Modeliranje i izračun karakteristika forme broda primjenom Python programskog sučelja aplikacije Rhinoceros

Rebić, Lovro Ante

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Lovro Ante Rebić

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

dr. sc. Pero Prebeg, dipl. ing.

Student:

Lovro Ante Rebić

Zagreb, 2020.

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

Zahvaljujem se mentorima dr.sc. Peri Prebegu i mag.ing. Ivanu Muniću na pomoći, sugestijama i strpljenju prilikom izrade završnog rada.

Također se želim zahvaliti svojoj djevojci, obitelji i prijateljima na podršci i razumijevanju tijekom studija.

Lovro Ante Rebić

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:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Lovro Ante Rebić** Mat. br.: 0035205197

Naslov rada na hrvatskom jeziku: **Modeliranje i izračun karakteristika forme broda primjenom Python programskog sučelja aplikacije Rhinoceros**

Naslov rada na engleskom jeziku: **Use of Rhinocerous and its Python application program interface for hull form modelling and hydrostatic characteristics calculation**

Opis zadatka:

Analitičko zadavanje forme broda ima široku primjenu u ranim fazama projektiranja broda, jer omogućuje jednostavno modificiranje forme broda putem promjene manjeg broja parametara. Aplikacija Rhinoceros omogućuje jednostavno modeliranje NURBS ploha temeljem zadanog polja točaka. Aplikacija posjeduje i bogato programsko sučelje (API) pomoću kojeg je iz programskih jezika, poput Pythona, moguće izraditi module koji proširuju osnovne funkcionalnosti aplikacije. U radu je potrebno, primjenom Rhinoceros API-a i programskog jezika Python izraditi module za izračun i ispis hidrostatičkih karakteristika broda.

Zadatak obuhvaća sljedeće:

- upoznavanje s aplikacijom Rhinoceros i njegovim Python programskim sučeljem
- odabir hidrostatičkih karakteristika forme bez privjesaka za koje će se izraditi matematički model, a koje minimalno moraju uključiti karakteristike vidljive na uobičajenom dijagramnom listu
- upoznavanjem s parametarskim modeliranjem forme trupa i nadgrađa fregate putem analitički zadanih brodskih linija
- izradu NURBS modela forme trupa i nadgrađa koristeći aplikaciju Rhinoceros, temeljem polja točaka generiranih iz parametarskog modela fregate
- izradu Python modula za izračun odabranih hidrostatičkih karakteristika forme fregate putem programskog sučelja aplikacije Rhinoceros odnosno funkcija za računanje površina i volumena te njihovih momenata
- usporedbu izračunatih hidrostatičkih karakteristika s proračunom pomoću Orca3D programskog dodatka aplikaciji Rhinoceros
- izradu Python modula koji omogućuje crtanje dijagramnog lista i tablični ispis izračunatih hidrostatičkih karakteristika.

U radu treba navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:
28. studenog 2019.

Datum predaje rada:
1. rok: 21. veljače 2020.
2. rok (izvanredni): 1. srpnja 2020.
3. rok: 17. rujna 2020.

Predviđeni datumi obrane:
1. rok: 24.2. – 28.2.2020.
2. rok (izvanredni): 3.7.2020.
3. rok: 21.9. - 25.9.2020.

Zadatak zadao:
Doc. dr. sc. Pero Prebeg

Predsjednica Povjerenstva:
Prof. dr. sc. Nastia Degiuli

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
KM_L	[m]	visina uzdužnog metacentra iznad osnovice
KM_0	[m]	visina poprečnog metacentra iznad osnovice
X_{CB}	[m]	uzdužni položaj težišta istisnine
A_{WL}	[m^2]	površina vodne linije
X_{WL}	[m]	uzdužni položaj težišta vodne linije
Z_{WL}	[m]	vertikalni položaj težišta vodne linije
I_L	[m^4]	uzdužni moment inercije vodne linije
I_B	[m^4]	poprečni moment inercije vodne linije
V	[m^3]	volumen istisnine
Δ	[t]	istisnina
KB	[m]	vertikalni položaj težišta istinine
M_0B	[m]	poprečni metacentarski radijus
M_LB	[m]	uzdužni metacentarski radijus
L	[m]	duljijna broda
L_{WL}	[m]	duljina vodne linije
B	[m]	širina broda
T	[m]	gaz broda
M_1	[tm/m]	jedinični moment trima
C_B	[-]	koeficijent punoće
C_{WL}	[-]	koeficijent vodne linije
C_P	[-]	prizmatički koeficijent
C_X	[-]	koeficijent glavnog rebra

SAŽETAK

U ovom radu prikazano je modeliranje i izračun karakteristika forme broda primjenom Python programskog sučelja aplikacije Rhinoceros. Na temelju polja točaka dobivenih iz parametarskog modela fregate te interpoliranjem kroz njih dobivene su vodne linije. Međusobnim povezivanjem vodnih linija generira se forma trupa broda nakon čega se računaju hidrostatske karakteristike forme. Izrađuje se programsko sučelje API pomoću kojeg je iz programskog jezika Python-a moguće izraditi module koji proširuju osnovne funkcionalnosti aplikacije tj. module za izračun hidrostatskih karakteristika forme broda . Na temelju dobivenih hidrostatskih karakteristika izrađuje se Python modul koji omogućava crtanje dijagramnog lista i tablični ispis izračunatih hidrostatskih karakteristika. Na kraju je dana usporedba hidrostatskih karakteristika s proračunom pomoću Orca3D programskog dodatka aplikaciji Rhinoceros.

Ključne riječi: vodne linije, forma trupa broda, hidrostatske karakteristike, Python programsko sučelje

SUMMARY

This paper presents modeling and calculation of ships shape characteristics using the Python program interface of the Rhinoceros application. Based on the field of points given from the frigate parameter model and interpolating through them, water lines are obtained. Connecting of water lines generates the ships hull form, after which the hydrostatic characteristics of the form are calculated. An API programming interface is created that allows Python programming language to produce modules that extend the basic functionality of the application, i.e. modules for calculating the hydrostatic characteristics of a ships shape. On the basis of the obtained hydrostatic characteristics, Python module, that allows drawing a diagram sheet and a table listing of the calculated hydrostatic characteristics, is created. At the end is given a calculation of hydrostatic characteristics, and compared with the calculation of Orca3D add-on to Rhinoceros.

Key words: water lines, ship hull form, hydrostatic characteristic, Python programming interface

1. UPOZNAVANJE S PARAMETARSKIM MODELIRANJEM FORME

1.1. Općenito o parametarskom modeliranju

Razvoj CAD sustava za parametarsko modeliranje pomoću značajki bitno je smanjio vrijeme potrebno od ideje pa do njezine realizacije, odnosno pojave novog proizvoda na tržištu. Korištenjem računala i računalom potpomognutog konstruiranja omogućilo je tako i smanjivanje grešaka koje se javljaju unutar konstrukcijskog procesa [1]. CAD sustavi za parametarsko modeliranje primjenjuju se radi izrade prostornih 3D računalnih parametarskih modela temeljenih na značajkama. Iz prostornog modela se pomoću ugrađenih rutina izrađuju plošni 2D crteži odnosno ravninske projekcije modela. Crteži su u većini sustava za parametarsko modeliranje dvosmjerno povezani s modelom.
Gotovo uvijek slijedi jednaki niz koraka prilikom izrade parametarskog modela [2]:

- uočavanje osnovne značajke i preostalih značajki dijela uz razmatranje o prikladnom redoslijedu izrade značajki
- skiciranje i označavanje profila osnovne značajke
- izrada osnovne značajke istiskivanjem, rotacijom ili pomicanje profila osnovne značajke
- dodavanje preostalih značajki
- izrada tehničke dokumentacije.

Parametarsko modeliranje koristi računalo za konstruiranje objekata ili sustava koji komponentama modela pripisuje realno ponašanje. Parametarski modeli koriste alate za modeliranje površina i 3D tijela temeljene na značajkama koje im omogućuju jednostavnije manipuliranje karakteristikama modela. Jedna od najvažnijih značajki parametarskog modeliranja je da dimenzije koje su međusobno povezane automatski mijenjaju svoje iznose. Drugim riječima, parametarsko modeliranje omogućava projektantu da definira čitav niz oblika, a ne samo određene varijante. Prije pojave parametara, uređivanje oblika nije bilo lak zadatak za projektante. Na primjer, da bi izmijenio samo jednu dimenziju modela, projektant je morao promijeniti duljinu, širinu i visinu.

Međutim, kod parametarskog modeliranja, projektant treba izmijeniti samo jedan parametar; ostala dva parametra se automatski podešavaju. Dakle, parametarski se modeli usredotočuju na korake u stvaranju oblika i parametriziraju ih. To je izuzetno korisno pružateljima usluga konstruiranja modela.

Parametarski proces modeliranja - Parametarski modeli se grade iz skupa matematičkih jednadžbi. Da bi parametarski modeli bili ispravni, moraju se temeljiti na stvarnim podacima o projektu. Postoje dvije popularne metode parametarskog modeliranja:

- konstruktivna geometrija 3D tijela (eng. *Constructive solid geometry* - CSG) - CSG definira model u smislu kombiniranja osnovnih (primitivnih) i generiranih (korištenjem operacije ekstrudiranja i pomicanja) čvrstih oblika. Za izradu modela koristi boolove operacije. CSG je kombinacija 3D geometrijskih tijela (na primjer cilindra, konusa, prizme, pravokutnika ili sfere) kojima se onda manipulira pomoću jednostavnih logičkih operacija
- granični prikaz (eng. *Boundary representation* - BR) - U BR-u se formira čvrsti model definiranjem površina koje formiraju njegove prostorne granice (točke, rubovi itd.) Objekt se dobije spajanjem tih prostornih točaka.

Prednosti 3D modeliranja u odnosu na tradicionalno 2D crtanje [3]:

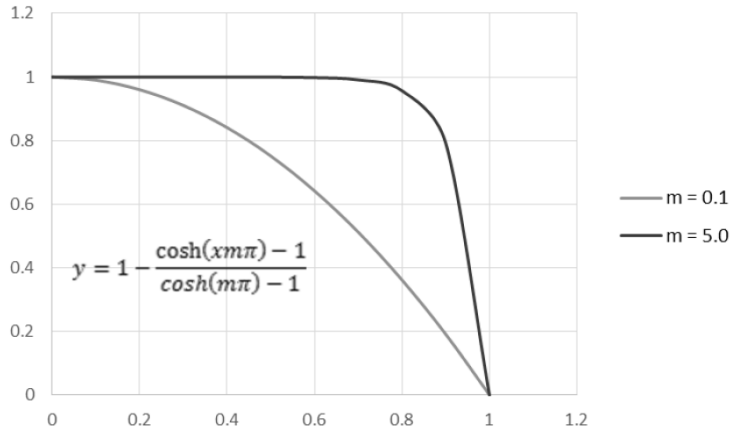
- sposobnost izrade promjenjivih dizajna
- mogućnost gledanja 3D modela iz raznih kuteva
- bolja integracija s drugim aplikacijama i skraćeno vrijeme modeliranja
- mogućnost korištenja postojećeg dizajna za kreiranje novog
- jednostavna uporaba i poboljšanje učinkovitosti.

1.2. Generiranje forme trupa putem analitički zadanih brodskih linija [4]Error! Reference source not found.

Ova metoda opisuje trup kao niz vodnih linija koje su sastavljene od segmenata opisanih funkcijama. Kako su vodne linije i odnos između vodnih linija definirani glatkim kontinuiranim funkcijama, rezultirajuća površina trupa također mora biti glatka i kontinuirana. Generator trupa pretpostavlja da je oblik trupa sastavljen od niza vodnih linija opisanih linijama i krivuljama koji imaju različite zapise ovisno o položaju vodne linije. Oblik trupa može se izvesti iz najosnovnijih konstrukcijih parametara, uključujući:

- potrebno opterećenje tereta
- blok koeficijent ili prizmatički koeficijent
- koeficijent glavnog rebra
- širina ili gaz (ovisno o potrebama korisnika).

Generator trupa je približan model, što znači da je potrebno podešavanje kako bi se osiguralo da je položaj težišta istisnine i promijena volumena različitih vodnih linija, u skladu sa stvarnim oblicima trupa. Utvrđeno je da oblik krivulje na Slici 1. može zadovoljiti većinu formi trupa. Oblici krivulja koje čine oblik trupa mijenjaju se modificiranjem koeficijenta, prikazanog kao „m“ na Slici 1., kako bi se udovoljilo izračunatoj istisnini broda. Ispitane su brojne krivulje zbog različitosti njihovih oblika. Zbog svog oblika i sposobnosti mijenjanja istog, izabrana je krivulja hiperbolnog kosinusa. Formula je zatim derivirana kako bi se potrebni dio krivulje pomaknuo i fiksirao u kvadrat dimenzija 1x1, kao što je prikazano na Slici 1. To omogućava da se dijelovi presjeka površine vodne linije, predstavljeni krivuljom, dobiju množenjem duljine i širine presjeka s integralom krivulje, u granicama od nula do jedan.

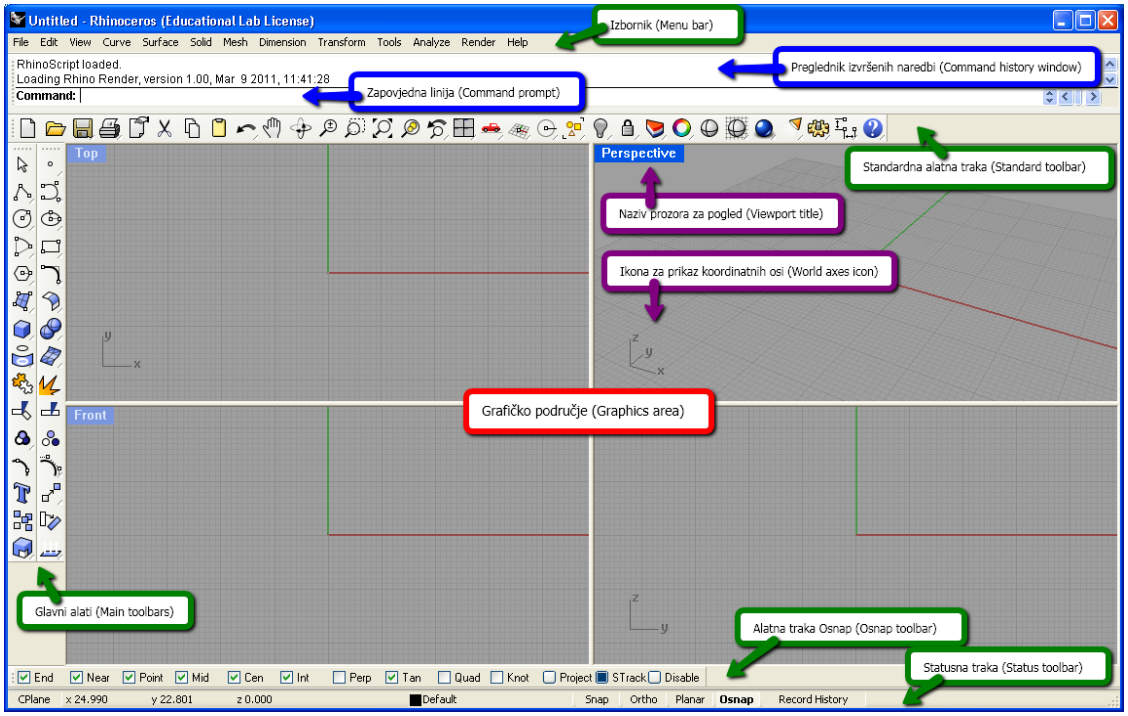


Slika 1. Krivulja korištena za opis forme pramčane i krmene vodne linije [4]

2. UPOZNAVANJE S APLIKACIJOM RHINOCEROS

2.1. Općenito o Rhinoceros aplikaciji

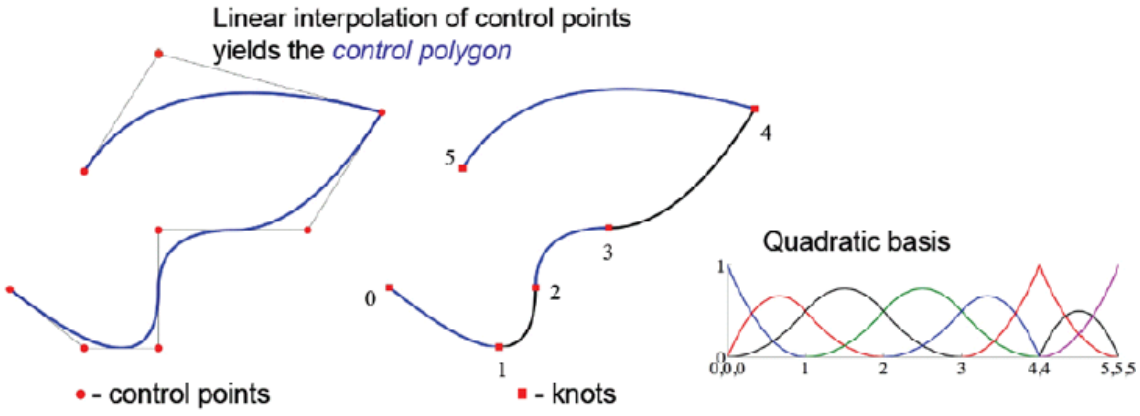
Rhinoceros ili skraćeno Rhino je 3D aplikativni softver koji objedinjuje računalnu grafiku i računalni programski dizajn. Rhino geometrija temelji se na NURBS matematičkom modelu usredotočenom na dobivanje matematički preciznog prikaza krivulja i površina slobodnih oblika u računalnoj grafici, za razliku od aplikacija zasnovanih na poligonskim mrežama [6]. Jedne od glavnih značajki Rhino aplikacije su izuzetno brz 3D prikaz, neograničene veličine te mogućnost pogleda iz svih kuteva. Rhino nema ograničenje na kompleksnost rada i stupanj složenosti, nego samo na hardverske mogućnosti računala. Također velika prednost Rhino aplikacije je mogućnost korištenja raznih programskih jezika, kao što su Python, Visual Basic for Applications, Grasshopper što ga čini pristupačnim svim korisnicima. Korisničko sučelje, Slika 2. jednostavno je i pregledno, te su sve opcije razvrstane po karticama. . Često se koristi u raznim granama industrije, poput arhitekture, industrijskog dizajna (auto-industrija, dizajn brodova i slično), kao i dizajna nakita [11].



Slika 2. Sučelje aplikacije Rhino [5]

2.2. Generiranje krivulja iz točaka

Dobivanje glatke krivulje iz točaka najjednostavnije je interpolacijom krivulje kroz te točke. Rhino to omogućuje naredbom „InterpCrv“(eng. *Interpolate curve*). Neki od osnovnih parametara koji opisuju krivulju su njezin stupanj (eng. *Degree*) i čvorovi (eng. *Knots*). Stupanj određuje stupanj krivulje ili površine. Prilikom crtanja krivulje visokog stupnja, dobivena krivulja neće biti traženog stupnja ukoliko nema barem jednu kontrolnu točku više od traženog stupnja. Na primjer, krivulja četvrtog stupnja mora imati minimalno pet kontrolnih točaka. Kontrolne točke su skup točaka koji se koristi za određivanje oblika krivulje, površine ili nekog višedimenzionalnog objekta. Na Slici 3. prikazana je krivulja n-tog reda s pripadajućim čvorovima i kontrolnim točkama [12].



Slika 3. Prikaz interpolacijske krivulje kroz čvorove [7]

Čvor određuje kako je interpolirana krivulja parametrizirana. Kada je razmak između odabranih točaka jednak za cijelu krivulju, sve tri parametrizacije generiraju istu krivulju. Kod crtanja interpolacijske krivulje, odabrane točke pretvaraju se u čvorne vrijednosti te krivulje. Rhino nudi mogućnost odabira raznovrsnih tipova čvorova, među kojima su Chord, SqrtChord, PersistentClose, StartTangent i EndTangent. Pomoću njih određuje raspored kontrolnih točaka i ponašanje krivulje.

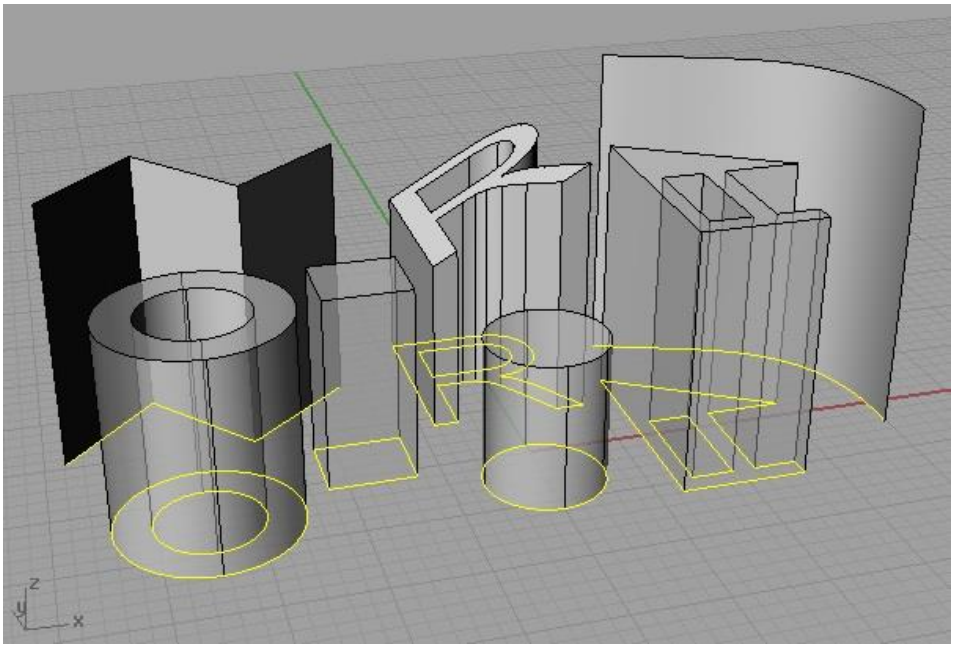
2.3. Generiranje ploha u iz krivulja [12]

Generiranje ploha iz krivulja u Rhino-u moguće je nizom naredbi od kojih će u okviru ovog završnog zadatka biti spomenute samo neke:

- ekstrudiranje krivulje (eng. *extrude curve*)
- rubna površina (eng. *edge surface)*
- loft
- mrežna površina (eng. *network surface)*
- ravninska površina (eng. *planar surface)*
- sweep.

2.3.1. Ekstrudiranje krivulje

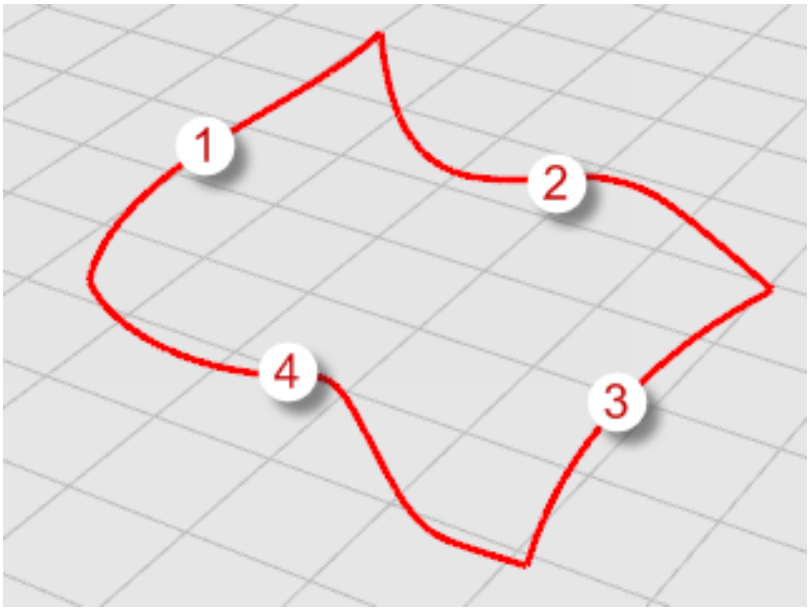
Ekstrudiranje krivulje ili presjeka po pravcu. Ako krivulja ili presjek nisu ravninski tj. ne nalaze se ni u jednoj od tri ravnine, ekstrudiranje će se vršiti u z-smjeru aktivnog gledišta konstrukcijske ravnine. Ako su krivulja i presjek ravninski, ekstrudiranje će se vršiti u smjeru okomitom na konstrukcijsku ravninu. Primjer ekstrudiranja u Rhino-u na Slici 4.



Slika 4. Primjer ekstrudiranja u Rhino [8]

2.3.2. Rubna površina

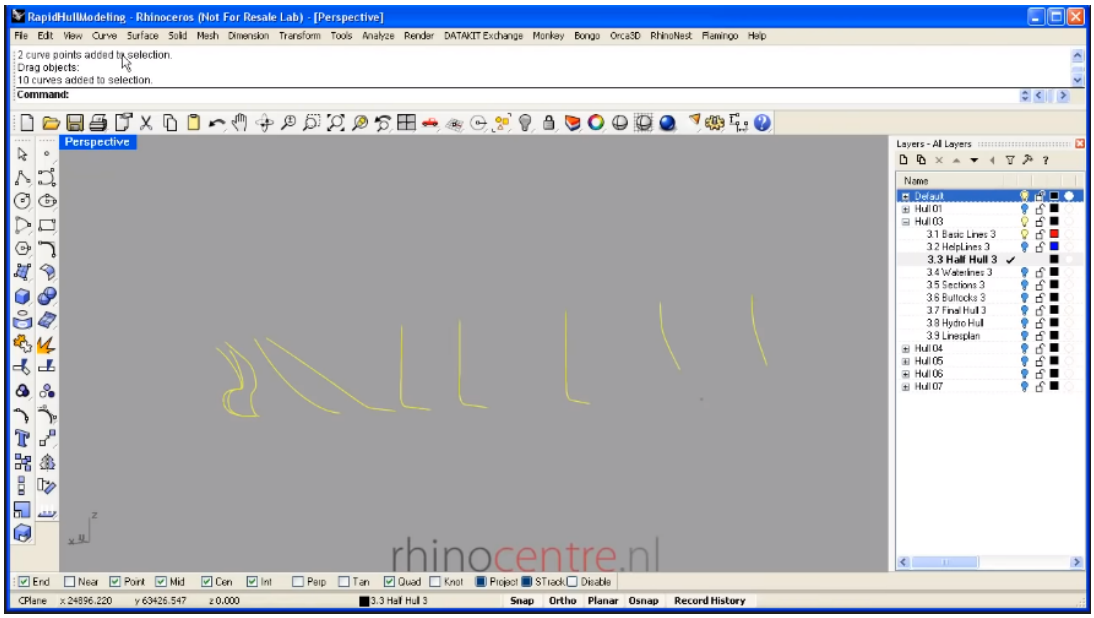
Kreiranje površine od dvije, tri ili četiri krivulje.



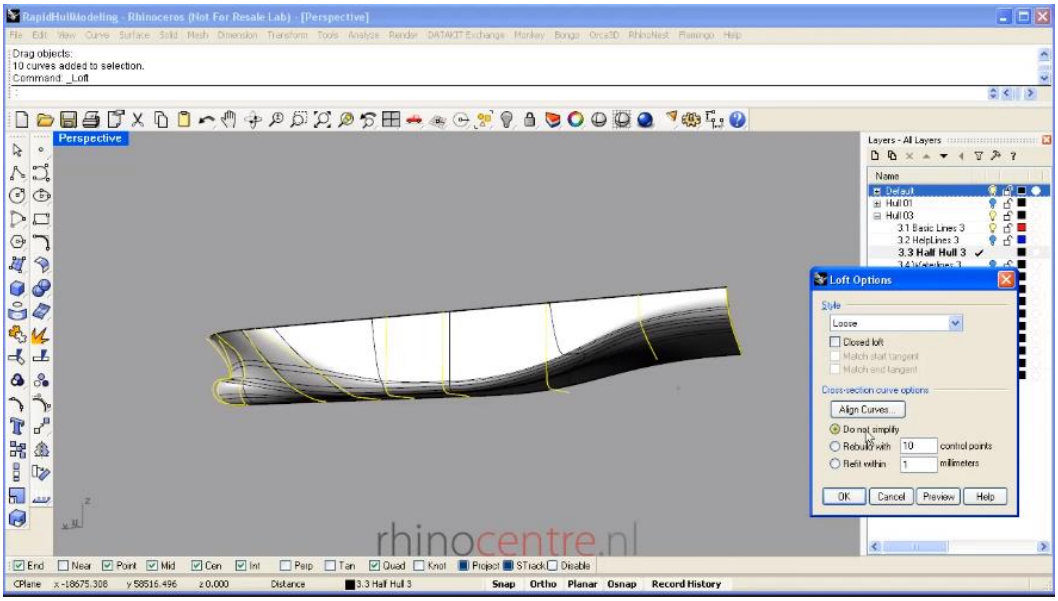
Slika 5. Generiranje plohe korištenjem rubnih krivulja [9]

2.3.3. Loft

Naredba Loft može se povezati s građom broda. Ravninski presjeci predstavljaju rebra broda raspoređena po njegovoj duljini. Lofting kreira površinu međusobnim povezivanjem rebara broda tj. ravninskih presjeka. Poprečni presjeci definiraju oblik konačne površine.



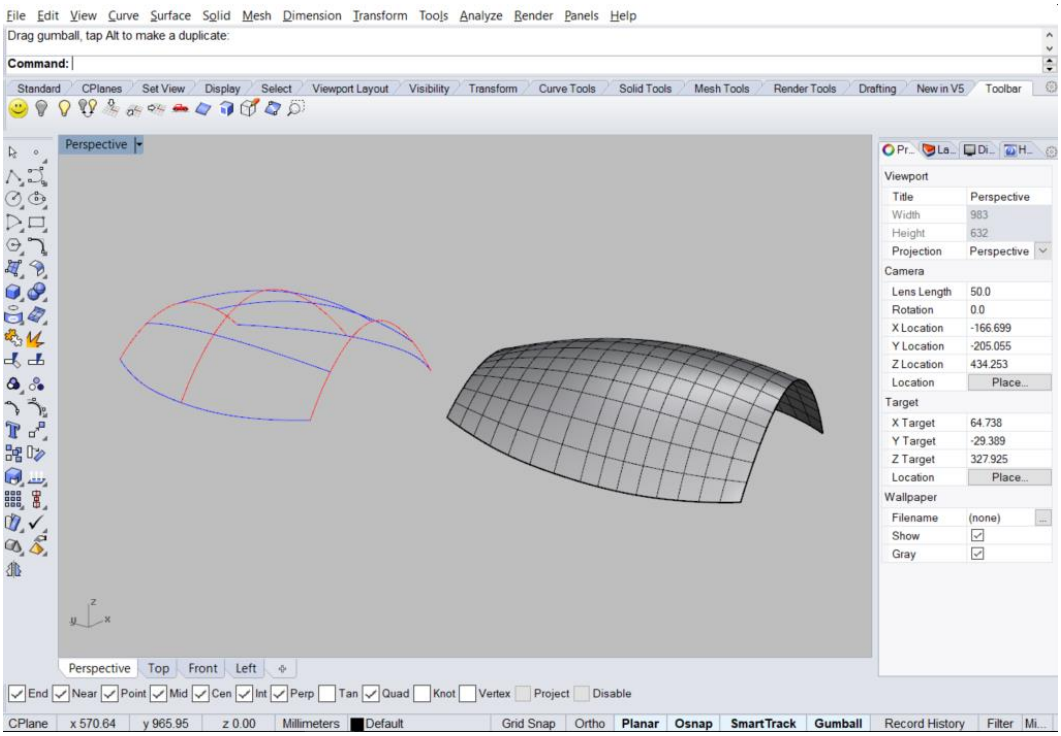
Slika 6. Generiranje trupa broda korištenjem Loft-a [10]



Slika 7. Trup broda dobiven Loft naredbom [10]

2.3.4. Mrežna površina

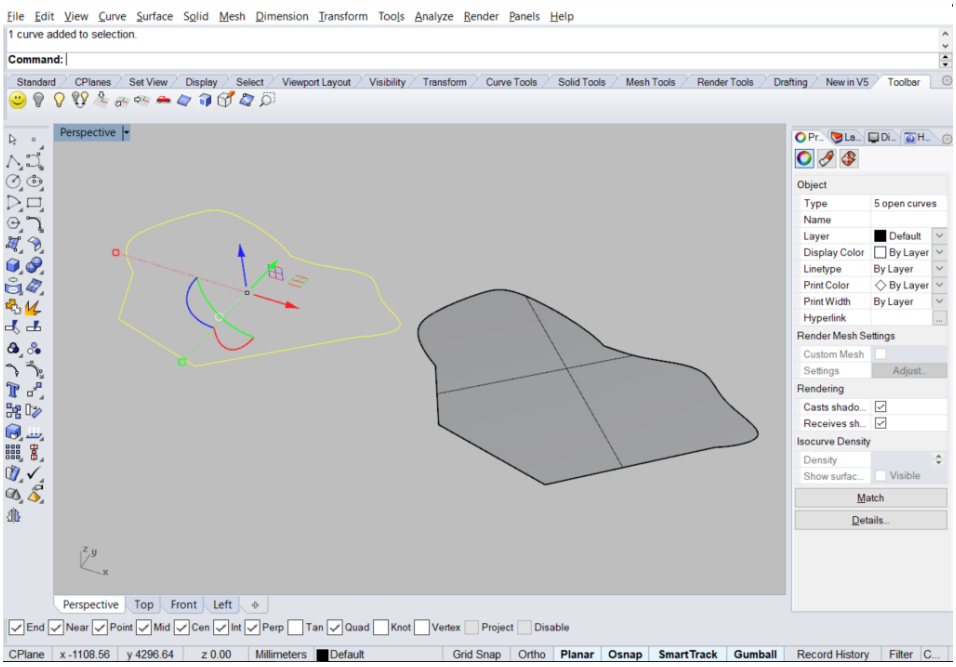
Kreiranje površine od mreže presijecajućih krivulja. Sve krivulje u jednom smjeru moraju presijecati sve krivulje u drugom smjeru, dok se krivulje koje se pružaju u istom smjeru ne smiju međusobno sjeći.



Slika 8. Generiranje plohe korištenjem mreže krivulja [12]

2.3.5. Planar surface

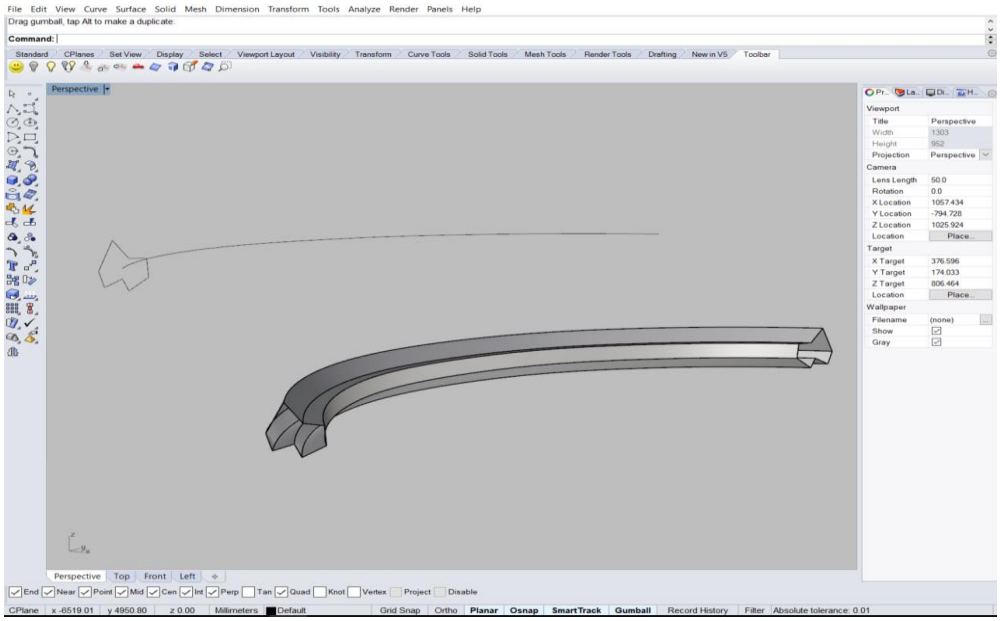
Kreiranje ravninske površine od ravninskih krivulja koje definiraju rubove površine.



Slika 9. Generiranje plohe korištenjem ravninskih krivulja [12]

2.3.6. Sweep

Naredba Sweep1 površine kreira povlačenjem presjeka ili krivulje po jednoj određenoj putanji. Također postoji i naredba Sweep2 koja radi istu stvar, ali površina nastaje provlačenjem po dvije putanje.



Slika 10. Generiranje plohe korištenjem Sweep-a [12]

2.4. Programiranje u Rhino aplikaciji

Rhino aplikacije ima mogućnost korištenja raznih programskih jezika, kao što su Python, Visual Basic for Applications, Grasshopper što ga čini pristupačnim svim korisnicima. Rhino također sadrži vlastiti alat za programiranje „RhinoScript“. RhinoScript je alat za programiranje (pisanje skripti, skriptiranje) temeljen na Microsoft-ovom Visual Basic for Application programskom jeziku. Pomoću RhinoScript-e vrlo lako možemo zamijeniti ručno crtanje automatskim generiranjem krivulja, ploha i tijela. Gotovo sve funkcije koje Rhino posjeduje, dostupne su u Rhino Script-i [11].

Python je interpretativni programski jezik opće namjene. Njegov jezični izražaj i objektno orijentirani pristup nastoje pomoći programerima u pisanju jasnog i logičnog koda za male i velike projekte. Podržava više tipova programiranja uključujući proceduralno, objektno orijentirano i funkcionalno programiranje [13]. Rhino posjeduje i mogućunost korištenja

Python script-a. Ispod je navedena uputa kako pokrenuti PythonScript u Rhino-u [14]:

- Pokrenuti Rhino
- U zapovjednu liniju (eng.Command propt) upisati „EditPythonScript“ i stisnuti Enter
- Otvara se prozor za uređivanje skripte
- U prozor za kod upisati željeni kod
- Kliknuti „Run the Script“
- Prozor za uređivanje skripte nestaje i ispod se pojavljuje prozor s porukama

PythonScript obično može sadržavati puno funkcija koje se mogu uvesti kao „knjižnica“ funkcija (eng. *Library of functions*) u druge skripte ili PythonScript-a može biti naredba koja se izvodi u Rhino-u. Jedna od ključnih karakteristika RhinoScript-a koja olakšava pisanje kompliciranijih skripti je velika „knjižnica“ specifičnih Rhino funkcija koje mogu biti pozvane iz skripte. Naša implementacija Python-a uključuje niz sličnih funkcija koje se mogu uvesti i koristiti u bilo kojoj PythonScript-i za Rhino. Ovaj set funkcija poznatiji je kao „Rhinoscriptsyntax package“ [15].

Here's RhinoScript:

```
Dim arrStart, arrEnd
arrStart = Rhino.GetPoint("Start of line")
If IsArray(arrStart) Then
  arrEnd = Rhino.GetPoint("End of line")
  If IsArray(arrEnd) Then
    Rhino.AddLine arrStart, arrEnd
  End If
End If
```

compared with Python:

```
import rhinoscriptsyntax as rs

start = rs.GetPoint("Start of line")
if start:
  end = rs.GetPoint("End of line")
  if end: rs.AddLine(start,end)
```

Slika 11. Usporedba Rhino i Python skripte [15]

2.5. OpenNURBS definiranje ploha i krivulja [16]

OpenNURBS inicijativa pruža CAD, CAM, CAE i programerima kompjuterskih grafičkih softvera, alate za precizan prijenos 3D geometrije među aplikacijama. OpenNURBS je „knjižnica“ otvorenog koda koja omogućava:

- Kreiranje, serijalizaciju, deserijalizaciju mnogih tipova geometrije, uključujući NURBS površine, BREP 3D tijela, mreže, oblake točaka i SubD objekte
- Učitavanje i ispisivanje 3DM formata datoteke bez potrebe za Rhinom
- Spremanje korisničkih podataka zajedno s geometrijom
- Rad na gotovo svim modernim preglednicima (desktop, cloud, mobitel)

Alati koje pruža openNURBS uključuju:

- Rhino 3DM set sučelja za pristup openNURBS-u s:
  - CPython - dostupan korištenjem standardnog PIP instalera
  - Javascript - „knjižnice“ dostupne na internetu
  - C# - dostupan kao kompletan NuGet package
- Tehničku podršku

2.5.1. NURBS [17]

NURBS ili neuniformni racionalni B-spline je matematički model koji se obično koristi u računalnoj grafici za generiranje i predstavljanje krivulja i površina. Nudi veliku fleksibilnost i preciznost u rukovanju kako analitičkih (površine definirane uobičajenim matematičkim formulama) tako i modeliranih oblika. NURBS se obično koristi u računalnom dizajnu (CAD), proizvodnji (CAM), inženjerstvu (CAE) i mnogim drugim industrijama. NURBS alati nalaze se u različitim softverskim paketima za 3D modeliranje i animaciju. Računalnim programima NURBS-om se može efikasno rukovati, a uz to omogućuju i jednostavnu ljudsku interakciju.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6
Control points	(0 , 0)	(0.45 , 1)	(1.45 , 2.05)	(3 , 2.05)	(4.15 , −0.3)	(5.4 , 0.44)	(6 , 1.5)
Weights	1	0.4	1.5	1	1	1	1

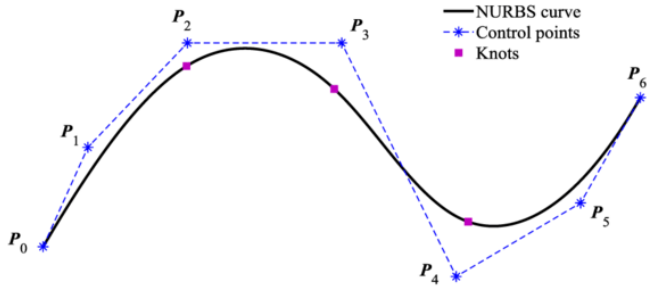
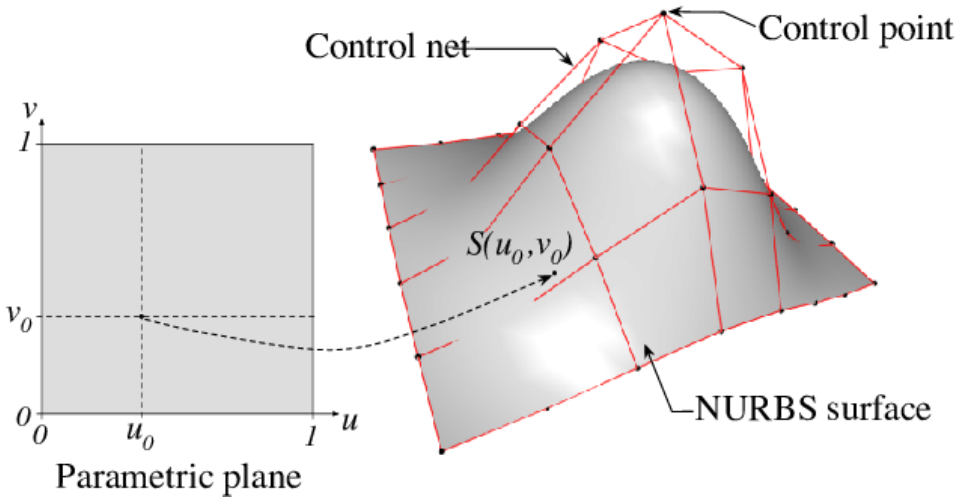


Fig. 1. A cubic NURBS curve with seven control points

Slika 12. Prikaz NURBS krivulje s pripadajućim kontrolnim točkama i čvorovima [18]

NURBS plohe su funkcije dva parametra koji se preslikavaju na površinu u trodimenzionalnom prostoru. Oblik plohe određuje se kontrolnim točkama. NURBS plohe u kompaktnom obliku mogu predstavljati jednostavne geometrijske oblike. NURBS krivulje i plohe korisne su nam iz mnogih razloga:

- nude jedan uobičajeni matematički oblik i za standardne analitičke oblike (npr. Stožac) i za slobodne oblike
- pružaju fleksibilnost u konstruiranju raznih oblika
- smanjuju upotrebu memorije pri pohranjivanju oblika
- mogu se razumno brzo procijeniti pomoću numerički stabilnih i točnih algoritama .

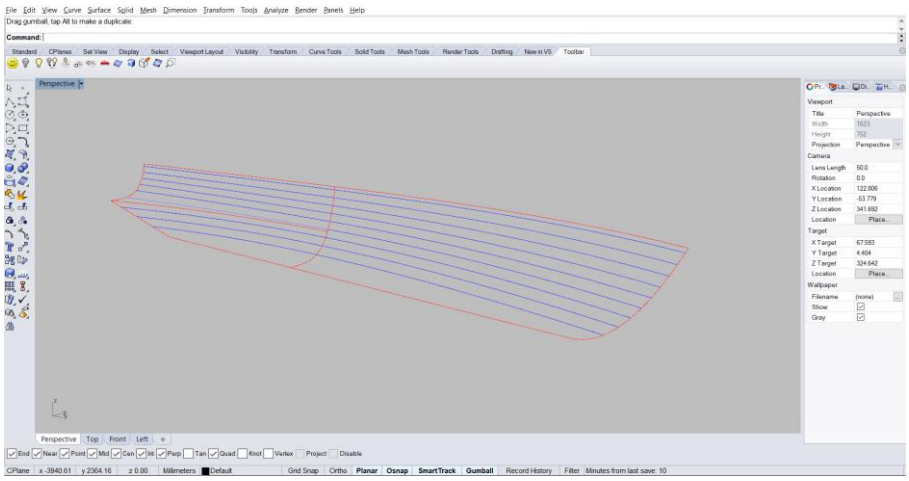


Slika 13. Prikaz NURBS plohe s pripadajućim kontrolnim točkama i kontrolnom mrežom [19]

3. IZRADA NURBS MODELA FORME

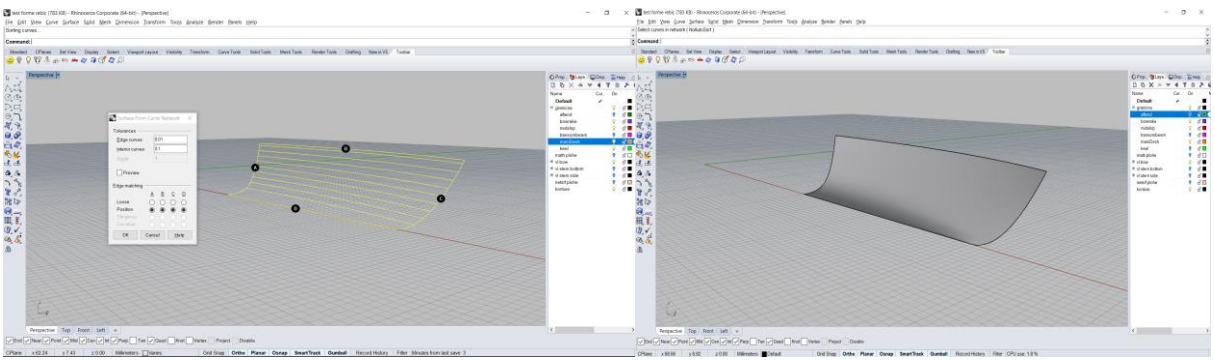
3.1. Generiranje plohe putem mreže krivulja

Plohe brodske forme mogu se generirati pomoću dvaju sustava krivulja koje se međusobno sijeku približno okomito. Jedan sustav krivulja će se upotrijebiti kao vodilice (ravnalice ili direktrise) po kojima će klizati druge krivulje (izvodnice ili genertrise). Svaka točka na plohi tako je određena položajem na izvodnici, a ta izvodnica je određena položajem na ravnalicama. U okviru ovog završnog zadatka, plohe brodske forme generiraju se na taj način. Pramčana statva, glavno rebro, razma, kobilica, skošenje krmene statve, krivulja na spoju dvije krmene plohe i zrcalo predstavljaju ravnalice, dok vodne linije predstavljaju izvodnice.

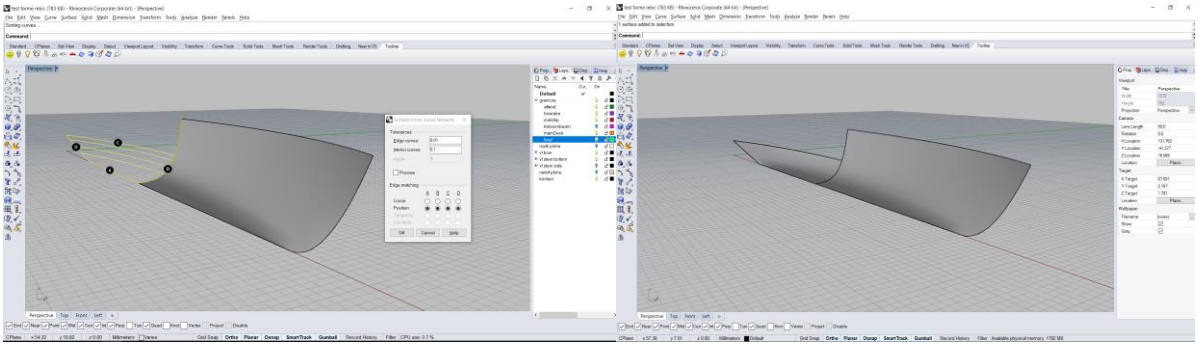


Slika 14. Prikaz ravnalica i izvodnica brodske forme [12]

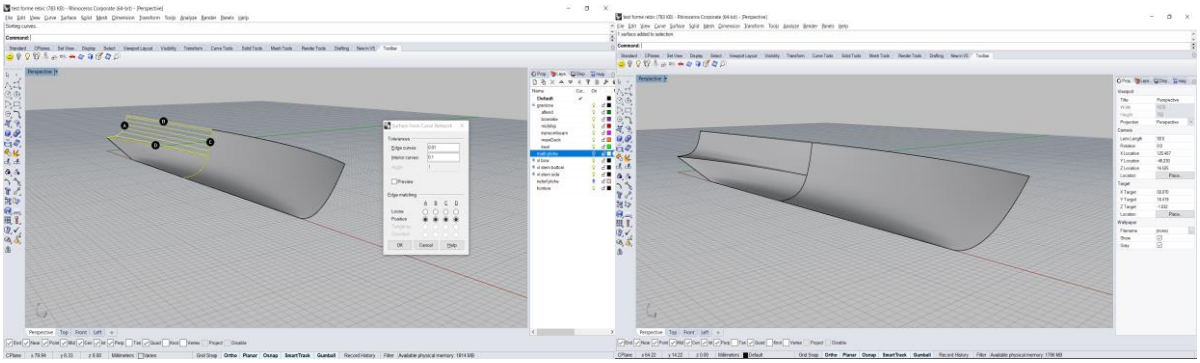
Plohe u Rhino-u dobivene su pomoću naredbe „NetworksSurface“, o kojoj je detaljnije objašnjeno u poglavlju 2.3.4.



Slika 15. Modeliranje pramčane plohe naredbom „NetworkSrf“ [12]



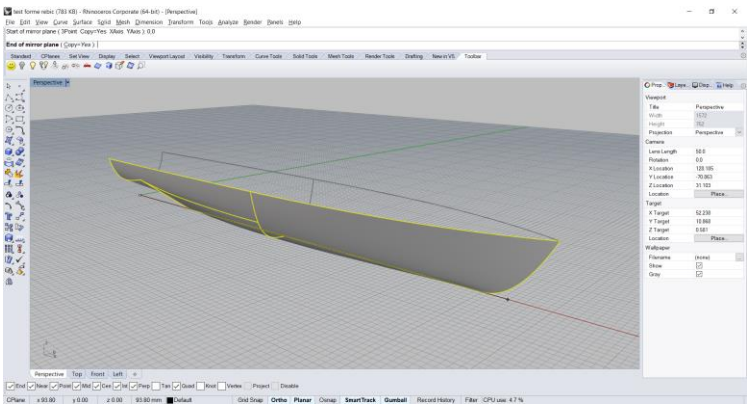
Slika 16. Modeliranje krmene plohe naredbom „NetworkSrf“ [12]



Slika 17. Modeliranje bočne plohe naredbom „NetworkSrf“ [12]

3.2. Odabir ploha

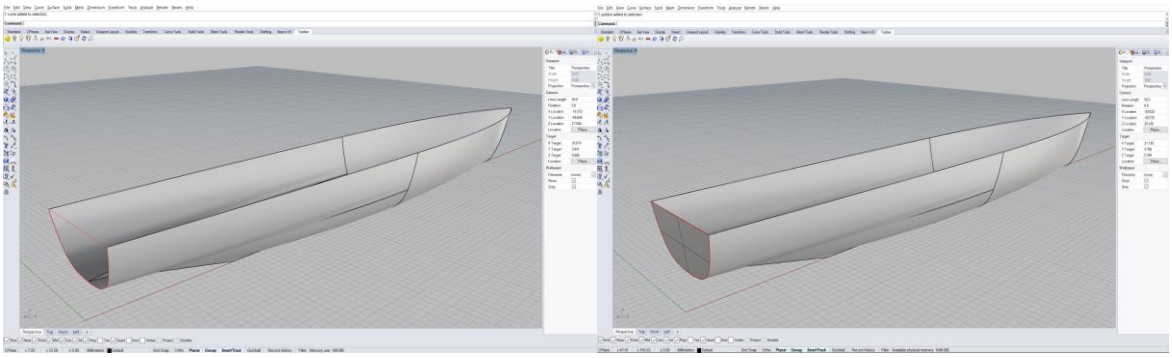
U prethodnom poglavlju 3.1, zbog jednostavnosti je izabrano modeliranje jedne polovice broda. Drugu polovicu broda lako je dobiti naredbom „Mirror“ (Slika 18.) oko uzdužne osi, tj. osi-x. Na pramcu je modelirana samo jedna ploha, jer sve izvodnice tj. vodne linije imaju istu matematičku funkciju, dok su na krmi modelirane dvije plohe, zbog nagle promijene brodske forme na početku zrcala i različitih matematičkih funkcija vodnih linija.



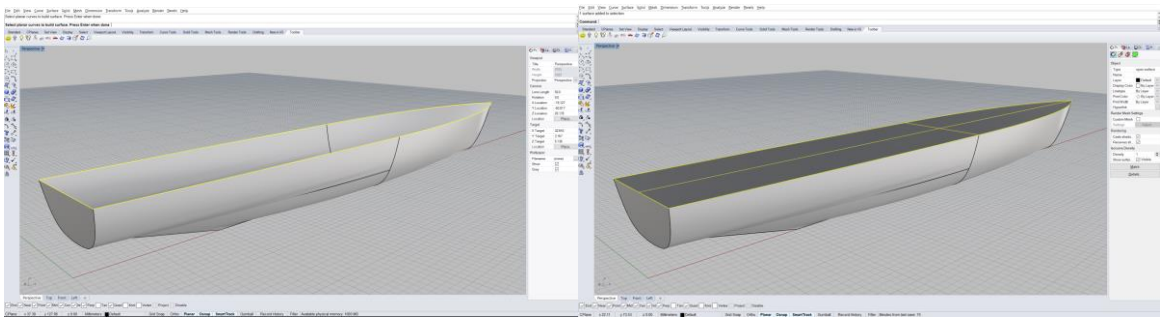
Slika 18. Kreiranje druge polovice broda naredbom „Mirror“ [12]

3.3. Zatvaranje broda

Na postojeći model broda, iz poglavlja 3.2 potrebno je dodati zrcalo i palubu. Obje plohe generiraju se naredbom „PlanarSurface“, o kojoj je detaljnije objašnjeno u poglavlju 2.3.5.

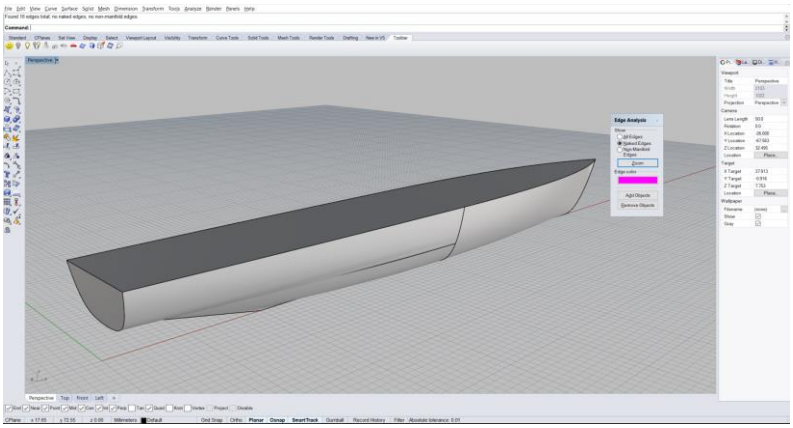


Slika 19. Kreiranje krmenog zrcala naredbom „PlanarSrf“ [12]



Slika 20. Kreiranje palube naredbom „PlanarSrf“ [12]

Sada kada je dobiven čitavi brod koji se sastoji od 6 ploha, zrcala i palube, sve plohe potrebno je povezati naredbom „Join“ te provjeriti jesu li sve plohe dobro spojene.



Slika 21. Provjeravanje rubova ploha naredbom „ShowEdges“ [12]

4. IZRADA PYTHON MODULA

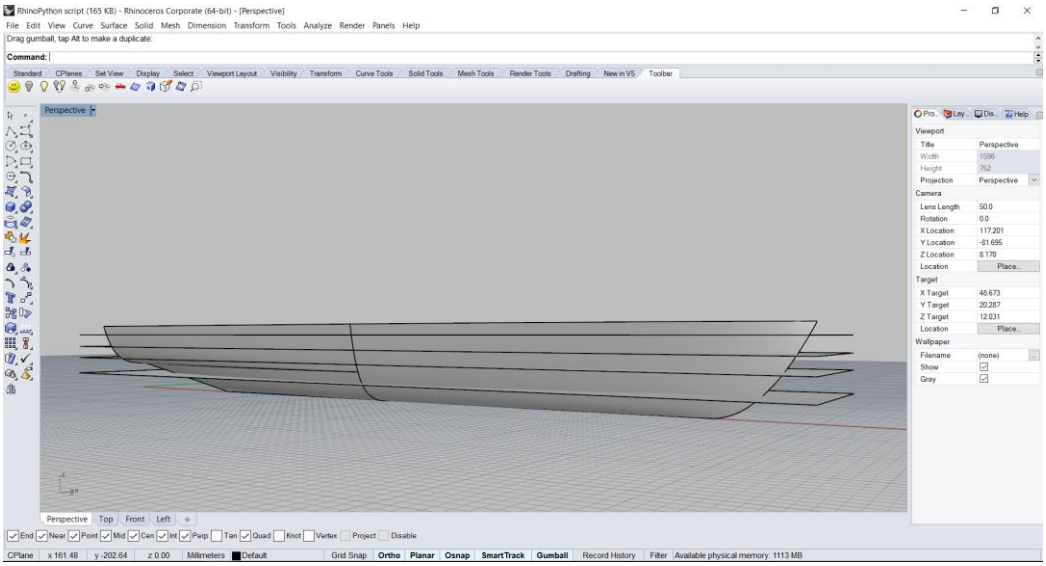
4.1. Odabir hidrostatskih karakteristika

Prilikom odabira hidrostatskih karakteristika treba se bazirati na tome da su dobivene sve potrebne karakteristike za konstrukciju dijagramnog lista. Jedan dio hidrostatskih karakteristika dobiven je korištenjem raznih alata aplikacije Rhino, a drugi dio dobiven je korištenjem iskustvenih formula.

4.2. Rezanje brodske forme

Za postojeći trup dobiven generiranjem analitičkih vodnih linija, u poglavlju 2.3.4, potrebno je izračunati sve hidrostatske karakteristike potrebne za konstrukciju dijagramnog lista. Pošto je ulazni podatak za dijagramni list, gaz broda, trup je potrebno rezati vodnim linijama na željenim visinama. Za svaku vodnu liniju moguće je izračunati:

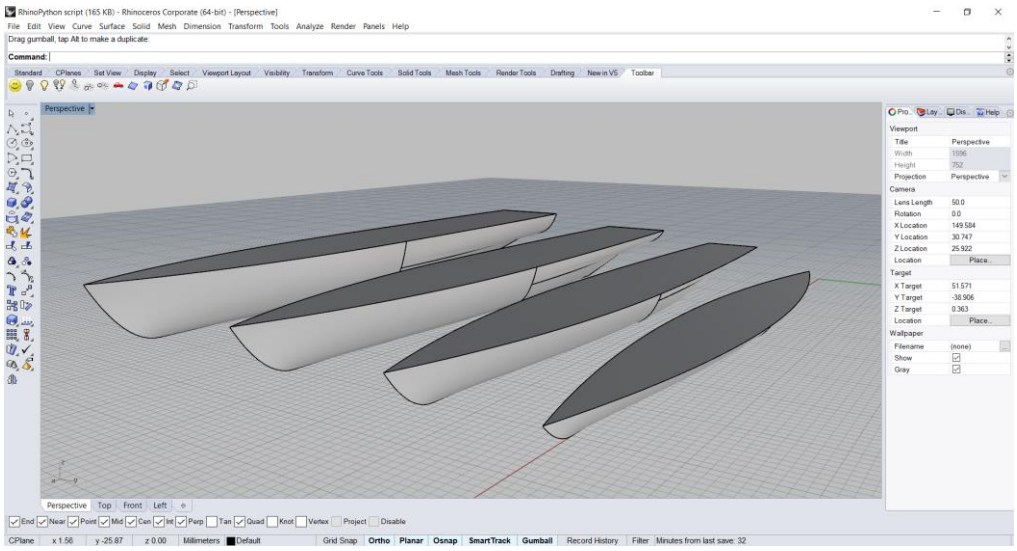
- površinu (A_{WL})
- težište (X_{WL})
- moment inercije oko poprečne osi težišta (I_L)
- moment inercije oko uzdužne osi težišta (I_B)



Slika 22. Prikaz rezanja forme trupa vodnim linijama [12]

Spajanjem izrezanog trupa sa pripadajućom vodnom linijom dobiva se niz zatvorenih volumena za koje je moguće izračunati:

- volumen (V)
- uzdužni položaj težišta volumena ($X_{CB)}$
- vertikalni položaj težišta volumena (KB)



Slika 23. Volumeni dobiveni rezanjem brodske forme vodnim linijama [12]

Na ovaj način, pomoću alata aplikacije Rhino, za željenu vodnu liniju izračunat je jedan dio hidrostatskih karakteristika potrebnih za dijagramni list. Drugi dio hidrostatskih karakteristika računa se prema iskustvenim formulama korištenjem prethodno dobivenih podataka.

$$M_0B = \frac{I_B}{V} \tag{1}$$

$$KM_0 = M_0B + KB \tag{2}$$

$$M_LB = \frac{I_L}{V} \tag{3}$$

$$KM_L = M_LB + KB \tag{4}$$

$$JZ = 0.01 \cdot A_{WL} \cdot \rho \tag{5}$$

$$\Delta = V \cdot \rho \tag{6}$$

$$M_1 = \frac{I_L}{L_{WL}} \tag{7}$$

$$C_{WL} = \frac{A_{WL}}{L_{WL} \cdot B_{WL}} \tag{8}$$

$$C_B = \frac{V}{L_{WL} \cdot B_{WL} \cdot T} \tag{9}$$

$$C_P = \frac{V}{A_X \cdot L_{WL}} \tag{10}$$

$$C_X = \frac{A_X}{B_{WL} \cdot T} \tag{11}$$

4.3. Crtanje dijagramnog lista

Osnovni problem prilikom crtanja dijagramog lista je taj da nisu sve karakteristike istog reda veličine, nego variraju od reda deset na prvu do deset na petu. Zato je za svaku krivulju potrebno odrediti odgovarajuće mjerilo. Postupak određivanja mjerila:

1) odrediti u kojem rasponu se kreće pojedina karakteristika
2) odrediti maksimalnu vrijednost svake karakteristike, iznimka su KM_L i KM_0
3) odrediti na kojem formatu papira se crta dijagram
4) odrediti položaj maksimalne vrijednosti pojedine karakteristike na papiru, za KM_L i KM_0 se određuje minimalna vrijednost
5) mjerilo je jednako kvocijentu maksimalne (minimalne kod KM_L i KM_0) vrijednosti svake karakteristike i položaju na papiru
6) koordinate uzdužnih težišta volumena i vodnih linija te jedinični moment trima crtaju se od glavnog rebra
7) koordinate koeficijenata brodske forme crtaju se od krmene okomice
8) ostale hidrostatske karakteristike ucrtavaju se od kremene okomice

Lijeva koordinatna os predstavlja krmenu okomicu, desna pramčanu okomicu, a srednja os predstavlja položaj glavnog rebra. Vrijednosti se očitavaju mjerenjem horizontalne udaljenosti od sjecišta gaza s krivuljom željene karakteristike do odgovarajuće osi, te množenjem tog očitanja sa pripadajućim mjerilom.

4.4. Dijagramni list [20]

4.4.1. Dijagramni list i njegova upotreba

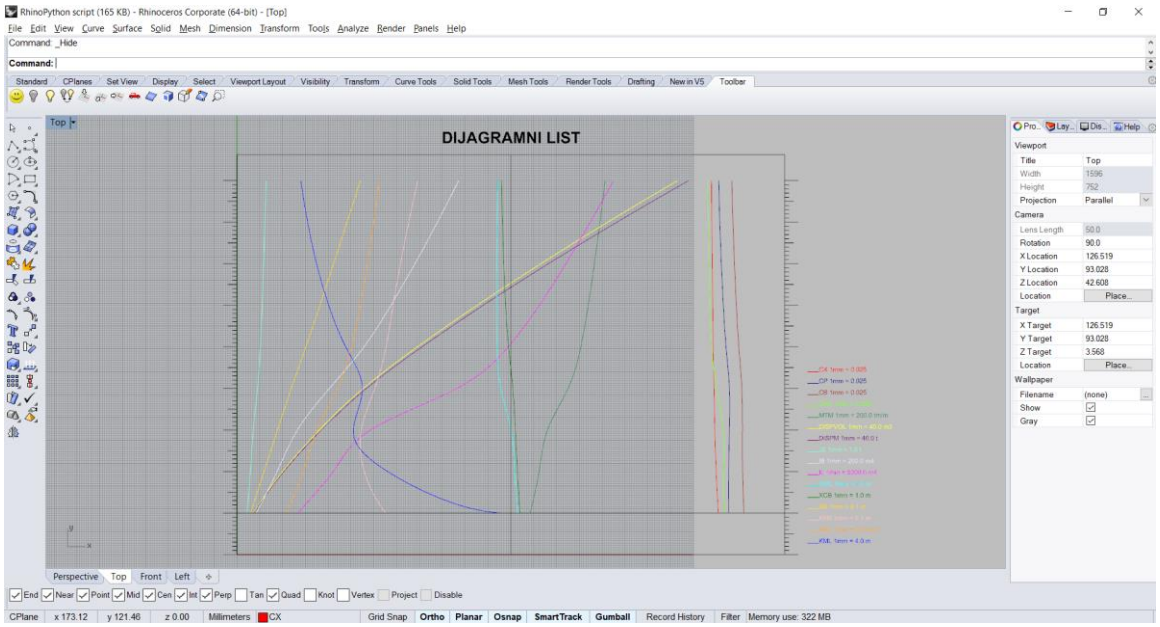
Rezultati proračuna plovnosti broda ucrtavaju se u formi krivulja u jedan dijagram koji se naziva „Dijagramni list“. Kako brod za vrijeme svoje eksploatacije mijenja gaz, već prema količini ukrcanog tereta i potrošku zaliha, potrebno je sve veličine, koje su karakteristične za plovnost broda, dakle istisnina i koordinate njegovog težišta, te položaje poprečnih i uzdužnih metacentara, unijeti u ovisnosti o gazu broda. Osim toga u dijagramni list se obično unose i površine, položaji težišta i momenti tromosti površina vodnih linija, te površine i statički momenti površina rebara, budući da se pomoću njih računaju gore navedene veličine karakteristične za plovnost broda.

Dijagramni list služi projektantu kao osnova za daljnje proračune, na primjer, za određivanje volumena skladišta, za proračun plovnosti kod prodora vode, za određivanje razmaka nepropusnih pregrada, za proračun porinuća itd.

Dijagramni list služi međutim i pomorcima za određivanje gazova, odnosno trima broda kod ukrcaja ili iskrcaja tereta, pa se u tu svrhu u njega ucrtavaju i krivulje koje daju ovisnost jediničnog momenta trima, ukupnog trima za moment trima od 100 tm, te tona po centimetru zagažaja, u ovisnosti od gaza broda.

Osim spomenutih krivulja dijagramni list obično sadrži i koeficijente punoće brodske forme, nanesene u ovisnosti od gaza i razne skale iz kojih se može očitati gaz i nosivost broda.

Dijagramni list ne sadrži položaje težišta sistema budući, da oni ovise o razmještaju težina na brodu i ne mogu se u pojedinostima predvidjeti. Dijagramni list sadrži samo veličine koje karakteriziraju brodsku formu, a ne daje nikakve podatke o položajima pojedinih težina, ni o položaju težišta sistema. Za svaki određeni slučaj opterećenja broda, položaj težišta sistema treba posebno proračunati te ucrtati taj položaj u dijagramni list. Tek onda se može iz dijagramnog lista, pomoću krivulja poprečnih odnosno uzdužnih metacentara, odrediti poprečna, odnosno uzdužna početna metacentarska visina za određeni gaz.



Slika 24. Crtanje dijagramnog lista u aplikaciji Rhino [12]

4.4.2. Skale u dijagramnom listu

Sve skale u dijagramnom listu crtaju se obično u metričkim i engleskim mjernim jedinicama, te za specifičnu težinu mora i slatke vode (budući da mnoge važne luke leže na ušćima velikih rijeka).

Osnovna skala u dijagramnom listu je skala za gaz broda.

Da se za kose vodne linije može očitati gaz na pramcu i na krmi, skale za gaz postavljaju se i na pramcu i na krmi broda. Uz skalu za gaz redovno se još ucrtava i skala za nadvođe broda. Nadvođe broda dobiva se ako se na visinu broda doda debljina opločenja i pokrova palube, pa se od tako dobivene vrijednosti oduzima konstrukcijski gaz broda na dotičnoj plovnoj liniji.

Radi upotrebe dijagramnog lista u eksploataciji broda u njega se unose stvarni, a ne konstrukcijski gazovi. Zbog toga se linija nultog gaza nalazi za visinu kobilice ispod osnovke VL_0 (koja prolazi kroz gornji brid kobilice).

Radi lakše upotrebe dijagramnog lista u eksploataciji broda, u njega se još ucrtavaju i skale paralelnog zagažaja u t/cm, te skale jediničnog momenta trima. One se konstruiraju iz odgovarajućih krivulja zagažaja i jediničnog momenta trima u dijagramnom listu.

4.4.3. Očitavanje vrijednosti u dijagramnom listu **[21]**

Od krmene okomice očitava se:

- volumen na rebrima
- volumen s privjescima
- istisnina s privjescima u morskoj vodi
- istisnina broda
- visina uzdužnog metacentra iznad osnovice
- visina poprečnog metacentra iznad osnovice
- poprečni moment inercije vodne linije
- uzdužni moment inercije vodne linije
- površina vodne linije
- jedinični zagažaj
- oplakana površina (rijetko).

Od glavnog rebra očitava se:

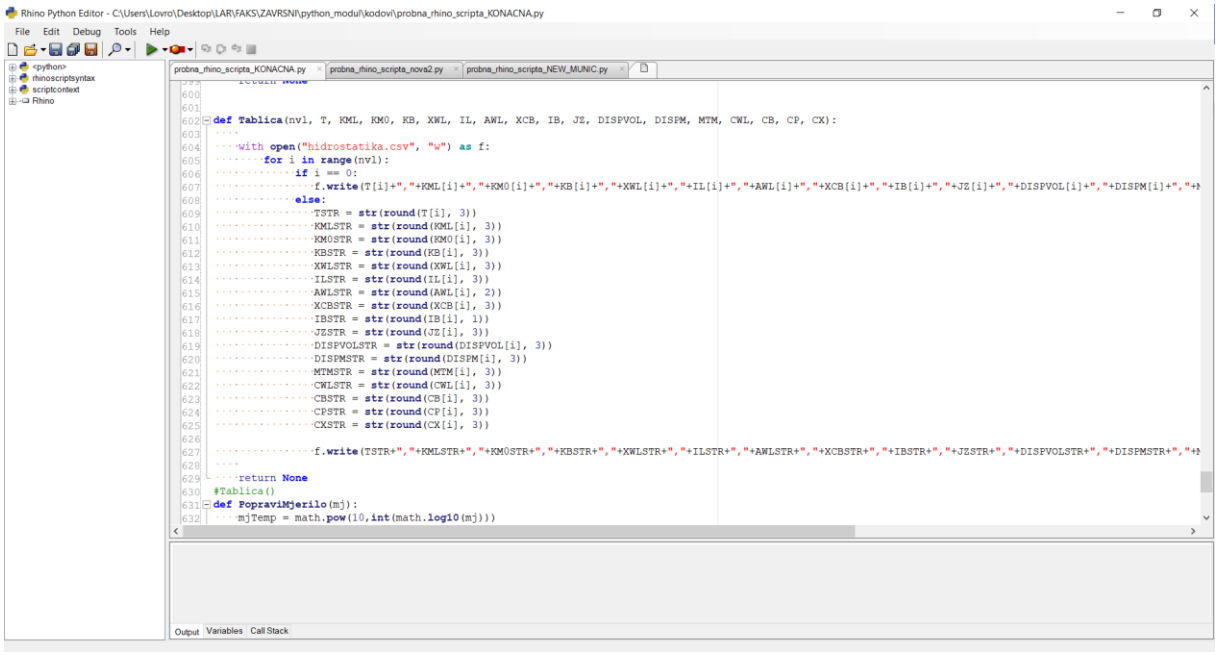
- položaj težišta istisnine po duljini
- položaj težišta vodne linije po duljini
- jedinični moment trima.

Od pramčane okomice očitava se:

- koeficijent punoće
- uzdužni prizmatički koeficijent
- koeficijent najvećeg rebra
- koeficijent vodne linije.

4.5. Ispis hidrostatskih karakteristika

Ispis hidrostatskih karakteristika moguće je izvršiti na nekoliko načina, npr. crtanjem dijagramnog lista ili tabličnim ispisom. Pomoću Python programskog sučelja aplikacije Rhino isprogramiran je kod koji stvara novu .csv datoteku i u nju ispisuje sve hidrostatske karakteristike potrebne za dijagramni list.



Slika 25. Kod za ispis hidrostatskih karakteristika u novu .csv datoteku. [12]

5. USPOREDBA HIDROSTATSKIH KARAKTERISTIKA S ORCA3D APLIKACIJOM

5.1. Općenito o programu Orca3D [22]

Orca3D je programski dodatak aplikaciji Rhino koji nam služi za računanje raznih hidrostatskih i hidrodinamičkih karakteristika broda. Orca3D pojednostavljuje proračun hidromehaničkih karakteristika korištenjem inovativnih alata koji se izvode u Rhino3D okruženju. Proces oblikovanja trupa i analiza hidrostatike moraju biti usko povezani. Trup je konstruiran koristeći jednu ili više NURBS ploha, a iste te plohe koriste se za izračun hidrostatike i stabiliteta. Hidrostatika i NURBS plohe usko su povezani kako bi prilikom modificiranja brodske forme hidrostatske karakteristike mogle biti ažurirane u realnom vremenu. Proračun Orca3D uključuje:

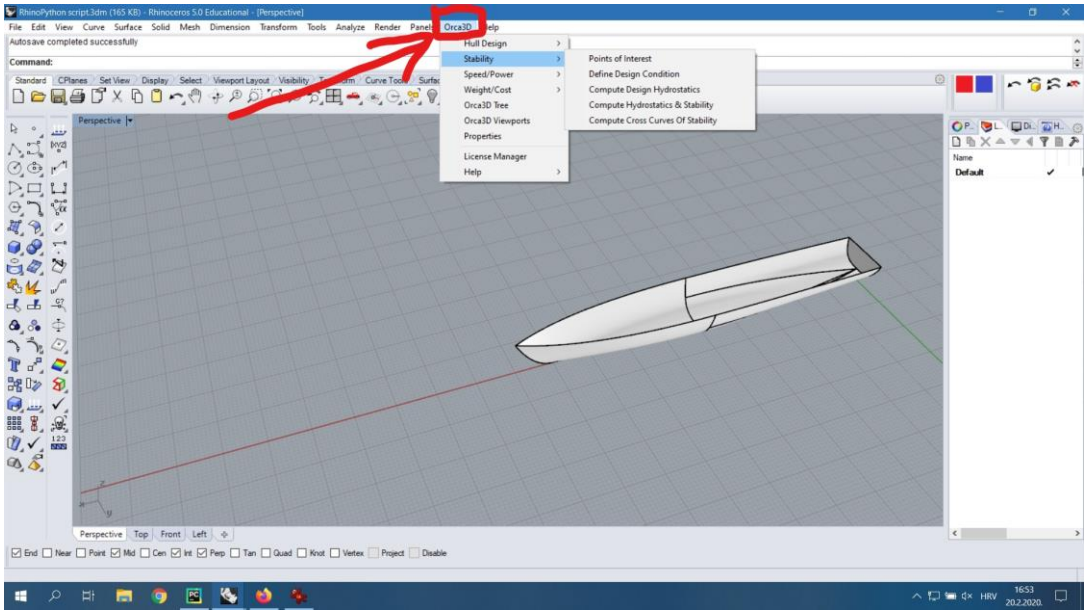
- glavne dimenzije broda i dimenzije vodne linije
- integrirane vrijednosti volumena, istisnine, težišta istisnine, oplakane površine
- površinu i težište vodne linije
- maksimalnu površinu poprečnog presjeka
- koeficijente brodske forme (C_B, C_P, C_X, C_{WL})
- momente inercija i metacentarske visine
- krivulju poluge stabiliteta
- kut trima u odnosu na kut nagiba broda.

Budući da Orca3D hidrostatska svojstva izražava na temelju površinskog modela, gotovo pa nema ograničenja za vrstu objekta koji može analizirati.

Orca3D izrađuje izvještaj koji sadrži tablične ispise za svaku vodnu liniji, kao i crteže odgovarajućih parametara. Izvješće se kreira i prikazuje pomoću Microsoft-ovog generatora izvještaja, datoteka se tada može ispisati ili spremiti u Adobe Acrobat® (pdf) ili

Microsoft Excel® formatu.

Budući da je Orca 3D programski dodatak aplikaciji Rhino, vrlo lako se pokreće iz same aplikacije.

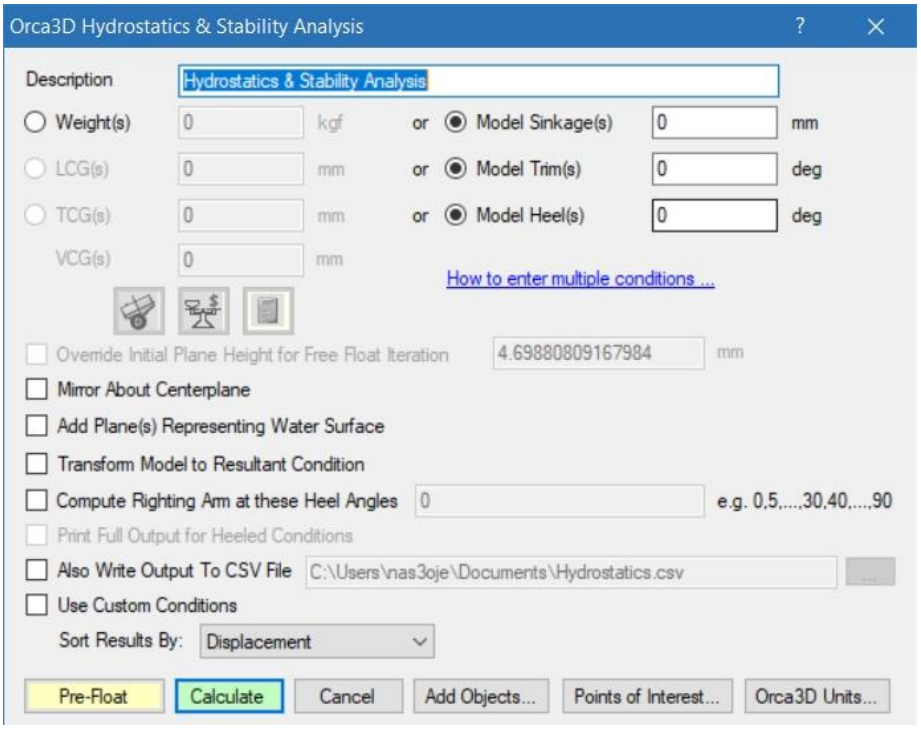


Slika 26. Pokretanje Orca3D u aplikaciji Rhino [12]

5.2. Ulazni podaci [12]

Na Slici 27. vidljivo je sučelje aplikacije Orca3D. Ulazni podaci su:

- težina broda
- raspon gazova
- trim broda
- bočni nagib
- zrcaljenje polovice broda (mogućnost izračuna s pola broda)
- dodavanje vodne linije
- dodavanje nagnutog broda
- računanje poluga stabiliteta za određeni raspon gazova



Slika 27. Sučelje aplikacije Orca3D

5.3. Usporedba dobivenih rezultata s rezultatima Python modula

Radi jednostavnosti, usporedba je izvršena za reducirani broj vodnih linija, u ovom slučaju za 9 vodnih linija.

Tablica 1. Prikaz hidrostatskih karakteristika izračunom pomoću Orca3D programa [22]

PRORAČUN POMOĆU ORCA3D															
T	KML	KM0	KB	XWL	IL	AWL	XCB	IB	JZ	DISPVOL	DISPM	MTM	CB	CP	CX
m	m	m	m	m	m^4	m^2	m	m^4	t	m^3		tm/m			
1	465.7249	6.539193	0.604652	53.04445	134003.1	434.3168	53.99299	170 9.766	4.451748	288.1042	295.3068	179 1.411	0.4 5273 6	0.6 3542 2	0.7 1249 7
2	264.5651	5.585091	1.193324	51.65973	210993.7	582.9804	52.91499	351 8.355	5.975549	801.1251	821.1532	260 8.736	0.4 7029 4	0.6 3646	0.7 3892 2
3	204.3738	5.448541	1.782114	50.21388	292646.6	700.9551	52.02727	529 6.207	7.184 79	144 4.514	148 0.627	337 7.549	0.4 7218 3	0.6 2627 7	0.7 5395 2
4	220.2266	6.045929	2.384182	45.60388	481769	884.6178	50.92791	809 8.129	9.067333	221 1.548	226 6.836	515 2.894	0.4 6694 6	0.6 1084 5	0.7 6442 6
5	190.9483	6.542804	3.0233	43.99947	594216.7	1005.907	49.01892	111 28.63	10.31055	316 1.989	324 1.039	640 4.951	0.5 0999 3	0.6 6025 5	0.7 7241 8
6	161.1094	6.845631	3.640113	43.79266	662514.3	1081.981	47.73824	134 86.45	11.0903	420 7.261	431 2.442	696 0.85	0.5 284	0.6 7843 1	0.7 7885 6
7	139.6455	7.170278	4.239657	43.86525	720483.6	1144.167	46.91811	155 93.6	11.72771	532 0.921	545 3.944	763 9.897	0.5 5846 5	0.7 1211 5	0.7 8423 5
8	124.0153	7.533024	4.882882	44.05331	773856	1198.84	46.38342	175 57.89	12.28811	649 2.815	665 5.135	814 5.478	0.5 7492 4	0.7 2833 2	0.7 8937 2
9	112.2807	7.929119	5.411652	44.29884	824686.2	1248.58	46.03307	194 26.77	12.79795	771 6.795	790 9.715	847 5.553	0.5 7837 8	0.7 2847 9	0.7 9395 3

Tablica 2. Prikaz hidrostatskih karakteristika izračunom pomoću Python programskog sučelja [22]

PRORAČUN POMOĆU PYTHON PROGRAMSKOG SUČELJA															
T	KML	KM0	KB	XWL	IL	AWL	XCB	IB	JZ	DISPVOL	DISPM	MTM	CB	CP	CX
m	m	m	m	m	m^4	m^2	m	m^4	t	m^3		tm/m			
1	463.421	6.512	0.603	53.041	134099.3	434.54	54.041	1712.2	4.454	289.745	296.989	1710.512	0.445	0.613	0.726
2	264.174	5.586	1.192	51.666	211297.9	583.69	52.939	3530.3	5.983	803.468	823.554	2489.559	0.461	0.614	0.75
3	204.193	5.449	1.781	50.215	292990	701.62	52.043	5310.4	7.192	1447.494	1483.682	3218.46	0.462	0.604	0.764
4	220.017	6.044	2.383	45.604	482140.7	885.14	50.935	8111.2	9.073	2215.374	2270.758	5031.095	0.468	0.604	0.774
5	190.783	6.541	3.022	44.007	594542.4	1006.36	49.025	11142.4	10.315	3166.488	3245.65	6146.306	0.502	0.642	0.782
6	160.989	6.843	3.639	43.799	662818.5	1082.36	47.744	13499	11.094	4212.375	4317.685	6794.191	0.529	0.671	0.788
7	139.567	7.168	4.238	43.873	720847.8	1144.59	46.924	15608.2	11.732	5326.647	5459.813	7331.298	0.55	0.693	0.793
8	123.96	7.531	4.827	44.062	774252.9	1199.26	46.39	17572.9	12.292	6499.065	6661.542	7816.955	0.566	0.71	0.797
9	112.242	7.927	5.41	44.307	825119.4	1249	46.04	19442	12.802	7723.538	7916.626	8273.176	0.579	0.723	0.801

Iz tabličnih ispisa vidljivo je da se vrijednosti u oba proračuna razlikuju minimalno, što nam je dokaz da je Python programski modul dobro napravljen.

6. ZAKLJUČAK

U ovom završnom zadatku generirana je forma broda putem analitički zadanih brodskih linija, jer omogućuje jednostavno modificiranje forme broda putem manjeg broja parametara. Aplikacija Rhino omogućuju jednostavno modeliranje NURBS ploha temeljem zadanog polja točaka. Aplikacija također posjeduje i programsko sučelje (API), pomoću kojeg je iz programskog jezika Python, izrađen modul za računanje hidrostatskih karakteristika i crtanje dijagramnog lista. Na kraju je kao završna provjera, napravljena usporedba izračunatih vrijednosti iz programskog modula s vrijednostima dobivenim putem Orca3D programskog dodatka aplikacije Rhino.

LITERATURA

[1] https://www.bib.irb.hr/982264

[2] https://www.sfsb.unios.hr/~tgaleta/predmeti/rnr/materijal/PostupakModeliranja.pdf

[3] https://www.designtechsys.com/articles/parametric-modelling

[4] a-data-driven-holistic-ship-design.pdf

[5] https://www.google.com/search?q=rhio+su%C4%8Delje&source=lnms&tbm=isch&sa=X&ved=2ahUKEwi_rPvbiuDnAhUMbcAKHW5WBk8Q_AUoAXoECAsQAw&biw=1536&bih=937

[6] https://en.wikipedia.org/wiki/Rhinoceros_3D

[7] https://www.google.com/search?q=linear+interpolation+of+control+points+yields+the+control+polygon&hl=hr&source=lnms&tbm=isch&sa=X&ved=2ahUKEwikpa7Ii-DnAhWqM-wKHYm0DJIQ_AUoAXoECAkQAw&biw=1536&bih=937

[8] https://www.google.com/search?q=extruce+curve+rhino&tbm=isch&ved=2ahUKEwitqMvJi-DnAhUOqLQKHd-bCkcQ2-cCegQIABAA&oq=extruce+curve+rhino&gs_l=img.3...90287.93244..93451...0.0..0.161.1938.10j9.....0....1..gws-wiz-img.....0j0i30j0i10i19j0i19j0i10i30i19.1oacfpdYXQE&ei=E3ZOXu3SHY7Q0gXft6q4BA&bih=937&biw=1536&hl=hr#imgrc=qs8VH0nFqH5faM&imgdii=qBvCPJpRdV-EKM

[9] https://www.google.com/search?q=edge+surface+rhino&tbm=isch&ved=2ahUKEwjcqq33i-DnAhVnXVAKHQaxA4UQ2-cCegQIABAA&oq=edge+surface+rhino&gs_l=img.3...73674.76535..76647...0.0..0.107.1553.16j2.....0....1..gws-wiz-img.....0j0i67j0i30j0i19j0i5i30i19j0i8i30i19j0i8i30.DzL4Z_QxzqA&ei=c3ZOXpylHOe6wQKG4o6oCA&bih=937&biw=1536&hl=hr#imgrc=eLsxoQgkOrJ6jM

[10] https://www.youtube.com/watch?v=3XkdIsleAqY

[11] Rosandić, A.: Završni rad, Fakultet strojarstva i brodogradnje, Zagreb, 2017.

[12] Aplikacija Rhinoceros

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

[14] https://www.google.com/search?ei=ltpHXuKkJJiX8gL57IKgBw&q=rhino+script&oq=rhino+script&gs_l=psy-ab.3..0j0i22i30l9.42819.48969..49045...3.1..0.148.1405.11j3.....0....1..gws-wiz.....0i71j0i67j0i131j0i10.CjK7jXD3f88&ved=0ahUKEwjix5G5vtPnAhWYi1wKHXm2AHQQ4dUDCAs&uact=5

[15] https://developer.rhino3d.com/guides/rhinopython/what-is-rhinopython/#what-is-python

[16] https://www.rhino3d.com/opennurbs

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

[18] https://www.google.com/search?q=cubic+nurbs+line+with+control+points&tbm=isch&ved=2ahUKEwioocKSjeDnAhULprQKHYaKB0UQ2-cCegQIABAA&oq=cubic+nurbs+line+with+control+points&gs_l=img.3...5972.12985..13178...0.0..0.127.2465.20j6.....0....1..gws-wiz-img.WClSPFCxVOs&ei=uHdOXqjlNIvM0gWGlZ6oBA&bih=937&biw=1536&hl=hr#imgrc=5K3yhOFMVeGjhM

[19] https://www.google.com/search?q=nurbs+surface&tbm=isch&ved=2ahUKEwj02IeajeDnAhVnHlAKHUhgD4cQ2-cCegQIABAA&oq=nurbs+surface&gs_l=img.3..0i19l2.59393.61087..61262...0.0..0.135.1289.9j4.....0....1..gws-wiz-img.....0i67j0j0i131j0i30j0i8i30i19.zgjBZfle-uE&ei=yHdOXvTUKee8wALIwL24CA&bih=937&biw=1536&hl=hr#imgrc=_V-zrYqG8aA01M

[20] Uršić J.:Plovnost broda, Sveučilište u Zagrebu, Zagreb, 1991.

[21] https://www.fsb.unizg.hr/geometrija.broda/500/560/gb565.htm

[22] https://orca3d.com/

[23] ITTC-nomenclature_2008

PRILOZI

I. CD-R disc

II. Python programski kod za računanje hidrostatskih karakteristika broda i crtanje dijagramnog lista

PRILOG II.

```
import rhinoscriptsyntax as rs
import math
#import numpy as np
#import matplotlib.pyplot as plt
from System.Drawing import Color

def proracun():
    trup = rs.GetObject("Odaberi", rs.filter.polysurface, True, True)
    if (trup == None): return
    rs.UnselectObjects(trup)
    redraw = rs.EnableRedraw(False)

    T = []
    KML = []
    KM0 = []
    KB = []
    XWL = []
    IL = []
    AWL = []
    XCB = []
    IB = []
    JZ = []
    DISPVOL = []
    DISPM = []
    MTM = []
    CWL = []
    CB = []
    CP = []
    CX = []
    XMS = []

    gabariti = rs.BoundingBox(trup)
    # XMS - polovica maksimalne duljine broda
    XMS = abs(gabariti[0][0] - gabariti[2][0]) / 2
    vl0 = 0
    # 0 je 1./8 tocaka, a 2 je Z koordinata (0-X, 1-Y, 2-Z)
    vlmax = gabariti [4][2]
    # 4 je 5./8 tocaka, a 2 je Z koordinata (0-X, 1-Y, 2-Z)
    korak = 1
    nvl = int(vlmax/korak)+1
    nvlmax = (nvl-1)*korak-vlmax
    if nvlmax > -0.01 : nvl = nvl-1
    for n in range(nvl):
        if n != 0:
            # GAZ
            T.append(vl0+n*korak)
            t1 = (-1, -abs(gabariti[0][1])-1, T[n])
            t2 = (abs(gabariti[1][0])+1, -abs(gabariti[0][1])-1, T[n])
            t3 = (abs(gabariti[1][0])+1, abs(gabariti[0][1])+1, T[n])
            t4 = (-1, abs(gabariti[0][1])+1, T[n])
            plane = rs.AddSrfPt([t1, t2, t3, t4])
```

```
            trupNew = rs.CopyObject(trup)
#"Trim Brep"
            # Trims a surface using an oriented cutter
            trupNew = rs.TrimBrep (trupNew,plane)
            plane = rs.SplitBrep (plane,trup,True)
            rs.DeleteObject(plane[0])


            #Calculate the area of a surface or polysurface object. The
results are based on the current drawing units          "SurfaceArea"
            A = rs.SurfaceArea(plane[1])
            AWL.append (A[0])
            #print "AWL = ", AWL[n]
            ##print "gaz = ", gaz[0]

            # racunanje osnovnih dimenzija vodne linije
            gabaritiWl = rs.BoundingBox(plane[1])
            LWL = abs (gabaritiWl [0][0] - gabaritiWl[2][0])
            BWL = abs (gabaritiWl[0][1])*2
            #print "LWL = ", round(LWL, 2)
            #print "BWL = ", round(BWL, 2)

            # Calculates the area centroid of a surface or polysurface
# "SurfaceAreaCentroid"
            PlaneWlCentroid = rs.SurfaceAreaCentroid(plane[1])
            XWL.append (PlaneWlCentroid [0][0])


            # Calculates area moments of inertia of a surface or
polysurface object                                          #
"SurfaceAreaMoments"
            Inertion = rs.SurfaceAreaMoments(plane[1])
            IL.append (Inertion[10][1])
            IB.append (Inertion [10][0])
            #print "IL = ", IL[n]
            #print "IB = ", IL[n]

            #Calculates volume of a closed surface or polysurface
#  "SurfaceVolume"
            trupNew = rs.JoinSurfaces([trupNew, plane[1]],
delete_input=True)
            VOL = rs.SurfaceVolume(trupNew)
            DISPVOL.append (VOL[0])
            #print "V = ", DISPVOL[n]


            #Calculates volume centroid of a closed surface or polysurface
# "SurfaceVolumeCentroid"
            VolumeCentroid = rs.SurfaceVolumeCentroid(trupNew)
            XCB.append (VolumeCentroid [0][0])
            KB.append (VolumeCentroid [0][2])
            #print "XCB = ", XCB[n]
            #print "KB = ", KB[n]

# Calculate the area of a surface or polysurface object. The results are
based on the current drawing units           "SurfaceArea"
            t1x = (XMS+0.01, -BWL/2, 0)
            t2x = (XMS+0.01, -BWL/2,T[n])
            t3x = (XMS+0.01, BWL/2, T[n])
            t4x = (XMS+0.01, BWL/2, 0)
            planex = rs.AddSrfPt([t1x,t2x ,t3x ,t4x])
```

```
            planepNew = rs.CopyObject(planex)
            planepNew = rs.TrimBrep (planepNew,trupNew)
            planex = rs.SplitBrep (planex,trup,True)
            rs.DeleteObject(planex[1])
            rs.DeleteObject(planex[0])
            A = rs.SurfaceArea(planex[2])
            AX = A[0]
            #print "AX = ", AX
            rs.DeleteObject(plane[1])
            rs.DeleteObject(planex[2])
            rs.DeleteObject(trupNew)

            #ostale karakteristike dijagramnog lista
            RO = 1.025
            M0B = IB[n] / DISPVOL[n]
            #print "M0B = ", M0B
            KM0.append (M0B + KB[n])
            #print "KM0 = ", KM0[n]
            MLB = IL[n] / DISPVOL[n]
            #print "MLB = ", MLB
            KML.append (MLB + KB[n])
            #print "KML = ", KML[n]
            JZ.append (0.01*AWL[n]*RO)
            #print "JZ = ", JZ[n]
            DISPM.append (DISPVOL[n]*RO)
            #print "DISPM = ", DISPM[n]
            MTM.append (IL[n] / LWL)
            #print "MTM = ", MTM[n]
            CWL.append (AWL[n] / (LWL * BWL))
            #print "CWL = ", CWL[n]
            CB.append (DISPVOL[n]/(LWL*BWL*T[n]))
            #print "CB = ", CB[n]
            CP.append (DISPVOL[n]/(AX*LWL))
            #print "CP = ", CP[n]
            CX.append (AX/(BWL*T[n]))
            #print "CX = ", CX[n]

        else:
            T.append ("T")
            KML.append ("KML")
            KM0.append ("KM0")
            KB.append ("KB")
            XWL.append ("XWL")
            IL.append ("IL")
            AWL.append ("AWL")
            XCB.append ("XCB")
            IB.append ("IB")
            JZ.append ("JZ")
            DISPVOL.append ("DISPVOL")
            DISPM.append ("DISPM")
            MTM.append ("MTM")
            CWL.append ("CWL")
            CB.append ("CB")
            CP.append ("CP")
            CX.append ("CX")


    # CRTANJE RAVNALA ZA GAZOVE
    gazovi = int(nvl*korak)+1
    visina = 200
```

```
    mj = gazovi/visina
    for k in range(gazovi-1):
        if k == 0:
            # velika crtica na Y = 0
            # 0
            point0 = [240, k/mj, 0]
            point00 = [246, k/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point0 = [-6, k/mj, 0]
            point00 = [0, k/mj, 0]
            crta0 = rs.AddLine(point0, point00)
        else:
            # velika crtica na Y je razlicito od 0
            # 1, 2, 3
            point0 = [240, k/mj, 0]
            point00 = [246, k/mj, 0]
            crta1 = rs.AddLine(point0, point00)
            # Druga strana
            point0 = [-6, k/mj, 0]
            point00 = [0, k/mj, 0]
            crta0 = rs.AddLine(point0, point00)

            # DL crta
            point0 = [0, T[1]/mj, 0]
            point00 = [240, T[1]/mj, 0]
            crta1 = rs.AddLine(point0, point00)

            # srednja crtica na Y = j - 0.5
            # 0.5, 1.5, 2.5
            point0 = [240, (k - 0.5)/mj, 0]
            point00 = [244, (k - 0.5)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-4, (k - 0.5)/mj, 0]
            point11 = [0, (k - 0.5)/mj, 0]
            crta1 = rs.AddLine(point1, point11)

            # mala crtica na Y = j - 0.1
            # 0.9, 1.9, 2.9
            point0 = [240, (k - 0.1)/mj, 0]
            point00 = [242, (k - 0.1)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.1)/mj, 0]
            point11 = [0, (k - 0.1)/mj, 0]
            crta1 = rs.AddLine(point1, point11)
            # 0.8, 1.8, 2.8
            point0 = [240, (k - 0.2)/mj, 0]
            point00 = [242, (k - 0.2)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.2)/mj, 0]
            point11 = [0, (k - 0.2)/mj, 0]
            crta1 = rs.AddLine(point1, point11)
            # 0.7, 1.7, 2.7
            point0 = [240, (k - 0.3)/mj, 0]
            point00 = [242, (k - 0.3)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.3)/mj, 0]
```

```
            point11 = [0, (k - 0.3)/mj, 0]
            crta1 = rs.AddLine(point1, point11)
            # 0.6, 1.6, 2.6
            point0 = [240, (k - 0.4)/mj, 0]
            point00 = [242, (k - 0.4)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.4)/mj, 0]
            point11 = [0, (k - 0.4)/mj, 0]
            crta1 = rs.AddLine(point1, point11)
            # 0.4, 1.4, 2.4
            point0 = [240, (k - 0.6)/mj, 0]
            point00 = [242, (k - 0.6)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.6)/mj, 0]
            point11 = [0, (k - 0.6)/mj, 0]
            crta1 = rs.AddLine(point1, point11)
            # 0.3, 1.3, 2.3
            point0 = [240, (k - 0.7)/mj, 0]
            point00 = [242, (k - 0.7)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.7)/mj, 0]
            point11 = [0, (k - 0.7)/mj, 0]
            crta1 = rs.AddLine(point1, point11)
            # 0.2, 1.2, 2.2
            point0 = [240, (k - 0.8)/mj, 0]
            point00 = [242, (k - 0.8)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.8)/mj, 0]
            point11 = [0, (k - 0.8)/mj, 0]
            crta1 = rs.AddLine(point1, point11)
            # 0.1, 1.1, 2.1
            point0 = [240, (k - 0.9)/mj, 0]
            point00 = [242, (k - 0.9)/mj, 0]
            crta0 = rs.AddLine(point0, point00)
            # Druga strana
            point1 = [-2, (k - 0.9)/mj, 0]
            point11 = [0, (k - 0.9)/mj, 0]
            crta1 = rs.AddLine(point1, point11)


    # 3 VERTIKALNE LINIJE I NATPIS "DIJAGRAMNI LIST"
    # lijeva koordinatna os
    # l = lijevo       lg = lijevo gore
    pointl = [0, 0, 0]
    pointll = [0, 175, 0]
    crtal = rs.AddLine(pointl, pointll)
    # glavno rebro
    # s = sredina       sg = sredina gore
    points = [120, 0, 0]
    pointsg = [120, 175, 0]
    crtas = rs.AddLine(points, pointsg)
    # desna koordinatna os
    # d = desno       dg = desno gore
    pointd = [240, 0, 0]
    pointdg = [240, 175, 0]
    crtad = rs.AddLine(pointd, pointdg)
```

```
    # Dvije horizontalne linije
    pointA = [0, 0, 0]
    pointB = [240, 0, 0]
    crtaAB = rs.AddLine(pointA, pointB)
    pointC = [0, 175, 0]
    pointD = [240, 175, 0]
    crtaCD = rs.AddLine(pointC, pointD)

    # natpis "Dijagramni list"
    # dl = dijagramni list, 1 = bold style, justification =
left/center/right/bottom/middle/top
    pointdl = [120, 180, 0]
    natpis = rs.AddText("DIJAGRAMNI LIST", pointdl, height=5, font_style=1,
justification=2)

    # Add a new layer to the document
"AddLayer"
    layerDL = rs.AddLayer("Dijagramni_list")

    # Dodjeljivanje layera nekom objektu
    naslovDl = rs.ObjectLayer([natpis,crtad,crtal,crtas,crtaAB,crtaCD],
layerDL)


    # Crtanje krivulja u dijagramnom listu
    i=0
    RazmakLegendi=5
    ime="KML"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Blue,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvKML = []
    max = 0
    mjKML = KML[1] / 120
    mjKML = PopraviMjerilo(mjKML)
    for n in range(nvl):
        if n != 0:
            pointcrvKML.append([KML[n]/mjKML, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvKML)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("KML 1mm = "+str(mjKML)+" m", point0, height=2,
font_style=0, justification=1)

    ime="mjAWL"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Orange,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvAWL = []
    mjAWL = AWL[nvl-1] / 100
    mjAWL = PopraviMjerilo(mjAWL)
    for n in range(nvl):
        if n != 0:
            pointcrvAWL.append([AWL[n]/mjAWL, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvAWL)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
```

```
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("AWL 1mm = "+str(mjAWL)+" m2", point0, height=2,
font_style=0, justification=1)

    ime="KM0"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Pink,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvKM0 = []
    mjKM0 = KM0[1] / 150
    mjKM0 = PopraviMjerilo(mjKM0)
    for n in range(nvl):
        if n != 0:
            pointcrvKM0.append([KM0[n]/mjKM0, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvKM0)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("KM0 1mm = "+str(mjKM0)+" m", point0, height=2,
font_style=0, justification=1)

    ime="KB"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Gold,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvKB = []
    mjKB = KB[n-1] / 80
    mjKB = PopraviMjerilo(mjKB)
    for n in range(nvl):
        if n != 0:
            pointcrvKB.append([KB[n]/mjKB, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvKB)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("KB 1mm = "+str(mjKB)+" m", point0, height=2, font_style=0,
justification=1)

    ime="XCB"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Green,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvXCB = []
    mjXCB = XCB[nvl-1] / 50
    mjXCB = PopraviMjerilo(mjXCB)
    for n in range(nvl):
        if n != 0:
            pointcrvXCB.append([120 + (XCB[n]-XMS)/mjXCB, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvXCB)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("XCB 1mm = "+str(mjXCB)+" m", point0, height=2,
font_style=0, justification=1)
```

```
    ime="XWL"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Cyan,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvXWL = []
    mjXWL = XWL[nvl-1] / 50
    mjXWL = PopraviMjerilo(mjXWL)
    for n in range(nvl):
        if n != 0:
            pointcrvXWL.append([120 + (XWL[n]-XMS)/mjXWL, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvXWL)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("XWL 1mm = "+str(mjXWL)+" m", point0, height=2,
font_style=0, justification=1)

    ime="IL"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Magenta,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvIL = []
    mjIL = IL[nvl-1] / 180
    mjIL = PopraviMjerilo(mjIL)
    for n in range(nvl):
        if n != 0:
            pointcrvIL.append([IL[n]/mjIL, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvIL)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("IL 1mm = "+str(mjIL)+" m4", point0, height=2, font_style=0,
justification=1)

    ime="IB"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Lavender,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvIB = []
    mjIB = IB[nvl-1] / 150
    mjIB = PopraviMjerilo(mjIB)
    for n in range(nvl):
        if n != 0:
            pointcrvIB.append([IB[n]/mjIB, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvIB)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("IB 1mm = "+str(mjIB)+" m4", point0, height=2, font_style=0,
justification=1)

    ime="JZ"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Aquamarine,"Dijagramni list")
```

```
    rs.CurrentLayer(ime)
    pointcrvJZ = []
    mjJZ = JZ[nvl-1] / 100
    mjJZ = PopraviMjerilo(mjJZ)
    for n in range(nvl):
        if n != 0:
           pointcrvJZ.append([JZ[n]/mjJZ, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvJZ)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("JZ 1mm = "+str(mjJZ)+" t", point0, height=2, font_style=0,
justification=1)

    ime="DISPM"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Purple,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvDISPM = []
    mjDISPM = DISPM[nvl-1] / 220
    mjDISPM = PopraviMjerilo(mjDISPM)
    for n in range(nvl):
        if n != 0:
           pointcrvDISPM.append([DISPM[n]/mjDISPM, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvDISPM)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("DISPM 1mm = "+str(mjDISPM)+" t", point0, height=2,
font_style=0, justification=1)

    ime="DISPVOL"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Yellow,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvDISPVOL = []
    mjDISPVOL = mjDISPM
    for n in range(nvl):
        if n != 0:
           pointcrvDISPVOL.append([DISPVOL[n]/mjDISPVOL, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvDISPVOL)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("DISPVOL 1mm = "+str(mjDISPVOL)+" m3", point0, height=2,
font_style=0, justification=1)

    ime="MTM"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.SeaGreen,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvMTM = []
    mjMTM = MTM[nvl-1] / 80
    mjMTM = PopraviMjerilo(mjMTM)
    for n in range(nvl):
```

```
        if n != 0:
            pointcrvMTM.append([120 + MTM[n]/mjMTM, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvMTM)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("MTM 1mm = "+str(mjMTM)+" tm/m", point0, height=2,
font_style=0, justification=1)

    ime="CWL"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Chartreuse,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvCWL = []
    mjC = 0.025
    for n in range(nvl):
        if n != 0:
            pointcrvCWL.append([240 - CWL[n]/mjC, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvCWL)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("CWL 1mm = "+str(mjC)+"", point0, height=2, font_style=0,
justification=1)

    ime="CB"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Brown,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvCB = []
    for n in range(nvl):
        if n != 0:
            pointcrvCB.append([240 - CB[n]/mjC, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvCB)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("CB 1mm = "+str(mjC)+"", point0, height=2, font_style=0,
justification=1)

    ime="CP"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.DarkBlue,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvCP = []
    for n in range(nvl):
        if n != 0:
            pointcrvCP.append([240 - CP[n]/mjC, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvCP)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
```

```
    rs.AddText("CP 1mm = "+str(mjC)+"", point0, height=2, font_style=0,
justification=1)

    ime="CX"
    rs.CurrentLayer("Dijagramni_list")
    rs.AddLayer(ime,Color.Red,"Dijagramni list")
    rs.CurrentLayer(ime)
    pointcrvCX = []
    for n in range(nvl):
        if n != 0:
           pointcrvCX.append([240 - CX[n]/mjC, T[n]/mj, 0])
    rs.AddInterpCurve(pointcrvCX)
    i=i+RazmakLegendi
    point0 = [250, i, 0]
    point00 = [255, i, 0]
    rs.AddLine(point0, point00)
    point0 = [255, i, 0]
    rs.AddText("CX 1mm = "+str(mjC)+"", point0, height=2, font_style=0,
justification=1)


    sakrijtrup = rs.HideObject(trup)


    Tablica (nvl, T, KML, KM0, KB, XWL, IL, AWL, XCB, IB, JZ, DISPVOL,
DISPM, MTM, CWL, CB, CP, CX)
    return None


def Tablica(nvl, T, KML, KM0, KB, XWL, IL, AWL, XCB, IB, JZ, DISPVOL,
DISPM, MTM, CWL, CB, CP, CX):

    with open("hidrostatika.csv", "w") as f:
        for i in range(nvl):
             if i == 0:

f.write(T[i]+","+KML[i]+","+KM0[i]+","+KB[i]+","+XWL[i]+","+IL[i]+","+AWL[i
]+","+XCB[i]+","+IB[i]+","+JZ[i]+","+DISPVOL[i]+","+DISPM[i]+","+MTM[i]+","
+CWL[i]+","+CB[i]+","+CP[i]+","+CX[i]+"\n")
             else:
                TSTR = str(round(T[i], 3))
                KMLSTR = str(round(KML[i], 3))
                KM0STR = str(round(KM0[i], 3))
                KBSTR = str(round(KB[i], 3))
                XWLSTR = str(round(XWL[i], 3))
                ILSTR = str(round(IL[i], 3))
                AWLSTR = str(round(AWL[i], 2))
                XCBSTR = str(round(XCB[i], 3))
                IBSTR = str(round(IB[i], 1))
                JZSTR = str(round(JZ[i], 3))
                DISPVOLSTR = str(round(DISPVOL[i], 3))
                DISPMSTR = str(round(DISPM[i], 3))
                MTMSTR = str(round(MTM[i], 3))
                CWLSTR = str(round(CWL[i], 3))
                CBSTR = str(round(CB[i], 3))
                CPSTR = str(round(CP[i], 3))
                CXSTR = str(round(CX[i], 3))


f.write(TSTR+","+KMLSTR+","+KM0STR+","+KBSTR+","+XWLSTR+","+ILSTR+","+AWLST
```

```
R+","+XCBSTR+","+IBSTR+","+JZSTR+","+DISPVOLSTR+","+DISPMSTR+","+MTMSTR+","
+CWLSTR+","+CBSTR+","+CPSTR+","+CXSTR+"\n")

    return None
#Tablica()
def PopraviMjerilo(mj):
    mjTemp = math.pow(10,int(math.log10(mj)))
    mjTemp2 = int(mj/mjTemp)+1
    mj = mjTemp2*mjTemp
    return mj

proracun()
redraw = rs.EnableRedraw(True)
```