Optimizacija okvirnog rebra jednostavne brodske konstrukcije

Banjedvorec, Dominik

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Dominik Banjedvorec

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Izv. prof. dr. sc. Pero Prebeg

Student:

Dominik Banjedvorec

Zagreb, 2022.

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

Zahvaljujem se mentoru izv. prof. dr. sc. Peri Prebegu na velikoj potpori tokom izrade ovog rada. Uvijek je bio na raspolaganju te mi je uvelike pomogao oko programiranja u Pythonu kao i s ostalim detaljima vezanim uz rad.

Dominik Banjedvorec

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija brodogradnje

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Dominik Banjedvorec** JMBAG: **0035215559**

Naslov rada na hrvatskom jeziku: **Optimizacija okvirnog rebra jednostavne brodske konstrukcije**

Naslov rada na engleskom jeziku: **Web frame optimization of simple ship structure**

Opis zadatka:

Pri projektiranju brodske konstrukcije neophodno je provesti analizu odziva metodom primjerene točnosti. U tu svrhu, uz analitičke metode, sve više se koristi metoda konačnih elemenata. Pristup izvorom kodu implementacije pojedine metode za analizu odziva omogućuje jednostavnije povezivanje istih s optimizacijskim metodama, odnosno optimizaciju brodske konstrukcije. Program otvorenog koda d3v-ssd (Design visualizer for Ship structural design) proširenje je programa linaetal-fsb/d3v, koji omogućuje trodimenzijsku vizualizaciju elemenata konstrukcije te jednostavno definiranje grafičkog sučelja za pojedine specifične namjene. U radu je potrebno, proširenjem funkcionalnosti programa otvorenog koda d3v-ssd u programskom jeziku Python, provesti optimizaciju okvirnog rebra jednostavne brodske konstrukcije primjenom dvije različite metode za analizu odziva, odnosno primjenom metode pomaka i metode konačnih elemenata.

Zadatak obuhvaća sljedeće:

- upoznavanje s trenutnom verzijom programa otvorenog koda d3v-ssd, koji je povezan s programom otvorenog koda OOFEM te omogućuje analizu odziva metodom konačnih elemenata
- upoznavanje s optimizacijskim metodama prikladnim za rješavanje jednociljnih optimizacijskih problema s ograničenjima, dostupnih u Python biblioteci scipy
- izradu Python modula za izračun odziva konstrukcije okvira metodom pomaka te vizualizaciju istog u d3v-ssd
- formulaciju optimizacijskog problema okvirnog rebra jednostavne brodske konstrukcije
- izradu Python modula koji omogućuje zadavanje i rješavanje formuliranog optimizacijskog problema, primjenom optimizacijskih metoda dostupnih u Python biblioteci scipy
- provedbu optimizacije primjenom implementiranih Python modula, na primjeru okvirnog rebra jednostavne brodske konstrukcije, primjenom dviju različitih metode za analizu odziva, odnosno primjenom metode pomaka i metode konačnih elemenata.
- usporedbu rezultata za optimizacijske probleme s različitim metodama izračuna odziva.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:

30. 11. 2021.

Zadatak zadao:

Izv. prof. dr. sc. Pero Prebeg

Datum predaje rada:

1. rok: 24. 2. 2022.
2. rok (izvanredni): 6. 7. 2022.
3. rok: 22. 9. 2022.

Predviđeni datumi obrane:

1. rok: 28. 2. – 4. 3. 2022.
2. rok (izvanredni): 8. 7. 2022.
3. rok: 26. 9. – 30. 9. 2022.

Predsjednik Povjerenstva:

Izv. prof. dr. sc. Ivan Ćatipović

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
φ_A	m	Kut zakreta u čvoru A
φ_B	m	Kut zakreta u čvoru B
u	m	Linearni pomak konstrukcije
ψ	m	Kut nagiba
M_{12}	Nm	Moment u čvoru 1
M_{21}	Nm	Moment u čvoru 2
k_{12}		Koeficijent krutosti nosača na savijanje
m_{12}	Nm	Moment u točki 1 potpuno upete grede oslonjene između čvorova 1 i 2 od vanjskog opterećenja
m_{21}	Nm	Moment u točki 2 potpuno upete grede oslonjene između čvorova 1 i 2 od vanjskog opterećenja
p_{D3}	N/mm^2	Tlačno opterećenje na palubi 3
p_{D2}	N/mm^2	Tlačno opterećenje na palubi 2
p_{D1}	N/mm^2	Tlačno opterećenje na palubi 1
p_B	N/mm^2	Tlačno opterećenje na dnu
b_f	mm	Širina pojasne trake
t_f	mm	Debljina pojasne trake
h_w	mm	Visina struka
t_w	mm	Debljina struka
$\sigma_{xdopušteno}$	N/mm^2	Dopušteno naprezanje u smjeru osi x
$\sigma_{x\text{kritično}}$	N/mm^2	Kritično naprezanje u smjeru osi x

SAŽETAK

U ovom radu provedena je optimizacija okvirnog rebra jednostavne brodske konstrukcije, pri čemu je za implementaciju optimizacijskog modela korišten programski jezik Python. Proračun odziva okvira proveden je metodom pomaka koja je jedna od metoda kojom se može dovoljno točno odrediti naprezanja u inicijalnoj fazi projektiranja konstrukcije. U okviru rada izrađen je matematički model proračuna odziva okvira primjenom objektno orijentiranog programiranja, u programskom jeziku Python. Za optimizaciju korištena je besplatna biblioteka SciPy u okviru koje su dostupne implementacije nekoliko različitih optimizacijskih metoda. U ovom radu, za optimizaciju okvirnog rebara jednostavne brodske konstrukcije, korištene su optimizacijske metode SLSQP i COBYLA.

Ključne riječi: optimizacija brodske konstrukcije, proračun okvira metodom pomaka, SciPy, SLSQP, COBYLA, Python, objektno orijentirano programiranje

SUMMARY

The thesis presents the structural optimization of a simple ship structure web frame, where Python programming language was used for the implementation of the optimization model. The calculation of the response of a web frame was performed by the displacement method, which is one of the methods by which the stresses in the initial design phase of the structure can be determined with sufficient accuracy. Mathematical model of frame response calculation was developed using object-oriented programming, in Python programming language. For optimization, free SciPy library was used, within which implementations of several different optimization methods are available. In this paper, SLSQP and COBYLA optimization methods were used to optimize the web frame of a simple ship structure.

Key words: Python, ship structural optimization, frame analysis with displacement method, SciPy, SLSQP, COBYLA, object-oriented programming

1. UVOD

Optimizacije je postupak kojim se pri projektiranju ili planiranju ostvaruje najbolji mogući izbor tehničkih veličina na temelju prethodno određenih kriterija. Tek u zadnja dva desetljeća je optimizacija postala nezaobilazni dio inženjerskih struka. Primjenom optimizacije dolazi se do uštede različitih resursa.

U ovom završnom radu optimizacija je rađena na primjeru brodskog okvira. Za statički proračun, tj. određivanje dijagrama momenta savijanja greda okvira korištena je metoda pomaka.

Optimizacija okvira rađena je u programskom jeziku Python; pomoću besplatne biblioteke SciPy koja se koristi za znanstveno i tehničko računalstvo. SciPy sadrži module za optimizaciju, linearnu algebru, integraciju, interpolaciju različitih zadataka u inženjerstvu. [1]

Ovim radom je proširena funkcionalnost programa otvorenog koda *d3v-ssd* u okviru kojeg je omogućena vizualizacija okvira koji se analizira metodom pomaka.

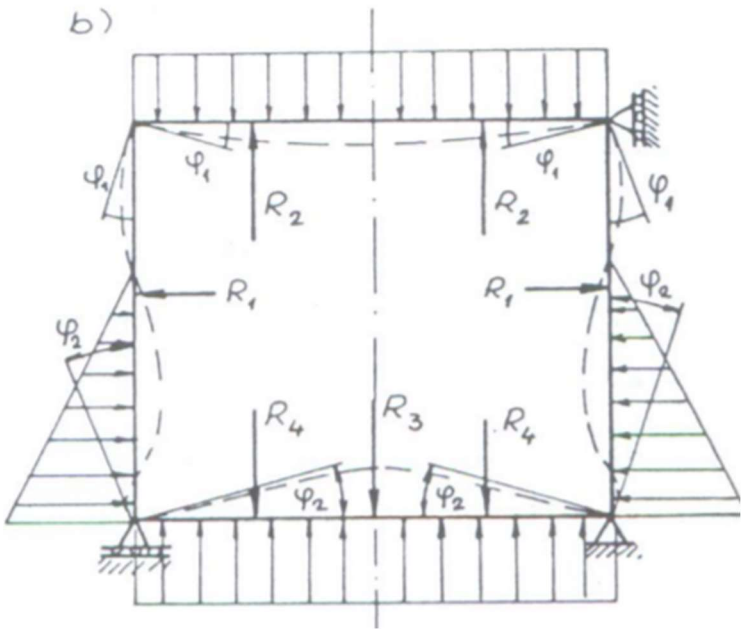
Program d3v-ssd (Design visualizer for Genereal Ship Design) proširenje je programa linaetal-fsb/d3v, koji omogućuje trodimenzijsku (3D) vizualizaciju geometrijskih entiteta te jednostavno definiranje grafičkog sučelja za pojedina namjene. Unutar ovog rada korišteno je i optimizacijsko sučelje programa d3v-ssd koje omogućuje jednostavnije povezivanje modela za analizu odziva s optimizacijskim algoritmima iz biblioteke SciPy.

2. PRORAČUN OKVIRA

Kod monotonih brodskih konstrukcija opterećenih kontinuiranim opterećenjem možemo pretpostaviti da svaki pojedinačni okvir preuzima jednako opterećenje u širini razmaka okvira. Dio opterećenja koje otpada na takav okvir je ono koje djeluje na polovici razmaka tog okvira i njemu susjednih (opterećenje na jednom razmaku okvira).

Kod proračuna čvrstoće stvarnu konstrukciju supstituiramo matematičkim modelom. Izdvajamo okvir, a upliv preostalog dijela konstrukcije nadomještamo rubnim uvjetima. Pomaci okvira kao cjeline u prostoru ne utječu na proračun okvira jer kod toga njegovi elementi ne dobivaju nikakve deformacije. Kako su stranice okvira u određenim razmacima povezane krutim poprečnim pregradama, možemo u većini slučajeva smatrati da se oba boka jednako progibaju u prostoru kod savijanja brodskog trupa, prema tome među njima nema relativnog pomaka. Iznimka su brodovi koji imaju veliku širinu i uzdužne pregrade, pa može doći do relativnih pomaka bokova broda i uzdužnih. [4]

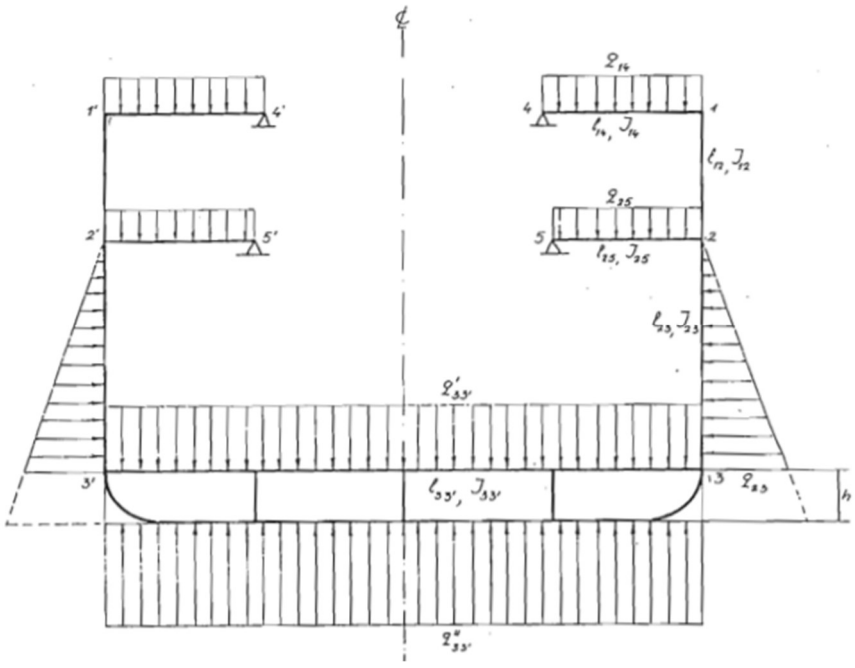
Proračun se može provesti dvjema metodama.



Slika 1. Pojednostavljen prikaz matematičkog modela okvira [4]

2.1. Opterećenje elemenata okvira

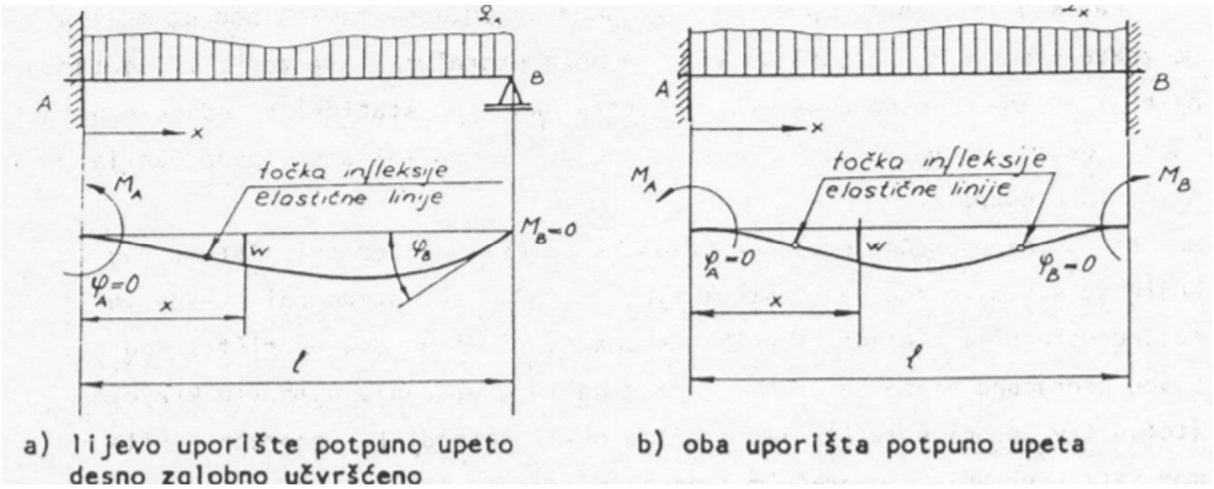
Različiti konstrukcijski elementi su opterećeni na različite načine, prvobitno zbog svoje težine, odnosno težine okvira. Nadalje, postoji opterećenje uslijed tereta; opterećenje u tankovima, opterećenje paluba. Opterećenje uslijed djelovanja hidrostatičkog tlaka na vanjskoj oplati broda i dnu, te hidrodinamički pritisak okolne tekućine.



Slika 2. Opterećenje brodskog okvira [4]

2.2. Modeliranje rubnih uvjeta

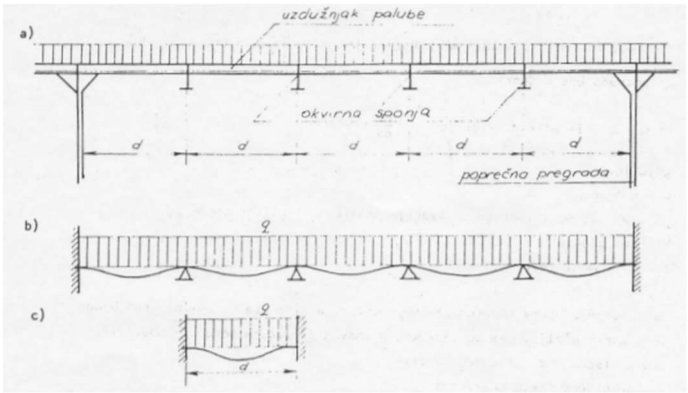
U statičkom proračunu okvira govorimo o statički neodređenim sustavima tj. raspodjela unutarnjih sila se ne može proračunati samo na osnovu uvjeta ravnoteže nego se moraju uzeti u obzir i uvjeti kompatibilnosti deformacija elemenata sustava. Najjednostavniji dijelovi statički neodređenih sustava su grede na dva oslonca, te ćemo s njima modelirati naš okvir. Način na koji su učvršćeni krajevi greda će definirati raspodjelu poprečnih sila i momenata savijanja u poprečnim presjecima greda.[4]



Slika 3. Dva primjera različito upete grede [4]

2.2.1. Potpuno upeta uporišta

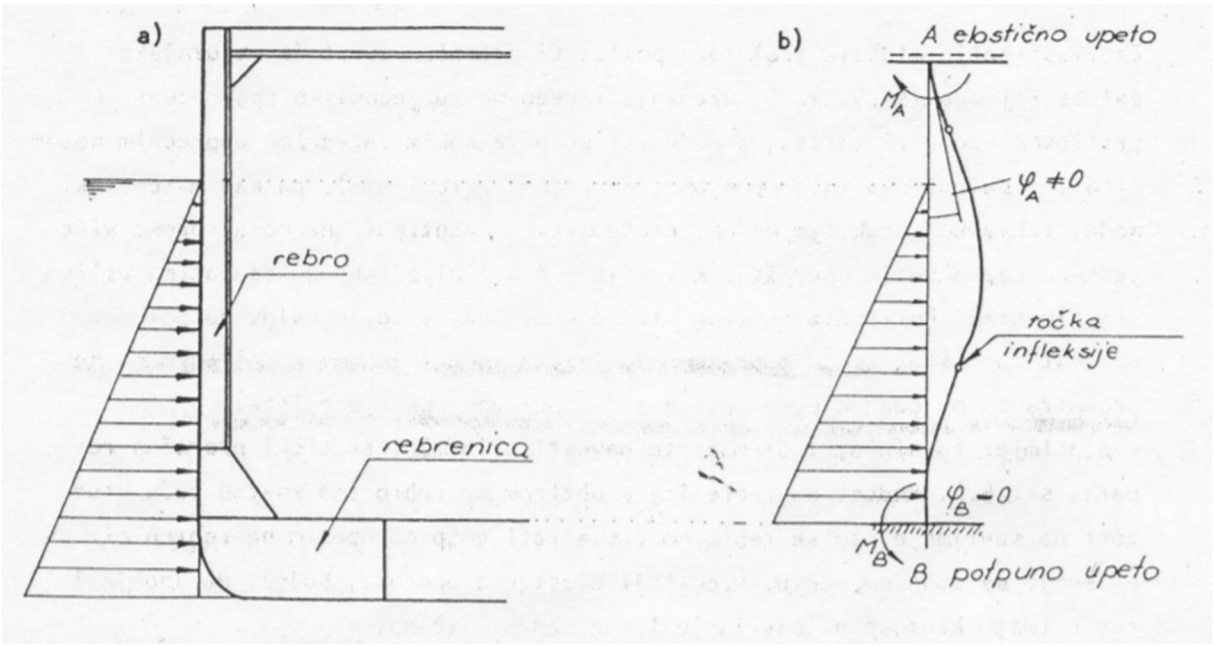
Potpuno upeta konzola ne postoji kod brodske konstrukcije ali ipak mi možemo proračunski model idealizirati sa takvim krajevima ako ona zadovoljava određene uvjete kao što su: simetrična konstrukcija i simetrično opterećenje, kad se jedan nosač male krutosti spaja sa nosačem velike krutosti.



Slika 4. Matematički model za proračun uzdužnjaka [4]

2.2.2. Modeliranje rubnih uvjeta rebara

Rebro se može smatrati potpuno upetim na rebrenici, a spoj rebra i sponje elastično upeto jer rebro i sponja imaju krutost na savijanje istog reda veličine.

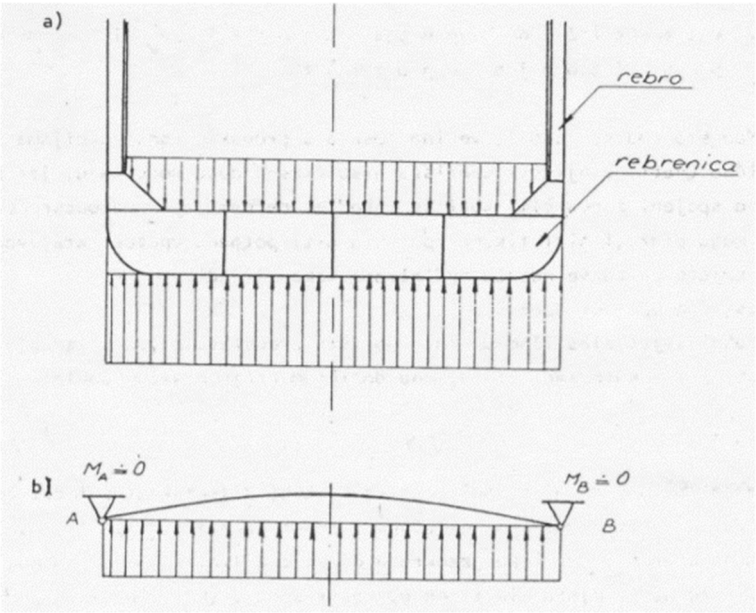


Slika 5. Matematički model za proračun rebara [4]

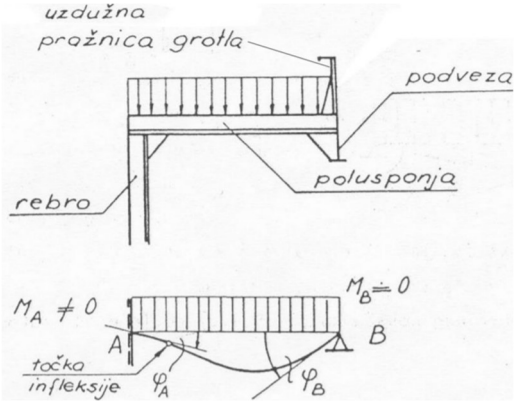
2.2.3. Zglobna uporišta

Zglobna uporišta također ne postoje kod brodskih konstrukcija ali ipak možemo proračunski model idealizirati s takvim rubnim uvjetima ako su ispunjeni slijedeći uvjeti: kada je razmatrani nosač relativno velike krutosti u odnosu na nosač s kojim se spaja, npr. rebrenica ima znatno veću krutost od rebra i rebro nije u stanju proizvesti veći moment upetosti.

Van serijski primjer bi bila uzdužna pražnica koja ima znatno veću krutost na savijanje od polusponje, ipak se uporište na uzdužnoj pražnici može smatrati zglobno jer moment upetosti polusponje opterećuje pražnicu na torziju, a njena torzijska krutost kao i svakog visokog nosača otvorenog presjeka je mala. [4]



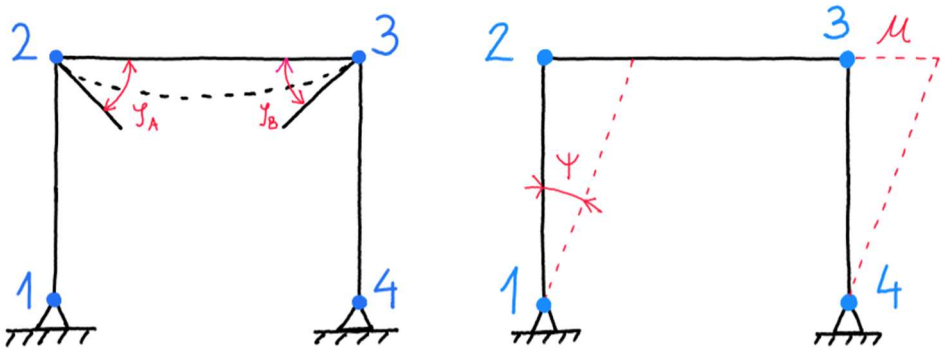
Slika 6. Matematički model za proračun rebrenice [4]



Slika 7. Spoj uzdužne pražnice i grotla [4]

2.3. Metoda pomaka

Metoda pomaka je metoda proračuna konstukcija u kojima se konstrukcija dijeli na sustav grednih elemenata u kojima su nepoznanice vrijednosti translacijskih pomaka i kutovi zakreta odabranih točaka sustava nazvanih čvorovima. Oni su zapravo istaknute točke konstrukcije tj. točke u kojima se spajaju dva gredna elementa. Primjena ove metode je kod rješavanja statički neodređenih sustava kao što je okvir ali se mogu rješavati i statički određeni sustavi. Slika 10. prikazuje čvorove 1,2,3,4 te pomak konstrukcije u, kutove zakreta φ_A, φ_B i kut nagiba ψ. [4]



Slika 8. Čvorovi i pomaci

2.3.1. Princip metode pomaka

Potrebno je nepoznate sile u čvorovima konstrukcije izraziti pomoću njihovih pomaka uz zadovoljenje uvjeta kompatibilnosti pomaka. Postavljanjem uvjeta ravnoteže svih sila/momenata u pojedinim čvorovima (suma svih momenata u pojedinom čvoru je jednaka nuli) dobivamo sustav jednadžbi iz kojeg se mogu odrediti svi pomaci/kutovi zakreta, a pomoću njih i sve sile/momenti. Često se zanemaruju male veličine drugog reda, kao npr. približavanje čvorova uslijed progibanja nosača, linearni pomaci uslijed djelovanja aksijalnih sila, itd. Ako osim zakretanja čvorova postoje i njihovi linearni pomaci, onda se broj jednadžbi povećava za broj kinematičkih nezavisnih linearnih pomaka. U sustavu jednadžbi njih zamjenjujemo s nezavisnim kutovima nagiba nosača ψ. Ako ne postoje linearni pomaci čvorova (uslijed uporišta koja sprječavaju te pomake ili zbog simetrije) nego samo njihovo zakretanje, broj nepoznanica u sustavu jednadžbi jednak je mogućem broju kutova zaokreta čvora φ. [4]

2.3.2. Rješavanje problema metodom pomaka

Za svaki nosač konstrukcije mogu se postaviti jednadžbe tipa 1 ili 2 za momente upetosti na njihovim krajevima tj. u čvorovima. Te jednadžbe se postavljaju samo za one krajeve nosača koji završavaju u čvorovima koji pod opterećenjem dobivaju kutove zaokreta (nisu potpuno upeti). Suma svih momenata upetosti na krajevima nosača koji se sastaju u jednom te istom čvoru mora biti jednaka nuli (uvjet ravnoteže čvorova).

$$\sum_{j=1}^{n} M_{ij} = 0 \tag{1}$$

Postavljanjem uvjeta ravnoteže za sve čvorove koji se zakreću dobivamo toliko jednadžbi koliko ima nepoznatih kutova zakreta.

Izrazi za računanje momenata na krajevima:

$$M_{12} = k_{12} \cdot (2 \cdot \varphi_1 + \varphi_2 - 3 \cdot \psi) + m_{12} \tag{2}$$

$$M_{21} = k_{21} \cdot (2 \cdot \varphi_2 + \varphi_1 - 3 \cdot \psi) + m_{21} \tag{3}$$

$$k_{12} = k_{21} = \frac{2 \cdot E \cdot I_{12}}{l_{12}} \tag{4}$$

2.3.3. Analiza modela okvira pojednostavljene brodske konstrukcije

Pri analizi okvira pontona zanemarit ćemo linearne pomake koji bi uzrokovali kutove nagiba nosača, odnosno pretpostavit ćemo da je $\psi = 0$.

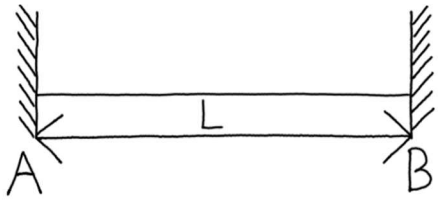
S toga se jednadžbe (2) i (3) mogu zapisati u obliku:

$$M_{12} = k_{12} \cdot (2 \cdot \varphi_1 + \varphi_2) + m_{12} \tag{5}$$

$$M_{21} = k_{21} \cdot (2 \cdot \varphi_2 + \varphi_1) + m_{21} \tag{6}$$

Izrazi i konvekcija smjera momenata m_{ij} :

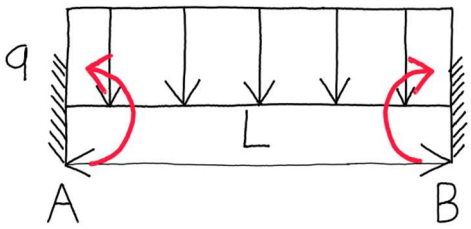
$$m_{A\_B} = m_{B\_A} = 0$$



Slika 9. Upeta greda bez opterećenja

$$m_{A\_B} = \frac{-q \cdot L^2}{12} \quad (7)$$

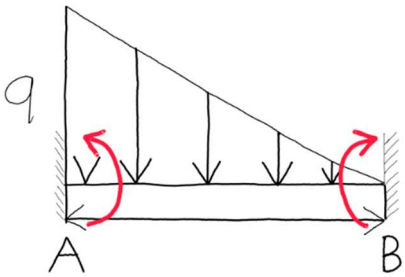
$$m_{B\_A} = \frac{q \cdot L^2}{12} \quad (8)$$



Slika 10. Upeta greda s konstantnim opterećenjem

$$m_{A\_B} = \frac{-q \cdot L^2}{20} \quad (9)$$

$$m_{B\_A} = \frac{q \cdot L^2}{30} \quad (10)$$



Slika 11. Upeta greda s konstantnim trokutnim opterećenjem

Konvencija korištena u ovom radu je to da je pozitivni smjer momenata u smjeru kazaljke na satu.

2.4. Metoda konačnih elemenata

Klasične metode rješavanja problema kontinuiranih sustava temelje se na rješavanju diferencijalnih jednadžbi čije je točno analitičko rješenje moguće dobiti samo za jednostavnije proračunske modele. U općem slučaju vrlo je teško dobiti rješenje koje zadovoljava diferencijalnu jednadžbu u cijelom području razmatranog modela. Stoga se rabe približne numeričke metode koje se temelje na diskretizaciji kontinuiranog sustava gdje se diferencijalne jednadžbe zamjenjuju sustavom algebarskih jednadžbi. [6]

Metoda konačnih elemenata numerička je metoda koja se temelji na fizičkoj diskretizaciji kontinuuma. Razmatrani kontinuum s beskonačno stupnjeva slobode gibanja zamjenjuje se s diskretnim modelom međusobno povezanih elemenata s ograničenim brojem stupnjeva slobode. Drugim riječima, područje kontinuuma dijeli se na konačan broj podpodručja koja se nazivaju konačni elementi, odnosno razmatrani kontinuum postaje mreža konačnih elemenata. Ti konačni elementi međusobno su povezani u točkama na konturi koje se nazivaju čvorovi. [6]

3. IMPLEMENTACIJA PRORAČUNA OKVIRA METODOM POMAKA

Python je interpreterski, interaktivni, objektno orijentirani programski jezik. Njegova filozofija dizajna naglašava čitljivost koda uz korištenje značajnog uvlačenja. On omogućuje programeru da više razmišlja o problemu koji ima nego o jeziku. On je nešto između tradicionalnih skriptnih jezika (kao što su Tcl, Schema i Perl) i sistemskih jezika (kao što su C, C++ i Java). To znači da nudi jednostavnost i lako korištenje skriptnih jezika, uz napredne programerske alate koji se tipično nalaze u sistemskim razvojnim jezicima. Python je besplatan (za akademske ustanove i neprofitnu upotrebu), open-source softver. Osim standardnih tipova podataka (brojevi, nizovi znakova i sl.) python ima ugrađene tipove podataka visoke razine kao što su liste, n-terci i rječnici. Python nudi sve značajke očekivane u modernom programskom jeziku: objektno orijentirano programiranje s višestrukim nasljeđivanjem, dohvaćanje izuzetaka, redefiniranje standardnih operatora, pretpostavljene argumente, prostore imena i pakete.[5]

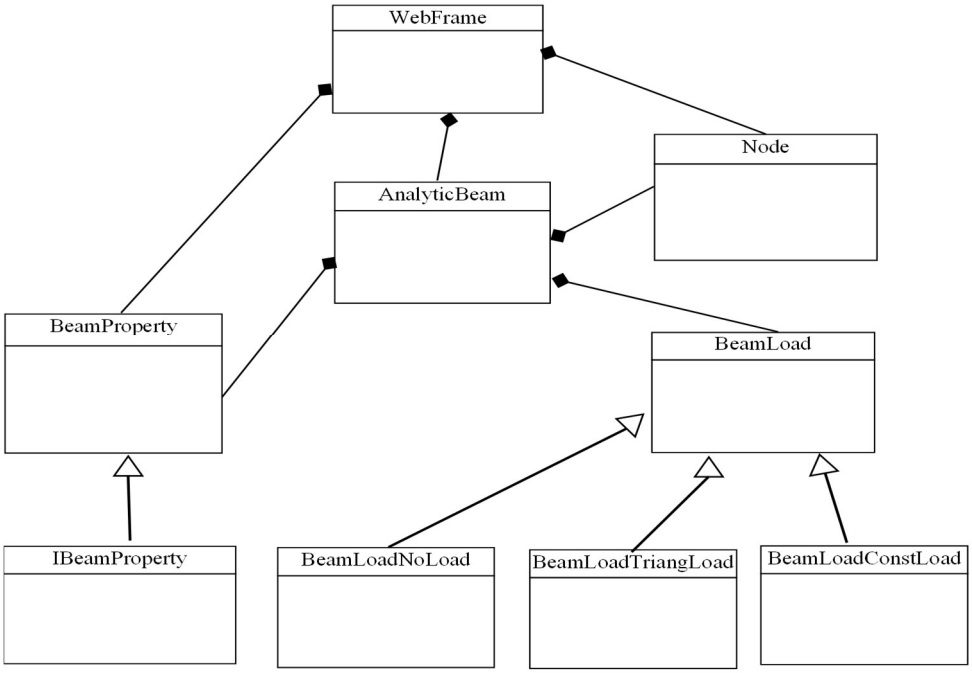
3.1. Pristup pisanju koda

Način pisanja koda u Pythonu je bio objektno orijentiran. Objektno-orijentirano programiranje (OOP) je programska paradigma bazirana na konceptu objekata koji mogu sadržavati podatke u obliku polja, znanih kao atributi i procedura, znanih kao metode. Procedure mogu pristupiti i modificirati objekte. Programi opisani OOP-em dizajnirani su tako da su objekti u stalnoj interakciji jedni s drugima. Ima značajnih razlika u objektno-orijentiranim programskim jezicima, ali najpopularniji su bazirani na klasama, što znači da su objekti neki oblik klasa koje određuju njihov tip.

3.2. Modeliranje metode pomaka u Python-u

Kao što je rečeno korišteno je objektno-orijentirano programiranje tj. više različitih klasa (*classa*) koje su međusobno povezana da bi se dobio jednostavniji kod. Unutar svake klase se nalaze različite metode koje definiraju što ta klasa radi i kome mora slati podatke koji su u njoj spremljeni. Klasa *AnalyticBeam* sadržava metode koje joj omogućuju nalaženje kritičnog momenta savijanja na gredi a samim time i računanje kritičnog naprezanja, klasa *IBeamProperty* sadržava sve podatke vezane uz poprečni presjek grede te računa njeno težište, moment tromosti i moment otpora, klasa *BeamLoad* je ta koja sadržava opterećenje koje je karakteristično za svaku gredu, klasa *Node* je ona u kojoj se nalaze svi čvorovi našeg modela a ti čvorovi služe za određivanje koja je koja greda, a u istom trenu računa duljinu grede na temelju koordinata dvaju čvorova koji čine gredu, klasu *WebFrame* sačinjavaju naše grede koje

se na temelju svojih node-ova spajaju i čine naš model te se unutar ove klase provodi proračun metodom pomaka. Ovisnosti klasa prikazane su sljedećim dijagramom.



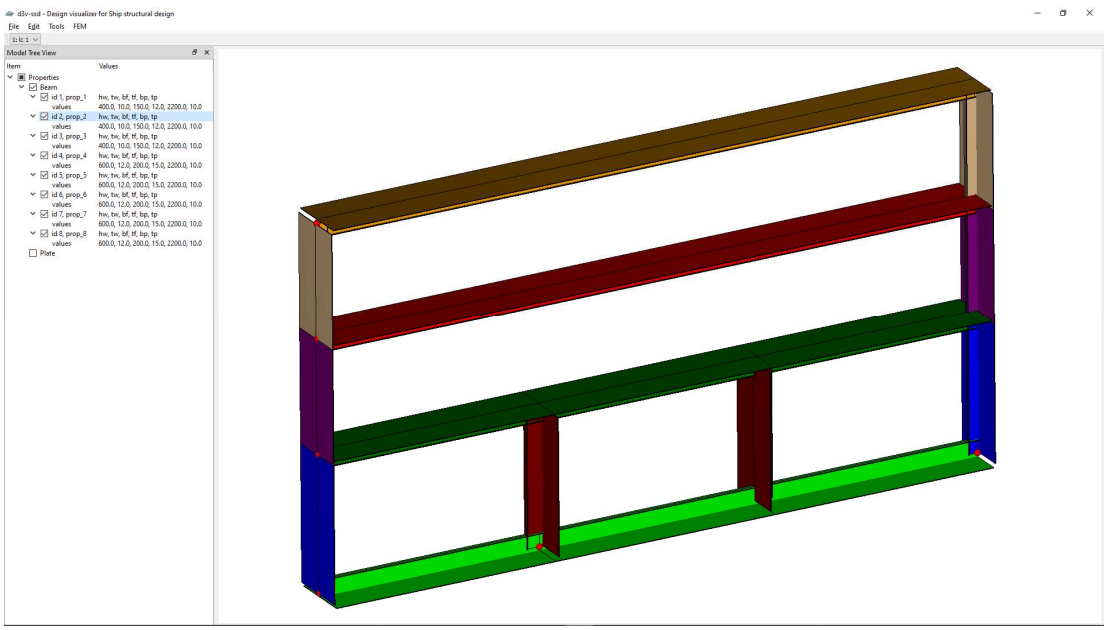
Slika 12. Uml dijagram klasa

4. MODEL OKVIRA JEDNOSTAVNE BRODSKE KONSTRUKCIJE

Korišten je model okvirnog rebra brodske konstrukcije. Okviri u statičkim brodograditeljskim proračunima najčešće nisu samostalni konstruktivni elementi, nego su čvrsto povezani sa ostalom brodskom konstrukcijom. Npr. okviri zajedno s poprečnim pregradama osiguravaju poprečnu čvrstoću, a sastoje se od rebrenica, rebara i sponja, povezani su međusobno i s poprečnim pregradama pomoću oplate paluba, vanjske oplate i uzdužnih nosača (proveza ili podveza) u jednu konstruktivnu cjelinu.[3]

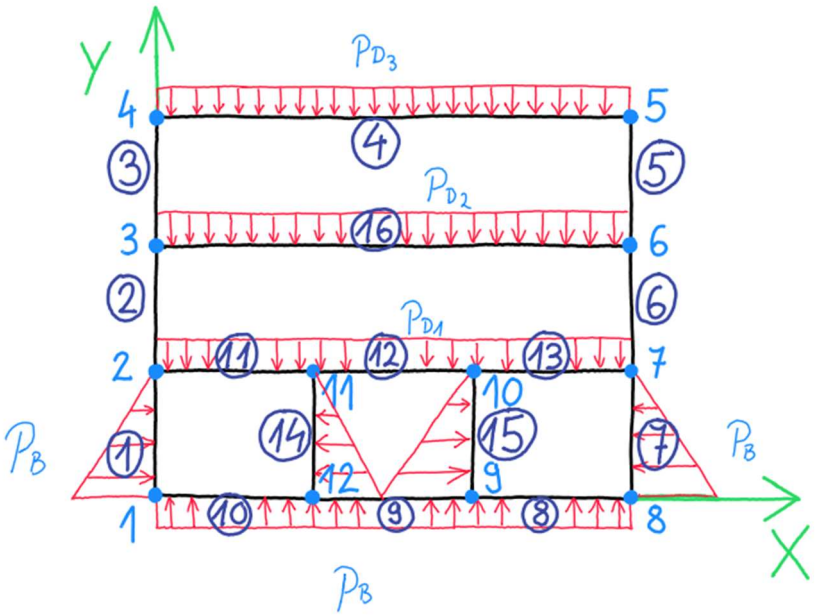
U statičkoj analizi ograničavamo se na 2D model, izdvajamo razmatrani okvir od preostalog dijela konstrukcije. Preostali dio konstrukcije zamjenjujemo rubnim uvjetima pomaka i/ili sila.

Korišten je model sa slike xx. Model je sljedećih dimenzija B = 24 m, D = 12 m. Grede od kojih je sastavljen su T profili dimenzija 600x12/200x15 i 400x10/150x12 ti profili su prikazani slikom 13.



Slika 1311. Model okvira jednostavne brodske konstrukcije

Opterećenja koja su zadana na modelu su u ovom konkretnom modelu interpretirana kao vanjski tlakovi koji djeluju na palubu 3 (p_{D3} = 0.02 N/mm2), palubu 2 (p_{D2} = 0.02), palubu 1 (p_{D1} = 0.13 N/mm2) i dno (p_B = 0.0483 N/mm2) okvira te je njihov raspored kao i raspored i numeriranje čvorova vidljiv slikom 14.



Slika 14. Raspored opterećenja i čvorova na modelu

5. OPTIMIZACIJA OKVIRNOG REBRA JEDNOSTAVNE BRODSKE KONSTRUKCIJE

Za optimizaciju okvirnog rebra korištena je biblioteka SciPy koja sadrži različite module za optimizaciju kako je navedeno u 1. poglavlju. U ovom radu korišten je modul SciPy.Optimize. koji sadržava nekoliko različitih metoda za minimizaciju (maksimizaciju) optimizacijskih problema s ograničenjima. Uključuje rješavače za nelinearne probleme (s podrškom za lokalne i globalne algoritme optimizacije), linearno programiranje, ograničene i nelinearne najmanje kvadrate, pronalaženje korijena i prilagođavanje krivulje. [1]

5.1. Formulacija optimizacijskog problema

Da bi SciPy.Optimize mogao nešto optimizirati moramo mu zadat varijable po kojima će iterirati, neka ograničenja na tim varijablama te jasni cilj do kojeg algoritam mora doći. Nakon zadavanja opterećenja i izračuna $\sigma_{x\text{kritično}}$ za svaku gredu cilj je bio spustiti tu $\sigma_{xkritično}$ ispod vrijednosti $\sigma_{xdopušteno}$ za svaku gredu uz to da nam masa konstrukcije bude što manja.

5.1.1. Funkcija cilja

Cilj ove optimizacije je minimizacija mase. Kod metoda koje smo koristili za optimizaciju želimo da nam funkcija cilja bude približno oko 1, zato smo provjeru optimizacije radili s normiranom masom čija vrijednost je prikazana u tablicama rješenja ove optimizacije.

5.1.2. Projektne varijable

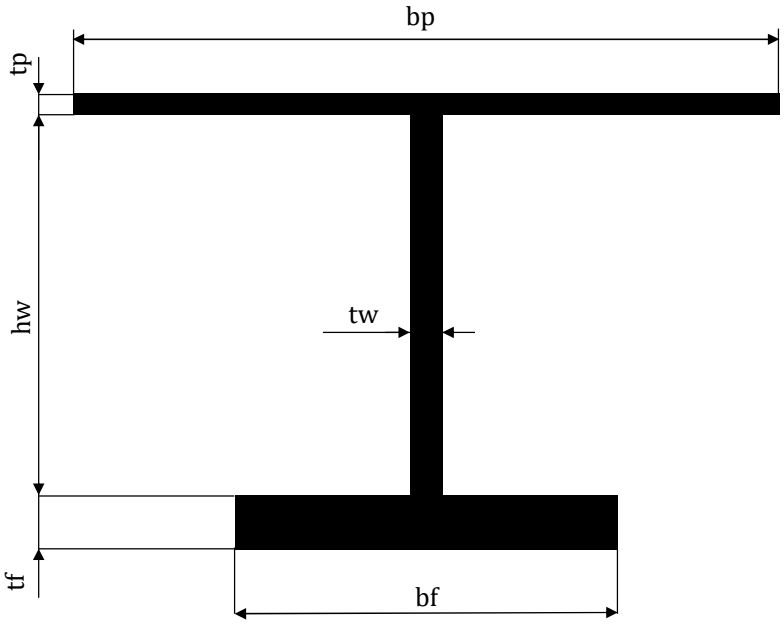
Varijable su dimenzije poprečnog presjeka (T profila) sa sunosivom širinom bp = 2000 mm i tp = 10 mm te one imaju neke svoje minimalne i maksimalne vrijednosti. Te su varijable kontinuirane.

$100 \leq \text{bf} \leq 2000$

$5 \leq \text{tf} \leq 30$

$200 \leq \text{hw} \leq 2000$

$5 \leq \text{tw} \leq 25$



Slika 15. T profil s dodanom sunosivom širinom i debljinom oplate

5.1.3. Ograničenja

Ograničenja koja postavljamo optimizaciji su odnosi između tih dimenzija i ona su redom

$h_s/t_s < 90$,

$4 < b_f/t_f < 25$,

$0{,}2 < b_f/h_s < 0{,}5$,

$1 < t_f/t_s < 3$

Uvjet za zadovoljavanje čvrstoće je $\sigma_{x\text{kritično}} \leq$ da $\sigma_{xdopušteno}$ koje je jednako 160 N/mm^2.

Ukupni broj takvih ograničenja je 72 a oni su prikazani u tablici 1.

Tablica 1. Ograničenja

Num constraints:		
72		
IBPR_1_hw_tw_ub = 40.0 < 90.0	IBPR_2_hw_tw_ub = 40.0 < 90.0	IBPR_3_hw_tw_ub = 40.0 < 90.0
IBPR_1_bf_tf_ub = 12.5 < 25	IBPR_2_bf_tf_ub = 12.5 < 25	IBPR_3_bf_tf_ub = 12.5 < 25
IBPR_1_bf_tf_lb = 12.5 > 4.0	IBPR_2_bf_tf_lb = 12.5 > 4.0	IBPR_3_bf_tf_lb = 12.5 > 4.0
IBPR_1_bf_hw_ub = 0.375 < 0.5	IBPR_2_bf_hw_ub = 0.375 < 0.5	IBPR_3_bf_hw_ub = 0.375 < 0.5
IBPR_1_bf_hw_lb = 0.375 > 0.2	IBPR_2_bf_hw_lb = 0.375 > 0.2	IBPR_3_bf_hw_lb = 0.375 > 0.2
IBPR_1_tf_tw_ub = 1.2 < 3.0	IBPR_2_tf_tw_ub = 1.2 < 3.0	IBPR_3_tf_tw_ub = 1.2 < 3.0
IBPR_1_tf_tw_lb = 1.2 > 1.0	IBPR_2_tf_tw_lb = 1.2 > 1.0	IBPR_3_tf_tw_lb = 1.2 > 1.0

IBPR_4_hw_tw_ub = 50.0 < 90.0	IBPR_5_hw_tw_ub = 50.0 < 90.0	IBPR_6_hw_tw_ub = 50.0 < 90.0
IBPR_4_bf_tf_ub = 13.334 < 25	IBPR_5_bf_tf_ub = 13.33 < 25	IBPR_6_bf_tf_ub = 13.34 < 25
IBPR_4_bf_tf_lb = 13.34 > 4.0	IBPR_5_bf_tf_lb = 13.34 > 4.0	IBPR_6_bf_tf_lb = 13.34 > 4.0
IBPR_4_bf_hw_ub = 0.333 < 0.5	IBPR_5_bf_hw_ub = 0.33 < 0.5	IBPR_6_bf_hw_ub = 0.33 < 0.5
IBPR_4_bf_hw_lb = 0.33 > 0.2	IBPR_5_bf_hw_lb = 0.33 > 0.2	IBPR_6_bf_hw_lb = 0.33 > 0.2
IBPR_4_tf_tw_ub = 1.25 < 3.0	IBPR_5_tf_tw_ub = 1.25 < 3.0	IBPR_6_tf_tw_ub = 1.25 < 3.0
IBPR_4_tf_tw_lb = 1.25 > 1.0	IBPR_5_tf_tw_lb = 1.25 > 1.0	IBPR_6_tf_tw_lb = 1.25 > 1.0
IBPR_7_hw_tw_ub = 40.0 < 90.0	IBPR_8_hw_tw_ub = 50.0 < 90.0	BE_1_Sx_max = 242.77 < 160.0
IBPR_7_bf_tf_ub = 12.5 < 25	IBPR_8_bf_tf_ub = 13.34 < 25	BE_2_Sx_max = 589.95 < 160.0
IBPR_7_bf_tf_lb = 12.5 > 4.0	IBPR_8_bf_tf_lb = 13.34 > 4.0	BE_3_Sx_max = 1400.41< 160.0
IBPR_7_bf_hw_ub = 0.375 < 0.5	IBPR_8_bf_hw_ub = 0.33 < 0.5	BE_4_Sx_max = 558.48 < 160.0
IBPR_7_bf_hw_lb = 0.375 > 0.2	IBPR_8_bf_hw_lb = 0.33 > 0.2	BE_5_Sx_max = 1400.41 < 160.0
IBPR_7_tf_tw_ub = 1.2 < 3.0	IBPR_8_tf_tw_ub = 1.25 < 3.0	BE_6_Sx_max = 589.95 < 160.0
IBPR_7_tf_tw_lb = 1.2 > 1.0	IBPR_8_tf_tw_lb = 1.25 > 1.0	BE_7_Sx_max = 242.77 < 160.0

BE_8_Sx_max = 235.87119771869928 < 160.0
BE_9_Sx_max = 196.25655971453298 < 160.0
BE_10_Sx_max = 235.87119771869928 < 160.0
BE_11_Sx_max = 565.2398995013722 < 160.0
BE_12_Sx_max = 522.5102277279583 < 160.0
BE_13_Sx_max = 565.2398995013722 < 160.0
BE_14_Sx_max = 107.14574435794533 < 160.0
BE_15_Sx_max = 107.14574435794535 < 160.0
BE_16_Sx_max = 644.1803611397598 < 160.0

5.2. Rješavanje optimizacijskog problema

SciPy.Optimize nudi nekoliko metoda za rješavanje ovakvih problema u ovom radu su uzete dvije metode COBYLA i SLSQP. COBYLA je zapravo ograničena optimizacija linearnom aproksimacijom, a temelji se na linearnim aproksimacijama ciljne funkcije i svakog ograničenja. Metoda obavija FORTRAN implementaciju algoritma. SLSQP koristi sekvencijalno programiranje najmanjih kvadrata za minimiziranje funkcije nekoliko varijabli s bilo kojom kombinacijom ograničenja, jednakosti i nejednakosti.[1] Važna je stvar način zadavanja prvog rješenja algoritmu tj. korisnik zadaje vrijednosti ili je stavljeno da program sam kreće on nekog slučajnog rješenja.

5.3. Rezultati optimizacije

Obje korištene optimizacijske metode su uspješno provele optimizaciju kad je za početno rješenje korištena inicijalni model konstrukcije. Metoda SLSQP je uspjela zadovoljiti ograničenja koja su stavljena, ali sa stajališta cilja, tj. mase okvira, koja je prevelika. Ova bi metoda trebala više vremena kako bi zadovolji krajnji cilj ove optimizacije. Metoda COBYLA je bila puno uspješnija i u puno kraćem roku je uspjela zadovoljiti sva ograničenja te je masa okvira manja od one koja se dobije metodom SLSQP. Ovi rezultati su dobiveni kao što se reklo u odlomku prije zadavanjem početnog rješenja, ali da nije bilo tako, da se kretalo od nekog proizvoljnog rješenja kojeg si metoda uzme oba dvije metode bi vratile slične rezultate mase, ali bi im trebalo puno više vremena za te nama nezadovoljavajuće rezultate. Rezultati dvaju metoda s konkretnim početnim rješenjem su prikazana sljedećim tablicama.

Tablica 2. Dimenzije poprečnog presjeka profila, masa okvira i vrijeme rada za metodu COBYLA

Optimization time: 36.1667786 seconds	
Frame mass = 38630.963501832266	
Num variables: 32	
IBP_1_hw = 409.19567669736665	IBP_2_hw = 402.23681844308874
IBP_1_tw = 5.0	IBP_2_tw = 7.9357869848528715
IBP_1_bf = 152.23491978182145	IBP_2_bf = 184.16327675247723
IBP_1_tf = 14.168125895601829	IBP_2_tf = 23.807360954675996
IBP_3_hw = 855.7185895425777	IBP_4_hw = 874.4789914349476
IBP_3_tw = 9.934042008365724	IBP_4_tw = 9.716433238138624
IBP_3_bf = 427.8592939743467	IBP_4_bf = 433.61156014401644
IBP_3_tf = 29.80212602515892	IBP_4_tf = 28.608616603690532
IBP_5_hw = 641.2077245823554	IBP_6_hw = 842.0281269542891
IBP_5_tw = 7.884002725960401	IBP_6_tw = 9.355868077237576
IBP_5_bf = 237.19078528584757	IBP_6_bf = 418.3691700548866
IBP_5_tf = 23.65200817803028	IBP_6_tf = 27.683010927896703

IBP_7_hw = 395.2481701518094	IBP_8_hw = 880.1124674319001
IBP_7_tw = 5.0	IBP_8_tw = 9.779027415910553
IBP_7_bf = 141.36927056714947	IBP_8_bf = 437.08693554692434
IBP_7_tf = 9.040169492724528	IBP_8_tf = 28.724675630930406

Tablica 3. Rezultati ograničenja i normirana masa za metodu COBYLA

Num constraints:
72

IBPR_1_hw_tw_ub = 81.83913533947333 < 90.0	IBPR_2_hw_tw_ub = 50.68644347571864 < 90.0
IBPR_1_bf_tf_ub = 10.744887566892618 < 25	IBPR_2_bf_tf_ub = 7.735560321157973 < 25
IBPR_1_bf_tf_lb = 10.744887566892618 > 4.0	IBPR_2_bf_tf_lb = 7.735560321157973 > 4.0
IBPR_1_bf_hw_ub = 0.37203452639215323 < 0.5	IBPR_2_bf_hw_ub = 0.4578478853957372 < 0.5
IBPR_1_bf_hw_lb = 0.37203452639215323 > 0.2	IBPR_2_bf_hw_lb = 0.4578478853957372 > 0.2
IBPR_1_tf_tw_ub = 2.8336251791203657 < 3.0	IBPR_2_tf_tw_ub = 3.0000000000147913 < 3.0
IBPR_1_tf_tw_lb = 2.8336251791203657 > 1.0	IBPR_2_tf_tw_lb = 3.0000000000147913 > 1.0
IBPR_3_hw_tw_ub = 86.14002123425229 < 90.0	IBPR_4_hw_tw_ub = 90.00000000025436 < 90.0
IBPR_3_bf_tf_ub = 14.356670178937852 < 25	IBPR_4_bf_tf_ub = 15.156676960328108 < 25
IBPR_3_bf_tf_lb = 14.356670178937852 > 4.0	IBPR_4_bf_tf_lb = 15.156676960328108 > 4.0
IBPR_3_bf_hw_ub = 0.4999999990686867 < 0.5	IBPR_4_bf_hw_ub = 0.49585131763142276 < 0.5
IBPR_3_bf_hw_lb = 0.4999999990686867 > 0.2	IBPR_4_bf_hw_lb = 0.49585131763142276 > 0.2
IBPR_3_tf_tw_ub = 3.000000000006216 < 3.0	IBPR_4_tf_tw_ub = 2.9443537461253717 < 3.0
IBPR_3_tf_tw_lb = 3.000000000006216 > 1.0	IBPR_4_tf_tw_lb = 2.9443537461253717 > 1.0
IBPR_5_hw_tw_ub = 81.33022613893702 < 90.0	IBPR_6_hw_tw_ub = 90.00000000031075 < 90.0
IBPR_5_bf_tf_ub = 10.02835714838657 < 25	IBPR_6_bf_tf_ub = 15.112849218048312 < 25
IBPR_5_bf_tf_lb = 10.02835714838657 > 4.0	IBPR_6_bf_tf_lb = 15.112849218048312 > 4.0
IBPR_5_bf_hw_ub = 0.36991255125683264 < 0.5	IBPR_6_bf_hw_ub = 0.49685890133881294 < 0.5
IBPR_5_bf_hw_lb = 0.36991255125683264 > 0.2	IBPR_6_bf_hw_lb = 0.49685890133881294 > 0.2
IBPR_5_tf_tw_ub = 3.000000000018909 < 3.0	IBPR_6_tf_tw_ub = 2.9588928252684834 < 3.0
IBPR_5_tf_tw_lb = 3.000000000018909 > 1.0	IBPR_6_tf_tw_lb = 2.9588928252684834 > 1.0
IBPR_7_hw_tw_ub = 79.04963403036189 < 90.0	IBPR_8_hw_tw_ub = 89.99999999999491 < 90.0
IBPR_7_bf_tf_ub = 15.637900448760675 < 25	IBPR_8_bf_tf_ub = 15.216427198790509 < 25
IBPR_7_bf_tf_lb = 15.637900448760675 > 4.0	IBPR_8_bf_tf_lb = 15.216427198790509 > 4.0
IBPR_7_bf_hw_ub = 0.35767216964686127 < 0.5	IBPR_8_bf_hw_ub = 0.49662622871632545 < 0.5

IBPR_7_bf_hw_lb = 0.35767216964686127 > 0.2	IBPR_8_bf_hw_lb = 0.49662622871632545 > 0.2
IBPR_7_tf_tw_ub = 1.8080338985449056 < 3.0	IBPR_8_tf_tw_ub = 2.937375508754085 < 3.0
IBPR_7_tf_tw_lb = 1.8080338985449056 > 1.0	IBPR_8_tf_tw_lb = 2.937375508754085 > 1.0
BE_1_Sx_max = 160.00000000029215 < 160.0	BE_9_Sx_max = 132.862917351635 < 160.0
BE_2_Sx_max = 160.00000000005087 < 160.0	BE_10_Sx_max = 160.00000000009737 < 160.0
BE_3_Sx_max = 160.00000000003783 < 160.0	BE_11_Sx_max = 160.00000441214434 < 160.0
BE_4_Sx_max = 160.00000000006855 < 160.0	BE_12_Sx_max = 150.9844827720813 < 160.0
BE_5_Sx_max = 160.00000000003783 < 160.0	BE_13_Sx_max = 160.00000000009112 < 160.0
BE_6_Sx_max = 160.0000000000509 < 160.0	BE_14_Sx_max = 160.00000000012656 < 160.0
BE_7_Sx_max = 160.00000000029215 < 160.0	BE_15_Sx_max = 160.00000000012656 < 160.0
BE_8_Sx_max = 160.00000000009737 < 160.0	BE_16_Sx_max = 160.00000000005326 < 160.0

Num objectives: 1
Normed frame mass =
0.7726192700366453

Tablica 4.Dimenzije poprečnog presjeka profila, masa okvira i vrijeme rada za metodu SLSQP

Optimization time: 653.6531675 seconds
Frame mass =
38734.684843564
Num variables: 32

IBP_1_hw = 400.0	IBP_2_hw = 423.66983175690837
IBP_1_tw = 5.105381723750323	IBP_2_tw = 8.034617447676627
IBP_1_bf = 155.6048905123899	IBP_2_bf = 166.98915131215972
IBP_1_tf = 15.316145171251206	IBP_2_tf = 24.10385234282977
IBP_3_hw = 873.1223370355359	IBP_4_hw = 978.1458411666771
IBP_3_tw = 9.817020082791263	IBP_4_tw = 10.868287123984594
IBP_3_bf = 418.15189856947546	IBP_4_bf = 480.1628783722321
IBP_3_tf = 29.451060248374162	IBP_4_tf = 21.11932196816725

IBP_5_hw = 606.9636357877121	IBP_6_hw = 893.8953517087468
IBP_5_tw = 9.399765141888484	IBP_6_tw = 9.93217057456162
IBP_5_bf = 207.61560003272626	IBP_6_bf = 446.09440264690636
IBP_5_tf = 28.199295425665465	IBP_6_tf = 23.703994230070265
IBP_7_hw = 394.2957005264713	IBP_8_hw = 962.4334093389247
IBP_7_tw = 5.0	IBP_8_tw = 10.693704548281111
IBP_7_bf = 146.8235764661131	IBP_8_bf = 481.2167046694349
IBP_7_tf = 8.793951792640415	IBP_8_tf = 22.322095170906035

Tablica 5.Rezultati ograničenja i normirana masa za metodu SLSQP

Num constraints: 72	
IBPR_1_hw_tw_ub = 78.34869587502011 < 90.0	IBPR_2_hw_tw_ub = 52.73055431897646 < 90.0
IBPR_1_bf_tf_ub = 10.15953353618404 < 25	IBPR_2_bf_tf_ub = 6.9279030147160015 < 25
IBPR_1_bf_tf_lb = 10.15953353618404 > 4.0	IBPR_2_bf_tf_lb = 6.9279030147160015 > 4.0
IBPR_1_bf_hw_ub = 0.3890122262809747 < 0.5	IBPR_2_bf_hw_ub = 0.39414926151261603 < 0.5
IBPR_1_bf_hw_lb = 0.3890122262809747 > 0.2	IBPR_2_bf_hw_lb = 0.39414926151261603 > 0.2
IBPR_1_tf_tw_ub = 3.000000000000046 < 3.0	IBPR_2_tf_tw_ub = 2.999999999975094 < 3.0
IBPR_1_tf_tw_lb = 3.000000000000046 > 1.0	IBPR_2_tf_tw_lb = 2.999999999975094 > 1.0
IBPR_3_hw_tw_ub = 88.93965069563981 < 90.0	IBPR_4_hw_tw_ub = 90.00000000074195 < 90.0
IBPR_3_bf_tf_ub = 14.198195074914473 < 25	IBPR_4_bf_tf_ub = 22.735714673793623 < 25
IBPR_3_bf_tf_lb = 14.198195074914473 > 4.0	IBPR_4_bf_tf_lb = 22.735714673793623 > 4.0
IBPR_3_bf_hw_ub = 0.4789155892966884 < 0.5	IBPR_4_bf_hw_ub = 0.4908908857594497 < 0.5
IBPR_3_bf_hw_lb = 0.4789155892966884 > 0.2	IBPR_4_bf_hw_lb = 0.4908908857594497 > 0.2
IBPR_3_tf_tw_ub = 3.000000000000038 < 3.0	IBPR_4_tf_tw_ub = 1.9432061121720128 < 3.0
IBPR_3_tf_tw_lb = 3.000000000000038 > 1.0	IBPR_4_tf_tw_lb = 1.9432061121720128 > 1.0
IBPR_5_hw_tw_ub = 64.57221288251979 < 90.0	IBPR_6_hw_tw_ub = 89.99999999981887 < 90.0
IBPR_5_bf_tf_ub = 7.362439270158709 < 25	IBPR_6_bf_tf_ub = 18.81937695044672 < 25
IBPR_5_bf_tf_lb = 7.362439270158709 > 4.0	IBPR_6_bf_tf_lb = 18.81937695044672 > 4.0
IBPR_5_bf_hw_ub = 0.34205607682457706 < 0.5	IBPR_6_bf_hw_ub = 0.49904544395959105 < 0.5
IBPR_5_bf_hw_lb = 0.34205607682457706 > 0.2	IBPR_6_bf_hw_lb = 0.49904544395959105 > 0.2
IBPR_5_tf_tw_ub = 3.0000000000000013 < 3.0	IBPR_6_tf_tw_ub = 2.3865875089561173 < 3.0
IBPR_5_tf_tw_lb = 3.0000000000000013 > 1.0	IBPR_6_tf_tw_lb = 2.3865875089561173 > 1.0

IBPR_7_hw_tw_ub = 78.85914010529426 < 90.0
IBPR_7_bf_tf_ub = 16.695972405601374 < 25
IBPR_7_bf_tf_lb = 16.695972405601374 > 4.0
IBPR_7_bf_hw_ub = 0.3723692048127114 < 0.5
IBPR_7_bf_hw_lb = 0.3723692048127114 > 0.2
IBPR_7_tf_tw_ub = 1.758790358528083 < 3.0
IBPR_7_tf_tw_lb = 1.758790358528083 > 1.0

IBPR_8_hw_tw_ub = 89.99999999940383 < 90.0
IBPR_8_bf_tf_ub = 21.557864572526267 < 25
IBPR_8_bf_tf_lb = 21.557864572526267 > 4.0
IBPR_8_bf_hw_ub = 0.49999999999997147 < 0.5
IBPR_8_bf_hw_lb = 0.49999999999997147 > 0.2
IBPR_8_tf_tw_ub = 2.0874052644828356 < 3.0

IBPR_8_tf_tw_lb = 2.0874052644828356 > 1.0

BE_1_Sx_max = 159.9999999885896 < 160.0
BE_2_Sx_max = 160.00000000804823 < 160.0
BE_3_Sx_max = 160.0000000029002 < 160.0
BE_4_Sx_max = 159.99999999368245 < 160.0
BE_5_Sx_max = 160.0000000029002 < 160.0
BE_6_Sx_max = 160.00000000804823 < 160.0
BE_7_Sx_max = 159.9999999885896 < 160.0
BE_8_Sx_max = 160.00000000274437 < 160.0

BE_9_Sx_max = 132.6914961991004 < 160.0
BE_10_Sx_max = 160.00000000274437 < 160.0

BE_11_Sx_max = 159.99999999748664 < 160.0
BE_12_Sx_max = 150.31782275888898 < 160.0

BE_13_Sx_max = 159.1613801578943 < 160.0
BE_14_Sx_max = 160.00000000274449 < 160.0

BE_15_Sx_max = 160.00000000274449 < 160.0
BE_16_Sx_max = 159.99999999598296 < 160.0

Num objectives: 1
Normed frame mass = 0.77469369687128

6. ZAKLJUČAK

U ovom radu implementiran je optimizacijski model koji omogućuje optimizaciju okvirnog rebra, pri čemu je za proračun odziva korištena metodom pomaka. Matematički model za proračun okvira metodom pomaka u okviru ovog rada je implementiran je u programskom jeziku Python primjenom objektno orijentiranog programiranja, a za optimizaciju su korištene optimizacijske metode koje su dostupne u biblioteci SciPy.

Izrađeni optimizacijski model testiran je na primjeru optimizacije okvirnog rebra jednostavne brodske konstrukcije. Korištene su dvije optimizacijske metode, SLSQP i COBYLA, koje su dostupne u biblioteci SciPy, a za obje metode je korišteno isto početno rješenje. Obje metode su uspjele zadovoljiti sva ograničenja, no s različitim kombinacijom varijabli te s različitom masom. Pri tome je vrijeme potrebno za provedbu optimizacije je kraće kod metode COBYLA, koja je ujedno pronašla rješenje s manjom masom.

U okviru nastavka rada na ovoj temi mogla bi se provesti detaljnija analizu uspješnosti pojedinog algoritma za što bi bilo potrebno usporediti utjecaja zadavanja nekih drugih početnih rješenja ili odabir nekog slučajnog rješenja kao početnog.

Daljnji razvoj izrađenog optimizacijskom modela mogao bi uključiti rad na automatizaciji standardizacije projektnih varijabli tj. dimenzija profila, omogućavanje analize za više slučajeva opterećenja te proračun neke kompleksnije konstrukcije okvira.

LITERATURA

[1] SciPy, https://scipy.org/

[2] Oreč I (2021), Diplomski rad

[3] Brodogradnja, Hrvatska enciklopedija, mrežno izdanje, Leksikografski zavod Miroslav Krleža, 2021. https://tehnika.lzmk.hr/brodogradnja-casopis/

[4] Uršić J, Čvrstoća broda 2. dio, Sveučilište u Zagrebu, Fakultet Strojarstva i Brodogradnje, Zagreb, 1983.

[5] Python, https://www.python.org/

[6] Sorić, J., Metoda konačnih elemenata, Golden marketing - Tehnička knjiga, Zagreb, 2004

[7] Podloge za nastavu iz kolegija Čvrstoća broda, Preddiplomski studij brodogradnje, Fakultet strojarstva i brodogradnje, Zagreb (dostupno na e-kolegiju Čvrstoća https://e-ucenje.fsb.hr/)

PRILOZI

I. Python skripte frameamalysis.py, frameinput.py, run_frameanalysis.py, run_frameopt.py i tekstualni dokument cb.ponton

```
 1 12 8 16 16
 2 1 0.0 0.0 0.0
 3 2 0.0 4800.0 0.0
 4 3 0.0 8800.0 0.0
 5 4 0.0 12800.0 0.0
 6 5 24000.0 12800.0 0.0
 7 6 24000.0 8800.0 0.0
 8 7 24000.0 4800.0 0.0
 9 8 24000.0 0.0 0.0
10 9 16000.0 0.0 0.0
11 10 16000.0 4800.0 0.0
12 11 8000.0 4800.0 0.0
13 12 8000.0 0.0 0.0
14 1 type I hw 400.0 tw 10.0 bf 150.0 tf 12.0 bp 2200.0 tp 10
   .0 E 204000
15 2 type I hw 400.0 tw 10.0 bf 150.0 tf 12.0 bp 2200.0 tp 10
   .0 E 204000
16 3 type I hw 400.0 tw 10.0 bf 150.0 tf 12.0 bp 2200.0 tp 10
   .0 E 204000
17 4 type I hw 600.0 tw 12.0 bf 200.0 tf 15.0 bp 2200.0 tp 10
   .0 E 204000
18 5 type I hw 600.0 tw 12.0 bf 200.0 tf 15.0 bp 2200.0 tp 10
   .0 E 204000
19 6 type I hw 600.0 tw 12.0 bf 200.0 tf 15.0 bp 2200.0 tp 10
   .0 E 204000
20 7 type I hw 400.0 tw 10.0 bf 150.0 tf 12.0 bp 2200.0 tp 10
   .0 E 204000
21 8 type I hw 600.0 tw 12.0 bf 200.0 tf 15.0 bp 2200.0 tp 10
   .0 E 204000
22 1 1 2 1
23 2 2 3 2
24 3 3 4 3
25 4 4 5 4
26 5 5 6 3
27 6 6 7 2
28 7 7 8 1
29 8 8 9 5
30 9 9 12 5
31 10 12 1 5
32 11 2 11 6
33 12 11 10 6
34 13 10 7 6
35 14 11 12 7
36 15 9 10 7
```

```
37 16 3 6 8
38 1 idbeam 1 type TL q0 106.24 1
39 2 idbeam 2 type CL q0 0.0
40 3 idbeam 3 type CL q0 0.0
41 4 idbeam 4 type CL q0 44.0
42 5 idbeam 5 type CL q0 0.0
43 6 idbeam 6 type CL q0 0.0
44 7 idbeam 7 type TL q0 106.24 2
45 8 idbeam 8 type CL q0 106.24
46 9 idbeam 9 type CL q0 106.24
47 10 idbeam 10 type CL q0 106.24
48 11 idbeam 11 type CL q0 286.0
49 12 idbeam 12 type CL q0 286.0
50 13 idbeam 13 type CL q0 286.0
51 14 idbeam 14 type TL q0 106.24 2
52 15 idbeam 15 type TL q0 106.24 1
53 16 idbeam 16 type CL q0 44.0
54
```

```python
from typing import Dict, List
import numpy as np

class BeamProperty:
    def __init__(self,id):
        self._id = id
    @property
    def id(self):
        return self._id
    def getIy(self):
        pass
    def getArea(self):
        pass
    def get_zna(self):
        pass
    def get_zmax(self):
        pass

    def getWmin(self):
        z_na = self.get_zna()
        z_max = self.get_zmax()
        Iy = self.getIy()
        dzp = z_na
        dzf = z_max - z_na
        dzmax = dzf
        if dzmax < dzp:
            dzmax = dzp
        return Iy / dzmax



class IBeamProperty(BeamProperty):
    def __init__(self, id, hw, tw, bf, tf, bp, tp, E):
        super().__init__(id)
        self._hw = hw
        self._tw = tw
        self._bf = bf
        self._tf = tf
        self._bp = bp
        self._tp = tp
        self._E = E

    def get_hw(self):
        return self._hw
```

```
45      def set_hw(self,value):
46          self._hw = value
47      def get_tw(self):
48          return self._tw
49      def set_tw(self,value):
50          self._tw = value
51      def get_bf(self):
52          return self._bf
53      def set_bf(self,value):
54          self._bf = value
55      def get_tf(self):
56          return self._tf
57      def set_tf(self,value):
58          self._tf = value
59
60      def getArea(self):
61          hw = self._hw
62          tw = self._tw
63          bf = self._bf
64          tf = self._tf
65          bp = self._bp
66          tp = self._tp
67          A = bf * tf + hw * tw + bp * tp
68          return A
69
70      def get_zna(self):
71          hw = self._hw
72          tw = self._tw
73          bf = self._bf
74          tf = self._tf
75          bp = self._bp
76          tp = self._tp
77          A = self.getArea()
78          z_na = ((bp * tp) * tp / 2 + hw * tw * (tp + hw /
    2) + bf * tf * (tp + hw + tf / 2)) / A
79          return z_na
80
81      def get_zmax(self):
82          z_max = self._tp + self._hw + self._tf
83          return z_max
84
85
86
87      def getIy(self):
```

```
 88          hw = self._hw
 89          tw = self._tw
 90          bf = self._bf
 91          tf = self._tf
 92          bp = self._bp
 93          tp = self._tp
 94          z = self.get_zna()
 95          Iy = (bp * tp ** 3) / 12 + ((z - tp / 2) ** 2) *
    bp * tp + (tw * hw ** 3) / 12 + (
 96                  ((tp + hw / 2)-z) ** 2) * tw * hw + (bf
     * tf ** 3) / 12 + (
 97                      ((tp + hw + tf / 2) - z) ** 2) * bf
     * tf
 98          return Iy
 99
100      def getE(self):
101          return self._E
102
103  class BeamLoad:
104      def __init__(self, id: int, q0):
105          self._id = id
106          self._q0 = q0
107
108      def getMx(self, x, L):
109          pass
110
111      def get_xcrit(self, L):
112          pass
113
114      def get_moment_clamped_beam_end(self, L, is2:bool):
115          pass
116
117
118  class BeamLoadNoLoad(BeamLoad):
119      def __init__(self, id: int, q0):
120          super().__init__(id, q0)
121          pass
122
123      def getMx(self, x, L):
124          return 0.0
125
126      def get_xcrit(self, L):
127          return 0.0
128
```

```
129     def get_moment_clamped_beam_end(self, L, is2:bool):
130         return 0.0
131 
132 
133 class BeamLoadConstContLoad(BeamLoad):
134     def __init__(self, id: int, q0):
135         super().__init__(id, q0)
136         pass
137 
138     def getMx(self, x, L):
139         return (self._q0 * x) / 2. * (L - x)
140 
141     def get_xcrit(self, L):
142         return L / 2
143 
144     def get_moment_clamped_beam_end(self, L, is2:bool):
145         return self._q0*L**2.0/12.0
146 
147 
148 class BeamLoadTriangleContLoad(BeamLoad):
149     def __init__(self, id: int, q0: float, is2: bool =
    True):
150         super().__init__(id, q0)
151         self._is2 = is2 # true if q0 is on second node
152 
153     def getMx(self, x, L):
154         W = self._q0 * L / 2.
155         if not self._is2:
156             x = L - x
157         Mx = W * x * (L ** 2.0 - x ** 2.0) / (3.0 * L **
    2.0)
158         return Mx
159 
160     def get_xcrit(self, L):
161         x = 0.5774 * L
162         if not self._is2:
163             x = L - x
164         return x
165 
166     def get_moment_clamped_beam_end(self, L, is2:bool):
167         if (is2 and self._is2) or ((not is2) and (not
    self._is2)):
168             return self._q0*L**2.0 / 20.0
169         else:
```

```
170             return self._q0 * L ** 2.0 / 30.0
171
172
173 class Node:
174     def __init__(self, id: int, x=0.0, y=0.0, z=0.0):
175         self._id = id
176         self._cords: List[float] = np.array([x, y, z])
177
178     @property
179     def id(self):
180         return self._id
181
182     @property
183     def cords(self):
184         return self._cords
185
186     def get_distance(self, node):
187         return np.linalg.norm(node.cords - self._cords)
188
189     @staticmethod
190     def get_distance(node1, node2):
191         return np.linalg.norm(node1.cords - node2.cords)
192
193
194 class AnalyticBeam():
195     def __init__(self, id: int, prop=None, node1=None,
    node2=None):
196         self._id = id
197         self._nodes: List[Node] = []
198         if node1 is not None:
199             self._nodes.append(node1)
200         if node2 is not None:
201             self._nodes.append(node2)
202         self._loads:List[BeamLoad] = []
203         self._prop: IBeamProperty = prop
204         self._M1:float = 0.0 # internal moment on the
    node 1 end calculated by displacement method
205         self._M2:float = 0.0 # internal moment on the
    node 2 end calculated by displacement method
206         pass
207
208     @property
209     def id(self):
210         return self._id
```

```
211
212     @property
213     def prop(self):
214         return self._prop
215
216     @property
217     def L(self):
218         return Node.get_distance(self.node1, self.node2)
219
220     @property
221     def node1(self):
222         return self._nodes[0]
223
224     @property
225     def node2(self):
226         return self._nodes[1]
227
228     def have_node(self, node: Node):
229         if self.node1 is node:
230             return True
231         if self.node2 is node:
232             return True
233         return False
234
235     def add_load(self, load: BeamLoad):
236         self._loads.append(load)
237
238     def set_prop(self, prop: BeamProperty):
239         self._prop = prop
240
241     def set_load(self, load):
242         self._loads = load
243
244
245     def check_beam(self):
246         if len(self._loads) == 0:
247             self.add_load(BeamLoadNoLoad())
248
249     def getSigmax_crit(self):
250         Mcrit = self.getMx_crit()
251         W = self._prop.getWmin()
252         return np.abs(Mcrit / W)
253
254     def getMx_crit(self):
```

```
255         Mxcrit = 0
256         xcand = []
257         for load in self._loads:
258             xcand.append(load.get_xcrit(self.L))
259 
260         for x in xcand:
261             Mx = abs(self.get_Mx(x) - self._M1)
262             if Mxcrit < Mx:
263                 Mxcrit = Mx
264         if Mxcrit < abs(self._M1):
265             Mxcrit = self._M1
266         if Mxcrit < abs(self._M2):
267             Mxcrit = self._M2
268         return Mxcrit
269 
270     def get_Mx(self, x):
271         Mx = 0
272         for load in self._loads:
273             Mx = Mx + load.getMx(x, self.L)
274         return Mx
275 
276     def get_k(self):
277         k = 2 * self._prop.getIy() * self._prop.getE() /
    self.L
278         return k
279 
280     def get_stifness_for_frame(self, current_node:Node):
281         if not self.have_node(current_node):
282             return None
283         k = self.get_k()
284         mb = 0.0
285         if self.node1 is current_node:
286             # node 1 is current, node 2 is other
287             for load in self._loads:
288                 mb -= load.get_moment_clamped_beam_end(
    self.L,False)
289             return (k, mb, self.node2.id)
290         else:
291             # node 2 is current, node 1 is other
292             for load in self._loads:
293                 mb += load.get_moment_clamped_beam_end(
    self.L, True)
294             return (k, mb, self.node1.id)
295 
```

```
296     def calculate_internal_end_moments(self, phi1, phi2):
297         k = self.get_k()
298         # node 1 is current, node 2 is other
299         mb = 0.0
300         for load in self._loads:
301             mb += load.get_moment_clamped_beam_end(self.L
    , False)
302         self._M1 = k * (2 * phi1 + phi2) + mb
303         # node 2 is current, node 1 is other
304         mb = 0.0
305         for load in self._loads:
306             mb += load.get_moment_clamped_beam_end(self.L
    , True)
307         self._M2 = k * (2 * phi2 + phi1) - mb #tu sam
    promjenio  + mb u -mb
308 
309     def get_mass(self):
310         ro = 7.85e-6
311         mass = self._prop.getArea()*self.L*ro
312         return mass
313 
314 
315 
316 class WebFrame():
317     def __init__(self):
318         self._nodes: Dict[int, Node] = {}
319         self._beams: Dict[int, AnalyticBeam] = {}
320         self._props: Dict[int, BeamProperty] = {}
321         self._k = {}
322         self._m = {}
323         self._mass = 0.0
324         pass
325 
326     def analyze(self):
327         self.assemble_stiff_matrix()
328         self.calculate_M()
329         self.calc_mass()
330 
331     def calc_mass(self):
332         mass = 0.0
333         for beam in self._beams.values():
334             mass += beam.get_mass()
335         self._mass = mass
336 
```

```
337     def get_mass(self):
338         return self._mass
339 
340     def get_normed_mass(self):
341         return self._mass/50000
342 
343     def assemble_stiff_matrix(self):
344         null = None
345         self._k.clear()
346         self._m.clear()
347         for idnode,node in self._nodes.items():
348             row_vals = {}
349             m = 0.0
350             for beam in self._beams.values():
351                 if not beam.have_node(node):
352                     continue
353                 res = beam.get_stifness_for_frame(node)
354 
355                 if res is not null:
356                     (k, mb, idother) = res
357                     row_vals[idnode] = row_vals.get(
    idnode,0.0)+ 2.0*k
358                     row_vals[idother] = row_vals.get(
    idother, 0.0) + k
359                     m+=mb
360             if len(row_vals) > 0:
361                 self._k[idnode] = row_vals
362                 self._m[idnode] = m
363 
364     def calculate_phi(self):
365         n = len(self._k)
366         k = np.zeros((n,n))
367         m = np.zeros(n)
368         ids = [-1]*n
369         indexes = {}
370         index = 0
371         for id_row, rowvals in self._k.items():
372             indexes[id_row] = index
373             ids[index] = id_row
374             index += 1
375 
376         for id_row,rowvals in self._k.items():
377             for id_col,kxx in rowvals.items():
378                 irow=indexes[id_row]
```

```
379                 icol = indexes[id_col]
380                 k[irow,icol] = kxx
381 
382         for id_row,mrow in self._m.items():
383             irow = indexes[id_row]
384             m[irow] = mrow
385 
386 
387         phi = np.linalg.solve(k,m)
388         dphi = {}
389         for i in range(n):
390             dphi[ids[i]] = phi[i]
391 
392         return dphi
393 
394     def calculate_M(self):
395         phi = self.calculate_phi()
396         for beam in self._beams.values():
397             beam.calculate_internal_end_moments(phi[beam.
    node1.id], phi[beam.node2.id])
398     def print_Sigmacrit(self):
399         for beam in self._beams.values():
400             sx = beam.getSigmax_crit()
401             print('Beam{0}: Sx_crit={1:6.1f}'.format(beam
    .id, sx))
402 
403     def print_Nodal_Moments(self):
404         for beam in self._beams.values():
405             print('Beam{0}: M1={1:10.3E}, M2={2:10.3E}'.
    format(beam.id,beam._M1,beam._M2))
406 
407 
408 
409     @property
410     def props(self):
411         return self._props
412 
413     @property
414     def nodes(self):
415         return self._nodes
416 
417     @property
418     def beams(self):
419         return self._beams
```

```
420
421     def add_node(self, node: Node):
422         self._nodes[node.id] = node
423
424     def add_prop(self, prop: IBeamProperty):
425         self._props[prop.id] = prop
426
427     def add_beam(self, beam: AnalyticBeam):
428         self._beams[beam.id] = beam
429
430     def get_node(self, idnode: int):
431         return self._nodes[idnode]
432
433     def get_prop(self, idprop: int):
434         return self._props[idprop]
435
436     def get_beam(self, idbeam: int):
437         return self._beams[idbeam]
438
439
```

```
from frameanalysis import *

class ReadFrameDataRecords:
    def __init__(self, file_path: str):
        self._filename = file_path

    def getFrame(self):
        frame = WebFrame()
        self.readfile(frame)
        return frame

    def readfile(self, frame: WebFrame):
        with open(self._filename, 'r') as f:
            lines = f.readlines()
            iline = 0
            line = lines[iline]
            split = line.split(' ')
            n_nodes = int(split[0])  # 4
            n_props = int(split[1])  # 4
            n_elem = int(split[2])  # 4
            n_loads = int(split[3])  # 4

            for i in range(1, n_nodes + 1):  # 1,2,3,4
                line = lines[i]
                split = line.split(' ')
                node = Node(int(split[0]), float(split[1
]), float(split[2]), float(split[3]))
                frame.add_node(node)

            for i in range(n_nodes + 1, n_nodes + n_props
 + 1):  # 5,6,7
                line = lines[i]
                split = line.split(' ')
                fsplit = []
                for idx in range(4, len(split), 2):
                    fsplit.append(float(split[idx]))

                beam_property = IBeamProperty(int(split[0
]), fsplit[0], fsplit[1], fsplit[2], fsplit[3], fsplit[4
], fsplit[5], fsplit[6])
                frame.add_prop(beam_property)


            for i in range(n_nodes + n_props + 1, n_nodes
```

```
 + n_props + n_elem + 1):  # 8,9,10
                line = lines[i]
                split = line.split(' ')
                beam = AnalyticBeam(id=int(split[0]), prop
=frame.get_prop(int(split[3])),
                                    node1=frame.get_node(
int(split[1])), node2=frame.get_node(int(split[2])))
                frame.add_beam(beam)

            for i in range(n_nodes + n_props + n_elem + 1
, n_nodes + n_props + n_elem + n_loads + 1):  # 11, 12, 13
                line = lines[i]
                split = line.split(' ')
                id_beam = int(split[2])
                if split[4] == 'CL':
                    load = BeamLoadConstContLoad(int(split
[0]), float(split[6]))
                elif split[4] == 'TL':
                    is2 = int(split[7])==2
                    load = BeamLoadTriangleContLoad(int(
split[0]), float(split[6]),is2)
                frame.get_beam(id_beam).add_load(load)
```

```
 1 from frameinput import ReadFrameDataRecords
 2 from frameanalysis import WebFrame
 3 import numpy as np
 4
 5 file = 'cb_ponton.txt'
 6 reader = ReadFrameDataRecords(file)
 7 frame = reader.getFrame()
 8 frame.analyze()
 9 frame.print_Nodal_Moments()
10 mass = frame.get_mass()
11 print('Frame mass =',mass)
12 frame.print_Sigmacrit()
13
14 print('Kraj')
```

```
 1 from optextendframe import *
 2 from frameinput import ReadFrameDataRecords
 3 from optlib_scipy import ScipyOptimizationAlgorithm
 4 import time
 5
 6 #dedign variables input data
 7 hw_lb =200.0
 8 hw_ub =2000.0
 9 tw_lb =5.0
10 tw_ub =25.0
11 bf_lb =100.0
12 bf_ub =2000.0
13 tf_lb =5.0
14 tf_ub =30.0
15 # Ratios input data
16 hw_tw_ub = 90.0
17 bf_tf_lb = 4.0
18 bf_tf_ub = 25
19 bf_hw_lb = 0.2
20 bf_hw_ub = 0.5
21 tf_tw_lb = 1.0
22 tf_tw_ub = 3.0
23 # Sigma_x max input
24 Sig_x_max = 160.0
25 # input file name
26 file = 'cb_ponton.txt'
27
28 # Problem implementation
29 reader = ReadFrameDataRecords(file)
30 frame = reader.getFrame()
31 fam = FrameAnayisisModel(frame)
32 op = OptimizationProblem()
33 op.add_analysis_executor(fam)
34 #Add all properties as design variables and ratio
   constraints
35 for prop in frame.props.values():
36     dvhw = DesignVariable('IBP_{0}_hw'.format(prop.id),
   CalbackGetSetConnector(prop.get_hw, prop.set_hw), hw_lb,
   hw_ub)
37     dvtw = DesignVariable('IBP_{0}_tw'.format(prop.id),
   CalbackGetSetConnector(prop.get_tw, prop.set_tw), tw_lb,
   tw_ub)
38     dvbf = DesignVariable('IBP_{0}_bf'.format(prop.id),
   CalbackGetSetConnector(prop.get_bf, prop.set_bf), bf_lb,
```

```
bf_ub)
    dvtf = DesignVariable('IBP_{0}_tf'.format(prop.id),
CalbackGetSetConnector(prop.get_tf, prop.set_tf), tf_lb,
tf_ub)
    op.add_design_varible(dvhw)
    op.add_design_varible(dvtw)
    op.add_design_varible(dvbf)
    op.add_design_varible(dvtf)
    # ratio constraints
    # hw/tw
    hw_tw = RatioDesignConnector(dvhw, dvtw)
    con_hw_tw_ub = DesignConstraint('IBPR_{0}_hw_tw_ub'.
format(prop.id),hw_tw,hw_tw_ub,ConstrType.LT)
    op.add_constraint(con_hw_tw_ub)
    # bf/tf
    bf_tf = RatioDesignConnector(dvbf, dvtf)
    con_bf_tf_ub = DesignConstraint('IBPR_{0}_bf_tf_ub'.
format(prop.id),bf_tf,bf_tf_ub,ConstrType.LT)
    con_bf_tf_lb = DesignConstraint('IBPR_{0}_bf_tf_lb'.
format(prop.id),bf_tf,bf_tf_lb,ConstrType.GT)
    op.add_constraint(con_bf_tf_ub)
    op.add_constraint(con_bf_tf_lb)
    # bf/hw
    bf_hw = RatioDesignConnector(dvbf, dvhw)
    con_bf_hw_ub = DesignConstraint('IBPR_{0}_bf_hw_ub'.
format(prop.id),bf_hw, bf_hw_ub, ConstrType.LT)
    con_bf_hw_lb = DesignConstraint('IBPR_{0}_bf_hw_lb'.
format(prop.id),bf_hw, bf_hw_lb, ConstrType.GT)
    op.add_constraint(con_bf_hw_ub)
    op.add_constraint(con_bf_hw_lb)
    # tf/tw
    tf_tw = RatioDesignConnector(dvtf, dvtw)
    con_tf_tw_ub = DesignConstraint('IBPR_{0}_tf_tw_ub'.
format(prop.id),tf_tw, tf_tw_ub, ConstrType.LT)
    con_tf_tw_lb = DesignConstraint('IBPR_{0}_tf_tw_lb'.
format(prop.id),tf_tw, tf_tw_lb, ConstrType.GT)
    op.add_constraint(con_tf_tw_ub)
    op.add_constraint(con_tf_tw_lb)

for beam in frame.beams.values():
    sxc = CalbackGetConnector(beam.getSigmax_crit)
    con_sx = DesignConstraint('BE_{0}_Sx_max'.format(beam.
id),sxc,Sig_x_max,ConstrType.LT)
    op.add_constraint(con_sx)
```

```
72
73 mc = CalbackGetConnector(frame.get_normed_mass)
74 obj_mass = DesignObjective('Normed frame mass',mc)
75 op.add_objective(obj_mass)
76
77 opt_ctrl = {'maxiter': 100000}
78 #opt_ctrl['method'] = 'COBYLA'
79 opt_ctrl['method'] = 'SLSQP'
80 op.opt_algorithm = ScipyOptimizationAlgorithm(opt_ctrl)
81 x0 = op.get_initial_design(False)
82 #x0 = op.get_ub_design()
83
84 frame.analyze()
85 frame.print_Sigmacrit()
86 print(op.get_info())
87 tStart = time.perf_counter()  # početak mjerenja vremena
88 res = op.optimize(x0)
89 tEnd = time.perf_counter()
90 dt = tEnd - tStart
91 print(res)
92 print('Optimization time: {0} seconds'.format(dt))
93 frame.print_Sigmacrit()
94 mass = frame.get_mass()
95 print('Frame mass =',mass)
96 print(op.get_info())
```