Primjena metode slobodne deformacije oblika pri optimizaciji brodske forme u ranim fazama projktiranja

Pavlović, Tomislav

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Tomislav Pavlović

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Pero Prebeg

Student:

Tomislav Pavlović

Zagreb, 2023.

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

Zahvaljujem se mentoru dr.sc. Peri Prebegu na brojnim konzultacijama, prijedlozima i korekcijama prilikom izrade diplomskog rada, dr.sc. Josipu Bašiću za ustupanje modula michell.py za proračun otpora, Božidaru Šariću za pomoć pri definiranju optimizacijskog problema, te svojoj obitelji i prijateljima na razumijevanju, strpljenju i pomoći tijekom studija.

Tomislav Pavlović

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 - 23 -**	

DIPLOMSKI ZADATAK

Student: **Tomislav Pavlović** JMBAG: 0035202907

Naslov rada na hrvatskom jeziku: **Primjena metode slobodne deformacije oblika pri optimizaciji brodske forme u ranim fazama projektiranja**

Naslov rada na engleskom jeziku: **Application of free-form deformation method for hull form optimization in early design stage**

Opis zadatka:

Pri projektiranju brodske forme najčešće se polazi od postojeće brodske forme koja se odgovarajućim transformacijama prilagodi glavnim izmjerama i zahtijevanim hidrostatičkim karakteristikama (npr. prizmatičkom koeficijentu, uzdužnom položaju težišta istisnine, itd.) novog projekta. Nakon toga može se pristupiti transformaciji brodske forme, najčešće sa ciljem minimizacije potrebne snage porivnog stroja uz održavanje hidrostatičkih karakteristika na zahtijevanim razinama. Jedna od metoda koja se koristi s tom svrhom je metoda slobodne deformacije oblika (*eng: free-form deformation*), koja između ostalog omogućuje i izolirano transformiranje pramčanog i krmenog dijela brodske forme. Program otvorenog koda d3v-gsd (*Design visualizer for General Ship Design*) proširenje je programa linaetal-fsb/d3v, koji omogućuje trodimenzijsku vizualizaciju brodske forme te izračun hidrostatičkih karakteristika i stabiliteta broda. U radu je potrebno, proširenjem funkcionalnosti programa otvorenog koda d3v-gsd u programskom jeziku Python, omogućiti optimizaciju brodske forme primjenom metode slobodne deformacije oblika.

Zadatak obuhvaća sljedeće:

- upoznavanje s trenutnom verzijom programa otvorenog koda d3v-gsd
- upoznavanje s metodom slobodne deformacije oblika i primjenom iste za optimizaciju brodske forme
- upoznavanje s Python bibliotekom PyGeM u okviru koje je implementirana metoda slobodne deformacije oblika
- upoznavanje sa optimizacijskim algoritmima koji omogućuju rješavanje problema s ograničenjima u okviru dostupnih Python biblioteka (npr. u okviru biblioteke SciPy)
- izradu Python modula koji omogućuje zadavanje i rješavanje optimizacijskog problema minimizacije zahtijevane snage porivnog stroja, uz zadovoljenje hidrostatičkih karakteristika, integracijom programa d3v-gsd i postojećih Python biblioteka s implementiranim optimizacijskim algoritmima i metodom slobodne deformacije oblika
- primjenu izrađenog Python modula za optimizaciju brodske forme u ranim fazama projektiranja na primjeru.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
4. svibnja 2023.	6. srpnja 2023.	17. – 21. srpnja 2023.

Zadatak zadao: Predsjednik Povjerenstva:

Izv. prof. dr. sc. Pero Prebeg Izv. prof. dr. sc. Ivan Ćatipović

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

Oznaka	Jedinica	Opis
D_0	-	FFD stvarna domena
D'_0	-	FFD referenta domena
ψ	-	FFD preslikavanje
$\boldsymbol{P}$	-	FFD kontrolne točke
$\boldsymbol{\mu}$	-	FFD pomak kontrolnih točaka
$\boldsymbol{T}'$	-	FFD preslikavanje pomaka kontrolnih točaka
L	m	Dužina broda
T	m	Gaz broda
B	m	Širina broda
Δ	m^3	Istisnina
LCB	m	Položaj istisnine
P_E	W	Efektivna snaga broda
P_T	W	Snaga porivnog stroja
η_H	-	Stupanj djelovanja trupa
v	m/s	Brzina broda
R_T	N	Ukupni otpor broda
C_T	-	Koeficijent ukupnog otpora
C_F	-	Koeficijent otpora trenja
C_R	-	Koeficijent preostalog otpora
S	m^2	Oplakana površina
ρ_W	kg/m^3	Gustoća vode
R_n	-	Reynoldsov broj
v	m^2/s	Kinematička viskoznost vode
b_mo_x1	-	Relativan pomak pramčanog reda 1
b_mo_x2	-	Relativan pomak pramčanog reda 2
a_mo_x1	-	Relativan pomak krmenog reda 1
a_mo_x2	-	Relativan pomak krmenog reda 2

SAŽETAK

U radu je opisan i primijenjen proces transformacije brodske forme primjenom metode slobodne deformacije oblika. U okviru rada izrađeni su *Python* moduli koji omogućuju komunikaciju između paketa za zapis brodske forme koristeći *NURBS* plohe, transformacije forme, izračun otpora i jednociljnu optimizaciju. Također je izrađen algoritam za interpolaciju *NURBS* krivulja koji je potreban za brzo određivanje točaka na formi broda koju zahtjeva korištena metoda za proračun otpora. Ti moduli uklopljeni su u program otvorenog koda za *D3V-GSD-OCC*. U njemu su dodani meniji koji omogućuju korisniku interakciju sa novim funkcionalnostima i vizualizaciju deformirane forme. Na kraju rad dan je primjer optimizacije jednostavne brodske forme u kojem se primjenjuju funkcionalnosti implementirane kroz ovaj rad.

Ključne riječi: Python, PythonOCC, PyGem, FFD, Openmesh, poligonska mreža, vizualizacija, d3v, NURBS plohe, optimizacija

SUMMARY

This thesis describes and applies the process of transforming a ship hull form using the free form deformation method. As part of the work, new Python modules were made that allow communication between packages for describing a ship hull using NURBS surfaces, hull transformation, resistance calculation and single objective optimisation. The work also included implementation of the algorithm for interpolating NURBS curves which is needed for quick calculation of points on the hull that the used method of resistance calculation needs. These modules are incorporated in the open source D3V-GSD-OCC. New menus were added that allows user interaction with the new functionalities and visualization of a deformed hull. Finally, optimization of a simple hull form is given as an example of usage of newly implemented functionalities .

Key words: Python, PythonOCC, PyGem, FFD, Openmesh, polygon mesh, visualization, d3v, NURBS surfaces, optimization

1. UVOD

Glavna svrha trgovačkih brodova je da prenose zadani teret od točke A do točke B na što ekonomičniji način, ostvarujući pri tome što veći profit. Veći brodovi poput tankera redovito prevoze taj teret preko jednog ili više oceana pa stoga, ako se uspije smanjiti potrebna snaga porivnog stroja za postizanje potrebne brzine, smanjit će se cijena broda uz istovremeno smanjenje potrošnja goriva, a s time i troškovi eksploatacije broda.

Jedan od načina smanjivanja gubitaka je optimizacijom forme broda. Trup je dio broda koji daje brodu sposobnost plovnosti, osigurava strukturalni integritet i štiti unutrašnjost broda od vode. Za vrijeme plovidbe javljaju se hidrodinamičke pojave uslijed gibanja vode oko trupa koje uzrokuju razne gubitke, te se predstavljaju silom otpora. Ta sila otpora se mijenja ovisno o obliku forme i brzini plovidbe. [46]

Optimizacijom oblika brodske forme moguće je postići zadanu brzinu s manjom snagom porivnog motora uz zadržavanje stabiliteta u zadovoljavajućim granicama. S obzirom da je osnivanje broda iterativan postupak koji ovisi o velikoj količini međusobno povezanih varijabli, među kojima su već navedene karakteristike, potrebno je u svakoj iteraciji imati mogućnost dobivanja bolje brodske forme.

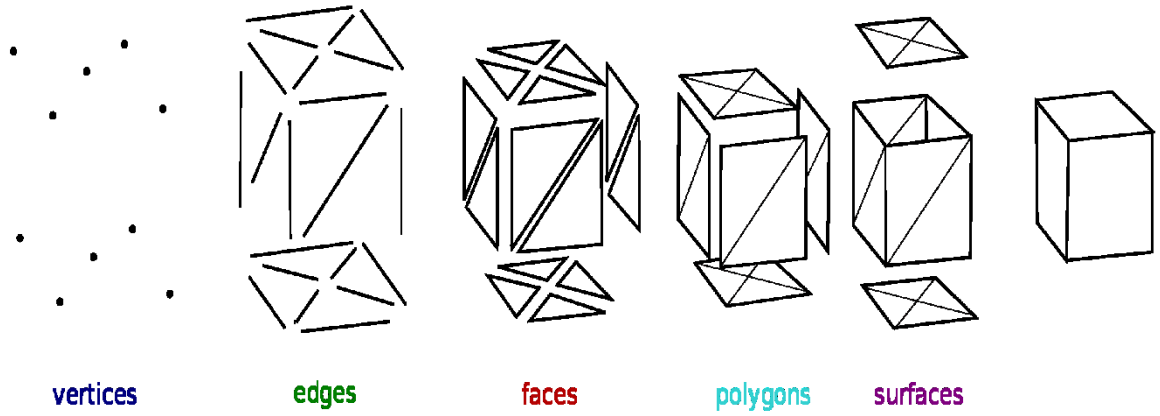
U ovom radu će se opisati teorija i opis implementacije postupka modificiranja brodske forme u ranoj fazi projektiranja koristeći metodu slobodne deformacije oblika i jednociljnu optimizaciju u programskom jeziku *Python*, te primjena implementiranih modula na primjeru optimizacije jednostavne forme.

2. NAČINI ZAPISA PLOHA I TRANSFORMACIJA PLOHE METODOM SLOBODNE DEFORMACIJE

U nastavku su prikazani načini zapisa plohe korišteni u ovom radu te metoda slobodne deformacije koja se u radu koristi za transformaciju brodske forme.

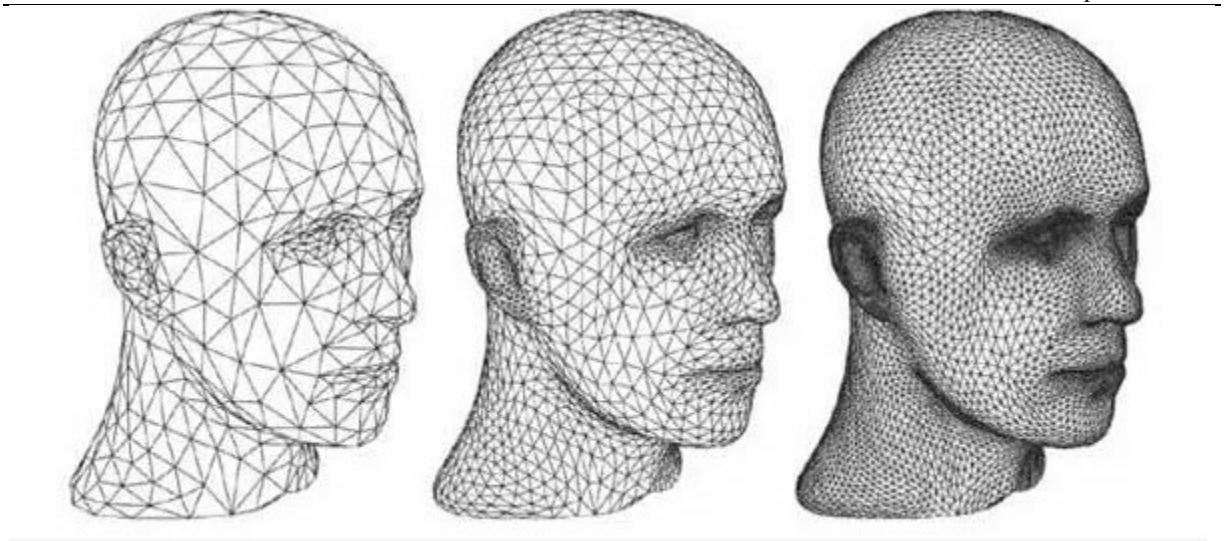
2.1. Poligonske mreže

U 3D Računalnoj grafici poligonska mreža je skup točaka (eng. *vertices*), rubova (eng. *edges*) i lica (eng. *faces*) koji zajedno definiraju Poliedarski objekt. Lica se mogu raditi od trokuta, četverokuta ili jednostavnih n-terokuta. [2]



Slika 1 Elementi poligonske mreže (46])

Sa povećavanjem gustoće mreže postižu se veći detalji, no operacije sa takvim mrežama su sporije zbog veće količine elemenata koje algoritam treba obraditi.

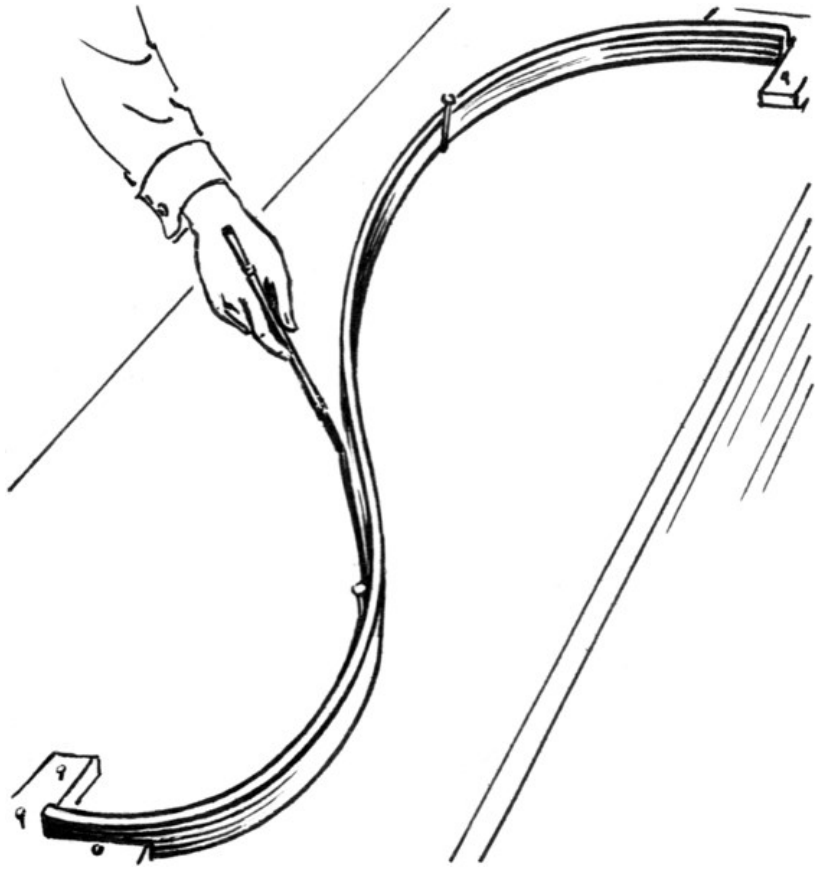


Slika 2 Promjena gustoće mreže ([3])

2.2. NURBS krivulje i plohe

Non-uniform rational basis spline (*NURBS*) je matematički model opisivanja krivulja i ploha u Računalnoj grafici koristeći *Basis spline*-ove (*B-spline*). Često se koriste u svrhe *CAD*-a (eng. *Computer Aided Design*), *CAM*-a (eng. *Computer Aided Manufacturing*) i *CAE*-a (eng. *Computer Aided Engineering*), te su dio mnogih industrijskih standarda poput *ACIS*-a, *STEP*-a i *IGES*-a. Za razliku od poligonskih mreža, *NURBS* je opisan sa matematičkim formulama, što omogućuje veliku točnost pri definiranju zakrivljenja. [4]

Prije računala, komplicirane krivulje su se crtale na papiru pomoću elastičnih, drvenih letvica koje su se pričvršćivale na određenim točkama, te bi se pod utjecajem sile te letvice deformirale i zauzimale najglađi mogući oblik.

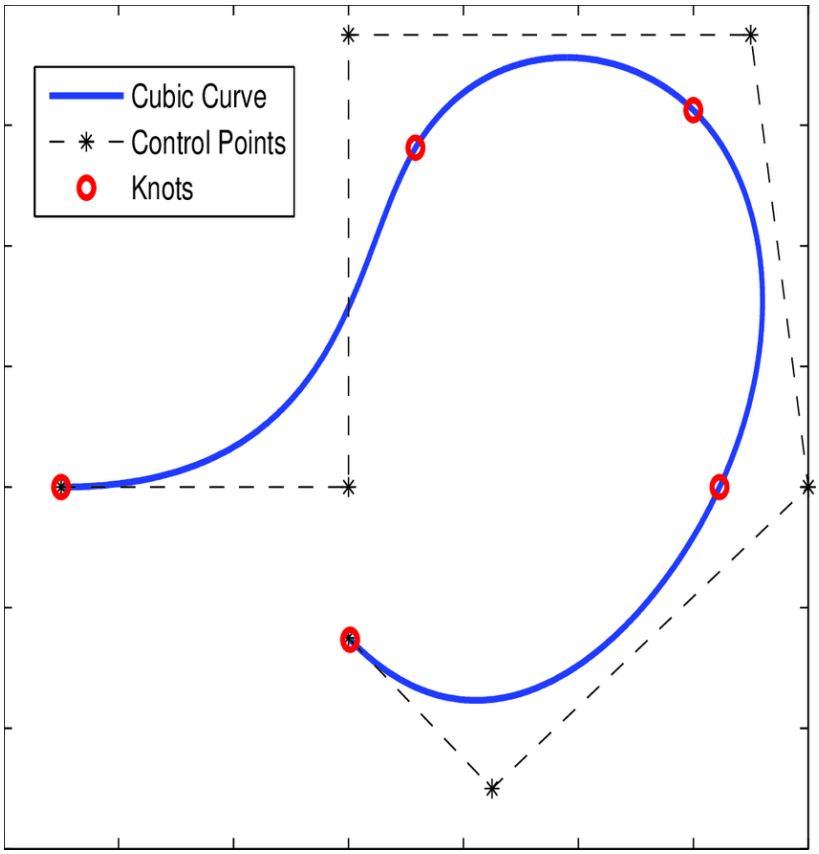


Slika 3 Drvena letvica za crtanje krivulja ([4])

Kasnije se razvio matematički model za opisivanje takvih krivulja, te su se, sa razvitkom računala, počele koristiti u *CAD* softveru proizvođača auta. Kasnije su postale standardni dio računalnih grafičkih paketa.

NURBS krivulje su definirane sa svojim redom (eng. *order*), kontrolnim točkama (eng. *control points*) i vektorom čvorova (eng. *knot vector*).

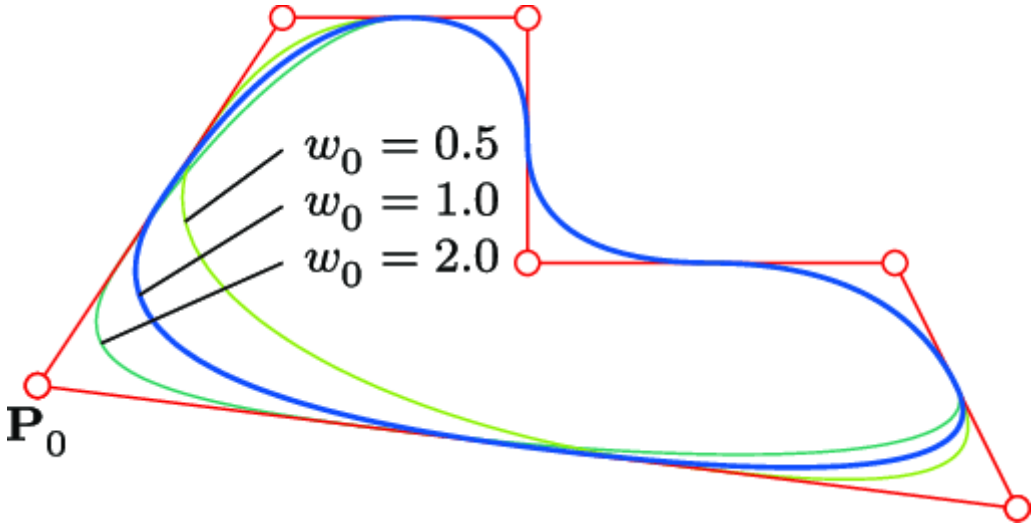
Red *NURBS* krivulje definira broj kontrolnih točaka koji utječu na položaj točke na krivulji. Krivulja je matematički predstavljena sa polinom jednog reda manjeg od reda *NURBS* krivulje. U praksi se najčešće koriste krivulje četvrtog reda, dok se krivulje petog i šestog reda povremeno koriste za dobivanje derivacija višeg reda, no za njihovo računanje je potrebno mnogo vremena i može doći do unutarnjih numeričkih greški.



Slika 4 ***NURBS*** **Krivulja četvrtog reda ([7])**

Kontrolne točke određuju oblik krivulje. Obično se svaka točka na krivulji računa kao suma kontrolnih točaka sa uračunatom težinom točke koja se mijenja ovisno o intervalu krivulje. Činjenica da jedna kontrolna točka ima utjecaj na samo jedan dio krivulje se smatra vrlo poželjnim svojstvom jer dopušta lokalno modeliranje krivulje bez mijenjanja ostatka. Dodavanjem kontrolnih točaka se krivulja može bolje aproksimirati.

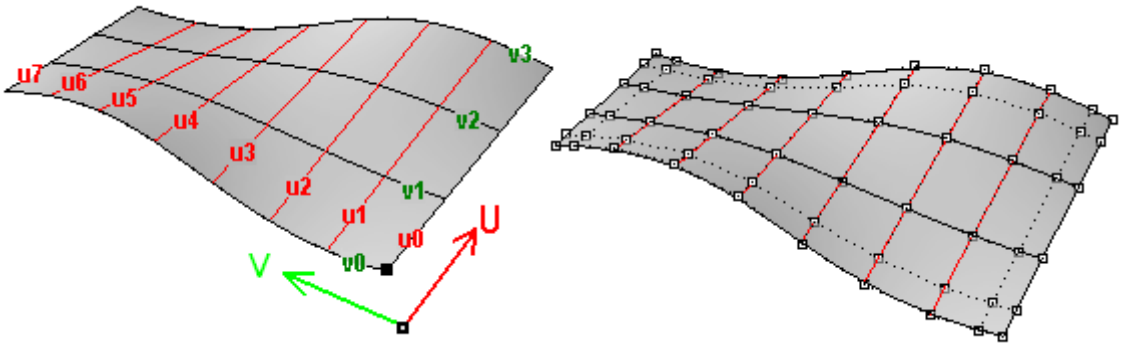
Svakoj kontrolnoj točki je pridružena skalarna vrijednost koja predstavlja težinu točke. Mijenjanjem te vrijednosti se omogućuje i mijenjanje oblika krivulje bez dodavanja novih kontrolnih točaka.



Slika 5 Utjecaj težine kontrolne točke na oblik *NURBS* krivulje (Error! Reference source not found.**)**

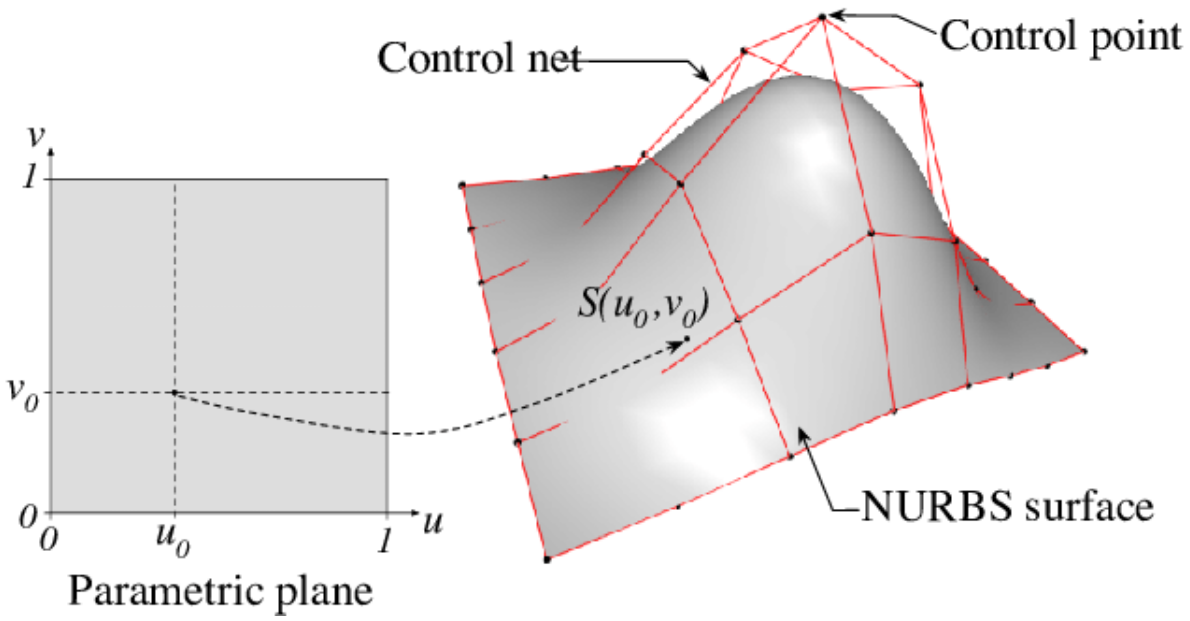
Vektor čvorova je niz parametara koji određuju gdje i kako kontrolne točke utječu na *NURBS* krivulju. Broj čvorova je uvijek jednak zbroju broja kontrolnih točaka i reda *NURBS* krivulje. Čvorovi dijele *NURBS* na intervale individualnih krivulja, na kojem se aktiviraju nove kontrolne točke, a stare se deaktiviraju.

NURBS plohe su dio 3D prostora koji je parametriziran sa dva parametra, obično nazvani u i v, koji zajedno opisuju 2D prostor.



Slika 6 Parametrizacija *NURBS* ploha 1 ([6])

Ti u i v parametri prate individualne *NURBS* krivulje, te se pomoću njih mogu dobiti bilo koje točke na plohi.



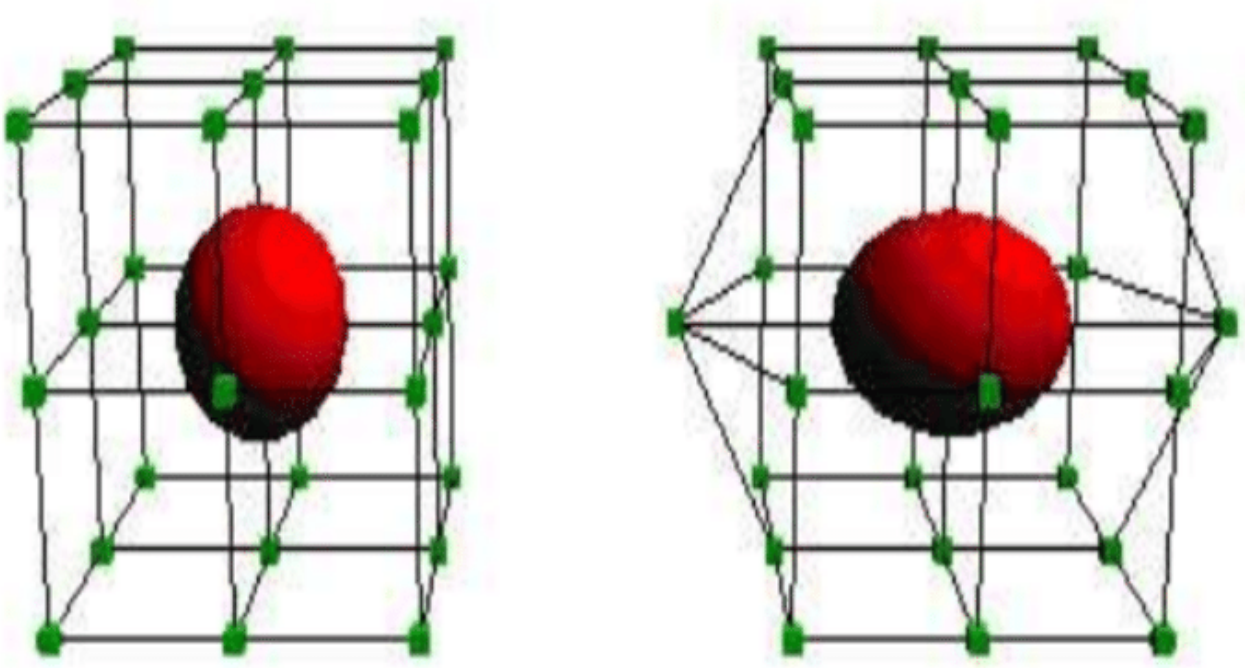
Slika 7 Parametrizacija *NURBS* ploha 2 ([5])

Ova svojstva omogućuju računalu efikasno računanje, a ljudima lagano korištenje *NURBS* ploha.

2.3. Metoda slobodne deformacije oblika

Metoda slobodne deformacije oblika (eng. *free-form deformation - FFD*) primjenjuje se u računalnoj grafici za deformiranje objekata. Originalno opisan u Sederberg and Parry (1986) [21], razvijao se uglavnom za svrhe računalne grafike, te se tek nedavno počeo koristiti u problemima transformacije brodske forme. [11]

Tehnika prvo postavlja skup kontrolnih točaka koji okružuju dio geometrije koju želimo deformirati, te se potom ta geometrija deformira na kontinuiran i gladak način pomicanjem kontrolnih točaka.

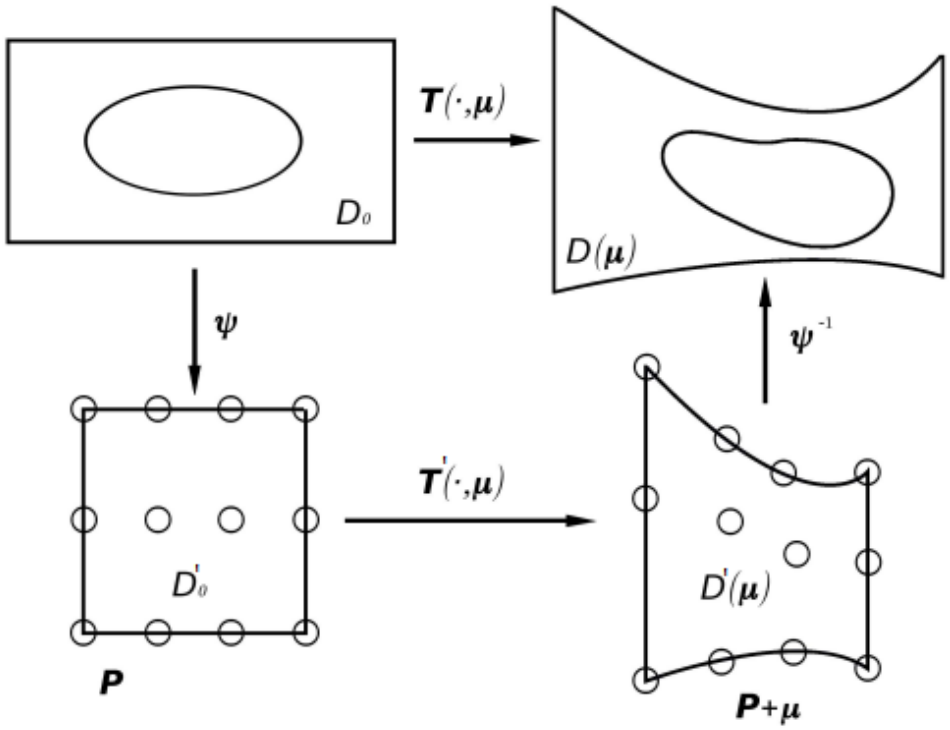


Slika 8 FFD pomicanje kontrolnih točaka ([12])

Deformacija se odvija u tri koraka:

1. Stvarna domena D_0 preslikava se na referentu domenu D'_0 primjenom preslikavanja ψ
2. Stvara se rešetka nepomaknutih kontrolnih točaka $\boldsymbol{P}$, te se neke od tih kontrolnih točaka pomiču za $\boldsymbol{\mu}$ preslikavanjem $\boldsymbol{T}'$.
3. Dobivenu domenu D' vraćamo natrag u stvarnu domenu D sa invertiranim preslikavanjem ψ^{-1}

$$\boldsymbol{T}(\cdot, \mu) = (\psi^{-1} \circ \boldsymbol{T}' \circ \psi)(\cdot, \mu) \quad (1)$$



Slika 9 Koraci FFD metode ([11])

FFD je neovisan od geometrije oblika koja će se deformirati, pa je stoga vrlo svestran i prikladan za parametrizaciju vrlo kompliciranih geometrija poput poligonskih mreža i *NURBS* ploha. Moguće je dobiti velike i male deformacije, što može postati mana ukoliko je kvaliteta početne geometrije loša. *FFD* metoda će se koristiti za deformaciju brodske forme u okviru primjera primjene optimizacije u poglavlju 5.

3. UVOD U PYTHON I NAJVAŽNIJE KORIŠTENE BIBLIOTEKE

Python je interpretirani, fleksibilni, objektno orijentirani programski jezik otvorenog koda jezik koji se sve više koristi u različitim znanstvenim poljima. Karakteriziran je dobrom čitljivosti i sposobnosti proširivanja funkcionalnosti pomoću biblioteka[13]. U nastavku je dan kratki opis najvažnijih korištenih biblioteka izuzev *d3v*-a i *PyGem*-a čiji opis je dan u posebnom poglavlju.

3.1. Biblioteka NumPy

NumPy je temeljni paket otvorenog koda za numeričko računarstvo u *Python* jeziku. Modul ima mogućnost rada sa višedimenzijskim poljima i objektima koji proizlaze iz njih (npr. matrice).[15]

NumPy omogućuje ove operacije sa nizovima:

1. Matematičke
2. Logičke
3. Manipulacije oblika
4. Redanje
5. I/O
6. Fourierove transformacije
7. Linearna algebra
8. Statistika
9. Simulacije Slučajnosti

3.2. Biblioteka SciPy

SciPy je temeljni paket otvorenog koda za znanstveno računarstvo u *Python* jeziku. Koristi višedimenzijska polja in *NumPy* modula kao osnovnu strukturu podataka. Dok *NumPy* sadrži neke osnovne funkcije za linearnu algebru, Fourierove transformacije i generaciju nasumičnih brojeva, *SciPy* sadrži proširene sposobnosti tih funkcija namijenjene za generalnu upotrebu.[14]

SciPy sadrži module za:

1. Optimizaciju
2. Linearnu algebru
3. Integraciju
4. Interpolaciju
5. Specijalne funkcije
6. *FFT*(eng. *Fast Fourier Transformation*)
7. Procesiranje slika
8. Diferencijalne jednadžbe

3.3. Biblioteka Openmesh

Za operacije sa poligonskim mrežama koristiti će se *Pyhton* biblioteka *Openmesh.* To je biblioteka sa jednostavnom i učinkovitom strukturom podataka za korištenje poligonskih mreža i korisnik ima mogućnost izrade vlastite vrste mreže potrebne za specifične aplikacije. To se postiže dinamičnim svojstvima mreže koje korisnik može mijenjati za vrijeme izvršavanja programa po volji. [16]

Glavne prednosti koda *Openmesha*:

1. Ne postoje ograničenja za trokutaste mreže, korištenje općih poligonskih mreža.
2. Eksplicitno predstavljanje točaka, rubova, polurubova i lica.
3. Lagano pristupanje susjednim elementima točke.
4. Mogućnost izrade ne raznolikih točaka (dva lica se susreću u samo jednoj točki)

Također sadrži poveznice za *Python* koje su korištene u ovom radu. *Openmesh* koristi *Python* modul *NumPy* za zapis poligonskih mreža.

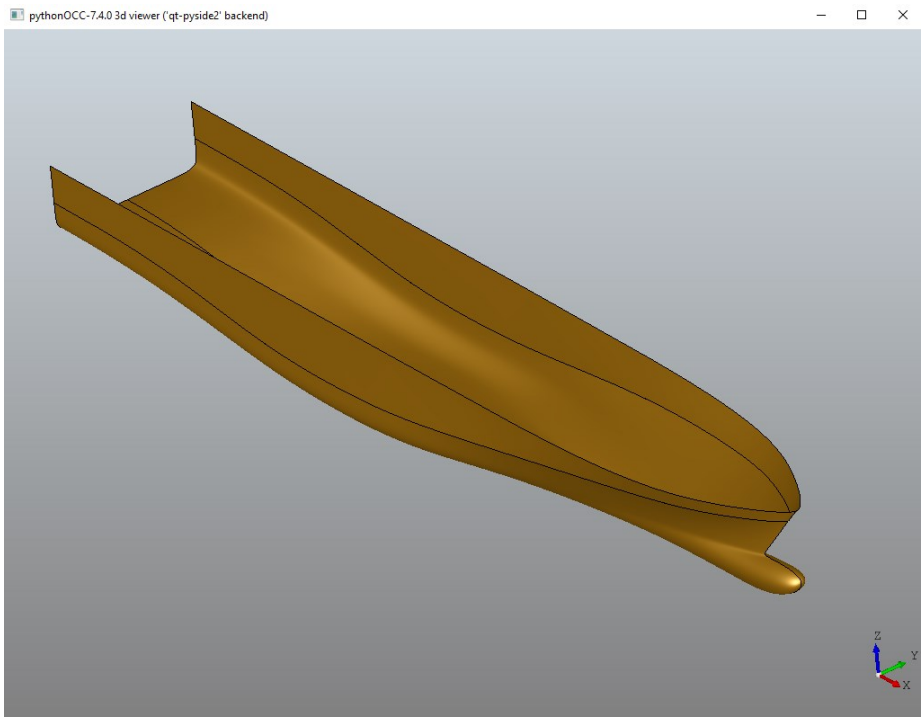
3.4. Biblioteka OpenCascade

OpenCascade(*OCC*) je objektno orijentirana biblioteka otvorenog koda pisana u *C++* jeziku. Namijenjena je za upotrebu u *CAD*, *CAM* i *CAE* svrhe. [9]

OpenCascade je podijeljen na sedam glavnih modula:

1. Temeljne klase
2. Podaci za modeliranje
3. Algoritmi za modeliranje
4. Vizualizacija
5. Izmjena podataka
6. Framework za aplikacije
7. DRAW Test Harness

Opencascade se može koristiti u *Python* jeziku pomoću biblioteke *PythonOCC.* Te poveznice omogućuju korištenje klasa i algoritama iz *OpenCascade*-a i koristiti će se u ovom radu za zapis *NURBS* ploha i komunikaciju sa njihovim svojstvima. Također u sebi sadrži tesselarizator koji pretvara mrežu točaka iz *NURBS* ploha u poligonske mreže.



Slika 10 PythonOCC vizualizator

3.5. Qt for Python(Pyside6)

Qt je više-platformna aplikacija za izradu grafičkih korisničkih sučelja i grafičkih elemenata. *Qt* sam po sebi nije programski jezik, nego okruženje pisano u *C++* jeziku. U sebi sadrži alate *Qt Creator* i *Qt Designer* za brzo i jasno stvaranje grafičkih sučelja. [17]

Trenutno podržavane platforme su:

1. Linux
2. OS X
3. Windows
4. QNX
5. Android
6. iOS
7. BlackBerry
8. Sailfish OS
9. i drugi

Također, *Qt* u sebi ima poveznice za *Python*. Sposobnosti *Qt*-a da radi na bilo kojoj podržavanoj platformi i na bilo kojem podržavanom jeziku su pretvorile *Qt* u industrijski standard za izradu grafičkih sučelja. Grafičko sučelje i vizualizacija poligonskih mreža u *linaetal-FSB/D3V-GSD-OCC*-u su izrađena pomoću *Qt*-a.

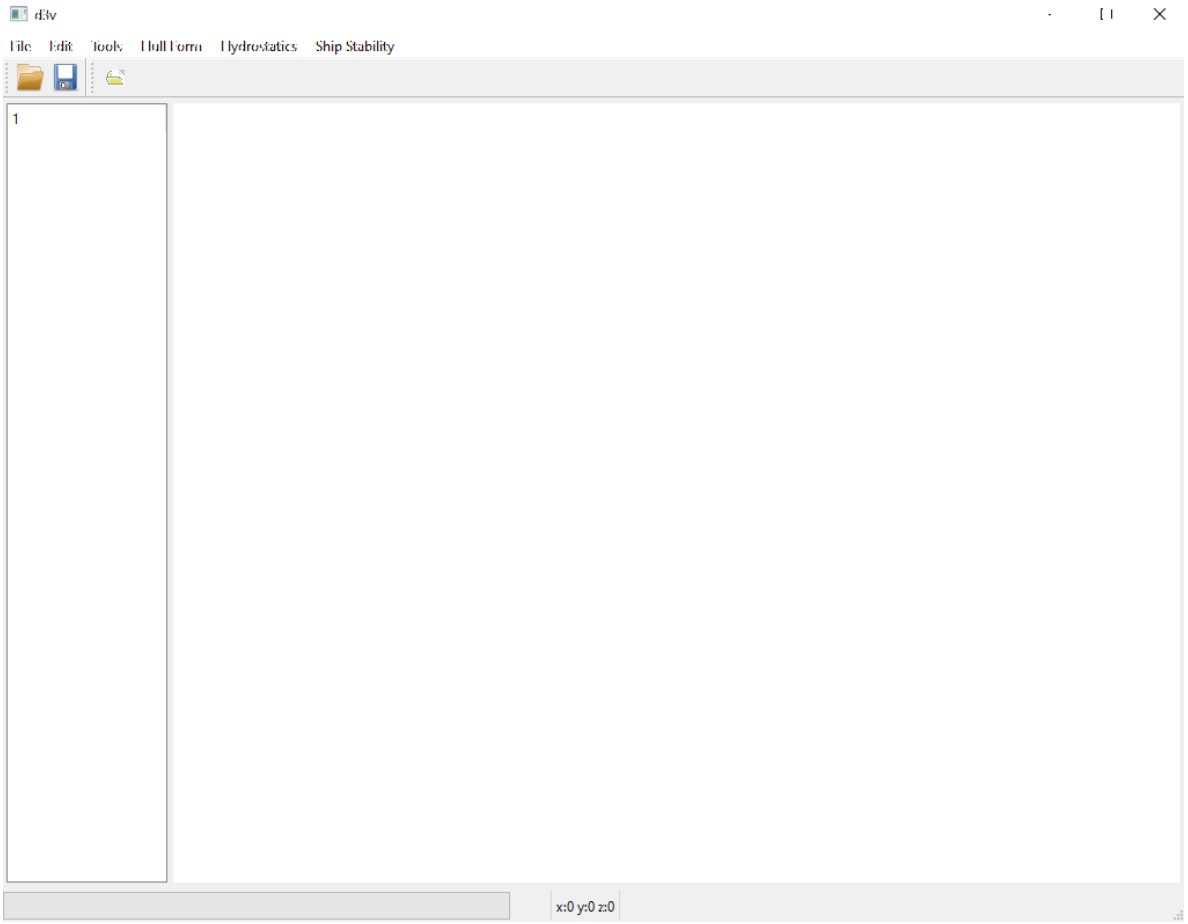
4. PROŠIRENJE FUNKCIONALNOSTI PROGRAMA D3V-GSD ZA TRANSFORMACIJU FORME METODOM SLOBODNE DEFORMACIJE OBLIKA

U ovom radu se koristi *D3V-GSD* odnosno *D3V-GSD-OCC* što je ekstenzija osnovnog programa sa funkcionalnostima koje omogućuje biblioteka *OpenCascade* odnosno mogućnost rada s brodskom formom zapisanom pomoću *NURBS* ploha.

D3v je modularna *Python* aplikacija otvorenog programskog koda, prvenstveno namijenjena za 3D vizualizaciju inženjerskih modela u fazi projektiranja. Omogućuje jednostavno definiranje grafičkog sučelja za pojedine namjene. Nastala je kao rezultat dugogodišnje suradnje USCS.d.o.o te obrta Linaetal s Fakulteta strojarstva i brodogradnja. Struktura programa temeljena je na dvanaestogodišnjem iskustvu razvoja programa ShipExplorer. Program je u ranoj fazi razvoja no već je moguća vizualizacija s ograničenim brojem funkci onalnosti. Implementacije je bazirana na bibliotekama QtForPython (PySide6) i OpenMesh.

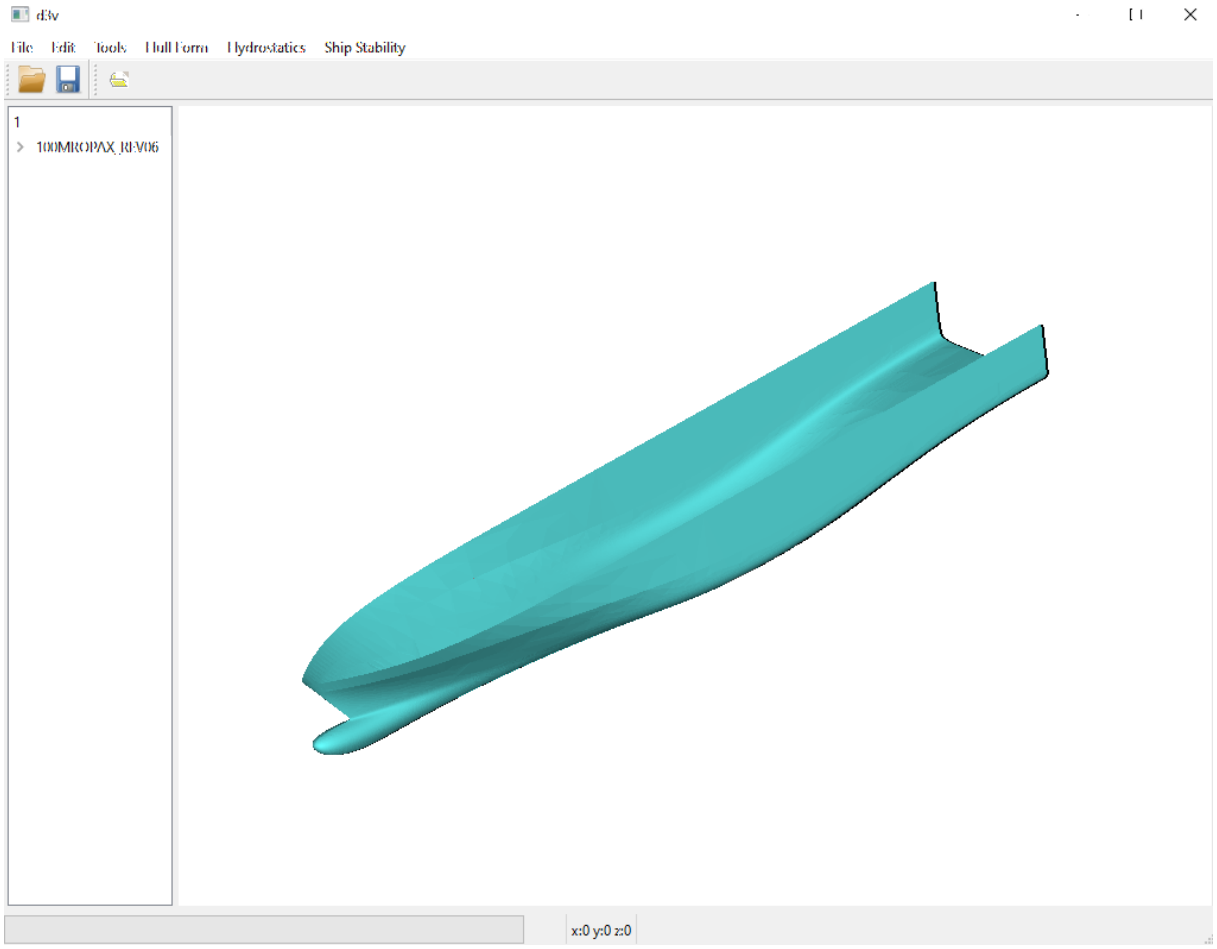
Daljnjim razvijanjem su dodane mogućnosti izračuna hidrostatičkih karakteristika u obliku programa *D3V-GSD* (eng. *D3sign Visualizer for General Ship Design*). Program omogućuje jednostavno analitičko zadavanje forme broda, primjenjivo se u ranim fazama projektiranja broda jer omogućuje jednostavno modificiranje forme broda putem promjene manjeg broja parametara. Također je omogućeno učitavanje forme pomoću *obj* ili *stl* datoteka. Ekstenzija *D3V-GSD-OCC* dodaje sposobnosti korištenja *OpenCascade* (*OCC*) biblioteke za modularno prikazivanje i manipulaciju 3D objekata u obliku *Python* biblioteke *PythonOCC*. [18] Implementacija je bazirana na bibliotekama *PySide6*, *OpenMesh* i *PythonOCC*. Programi su dostupni sa poveznica:

- https://github.com/pprebeg/d3v-gsd-occ
- https://github.com/pprebeg/d3v-gsd
- https://github.com/linaetal-fsb/d3v



Slika 11 ***Linaetal-FSB/D3V-GSD-OCC*** **Prozor**

Kada se *linaetal-FSB/D3V-GSD-OCC* pokrene pomoću izbornika „File“ i „Import Geometry“ može se učitati datoteka podržanog formata.



Slika 12 ***Linaetal-FSB/D3V-GSD-OCC*** **Prozor sa učitanom formom**

Funkcionalnosti *Linaetal-FSB/D3V-GSD-OCC* prije implementacija novih funkcionalnosti koje su napravljeni u ovom radu su:

- Vizualizacija mreža u raznim bojama
- Rotacija, translacija i povećavanje/smanjivanje pogleda
- Učitavanje geometrije forme iz datoteka u Geometry Tree
- Interaktivna selekcija individualnih mreža mišem
- Izračunavanje hidrostatičkih značajki forme
- Izračunavanje krivulja stabiliteta forme

Neke od navedenih funkcionalnosti su dostupne preko padajućih menija. Svi učitani objekti su prikazani u lijevom prozoru, te se mogu selektirati klikom na ime objekta.

4.1. Korišteni prethodno razvijeni moduli programa D3V-GSD-OCC

D3V-GSD-OCC koristi više modula za ostvarivanje svojih funkcija, no ovdje će se opisati samo oni koji su važni za implementaciju novih funkcionalnosti potrebne za optimizaciju brodskih formi.

Modul grafičkog sučelja *hullmod_command.py* se koristi za stvaranje padajućih menija i gumbova pomoću kojih se omogućuje grafička interakcija sa ostalim funkcionalnostima *D3V-GSD-OCC*-a. Stvaranje novih gumbova se postiže metodom *addAction*("Ime gumba"), te se povezuje sa izvršavanjem druge funkcije pomoću *triggered.connect*(funkcija).

U modulu *occhullform.py* se nalazi klasa *OCCHullform*, koja služi kao glavna klasa u kojoj se zapisuje i koristi učitana forma broda. Klasa *OCCHullform* nasljeđuje klasu *Hullform* u kojoj se definiraju operacije vezane za trokutaste mreže koje opisuju formu. *OCCHullform* dodaje mogućnost korištenja, manipuliranja i učitavanja formi opisanih pomoću *NURBS* ploha u *Opencascade*-u. Od veće važnosti su sposobnosti učitavanja datoteka koje sadrže formu broda u obliku *NURBS* ploha i pretvoriti ju u plohu od poligonske mreže za vizualizaciju. [9]

Modul *shipstability.py* se koristi za izračun hidrostatičkih značajki i krivulja stabiliteta. Za ovaj rad je od posebne važnosti metoda *calculate_displacement_and_displacementCG* koja izračunava istisninu i centar istisnine, vrijednosti koje će se koristiti kao granice optimizacije.

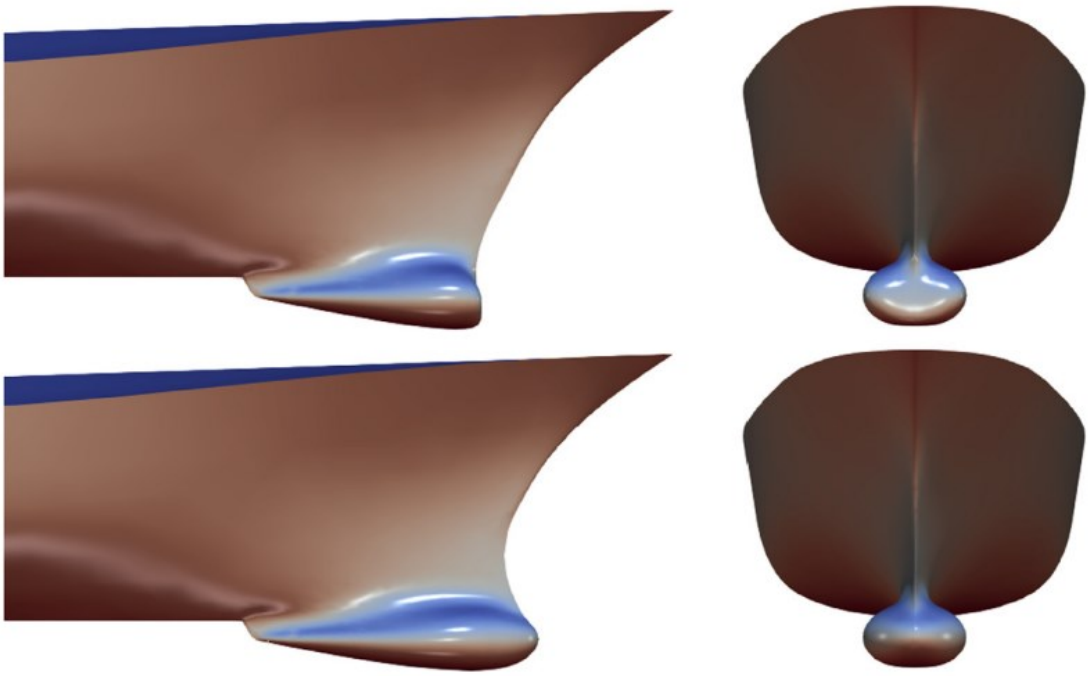
4.2. Biblioteka PyGem

PyGem je biblioteka za programski jezik *Python* koja koristi *FFD*, *RBF* (*Radial Basis Function*) i *IDW* (*Inverse Distance Weighting*) za deformaciju geometrije objekata. [19] Pogodna je industrijske probleme jer je sposobna koristiti:

1. *CAD* datoteke (u .iges, .step i .stl formatu)
2. Poligonske Mreže (u .unv, Elmer i OpenFOAM formatu)
3. Izlazne datoteke (u .vtk formatu)
4. LS-Dyna Keyword datoteke (u .k formatu)

Između ostalog, sposobna je raditi na *NumPy* poljima, povećavajući funkcionalnost nekih softvera.

Korištenje *PyGem*-a je jednostavno, prvi korak je učitati geometriju koju želimo deformirati (točke za poligonske mreže i kontrolne točke za *NURBS* objekte), te se nakon toga može odrediti željena metoda deformacije, zajedno sa pripadajućim parametrima. Postavljanje parametara je moguće kroz kod ili tekstualnu datoteku. Deformirani objekt se potom može zapisati u originalnom formatu datoteke.



Slika 13 Primjer deformacije pramčanog bulba pomoću *PyGem* paketa [19]

PyGem biblioteka će se koristit u ovom radu za deformaciju *NURBS* ploha brodske forme pomoću *FFD* metode.

4.3. Biblioteka moobench

Biblioteka *moobench* je softver otvorenog koda koji je namijenjen rješavanju jednociljnih i višeciljnih optimizacijskih problema s ograničenjima. Korisnik može postaviti optimizacijski problem te odabrati željeni optimizacijski algoritam za njegovo rješavanje. Biblioteka *moobench* koristi biblioteke *SciPy.Optimize*, *Pymoo* i *jMetalPy*, te implementira jednostavne naredbe koje omogućuje korisniku jednostavno korištenje funkcionalnosti tih biblioteka. Naime, najzahtjevniji dio postavljanja optimizacije je ukomponirati svoj složeni problem u optimizaciju. Uz to, često je puta mnogo toga potrebno iznova ponovno napisati samo za drugu biblioteku jer su njihove implementacije gotovo uvijek različite. Biblioteka *moobench* nastoji smanjiti razlike prema korisniku u definiranju problema za različite optimizacijske biblioteke. Tako, korisnik može jednom definirati problem unutar zasebnog modula korištenjem funkcionalnosti *moobench* biblioteke, i pritom se fokusirati jednostavno na definiranje optimizacije i dodjeljivanja čisto optimizacijskih operatora, algoritama, kriterija zaustavljanja, itd.

Drugim riječima, *moobench* biblioteka ima za cilj omogućiti korisniku jednostavne naredbe u korisničkom programu neovisne ili vrlo malo ovisne o načinu implementacije modela problema ili o načinu implementacije pojedinih optimizacijskih biblioteka. Kreiranjem takvog sučelja korištenje optimizacije postaje znatno jednostavnije i brže – što znači da bi se razni korisnici u potrebi češće odlučivali na nju. [20]

4.4. Modul michell

Modul *michell.py* se koristi za izračun otpora valova koristeći osnovnu *Michell's thin-ship theory*, a razvijen je u okviru rada [22]. Modul je prilagođen za preuzimanje podataka o formi broda s NURBS zapisa plohe, kao što je opisano u sljedećem potpoglavlju.

4.5. Implementacija novih funkcionalnosti u D3V-GSD-OCC

U okviru ovog rada su dodane nove funkcionalnosti potrebne za deformaciju brodskih formi zapisanih NURBS ploha pomoću *FFD* metode i komunikaciju sa programima za računanje otpora brodskih formi i generalnu višeciljnu optimizaciju.

Modul *pygemhullform.py* je, analogno *OCCHullform.py*, modul zadužen za omogućavanje korištenja biblioteke *PyGem* na učitanoj formi broda. U sebi sadrži dvije klase, *PyGemHullform* i *ffd_maker.*

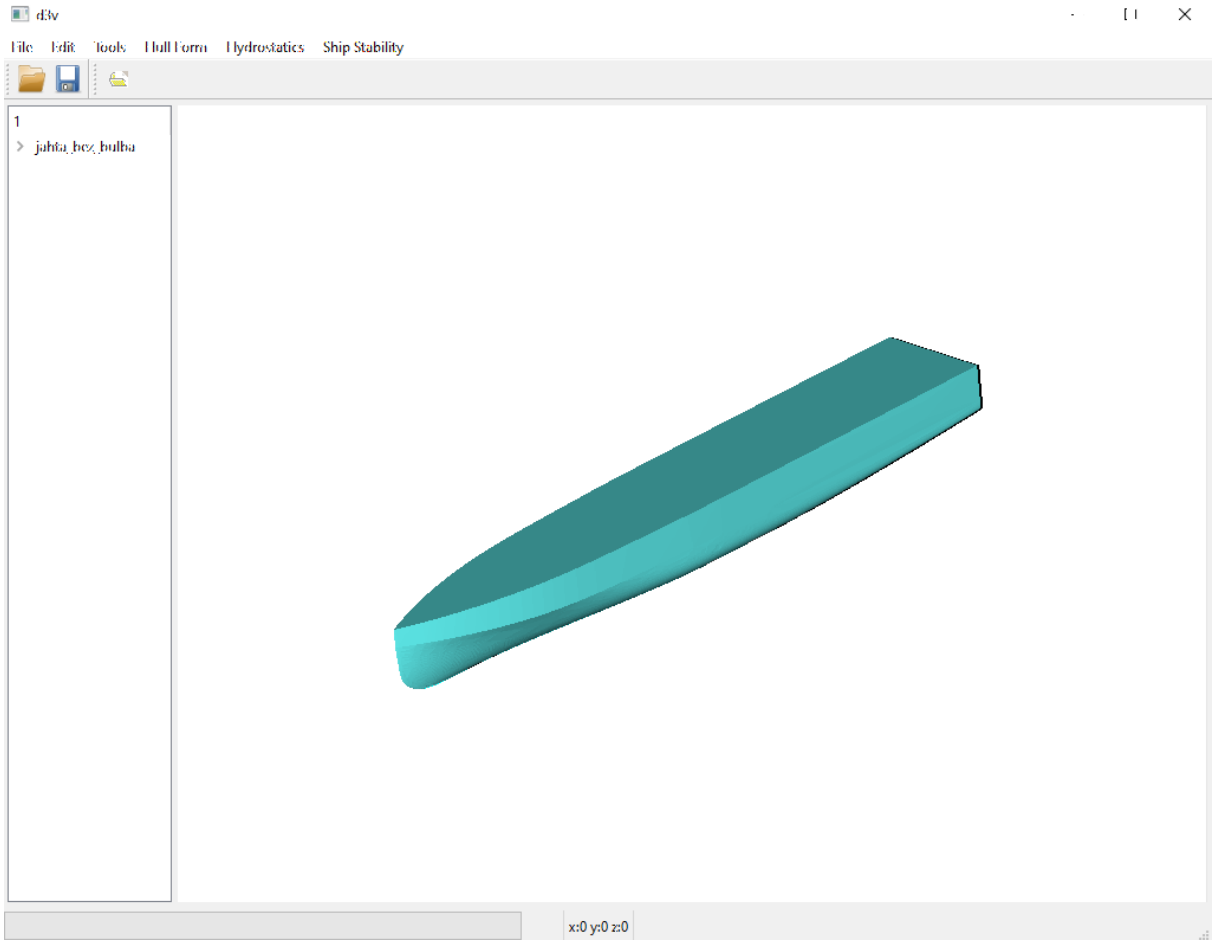
Klasa *ffd_maker* u sebi sadrži metode potrebne za generaciju i manipulaciju *FFD* volumena potrebnih za deformaciju brodske forme. Jedna od važnijih metoda unutar *ffd_maker* klase je *make_ffd_volume* koja stvara *FFD* volumen sa zadanim koordinatama i dimenzijama. Sposobna je stvoriti volumene sa proizvoljnim brojem kontrolnih točaka za finiju kontrolu, stvoreni volumen zarotirati oko glavnih osi za zadani kut i, za vrijeme deformacije, povećati gustoću čvorova u zahvaćenim NURBS objektima u svrhu povećavanja detalja deformiranog objekta. Također sadrži metodu *move_ffd_pole* koja je sposobna pomaknuti individualne kontrolne točke *FFD* volumena kako bi se objekt unutar volumena mogao kasnije deformirati. Kao argument je potrebno zadati indeks kontrolne točke i željeni vektor deformacije u smjeru glavnih osi. Važno je napomenuti da vektor deformacije nije apsolutan, nego pomiče volumen relativno o početnoj veličini *FFD* volumena, tako da na primjer, pomak od 1.5 u smjeru x će pomaknuti čvor za 1.5 dužine *FFD* volumena u x smjeru.

Za optimizaciju, koja će se opisati kasnije, važne su metode *make_form_ffd_cage* i *move_form_ffd_row.* Metoda *make_form_ffd_cage* stvara *FFD* volumen sa 7x2x2 kontrolnih točaka oko cijele forme broda i koristi se za optimizaciju pramca i krme. Tu je da korisnik ne mora unositi ručno točne podatke o dimenzijama i položaju broda. Metoda *move_form_ffd_row* je napravljena za pomicanje redova točaka napravljenih sa metodom *make_form_ffd_cage* kako bi se u optimizaciji forma mogla deformirati.

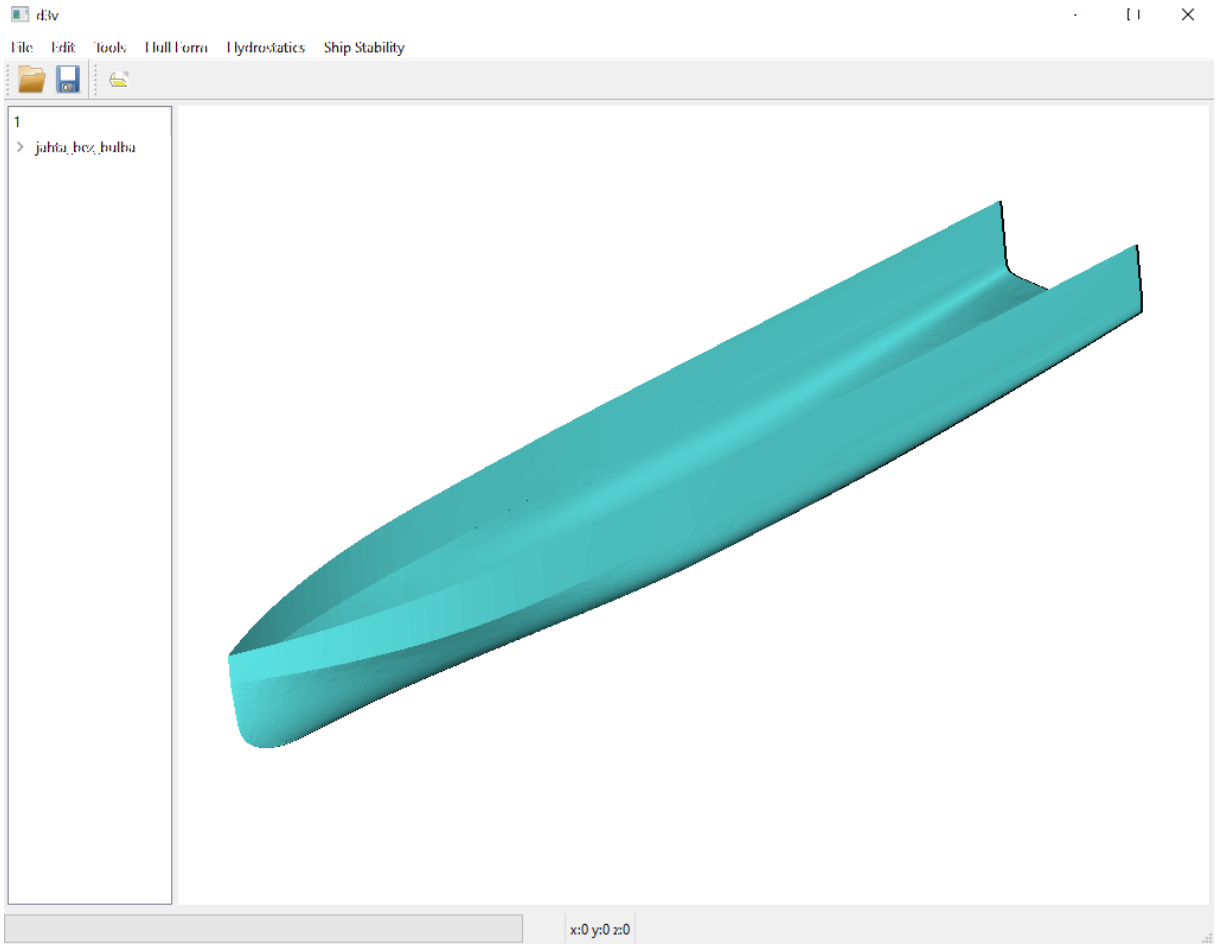
Od manje važnosti je metoda *make_ffd_box_mesh* koja stvara kocke od poligonskih mreža na koordinatama kontrolnih točaka *FFD* volumena u prozoru *linaetal-FSB/D3V-GSD-OCC*-a i služi za vizualizaciju stvorenih točaka.

PyGemHullform je klasa koja nasljeđuje klasu *ffd_maker* i zadužena je za obradu forme kako bi se mogla koristiti za deformaciju pomoću *FFD* volumena i izračun otpora. Također sadrži metodu *ffd_deform_surfaces*, koja koristi stvoreni *FFD* volumen pomoću metoda u *ffd_maker* klasi kako bi deformirala formu unutar tog volumena.

Za svrhe obrade forme sadrži metodu *remove_form_deck_and_aft* koja se koristi za micanje *NURBS* ploha glavne palube i krmenog zrcala, te se pomoću argumenata može izabrati micanje samo glavne palube ili krmenog zrcala, ukoliko su te plohe potrebne. Također sadrži metodu *position_form* koja pomiče NURBS plohe i poligonsku mrežu u željenom smjeru, ukoliko je to potrebno za neke druge svrhe.



Slika 14 Jahta sa glavnom palubom i krmenim zrcalom

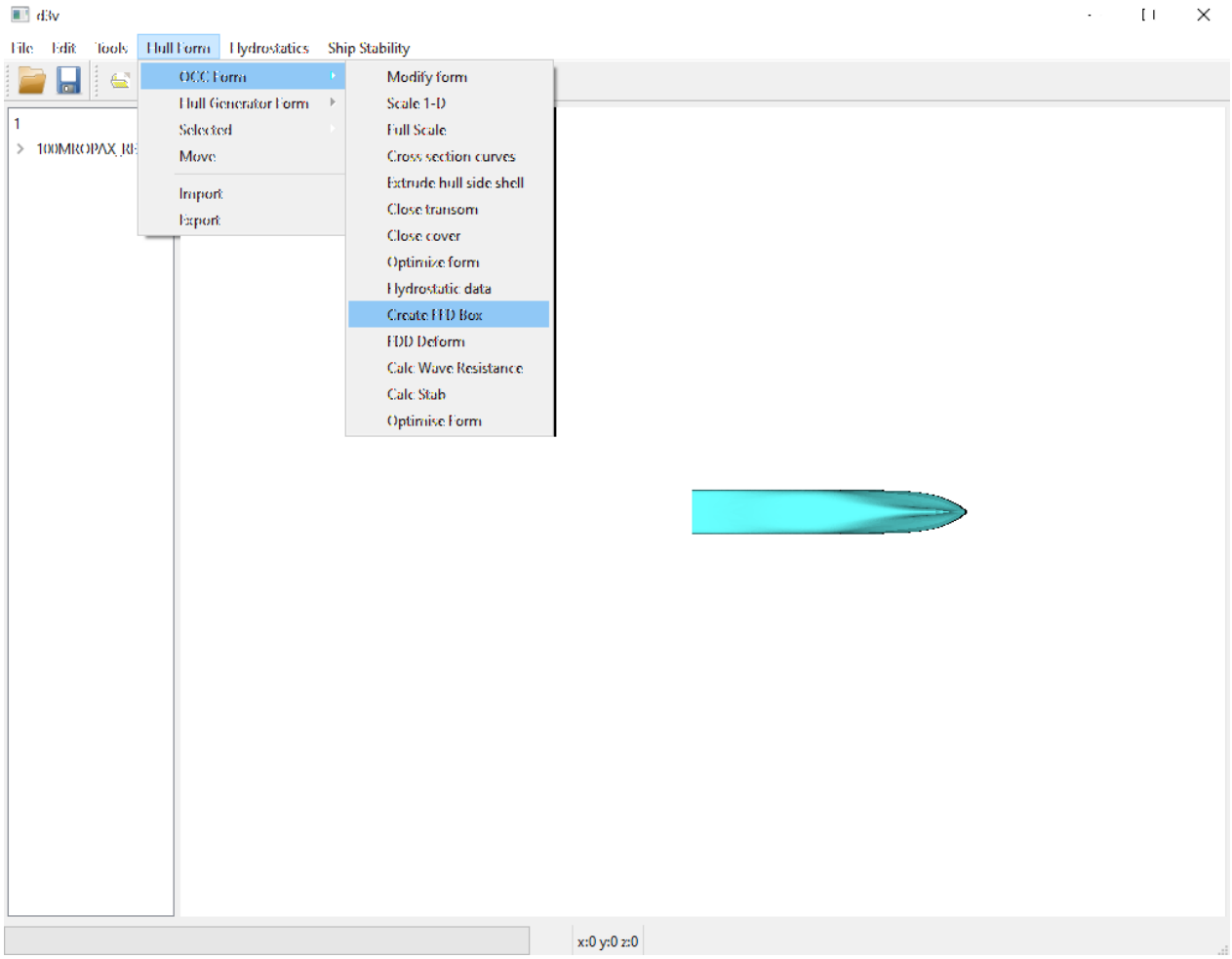


Slika 15 Jahta bez glavne palube i krmenog zrcala

Metoda *calc_stab* se koristi za povezivanje sa modulom *shipstability.py* za izračun istisnine i njenog položaja za trenutno učitanu formu. Ti podaci se kasnije koriste kao ograničena za optimizaciju.

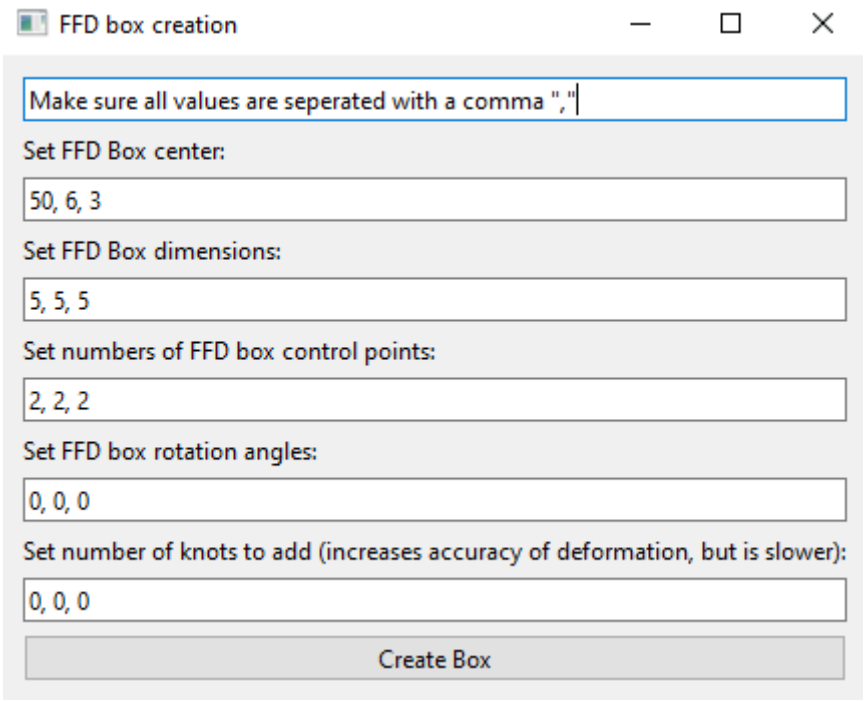
Modul *pygemmenus.py* koristi *PySide6* biblioteku za stvaranje menija potrebnih za interaktivno korištenje *FFD* volumena. Trenutno su implementirana dva menija: *CreateFFDBox_menu* i *DeformFFDBox_menu.*

Meni *CreateFFDBox_menu* je povezan sa klasom *ffd_maker* i koristi se za interaktivno stvaranje novog *FFD* volumena. Pristupa mu se sa klikom na padajući izbornik „Hull Form“, „OCC Form“ i „Create *FFD* Box“.



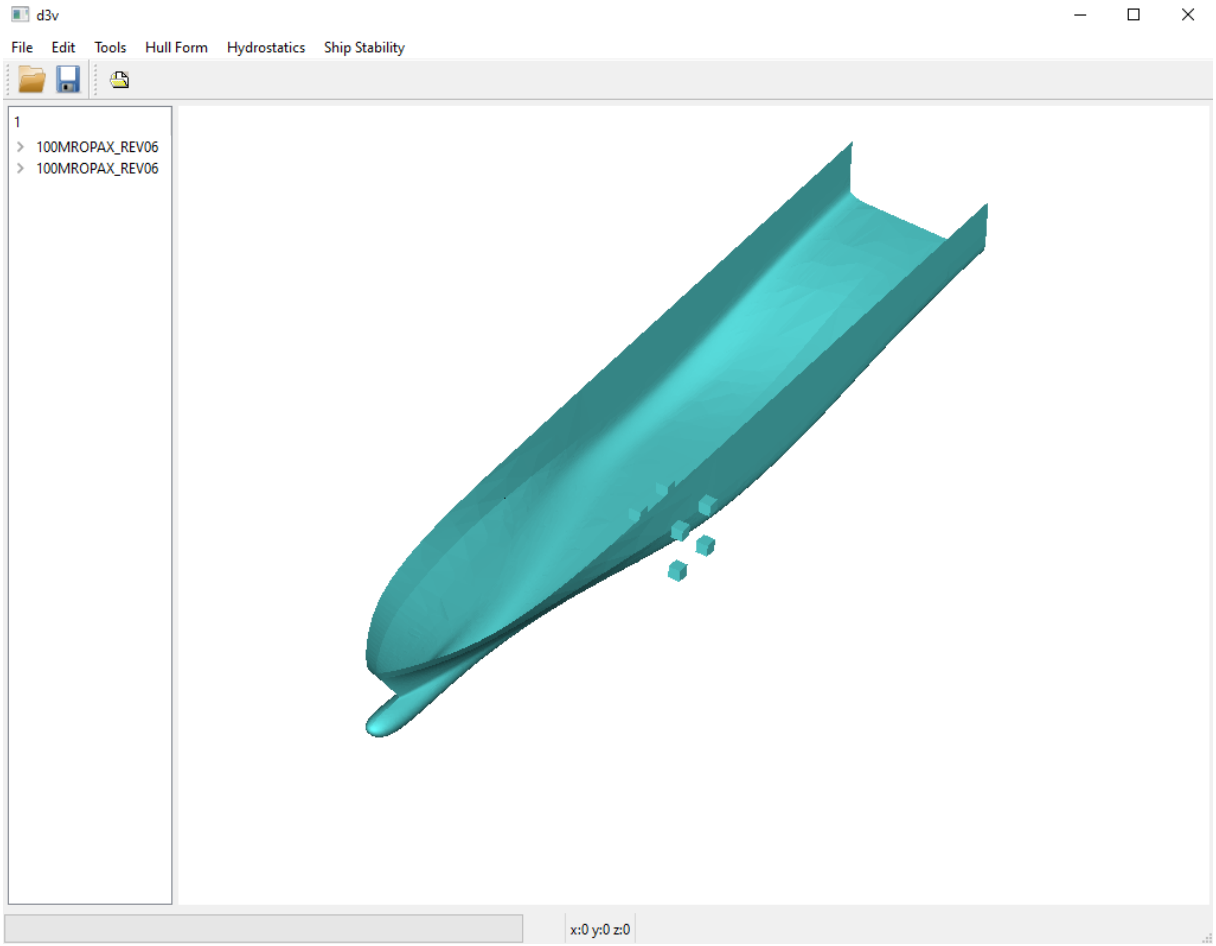
Slika 16 Pristupnje Create FFD Box meniju

Klikom na „Create FFD Box“ otvara meni koji je povezan sa metodom *make_ffd_volume* kako bi interaktivno stvorio novi *FFD* volumen.



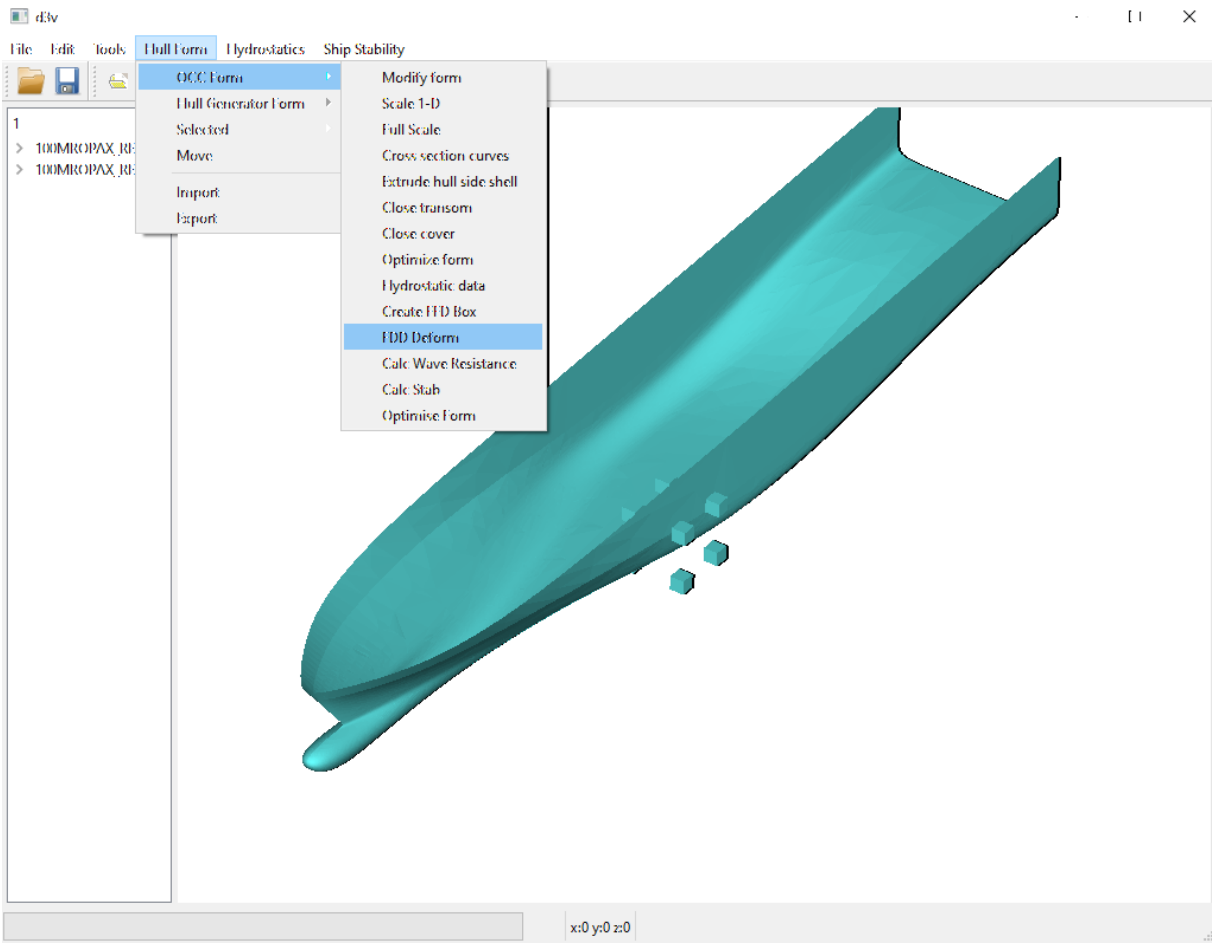
Slika 17 Create FFD Box meni

U njemu se mogu zadati svi potrebni podaci za željeni volumen i klikom na gumb „Create Box“ stvara se novi *FFD* volumen.



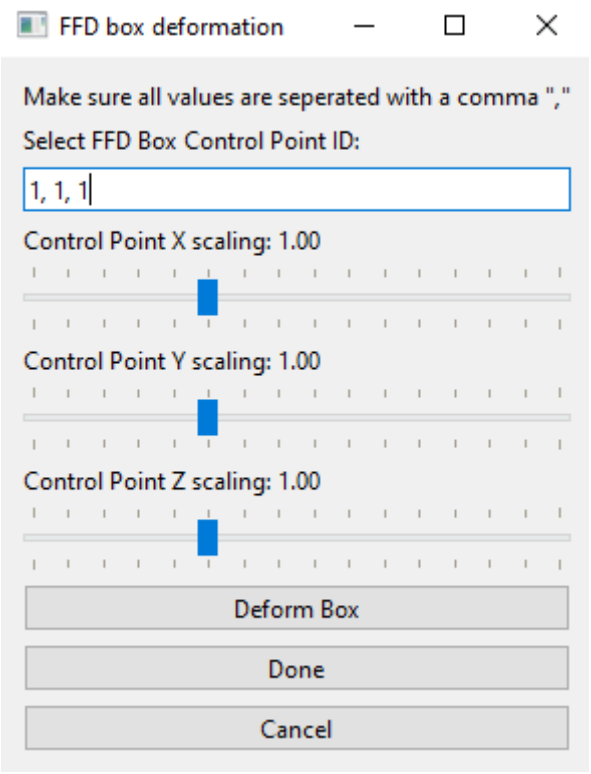
Slika 18 Stvoreni FFD volumen

Meni *DeformFFDBox_menu* je povezan sa klasom *PyGemHullform* i koristi se za interaktivnu deformaciju brodske forme. Pristupa mu se sa klikom na padajući izbornik „Hull Form“, „OCC Form“ i „FFD Deform“.



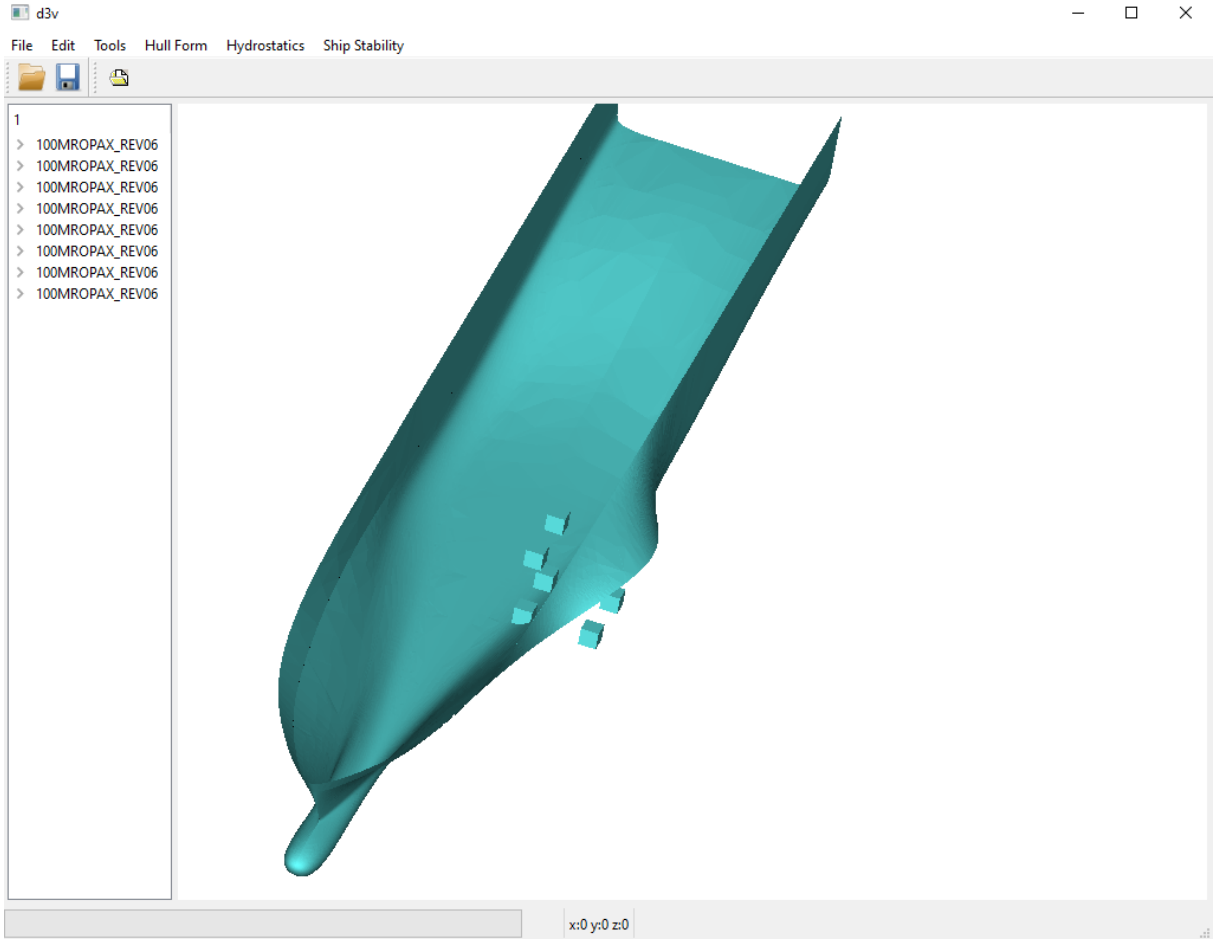
Slika 19 Pristupanje FFD Deform meniju

Nakon klika se otvara meni u kojemu se unosi indeks kontrolne točke volumena, te se sa klizačem može pomaknuti u x, y i z smjeru.



Slika 20 FFD Deform meni

Klikom na „Deform Box“ kontrolna točka na zadanom indeksu se pomiče i forma se deformira.



Slika 21 Brodska forma deformirana FFD metodom

Modul *michell_adaptor.py* je modul zadužen za komunikaciju sa modulom *michell.py* koji se koristi za izračun otpora forme. U sebi sadrži klasu *michell_resitance* koji nasljeđuje klasu *michell* koja se nalazi unutar *michell.py*. Klasa *michell* se koristi za numeričku integraciju forme broda kako bi se izračunao njen otpor, no originalni kod koristi eksplicitnu *NumPy* funkciju kao zapis forme. Iako takav zapis funkcije omogućuje brzu interpolaciju forme kako bi se dobile potrebne točke za numeričku integraciju, njih nije moguće deformirati na jednostavan način, pa za svrhe optimizacije je potrebno omogućiti interpolaciju forme zapisane *NURBS* plohama. Metode u klasi *michell* za izračun otpora koriste *NumPy* polja ispunjena y koordinatama dobivenim pomoću očitavanja forme na nekim x i z koordinatama unutar granica integracije koje određuje sama klasa *michell*. Iako je očitavanje točaka sa *NURBS* ploha lagano ukoliko znamo njihove u i v parametre, to nije slučaj ukoliko znamo željene x i z koordinate, a vrijednosti parametara su nepoznati. Kako bi se omogućila komunikacija sa klasom *michell* i forme broda opisanom *NURBS* ploha napravljen je algoritam za interpolaciju unutar klase

michell_resistance. U početku j napravljena direktna interpolacija pravca i cijele *NURBS* ploha forme, ali se pokazalo da takav način dolaženja do y koordinata je uzimao previše vremena. S obzirom da se interpolacija računa mnogo puta kako bi se samo jednom izračunao otpor, a niti ne govoreći o činjenici da se tokom optimizacije otpor računa stotinama puta, takva metoda se pokazala neiskoristivom. Umjesto toga osmišljen je algoritam koji će u inicijalizaciji na određenim intervalima x osi interpolirati formu broda sa y-z ravninom, te će se dobivene krivulje aproksimirati sa konačnim brojem točaka i na taj način dobiti mrežu točaka pomoću koje se može, linearnom interpolacijom, doći do željenih očitanja. Prednost ove metode je što se generacija mreže točaka se odvija tokom inicijalizacije i nije ju potrebno ponavljati tokom interpolacije, no potrebno ju je ponoviti nakon *FFD* deformacije forme kako bi se dobila nova očitanja. Nažalost, korištenje *FFD* deformacije za pomicanje tih krivulja u svrhu ubrzavanja interpolacije nije korisno, jer te krivulje tokom deformacije ne prate deformiranu formu broda.

U klasi *michell_resitance* se tokom inicijalizacije postavljaju parametri potrebni za numeričku integraciju, među kojima su najvažnije granice integracije po z osi i glavne izmjere broda. Također se pozivaju ostale metode potrebne za pripremu podataka potrebnih za interpolaciju.

Prva od tih metoda je *prepare_hullform_approximation*, u kojoj se definiraju parametri interpolacije. Najvažniji od tih parametara je podjela broda na niz x koordinata na kojima će se nalaziti y-z ravnine za presijecanje forme. Korisnik može definirati gustoću interpolacije ravnina na krmi, sredini i pramcu broda, zajedno sa postotkom ukupne dužine broda koja se smatra kao pramac i krma. Također se može definirati broj točaka koji će se uzeti za aproksimaciju krivulje.

Nakon toga, poziva se metoda *get_form_nurbs* u kojoj se računa interpolacija *NUBRS* ploha i y-z ravnina. Za definiciju ravnina i računanje interpolacija se koriste *OCC* funkcije. Na tim dobivenim krivuljama se potom vrši kontrola protiv pogrešaka npr. krivulja kojoj je dužina 0 se odbacuje. Zadovoljavajuće krivulje se potom aproksimiraju sa točkama i od tih točaka se računaju koeficijent nagiba pravca k i konstanta pravca c za svaki interval po z osi. To se radi kako bi se u funkciju uvrštavao samo x i z za što bržu interpolaciju.

Za interpolaciju po z osi na nekoj od krivulja se koristi funkcija *interpolate_curve* koja koristi linearnu interpolaciju na intervalu u kojemu se z koordinata tražene točke nalazi. Poziva se u metodi *interpolate_surface.*

S obzirom da tražene točke u gotovo svim slučajevima nisu na istoj x koordinati kao i aproksimirane krivulje, potrebno je napraviti još jednu linearnu interpolaciju između dvije susjedne aproksimirane krivulje na istoj z koordinati. Ta interpolacija se odvija u funkciji *interpolate_surface* koja se poziva za vrijeme računanja otpora.

Na kraju se poziva metoda *wave_resistance* iz *michell* klase, kojoj kao ulaznu vrijednost treba dati brzinu broda u čvorovima i vraća silu otpora koja se koristi kao funkcija cilja u optimizaciji.

Modul *pygemoptimisation.py* se koristi za optimizaciju brodske forme uz zadanu funkciju cilja i ograničenja. Komunicira sa modulom za generalnu višeciljnu optimizaciju *moobench.py* i stvaranje konkretnog slučaja za učitanu formu. Sadrži dvije klase: *form_analysis_model* i *form_OptimizationProblem*.

Klasa *form_analysis_model* služi za definiranje procesa, na bazi čijih rezultata se mijenjaju ulazne varijable i ponovo pokreće analiza. Kada se pozove instanca ove klase, u njoj se inicijaliziraju sve potrebne vrijednosti poput početnog otpora, deformacija i hidrostatike. U metodi *analyze* se definirao proces koji će se iterirati za vrijeme optimizacije i davati rezultate za usporedbu. Tu se pomiču redovi *FFD* volumena pomoću metode *move_form_ffd_row*, računa otpor pomoću metode *calc_resistance* i računa istisnina zajedno sa njenim položajem pomoću metode *calc_stab*. Također su definirane „get“ i „set“ svih varijabli, ograničenja i rezultata potrebnih za interakciju sa modulom *form_OptimizationProblem*.

Klasa *form_OptimizationProblem* je glavna klasa vezana za optimizaciju brodske forme. U njoj se definiraju varijable problema, u ovom slučaju pomaci redova *FFD* volumena u smjeru x osi, ograničenja istisnine i njenog položaja, te funkcije cilja koja je u ovom slučaju porivna snaga broda. Varijable se stvara pomoću *DesignVariable* u kojoj je potrebno povezati varijablu sa njenim „get“ i „set“ funkcijama, a dodaje u problem pomoću *add_design_variable* metode. Funkcija cilja se definira pomoću *add_objective* metode u kojoj je potrebno definirati „get“

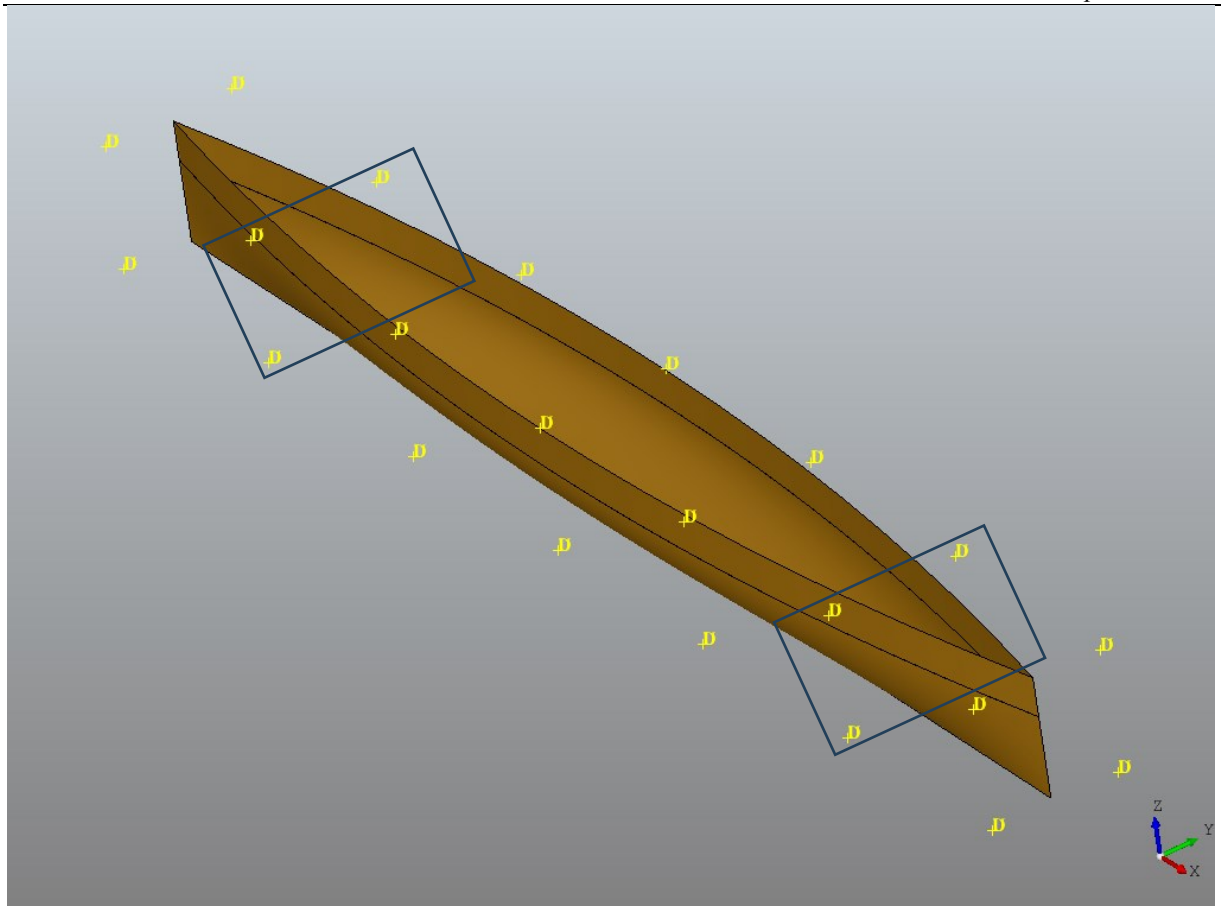
funkciju. Ograničenja se definiraju pomoću *DesignConstraint*, i dodaju pomoću metode *add_constraint.* Ograničenja su postavljena tako da istisnina i njen položaj po x osi moraju biti unutar određenog postotka početnih vrijednosti. Na kraju, sam proces analize problema se postavlja pomoću *add_analysis_executor* metode.

5. PRIMJENA IMPLEMENTIRANIH NOVIH FUNKCIONALNOSTI PROGRAMA D3V-GSD ZA OPTIMIZACIJU JEDNOSTAVNE BRODSKE FORME

U ovom poglavlju će se opisati primjena implementiranih novih funkcionalnosti za rješavanje optimizacijskog problema minimalizacije potrebne snage porivnog stroja u ranim fazama projektiranja broda.

5.1. Definicija optimizacijskog problema

Pri projektiranju broda uobičajena procedura je koristiti brodsku formu sličnog broda koja se skalira na potrebne glavne dimenzije, a nakon toga modificira jednostavnim transformacijama kako bi se postigle zahtijevane hidrostatičke karakteristike odnosno zahtijevani stabilitet. Nakon toga se, najčešće ručnim, modifikacijama pramčanog i krmenog dijela forme nastoji smanjiti otpor generiranja valova. U nastavku će biti definiran jednostavni optimizacijski problem u kojem se modifikacija pramčanog i krmenog dijela nastoji minimizirati snaga pogonskog stroja uz zadovoljenje osnovnih hidrostatičkih karakteristika. Za upravljanje transformacijom forme generira se *FFD* volumen oko cijele brodske forme, s rešetkom od 7x2x2 kontrolnih točaka.



Slika 22 FFD 7x2x2 kavez

FFD deformacija broda će se raditi pomicanjem redova kontrolnih točaka volumena na indeksima 1,2,4,5 u x smjeru. Ti indeksi pripadaju redovima kontrolnih točaka koji imaju najveći utjecaj nad pramcem i krmom, a time i na otpor generiranja valova. . Pomaci tih kontrolnih točaka u x smjeru odabiru se za varijable u optimizacijskom problemu i definirane su u klasi *Form_analysis_model*. U Python modulu, te četiri varijable su nazvane *b_mo_x1* i *b_mo_x2* za pramac, te *a_mo_x1* i *a_mo_x2* za krmu. Pri inicijalizaciji optimizacijskog problema, te vrijednosti su zadane kao 0, što odgovara početnoj formi broda, no tu se mogu promijeniti ukoliko je poželjno da optimizacije kreće iz deformirane forme odnosno ukoliko se za rješavanje optimizacijskog problema koriste gradijentni algoritmi koji za početak rada zahtijevaju početnu točku.

Pri tome je potrebno zadržati istisninu i uzdužni položaj istisnine unutar određenog postotka početne vrijednosti (npr. 1 do 2%). što se u optimizacijski problem dodaje kao četiri ograničenja nejednakosti.

Kao funkcija cilja, zadana je porivna snaga broda. Porivna snaga P_T se dobiva prema iz odnosa efektivne snage P_E i stupnja djelovanja trupa η_H:

$$P_T = \frac{P_E}{\eta_H} \tag{2}$$

Zbog jednostavnosti pretpostaviti će se da je stupanj djelovanja trupa $\eta_H = 0.98$ zbog nedostatka metode njegovog računanja. Efektivna snaga P_E je umnožak ukupne sile otpora R_T i brzine broda v u metrima po sekundi:

$$P_E = R_T \cdot v \tag{3}$$

Sila ukupnog otpora R_T jednaka je:

$$R_T = 0.5\rho_w \cdot v^2 \cdot S \cdot C_T \tag{4}$$

Gdje je gustoća vode $\rho_w = 1000\ \frac{kg}{m^3}$, C_T koeficijent ukupnog otpora i S oplakana površina forme koja se računa aproksimacijom unutar modula *michell.* Koeficijent ukupnog otpora C_T će se rastaviti na sumu koeficijent otpora trenja C_F i koeficijent preostalog otpora C_R. Koeficijent preostalog otpora u sebi sadrži otpor valova C_W, te će se zbog jednostavnosti, ostale komponente zanemariti.

$$C_T = C_F + C_W \tag{5}$$

Koeficijent otpora C_F će se računati prema *ITTC* (eng. *International Towing Tank Conference*) – 1957 [23] korelacijskoj liniji:

$$C_F = \frac{0.075}{(\log R_n - 2)^2} \tag{6}$$

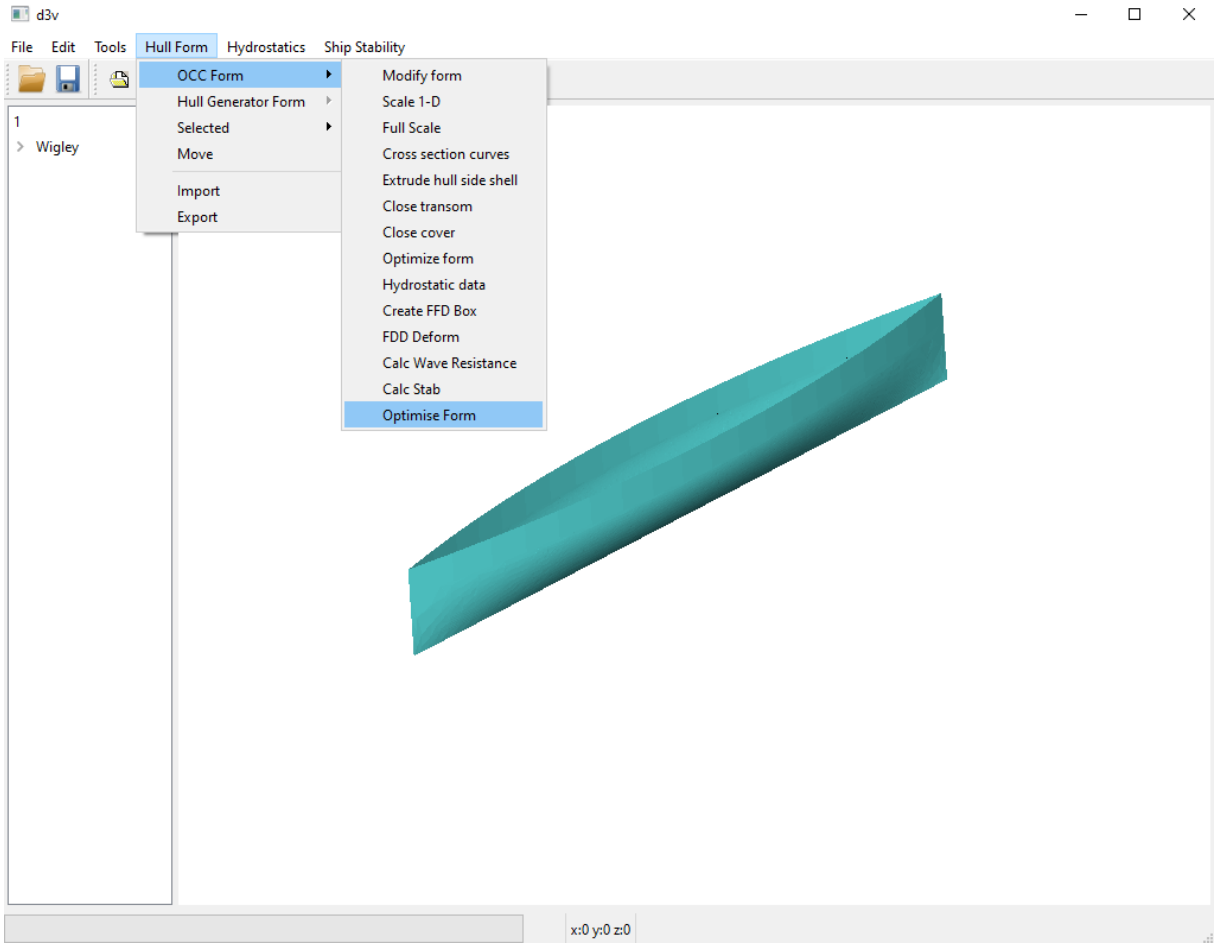
Gdje je *Rn* Reynoldsov broj:

$$R_n = \frac{v \cdot L}{\nu} \tag{7}$$

Gdje je L dužina broda, a ν kinematička viskoznost fluida.

5.2. Pokretanje i podešavanje optimizacije

Za pokretanje optimizacije prvo je potrebno učitati formu iz datoteke, te klikom gumba na „Hull Form“, „OCC Form“ i „Optimise Form“.



Slika 23 Pristupanje optimizaciji forme

U modulu *hullmod_command.py* moguće je zadati brzinu (u čvorovima) za koju će se računati otpor valova odnosno potrebna snaga porivnog stroja. Ona se unosi kao ulazni podatak u instanci klase *Form_OptimizationProblem*. Tu je također moguće izabrati optimizacijski algoritam iz modula *moobench.py* koji će se koristiti, zajedno sa njihovim vrijednostima. U ovom radu najviše se radilo pomoću genetskog algoritma, no omogućeno je korištene i ostalih poput *SLSQP* i *Nelder-mead* koji su dostupni putem biblioteke moobench. Za genetski algoritam može se podesiti broj populacije i iteracije u varijablama *pop_size* i *num_iter*.

Unutar modula *michell_adaptor.py* moguće je promijeniti preciznost aproksimacije brodske forme pomoću krivulja. Varijable unutar klase *michell_resistance* koje se mogu podesiti:

- *bow_n_curves*
- *aft_n_curves*
- *center_n_curves*
- *bow_range*
- *aft_range*
- *n_z_samples*

Varijable *bow_n_curves*, *aft_n_curves* i *center_n_curves* određuju količinu ravnina s kojima će se brodska forma interpolirati na pripadnim mjestima, a time, i broj krivulja. Varijable *bow_range* i *aft_range* opisuju, u postotku ukupne dužine broda, koja se smatraju krmom i pramcem. Na kraju, *n_z_samples* opisuje s koliko točaka će se te krivulje aproksimirati. Važno je primijetiti da sa povećavanjem ovih varijabli se povećava preciznost i potrebno vrijeme računa.

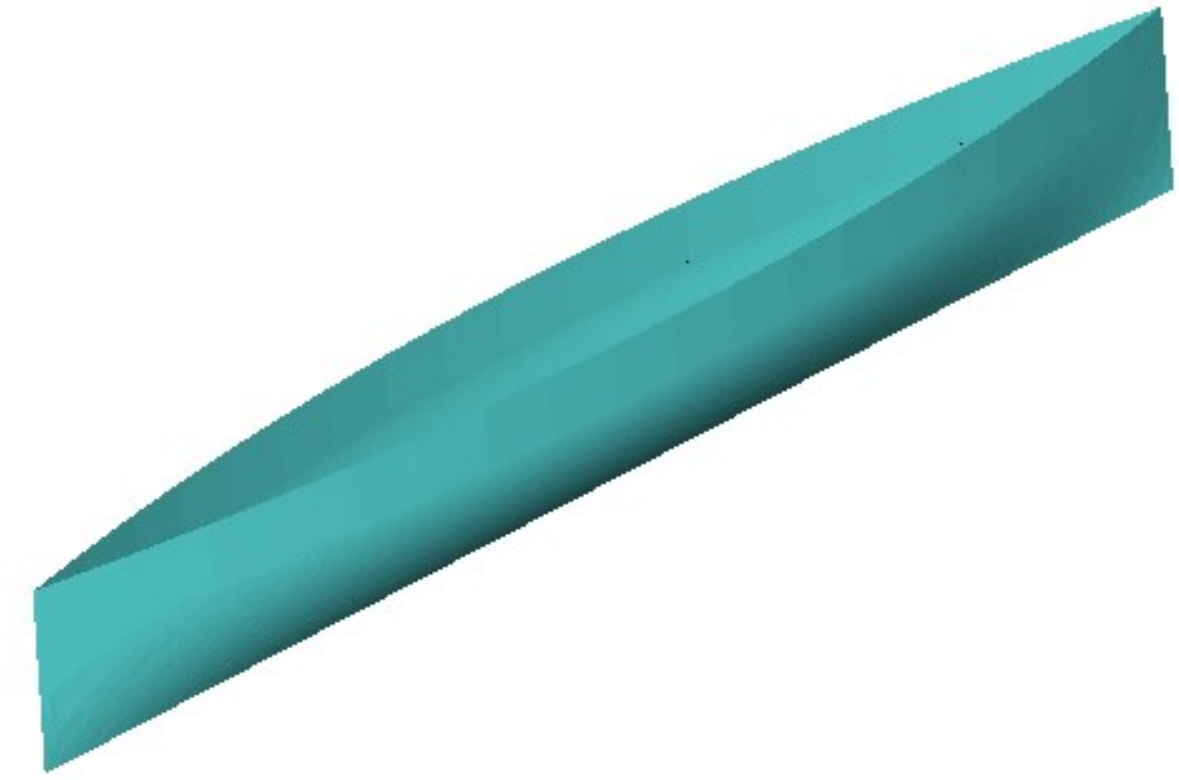
5.3. Primjeri optimizacije

Rješavanje optimizacijskog problema testirano je na Wigley formi zapisanoj *NURBS* plohama. To je testna forma koja se vrlo često koristi za usporedbu algoritama za računanje otpora ili kalibraciju mjerenja. U ovom radu korištene su dimenzije wigley forme iz rada [22]

L = 4 m

T = 0.25 m

Brzina za koju se traži minimalna porivna snaga je 3.653 čv



Slika 24 Wigley Forma

Riješena su dva optimizacijska problema koristeći genetski algoritam. U oba slučaja očekivalo se ostvariti manju potrebnu porivnu snagu od početne, pridržavajući se zadanih ograničenja i time dokazati da je algoritam sposoban konvergiranju prema boljem rješenju.

U prvom primjeru kao ograničenje je zadan samo dozvoljeni raspon istisnina, dok je u drugom korišten i dozvoljeni raspon uzdužnog položaja težišta. Oba primjera su odrađena na istoj gustoći aproksimacije:

bow_n_curves = 20

aft_n_curves = 20

center_n_curves = 20

bow_range = 0.15

aft_range = 0.15

n_z_samples = 12

Karakteristike inicijalne forme:

Cw = 0.001522

Pe = 37.678 W

$\Delta = 0.1774\ m^3$

LCB = - 0.002901

5.3.1. Primjer 1

Gornje i donje vrijednosti projektnih varijabli:

-0.1 <= b_mo_x1 <= 0.1

-0.1 <= b_mo_x2 <= 0.1

-0.1 <= a_mo_x1 <= 0.1

-0.1 <= a_mo_x2 <= 0.1

Ograničenja:

0.1739 <= Δ <= 0.181

Cilj:

Minimizacija snage porivnog stroja Pe:

Optimizacijski algoritam i kontrolni parametri:

Genetski algoritam (bibliteka pymoo)

pop_size = 30

num_iter = 30

Rezultati:

Runtime = 31803 s

b_mo_x1 = -0.09996

b_mo_x2 = 0.06141

a_mo_x1 = 0.00532

a_mo_x2 = 0.09998

Cw = 0.001123

Pe = 34.464 W

Δ <= 0.1739 m^3

5.3.2. Primjer 2

Gornje i donje vrijednosti projektnih varijabli:

-0.1 <= b_mo_x1 <= 0.1

-0.1 <= b_mo_x2 <= 0.1

-0.1 <= a_mo_x1 <= 0.1

-0.1 <= a_mo_x2 <= 0.1

Ograničenja:

0.1739 <= Δ <= 0.181

-0.08029 <= LCB <= 0.0797

Cilj:

Minimizacija snage porivnog stroja Pe:

Optimizacijski algoritam i kontrolni parametri:

Genetski algoritam (bibliteka pymoo)

pop_size = 40

num_iter = 40

Rezultati:

Runtime = 31941 s

b_mo_x1 = -0.091326

b_mo_x2 = 0.019702

a_mo_x1 = -0.030557

a_mo_x2 = 0.099897

Cw = 0.001117

Pe = 34.416 W

Δ = 0.1739 m^3

LCB = -0.003721

U primjeru 1 postignuto je smanjenje snage za 9.3%, a u primjeru 2 za 9.5%.

6. ZAKLJUČAK

D3V-GSD-OCC je ekstenzija osnovnog programa *D3V-GSD* sa funkcionalnostima koje omogućuje biblioteka *OpenCascade*, odnosno mogućnost rada s brodskom formom zapisanom pomoću *NURBS* ploha. U ovom radu, toj ekstenziji je dodana funkcionalnost biblioteke *PyGem* za transformaciju brodske forme koristeći metodu slobodne deformacije oblika. Dodane su mogućnosti jednociljne optimizacije s ograničenjima, pri čemu je za cilj moguće koristiti minimalizaciju porivne snage motora. U grafičko korisničko sučelje programa dodani su meniji koji omogućuju interaktivnu transformaciju forme.

Primjena implementiranog proširenja funkcionalnosti programa d3v-gsd prikazana je na primjeru optimizacije jednostavne forme broda.

U okviru diplomskog rada utvrđeni su i neki mogući daljnji koraci na razvoju biblioteke *D3V-GSD* odnosno proširenja *D3V-GSD*:

- izrada novog, bržeg, algoritma za određivanje točaka s forme broda zapisane putem *NURBS* ploha i biblioteke OpenCascade
- integracija drugih, po mogućnosti besplatnih biblioteka za točnije predviđanje otpora, odnosno porivne snage broda.

LITERATURA

[1] https://glomeep.imo.org/technology/hull-form-optimization/

[2] https://en.wikipedia.org/wiki/Polygon_mesh

[3] https://www.androidauthority.com/qualcomm-opengl-es-3-snapdragon-600-800-156906/

[4] https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline

[5] https://www.researchgate.net/figure/NURBS-surface-and-its-parametric-plane_fig3_228403711

[6] https://www.wiki.mcneel.com/_media/legacy/en/w_uvsrf.png

[7] https://www.researchgate.net/figure/Cubic-NURBS-curve-with-control-points-5-0-0-0-0-075-7-075-8-0_fig2_253449097

[8] https://www.researchgate.net/figure/Example-of-a-closed-NURBS-curve-with-the-associated-control-points-and-the-effect-of-the_fig1_346148495

[9] https://en.wikipedia.org/wiki/Open_Cascade_Technology

[10] https://en.wikipedia.org/wiki/Computer_graphics

[11] F. Salmoiraghia , A. Scardiglibc∗ , H. Telibb and G. Rozzaa. Free Form Deformation, mesh morphing and reduced order methods: enablers for efficient aerodynamic shape optimization. Italy.

[12] https://www.researchgate.net/figure/Free-Form-Deformation-FFD-2_fig1_316253868

[13] https://en.wikipedia.org/wiki/Python_(programming_language)

[14] https://en.wikipedia.org/wiki/SciPy

[15] https://numpy.org/doc/stable/user/whatisnumpy.html

[16] https://www.graphics.rwth-aachen.de/media/openmesh_static/Documentations/OpenMesh-8.0-Documentation/index.html

[17] https://wiki.qt.io/About_Qt

[18] Evandro Ferreira de Paula Filho. Hull form modelling concepts and optimization in early design stage. Master Thesis, Universidade da Coruña, 2022/2023.

[19] Marco Tezzele, Nicola Demo, Andrea Mola, Gianluigi Rozza. PyGeM: Python Geometrical Morphing. 2021.

[20] Veron Hrvojić. Usporedba rješenja višeciljne optimizacije na primjeru dimnzioniranja jednostavne brodske konstrukcije. Diplomski rad, 2022, Fakultet Strojarstva i Brodogradnje.

[21] T. Sederberg, S. Parry. Free-form deformation of solid geometric models. 1986.

[22] Josip Bašić, Branko Blagojević, Martina Andrun. Improved estimation of ship wave-making resistance. 2020.

[23] https://ittc.info/media/3089/index.pdf