\ u_{x,G,2} \\ u_{y,G,2} \end{Bmatrix} \tag{13}$$

Gdje $u_{x,L}$ predstavlja pomak s obzirom na lokalni koordinatni sustav, a $u_{x,G}$ predstavlja pomak s obzirom na globalni koordinatni sustav. Izraz (13) možemo predstaviti u skraćenom obliku kao:

$$\{u_{x,L}\} = [T]\{u_G\} \tag{14}$$

Analogno procesu dobivanja izraza (14) koji povezuje pomake ovisno o lokalnom i globalnom koordinatnom sustavu izvodimo izraz koji povezuje iznose odnose iznos sila lokalnog i globalnog koordinatnog sustava.

$$\begin{Bmatrix} F_{x,G,1} \\ F_{y,G,1} \\ F_{x,G,2} \\ F_{y,G,2} \end{Bmatrix} = \begin{bmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ 0 & cos\,\theta \\ 0 & sin\,\theta \end{bmatrix} \begin{Bmatrix} F_{x,L,1} \\ F_{x,L,2} \end{Bmatrix} \tag{15}$$

Izraz (15) možemo urednije pokazati u ovome obliku:

$$\{F_G\} = [T]^T\{F_L\} \tag{16}$$

Ako izraz (12) prikažemo u skraćenom obliku:

$$\{F_L\} = K_L\{u_L\} \tag{17}$$

I umetnemo izraz (17) u izraz (16) dobivamo:

$$\{F_G\} = [T]^T K_L\{u_L\} \tag{18}$$

Nakon tog umećemo (14) u (18) i dobivamo izraz:

$$\{F_G\} = [T]^T K_L[T] \quad \{u_G\} \tag{19}$$

Stoga $[T]^T K_L[T]$ identificiramo kao globalnu matricu krutosti $[K_G]$ koja povezuje iznose globalnih sila s iznosima globalnih pomaka u čvorovima konstrukcije

Zbog jasnoće u daljnjem označavanju *sinus* i *cosinus* označavat će se s *s* i *c* :

$$[K_G] = [T]^T K_L[T] = \frac{EA}{L}\begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix} \tag{20}$$

Sada možemo izračunati globalnu matricu krutosti pojedinog elementa ovisno o njegovoj orijentaciji koje imenujemo $K_{G,A}, K_{G,B}, K_{G,C}$

Svaka matrica krutosti elementa je dimenzije 4x4, no mi ćemo ju rastaviti u manje kvadrante kako bi matrica ispala dimenzije 2x2 :

$$[K_G] = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \tag{21}$$

Gdje je:

$$K_{11} = EA\begin{bmatrix} c^2 & cs \\ cs & s^2 \end{bmatrix} \tag{22}$$

$$K_{12} = EA\begin{bmatrix} -c^2 & -cs \\ -cs & -s^2 \end{bmatrix} \tag{23}$$

$$K_{21} = EA\begin{bmatrix} -c^2 & -cs \\ -cs & -s^2 \end{bmatrix} \tag{23}$$

$$K_{22} = EA\begin{bmatrix} c^2 & cs \\ cs & s^2 \end{bmatrix} \tag{24}$$

Sad možemo postaviti obrazac osnovne matrice krutosti pomoću koje će biti jednostavno vidjeti gdje pojedini kvadranti trebaju biti postavljeni. Pošto konstrukcija(Slika 5.) ima četiri čvora, ovaj obrazac će biti dimenzija 4x4:

$$[K_p] = \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \tag{25}$$

Svaki element u izrazu (25) predstavlja matricu dimenzije 2x2 pa je ovaj izraz ustvari matrica dimenzije 8x8 .

$$\begin{Bmatrix} F_{x,G,1} \\ F_{y,G,1} \\ F_{x,G,2} \\ F_{y,G,2} \\ F_{x,G,3} \\ F_{y,G,3} \\ F_{x,G,4} \\ F_{y,G,4} \end{Bmatrix} = [K_p] \begin{Bmatrix} u_{x,G,1} \\ u_{y,G,1} \\ u_{x,G,2} \\ u_{y,G,2} \\ u_{x,G,3} \\ u_{y,G,3} \\ u_{x,G,4} \\ u_{y,G,4} \end{Bmatrix} \tag{26}$$

Sad kad imamo izraz za osnovnu matricu krutosti moramo u istoj matrici maknuti stupnjeve slobode gibanja koji su ograničeni vezama s okolinom modela. Čvor 1,2 i 3 su zglobni oslonci što znači da ovaj model konstrukcije ima šest stupnjeva slobode manje. To radimo tako što za pomake *u* čiji iznos je 0 izjednačimo s 0 u izrazu (26), a kako ne bi izgubili dimenziju matrice dijagonalno postavljamo jedinice po osnovnoj matrici krutosti.

$$\begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ F_{x,G,4} \\ F_{y,G,4} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & broj & broj \\ 0 & 0 & 0 & 0 & 0 & 0 & broj & broj \end{bmatrix} \begin{Bmatrix} u_{x,G,1} \\ u_{y,G,1} \\ u_{x,G,2} \\ u_{y,G,2} \\ u_{x,G,3} \\ u_{y,G,3} \\ u_{x,G,4} \\ u_{y,G,4} \end{Bmatrix} \tag{26}$$

Ovu reduciranu matricu krutosti definiramo kao matricu krutosti strukture $[K_S]$. Sad možemo vidjeti da postavljanjem 0 u osnovnu matricu krutosti definiramo veze s okolinom:

$$\begin{aligned} 0 &= 1\ x\ u_{x,G,1} \\ 0 &= 1\ x\ u_{y,G,1} \\ 0 &= 1\ x\ u_{x,G,2} \\ 0 &= 1\ x\ u_{y,G,2} \\ 0 &= 1\ x\ u_{x,G,3} \\ 0 &= 1\ x\ u_{y,G,3} \end{aligned} \tag{28}$$

Postavljanje 0 u stupce osnovne matrice krutosti osigurava da poznati iznosi pomaka ostanu iznos 0. Sad možemo iz izraza (26) izračunati pomake čvora četiri:

$$\begin{Bmatrix} F_{x,G,4} \\ F_{y,G,4} \end{Bmatrix} = \begin{bmatrix} broj & broj \\ broj & broj \end{bmatrix} \begin{Bmatrix} u_{x,G,4} \\ u_{y,G,4} \end{Bmatrix} \tag{29}$$

A kad izračunamo pomake $u_{x,G,4}$ i $u_{y,G,4}$ uvrštavamo ih u osnovnu matricu krutosti i računamo reakcije u osloncima konstrukcije.

$$\begin{Bmatrix} F_{x,G,1} \\ F_{y,G,1} \\ F_{x,G,2} \\ F_{y,G,2} \\ F_{x,G,3} \\ F_{y,G,3} \\ F_{x,G,4} \\ F_{y,G,4} \end{Bmatrix} = [K_p] \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ u_{x,G,4} \\ u_{y,G,4} \end{Bmatrix} \tag{30}$$

Sile u elementima računamo izrazom:

$$F = k\ x\ \delta \tag{31}$$

Odnosno kažemo da je sila u elementu između čvora *i* i *j* određena sljedećim izrazom:

$$F_{i,j} = \frac{EA}{L}\left(u_{x,L,j} - u_{x,L,i}\right) \tag{32}$$

Da bi mogli izračunati sile u pojedinom elementu moramo pretvoriti pomake iz globalnih koordinata u pomake lokalnih koordinata:

$$\begin{Bmatrix} u_{x,L,1} \\ u_{y,L,4} \end{Bmatrix} = [T] \begin{Bmatrix} u_{x,G,1} \\ u_{y,G,1} \\ u_{x,G,4} \\ u_{y,G,4} \end{Bmatrix} \tag{33}$$

Izrazi za ostale članove su analogni izrazu (33)

3.1.3. 3D gredni element s 12 stupnjeva slobode

U ovom potpoglavlju opisat ćemo osnovni gradivni element solvera, dijela mehanizma metode konačnih elemenata s kojime računamo unutarnje sile konstrukcije i određujemo pomake referentnih točaka koji su nužni za računanje ili „ocjenjivanje“ fitnes jedinki u populaciji.

Slika 7. Gredni element s 12 stupnjeva slobode

Ovakav gredni element sa sobom povlači neke pretpostavke koje će nam bitno olakšati izračun i smanjiti opterećenje procesora izračunima. Osnovne pretpostavke su da ne postoje međusobne ovisnosti sila u različitim ravninama. Tako sila u y,z ravnini ne utječe na iznos sile u x,z ravnini, a analogno njima vrijede i međusobni odnosi svih ostalih unutarnjih sila i momenata.

$$\begin{Bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \\ M_{x,i} \\ M_{y,i} \\ M_{z,i} \\ F_{x,j} \\ F_{y,j} \\ F_{z,j} \\ M_{x,j} \\ M_{y,j} \\ M_{z,j} \end{Bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{-EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI}{L^3} & 0 & 0 & 0 & \frac{-6EI}{L^2} & 0 & \frac{-12EI}{L^3} & 0 & 0 & 0 & \frac{-6EI}{L^2} \\ 0 & 0 & \frac{12EI}{L^3} & 0 & \frac{6EI}{L^2} & 0 & 0 & 0 & \frac{-12EI}{L^3} & 0 & \frac{6EI}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{-GJ}{L} & 0 & 0 \\ 0 & 0 & \frac{6EI}{L^2} & 0 & \frac{4EI}{L} & 0 & 0 & 0 & \frac{-6EI}{L^2} & 0 & \frac{2EI}{L} & 0 \\ 0 & \frac{-6EI}{L^2} & 0 & 0 & 0 & \frac{4EI}{L} & 0 & \frac{6EI}{L^2} & 0 & 0 & 0 & \frac{2EI}{L} \\ \frac{-EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-12EI}{L^3} & 0 & 0 & 0 & \frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & 0 & 0 & 0 & \frac{6EI}{L^2} \\ 0 & 0 & \frac{-12EI}{L^3} & 0 & \frac{-6EI}{L^2} & 0 & 0 & 0 & \frac{12EI}{L^3} & 0 & \frac{-6EI}{L^2} & 0 \\ 0 & 0 & 0 & \frac{-GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & \frac{6EI}{L^2} & 0 & \frac{2EI}{L} & 0 & 0 & 0 & \frac{-6EI}{L^2} & 0 & \frac{4EI}{L} & 0 \\ 0 & \frac{-6EI}{L^2} & 0 & 0 & 0 & \frac{2EI}{L} & 0 & \frac{6EI}{L^2} & 0 & 0 & 0 & \frac{4EI}{L} \end{bmatrix} \begin{Bmatrix} U_{x,i} \\ U_{y,i} \\ U_{z,i} \\ \phi_{x,i} \\ \phi_{y,i} \\ \phi_{z,i} \\ U_{x,j} \\ U_{y,j} \\ U_{z,j} \\ \phi_{x,j} \\ \phi_{y,j} \\ \phi_{z,j} \end{Bmatrix} \quad (13)$$

Iz izraza (13) koji predstavlja matricu krutosti grednog elementa s dvanaest stupnjeva slobode lako možemo zaključiti zašto vrijedi pretpostavka da unutarnje sile i momenti u ovom elementu nemaju međusobne odnose.

$$\{F_{x,i}\} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & \frac{-EA}{L} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} U_{x,i} \\ U_{y,i} \\ U_{z,i} \\ \phi_{x,i} \\ \phi_{y,i} \\ \phi_{z,i} \\ U_{x,j} \\ U_{y,j} \\ U_{z,j} \\ \phi_{x,j} \\ \phi_{y,j} \\ \phi_{z,j} \end{Bmatrix} \quad (14)$$

Ako pomnožimo prvi redak matrice krutosti s vektorom stupicom(izraz (14)) koji sadrži u sebi pomake i rotacije kako bi izračunali silu $F_{x,i}$ dobivamo sljedeći izraz:

$$F_{x,i} = \frac{EA}{L} * U_{x,i} - \frac{EA}{L} * U_{x,j} \quad (15)$$

Iz izraza (15) jasno se može zaključiti da nema međusobnog odnosa sila i pomaka iz različitih ravnina djelovanja. Dokažimo ovu pretpostavku tako što ćemo prikazati izraz za izračun $M_{x,j}$ izvađen iz izraza (13).

$$\{M_{x,j}\} = \begin{bmatrix} 0 & 0 & 0 & \frac{-GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \end{bmatrix} \begin{Bmatrix} U_{x,i} \\ U_{y,i} \\ U_{z,i} \\ \phi_{x,i} \\ \phi_{y,i} \\ \phi_{z,i} \\ U_{x,j} \\ U_{y,j} \\ U_{z,j} \\ \phi_{x,j} \\ \phi_{y,j} \\ \phi_{z,j} \end{Bmatrix} \tag{16}$$

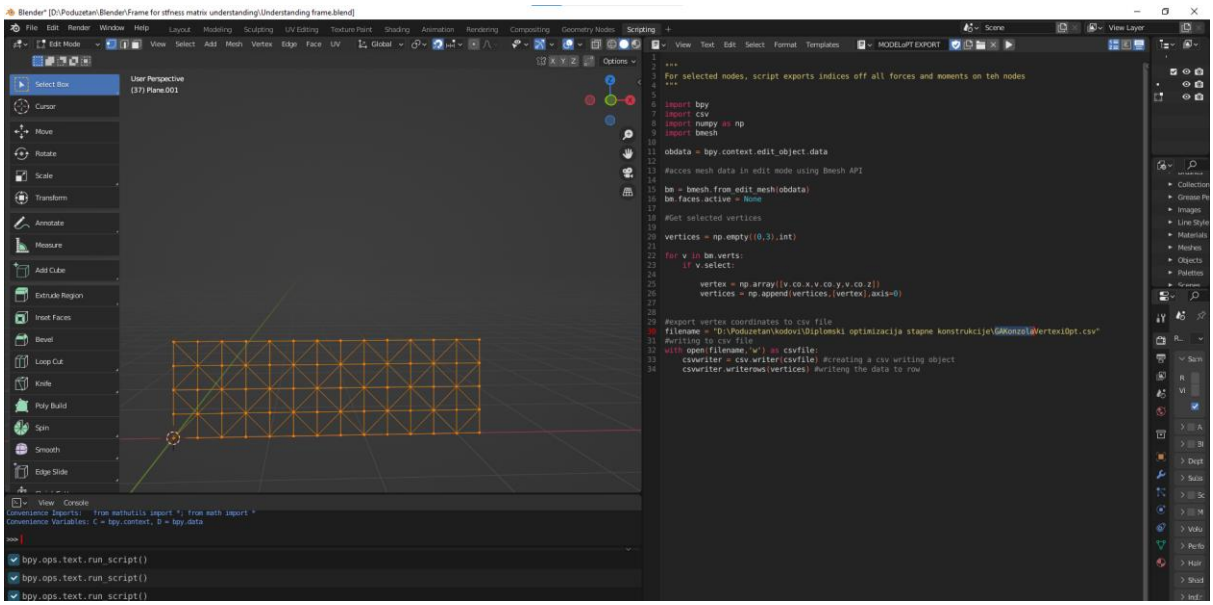
Množenjem retka matrice krutosti koji opisu torzijski moment elementa s vektorom pomaka i rotacija dobivamo sljedeći izraz:

$$M_{x,j} = \frac{-GJ}{L} \phi_{x,i} + \frac{GJ}{L} \phi_{x,j} \tag{17}$$

Iz izraza (17) opet se jasno vidi da torzijski moment $M_{x,j}$ ovisi smo o rotacijama čvorova ***i*** i ***j*** *s* obzirom na x os. Ovakav pristup problema značajno smanjuje težinu ovog algoritma za procesor. Iako to nije vjerni prikaz realnog problema, dovoljno je dobar za istraživanje, izučavanje i uvod u problematiku metode konačnih elemenata.

3.2. Generiranje podataka o konstrukciji

Programiranje software-a za generiranje podataka o konstrukciji je sam za sebe zahtjevan problem. Taj problem se neće obrađivati u sklopu ovog diplomskog rada, već će se koristiti gotovi software za dizajn „Blender“. Blender je besplatan i „open source“ program za izradu animiranih filmova, vizualnih efekata, umjetnosti, modela za aditivne tehnologije, interaktivne 3D aplikacije….



Slika 8. Grafičko sučelje Blender-a

Blender u sebi sadrži module pomoću kojih se dizajnira konstrukcija kao i u svakome inženjerskom CAD programu. U njega je ugrađena python konzola pomoću koje će se napisati kod za izvoz podataka o konstrukciji u obliku CSV datoteka.

Koji su to generirani podaci nužni za rad Evolucijskog algoritma i algoritma metode konačnih elementa?

Prvi podaci izvezeni za evolucijski algoritam su pozicije čvorova konstrukcije i broj čvorova koji opisuju svaki element u ovoj konstrukciji. Skripta za izvoz tih podatak se naziva *1-Model export.py*, a nalazi se u prilogu ovog diplomskog rada.

„.csv“ datoteka koja u sebi sadrži podatke *n* čvorova sprema te podatke na sljedeći način:

Red	X koordinata	Y koordinata	Z koordinata
1	0	0	0
2	5	2	3
…….			
n-1	3	4	7
n	5	3	9

Tablica 1. Primjer sadržajaCSV datoteke čvorova konstrukcije

Broj reda u csv datoteci predstavlja broj čvora u konstrukciji ali zbog sintakse Pythona čiji pristup podacima počinje numeriranjem od 0 do (duljina datoteke-1) ako želimo pristupiti podatku čvora ***n*** u .csv datoteci moramo ga pozvati s brojem ***n-1 .*** Ova numeracija je vrlo bitna jer se u evolucijskom algoritmu određeni čvorovi mutiraju, a čvorovi koji predstavljaju veze s okolinama i opterećenjima preskaču u procesu perturbacije mutacijama.

Druga CSV datoteka koja se nadovezuje na datoteku s pozicijama i brojevima čvorova je datoteka s čvorovima koji tvore pojedine elemente u konstrukciji, ta datoteka je generirana istom skriptom kao datoteka brojeva i pozicija čvorova. Svaki redak u toj datoteci predstavlja jedan element omeđen s dva čvora:

Element	Čvor *i*	Čvor *j*
1	25	4
…		
n -1	15	24
n	3	7

Tablica 2. Primjer sadržajaCSV datoteke elementa konstrukcije

Sljedeća skripta koju pokrećemo u blenderu je skripta *2-restraint export.py* koja se isto nalazi u prilogu, s njom izvozimo podatke o stupnjevima slobode koje ćemo ukloniti iz osnovne matrice krutosti $[K_p]$ kako bi dobili matricu krutosti konstrukcije $[K_s]$. Pokretanjem te skripte dobivamo dvije datoteke od kojih jedna sadrži brojeve čvorova čije stupnjeve slobode uklanjamo dok druga sadrži ograničenja stupnjeva slobode pojedinog čvora(prisjetimo se, svaki čvor ima 6 stupnjeva slobode što znači da konstrukcija ima 6**n* stupnjeva slobode gibanja gdje je *n* broj čvorova). Primjer sadržajadatoteka s brojevima stupnjeva slobode možete vidjeti u sljedećoj tablici gdje su *U* pomaci, a ϕ zakreti. Nule predstavljaju uklonjene stupnjeve slobode gibanja za čvor naveden u istom redu u datoteci s čvorovima za uklanjanje stupnjeva slobode gibanja.

Red	U_x	U_y	U_z	ϕ_x	ϕ_y	ϕ_z
1	25	26	27	0	0	0
…						
n-1	61	62	63	0	0	0
n	72	73	74	0	0	0

Tablica 3. Primjer sadržajaCSV datoteke uklonjenih stupnjeva slobode konstrukcije

Sljedeća skripta koju trebamo pokrenuti je *3-point force export.py* koja nam vrača podatke stupnjeva slobode po kojima djeluje opterećenje. Izgled tih podataka je analogan *Tablici 3.* s razlikom da nema 0 u podacima jer je to samo spremnik stupnjeva slobode gibanja.

Zadnja skripta koju moramo pokrenuti je *MODELoPT EXPORT.py* u kojoj se pohranjuju brojevi čvorova koje ćemo perturbirati kroz generacije Evolucijskog algoritma. Izvoz tih podataka je potpuno isti kao izvoz brojeva čvorova u *2-restraint export.py datoteku.*

4. Optimizacija konstrukcije s evolucijskim algoritmom

U ovom poglavlju pobliže ćemo obraditi kako je analiza metodom konačnih elemenata predstavljena u pogledu evolucijskih algoritama. Slijedno tome obradit ćemo sljedeća pitanja:

1. Kako su definirane jedinke u smislu analize metodom konačnih elemenata?
2. Po čemu su pojedine jedinke u populaciji rješenja analize metodom konačnih elemenata drugačije?
3. Na koji način se vrši mutacija pojedinih jedinka i zašto to ima smisla?
4. Kako se izvršava reprodukcija roditelja?
5. Što predstavlja i kako se računa fitnes rješenja?

Nakon toga analizirat ćemo nekoliko osnovnih štapnih konstrukcija u smislu optimizacije Evolucijskim algoritmom. Usporedit ćemo konvergenciju problema ka rješenju za više različitih postavka parametra , tablično i grafički ih prikazati te za svaku ostaviti komentar na rješenje. Primjeri konstrukcija koje ćemo analizirati su:

1. Opterećenje konstrukcije mosta
2. Tlačno opterećenje konstrukcije
3. Vlačno opterećenje konstrukcije
4. Opterećenje konstrukcije u koso

4.1. Što predstavljaju jedinke u ovoj optimizaciji Konstrukcije evolucijskim algoritmom?

Svaka jedinka u populaciji Evolucijskog algoritma predstavlja potencijalno rješenje optimizacijskog problema, a u ovom slučaju svaka jedinka ima svoje osnovne podatke prema kojima se ocjenjuje njezin fitnes i koji se dalje manipuliraju skupa sa podacima ostalih jedinki onako kako je mehanizam Evolucijskog algoritma zamišljen.

Pa kako onda izgledaju naše jedinke populacije koje inače u prirodi predstavljaju živa bića?

Prisjetimo se da u prirodi, a uzmimo za primjer vukove, taj svaki vuk je drugačije veličine, spretnosti, boje, snage … Te osobine znatno utječu na njihovu sposobnost preživljavanja i mogućnost reprodukcije pa i samu poziciju u čoporu. U čoporu vukova smo se alfa mužjak i ženka pare dok ostatak čopora jednostavno to prihvaća, niti jedan drugi mužjak i ženka se ne reproduciraju i time ne propagiraju svoje gene u daljnje generacije. Naravno svako malo se pronađe neki izazivač koji će možda parirati alfa mužjaku i pokušati preuzeti njegovo vodstvo, a time i njegov čopor. Samo vuk boljeg fitnesa od ostalih može pobijediti i na taj način propagirati svoje gene u buduće generacije.

No samo veličina nije bitna u sklopu fitnes, jer što će ti veličina koja može otjerati sve mužjake koji ti pariraju za vodstvo čopora ako taj vuk nema brzinu i okretnost koji su nužni za uspješan lov. Ako je lov često neuspješan njihova mladunčad neće imati dovoljno hrane, a time i snage za borbu života i preživljavanja te se ni njihovi geni neće propagirati u daljnje generacije. Boja samog vuka je vrlo bitna u igri lovca i lovine jer samo jedna kriva nijansa krzna vuka čini puno uočljivijim u okolini. Takav vuk će u prosjeku imati manje uspješan lov, a time i manje resursa za othraniti mladunčad i manju vjerojatnost propagiranju svojih gena u buduće generacije. Uzmimo za primjer životinje koje su zaslužujući nasumičnim mutacijama gena rođene kao albino životinje, odnosno zbog nedostatka pigmenta u svojoj, koži, krznu, pa čak i očima znatno uočljivije, kao u primjeru bijelog krzna u ljetnoj šumskoj okolini ili bijele krokodilske kože u močvarnom području (Slika 8.). Shodno tome možemo zaključiti da za fitnes jedinke nije bitan samo jedan parametar već više njih koji će u svojoj kombinaciji dati najbolji fitnes koji osigurava propagiranje dobrih gena, a time i veću stopu uspjeha generacija i uspjeha u rješavanju problema.

Slika 9. Albino krokodil [10]

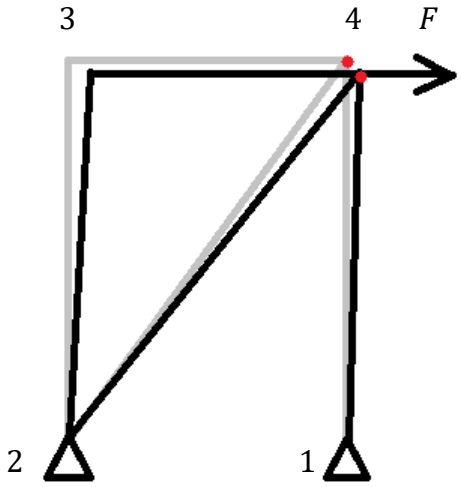
A sad ćemo se vratiti na naš osnovni problem Evolucijskog algoritma i metode konačnih elemenata, točnije fitnes jedinki. Što jednu konstrukciju čini boljom od ostalih?

U našem slučaju bolja konstrukcija je ona koja će za isto opterećenje kao i ostale konstrukcije imati najmanji progib, odnosno ona koja će imati najveću krutost, a pri tome da ima što manju sveukupnu masu. Ako imamo dvije konstrukcije koje obavljaju istu zadaću dok jedna od njih koristi puno manje materijala onda ćemo sigurno odabrati konstrukciju s manje materijala. Pogotovo u današnje vrijeme 2022. godine kad je cijena samog čelika u prosjeku skočila 30% , a sveukupna cijena projekta nije smo u cijeni čelika nego i u svoj periferiji tog proizvoda koji je jedan od sirovina potrebnih za konstrukcije. Smanjenjem obujma i mase čelika u samoj konstrukciji štedimo i na prijevozu čelika, a prijevoz u ovo doba postaje sve skuplji i skuplji. Taj čelik će se na kraju trebati premazati temeljnim bojama, zaštitnim sredstvima, i konačno bojama koje služe samo za estetiku konstrukcije, a to sve nadodaje nepotrebne troškove.

4.2. Kako se računa fitnes jedinki?

Računanje fitnesa objasniti ćemo kroz jednostavnu formulu i primjer tako da funkcija za fitnes glasi ovako:

$$fitnes = pomak\ referentne\ točke + masa\ konstrukcije + penal\ ako\ je\ sila\ u\ članovima\ veća\ od\ dopuštene$$



Slika 10. Deformirana konstrukcija

Slika 9. predstavlja konstrukciju s četiri čvora. Siva predstavlja konstrukciju bez opterećenja a crna konstrukciju deformiranu pod opterećenjem. Referentna točka je točka broj četiri označena s crvenom bojom, a njen pomak u odnosu na osnovnu konstrukciju se unosi u fitnes funkciju. Masa konstrukcije se računa pojednostavljeno kao suma svih međusobnih udaljenosti između dvaju točaka.

Formula korištena za prva dva parametra je sljedeća:

$$Udaljenost = \sqrt{(v_2 - v_1)^2 + (h_2 - h_1)^2} \qquad (18)$$

Gdje su v i h vertikalna i horizontalna koordinata referentne točke prije i poslije deformacije, a analogno tome računamo i masu odnosno sumu svih međusobnih udaljenosti između dvaju točaka.

Koristeći izraz (32) u mehanizmu se provjerava sila svakog člana u konstrukciji te ako sila premašuje dopuštenu makar u jednom članu u fitnes za koji želimo da bude što manji zbrajamo penal od npr. iznosa 10 koji time pogoršava fitnes i eliminira jedinku kao rješenje problema.

4.3 Što su to zapravo mehanizmi evolucijskog algoritma u smislu metode konačnih elemenata?

Prisjetimo se Pseudokoda evolucijskog algoritma (slika 2.) s početka ovog rada u kojemu su navedene osnovne operacije Evolucijskog algoritma, a od kojih su pet ponavljajući kroz svaku generaciju rješenja:

1. IZABERI
2. SPOJI
3. REPRODUKCIJA
4. MUTIRAJ
5. OCIJENI

U daljnjem textu ovog potpoglavlja podrobnije ćemo opisati što te operacije predstavljaju u smislu metode konačnih elemenata:

4.3.1. IZABERI

Jedinke podobne za daljnje operacije odabiru se prema par pravila postavljenim na početku algoritma:

- Elitizam
- Prvih *n* jedinki s najboljim fitnesom

Elitizam osigurava da najbolje jedinke opstaju nepromijenjenei time propagiraju svoje gene u daljnje generacije. Ako navedemo u algoritmu od svake generacije jednu jedinku s najboljim fitnesom šaljemo u sljedeću generaciju kao nepromjenjenu, osiguravamo sigurnu konvergenciju prema rješenju bez straha da ćemo kroz reprodukciju i spajanje jedinki u novo poitomstvo izgubiti kvalitetan genetski materijal. U pravilu se razmjerno veličini populacije određuje 1,2,3 ili više elitnih jedinki koje nepromijenjeneidu u sljedeću generaciju no treba

imati na umu ako se previše elitnih jedinki propagira da to dovodi do konvergencije rješenja u neki lokalni minimum.

Na početku algoritma navodimo koliko će najboljih jedinki iz prethodne generacije uči u borbe za reprodukciju kako bi time širile dalje svoje gene. Ako npr imamo populaciju od 100 jedinki odabirom parametra da u izbor za borbe idu prvih 30 jedinki najboljeg fitnes rješenje će sigurno ići prema boljem no tu postoji velika vjerojatnost da će rješenje završiti u nekom lokalnom minimumu. To rješavamo tako da uzmemo veći broj najboljih jedinki, npr 70 i više i time povećavamo raznolikost gena i mogučnost konvergencija ka globalnom minimumu. Ovdje bi se nadovezao na operaciju OCIJENI koja je ustvari računanje fitnesa objašnjeno u prošlom potpoglavlju 4.2. i operaciju SPOJI koja je objašnjena u drugom poglavlju na osmoj stranici.

4.3.2. REPRODUKCIJA

Nakon odabira jedinki iz populacija, njihovih međusobnih borba za pozicije očeva i majka sad kad imamo parove za reprodukciju objasniti ćemo kako ta reprodukcija funkcionira u smislu metode konačnih elemenata. Kao što je objašnjenu u potpoglavlju 3.2. svaka jedinka ima i svoj spremnik podataka o pozicijama čvorova. Svaka jedinka predstavlja konstrukciju s *b* čvorova koji su u obliku CSV datoteke pohranjeni točnim redoslijedom. Redoslijed čvorova od 1 do *n* je potpuno isti za svaku jedinku neovisno o koordinatama tih pojedinih čvorova. Pa kako onda vršimo reprodukciju očeva i majka kako bi dobili nove potomke?

Broj čvora	Koordinate čvorova za oca	Koordinate čvorova majke
1	[1,2,3]	[0.9,2.1,3.4]
2	[1,4,3]	[1,4.1,3]
…	…	….
n-1	[9,8,7]	[9.3,8.1,8.8]
n	[10,10,10]	[9.9,10.1,10.2]

Tablica 4. Koordinate čvorova majke i oca

Koliko će biti gena oca ili majke u potomstvu govori nam nasumični parametar prijelazne granice. Prijelazna granica određuje gdje i kako spajamo gene majke i oca pa ćemo prema tablici broj četiri prikazati kako se vrši reprodukcija. Recimo da je prijelazna granica dobivena nasumično vrijednosti 2. To znači da će jedan od dva potomka, nazovimo ga sinom, dobiti gene svog oca ili ti koordinate čvorova svog oca od čvora broj jedan do čvora broj dva. Od majke će dobiti gene ili ti koordinate čvorova od broja tri pa sve do broja *n.* Drugi potomak, nazovimo ga kćer, dobiti će prvi dio gena od majke odnosno čvorove od jedan do dva, a drugi dio gena će dobiti od oca odnosno čvorove koordinata od tri do *n* . U stvarnom svijetu ako majka i otac dobiju dvoje djece, njihova raspodjela gena neće biti ovako recipročna već nasumičnija, a za naš evolucijski algoritam to nije nužno jer evolucijski algoritmi su samo inspirirani mehanizmima iz prirode koji ne moraju biti vjerno prikazani kako bi odradili svoj posao, a ne moraju biti nužno bolji.

Broj čvora	Koordinate čvorova za sina	Koordinate čvorova kćer
1	[1,2,3]	[0.9,2.1,3.4]
2	[1,4,3]	[1,4.1,3]
…	…	….
n-1	[9.3,8.1,8.8]	[9,8,7]
n	[9.9,10.1,10.2]	[10,10,10]

Tablica. 5 Koordinate čvorova sina i kćeri

Stvaranjem dva potomka s recipročnim sadržajem gena oca i majke ne gubimo genetski materijal , ali možemo izgubiti kvalitetnu kombinaciju genetskog materijala oca ili majke. To se rješava pravilom elitizma koje najbolju jedinku šalje neprmijenjenuu sljedeću generaciju i kao takva ona dalje širi svoje gene u cijelu populaciju rješenja.

4.3.3. MUTIRAJ

Mutacija je jedan od pokretača mehanizma evolucijskog algoritma. Mutacija je nasumična promjena u genetskom materijalu jedinke koja ako bude loša smanjuje fitnes jedinke, a time i njezinu mogućnost reprodukcije. S druge strane mutacije koje poboljšavaju fitnes jedinke će ubrzati konvergenciju rješenja u neki minimum. Ako konstrukcija ima *n* čvorova, nećemo mutirati svaki čvor već samo njih par, recimo četvrtinu tako što ćemo koordinate četvrtine nasumičnih čvorova perturbirati za neku malu udaljenost. To su parametri koji se postavljaju nakon što dobijemo neko malo iskustvo izvođenjem evolucijskog algoritma s različitim parametrima.

Broj čvora	Koordinate čvorova prije mutacije	Koordinate čvorova poslije mutacije
1	[1,2,3]	[1.0001,2.0001,2.999]
2	[1,4,3]	[1,4,3]
…	…	…
n-1	[9,8,7]	[9,8.0001,7.0001]
n	[10,10,10]	[9.9999,10,10]

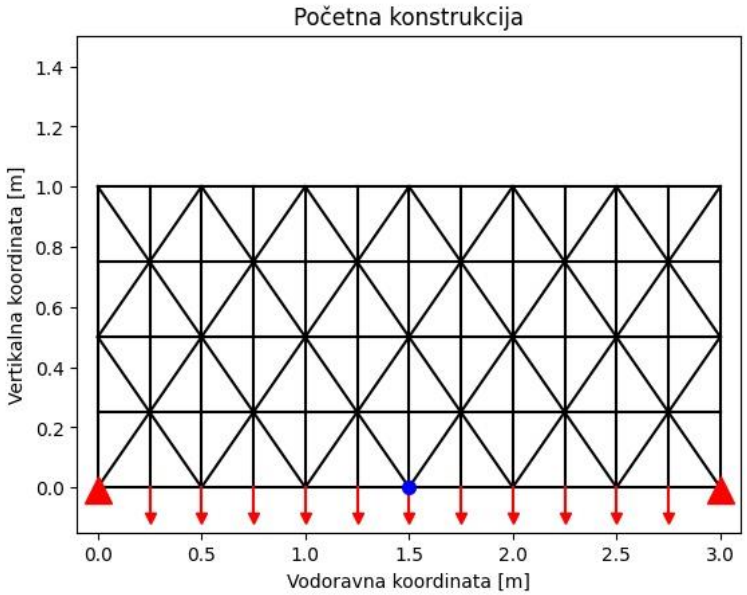
Tablica 6. Koordinate čvorova prije i poslije mutacije

4.4 Analiza rješenja Evolucijskog algoritma

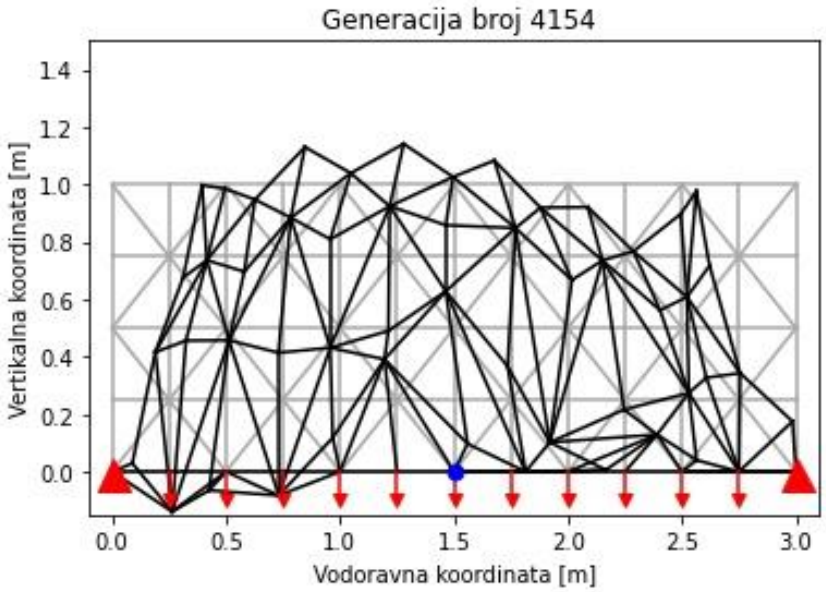
U ovome potpoglavlju analizirati ćemo evoluciju nekoliko osnovnih štapnih konstrukcija, usporediti ćemo ishode evolucijskog algoritma za različito postavljene parametre i komentirati zašto je ishod takav kakav je. Grafički ćemo prikazati konvergenciju rješenja, a isto tako ćemo prikazati konstrukcije prije i poslije optimizacije evolucijskim algoritmom. U daljnjem tekstu ispitat ćemo evoluciju tlačno opterećenje konstrukcije, vlačno opterećenje konstrukcije, koso opterećenje konstrukcije i opterećenje konstrukcije mosta.

4.4.1. Evolucija konstrukcije mosta

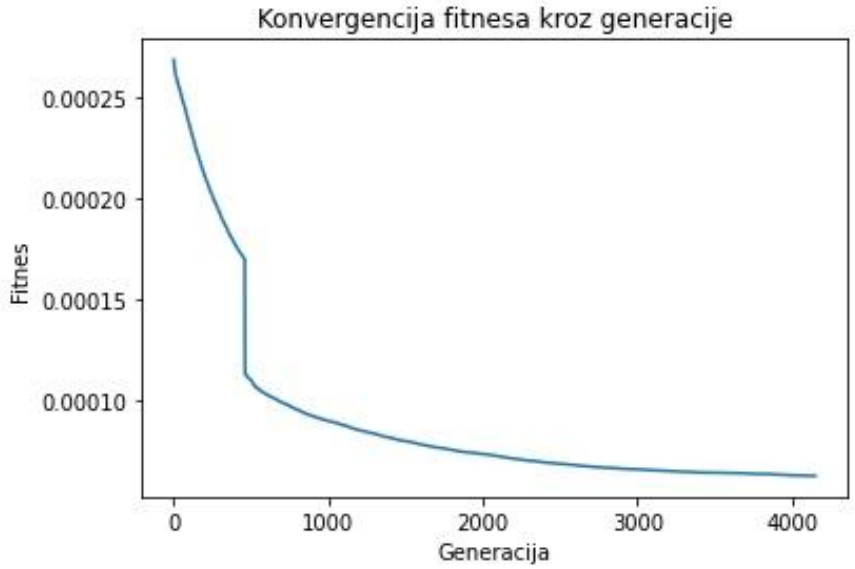
U ovom potpoglavlju prikazati ćemo i prokomentirati rezultate pokus evolucije konstrukcije mosta. Početni oblik konstrukcije(slika 10.) je opterećen s jedanaest koncentriranih opterećenja iznos 1000 N usmjernim prema dolje, a oslonjena je na dva zglobna oslonca označena s trokutima. Plava točka je referentna točka čiji pomak mjerimo i uračunavamo u funkciju fitnesa. Masa konstrukcije je određena ukupnim zbrojem duljina svakog pojedinog člana i kao takva je navedena bezdimenzijski u tablici parametra i rezultata pokusa.



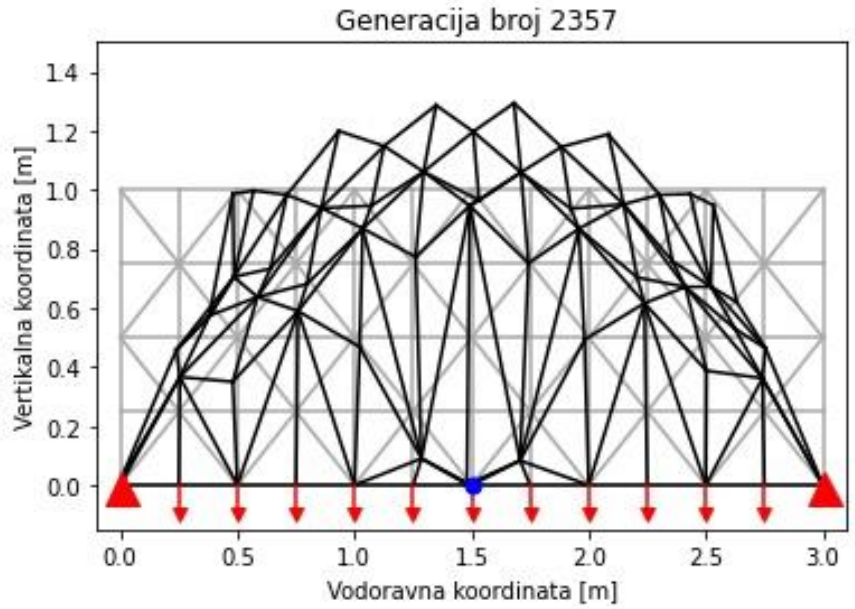
Slika 11. Početni oblik konstrukcije mosta



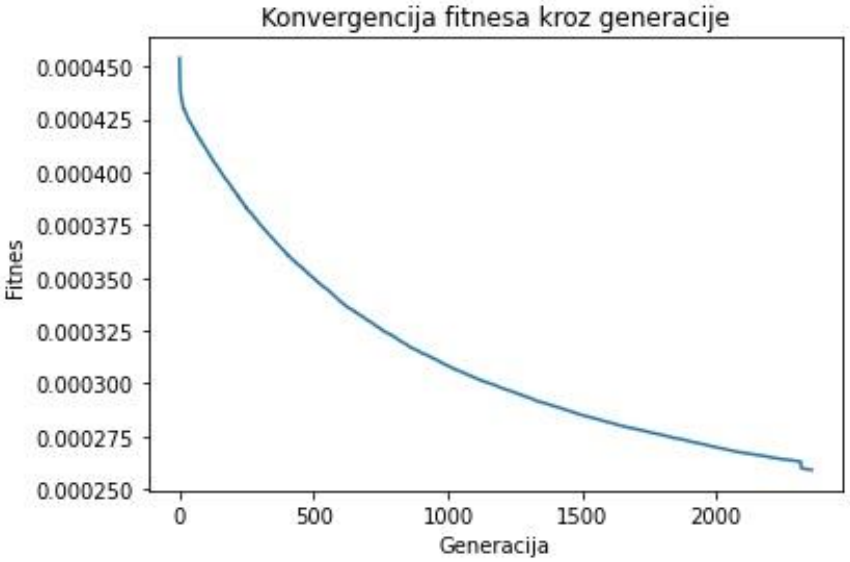
Slika 12. Prvi pokus evolucije mosta



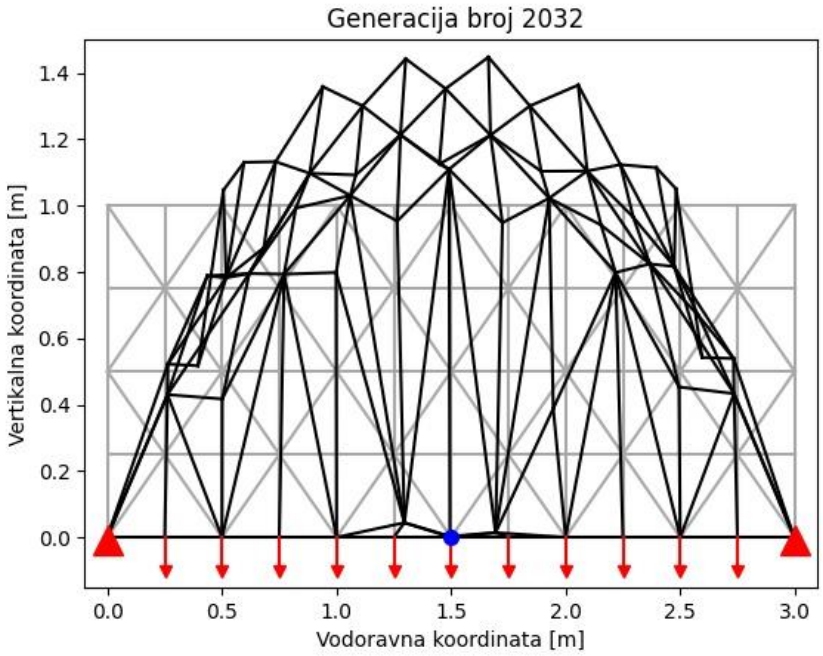
Slika 13. Fitnes kroz generacije za prvi pokus evolucije mosta



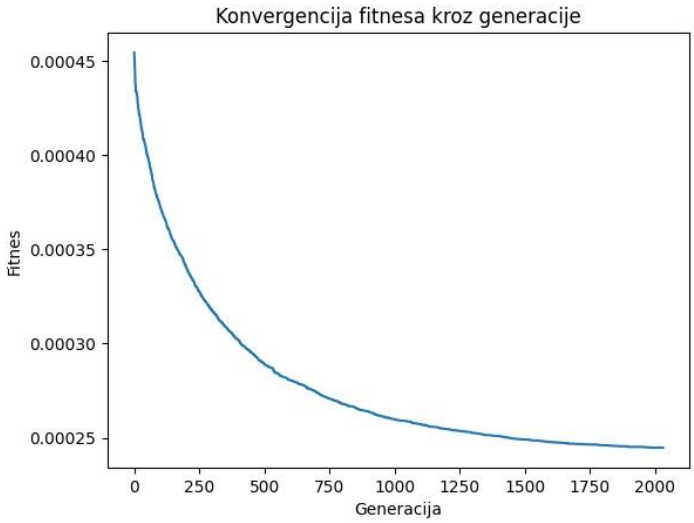
Slika 14. Drugi pokus evolucije mosta



Slika 15. Fitnes kroz generacije za drugi pokus evolucije mosta



Slika 16. Treći pokus evolucije mosta



Slika 17. Fitnes kroz generacije za za treći pokus evolucije mosta

Pokus	roditelji	elitni	Roditelji za potomstvo	Iznos mutacije	Broj mutacija čvorova	Generacija do rješenja	Progib referentne točke	Mas	Fitnes
Početna konstrukcija	/	/	/	/	/	/	$6{,}9878218 * 10^{-7}$	6,75	0,0004716779
1	70	1	62	0.003	28	4154	$5{,}42786859 * 10^{-7}$	6,13195	0,000332834677
2	120	1	115	0.003	35	2357	$1{,}620776 * 10^{-35}$	5,56818	$9{,}024781667\ 10^{-33}$
3	120	1	115	0.003	18	2032	$4{,}423411 * 10^{-7}$	5,53331109	0.0002447522692890891

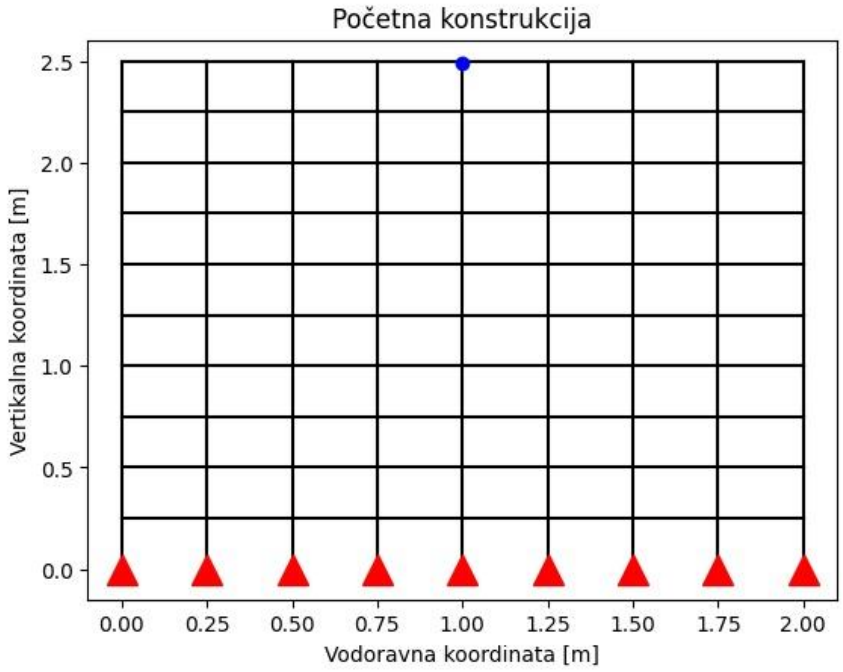
Tablica 7. Parametri i rezultati evolucija konstrukcije mosta

U tablici broj 7 možemo vidjeti parametre pojedinog pokus koji su rezultirali različitim rješenjima problema optimizacije konstrukcije mosta. Kroz pokuse evolucije mosta i evolucije drugih konstrukcijskih problema za ovaj pokus sam zaključio da je početna veličina populacije vrlo bitna za krajnje rješenje problema koji tad ima značajno drugačiji ishod i izgled. Postavljanjem parametra populacije na manje iznose smanjujemo raznolikost gena pojedinih jedinki i upadamo u lokalne minimume. Jedan takav lokalni minimum je prikazan prvim pokusom(Slika 11.) čiji fitnes i masa su bolji od fitnesa i mase početne konstrukcije. Pokus s takvim parametrima, a i takav algoritam je neefikasan i procesorski puno skuplji od sljedeća dva pokusa. U pokusu dva smo povećanjem populacije dobili vrlo nizak fitnes i progib referentne točke čak iznos 30 redova niži od pokusa broj jedan što je čudno, a može se pripisati stohastičkoj prirodi procesa ovog algoritma. Konstrukcija pokusa broj dva je počela stvarno ličiti na most naspram prvog pokus iz kojeg se tek može nagađati je li to most ili nije. Daljnjim pokusom broj tri i smanjenjem broja čvorova koje smo mutirali u svakoj generaciji postigli smo bržu konvergenciju ka rješenju za čak 300 generacija, s time da smo obavljali 15 računskih

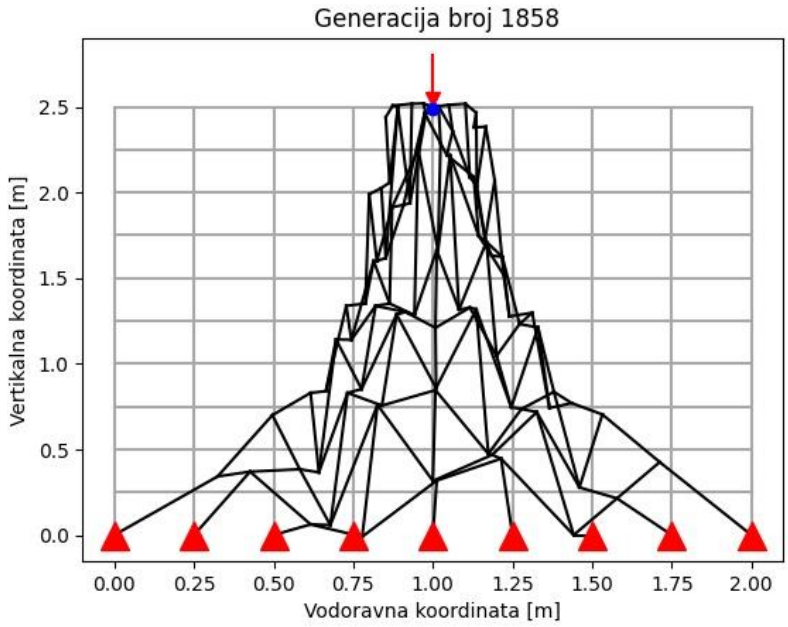
operacija mutacije manje naspram drugog pokusa (drugi pokus je imao 35 mutacija čvorova po jedinki dok je treći pokus imao 18 mutacija čvorova po jedinki). Trećim pokusom nismo postigli bolji fitnes no imali smo manje računskih operacija a konstrukcija je dobila pravilniji oblik. Mase konstrukcija su kao i očekivano smanjene nakon primjene evolucijskog algoritma. Krivulje fitnesa tijekom generacija u ova tri pokusa prate sličan trend konvergencije no prva ima jedno zanimljivo svojstvo. Krivulja iznos fitnes po generacijama prvog pokus(Slika 12.) ima skok u iznosu fitnes koji nam govori da se u jednoj generaciji dogodila značajna promjena u genomu populacije prema kojoj su počele težiti sve jedinke, dogodio se slučajni spoj genoma oca i majke,a time i poboljšanje fitnesa. Krivulja fitnesa kroz generacije drugog i trećeg pokusa nemaju skokove no jasno se vidi da krivulja trećeg pokusa konvergira puno strmije naspram krivulje drugog pokusa što možemo pripisati manjem broju mutacija čvorova a time i usmjerenijoj konvergenciji prema globalnom minimumu.

4.4.2. Evolucija konstrukcije pod tlačnim opterećenjem

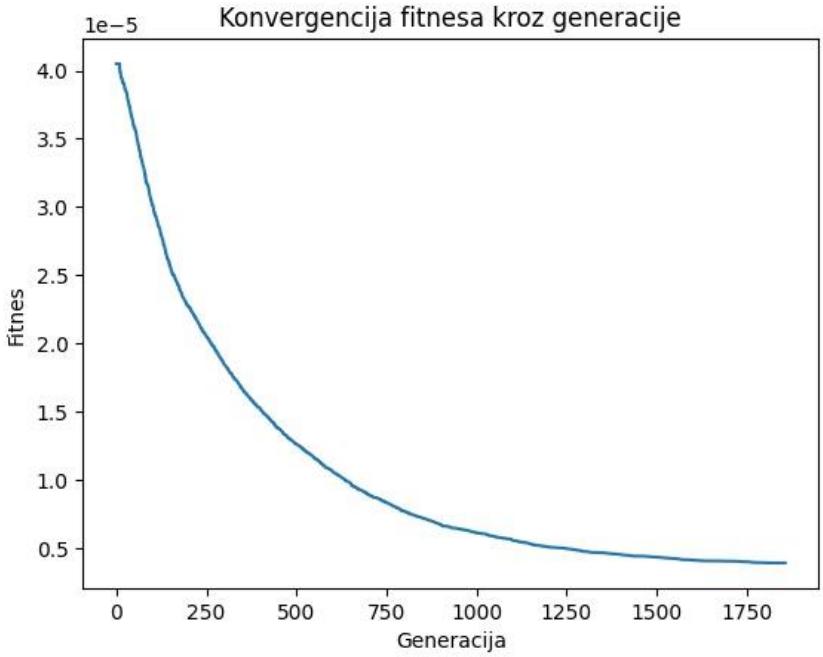
U ovom potpoglavlju prikazati ćemo i prokomentirati rezultate pokusa evolucije konstrukcije pod tlačnim opterećenjem. Početni oblik konstrukcije(slika 17.) je opterećen s koncetriranim opterećenjem iznos 1000 N usmjernim prema dolje, a oslonjena je na zglobne oslonce označene s trokutima. Plava točka je referentna točka čiji pomak mjerimo i uračunavamo u funkciju fitnes. Masa konstrukcije je određena ukupnim zbrojem duljina svakog pojedinog člana i kao takva je navedena bezdimenzijski u tablici parametra i rezultata pokusa.



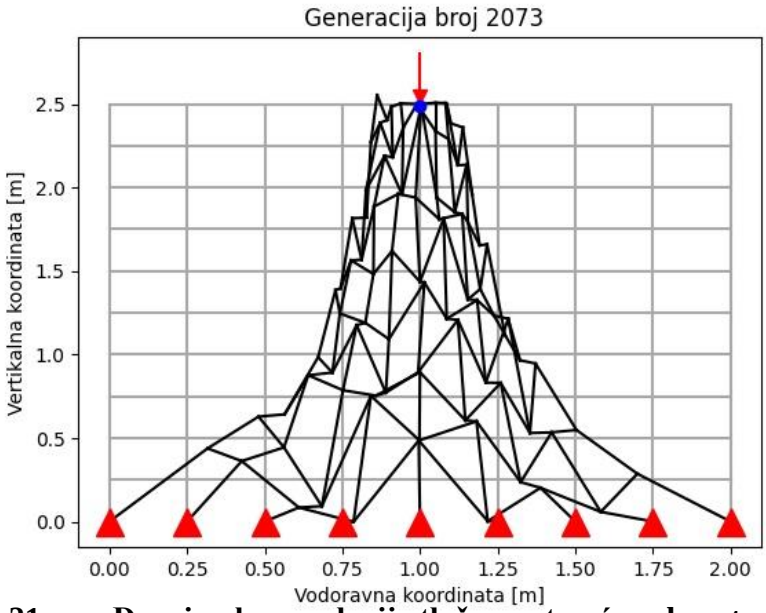
Slika 18. Početni oblik tlačno opterećene konstrukcije



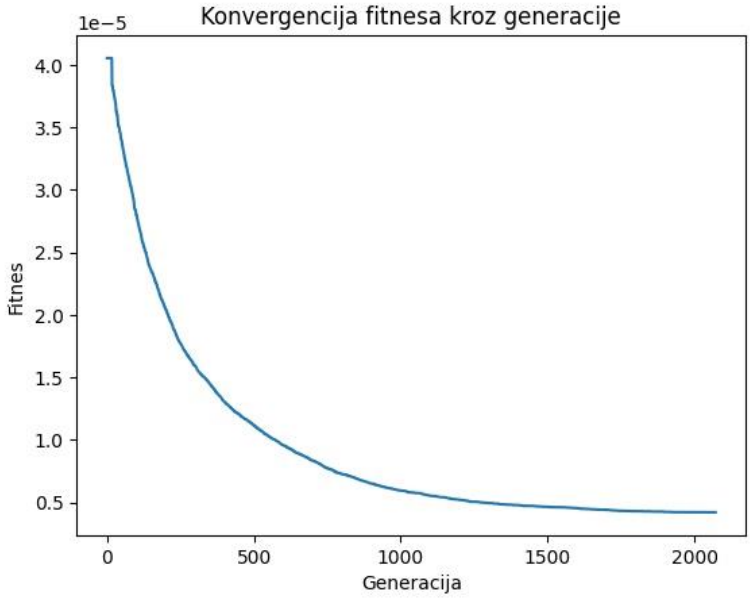
Slika 19. Prvi pokus evolucije tlačno opterećene konstrukcije



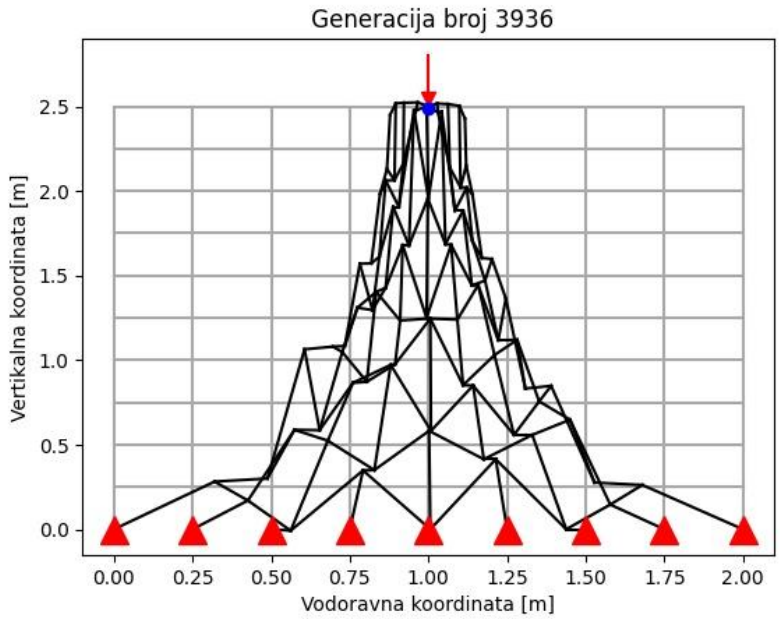
Slika 20. Fitnes kroz generacije za prvi pokus evolucije tlačno opterećene konstrukcije



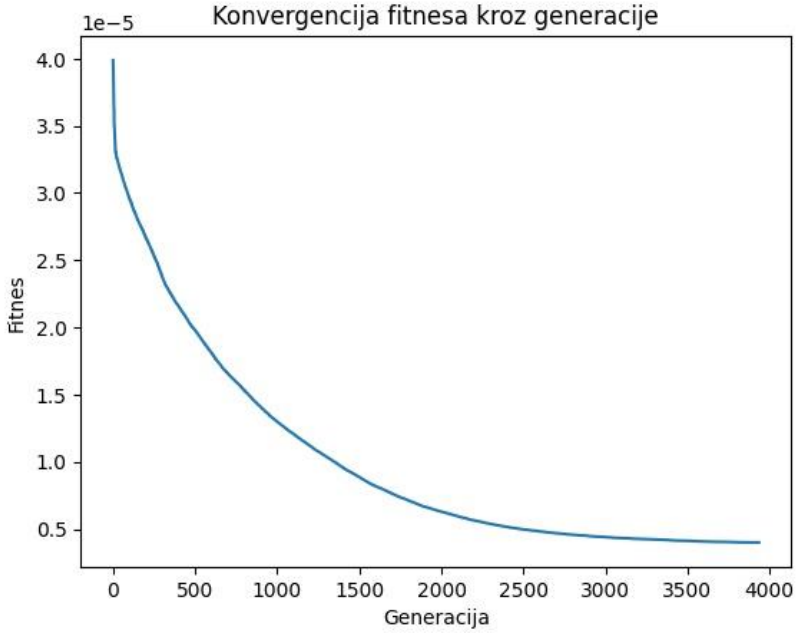
Slika 21. Drugi pokus evolucije tlačno opterećene konstrukcije



Slika 22. Fitnes kroz generacije za drugi pokus evolucije tlačno opterećene konstrukcije



Slika 23. Treći pokus evolucije tlačno opterećene konstrukcije



Slika 24. Fitnes kroz generacije za treći pokus evolucije tlačno opterećene konstrukcije

Pokus	roditelji	elitni	Roditelji za potomstvo	Iznos mutacije	Broj mutacija čvorova	Generacija do rješenja	Progib referentne točke	Mas	Fitnes
Početna konstrukcija	/	/	/	/	/	/	$8{,}17889783*10^{-8}$	5	0,0000408944891
1	120	1	115	0.004	40	1858	$3{,}21986966*10^{-8}$	1,2263119	0,000003948564
2	120	1	115	0.004	20	2073	$3{,}50195927*10^{-8}$	1,201871	0.000004208906802
3	120	1	115	0.002	20	3936	$3{,}413216099967*10^{-8}$	1,1718599	0,000003999811

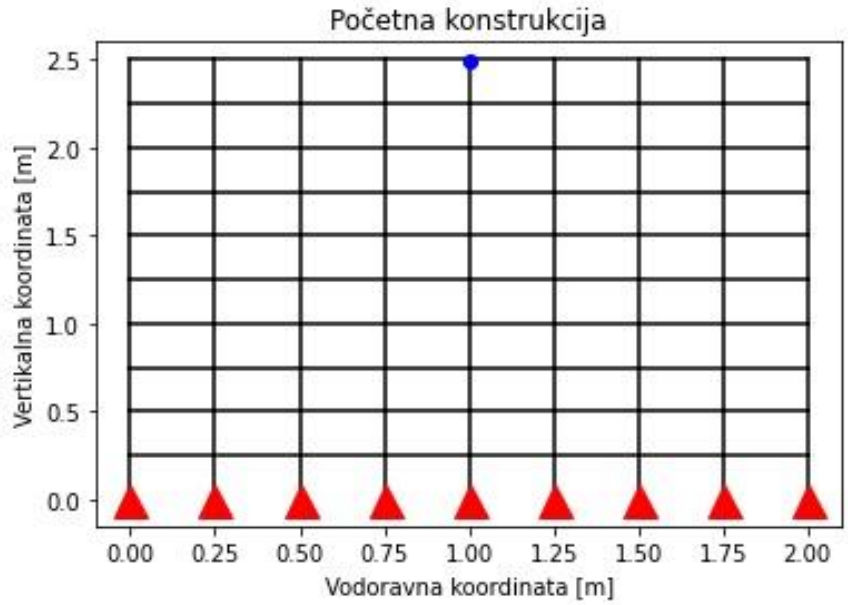
Tablica 8. Parametri i rezultati evolucija tlačno opterećene konstrukcije

Pri evoluciji tlačno opterećene konstrukcije pokus broj jedan poprima najbolji iznos fitnesa, a pošto ima najmanji iznos progiba referentne točke možemo zaključiti da je ta konstrukcija najkruća u pokusima tlačno opterećene konstrukcije. Konstrukcija poprima piramidalni oblik što bi bilo analogno širenju zamišljenih krivulja naprezanja u materijalu koji tlačimo. Dijelovi konstrukcije koji nisu opterećeni bivaju eliminirani iz genoma jedinki i ta konstrukcija postepeno poprima piramidalni oblik, a kasnije oblik tijela s krivuljama višeg reda. U ovom pokusu tlačno opterećene konstrukcije sva tri primjera poprimaju sličan oblik no imaju bitne razlike unutarnje strukture konstrukcije, njezine simetričnosti i u iznosu fitnesa, masa i progiba referentne točke. Isto tako jasno se vidi da je konstrukcija u trećem pokusu manjih dimenzija od prve dvije, a i njezina masa je manja naspram prva dva pokusa i početne konstrukcije. Masa, progib i fitnes sva tri pokusa su bolji naspram osnovne konstrukcije, od kojih je masa najznačajnije od svih rezultantnih vrijednosti smanjena evolucijskim algoritmom. Konstrukcija pokus broj tri postiže najbolji iznos mase no ne i najmanji iznos progiba točke. Simetričnost i izgled strukture konstrukcije pokus broj tri je najbolja no uz skoro 2000 generacija više do rješenja evolucijskog algoritma naspram pokusa broj jedan koji ima najmanji iznos fitnesa. Krivulje fitnesa kroz generacije su skoro jednakog izgleda u sva tri pokus. Fitnes i progib pokusa tri je istog reda veličine kao i u pokusima broj jedan i dva te se postavlja pitanje što mi zapravo trebamo? Trebamo li konstrukciju simetričnijeg i pravilnijeg izgleda? Trebamo li konstrukciju manje mase ili konstrukciju čiji je progib manji, odnosno konstrukciju koja ima veću krutost?

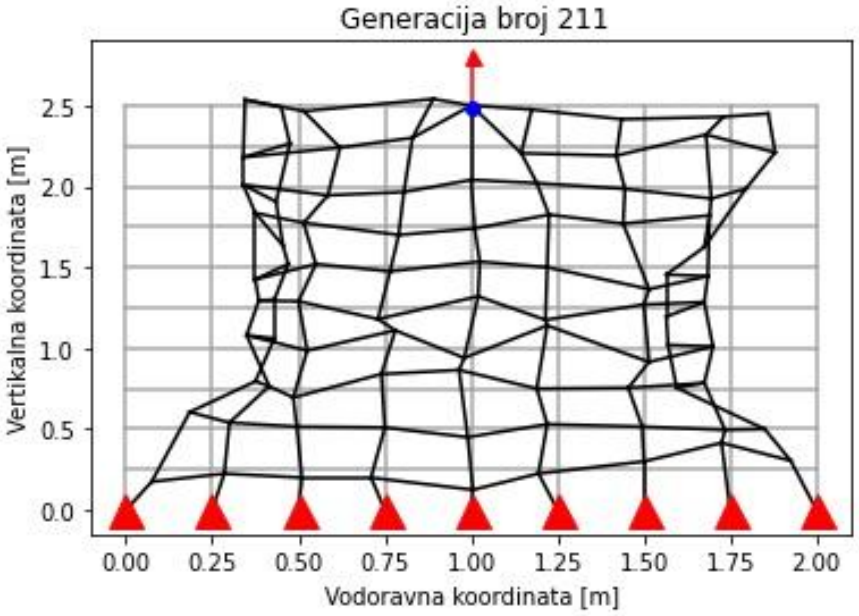
4.4.3. Evolucija konstrukcije pod vlačnim opterećenjem

U ovom potpoglavlju prikazati ćemo i prokomentirati rezultate pokusa evolucije konstrukcije pod tlačnim opterećenjem. Početni oblik konstrukcije(slika 17.) je opterećen s koncetriranim

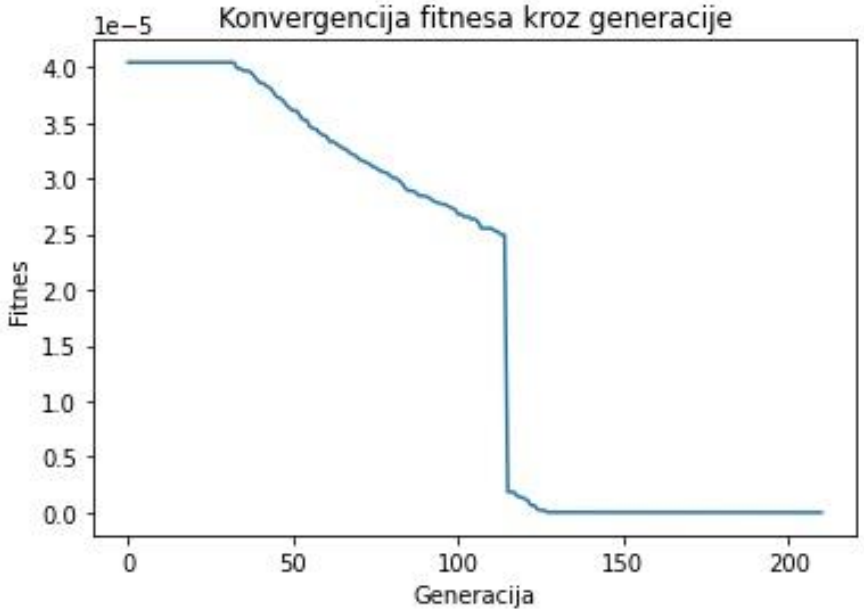
opterećenjem iznos 1000 N usmjernim prema gore, a oslonjena je na zglobne oslonce označene s trokutima. Plava točka je referentna točka čiji pomak mjerimo i uračunavamo u funkciju fitnesa. Masa konstrukcije je određena ukupnim zbrojem duljina svakog pojedinog člana i kao takva je navedena bezdimenzijski u tablici parametra i rezultata pokusa.



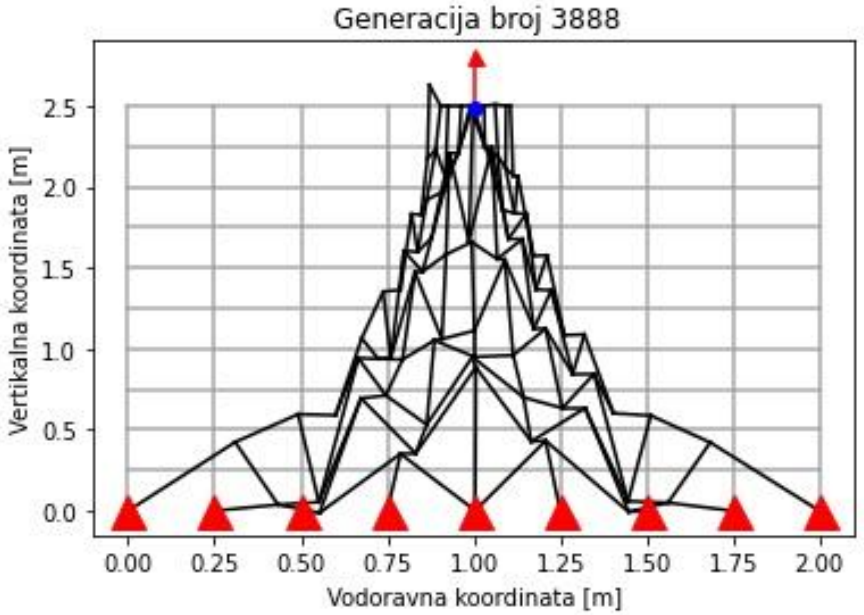
Slika 25. Početni oblik vlačno opterećene konstrukcije



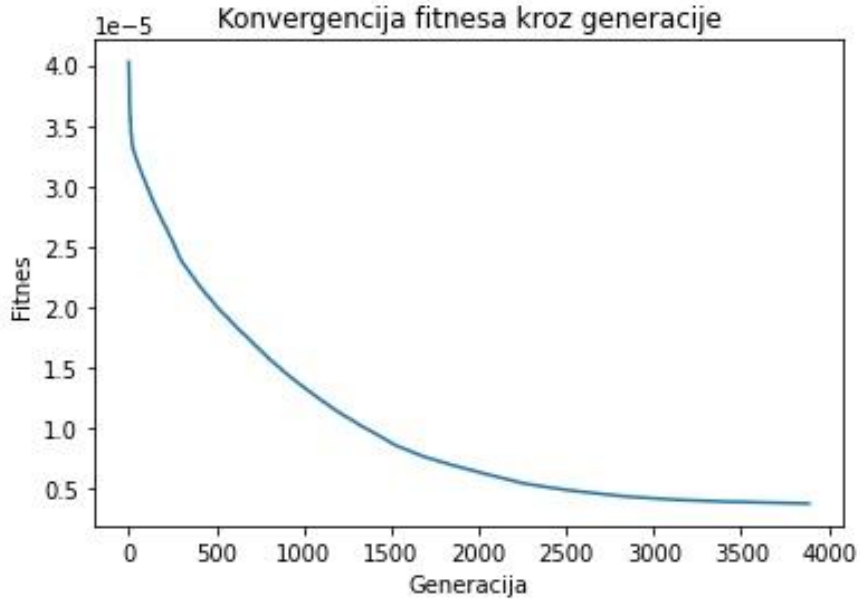
Slika 26. Prvi pokus evolucije vlačno opterećene konstrukcije



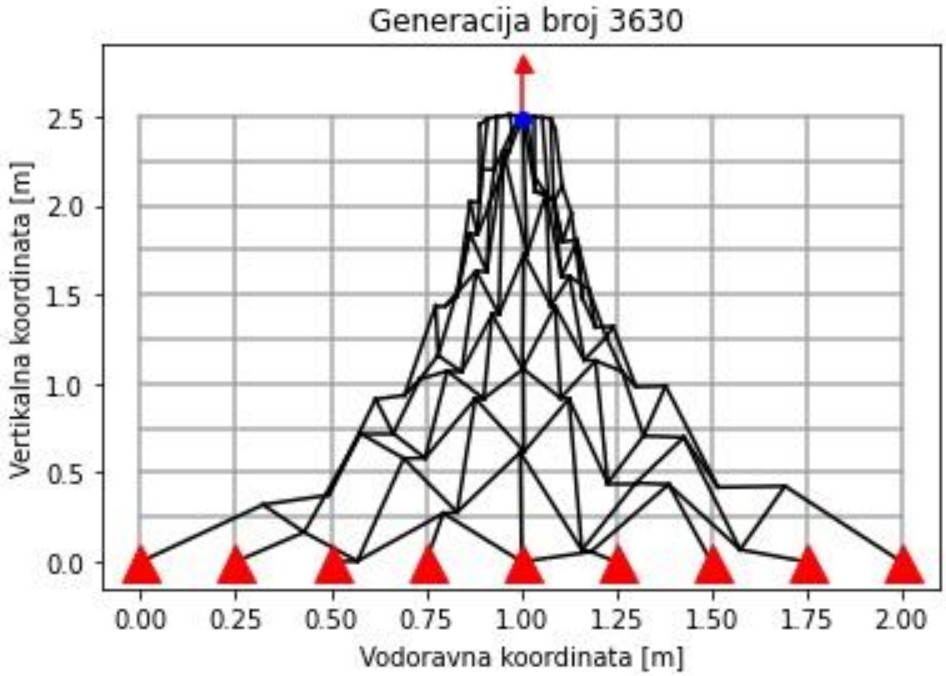
Slika 27. Fitnes kroz generacije za prvi pokus evolucije vlačno opterećene konstrukcije



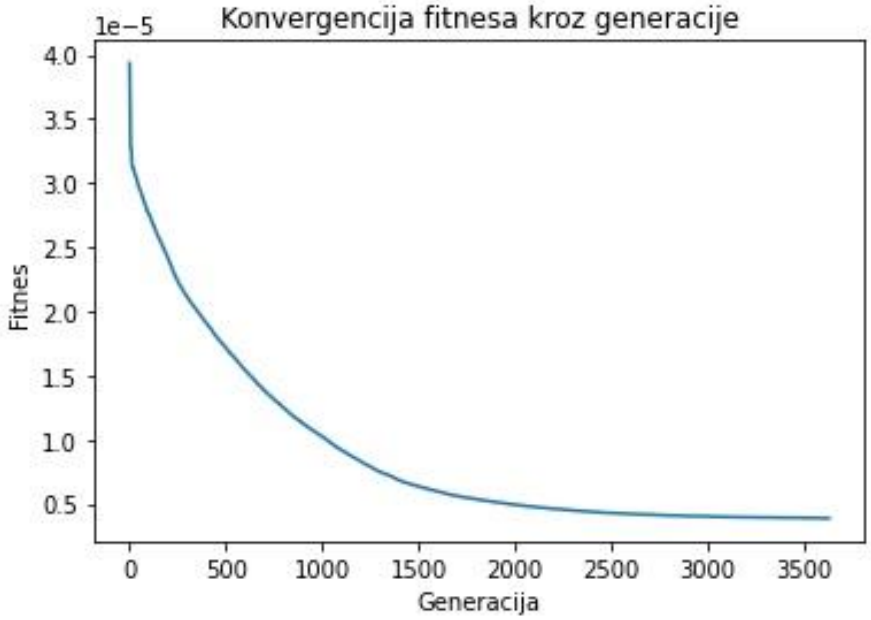
Slika 28. Drugi pokus evolucije vlačno opterećene konstrukcije



Slika 29. Fitnes kroz generacije za drugi pokus evolucije vlačno opterećene konstrukcije



Slika 30. Treći pokus evolucije vlačno opterećene konstrukcije



Slika 31. Fitnes kroz generacije za treći pokus evolucije vlačno opterećene konstrukcije

Pokus	roditelji	elitni	Roditelji za potomstvo	Iznos mutacije	Broj mutacija čvorova	Generacija do rješenja	Progib referentneto čke	Mas	Fitnes
Početna konstrukcija	/	/	/	/	/	/	8,1788978346*10^{-8}	5	0,0000408944891
1	110	1	100	0.005	50	211	6,39341382*10^{-13}	3,75375	2,39992771*10^{-10}
2	110	1	100	0.002	20	3888	3,210177*10^{-8}	1,182651	0,0000037965227
3	110	1	100	0.002	35	3630	3,3515971*10^{-8}	1,1555239	0,00000387285097

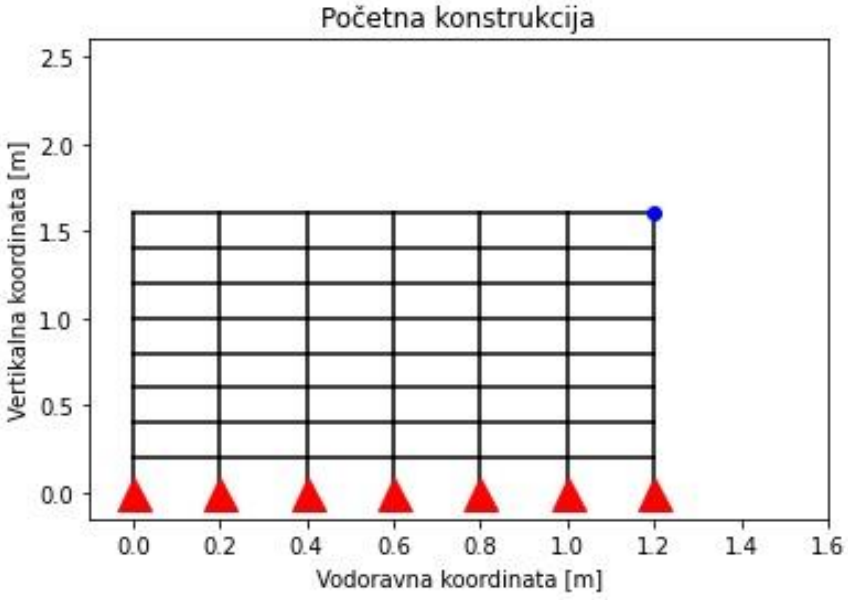
Tablica 9. Parametri evolucija vlačno opterećene konstrukcije

U pokusu evolucije vlačno opterećene konstrukcije započeli smo s parametrima 110 roditelja od kojih će njih 100 od svake generacije ići u borbe za preživljavanje i širenja gena u sljedeću generaciju. Iznos mutacije je 0,005 a broj mutacija čvorova potomaka je 50. Na slici(Slika 25.) vidimo je konstrukcija poprimila neki deformirani oblik i tu završila nakon 211 generacija. Ako

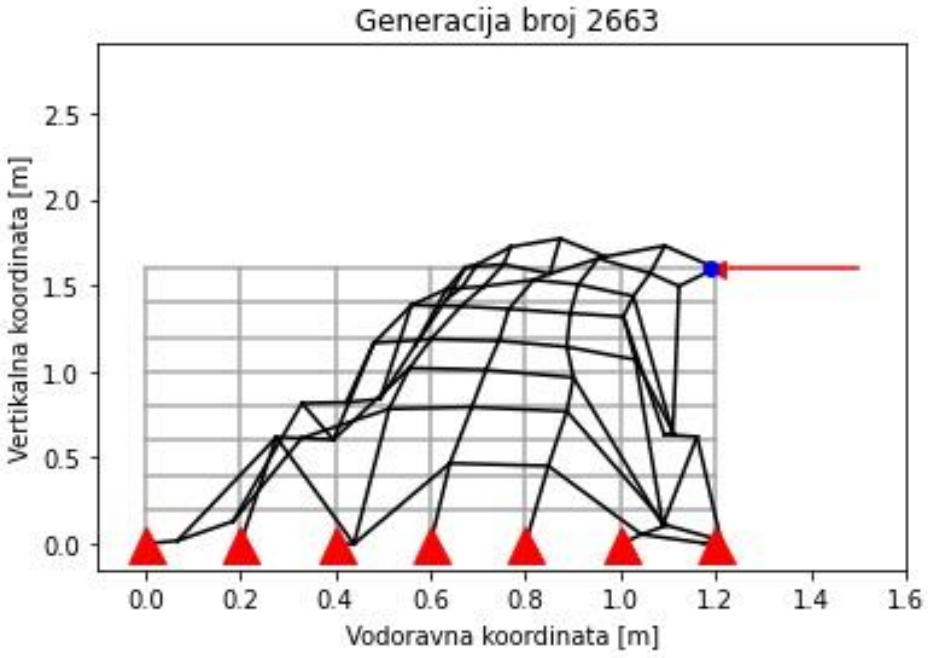
ju usporedimo s pokusom broj 2 (Slika 27.) možemo zaključiti da će populacija konvergirati ka nekom „ljepšem rješenju“ onda kad parametre izoštrimo, odnosno kad smanjimo iznose mutacije i broj mutacija u pojedinim jedinkama. Na taj način dovoljno istražujemo genetsku okolinu s nasumičnim mutacijama, a da u isto vrijeme ne preskočimo neku potencijalnu kombinaciju gena jer su mutacije bile prevelike. Jedina čudna stvar je manji iznos fitnes i progiba referentne točke naspram druga dva pokus uz značajnije veću masu konstrukcije što se može pripisati stohastičkoj prirodi evolucijskih algoritama. Ako uspoređujemo pokus broj dva i broj 3 možemo vidjeti da je fitnes istog reda iznos gdje je fitnes pokus broj dva malo bolji uz manji progib referentne točke i veću masu pa se tu postavlja pitanje trebamo li konstrukciju s manjom masom ili manjim progibom? Krivulje fitnesa kroz generacije pokusa dva i tri prate sličan trend, a krivulja fitnes kroz generacije pokus broj jedan ne prati trend kao u sljedeća dva pokusa što se može pripisati niskom broju generacija pri kojem se evolucija prekinula.

4.4.4. Evolucija konstrukcije pod kosim opterećenjem

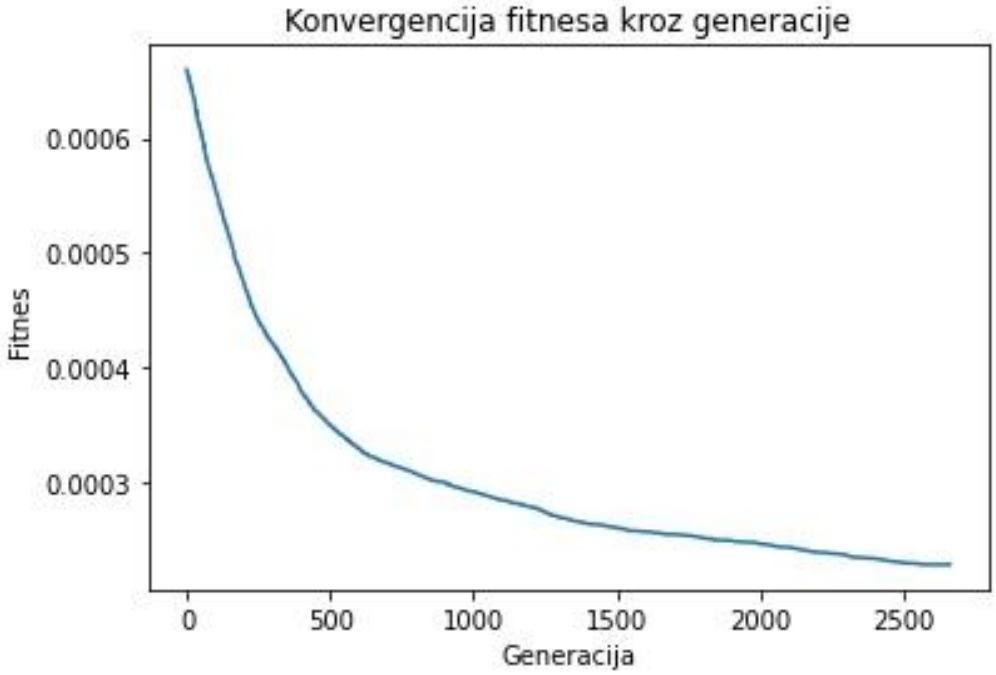
U ovom potpoglavlju prikazati ćemo i prokomentirati rezultate pokus evolucije konstrukcije pod kosim opterećenjem. Početni oblik konstrukcije(slika 31.) je opterećen s koncetriranim opterećenjem iznos 1000 N usmjernim prema lijevo, a oslonjena je na zglobne oslonce označene s trokutima. Plava točka je referentna točka čiji pomak mjerimo i uračunavamo u funkciju fitnesa. Masa konstrukcije je određena ukupnim zbrojem duljina svakog pojedinog člana i kao takva je navedena bezdimenzijski u tablici parametra i rezultata pokusa.



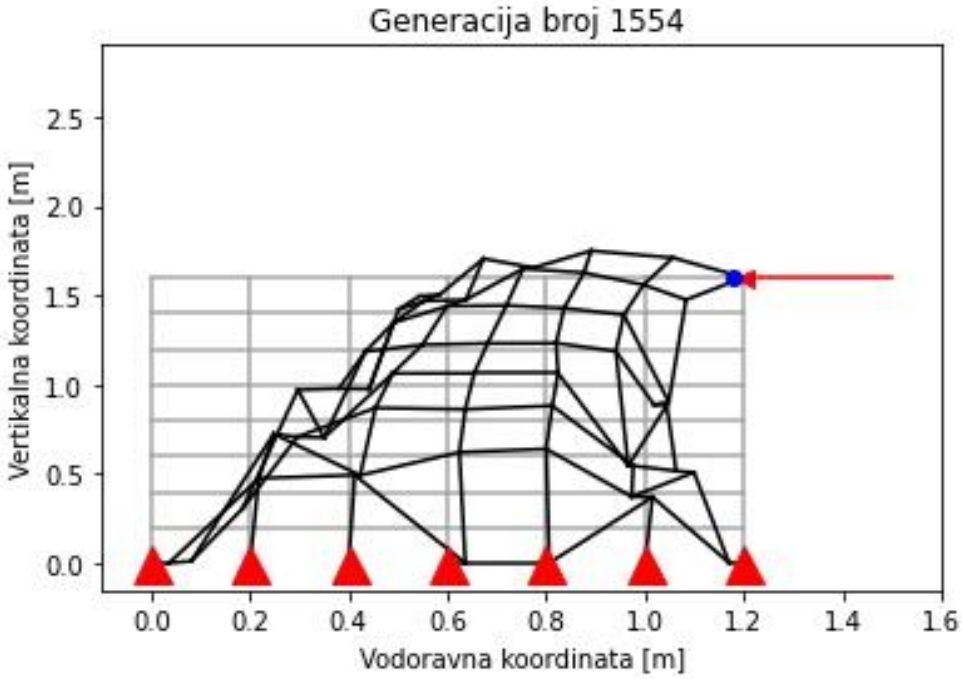
Slika 32. Početni oblik koso opterećene konstrukcije



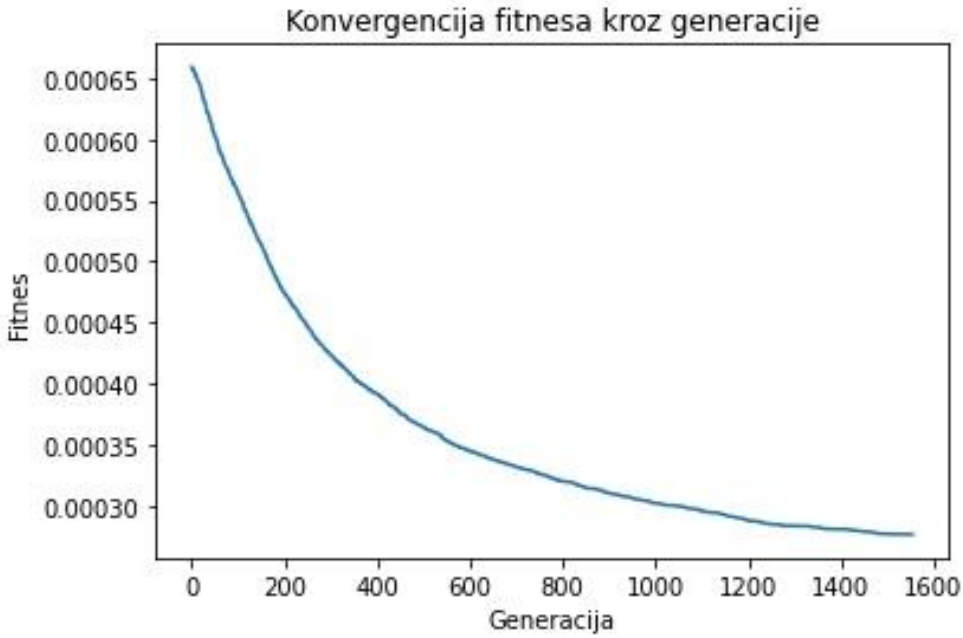
Slika 33. Prvi pokus evolucije koso opterećene konstrukcije



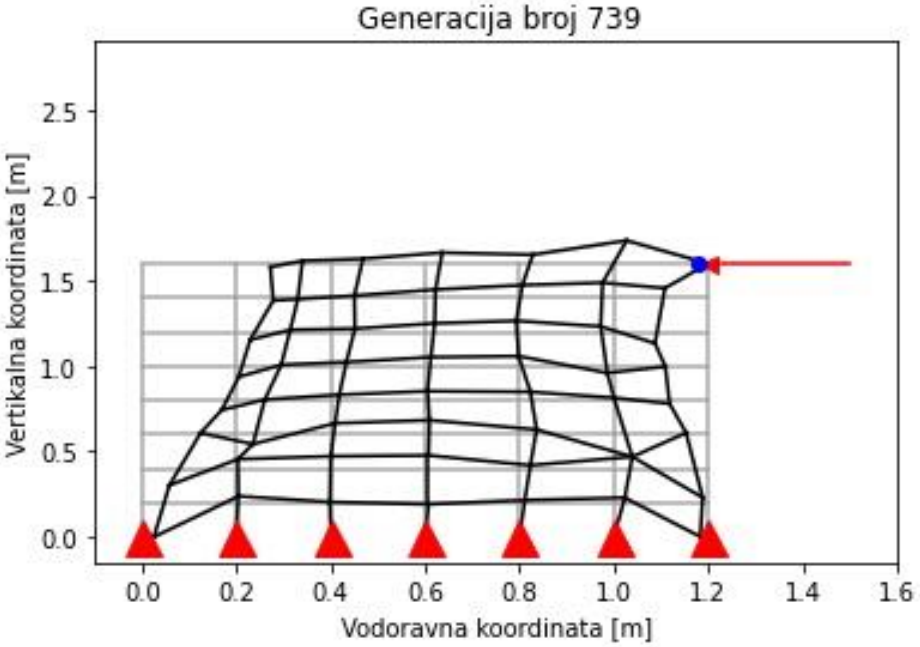
Slika 34. Fitnes kroz generacije za prvi pokus evolucije koso opterećene konstrukcije



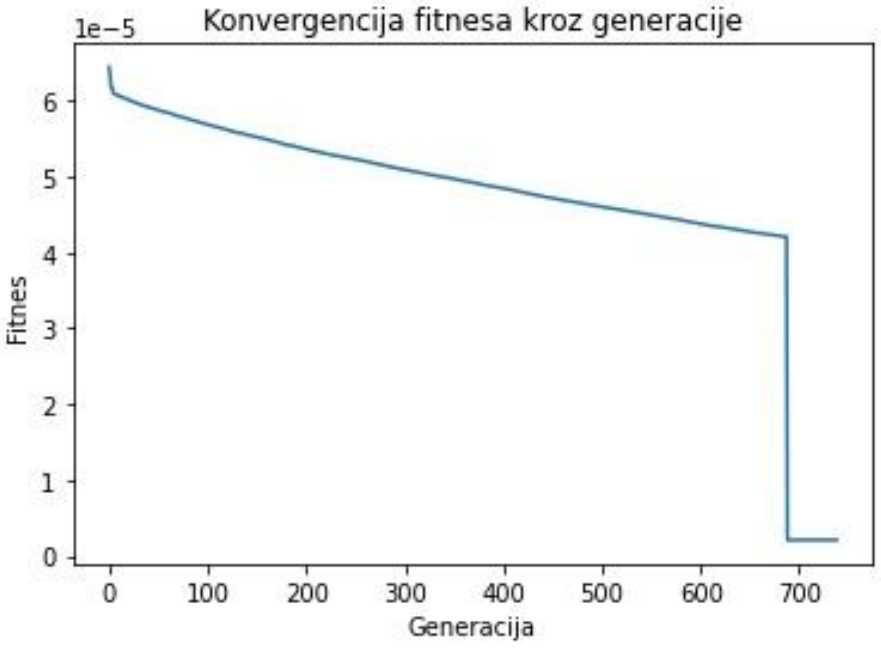
Slika 35. Drugi pokus evolucije koso opterećene konstrukcije



Slika 36. Fitnes kroz generacije za drugi pokus evolucije koso opterećene konstrukcije



Slika 37. Treći pokus evolucije koso opterećene konstrukcije



Slika 38. Fitnes kroz generacije za treći pokus evolucije koso opterećene konstrukcije

Pokus	roditelji	elitni	Roditelji za potomstvo	Iznos mutacije	Broj mutacija čvorova	Generacija do rješenja	Progib referentne točke	mas	Fitnes
Početna konstrukcija	/	/	/	/	/	/	$3{,}459438512 * 10^{-7}$	1,92000015	0,000066421
1	140	1	135	0.003	20	2663	$5{,}7243212202 * 10^{-8}$	0,975227075835	0,00000558251306
2	116	1	104	0.003	20	1554	$8{,}2525582633 87 * 10^{-8}$	1,02984609	0,00000849886493
3	96	1	88	0.003	20	739	$7{,}9842285027 * 10^{-8}$	1,386427121	0,00001106955093

Tablica 10. Parametri evolucija koso opterećene konstrukcije

U pokusu evolucije koso opterećene konstrukcije mijenjali smo parametre veličine populacije. Pokus broj jedan(Slika 32.) možemo si predočiti kao primjer toka sile u nekoj konstrukciji. Evolucija ove konstrukcije je potisnula ovu konstrukciju da članove iz gornjeg lijevog kuta sve više približava sredini konstrukcije gdje se usmjerava tok opterećenja sile. Konstrukcija poprima blago zakrivljeni oblik i kralježnicu koja prenosi opterećenje. Krivulja fitnes kroz generacije konstantno konvergira prema minimumu(Slika 33.) uz značajno strmiju konvergenciju u prvoj polovici od ukupno generacija. U pokusu broj 2 smanjenjem populacije dobivamo sličan ishod kao u pokusu broj jedan uz veći iznos fitnes, veću masu i veći progib. Pokus 2 staje nakon 1554 generacije što je skoro 1000 generacija manje od pokusa broj jedan no iznos fitnes ostaje veći a isto tako je krivulja fitnesa kroz generacije(Slika 35.) manje strma nego kako kod prvog pokusa. Za treći pokus postavljamo populaciju na 96 zbog čega u daljnjim generacijama imamo manju genetsku raznolikost i kao takva populacija konvergira u lokalnom minimumu bez značajne formacije kralježnice u evoluciji konstrukcije. Pokus kao takav konvergira nakon 739 generacije te se vidi značajna razlika naspram prva dva pokusa. Masa konstrukcija u svim pokusima se smanjuje, a isto tako i progib referentnih točaka što potvrđuje rad evolucijskog algoritma.

5. ZAKLJUČAK

U ovom radu uspješno sam samostalno napisao algoritam metode konačnih elemenata i povezao ga s evolucijskim algoritmom. Odrađeni pokusi su bili uspješni te njihov konačan rezultat jasno pokazuje da ovaj algoritam radi. Prvobitno je bio plan analizirati evoluciju trodimenzionalne konstrukcije no zbog vremenskog ograničenja za izradu diplomskog rada i vremena potrebnog za pokuse taj se eksperiment može ostaviti za nekoga drugog. Moja želja je da ovaj algoritma napisan u python kodu bude referenca za druge studente koji žele proširiti ovaj problem na više dimenzija i složenije probleme kako bi bolje razumjeli problematiku metode konačnih elemenata i evolucijskih algoritama. Nadodao bih da za efikasnije proračune konstrukcija je nužno ovladati okružjem za paralelno računanje jer cijena procesorske snage raste sa svakim stupnjem slobode, a time i vrijeme potrebno za analizu.

LITERATURA

[1] Kraut, B.: Strojarski priručnik, Tehnička knjiga Zagreb, 1970.

[2] Decker, K. H.: Elementi strojeva, Tehnička knjiga Zagreb, 1975.

[3] Herold, Z.: Računalna i inženjerska grafika, Zagreb, 2003.

[4] https://hr.wikipedia.org/wiki/Evolucijski_algoritam datum pristupa: 01. 06. 2022.

[5] https://en.wikipedia.org/wiki/Evolutionary_computation datum pristupa: 01. 07. 2022.

[6] https://hr.wikipedia.org/wiki/Evolucija datum pristupa: 03. 07. 2022.

[7] https://www.degreetutors.com/direct-stiffness-method/ datum pristupa: 02. 07. 2022.

[8] https://courses.degreetutors.com/view/courses/finite-element-analysis-of-3d-structures-using-python datum pristupa: 02. 07. 2022.

[9] https://hr.wikipedia.org/wiki/Hookeov_zakon datum pristupa: 02. 07. 2022.

[10] https://blog.nwf.org/2014/12/winter-white-wildlife/ datum pristupa: 06. 07. 2022.

[11] Ćurković, Petar. Optimization of Generatively Encoded Multi- Material Lattice Structures for Desired Deformation Behavior // Symmetry, 13 (2021), 2; 293, 14 doi:10.3390/sym13020293

[12] Ćurković, Petar; Čubrić, Goran. FUSED DEPOSITION MODELLING FOR 3D PRINTING OF SOFT ANTHROPOMORPHIC ACTUATORS // International journal of simulation modelling, 20 (2021), 2; 303-314 doi:10.2507/IJSIMM20-2-560

[13] Ćurković, Petar; Jambrečić, Antonio. Improving structural design of soft actuators using finite element method analysis // Interdisciplinary description of complex systems, 18 (2020), 4; 490-500 doi:10.7906/indecs.18.4.8

[14] Ćurković, Petar; Čehulić, Lovro. Diversity maintenance for efficient robot path planning // Applied Sciences-Basel, 10 (2020), 5; 10051721, 15 doi:10.3390/app10051721

[15] Introduction To evokutionary Computing. A.E. Eiben, J.E. Smith, Springer Berlin, 2008.

PRILOZI

I. CD-R disc

II. Python kod

```
################### Solver 3D konstrukcije ####################

#%%
#_____________________ 1.0 Zavisnosti _____________________________

# Nužne su za funkcije korištene u kodu
# importamo libraryse(biblioteke) i dependencyse



from ast import Break
from bdb import Breakpoint
from logging import exception
import math #.....Matematičke funkcije
import copy #.....Kopiranje objekta
import csv
from tkinter import W, X
from turtle import rt
from jinja2 import pass_context
from matplotlib.font_manager import json_load #.....Izvoz(export)
podataka u obliku CSV datoteke
import numpy as np #.....Numpy za rad s matricama(array)
from glob import glob #.....Za provjeru postoje li određeni podaci
import matplotlib.colors #.....Funkcije za boje
import ipywidgets as widgets #.....Funkcije za kontrolu widgeta
from numpy import NaN, genfromtxt #.....Uvoz podataka iz csv fajlova
import matplotlib.pyplot as plt #.....Funkcije plotanja
from mpl_toolkits.mplot3d import Axes3D #.....Funkcije 3D plotanja
from mpl_toolkits.mplot3d.art3d import Poly3DCollection #..Plotanje 3D patches
import winsound
duration = 2500  # milliseconds
freq = 440  # Hz

import random #.....Random funkcije

import pickle #sejvanje podataka

#_____________________ 2.0 Automatski uvoz podatak o strukturi i
opterećenju_____________________

#uvoz .csv datoteka koje definiraju strukturu i opterećenja generirane u blenderu

#NUŽNI UVOZ: KOORDINATE ČVOROVA
if glob('D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoVertices.csv'):
    cvorovi = genfromtxt('D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoVertices.csv', delimiter=',')
    print('1. □ KosoVertices.csv uvezeni!')
else:
```

```
    print('1. ⬢ STOP: KosoVertices.csv nisu pronađeni!')

#NUŽNI UVOZ: ODREĐENJA ČLANOVA
if glob('D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoEdges.csv'):
    clanovi = genfromtxt('D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoEdges.csv', delimiter=',')
    clanovi = np.int_(clanovi)  #prebaci iz float u int
    print('2. □ KosoEdges.csv uvezeni!')
else:
    print('2. ⬢ STOP: KosoEdges.csv nisu pronađeni!')

#NUŽNI UVOZ: KOORDINATE ČVOROVA opt
if glob('D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoVertexiOpt.csv'):
    cvoroviOpt = genfromtxt('D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoVertexiOpt.csv', delimiter=',')

    print('3. □ KosoVertexiOpt.csv uvezeni!')
else:
    print('3. ⬢ STOP: KosoVertexiOpt.csv nisu pronađeni!')

#NUŽNI UVOZ: CVOROVI POTPORE
if glob('D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoRestraint-
Nodes.csv'):
    potporniCvorovi = genfromtxt('D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoRestraint-Nodes.csv', delimiter=',')
    print('4. □ KosoRestraint-Nodes.csv uvezeni!')
else:
    print('4. ⬢ STOP: KosoRestraint-Nodes.csv nisu pronađeni!')

#NUŽNI UVOZ: POTPORA DoF (index pocinje s 1) [potrebni različiti uvozi za različite
potpore]
if glob('D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoRestraint-
DoF.csv'):
    podaciPotpore = genfromtxt('D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoRestraint-DoF.csv', delimiter=',')
    podaciPotpore = np.int_(podaciPotpore) #prebaci određenja clanova iz floata u int
    flatData = podaciPotpore.flatten() #DoF data-->listu liste prebaci u jednu listu [[1,2],[3,4]]-
-->[1,2,3,4]
    podaciPotpore = flatData[np.nonzero(flatData)[0]].tolist() #makni nule iz liste
    print('5. □ KosoRestraint-DoF.csv uvezeni!')
else:
    print('5. ⬢ STOP: KosoRestraint-DoF.csv nisu pronađeni!')

#NEOBAVEZAN UVOZ: PODACI KOORDINATA SILA
if glob('D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoForce-
Data.csv'):
    podaciKoordinataSila = genfromtxt('D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoForce-Data.csv', delimiter=',')
```

```
    podaciKoordinataSila = np.int_(podaciKoordinataSila) #prebaci iz float u int
    print('6.  KosoForce-Data.csv uvezeni!')

    brSila = len(np.array(podaciKoordinataSila.shape))
    if brSila<2:
        podaciKoordinataSila = np.array([podaciKoordinataSila])

else:
    podaciKoordinataSila = []
    print('6. ⚠ KosoForce-Data.csv nisu pronađeni!')


print("broj čvorova za opt ",len(cvoroviOpt))
print("broj čvorova ", len(cvorovi))


#_____ 3.0 Ručni unos podataka
_____

#unos konstanti, dodijeljivanje iznos koncetriranih sila te njihovog smjera

#KONSTANTE
A_beam = 0.027385 #(m^2)
#č0561 550 N/mm2 ....55x10e7 N/m2
dopustenaSila = A_beam*55*10**7

ModElas = 200*10**9  #(N/m^2)
ModSmic = 200*10**9  #(N/m^2)
Izz = 250*10**-6  #(m^4)
Iyy = 100*10**-6  #(m^4)
Ip = (Izz+Iyy)  #(m^4)

#DODIJELI KONCETRIRANA OPTEREĆENJA
P = 1000 #(N) Iznos koncetriranog opterećenja (predznak određuje smjer)
koncOptOs = 'x' # globalna os s kojom je paralelna koncetrirana sila

#_____ definiranje parametra za EA _____
##########################################################################
#####
##########################################################################
#####
##########################################################################
#####
#faktori koji su potrebni za inicijalizaciju, mutaciju i ostalo
#faktori koji su potrebni za inicijalizaciju, mutaciju i ostalo
brojRoditelja = 96  #broj jedinki koje će se vrtiti u svakoj generacinji

brojKoraka = 20000 # broj generacija do rješenja

elitniRoditelji = 1
```

```
brojDjece = brojRoditelja-elitniRoditelji + 1
brojNajboljihRoditeljaOdKojihPravimoDjecu = 88


vodoravnaGranicaModela = 2
okomitaGranicaModela = 2
generacijaKojuPlotamo = 50

gornjaGranicaGena = 0.1
donjaGranicaGena = 0.9

cvorPracenja = 17

iznosPerturbacije = 0.005
brojPertubacijaČvorova = 35
iznosMutacije = 0.001
brojMutacijaČvorova = 22

fitPrije = 100
treshold = 0.000000000001
granicaGeneracijaBezPomaka = 50

#_____________________ 5.0 Napravi niz površina clanova_____________________

Povrsina = A_beam #Inicijaliziraj sve površine članova u površine greda

# Plotanje konstrukcije za potvrdu unos
def plotajkonstrukciju(label_offset=0.01,
                       xMargina = 1,
                       yMargina = 1,
                       zMargina = 1,
                       elevacija=10,
                       rotacija=270):

    #Kod za plotanje
    fig = plt.figure()
    axes = fig.add_axes([0.1,0.1,3,3],projection='3d') #Predodredi 3D plotanje
    axes.view_init(elevacija, rotacija) #postavi kut gledanja plota

    #postavi label offset udaljenost
    dx = label_offset
    dy = label_offset
    dz = label_offset

    #Osiguraj mjesta oko strukture
    x_margina = xMargina
    y_margina = yMargina
    z_margina = zMargina
```

```
    #Plotaj clanove
    for n, cln in enumerate(clanovi):
        cvor_i = cln[0] #broj čvora za čvor i ovog člana
        cvor_j = cln[1] #broj čvora za čvor j ovog člana

        ix = cvorovi[cvor_i-1,0] #x-koord čvora i ovog člana
        iy = cvorovi[cvor_i-1,1] #y-koord čvora i ovog člana
        iz = cvorovi[cvor_i-1,2] #z-koord čvora i ovog člana
        jx = cvorovi[cvor_j-1,0] #x-koord čvora j ovog člana
        jy = cvorovi[cvor_j-1,1] #y-koord čvora j ovog člana
        jz = cvorovi[cvor_j-1,2] #z-koord čvora j ovog člana

        axes.plot3D([ix,jx],[iy,jy],[iz,jz],'b')

    #Plot cvorovi
    for n, cvor in enumerate(cvorovi):
        axes.plot3D([cvor[0]], [cvor[1]], [cvor[2]], 'bo', ms=6) #Plot 3D cvor
        label = str(n+1)
        axes.text(cvor[0]+dx, cvor[1]+dy, cvor[2]+dz, label, fontsize=16)

    #Set axis limits
    maxX = cvorovi.max(0)[0]
    maxY = cvorovi.max(0)[1]
    maxZ = cvorovi.max(0)[2]
    minX = cvorovi.min(0)[0]
    minY = cvorovi.min(0)[1]
    minZ = cvorovi.min(0)[2]
    axes.set_xlim([minX-x_margina, maxX+x_margina])
    axes.set_ylim([minY-y_margina, maxY+y_margina])
    axes.set_zlim([0, maxZ+z_margina])

    axes.set_xlabel('X-koordinata (m)')
    axes.set_ylabel('Y-koordinata (m)')
    axes.set_zlabel('Z-koordinata (m)')
    axes.set_title('Konstrukcija za analizu')
    axes.grid()

widgets.interact(plotajkonstrukciju,
                 label_offset=(0.01,0.1,0.01),
                 xMargina=(0.25, 100, 0.25),
                 yMargina=(0.25, 100, 0.25),
                 zMargina=(0.5, 3, 0.25),
                 elevacija=(0,360,10),
                 rotacija=(0,360,10))

plt.show()

def napraviLokalniRF(brClana,clanovi,cvorovi):
    """
    Za ne-vertikalne članove, lokalna x-y ploha je određena odmakom
```

```
    mid-pointa člana vertikalno za +1m  u pozitvnom globalnom z-smjeru

    Za vertikalne clanove, lokalna x-y ploha je određena odmakom
    mid-pointa člana za -1m u smjeru globalne negativne x-osi

    Za sve članove:
    - pretpostavljamo da je lokalna z os glavna os
    - pretpostavljamo da je lokalna y os sporedna os
    """

    indexClana = brClana -1 #index koji određuje člana u nizu članova
    cvor_i = clanovi[indexClana][0] #broj čvora za čvor i ovog člana
    cvor_j = clanovi[indexClana][1] #broj čvora za čvor j ovog člana


    ix = cvorovi[cvor_i-1,0] #x-koord čvora i ovog člana
    iy = cvorovi[cvor_i-1,1] #y-koord čvora i ovog člana
    iz = cvorovi[cvor_i-1,2] #z-koord čvora i ovog člana
    jx = cvorovi[cvor_j-1,0] #x-koord čvora j ovog člana
    jy = cvorovi[cvor_j-1,1] #y-koord čvora j ovog člana
    jz = cvorovi[cvor_j-1,2] #z-koord čvora j ovog člana

    #duljina clanova
    dx = jx-ix #x-komponenta od vektora duž člana
    dy = jy-iy #y-komponenta od vektora duž člana
    dz = jz-iz #z-komponenta od vektora duž člana
    length = math.sqrt(dx**2 + dy**2 + dz**2) #duljina člana

    if(abs(dx)<0.001 and abs(dy)<0.001):
        #Član je vertikalan-ofset delamo u negativnom globalnom x smjeru za defniciju lolane x-
y plohe
        i_offset = np.array([ix-1, iy, iz]) #Offset čvora i za 1m u negativnoj globalnoj x-osi
        j_offset = np.array([jx-1, jy, jz]) #Offset čvora j za 1m u negativnoj globalnoj x-osi
    else:
        #član je ne vertikal-ofset delamo u pozitivnoj globalnoj z osi za definiranje lokalne x-y
ravnine
        i_offset = np.array([ix, iy, iz+1]) #Offset čvora i za 1m u pozitivnoj globalnoj z-osi
        j_offset = np.array([jx, jy, jz+1]) #offset čvora j za 1m u pozitivnoj globalnoj z osi

    cvor_k = i_offset + 0.5*(j_offset-i_offset) #tocka u lokalnoj x-y plohi

    #lokalni x-vektor u globalnom referentnom okviru duž člana
    lokalni_x_vektor = cvorovi[cvor_j-1] - cvorovi[cvor_i-1] #Vektor duž lokalne x osi
    lokalni_x_jedinicniVektor = lokalni_x_vektor/length #Local unit vector defining local x-
axis

    #lokalni y vektor u globalnom referentnom okviru koristeći Gram-Schmidt proces
    vektor_u_ravnini = cvor_k-cvorovi[cvor_i-1] #Vektor u lokalnoj x-y ravnini
```

```
    lokalni_y_vektor = vektor_u_ravnini -
np.dot(vektor_u_ravnini,lokalni_x_jedinicniVektor)*lokalni_x_jedinicniVektor #lokalni y
vektor u globalnom RF (Gram-Schmidt)
    magY = math.sqrt(lokalni_y_vektor[0]**2 +
lokalni_y_vektor[1]**2+lokalni_y_vektor[2]**2) #duljina lokalnog y vektora
    lokalni_y_jedinicniVektor = lokalni_y_vektor/magY #lokalni jedinični vektor koji definira
y os

    #lokalni z vektor u globalnom RF pomoću cross produkta
    lokalni_z_jedinicniVektor = np.cross(lokalni_x_jedinicniVektor,
lokalni_y_jedinicniVektor) #lokalni kjedinični z vektor u globalnom RF

    #skombiniraj RF u standardnu rotacijsku matricu x,y,z => columns 1,2,3
    rotacijskamatrica = np.array([lokalni_x_jedinicniVektor, lokalni_y_jedinicniVektor,
lokalni_z_jedinicniVektor,]).T

    return [length, rotacijskamatrica]

#_____ Definiraj funkciju izračuna globalne matrice krutosti
članova_____

def izracunKg3Dbeam(brClana,lengths,TransformacijskeMatrice):
    #Izvuci pojedina svojstva članova

    A = Povrsina
    E = ModElas
    L = lengths[brClana-1]
    Iz = Izz
    Iy = Iyy
    G = ModSmic
    J = Ip

    Kl = np.zeros((12,12))
    #Red #1
    Kl[0,0] = E*A/L
    Kl[0,6] = -E*A/L
    #Red #2
    Kl[1,1] = 12*E*Iz/L**3
    Kl[1,5] = -6*E*Iz/L**2
    Kl[1,7] = -12*E*Iz/L**3
    Kl[1,11] = -6*E*Iz/L**2
    #Red 3
    Kl[2,2] = 12*E*Iy/L**3
    Kl[2,4] = 6*E*Iy/L**2
    Kl[2,8] = -12*E*Iy/L**3
    Kl[2,10] = 6*E*Iy/L**2
    #Red 4
    Kl[3,3] = G*J/L
    Kl[3,9] = -G*J/L
    #Red 5
```

```
    Kl[4,2] = 6*E*Iy/L**2
    Kl[4,4] = 4*E*Iy/L
    Kl[4,8] = -6*E*Iy/L**2
    Kl[4,10] = 2*E*Iy/L
    #Red 6
    Kl[5,1] = -6*E*Iz/L**2
    Kl[5,5] = 4*E*Iz/L
    Kl[5,7] = 6*E*Iz/L**2
    Kl[5,11] = 2*E*Iz/L
    #Red 7
    Kl[6,0] = -E*A/L
    Kl[6,6] = E*A/L
    #Red 8
    Kl[7,1] = -12*E*Iz/L**3
    Kl[7,5] = 6*E*Iz/L**2
    Kl[7,7] = 12*E*Iz/L**3
    Kl[7,11] = 6*E*Iz/L**2
    #Red 9
    Kl[8,2] = -12*E*Iy/L**3
    Kl[8,4] = -6*E*Iy/L**2
    Kl[8,8] = 12*E*Iy/L**3
    Kl[8,10] = -6*E*Iy/L**2
    #Red 10
    Kl[9,3] = -G*J/L
    Kl[9,9] = G*J/L
    #Red 11
    Kl[10,2] = 6*E*Iy/L**2
    Kl[10,4] = 2*E*Iy/L
    Kl[10,8] = -6*E*Iy/L**2
    Kl[10,10] = 4*E*Iy/L
    #Red 12
    Kl[11,1] = -6*E*Iz/L**2
    Kl[11,5] = 2*E*Iz/L
    Kl[11,7] = 6*E*Iz/L**2
    Kl[11,11] = 4*E*Iz/L

    #napravi cijelu transformacijsku matricu za ovaj element
    TM = np.zeros((12,12))
    T_repeat = TransformacijskeMatrice[brClana-1,:,:]
    TM[0:3,0:3] = T_repeat
    TM[3:6,3:6] = T_repeat
    TM[6:9,6:9] = T_repeat
    TM[9:12,9:12] = T_repeat

    #izračunaj globalu matricu krutosti elementa
    Kg = TM.T.dot(Kl).dot(TM)

    K11g = Kg[0:6,0:6]
    K12g = Kg[0:6,6:12]
    K21g = Kg[6:12,0:6]
```

```
    K22g = Kg[6:12,6:12]

    return [K11g, K12g, K21g, K22g]



def fitness(clanovi,cvorovi,cvorPracenja):
    #_____________________________ 9.0 Build member orientation reference
frames_____________________________________


    #odredi orijentaciju pojedinog člana

    rotacijskeMatrice = np.empty((len(clanovi),3,3)) #inicijaliziraj spremnik podataka

    lengths = np.zeros(len(clanovi))

    for n,cln in enumerate(clanovi):

        #izračunaj rotacijsku matricu koja definira orijentaciju članova
        [length,rotacijskaMatrica] = napraviLokalniRF(n+1,clanovi,cvorovi)

        #spremi rotacijsku matricu i duljinu
        rotacijskeMatrice[n,:,:] = rotacijskaMatrica
        lengths[n] = length

    #_____________________________________ 10.0 Pohrani transformacijsku matricu za svaki clan
_____________________________

    #spremi ponavljajući dio matrice za svaki element tak da ne mroamo svaki put računat
tranformaciju

    #izračunaj transformacijsku matricu za svaki član

    TransformacijskeMatrice = np.empty((len(clanovi),3,3)) #Inicijaliziraj spremnik podataka

    for n, cln in enumerate(clanovi):

        rMatrica = rotacijskeMatrice[n,:,:] #defoltna orijentacijska rotacijska matrica za ovaj
član
        TransformacijskeMatrice[n,:,:] = rMatrica.T #da bi nam bilo lakše, spremi
transformacisjku matricu direktno

    #_____________________________________ 13.0 Napravi globalni vektor sila i dodaj koncetrirane
sile

    #inicijaliziraj prazni vektor sila i momenta

    vektorSila = np.array([np.zeros(len(cvorovi)*6)]).T
```

```
   #Dodaj koncetrirana opterećenja vektoru sila

   if(len(podaciKoordinataSila)>0):
      #rastavi podatke o lokaciji sila ( index počinje s 0)
      xSila = podaciKoordinataSila[:,1]
      ySila = podaciKoordinataSila[:,2]
      zSila = podaciKoordinataSila[:,3]

      #Dodjeli sile stupnjevima slobode
      if(koncOptOs=='x'):
         vektorSila[xSila] = P
      elif(koncOptOs=='y'):
         vektorSila[ySila] = P
      else:
         vektorSila[zSila] = P

   #_____ 20.0 Napravi osnovnu matricu krutosti, Kp

   nDoF = np.amax(clanovi)*6 # Broj stupnjeva slobode u problemu
   Kp = np.zeros([nDoF,nDoF]) #inicijaliziraj osnovnu matricu krutosti

   for n,cln in enumerate(clanovi):
      cvor_i = cln[0]#broj čvora za čvor i ovog člana
      cvor_j = cln[1]#broj čvora za čvor j ovog člana

      [K11,K12,K21,K22] = izracunKg3Dbeam(n+1,lengths,TransformacijskeMatrice)
      #indexi osnovne matrice krutosti povezane s svakim čvorom
      ia = 6*cvor_i-6 #index 0 (npr. cvor 1)
      ib = 6*cvor_i-1 #index 5 (npr. cvor 1)
      ja = 6*cvor_j-6 #index 6 (npr. cvor 2)
      jb = 6*cvor_j-1 #index 11(npr. cvor 2)

      Kp[ia:ib+1, ia:ib+1] = Kp[ia:ib+1, ia:ib+1] + K11
      Kp[ia:ib+1, ja:jb+1] = Kp[ia:ib+1, ja:jb+1] + K12
      Kp[ja:jb+1, ia:ib+1] = Kp[ja:jb+1, ia:ib+1] + K21
      Kp[ja:jb+1, ja:jb+1] = Kp[ja:jb+1, ja:jb+1] + K22

   #_____ 21.0 matrica krutosti konstrukcije,
Ks_____

   #Ukloni ograničene (i rotacijske) oslobođene stupnjeve slobode kako bi napravili
   # matricu krutosti konstrukcije
   #Ukloni stupnjeve slobode koji nisu slobodni

   uklonjeniDoF = podaciPotpore #tu se dodaju zglobni spojevi
   uklonjeniIndex = [x-1 for x in uklonjeniDoF]#index svakog uklonjenog stupnja slobode

   #Prebaci u matricu krutosti konstrukcije uklanjanjem redova i stupca po uklonjenim
stupnjevim slobode
   Ks = np.delete(Kp,uklonjeniIndex,0)#obriši redove
```

```
    Ks = np.delete(Ks,uklonjeniIndex,1)#obriši stupce
    Ks = np.matrix(Ks)

    #_____ 22.0 Riješi pomake
_____

    vektorSilaRed = copy.copy(vektorSila)
    vektorSilaRed = np.delete(vektorSilaRed,uklonjeniIndex,0) #obriši retke prema
ograničenim DoF

    U = Ks.I*vektorSilaRed

    #_____ 23.0 Rješi reakcije
_____

    #Složi globalni vektor pomaka

    UG = np.zeros(nDoF) #inicijaliziraj niz koji sprema vektor globalnih pomaka
    c = 0 #Inicijaliziraj brojač koji prati koliko smo ograničenja nametnuli

    for i in np.arange(nDoF):
        if i in uklonjeniIndex:
            #pretstavlja pomak 0
            UG[i] = 0
        else:
            #dodijeli stvarni pomak
            UG[i] = U[c]
            c=c+1
    UG = np.array([UG]).T


    #_____ 24.0 Riješi sile članova(aksijalne sile, momenti,
smične)_____

    silaXcln = np.array([]) #Inicijaliziraj niz koji sprema aksijalne sile članova

    ukupnaDuljinaClanova = 0
    silaUClanuJePrevelika = None
    for n,cln in enumerate(clanovi):

        silaUClanuJePrevelika = 0

        #Izdvoji svojstva pojedinog člana
        A = Povrsina
        E = ModElas
        L = lengths[n]

        cvor_i = cln[0] #Broj čvora za čvor i ovog člana
        cvor_j = cln[1] #Broj čvora za čvor j ovog člana
```

```
iHorizontalno = cvorovi[cvor_i-1][0]
iVertikalno = cvorovi[cvor_i-1][1]

jHorizontalno = cvorovi[cvor_j-1][0]
jVertikalno = cvorovi[cvor_j-1][1]

duljinaClana = (iHorizontalno-jHorizontalno)**2-(iVertikalno-jVertikalno)**2
L = duljinaClana
ukupnaDuljinaClanova += duljinaClana

#indexi primarne matrice krutosti određene svakim čvorom
ia = 6*cvor_i-6  #index 0 (npr. cvor 1)
ib = 6*cvor_j-1  #index 5 (npr. cvor 1)
ja = 6*cvor_i-6  #index 6 (npr. cvor 2)
jb = 6*cvor_j-1  #index 11 (npr. cvor 2)

#Napravi punu transformacijsku matricu za ovaj element
TM = np.zeros((12,12))
T_repeat = TransformacijskeMatrice[n,:,:]
TM[0:3,0:3] = T_repeat
TM[3:6,3:6] = T_repeat
TM[6:9,6:9] = T_repeat
TM[9:12,9:12] = T_repeat

pomak = np.array([[ UG[ia,0],
            UG[ia+1,0],
            UG[ia+2,0],
            UG[ia+3,0],
            UG[ia+4,0],
            UG[ib,0],
            UG[ja,0],
            UG[ja+1,0],
            UG[ja+2,0],
            UG[ja+3,0],
            UG[ja+4,0],
            UG[jb,0]]]).T

pomak_lokalni = np.matmul(TM,pomak)

F_aksijalna = (A*E/L)*(pomak_lokalni[6] - pomak_lokalni[0])[0]


#Spremi momente i sile članova
silaXcln = np.append(silaXcln,F_aksijalna)



if F_aksijalna> dopustenaSila:
   silaUClanuJePrevelika = -10
```

```
        if F_aksijalna> dopustenaSila:
            silaUClanuJePrevelika = -10


    #print("cvorPracenja je ",cvorPracenja)
    #print("duljina ug je ", len(UG))
    fit = math.sqrt((UG[int(cvorPracenja*6-6)]**2 + UG[int(cvorPracenja*6-4)]**2))

    fit = (-1*fit*ukupnaDuljinaClanova)*100+silaUClanuJePrevelika

    return fit,UG


#_____ EA
_____
##########################################################################
##########################################################################
##########
##########################################################################
##########################################################################
##########


fitOsnovni,UG = fitness(clanovi,cvorovi,cvorPracenja)

roditeljiCvoroviKoordinate = []
roditeljiFitnes = np.array([])

spremnikPozicijaČvorova = []
spremnikFitnes = []

#sad inicijaliziramo niz u koji ćemo staviti info pojedinih roditelja

brojeviČvorovaZaPertubaciju = []
for i in cvoroviOpt:

    for n,j in enumerate(cvorovi):
        if i[0]==j[0] and i[2]==j[2]:
            brojeviČvorovaZaPertubaciju.append(int(n+1))

for i in range(brojRoditelja):

    roditeljCvorKoordinate = cvorovi.copy()

    nasumicno1 = [1,-1] #pertubiramo cvorove ili u pozitivnu stranu osi ili u negativnu
    indexiZaPerturbacije =
np.random.uniform(0,int(len(brojeviČvorovaZaPertubaciju)),brojPertubacijaČvorova)
    for n,j in enumerate(indexiZaPerturbacije):

        indexZaPertubaciju = int(j)
```

```
        cvorKojiPertubiramo = brojeviČvorovaZaPertubaciju[indexZaPertubaciju]

        pertubacijaVodoravneKoordinate =
nasumicno1[np.random.randint(0,2)]*iznosPerturbacije
        pertubacijaOkomiteKoordinate = nasumicno1[np.random.randint(0,2)]*iznosPerturbacije

        roditeljCvorKoordinate[cvorKojiPertubiramo-1][0] += pertubacijaVodoravneKoordinate
#dodajemo pertubaciju vodoravnoj koordinati cvora koji pertubiramo
        roditeljCvorKoordinate[cvorKojiPertubiramo-1][2] += pertubacijaOkomiteKoordinate
#dodajemo pertubaciju okomitoj koordinati cvora koji pertubiramo

    fit,UG = fitness(clanovi,roditeljCvorKoordinate,cvorPracenja)


    roditeljiFitnes = np.append(roditeljiFitnes,[fit],axis = 0)
    roditeljiCvoroviKoordinate.append(roditeljCvorKoordinate)

brojKorakaBezPomaka = 0
for i in range(brojKoraka):

    print("generacija broj ", i+1)

    roditeljiSortiraniFitnes = -np.sort(-roditeljiFitnes)

    razlikaFitnes = abs(fitPrije)-abs(roditeljiSortiraniFitnes[0])
    print("razlika fitnes je ", razlikaFitnes)


    fitPrije = roditeljiSortiraniFitnes[0]

    if razlikaFitnes<treshold:
        brojKorakaBezPomaka +=1

        print("Broj generacija bez poboljšanja fitnes", brojKorakaBezPomaka)
    else:
        brojKorakaBezPomaka = 0

    if brojKorakaBezPomaka==granicaGeneracijaBezPomaka:
        print("GA se prekida nakon ",i," koraka jer nema pomaka u poboljšanju fitnes")
        break

    indexiRoditeljaZaPravljenjeDjece = []


    for j in range(brojNajboljihRoditeljaOdKojihPravimoDjecu):
        A = np.where(roditeljiFitnes==roditeljiSortiraniFitnes[j])[0][0]
        indexiRoditeljaZaPravljenjeDjece.append(A)

    djecaCvoroviKoordinate = []
```

```
    for j in range(int(brojDjece/2)):
        prviBoracIndexZaOca =
indexiRoditeljaZaPravljenjeDjece[np.random.randint(0,brojNajboljihRoditeljaOdKojihPravi
moDjecu)]
        drugiBoracIndexZaOca =
indexiRoditeljaZaPravljenjeDjece[np.random.randint(0,brojNajboljihRoditeljaOdKojihPravi
moDjecu)]

        if roditeljiFitnes[prviBoracIndexZaOca]>roditeljiFitnes[drugiBoracIndexZaOca]:
            otacCvoroviKoordinate = roditeljiCvoroviKoordinate[prviBoracIndexZaOca]
        else:
            otacCvoroviKoordinate = roditeljiCvoroviKoordinate[drugiBoracIndexZaOca]


        prviBoracIndexZaMajku =
indexiRoditeljaZaPravljenjeDjece[np.random.randint(0,brojNajboljihRoditeljaOdKojihPravi
moDjecu)]
        drugiBoracIndexZaMajku =
indexiRoditeljaZaPravljenjeDjece[np.random.randint(0,brojNajboljihRoditeljaOdKojihPravi
moDjecu)]
        if roditeljiFitnes[prviBoracIndexZaMajku]>roditeljiFitnes[drugiBoracIndexZaMajku]:
            majkaCvoroviKoordinate = roditeljiCvoroviKoordinate[prviBoracIndexZaMajku]
        else:
            majkaCvoroviKoordinate = roditeljiCvoroviKoordinate[drugiBoracIndexZaMajku]


        granicaGena =
round(np.random.uniform(donjaGranicaGena*len(cvorovi),gornjaGranicaGena*len(cvorovi))
)

        cvoroviOdOcaZaSina = otacCvoroviKoordinate[0:granicaGena]
        cvoroviodMajkeZaSina = majkaCvoroviKoordinate[granicaGena:len(cvorovi)]

        for k in cvoroviodMajkeZaSina:
            cvoroviOdOcaZaSina = np.append(cvoroviOdOcaZaSina,[k],axis = 0)

        sin = cvoroviOdOcaZaSina

        djecaCvoroviKoordinate.append(sin)

        cvoroviOdMajkeZaKcer = majkaCvoroviKoordinate[0:granicaGena]
        cvoroviodOcaZaKcer = otacCvoroviKoordinate[granicaGena:len(cvorovi)]

        for k in cvoroviodOcaZaKcer:
            cvoroviOdMajkeZaKcer = np.append(cvoroviOdMajkeZaKcer,[k],axis = 0)

        kcer = cvoroviOdMajkeZaKcer

        djecaCvoroviKoordinate.append(kcer)
```

```
   #mutiranje djece!

   for j in range(brojDjece):

      indexiZaMutiranje =
np.random.uniform(0,int(len(brojeviČvorovaZaPertubaciju)),brojMutacijaČvorova)
      pomakPlusIliMinus = [1,-1] #pertubiramo cvorove ili u pozitivnu stranu osi ili u
negativnu

      for n,k in enumerate(indexiZaMutiranje):

         indexZaMutiranje = int(k)
         cvorKojiPertubiramo = brojeviČvorovaZaPertubaciju[indexZaMutiranje]

         pertubacijaVodoravneKoordinate =
pomakPlusIliMinus[np.random.randint(0,2)]*iznosMutacije
         pertubacijaOkomiteKoordinate =
pomakPlusIliMinus[np.random.randint(0,2)]*iznosMutacije

         djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][0] +=
pertubacijaVodoravneKoordinate #dodajemo pertubaciju vodoravnoj koordinati cvora koji
pertubiramo
         djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][2] +=
pertubacijaOkomiteKoordinate #dodajemo pertubaciju okomitoj koordinati cvora koji
pertubiramo


         if djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][0] < 0:
            djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][0] += 100*iznosMutacije
         elif djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][0] >
vodoravnaGranicaModela:
            djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][0] -= 100*iznosMutacije

         if djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][1] < 0:
            djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][1] += 100*iznosMutacije
         elif djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][1] > okomitaGranicaModela:
            djecaCvoroviKoordinate[j][cvorKojiPertubiramo-1][1] -= 100*iznosMutacije

   #provjera fitnes


   fitnesZaPlotanje = 100

   roditeljiCvoroviZaSljedecuGeneraciju = []
   roditeljiFitnesZaSljedecuGeneraciju = np.array([])

   for j in range(elitniRoditelji):
      indexElitnogRoditelja = indexiRoditeljaZaPravljenjeDjece[j]
      koordinateČvorovaElitnogRoditelja =
roditeljiCvoroviKoordinate[indexElitnogRoditelja]
```

```
        fitnesElitnogRoditelja = roditeljiFitnes[indexElitnogRoditelja]

        if math.isnan(fitnesElitnogRoditelja):
            koordinateČvorovaElitnogRoditelja = cvorovi.copy()
            fitnesElitnogRoditelja = fitOsnovni


        if abs(fitnesElitnogRoditelja)<fitnesZaPlotanje:
            cvoroviZaPrikaz = koordinateČvorovaElitnogRoditelja
            fitnesZaPlotanje = abs(fitnesElitnogRoditelja)
        roditeljiCvoroviZaSljedecuGeneraciju.append(koordinateČvorovaElitnogRoditelja)
        roditeljiFitnesZaSljedecuGeneraciju =
np.append(roditeljiFitnesZaSljedecuGeneraciju,[fitnesElitnogRoditelja], axis = 0)

    for n,j in enumerate(djecaCvoroviKoordinate):
        djeteCvoroviKoordinate = j

        try:
            fit,UG = fitness(clanovi,djeteCvoroviKoordinate,cvorPracenja)

        except np.linalg.LinAlgError:
            djeteCvoroviKoordinate = cvorovi.copy()
            fit = fitOsnovni

        if math.isnan(fit):
            djeteCvoroviKoordinate = cvorovi.copy()
            fit = fitOsnovni

        if abs(fit)<fitnesZaPlotanje:
            cvoroviZaPrikaz = djeteCvoroviKoordinate
            fitnesZaPlotanje = abs(fit)

        roditeljiCvoroviZaSljedecuGeneraciju.append(djeteCvoroviKoordinate)
        roditeljiFitnesZaSljedecuGeneraciju =
np.append(roditeljiFitnesZaSljedecuGeneraciju,[fit], axis = 0)



    roditeljiFitnes = roditeljiFitnesZaSljedecuGeneraciju.copy()
    roditeljiCvoroviKoordinate = roditeljiCvoroviZaSljedecuGeneraciju.copy()

    spremnikPozicijaČvorova.append(cvoroviZaPrikaz)
    spremnikFitnes.append(fitnesZaPlotanje)

    if (i+1)%generacijaKojuPlotamo == 0:
        vodoravna,dubinska,okomita = cvoroviZaPrikaz.T
        plt.xlim([-0.1, vodoravnaGranicaModela+1])
        plt.ylim([-0.1, okomitaGranicaModela+1])
```

```
        for n, cln in enumerate(clanovi):
            cvor_i = cln[0] #broj čvora za čvor i ovog člana
            cvor_j = cln[1] #broj čvora za čvor j ovog člana

            iVodoravno = cvoroviZaPrikaz[cvor_i-1,0] #x-koord čvora i ovog člana
            iOkomito = cvoroviZaPrikaz[cvor_i-1,2] #z-koord čvora i ovog člana
            jVodoravno = cvoroviZaPrikaz[cvor_j-1,0] #x-koord čvora j ovog člana
            jOkomito = cvoroviZaPrikaz[cvor_j-1,2] #z-koord čvora j ovog člana

            plt.plot([iVodoravno,jVodoravno],[iOkomito,jOkomito],'k')

        plt.text(vodoravnaGranicaModela+0.1, okomitaGranicaModela, 'generacija broj ' +
str(i+1), fontsize = 7)
        plt.text(vodoravnaGranicaModela+0.1, okomitaGranicaModela-0.1, 'broj roditelja: ' +
str(brojRoditelja), fontsize = 7)
        plt.text(vodoravnaGranicaModela+0.1, okomitaGranicaModela-0.2, 'broj djece: ' +
str(brojDjece), fontsize = 7)
        plt.text(vodoravnaGranicaModela+0.1, okomitaGranicaModela-0.3, 'iznos fitnes: ' +
str(fitnesZaPlotanje), fontsize = 7)
        plt.xlabel("vodoravna koordinata")
        plt.ylabel("vertikalna koordinata")
        plt.title("Evolucija konstrukcije s GA")
        plt.show()

    filename = 'KosoSpremnikPozicijaČvorova1'
    outfile = open(filename,'wb')
    pickle.dump(spremnikPozicijaČvorova,outfile)
    outfile.close()

    filename = 'KosoSpremnikFitnes1'
    outfile = open(filename,'wb')
    pickle.dump(spremnikFitnes,outfile)
    outfile.close()

winsound.Beep(freq, duration)
```

Kodovi za blender python skriptu

Skripta za exportanje članova i čvorova

```
"""

skripta exporta sve čvorove i rubove u dve csv datoteke

cvor blizi ishodistu je cvor i  a cvor dalji je cvor j
"""

import bpy
import csv
import math
import numpy as np

obdata = bpy.context.object.data

# izvadi sve vertexe i spakiraj ih za dalje
vertices = np.empty((0,3),int)
for v in obdata.vertices:
    vertex = np.array([v.co.x,v.co.y,v.co.z])
    vertices = np.append(vertices,[vertex],axis=0)

# izvadi sve rubove(cvor i i cvor j) i spakiraj ih za dalje

edges = np.empty((0,2),int)
for e in obdata.edges:

    vertA = e.vertices[0]
    vertA_x = obdata.vertices[vertA].co.x
    vertA_y = obdata.vertices[vertA].co.y
    vertA_z = obdata.vertices[vertA].co.z

    vertB = e.vertices[1]
    vertB_x = obdata.vertices[vertB].co.x
    vertB_y = obdata.vertices[vertB].co.y
    vertB_z = obdata.vertices[vertB].co.z

    #odredi cvor i i j na temelju udaljenosti od ishodišta
    dA = math.sqrt(vertA_x**2+vertA_y**2+vertA_z**2)
    dB = math.sqrt(vertB_x**2+vertB_y**2+vertB_z**2)

    if(dA<dB):
        edge = np.array([vertA+1,vertB+1])
    else:
        edge = np.array([vertB+1,vertA+1])

    edges = np.append(edges,[edge],axis = 0)
```

```
#exportaj vertexe u csv datoteku
filename = "D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoVertices.csv"
#piši u csv csv file
with open(filename,'w') as csvfile:
   csvwriter = csv.writer(csvfile) #creating a csv writing object
   csvwriter.writerows(vertices) #writeng the data to rows

#exportaj rubove u csv datoteku
filename = "D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoEdges.csv"
#piši u csv file
with open(filename,'w') as csvfile:
   csvwriter = csv.writer(csvfile) #creating a csv writing object
   csvwriter.writerows(edges) #writeng the data to rows
```

Skripta za eksportanje veza s okolinom

```
"""
Skripta exporta stupnjeve slobode
    - Cvorovi selektorani u edit modu
    - Cvorovi koji su editirani maskom

=====================================================
Određivanje maske za stupnjeve slobode gibanja, maska = [ux, uy, uz, theta_x, theta_y,
theta_z]
#(0 = nije fiksno, 1 = fiksno

for example
uklještenje maska = [1,1,1,1,1,1]
zglobni oslonac maska = [1,1,1,0,0,0]
"""

import bpy
import csv
import numpy as np
import bmesh

mask = [1,1,1,0,0,0]

obdata = bpy.context.edit_object.data
bm = bmesh.from_edit_mesh(obdata)
bm.faces.active = None

#dohvati označene vertexe
DoF = np.empty((0,6),int)
supportNode = np.empty((0,1),int)

for v in bm.verts:
```

```
    if v.select:

        indexNo = v.index+1
        ux_DoF = 6*indexNo-5
        uy_DoF = 6*indexNo-4
        uz_DoF = 6*indexNo-3
        theta_x_DoF = 6*indexNo-2
        theta_y_DoF = 6*indexNo-1
        theta_z_DoF = 6*indexNo

        vertex = np.array([ux_DoF,uy_DoF,uz_DoF,theta_x_DoF,theta_y_DoF,theta_z_DoF])
        vertex = vertex*mask
        DoF = np.append(DoF,[vertex],axis = 0)
        supportNode = np.append(supportNode, [np.array([indexNo])],axis = 0)


#exportaj brojeve čvorova veza s okolinom
filename = "D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoRestraint-
DoF.csv"

piši u csv datoteku
with open(filename,'w') as csvfile:
    csvwriter = csv.writer(csvfile) #creating a csv writing object
    csvwriter.writerows(DoF) #writeng the data to rows

#exportaj koordinate čvorova veza s okolinom
filename = "D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoRestraint-
Nodes.csv"
#piši u csv datoteku
with open(filename,'w') as csvfile:
    csvwriter = csv.writer(csvfile) #creating a csv writing object
    csvwriter.writerows(supportNode) #writeng the data to rows
```

Skripta za eksportanje čvorova u kojima djeluje sila

```
"""
Za odabrane cvorove skripta eksporta stupnjeve slobode momenta i sila za taj cvor
"""

import bpy
import csv
import numpy as np
import bmesh

obdata = bpy.context.edit_object.data

#pristupi podacima o cvorovim i članovima
```

```
bm = bmesh.from_edit_mesh(obdata)
bm.faces.active = None

#dohvati označene vertexe

vertices = np.empty((0,7),int)

for v in bm.verts:
   if v.select:
      IndexNo = v.index+1
      xForceIndex = 6*IndexNo-6
      yForceIndex = 6*IndexNo-5
      zForceIndex = 6*IndexNo-4
      xMomentIndex = 6*IndexNo-3
      yMomentIndex = 6*IndexNo-2
      zMomentIndex = 6*IndexNo-1

      vertex =
np.array([IndexNo,xForceIndex,yForceIndex,zForceIndex,xMomentIndex,yMomentIndex,zM
omentIndex])
      vertices = np.append(vertices,[vertex], axis = 0)


#exportaj vertexe u csv fajl
filename = "D:/Poduzetan/kodovi/Diplomski optimizacija stapne konstrukcije/KosoForce-
data.csv"
#piši u csv datoteku
with open(filename,'w') as csvfile:
   csvwriter = csv.writer(csvfile) #creating a csv writing object
   csvwriter.writerows(vertices) #writeng the data to row
```

Skripta za eksportanje čvorova koji se smiju mutirati

```
import bpy
import csv
import numpy as np
import bmesh

obdata = bpy.context.edit_object.data


bm = bmesh.from_edit_mesh(obdata)
bm.faces.active = None


vertices = np.empty((0,3),int)

for v in bm.verts:
```

```
    if v.select:
        vertex = np.array([v.co.x,v.co.y,v.co.z])
        vertices = np.append(vertices,[vertex],axis=0)


filename = "D:/Poduzetan/kodovi/Diplomski optimizacija stapne
konstrukcije/KosoVertexiOpt.csv"

with open(filename,'w') as csvfile:
    csvwriter = csv.writer(csvfile) #creating a csv writing object
    csvwriter.writerows(vertices) #writeng the data to row
```