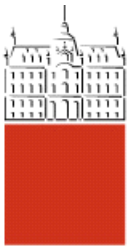
Univerza
v Ljubljani

Fakulteta za
gradbeništvo in
geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si



UNIVERZITETNI ŠTUDIJSKI
PROGRAM GRADBENIŠTVO
KONSTRUKCIJSKA SMER

Kandidat:

ROK SOKLIČ

PARAMETRIČNO MODELIRANJE IN OPTIMIZACIJA KONSTRUKCIJ S PROGRAMOM RHINOCEROS-GRASSHOPPER

Diplomska naloga št.: 3430/KS

PARAMETRIC MODELLING AND OPTIMIZATION OF BUILDING STRUCTURES WITH RHINOCEROS-GRASSHOPPER

Graduation thesis No.: 3430/KS

Mentor:
doc. dr. Tomo Cerovšek

Somentor:
prof. dr. Boštjan Brank

Član komisije:
doc. dr. Tomaž Maher
prof. dr. Boštjan Brank

Predsednik komisije:
doc. dr. Tomo Cerovšek

Ljubljana, 30. 03. 2015

STRAN ZA POPRAVKE, ERRATA

Stran z napako	Vrstica z napako	Namesto	Naj bo

IZJAVE

Podpisani Rok Soklič izjavljam, da sem avtor diplomskega dela z naslovom »Parametrično modeliranje in optimizacija konstrukcij s programom Rhinoceros – Grasshopper«.

Izjavljam, da je elektronska različica v vsem enaka tiskani različici.

Izjavljam, da dovoljujem objavo elektronske različice v digitalnem repozitoriju.

Kranj, 28.2.2015

Rok Soklič

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK:	**004.42:624.04 (043.2)**
Avtor:	**Rok Soklič**
Mentor:	**doc. dr. Tomo Cerovšek**
Somentor:	**prof. dr. Boštjan Brank**
Naslov:	**Parametrično modeliranje in optimizacija konstrukcij s programom Rhinoceros – Grasshopper**
Tip dokumenta:	**diplomska naloga – univerzitetni študij**
Obseg in oprema:	**89 str., 8 pregl., 85 sl., 16 en.**
Ključne besede:	**parametrizacija, NURBS, Rhinoceros, Grasshopper, SAP2000, genetski algoritmi, optimizacija konstrukcij**

Izvleček

Diplomska naloga obravnava uporabo sodobnih računalniških orodij za geometrijsko modeliranje v procesu projektiranja objektov. Z vpeljavo parametričnega opisa geometrije v modelirniku 3D ter s povezavo modelirnika s programom za računanje konstrukcij, se odpirajo možnosti za avtomatično izmenjavo podatkov med programi za design in analizo konstrukcij. To lahko vodi k izboljšavi procesa projektiranja objektov. V prvem delu naloge sem predstavil osnove matematičnega zapisa geometrijskih elementov NURBS (angl. Non-Uniform Rational Basis Spline), ki jih uporabljata računalniški program za geometrijsko oblikovanje Rhinoceros in njegov dodatek Grasshopper. V drugem delu naloge sem predstavil parametrizacijo geometrije enostavnih ploskovnih in prostorskih paličnih konstrukcij v programu Rhinoceros (Grasshopper) ter njegovo povezavo s programom za analizo konstrukcij SAP2000. Znotraj okolja Grasshopper sem uporabil različne dodatke za ovrednotenje geometrije, vključno z genetskim optimizacijskim algoritmom, ki sem ga uporabil za interaktivno iskanje primerne oblike konstrukcije pri predpisanih pogojih. V zadnjem delu naloge je prikazana specifična nadgradnja omenjene povezave z vpeljavo novega vmesnika, ki sem ga izdelal v programskem jeziku Visual Basic.NET za uporabo v okolju Grasshopper.

BIBLIOGRAPHIC – DOCUMENTALISTIC INFORMATION AND ABSTRACT

UDC:	**004.42:624.04 (043.2)**
Author:	**Rok Soklič**
Supervisor:.	**assist. prof. Tomo Cerovšek, Ph. D.**
Co-supervisor:	**prof. Boštjan Brank, Ph. D.**
Title:	**Parametric modelling and structural optimization using the computer software Rhinoceros – Grasshopper**
Document type:	**Graduation thesis – University studies**
Scope and tools:	**89 p., 8 tab., 85 fig., 16 eq.**
Keywords:	**parametrization, NURBS, Rhinoceros, Grasshopper, SAP2000, genetic algorithms, structural optimization**

Abstract

This graduation thesis comprises the use of modern geometry design software in the process of structural design. By implementing parametrically designed geometry and interactively linking 3D computer graphics application software with structural analysis software, tasks involved in structural design (e.g. automatic design data transfer) can be significantly improved. In the first part of the thesis, the basics of NURBS (Non-uniform Rational Basis Spline) mathematical elements are briefly presented, since the here used geometry design software Rhinoceros, and its add-on Grasshopper, are based on the NURBS. In the second part of the thesis examples of parametric geometrical design of simple space frame and shell structures are presented, along with a linking procedure beetwen Rhinoceros and structural analysis software SAP2000. In order to evaluate suitability of structural geometry, several Grasshopper based add-ons were tested out including specific genetic optimization algorithms. The genetic algorithms allow us to find optimal structural geometry due to predefined objective function and constraints. In the final part of the thesis, a new Grasshopper based Rhinoceros-SAP2000 interface/component is introduced which was programmed by using Visual Basic.NET scripting language.

POSVETILO

Diplomsko nalogo posvečam očetu, ki mi je bil predvsem v študijskih letih vzgled.

V dobrem in slabem.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

»Ta stran je namenoma prazna«

1 UVOD

Sodobno gradbeništvo je na področju načrtovanja, analize in dimenzioniranja konstrukcij tesno povezano z uporabo številnih računalniških orodij, ki služijo namenu doseganja učinkovitejšega delovnega procesa. Kljub raznovrstnosti komercialnih programskih orodij je sam proces modeliranja v veliki meri še vedno neodvisen od izbire orodja.

Konvencionalna metoda digitalnega modeliranja obsega grafično (ročno) definiranje prostorskih geometrijskih elementov (teles, ploskev/površin itd.) ter njihovo oblikovanje oz. transformacijo. Vsaka sprememba glavne geometrije osnovnega modela povzroči potrebo po spreminjanju vseh medsebojno povezanih geometrijskih elementov. Običajno to pomeni prirejanje, večanje/zmanjševanje, reorientiranje vsakega posameznega elementa znotraj hierarhično urejene in medsebojno povezane celote.

Drug način predstavlja t. i. generativni princip, kjer končno/optimalno geometrijo izberemo izmed množice parametrično podanih različic geometrije modela. Pri parametričnem modeliranju je končna oblika oz. forma modela določena z vrednostjo različnih vhodnih parametrov, ki določajo geometrijo modela (npr. višina, širina, radij, ukrivljenost itd.) in enačbami, ki opisujejo razmerja med parametri.

Namesto grafičnega podajanja (modeliranja) modela konstrukcije v generativnem modeliranju vhodne podatke predstavljajo podatki, ki določajo geometrijo. Zasnovani model je posledica matematičnih operacij, medsebojnih razmerij in funkcij vhodnih podatkov. Končni modeli vsebujejo veliko število spremenljivk, ki so znotraj urejene strukture (algoritma) lahko uporabljene na poljuben način. Tak način modeliranja dovoljuje velike možnosti spreminjanja v samem procesu zasnove in izdelave modela, kar pri klasičnih metodah ni možno. Kriterij za izbor optimalne rešitve je lahko tehnične ali povsem estetske narave.

V diplomski nalogi sem se posvetil povezovanju različnih programskih orodij za modeliranje 3D s programom za računanje konstrukcij. S tem sem skušal prikazati načine, s katerimi lahko izboljšamo proces računske analize konstrukcij. Prednosti generativnega principa modeliranja namreč lahko uporabimo tudi v procesu analize konstrukcije, v kolikor imamo na razpolago ustrezna orodja (npr. evolucijske algoritme) in kriterije za izbiro optimalne rešitve.

1.1 Cilj in namen naloge

Cilj naloge je predstaviti potek parametrične zasnove in modeliranja konstrukcij z uporabo sodobnih programskih orodij in podati teoretične osnove na katerih temelji delovanje glavnega uporabljenega programskega orodja Rhinoceros [7]. Prikazati sem želel povezovanje ustvarjenih modelov geometrije konstrukcij z drugimi programskimi orodji (SAP2000 [20]) ter optimizacijo konstrukcij z evolucijskimi algoritmi, ki so na voljo v okolju Grasshopper.

1.2 Metode dela

Pri procesu izdelave diplomske naloge sem uporabil naslednje metode:

- analiza in interpretacija primarnih in sekundarnih virov v sklopu spoznavanja teoretičnega ozadja krivulj in ploskev NURBS;
- študija primerov objektov primernih za predstavitev parametričnega modeliranja;
- uporaba svetovnega spleta kot glavni vir informacij v zvezi z uporabo programskih orodij in pomoč pri zasnovi algoritmov.

2 BERNSTEIN-BEZIEREVA PREDSTAVITEV POLJUBNIH KRIVULJ IN PLOSKEV (NURBS)

To poglavje na kratko predstavi teoretično ozadje načrtovanja poljubnih krivulj in ploskev s programom Rhinoceros, ki ga uporabljam skozi celotno nalogo. Orodja programa Rhinoceros temeljijo na t. i. NURBS (angl. Non-Uniform Rational Basis Spline), matematičnih formah za opis poljubnih geometrijskih oblik. Po zaslugi mednarodnih standardov za izmenjavo digitalnih informacij (npr. IGES, STEP itd.) med t. i. CAD (angl. computer aided design) sistemi so NURBS postale močno orodje za modeliranje. Razširjenost NURBS je v veliki meri posledica naslednjega:

- Točen opis različnih geometrijskih oblik, tako analitičnih (npr. kvadrat, krogla, stožec itd.) kot tudi popolnoma organskih (npr. za potrebe oblikovanja ladijskih trupov).
- Računalniški algoritmi, ki vsebujejo reprezentacije NURBS, so numerično stabilni in ne zahtevajo shranjevanja velikih količin informacij.
- Krivulje in ploskve NURBS so enostavne (v smislu računalniške obdelave podatkov) in enostavno podvržene običajnim geometrijskim transformacijam (translacija, rotacija itd.).

Razlaga teoretičnega ozadja povzeta po [1].

2.1 Krivulje NURBS

Parametrično obliko enačbe poljubne ravninske krivulje v kartezičnem koordinatnem sistemu *x, y* lahko zapišemo kot:

$$C(u) = \big(x(u), y(u)\big), \quad a \leq u \leq b \tag{1}$$

Zaradi enostavnejšega zapisa v računalniških algoritmih je bila vpeljana t. i. *Bezier-jeva krivulja*.

Za množico koordinat *n*-kontrolnih točk je Bezier krivulja stopnje *n* definirana kot:

$$C(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \text{ , na območju } 0 \leq u \leq 1 \text{ ,} \tag{2}$$

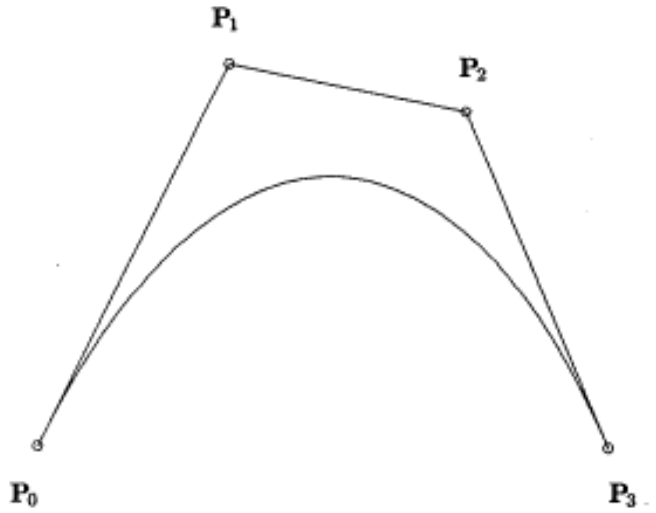
kjer P_i predstavlja lego *i*-te kontrolne točke, $P_i = (x_i, y_i)$, v izbranem ravninskem koordinatnem sistemu, členi $B_{i,n}(u)$ pa v zgornjem zapisu predstavljajo Bernstein polinome stopnje *n*, ki so definirani kot:

$$B_{i,n}(u) = \binom{n}{k} u^i (1-u)^{n-i} \text{ ,} \tag{3}$$

pri čemer $\binom{n}{k}$ predstavlja binomski koeficient, na območju $0 \leq u \leq 1$. Definicija povzeta po [36].

Poljubno Bezier krivuljo (slika 1) stopnje *n* = 3, skladno z enačbo (2), torej lahko zapišemo kot:

$$C(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3 P_3 \tag{4}$$

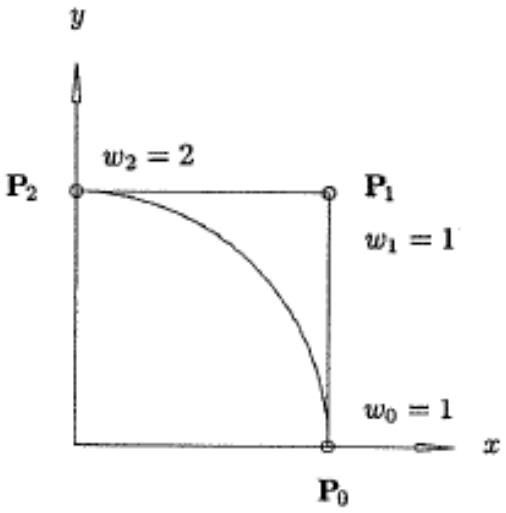


Slika 1: Primer poljubne Bezier krivulje stopnje n=3, podane z enačbo (4). Točke z oznakami P1 do P4 imenujemo kontrolne točke (vir: Les Piegl, Wayne Tiller, 1997, str.13).

Bezier krivulje (v načinu zapisa samo z Bernstein polinomi) pa ne zagotavljajo točnega opisa številnih pomembnih osnovnih oblik krivulj in ploskev (kot npr. krožnica, elipsa, valj itd.). Zaradi tega je bil v definicijo Bezier krivulje vpeljan faktor uteži w_i, ki predstavlja skalar. Racionalna Bezier krivulja je tako podana kot:

$$C(u) = \frac{\sum_{i=0}^{n} B_{i,n}(u) w_i P_i}{\sum_{i=0}^{n} B_{i,n}(u) w_i}, \qquad 0 \leq u \leq 1 \tag{5}$$

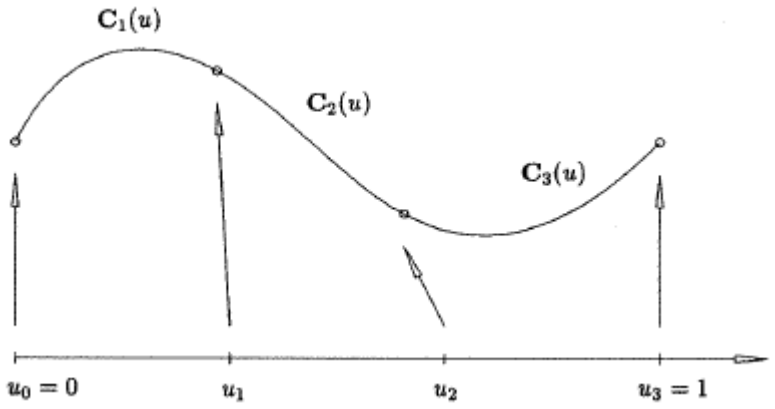
Z vpeljavo uteži lahko sedaj lažje kontroliramo obliko Bezier krivulj (slika 2).



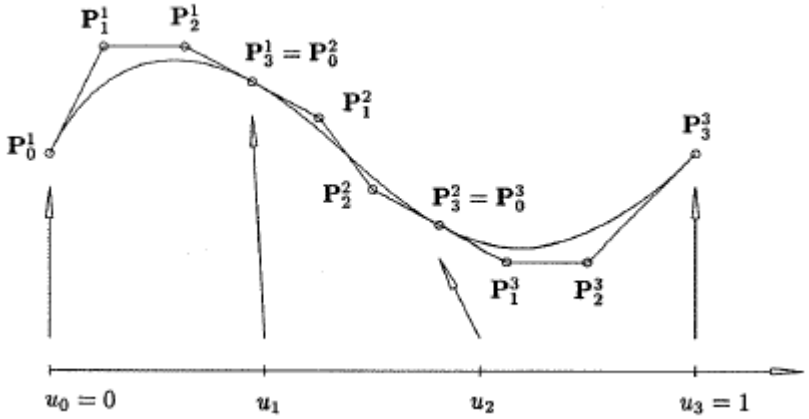
Slika 2: Prikaz odseka enotske krožnice, opisane s pomočjo uteži dodanih Bezier krivuljam (vir: Les Piegl, Wayne Tiller, 1997, str.29).

Poleg težav z opisom številnih osnovnih oblik so Bezier krivulje višjih stopenj kompleksne in numerično nestabilne, kar lahko odpravimo z vpeljavo *odsekovnih/periodičnih polinomskih krivulj* (angl. piecewise polynomial curves). Osnovna krivulja je tako razdeljena na posamezne odseke (*m* predstavlja število odsekov), pri čemer množica $U = \{u_0, \dots, u_m\}$, ki predstavlja nepadajoče zaporedje realnih števil $u_i \leq u_{i+1}$, $i \in [0, \dots, m+1]$, označuje mejne točke posameznih odsekov. Vrednosti u_i predstavljajo t. i. vozle in označujejo vrednosti parametra *u*, pri katerih se posamezni odseki krivulje začnejo in končajo.

U imenujemo vozliščni vektor. Na tem mestu je potrebno povedati, da izraza vektor v tem kontekstu ne smemo jemati v klasičnem smislu (kot količino, ki ima določeni velikost in smer).



Slika 3:Odsekovna kubična polinomska krivulja s tremi odseki. Točke, kjer se končajo in začnejo posamezni odseki, so označene z vrednostjo parametra u (vir: Les Piegl, Wayne Tiller, 1997, str.48).



Slika 4: Krivulja iz slike 3, prikazana z odsekovnimi polinomi v Bezier obliki (vir: Les Piegl, Wayne Tiller, 1997, str.49).

Pri tako sestavljenih krivuljah se na mestih stikovanja posameznih odsekov lahko pojavijo težave z zveznostjo (gladkostjo) celotne krivulje. Zaradi tega je bila vpeljana posplošitev Bezier krivulj s t. i.

baznimi racionalnimi polinomskimi funkcijami (angl. basis functions). V literaturi zasledimo tudi izraz *normirani B-zlepek.*

Enačbi za *i*-to bazno racionalno polinomsko funkcijo stopnje *p* oziroma reda *p + 1* se glasita:

$$N_{i,0}(u) = \begin{cases} 1 & \text{če } u_i \leq u < u_{i+1} \\ 0 & sicer \end{cases} \quad , \tag{6}$$

$$N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u) \quad , \tag{7}$$

pri čemer je $N_{i,p}(u)$ definiran na območju vozliščnega vektorja

$$U = \{\underbrace{a, \ \dots, a}_{p+1}, u_{p+1}, \dots, u_{m-p-1}, \underbrace{b, \dots, b}_{p+1}\}$$

Sedaj lahko podamo enačbo krivulje NURBS stopnje *p*, ki jo določimo z baznimi racionalnimi polinomskimi funkcijami $N_{i,p}(u)$, utežmi w_i (skalarji) in koordinatami *i*-te kontrolne točke:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}, \ a \leq u \leq b, \tag{8}$$

kjer P_i predstavlja lego *i*-te kontrolne točke, $P_i = (x_i, y_i)$, v izbranem ravninskem koordinatnem sistemu. Običajno privzamemo a = 0 in b = 1.

2.1.1 Pomen kontrolnih točk

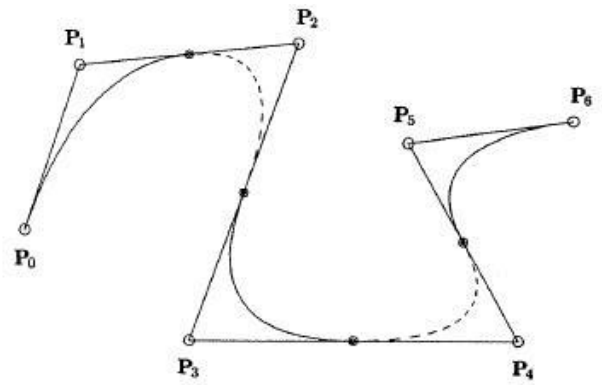
Obliko krivulj NURBS je možno spreminjati s premikanjem kontrolnih točk, ki se v splošnem ne nahajajo na sami krivulji. Vse kontrolne točke skupaj tvorijo kontrolni poligon. Premik posamezne kontrolne točke ne vpliva na obliko celotne krivulje, temveč samo na obliko v območju, ki ga »zajema« posamezna kontrolna točka. Ta lastnost je zaželena, saj omogoča lokalne spremembe brez vpliva na celostno obliko krivulje.

2.1.2 Pomen baznih racionalnih polinomskih funkcij

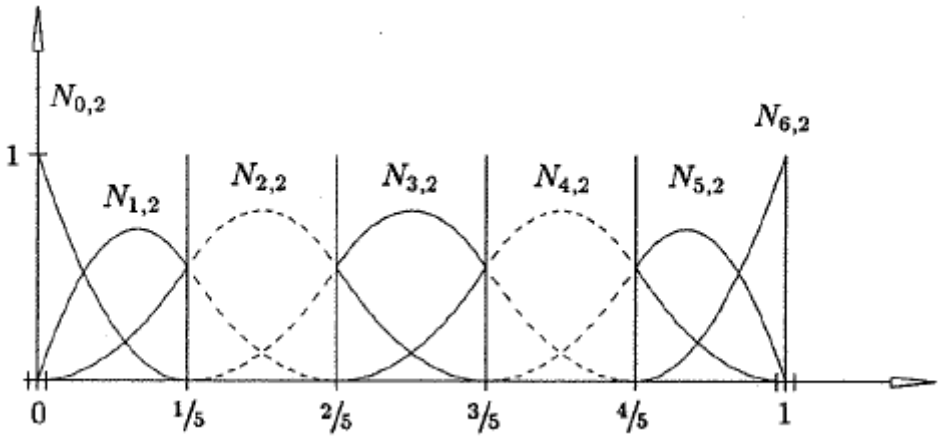
$N_{i,p}(u)$ predstavljajo bazne racionalne funkcije, ki posredno določajo, kakšen vpliv ima posamezna kontrolna točka *i* na obliko krivulje v neki točki (določeni z vrednostjo parametra *u).* Vrednost izvrednotenih baznih funkcij v neki točki na krivulji je numerična (npr. 0.5, 0.3, 0.2), kar za to točko predstavlja 50-odstotni vpliv kontrolne točke *i*, 30-odstotni vpliv točke *i + 1* in 20-odstotni vpliv točke *i + 2*. Vsota izvrednotenih baznih funkcij za vsako vrednost parametra *u* je enaka 1.

Red (pozitivno celo število) celotne krivulje NURBS je določen z redom (stopnjo polinoma) najvišje vsebovane bazne racionalne polinomske funkcije. Število baznih racionalnih funkcij je enako številu kontrolnih točk.

Zmanjšanje stopnje krivulje vpliva na njeno obliko, medtem ko povečanje stopnje ni nujno povezano s spremembo oblike (stopnja 1 = linearen potek krivulje, stopnja 2 = kvadraten potek itd.).



Slika 5: Krivulja NURBS s prikazanimi kontrolnimi koordinatami točk Pi (i ∈ [0,6]), ki tvorijo kontrolni poligon (vir: Les Piegl, Wayne Tiller, 1997, str.85).



Slika 6: Primer kvadratnih osnovnih racionalnih funkcij, ki določajo krivuljo na sliki 5. Vozliščni vektor za podano krivulje je definiran kot U={0,0,0,1/5,2/5,3/5,4/5,1,1,1} (vir: Les Piegl, Wayne Tiller, 1997, str.85).

2.1.3 Pomen vozlov in vozliščnega vektorja

Zaradi potrebe po večjem vplivu posameznih (pomembnejših) kontrolnih točk na obliko krivulje je smiselno določiti območje delovanja ustreznih baznih funkcij. Območje delovanja določa urejena množica vrednosti parametra *u*, ki jo imenujemo vozliščni vektor.

Vozliščni vektor določa torej razdrobitev krivulje oz. domene parametra *u* na poljubne podintervale. Posledica tega je spremenjen vpliv posameznih kontrolnih točk (baznih funkcij) na širšem/ožjem območju krivulje. Število vozlov *m* je povezano s številom kontrolnih točk *n* in stopnjo krivulje *p*.

$$m = n - 1 + p \tag{9}$$

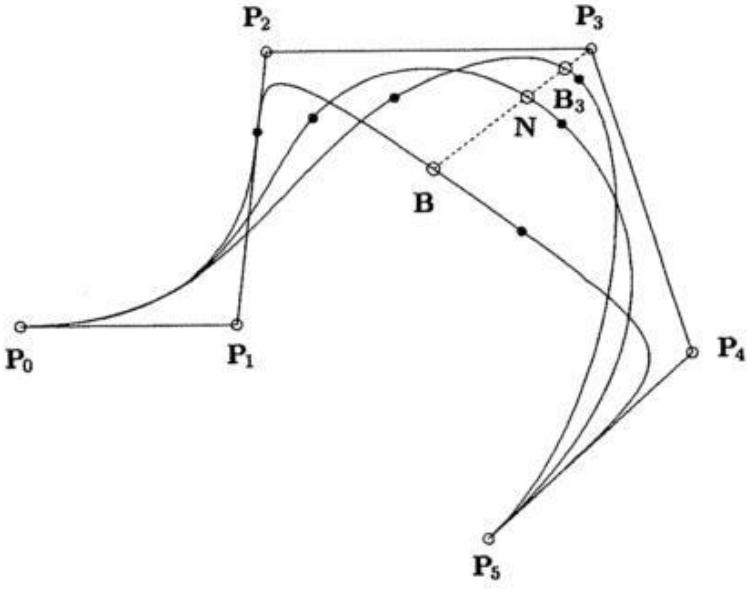
V kolikor je vozliščni vektor osnovnih funkcij podan z enakomernimi intervali med posameznimi vozli govorimo o enakomernem vozliščnem vektorju (npr. $U = \left\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\right\}$).

Premikanje kontrolnih točk pri uporabi enakomernega vozliščnega vektorja ne vpliva na začetno in končno točko same krivulje. Pri samem modeliranju to velikokrat ni zaželjeno, zato je možno vozliščni vektor zapisati tudi z neenakomernimi intervali (slika 6). Pri vrednosti parametra $u = 0$ (začetna točka krivulje NURBS) je s slike razvidno, da na položaj začetne točke vpliva samo bazna funkcija $N_{0,2}$, ki zavzame vrednost 1. To pomeni, da kontrolna točka določena s koordinatami P_0 sovpada z začetkom krivulje, kar poenostavi urejanje krivulje. V kolikor so torej posamezni razdelki med vozli različnih dolžin, govorimo o neenakomernem (angl. non uniform) vozliščnem vektorju.

2.1.4 Pomen uteži

Urejanje lokalne oblike krivulje NURBS je možno tudi s spreminjanjem vrednosti t. i. uteži posameznih kontrolnih točk, ki pripadajoči del krivulje sorazmerno približa oziroma oddalji od taistih kontrolnih točk.

Če $u \in [u_i, u_{i+p+1}]$ in povečamo w_i se točka na krivulji $C(u)$ približa kontrolni točki i. Premik točke $C(u)$ za fiksen u poteka v ravni liniji (slika 7).



Slika 7: Prikaz deformacije krivulje NURBS zaradi vpliva povečanja uteži w3. Premik točke B poteka v ravni liniji (B - N – B3) proti pripadajoči kontrolni točki s koordinatami P3 (vir: Les Piegl, Wayne Tiller, 1997, str.124).

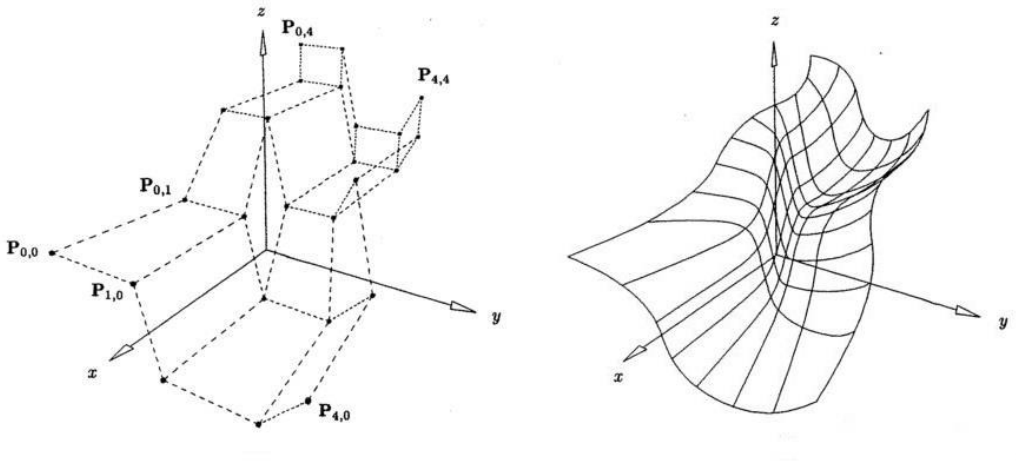
2.2 Ploskve NURBS

Po analogiji za krivulje NURBS je podana tudi parametrična definicija ploskve NURBS stopnje p v smeri u in stopnje q v smeri v, določene z baznimi racionalnimi funkcijami $N_{i,p}(u)$ in $N_{j,q}(v)$ ter utežmi $w_{i,j}$ (slika 8):

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}}, \quad 0 \le u,v \le 1 \tag{10}$$

Koordinate $P_{i,j}$ tvorijo kontrolno mrežo točk (slika 8), $w_{i,j}$ predstavljajo uteži, $N_{i,q}(u)$ in $N_{j,q}(v)$ pa racionalne bazne funkcije definirane z vozliščnima vektorjema

$$U = \{\underbrace{0,\dots,0}_{p+1}, u_{p+1},\dots,u_{r-p-1}, \underbrace{1,\dots,1}_{p+1}\} \text{ in } V = \{\underbrace{0,\dots,0}_{q+1}, v_{q+1},\dots,v_{s-q-1}, \underbrace{1,\dots,1}_{q+1}\}.$$

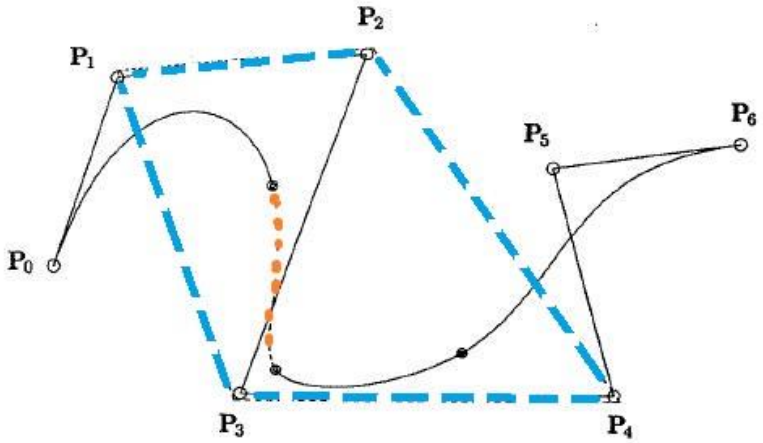


Slika 8: Ploskev NURBS (desno) in pripadajoča kontrolna mreža (levo) točk (vir: Les Piegl, Wayne Tiller, 1997, str.104 in 105).

2.3 Lastnosti krivulj in ploskev NURBS

Posledice lastnosti baznih racionalnih funkcij, ki določajo zapis elementov NURBS, se odražajo kot sledeče pomembne lastnosti krivulj in ploskev NURBS:

1) $C(0) = P_0$ in $C(1) = P_1$.
2) Invariantnost pri geometrijskih transformacijah. Transformacija (npr. pomik) kontrolnih točk krivulje ali ploskve NURBS se odraža kot taista transformacija krivulje ali ploskve NURBS.
3) Če $u \in [u_i, u_{i+1}]$ potem *C(u)* leži znotraj območja, ki ga definirajo koordinate točk P_{i-p}, …, P_i (t. i. konveksna ogrinjača, slika 9).
4) Krivulja *C(u)* je zvezno odvedljiva znotraj odsekov vozliščnega vektorja (med posameznimi vozli) in *p - k* krat odvedljiva v vozlih množljivosti *k.* Množljivost vozla je določena kot število ponovitev posamezne koordinate parametra *u* v zapisu vozliščnega vektorja. Na sliki 6 je množljivost prvega in zadnjega vozla enaka 3, ostali vozli so množljivosti 1.
5) Nobena ravna linija, nima več presečišč z osnovno krivuljo, kakor jih ima s kontrolnim poligonom.
6) Če je *n* = *p* in *U* = {0, …, 0, 1, …, 1} je zapis NURBS krivulje enak zapisu Bezier krivulje.
7) Če premaknemo kontrolno točko *i* ali spremenimo utež w_i, vplivamo samo na del krivulje na intervalu $u \in [u_i, u_{i+p+1}]$.



Slika 9: Prikaz območja konveksne ogrinjače (območje znotraj modre črtkane črte) za posamezen odsek krivulje (rdeča barva) (vir: Les Piegl, Wayne Tiller, 1997, str.119).

3 UPORABLJENA PROGRAMSKA ORODJA

Med izdelavo diplomske naloge sem v procesu geometrijskega in numeričnega modeliranja ter optimizacije konstrukcij, poleg programa Rhinoceros, uporabil tudi številne dodatke (t. i. razširitve ali vtičnike). Nekatere izmed njih zgolj za potrebe razumevanja in učenja samega programskega okolja, drugi pa so za delovanje končnih skript/algoritmov posameznih primerov nujno potrebni. Zaradi lažje preglednosti podajam kompleten seznam uporabljenih programskih orodij in pripadajoče verzije:

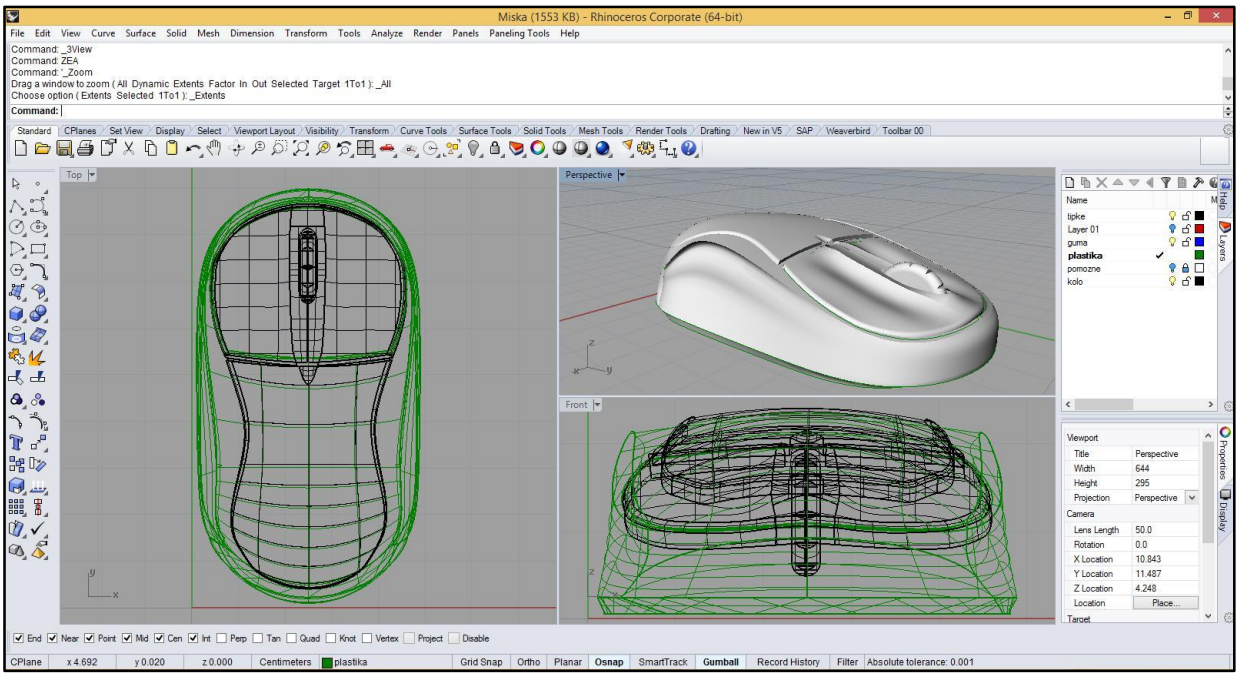
- Rhinoceros (verzija 5 SR7 64-bit, 5.7.31113.14095, 13.11.2013).
- Grasshopper (verzija 0.9.0072 , 6.marec, 2014).
- Geometry Gym/Bullant +Rhino/GH SAP2000 (verzija 1.0.16 GH0.9.0014 v16.1).
- Paneling Tools (verzija 2013.5.8.0).
- Weaverbird (verzija 0.9.0.1).
- Mesh Edit (verzija 25.julij 2011) [2].
- Kangaroo Physics (verzija 0.096, 26.december 2013).
- Octopus (verzija 0.3.3, 8.januar 2014).
- TT Toolbox (verzija 1.4, 25.februar 2014) [3].
- GHPython (verzija 0.6.0.3, 11.december 2013) [4].
- Grasshopper assembly (verzija 1.2.2.1, december 2014) [5].
- SAP2000 (verzija 16.0, 16.1 in 17.1).
- Microsoft Visual Studio Community 2013 (verzija 12.0, december 2014) [6].

3.1 Rhinoceros

Rhinoceros™ je programska oprema za modeliranje (oblikovanje in načrtovanje) podjetja Robert McNeel & Associates [7]. Program odlikujejo visoko zmogljiv in uporabniku prijazen sistem načrtovanja in modeliranja še tako zapletenih oblik oziroma objektov. Uporaba grafičnega vmesnika je relativno enostavna, a zaradi svoje dovršenosti obenem omogoča delovanje na veliko področjih, kjer je potrebna vizualizacija in dokumentacija v 2D in 3D kot npr. industrijsko oblikovanje, navtika, grafično oblikovanje, računalniško podprto načrtovanje (CAD) in proizvodnja (CAM – angl. computer aided manufacturing), arhitektura, strojništvo itd.

Glavne prednosti programa so možnost podajanja, analiza, obdelava in animiranje krivulj NURBS, površin in teles. Povzeto po [8].

Rhinoceros med drugimi omogoča tudi uporabo drugih programskih jezikov (npr. Visual Basic) za izvajanje zaporedij ukazov, kar je nedvomno prispevalo k njegovi razširjenosti in s tem povezanimi področji uporabe. Rhinoceros SDK (angl. software development kit) je programerski pripomoček, ki omogoča direktno branje in pisanje Rhinoceros datotek. Posledica tega je množica dodatkov oziroma aplikacij, ki dodatno povečujejo samo aplikativnost programa, tudi na področjih, kjer uporaba osnovne različice programa prvotno najbrž ni bila predvidena.



Slika 10: Grafični vmesnik programskega okolja Rhinoceros.

Datoteke, ki jih izdelamo v Rhinoceros, imajo končnico .3DM in omogočajo zapis geometrije NURBS. S tem namenom so avtorji okolja Rhinoceros tudi ustanovili odprtokodni projekt (OpenNurbs Initiative) za lažjo izmenjavo geometrije med različnimi programi (CAD, CAM) za obdelavo grafike 3D. Formati, ki jih je mogoče izmenjati, vključujejo tudi splošno uveljavljene formate za izmenjavo datotek CAD: DWG/DXF, IGES, 3DS, STEP, SketchUp, STL itd.

V zadnjih letih so k razvoju in razširjenosti programa veliko prispevali tudi dodatki oz. razširitvena orodja na osnovi okolja Rhinoceros. Na voljo je preko 100 različnih dodatkov in aplikacij namenjenih specifičnim področjem uporabe. Med pisanjem diplomske naloge sem uporabil le nekatere razširitve, ki sem jih potreboval za rešitev točno določenih problemov. Pretežni del izdelave naloge sloni na enem izmed glavnih dodatkov za parametrično modeliranje geometrije: Grasshopper® [9].

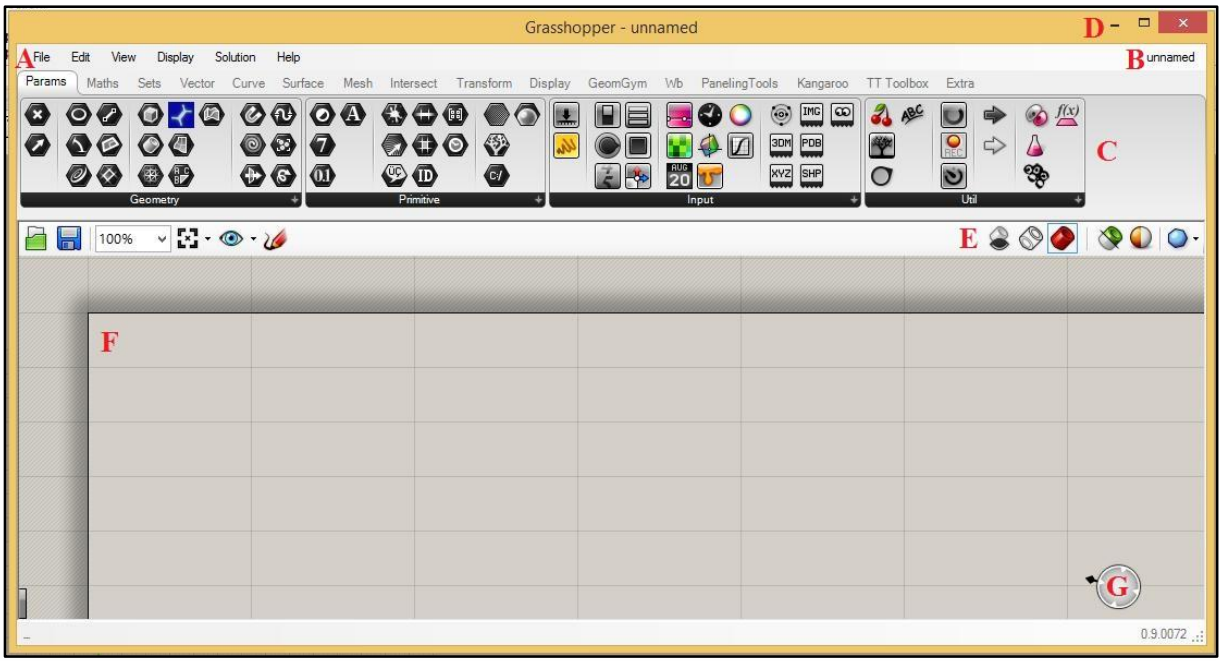
3.2 Grasshopper

Grasshopper je programska razširitev, ki predstavlja grafični urejevalnik algoritma, tesno povezanega z modelirnimi orodji iz programskega okolja Rhinoceros. Temelji na vizualnem programskem jeziku »Grasshopper«. Vizualni programski jeziki operirajo z grafičnimi elementi oziroma ikonami, ki predstavljajo osnovne gradnike kode. Uporabnik te elemente medsebojno poljubno povezuje z interaktivnimi povezavami. V primerjavi s klasičnimi programskimi jeziki uporabnik v Grasshopper ustvarja zaporedja ukazov z grafičnim vlečenjem komponent na t. i. *platno*, namesto podajanja ukazov s tekstom. Predznanje programiranja (npr. v drugih programskih jezikih) praktično ni potrebno, saj je sam proces dokaj intuitiven.

Datoteke ustvarjene s trenutnimi verzijami Grasshopper imajo končnico .gh oziroma .ghx. Povzeto po [10].

3.2.1 Grafični vmesnik okolja Grasshopper

Vmesnik vsebuje številne elemente in je v določeni meri podoben grafičnem okolju osnovnega Rhinoceros.



Slika 11: Grafični vmesnik programskega okolja Grasshopper.

Prikaz sestavnih delov grafičnega vmesnika (slika 11):

A) Osnovni meni, podoben klasičnim programom iz okolja Windows.

B) Iskanje datotek in možnost preklapljanja med že odprtimi datotekami.

C) Paneli z naborom Grasshopper komponent. Razvrščeni v kategorije (npr. »krivulje, ploskve« itd.).

D) Okenska naslovna vrstica.

E) Pripomočki za upravljanje t. i. Grasshopper platna (npr. približevanje, ustvarjanje opomb itd.).

F) Platno predstavlja dejanski urejevalnik skripte oz. zaporedja ukazov.

G) Dodatki za lažji pregled platna.

Ukazi, ki predstavljajo grafične reprezentacije geometrije (npr. ukaz za izris ploskve), se prikazujejo v osnovnem okolju Rhinoceros, ki neprestano poteka v ozadju in predstavlja, v kolikor ta obstaja, ažuren grafičen prikaz Grasshopper skripte (ukazov na platnu).

3.2.2 Osnovni elementi definicij okolja Grasshopper

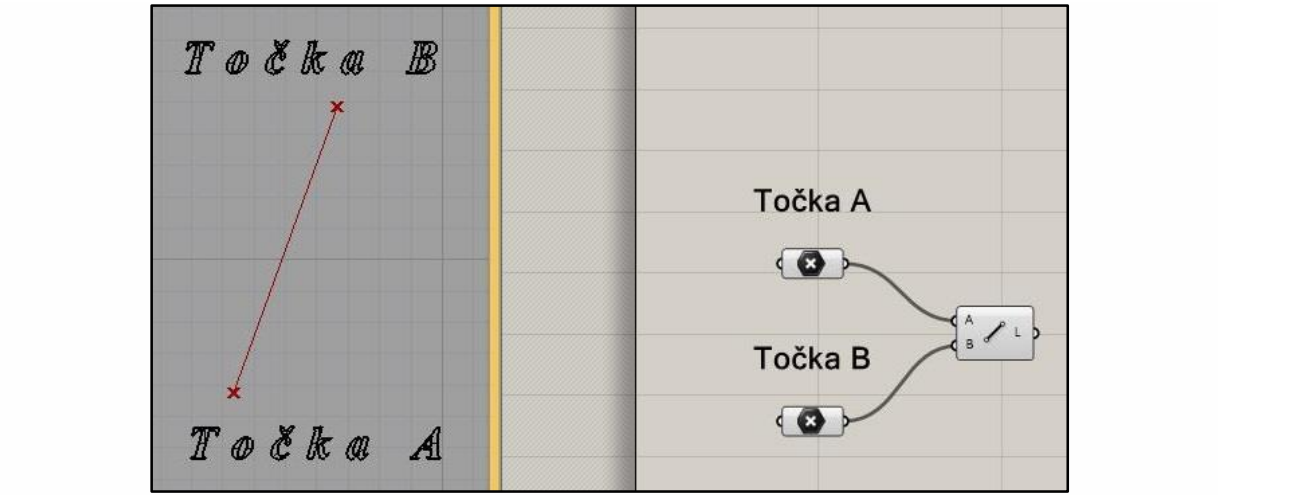
Zapis programa/skripte v Grasshopper v večini sestoji iz sledečih elementov:

- Spremenljivke/parametri (angl. parameters) vsebujejo oziroma shranjujejo podatke. Glede na proces hranjenja teh podatkov v parametrih, Grasshopper razlikuje med stalnimi (angl. persistent) in trenutnimi (angl. volatile) podatki. Trenutni tip podatkov je posredovan iz enega ali več virov/parametrov in se po vsakem zagonu skripte izbriše, ter je na novo pridobljen iz obstoječih parametrov. Stalni podatki so točno določeni s strani uporabnika in so neodvisni od zagona programa.
- Komponente (angl. components) izvršujejo ukaze. Za svoje delovanje potrebujejo podatke, na osnovi katerih ustvarijo določen rezultat (slika 12). Večina komponent vsebuje vhodne podatke (angl. input) in izhodne podatke (angl. output).Vhodni podatki se nahajajo na levi strani posamezne komponente, izhodni podatki pa na desni strani. Podatki lahko predstavljajo različne geometrijske forme (točke, krivulje, ploskve itd.) ali pa so povsem splošni (tekst, cela števila itd.). Zaradi lažje preglednosti algoritma lahko posamezne komponente poljubno preimenujemo. V nadaljevanju naloge je za navajanje Grasshopper komponent uporabljen zapis z zavitim oklepajem npr. {DivideCurve}.

Slika 12: Primer tipične komponente v programskem okolju Grasshopper.

3.2.3 Povezave med komponentami/parametri

Vhodne podatke (spremenljivke) torej obdelujemo s pomočjo komponent. Za dosego končne rešitve je potrebno parametre in komponente povezati v neko logično zaporedje – algoritem. Medsebojne povezave predstavljajo linije, ki jih uporabnik vzpostavi/vleče med relevantnimi izhodnimi podatki posameznih komponent in/ali spremenljivk ter vhodnimi podatki izbranih komponent/spremenljivk. V kolikor se v povezovanju podatkov pojavijo rekurzivne povezave (zanke v povezavah), Grasshopper na napako opozori in skripte ne izvrši. Vsaka komponenta običajno zahteva določen tip vhodnih podatkov (npr. število, vektor, točko itd.), zato povezave delujejo le pri zahtevani vrsti izhodnih in vhodnih podatkov.

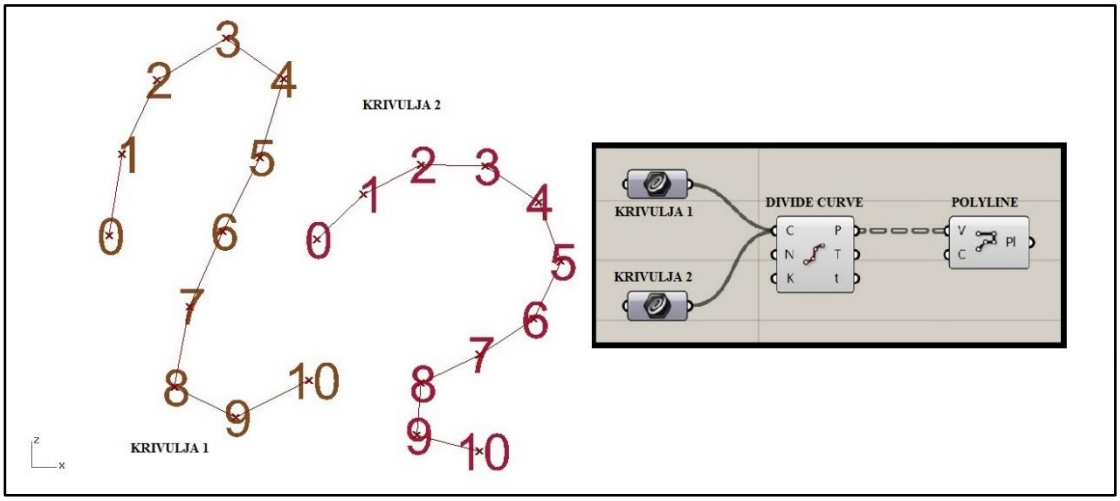


Slika 13: Povezave med izhodnimi in vhodnimi podatki Grasshopper komponent (desna stran slike) ter grafičen prikaz rezultata v okolju Rhinoceros (leva stran slike).

3.2.4 Podatkovne strukture

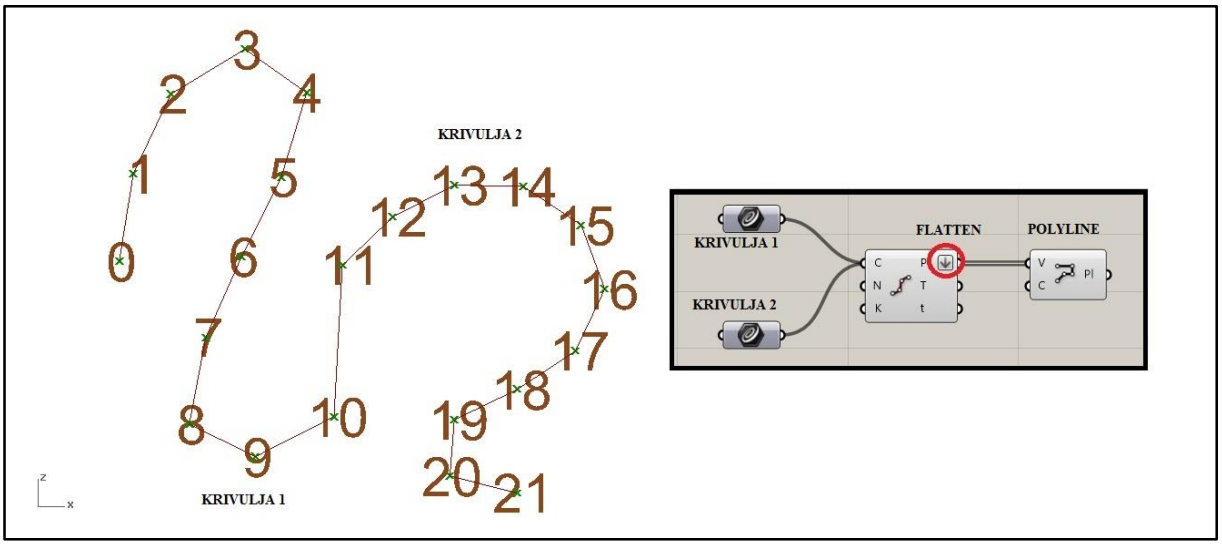
Podatki se v Grasshopper parametrih (verzije 0.6.00xx in kasnejše) shranjujejo v obliki seznamov. Znotraj ene spremenljivke je možno hraniti več kot en seznam podatkov. Zaradi tega je potrebno pri povezovanju komponent in/ali spremenljivk nameniti pozornost lokaciji podatkov znotraj posameznih spremenljivk. Podatki so namreč urejeni v t. i. drevesni strukturi (angl. data tree) oziroma se shranjujejo hierarhično (v nivojih).

Način shranjevanja podatkov v seznamih lahko ponazorimo z enostavnim primerom dveh poljubnih krivulj (slika 14). Krivulji razdelimo na deset enakih delov (komponenta {DivideCurve}), s čimer dobimo seznam 22 točk (11 vrednosti za vsako krivuljo posebej), ki določajo položaj segmentov. Nato uporabimo ukaz za izris krivulje iz točk (komponenta {PolyLine}), kjer kot vhodne podatke vnesemo dobljen seznam točk. Zaradi urejenosti podatkov v t. i. drevesni strukturi, se novi krivulji izrišeta po sledeh prvotnih dveh krivulj, saj Grasshopper loči »lokacijo« podatkov točk za posamezno krivuljo.



Slika 14: Strukturno urejeni podatki vplivajo na potek izvrševanja ukazov posameznih komponent. V tem primeru ukaz »PolyLine« izriše dve krivulji, saj so vhodni podatki (koordinate točk) urejeni v dveh seznamih.

V primeru, da seznam točk ne bi bil ločen, oziroma strukturno urejen (slika 15), bi ukaz {PolyLine} izrisal eno krivuljo skozi vseh 22 točk iz seznama. Okolje Grasshopper dopušča tudi urejanje same strukture podatkov (v tem primeru z ukazom {Flatten}, izničimo kakršnokoli nivojsko ureditev podatkov).



Slika 15: Strukturno urejeni podatki vplivajo na potek izvrševanja ukazov posameznih komponent. Ukaz {PolyLine} v tem primeru izriše eno povezano krivuljo, saj so vhodni podatki (točke) zaradi dodatnega ukaza »Flatten« strnjeni v enem samem seznamu.

3.2.5 Parametrična zasnova geometrije v okolju Grasshopper

Proces modeliranja se v okolju Rhinoceros običajno prične z eksplicitnim vnosom elementov (točke, krivulje, ploskve). Spreminjanje teh elementov v Rhinoceros okolju pomeni izbris in nov izris elementa, oziroma serijo ukazov za transformacije, kot so translacija, rotacija itd. Z naraščanjem števila elementov in kompleksnosti samega modela, začne proces urejanja predstavljati pretežen del delovnega procesa. Kreiranje povsem novih modelov oz. variacij le-teh, zaradi zamudnosti samega oblikovanja torej običajno ni smiselno. Na tem mestu se pokaže prva izmed mnogih prednosti uporabe okolja Grasshopper, ki proces urejanja elementov (ob primerni zasnovi samega modela) bistveno skrajša. Zasnova in modeliranje v okolju Grasshopper namreč poteka parametrično.

Prenos elementa v okolje Grasshopper lahko izvedemo na dva načina tj. direktno z referenco na element, ustvarjen v okolju Rhinoceros, ali pa s parametrično definicijo samega elementa v Grasshopper okolju.

Vsak element (v Rhinoceros in Grasshopper okolju) je namreč definiran z osnovnimi spremenljivkami (parametri) kot npr. radij in središče, ki določata vse potrebne podatke za izris krožnice. Vrednosti teh spremenljivk lahko v Grasshopper okolju definiramo npr. z drsnikom, ki omogoča hitrejše izbiranje (drsenje) prednastavljenih numeričnih vrednosti. Izris geometrijskih elementov za izbrano vrednost spremenljivke (numerične vrednosti drsnika) istočasno poteka v okolju Rhinoceros. Parametrična zasnova in spreminjanje modelov ter možnost praktično neomejenega zapisa algoritma so nedvoumno prednosti uporabe okolja Grasshopper.

3.3 Iskanje optimalne rešitve s pomočjo evolucijskih algoritmov v okolju Grasshopper

Grasshopper v svojem naboru vsebuje tudi komponento {Galapagos}, ki v grobem predstavlja iskalnik optimalne rešitve oziroma numeričnih vrednosti tistih parametrov, ki določajo iskano/optimalno rešitev algoritma. {Galapagos} temelji na t. i. genetskem algoritmu [11], ki spada v skupino t. i. evolucijskih algoritmov [12].

3.3.1 Evolucijski algoritmi

Genetski algoritmi predstavljajo primer (meta)hevurističnega iskanja rešitve po vzoru biološke evolucije (naravnega izbora). Kot taki izkoriščajo prirejeno naključno iskanje z uporabo informacij o ustreznosti posameznih rešitev in napotitvi iskanja v primernejšo »regijo« znotraj razpoložljivih rešitev, dokler ni dosežen globalni maksimum ali minimum. Spadajo v večjo skupino evolucijskih algoritmov, ki iskanje rešitev optimizacijskih problemov rešuje s principi privzetimi iz evolucije (naravnega izbora) kot so dednost, mutacija, selekcija in rekombinacija [13].

Prednosti uporabe genetskih algoritmov so sledeče [14]:

- Sočasno iskanje množice rešitev.
- Vrednotenje z *namensko / sposobnostno / uspešnostno / fitnes* (v nadaljevanju naloge je uporabljen izraz namenska) funkcijo, ki določa, kako primerna/ustrezna je posamezna rešitev.
- Uporaba verjetnostnih operatorjev (izbor, rekombinacija, mutacija), ki preprečujejo oziroma določajo, v katero smer se bo spreminjala populacija (množica potencialnih rešitev). Brez motenj bi bila selekcija deterministična (algoritem kot najprimernejšo rešitev vrne lokalni maksimum in ne globalnega).
- Podrobnosti problema, za katerega iščemo optimalno rešitev, niso relevantne za sam proces iskanja.

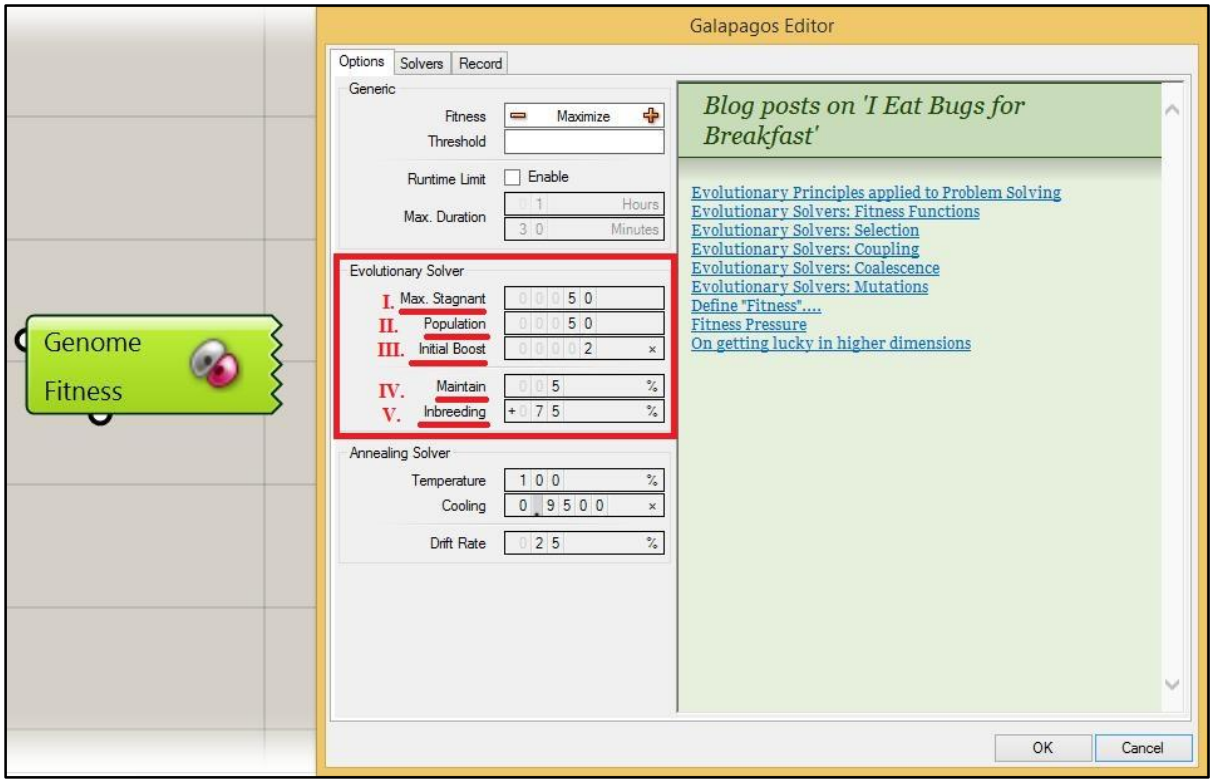
Omejitve genetskih algoritmov:

- V primeru, da je nabor možnih rešitev velik in je čas izračuna posamezne iteracije relativno dolg, lahko čas iskanja optimalne rešitve predstavlja pretežen del celotnega procesa parametričnega modeliranja.
- Zagotovila, da bomo našli optimalno rešitev, ni. Izjemno pomemben je začetni približek (začetna okvirna vrednost) rešitve, ki jo iščemo, sicer proces iskanja lahko poteka v nedogled (oz. do prekinitve s strani uporabnika) in rešitve ne najde, oziroma jo, kot take, v množici ostalih ne prepozna kot optimalne.

3.3.2 Komponenta Galapagos

Uporaba komponente {Galapagos} se nekoliko razlikuje od klasičnih komponent v okolju Grasshopper. Vhodne podatke določimo samo z drsnimi parametri, za katere želimo spreminjati naš parametrični model. Numerične vrednosti drsnih parametrov predstavljajo spremenljivke našega modela. Kriterij za določitev optimalne rešitve predstavlja namenska funkcija, ki jo določimo kot poljubno kombinacijo parametrov in/ali izhodnih podatkov komponent. Namenska funkcija mora predstavljati numerično vrednost, saj kriterij iskanja določimo v obliki pogoja, da vrednost namenske funkcije zavzame bodisi maksimalno ali minimalno vrednost. Ostale nastavitve vplivajo na način izvedbe in čas trajanja algoritma, ter jih je potrebno prirediti glede na število vhodnih podatkov (drsnikov). Kot že omenjeno, je bistvenega pomena nastavitev drsnih parametrov na vrednosti, ki predstavljajo »dober začetni približek« končne optimalne rešitve.

Zagon algoritma povzroči dinamično spreminjanje vseh nastavljenih vhodnih parametrov in sproten prikaz različice modela ter izračun namenske funkcije. Končni rezultat predstavljajo vrednosti vhodnih parametrov, ki določajo optimalno rešitev (model konstrukcije) na podlagi določene namenske funkcije.

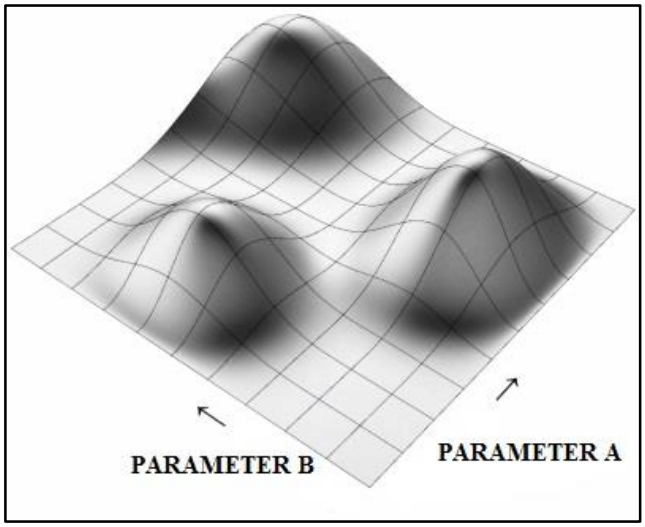


Slika 16: Prikaz osnovnih nastavitev komponente {Galapagos}.

Znotraj okvira {Galapagos} zaradi lažje ponazoritve osnovnih pojmov evolucijskih algoritmov uporabimo privzeti optimizacijski algoritem »evolutionary solver«. Drugo možnost predstavlja »annealing solver«, ki uporablja rahlo drugačen postopek iskanja rešitve, in ga za prikaz iskanja optimalnih rešitev v sklopu diplomske naloge nisem uporabljal. Povzeto po [15].

3.3.3 Ploskev namenske funkcije

Nabor vseh možnih rešitev najlažje ponazorimo s t. i. *ploskvijo/pokrajino namenske* funkcije (ang. fitness landscape). Zaradi lažjega grafičnega prikaza privzemimo, da v modelu nastopata samo dva osnovna parametra, ki ju lahko spreminjamo. Po analogiji iz biologije, ta parametra v našem naboru rešitev predstavljata gena, ki skupaj tvorita/določata genotip (*x* in *y* koordinati posamezne točke ploskve na sliki 17). Izvrednotenje algoritma za kombinacijo osnovnih dveh parametrov (genotip), vrne numerično vrednost, ki na sliki 17 predstavlja višino oziroma *z* koordinato točk. Celotno površino sestavljeno iz vseh točk (koordinate *x*, *y*, *z*) imenujemo ploskev namenske funkcije. Osnovna naloga algoritma je iskanje optimalne vrednosti namenske funkcije, oziroma točke, ki v prostoru leži najvišje.

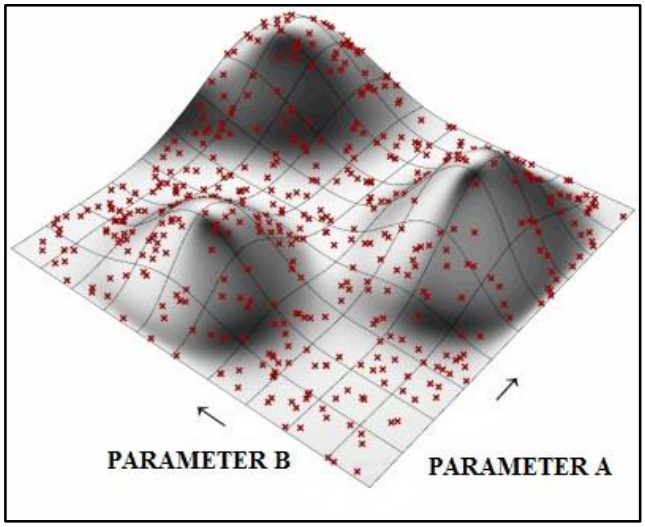


Slika 17: Ploskev namenske funkcije dveh spremenljivk [16].

V našem primeru smo zaradi lažjega grafičnega prikaza privzeli zgolj dva osnovna parametra/spremenljivki. Enostaven prikaz ploskve namenske funkcije z večjim številom (>2) parametrov ni možen. Model z npr. petimi spremenljivkami bi predstavljal »ploskev« v 6 dimenzijah, kar onemogoča vizualno reprezentacijo.

3.3.4 Začetna populacija

Prvi korak iskanja optimalne rešitve je torej izračun naključnih kombinacij osnovnih parametrov, ki naseljujejo dano ploskev. Oblika ploskve namenske funkcije je v tem koraku še neznana (iščemo najvišji vrh le-te).



Slika 18: Ploskev namenske funkcije. Rdeče pike predstavljajo izvrednotene namenske funkcije za posamezno kombinacijo osnovnih dveh parametrov. Površina, na kateri ležijo vse rdeče pike, predstavlja ploskev namenske funkcije [17].

Število kombinacij osnovnih parametrov (s pripadajočimi namenskimi funkcijami) predstavlja začetno populacijo, ki jo določimo pred zagonom algoritma (slika 16, oznaka II.). Algoritem nato naključno izbere kombinacije osnovnih parametrov ter za vsako posebej vrne vrednost namenske funkcije. Izbira velikega števila začetnih kombinacij, tj. začetne populacije, ugodno vpliva na bolj enakomerno poselitev celotne ploskve in s tem večjo možnost, da se že v začetku približamo iskani optimalni rešitvi. Določimo lahko tudi večkratnik prve populacije (slika 16, oznaka III.). Negativna plat je seveda daljši čas izračuna, saj se le-ta veča sorazmerno s številom populacije. Po končanem izračunu lahko populacijo prve generacije že razvrstimo po ustreznosti namenske funkcije. Osredotočimo se na točke (slika 19), ki ležijo višje (koordinata *z*) na ploskvi namenske funkcije.

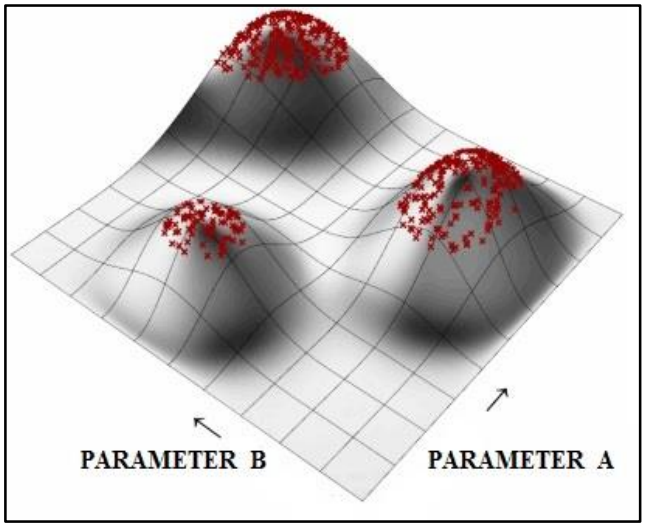
Najustreznejša vrednost namenska funkcija prve populacije še ne predstavlja nujno iskane rešitve (populacija je bila namreč izbrana naključno). Iskanje globalne optimalne rešitve, algoritem nato nadaljuje s kombiniranjem parametrov ustreznejših (»višje ležečih«) genotipov. Iz oblike ploskve (slika 19) je opazna tendenca, da se višje ležeče vrednosti namenske funkcije pojavljajo na treh lokalnih vrhovih celotne ploskve.

3.3.5 Dedovanje

Kot že omenjeno, število prve populacije določimo pred začetkom iskanja optimalne rešitve, ki pa je običajno občutno manjše od števila vseh možnih rešitev. Algoritem nadaljuje iskanje optimalne rešitve z ustvarjanjem novih populacij, ki predvidoma vsebujejo zgolj kombinacije osnovnih parametrov, ki so v začetni populaciji določale najustreznejše (najvišje ležeče) točke.

Iz biologije lahko potegnemo vzporednico s procesom naravnega izbora (naravne selekcije), s katerimi postajajo iz generacije v generacijo pogostejše tiste dedne lastnosti organizmov (v našem primeru parametrov), ki jim dajejo večjo možnost za preživetje in razmnoževanje (kar na nek način predstavlja ustreznost namenske funkcije).

Drugo generacijo torej predstavljajo točke, ki niso več naključno izbrane, saj so že strnjene okoli lokalnih vrhov na ploskvi namenske funkcije (slika 19).



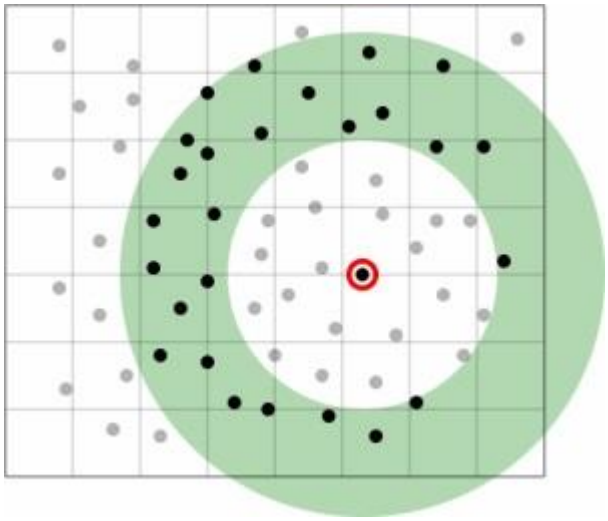
Slika 19: Strnjenost druge generacije izvrednotene namenske funkcije [18].

Pomembno vlogo pri nastanku oziroma izbiri nove populacije ima *faktor homogamije* (slika 16, točka V), ki posredno določa, katere kombinacije osnovnih parametrov bodo izbrane za prenos na potomce oziroma naslednjo generacijo.

Osebki nove generacije torej nastajajo s kombiniranjem osnovnih parametrov (ustreznejših) osebkov stare generacije. Potomce dobimo s križanjem osebkov, ki so na ploskvi namenske funkcije med seboj različno oddaljeni. Osebki, ki ležijo bližje so določeni z osnovnimi parametri, katerih vrednosti se ne razlikujejo veliko. Od tod sledi, da si potomci sorodnih osebkov običajno delijo tudi sorodne osnovne parametre, kar za raznovrstnost same populacije ni najbolj primerno. Nizka raznovrstnost populacije namreč zmanjšuje možnost najdbe alternativnih vrhov na ploskvi namenske funkcije, kar pomeni, da končna rešitev lahko zaznamuje zgolj lokalni optimum (in ne globalno optimalne rešitve).

Drug primer je izbor čimbolj različnih osebkov (različni osnovni parametri), z namenom ustvariti potomce, ki se od prejšnje generacije razlikujejo. Tak način križanja zopet lahko privede do neuspešnega iskanja globalne rešitve, saj v primeru, da je ploskev namenske funkcije sestavljena iz več očitnih vrhov, potomci padejo v prostor vmes (doline), kjer vrednosti namenske funkcije zavzemajo manj ustrezne vrednosti.

V Galapagos-u faktor homogamije predstavlja razmerje med izbiro istovrstnih in raznovrstnih osebkov za nadaljnjo kombinacijo osnovnih parametrov (slika 20).



Slika 20: Spreminjanje faktorja homogamije lahko prikažemo kot spreminjanje notranjega in zunanjega radija zelenega pasu. Točka v sredini predstavlja osebek prve generacije, za katerega iščemo ustreznega partnerja nekje iz območja zelenega pasu. Tem večja razdalja med osebki pomeni tem večjo raznovrstnost osnovnih parametrov, s čimer vplivamo na kombinacijo, ki bo določala potomce [19].

{Galapagos} omogoča tudi prenos istih osebkov med posameznimi generacijami, kar določimo s faktorjem »Maintain« (slika 16, točka IV), ki določa, kakšen procentualni delež najustreznejših osebkov posamezne generacije se (brez sprememb) prenese v naslednjo generacijo. Namen je ohraniti oziroma povečati delež ustreznejših vrednosti namenskih funkcij.

3.3.6 Časovni okvir izračuna

Nabor možnih končnih rešitev oziroma različic modela je direktno odvisen od števila parametrov, ki nastopajo v algoritmu. Končno različico modela določajo posamezni parametri, ki zavzamejo točno določene (numerične) vrednosti. Običajno te vrednosti predpostavimo znotraj določenega območja logičnih (fizikalno smiselnih) mejnih vrednosti.

Glede na število osnovnih parametrov in nastavljene vrednosti, je čas iskanja rešitve lahko dolg. Omejitve časa izračuna lahko določimo z minimalno/maksimalno vrednostjo namenske funkcije, ki jo želimo doseči. Ko algoritem najde kombinacijo osnovnih parametrov, ki določajo predpisano vrednost namenske funkcije, se proces prekine.

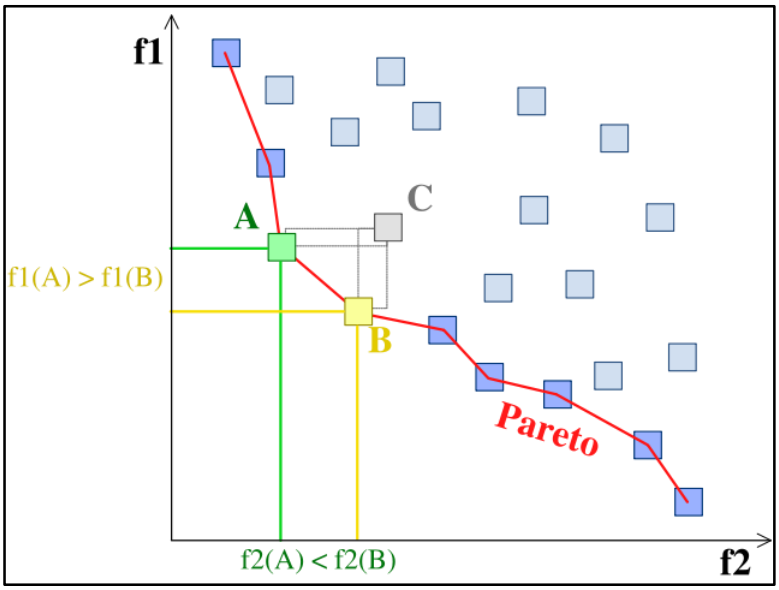
V kolikor okvirne iskane vrednosti ne poznamo, lahko proces omejimo časovno (maksimalna dolžina računanja) oziroma določimo maksimalno število novih generacij (slika 16, točka I.), v katerih algoritem ne najde ustreznejše vrednosti namenske funkcije (nove vrednosti so manjše ali enake že najdenim).

3.3.7 Optimizacija z večkratno namensko funkcijo

V kolikor želimo pri izbiri optimalne rešitve upoštevati več kot en pogoj (namensko funkcijo), je v okolju Grasshopper na voljo dodatek Octopus, ki temelji na algoritmih »SPEA2« (iskanje oziroma aproksimacija Pareto optimuma optimizacijskih problemov z več kot eno namensko funkcijo [27]) in »HypE« (evolucijsko iskanje multikriterijskih optimizacijskih rešitev z uporabo t. i. hipervolumskega

indikatorja [28]). Podrobnejša dokumentacija o tem, kako sta omenjena algoritma dejansko implementirana v delovanje komponente Octopus, žal (zaenkrat) ni na voljo. Pri komentiranju dobljenih rezultatov (v kolikor le-ti niso trivialni) je zato priporočena velika mera previdnosti in obvezna kritična presoja. Zaradi zahtevnosti tematike, ki nekoliko presega okvire te diplomske naloge, je v nadaljevanju podan zgolj enostaven opis uporabe in delovanja komponente.

Optimalne rešitve, ki jih komponenta {Octopus} vrne kot končni rezultat, so lahko povezane z linijo in/ali mrežo, ki predstavlja povezavo nedominantnih rešitev oziroma t. i. Pareto optimum (slika 21). Za tak nabor rešitev velja, da ne obstaja trivialna rešitev, ki bi bila istočasno ustreznejša glede na vse dane kriterije (namenske funkcije). Obenem ustreznosti posamezne rešitve ni možno premakniti bližje posameznemu kriteriju, brez padca ustreznosti glede na ostale kriterije [29].



Slika 21: Primer rešitev (točke na rdeči liniji), ki skupaj tvorijo t.i. Pareto optimum. Koordinatni osi f1 in f2 predstavljata namenski funkciji, pri čemer je ustreznost vsake rešitve prikazana z relativnim položajem glede na posamezno os [29]. Ustreznejše rešitve so bližje koordinatnemu izhodišču.

3.4 Metoda Catmull-Clark

Metoda Catmull-Clark je na področju računalniške grafike namenjena aproksimaciji poljubnih površin z drobitvijo odsekovno linearnih ploskev poligonske mreže na manjše dele.

Rezultat posamezne iteracije metode so vedno štiri-vozliščne ploskve, ne glede na začetno obliko mreže.

V obliki rekurzivnega algoritma sta metodo vpeljala Edwin Catmull in Jim Clark že leta 1978 [34]. Tematika ki obravnava problem aproksimacije poljubnih površin z uporabo mreže sestavljene iz izključno štiri-vozliščnih elementov je pomembna zaradi implikacij na področjih računalniške grafike, ter v našem primeru - metodi končnih elementov. V nadaljevanju je podan eden izmed bolj uveljavljenih

algoritmov za dosego tako oblikovane mreže. V poglavju 4.2.9 je bil ta algoritem implementiran v parametrično zasnovo modela. Metoda [35] je opisana v nadaljevanju.

Izhodišče predstavlja mreža poljubnega poliedra (tj. trirasežnega geometrijskega telesa, omejenega z mnogokotniki).Vsa vozlišča mreže predstavljajo t. i. *izvorne* točke.

i. Za vsako ploskev dodamo novo *središčno* točko, ki predstavlja povprečje vseh izvornih točk izbrane ploskve.

ii. Na vsak rob dodamo novo *robno* točko, ki predstavlja aritmetično sredino dveh sosednjih središčnih točk in obeh končnih točk taistega robu.

iii. Za vsako *središčno* točko dodamo toliko novih robov, kolikor je število robov, ki povezujejo *središčno* točko z vsemi *robnimi* točkami ploskve.

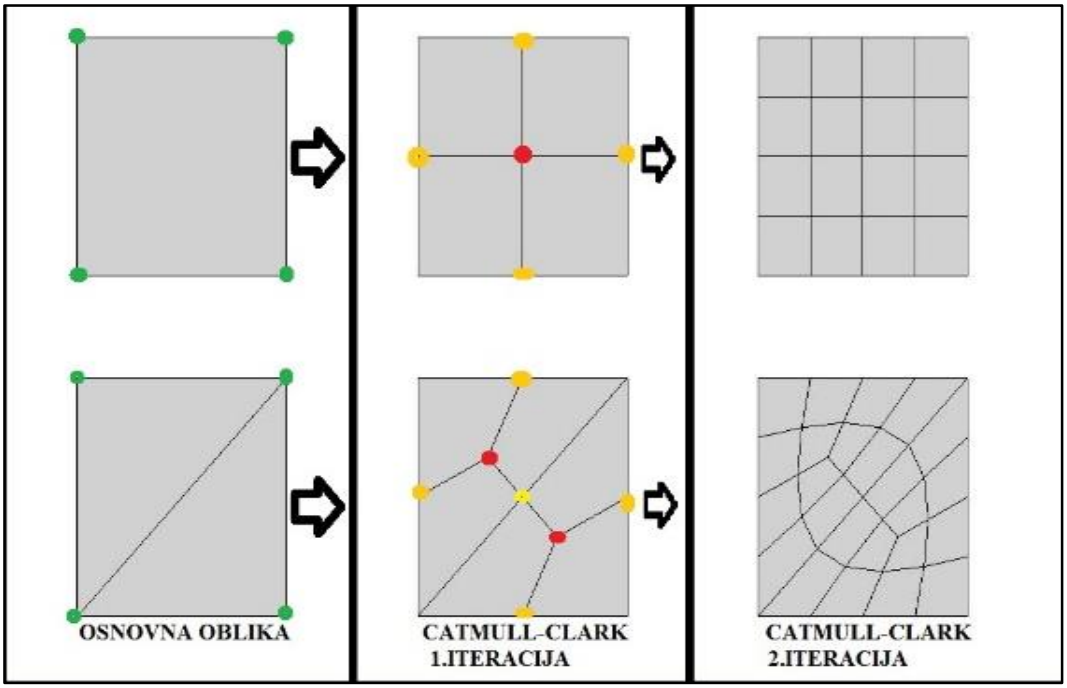
iv. Za vsako *izvorno* točko *P*, definiramo *F* kot povprečje vseh novih *n-središčnih* točk za ploskve, ki se dotikajo *izvorne* točke *P*. Definicija *R* predstavlja povprečje vseh *n*-aritmetičnih sredin robov, ki se dotikajo *P*. Sedaj prestavimo vsako *izvorno* točko na položaj ki ga določa izraz:

$$S = \frac{F+2R+(n-3)P}{n} \tag{11}$$

Točka *S* predstavlja težišče (baricenter) točk *P*, *R* in *F* s pripadajočimi utežmi (*n* - 3), 2 ter 1.

v. Sedaj povežemo vsako novo vozliščno točko z novimi robnimi točkami vseh izvornih robov, ki se dotikajo izvornih vozlišč.

vi. Novi robovi omejujejo ploskve, ki skupaj tvorijo novo mrežo sestavljeno zgolj iz štiri-vozliščnih elementov, ki v splošnem niso ravninski. Nova mreža je v primerjavi z izvorno (v trirazsežnem prostoru) bolj gladka.



Slika 22: Prikaz delitve mreže sestavljene iz dveh tri-vozliščnih elementov (skrajno levo spodaj) oziroma enega štirivozliščnega (skrajno levo zgoraj) z metodo Catmull-Clark. Zelene barva predstavlja izvorne točke, rdeča središčne točke in rumena robne točke.

3.5 Ostali dodatki/vtičniki

Nabor komponent v Grasshopper okolju v osnovi predstavlja pretvorbo ukazov iz okolja Rhinoceros v programski jezik na katerem temelji sam Grasshopper. A vendar je osnovno okolje programa Rhinoceros preobsežno, zato zbirka vgrajenih komponent ne predstavlja vseh možnih ukazov.

Parametrično modeliranje v okolju Grasshopper zahteva nekoliko drugačen vnos vhodno/izhodnih podatkov, zato pretvorba določenih ukazov tudi ni smiselna.

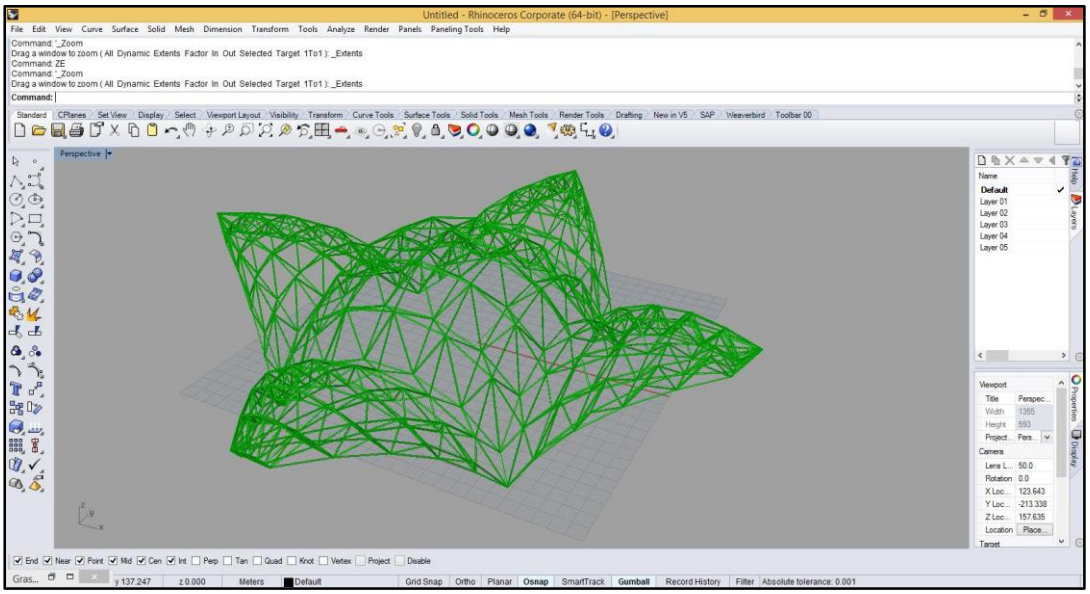
Kot že omenjeno, so dodatki oziroma razširitve Rhinoceros pripomogle k popularnosti in razširjenosti programa. Osnovna ideja teh razširitev oziroma dodatkov velja tudi za samo Grasshopper okolje, saj kot že rečeno, temelji na svojem programskem jeziku. Na voljo je množica dodatkov/vtičnikov, med katerimi nekateri dodajajo komponente k osnovnem naboru, drugi pa omogočajo izmenjavo podatkov z ostalimi programi (npr. Microsoft Excel, Autodesk AutoCad itd.).

3.5.1 GeometryGym / Bullant

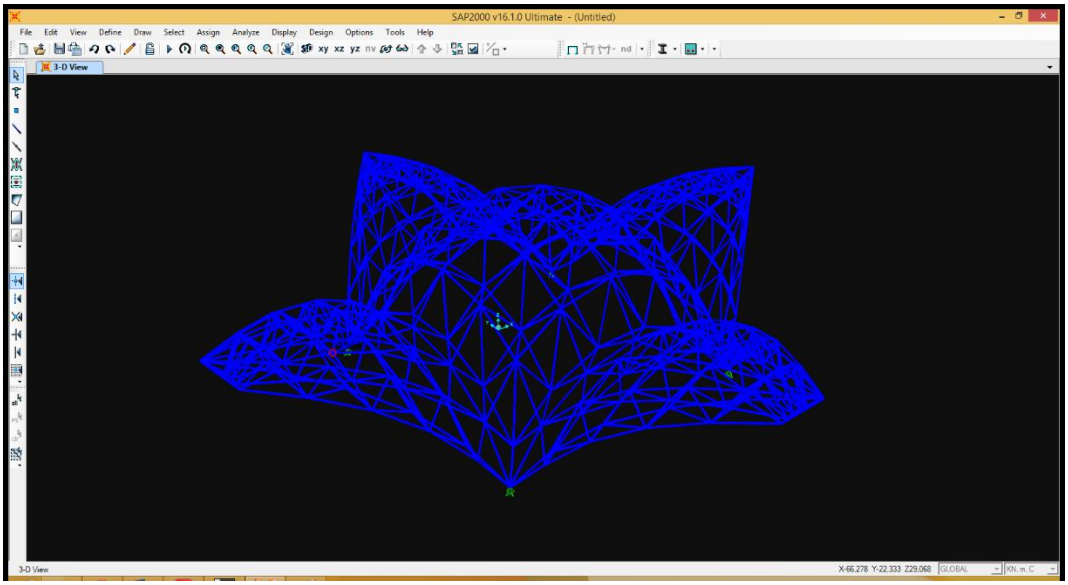
Geometrygym je dodatek [21] namenjen izmenjavi informacij med različnimi programskimi orodji, ter programom Rhinoceros/Grasshopper. Omogoča pretvorbo geometrije NURBS, ustvarjene v okolju Rhinoceros, v formate primerne za nadaljnjo obdelavo in analizo v programih, kot so Revit, ArchiCAd, Tekla, Vasari, Robot, Sofistik, Oasys, Robot, Strand7 ter SAP2000. Stremi k integraciji in izmenjavi BIM (angl. building information model) standardov v različna programska orodja.

Dodatek je zaenkrat še v procesu razvoja. Brez pomoči avtorja programa (Jon Mirtschin) bi bil sam proces programiranja s komponentami GeometryGym sorazmerno težji [22]. Vsi modeli konstrukcij v tej diplomski nalogi so prirejeni izključno za uporabljene verzije programov (glej seznam orodij). Komunikacija GeometryGym komponent s programskim okoljem SAP2000 poteka preko vmesnika OAPI [23]. Vse ostale kombinacije verzij programov so zaradi razlik med posameznimi verzijami SAP2000 programskega vmesnika nekompatibilne.

Prenos modela v programsko okolje SAP2000 poteka preko nabora Bullant komponent iz okolja Grasshopper, ki geometrijske elemente (točke, krivulje, ploskve) modela izvozi v SAP2000 okolje. Definicijo in vrsto konstrukcijskih elementov, prečnih prerezov, podpor, obtežbe in obtežnih kombinacij, prostostnih stopenj, vrsto analize in ostalih podatkov potrebnih za analizo konstrukcije, določimo preko posameznih komponent v samem Grasshopper algoritmu.



Slika 23: Zasnova modela v okolju Grasshopper/Rhinoceros.



Slika 24: Prenos modela v programsko okolje SAP2000 za nadaljnjo analizo konstrukcije.

GeometryGym komponente v okolju Grasshopper se ne razlikujejo od »klasičnih« komponent osnovnega nabora, zato vhodne in izhodne podatke lahko vnašamo in spreminjamo parametrično. Vsaka posodobitev modela iz okolja Grasshopper se samodejno ažurira v programu SAP2000.

Povezava parametrično zasnovanega Grasshopper modela konstrukcije s programskim okoljem SAP2000 nam torej nudi ogromno »manevrskega prostora« pri samem določevanju geometrije konstrukcijskega modela.

Obenem je možno iz Grasshopper v SAP2000 posredovati tudi ukaz za analizo konstrukcijskega modela po metodi končnih elementov (MKE). Proces analize izbrane različice (izbranih parametrov) izvoženega modela se v programu SAP2000 ne razlikuje od klasično zasnovanih (ustvarjenih v okolju SAP2000) modelov. Določene rezultate analize je nato možno uvoziti nazaj v okolje Grasshopper, kjer jih lahko uporabimo kot morebitne vhodne podatke v nadaljevanju skripte. Pregled vseh rezultatov v obliki tabel (datoteka *.xlsx*) za izbran model je torej zaenkrat mogoč samo v okolju SAP2000. Pomanjkanje možnosti prikaza vseh rezultatov je zaenkrat ena občutnejših pomanjkljivosti vtičnika GeometryGym.

Skupaj z uporabo genetskih algoritmov (Grasshopper komponenta {Galapagos}) lahko vmesnik GeometryGym uporabimo za določitev optimalne geometrije modela konstrukcije. Optimizacija oziroma izkoriščenost prečnih prerezov je tu izvzeta, poudarek je na ustrezni parametrični izhodiščni zasnovi geometrije modela, s čimer ustvarimo ogromno množico možnih različic modelov konstrukcije.

Veliko število iteracij v procesu optimizacije pomeni tudi veliko število izvozov samih modelov konstrukcij v SAP2000, pri čemer je možnost napake večja, saj kontrola posameznih iteracij (modela izvožene konstrukcije v SAP2000) v realnem času ni možna. Večina uporabljenih programskih dodatkov oziroma vtičnikov, namreč še ni izdanih v končnih (stabilnih) verzijah. Vsako reševanje

optimizacijskih problemov v okolju Grasshopper z uporabo komponente {Galapagos} je torej zaenkrat smiselno omejiti na reševanje manjših ali lokalnih problemov. Kritična presoja rezultatov je nujno potrebna.

Pozornost je treba nameniti tudi obliki končnih rezultatov, ki so izvoženi nazaj v Grasshopper. Ker je vtičnik še v fazi razvoja, je pregled rezultatov zaenkrat omejen. Vseh možnih rezultatov, ki jih lahko pregledujemo v programu SAP2000 po končani analizi konstrukcije (v obliki tabel), zaenkrat še ne moremo izvoziti preko vmesnika GeometryGym.

Tabela 1: Količine, ki jih lahko interaktivno uvozimo iz programa SAP2000.

LINIJSKI KONČNI ELEMENTI	PLOSKOVNI KONČNI ELEMENTI	PODPORE/VOZLIŠČA
Vrednosti notranjih sil *Fx, Fy, Fz, Mx, My, Mz* za posamezne končne elemente. Smeri in oznake notranjih statičnih količin se ujemajo z lokalnimi osmi elementov iz programa SAP2000. Prikaz vrednosti navedenih količin je možen samo v eksplicitnih točkah (določeno s SAP2000 številom t.i. »output stations«) v tabelah, ki jih izvozi SAP2000.	Vrednosti »povprečnih« normalnih napetosti *SMax* in *SMin* za posamezne ploskovne končne elemente. Obe količini lahko prikažemo za spodnjo ali zgornjo ploskev končnega elementa. Vrednost napetosti v posameznem končnem elementu, ki jo izvozimo nazaj v Grasshopper je definirana kot povprečje štirih vozliščnih vrednosti izračunanih po MKE (tabela 2). Podobno lahko prikažemo vrednosti smeri glavnih napetosti (v vektorski obliki).	Vrednosti reakcijskih sil in momentov (v vektorski obliki). Pomike konstrukcije je možno prikazati za vsako točko konstrukcije.

Tabela 2: Povprečne vrednosti SMax in Smin, ki jih za ploskovne končne elemente, lahko interaktivno uvozimo iz programa SAP2000.

TABLE: Element Stresses - Area Shells									
Area	AreaElem	ShellType	Joint	OutputCase	CaseType	SMaxTop	SMinTop	SMaxBot	SMinBot
Text	Text	Text	Text	Text	Text	KN/m2	KN/m2	KN/m2	KN/m2
ggGHa1	ggGHa1	Shell-Thin	ggGHn20	LASTNA TEZA	LinStatic	536,76	-185,08	-687,04	-4345,25
ggGHa1	ggGHa1	Shell-Thin	ggGHn2	LASTNA TEZA	LinStatic	-1092,81	-3211,65	469,45	-1413,46
ggGHa1	ggGHa1	Shell-Thin	ggGHn1	LASTNA TEZA	LinStatic	2195,81	1118,71	-1944,77	-6460,76
ggGHa1	ggGHa1	Shell-Thin	ggGHn19	LASTNA TEZA	LinStatic	-553,66	-5064,26	912,62	175,02
					AVERAGE=	**271,525**	**-1835,57**	**-312,435**	**-3011,11**

Tabela 2 predstavlja tipičen izvoz rezultatov v obliki tabele iz programa SAP2000. Za ploskovne končne elemente so podane vrednosti izračunanih napetosti. V tabeli so izbrane maksimalne (SMaxTop, SMaxBot) in minimalne (SMinTop, SMinBot) glavne napetosti na zgornji in spodnji ploskvi elementa v vseh štirih vozliščih . Vmesnik GeometryGym za izbrano napetost v končnem elementu vrne povprečje vrednosti v vozliščih končnega elementa (odebeljene vrednosti na dnu tabele). GeometryGym rezultate lahko torej v primeru dovolj goste mreže končnih elementov uporabimo zgolj kot približek ocene napetostnega stanja. Na enak način kot napetosti so izračunane tudi vrednosti smeri glavnih napetosti za posamezen končni element, ki jih prav tako lahko izvozimo v Grasshopper.

Preostala področja uporabe osnovnih funkcij okolja SAP2000, ki jih vmesnik GeometryGym (zaenkrat) še ne podpira, obsegajo sledeče procese modeliranja: uporaba končnih elementov tipa »solid«, podajanje anizotropnih in ortotropnih materialov, definiranje poljubnih prečnih prerezov, podajanje premične obtežbe itd. V zadnjem delu diplomske naloge je prikazana implementacija določenih manjkajočih funkcij s pomočjo programiranja lastne Grasshopper komponente.

Ostali uporabljeni dodatki/programski vmesniki so bili uporabljeni manj pogosto, zato so v nadaljevanju povzeti zgolj s kratkim opisom:

- **Paneling Tools**

  Paneling Tools je dodatek za okolje Grasshopper, ki omogoča racionalizacijo NURBS površin za enostavnejšo nadaljnjo obravnavo [24].

- **Weaverbird**

  Dodatek Weaverbird je namenjen predvsem obravnavi t. i. mrež (angl. mesh) v okolju Grasshopper in z njimi povezanimi geometrijskimi lastnostmi/operacijami (transformacije, cepitve…). Uporaben za obdelavo morebitne mreže končnih elementov [25].

- **Kangaroo Physics**

  Omogoča interaktivno simulacijo modelov ustvarjenih v okolju Grasshopper (obnašanje geometrije v prostoru in njeno vzajemno delovanje) [25].

- **Octopus**

  Dodatek z evolucijskimi algoritmi za iskanje rešitev z več kot eno namensko funkcijo [26]. Obsega relativno zahtevne iteracijske postopke, ki jih komponenta Galapagos (zaenkrat) ne podpira.

- **SAP2000 OAPI**

  Kljub temu da določeni dodatki za Grasshopper (Karamba, Kangaroo itd.) že omogočajo (enostavnejše) analize konstrukcij, je bil za potrebe računanja konstrukcij po metodi končnih elementov uporabljen računalniški program SAP2000. Odlika izbranega programa je (nedavno) razviti programski vmesnik OAPI (angl. open application programming interface), ki dovoljuje povezovanje oziroma dostop do elementov in funkcij tudi »zunanjim« programom. Na ta način je možno z zunanjimi aplikacijami upravljati s programom brez uporabe grafičnega vmesnika, z ukazi spisanimi v podprtih programskih jezikih Visual Basic, Visual Basic for Applications, Visual C#, Intel Visual Fortran, Microsoft Visual C++, Matlab in Python. Vmesnik GeometryGym torej predstavlja prevajalnik ukazov iz programskega jezika Grasshopper, v format podprt s strani SAP2000 OAPI vmesnika.

4 PRIMERI

V nadaljevanju prikazujem štiri primere parametrično zasnovanih konstrukcij oziroma posameznih delov konstrukcij, modeliranih in analiziranih z uporabo naštetih programskih orodij (poglavje 3). Zaradi preglednosti in lažjega razumevanja dejanskega postopka sem prvi primer (prosto ležeča plošča) prikazal podrobneje (z grafičnim prikazom in opisom dejanskega poteka algoritma v okolju Grasshopper) kot ostale, katerih algoritme sem zaradi obsežnosti podal zgolj v obliki diagrama poteka.

V prilogah so podane razpredelnice z naštetimi Grasshopper komponentami in diagrami poteka algoritma za posamezne primere.

4.1 Armiranobetonska prosto ležeča plošča

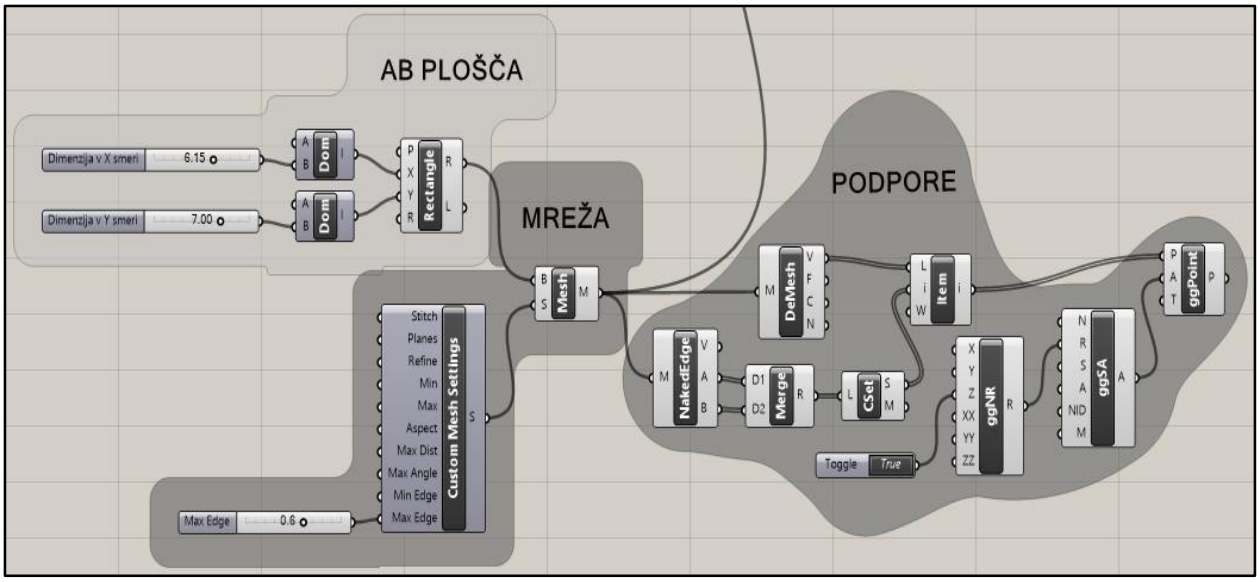
Armiranobetonska plošča predstavlja tipičen primer ploskovne konstrukcije, katere parametrična ponazoritev v okolju Grasshopper predstavlja dokaj enostaven zapis skripte, ki omogoča izvoz izbranega modela konstrukcije v okolje SAP2000, kjer se po metodi končnih elementov izvrši izračun notranje statičnih količin.

4.1.1 Zasnova in parametrizacija geometrije konstrukcije

Zasnova modela plošče v okolju Grasshopper izhaja iz pravokotnika, ki mu določimo dimenziji v *x* in *y* smeri koordinatnega sistema. Dimenziji podajamo z dvema drsnima komponentama (slika 25), ki omogočata hitrejše izbiranje med (okvirno prednastavljenimi) vrednostmi. Izhodni podatek komponente {Rectangle} je torej pravokotnik dimenzij, določenih z drsnikoma {Dimenzija v X smeri} in {Dimenzija v Y smeri}.

Komponenta {Mesh} geometrijo vhodnega podatka (pravokotnik) prekrije s poligonsko mrežo, sestavljeno iz ravnih linij in vozlišč na mestih stikov linij. Dimenzije osnovnih celic, ki sestavljajo mrežo, zopet določimo z drsno komponento {MaxEdge}. V zadnjem delu algoritma se namreč celice poligonske mreže pretvorijo v mrežo končnih elementov, katerih dimenzije in s tem posledično število elementov je smiselno kontrolirati, saj pomembno vplivajo na rezultate analize konstrukcije.

Poligonsko mrežo nato s serijo ukazov razdrobimo na osnovne gradnike. {NakedEdge} komponenta poišče indekse začetnih in končnih vozlišč, ki ležijo na zunanjih robovih modela plošče. {Merge} in {CSet} združita indekse vozlišč v seznam, ki ne vsebuje podvojenih indeksov. Izmed vseh vozlišč mreže s komponento {Item} izberemo tiste, ki ležijo na zunanjih robovih. Komponenta {ggPoint} pretvori ta vozlišča v točke, ki jih program SAP2000 prepozna kot podporne točke. Z dodajanjem komponente {ggNR} in {ggSA} določimo prostostne stopnje izbranih podpornih točk.



Slika 25: Prvi del algoritma, ki opredeljuje geometrijo modela armiranobetonske plošče in način podpiranja na robovih. Zgornja izhodna povezava komponente {Mesh} se navezuje na drugi del algoritma (slika 26).

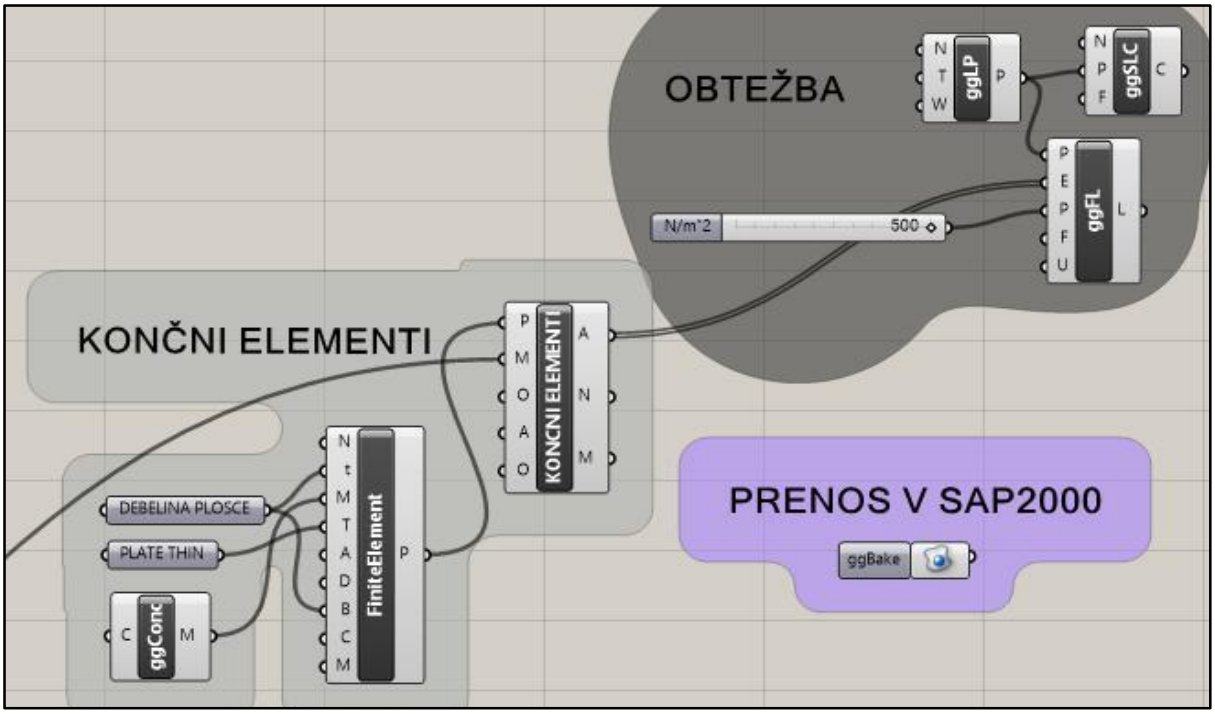
Končne elemente za prenos v program SAP2000 definiramo s komponento {ggMFE}, ki poligonsko mrežo pretvori v dimenzijsko enakovredno mrežo ploskovnih končnih elementov (izberemo tip »Plate«). Glede na to, da modeliramo armiranobetonsko ploščo, je v modelu uporabljen račun po Reissner-Mindlinovi teoriji (»thick plate «). Z {ggAP} določimo še preostale potrebne lastnosti (debelina, tip končnega elementa itd.), ki jih za uspešen prenos zahteva programsko okolje SAP2000, ter z {ggConc} določimo vrsto betona.

4.1.2 Obtežba in podpiranje konstrukcije

Velikost enakomerne površinske obtežbe končnih elementov določimo z {ggFL}, ki je povezan z drsnikom za hitrejšo izbiro vrednosti. Obtežni vzorec (»SAP2000 load pattern«) podamo z {ggLP}, obtežni primer (»SAP2000 static load case«) pa z {ggSLC}.

Izberemo sledeče vrednosti drsnikov: Dimenzija v X smeri = 6.15, Dimezija v Y smeri = 7.0, Velikost obtežbe = 500 in Maksimalen rob = 0.6. Debelino plošče podamo kot t = 0.14.

Vsi parametri so podani brez enot, saj v okviru nastavitev zadnje komponente {ggBake} določimo tudi le-te. Izberemo kN (kilonewton), m (meter) in C (stopinja celzija), ki sedaj predstavljajo osnovne merske enote za vse uporabljene parametre. Klik na komponento {ggBake} izvrši prenos 2D-modela plošče iz okolja Grasshopper/Rhinoceros v program SAP2000.



Slika 26: Drugi del algoritma, ki opredeljuje uporabljene končne elemente, ter površinsko obtežbo modela plošče. Povezava komponente {KONCNI ELEMENTI} z levega robu slike se navezuje na prvi del algoritma (slika 25).

4.1.3 Analiza konstrukcije in optimizacija geometrije

Posledica parametričnega modeliranja je hiter in enostaven izris množice različnih modelov armiranobetonske plošče. Osnovne parametre, ki določajo končno formo modela, v našem primeru predstavljajo:

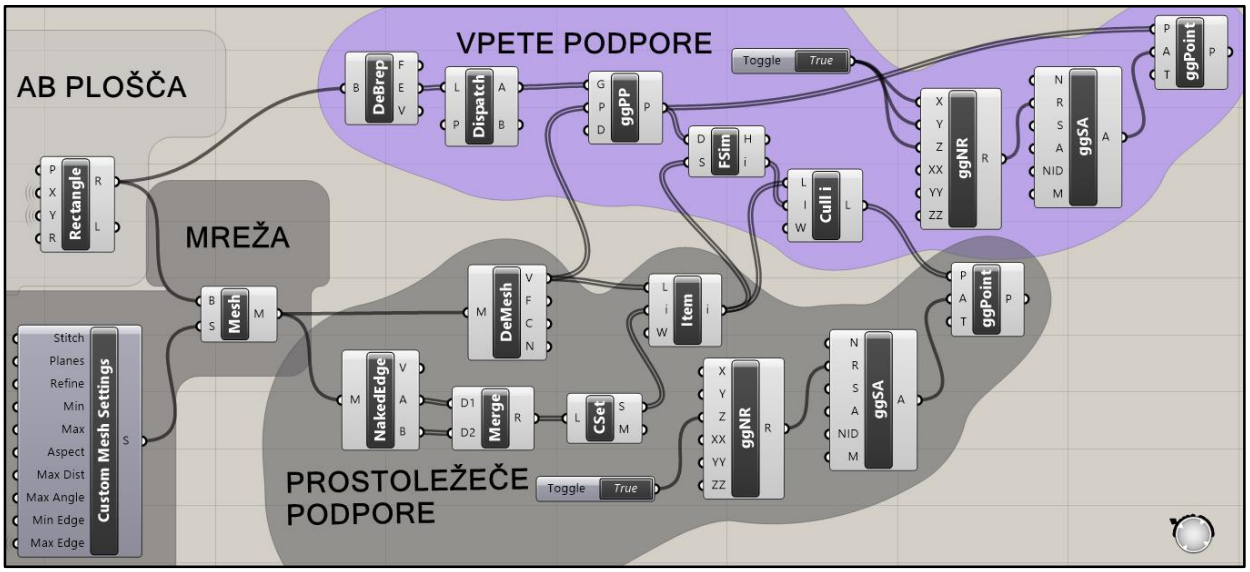
- Dimenzija plošče v *x* smeri (v metrih).
- Dimenzija plošče v *y* smeri (v metrih).
- Maksimalna dimenzija stranice posameznega pravokotnega končnega elementa (v metrih).
- Velikost enakomerne površinske obtežbe končnih elementov (v N/m^2).

Že uporaba samo štirih (spremenljivih) parametrov omogoča izdelavo praktično neomejenega števila modelov s preprostim spreminjanjem vrednosti drsnikov izbranih parametrov.

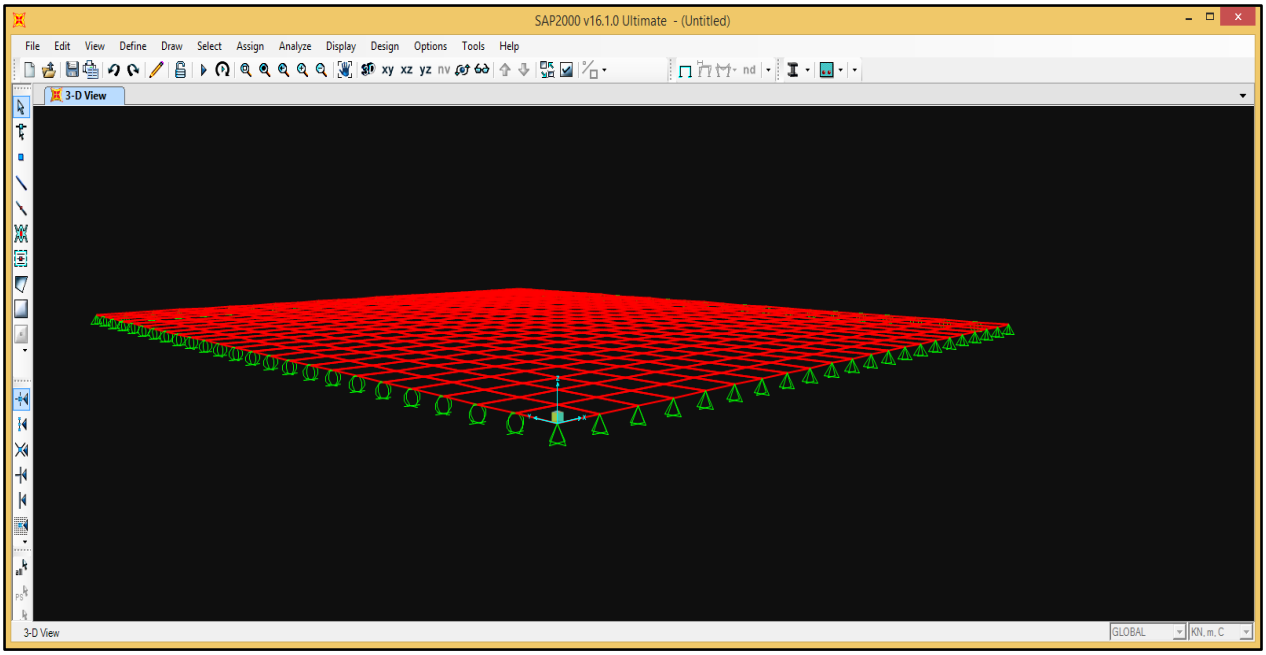
V kolikor nam takšna zasnova modela ne ponuja želene/iskane rešitve, lahko z dokaj preprostim spremembami samega algoritma, model ustrezno spremenimo oziroma dopolnimo. Spreminjanje ostalih fizikalnih parametrov modela kot npr. debelina plošče, način podprtja, vrste obtežbe, izbire trdnostnega razreda betona itd., poteka v okolju Grasshopper z logičnim dodajanjem/spreminjanjem zaporedja komponent in/ali relevantnih parametrov. V nadaljevanju je podan primer, kjer obstoječi model armiranobetonske plošče želimo podpreti na drugačen način.

V kolikor želimo spremeniti način podprtja modela plošče tako, da sta robova vzdolž globalne koordinatne smeri *x* vpeta (translacije v smereh osi *x, y, z* preprečimo), vzdolž smeri *y* pa ostaneta prostoležeča, je sprememba algoritma sledeča (slika 27).

{DeBrep} komponenta pravokotnik, ki določa geometrijo plošče, »razgradi« na 4 stranice, {Dispatch} pa nato iz seznama teh stranic izbere 2 para glede na podan vzorec (zaporedje True/False trditev) izbiranja. Z ukazom {ggPP} poiščemo vsa vozlišča poligonske mreže, ki so v neposredni bližini izbranega para stranic (stranici vzdolž globalne *x* smeri koordinatnega sistema). Indekse teh vozlišč poiščemo z {FSim}, ter jih z {Cull i} odstranimo iz osnovnega seznama vozlišč, ki ležijo na zunanjih robovih modela. Odstranjene točke sedaj z {ggNR}, {ggSA} in {ggPoint} definiramo kot vpete podporne točke v programu SAP2000. Osnovni seznam vozlišč (brez odstranjenih/vpetih točk) predstavlja sedaj novi seznam prostoležečih podpor.



Slika 27: Spremenjeni del algoritma, ki vpliva na način in položaj podpiranja modela plošče.



Slika 28: Sprememba načina vpetja na končnem modelu prosto ležeče plošče v programu SAP2000. Dotikajoči robovi plošče so vpeti različno.

Primer spremembe načina vpetja je možno v samem algoritmu spremeniti na veliko načinov. Spreminjanje modela je relativno občutljivo na morebitne napake v skripti, zato je izredno pomembna čim enostavnejša izhodiščna zasnova samega modela. Z večanjem števila komponent in parametrov, se zaradi same količine vhodnih in izhodnih podatkov ter njihove medsebojne povezanosti, veča tudi možnost nepredvidene napake v sami skripti in posledično nepravilen izris končnega modela.

Razpon plošče določata dva drsna parametra (dimenzija plošče v x in y smeri), vsak omejen z maksimalno vrednostjo 10 metrov in možnostjo izbire do 0.01 metra natančno. Posamezen drsnik torej lahko zavzame 10^3 različnih vrednosti (od 0.01m do 10.00m), razpon polja plošče pa določata oba, ki skupaj predstavljata kar 10^6 možnih kombinacij izbranih vrednosti. Posledično to pomeni tudi enako število možnih različic končnega modela plošče. Z vsakim nadaljnjim spremenljivim parametrom (npr. različne debeline modela plošče) se število možnih modelov ustrezno poveča. Smiselnost takega izbora robnih vrednost drsnikov je seveda vprašljiva (dejanska izvedba plošče z mejnimi dimenzijami), a jo upravičimo zaradi kasnejšega prikaza iskanja optimalne rešitve v množici vseh možnih.

Ob takšni količini oziroma naboru možnih rešitev – končnih modelov konstrukcije, je potrebno pozornost nameniti tudi kriteriju za določitev/izbiro željene različice končnega modela konstrukcije. Velikostni red možnih rešitev namreč običajno krepko presega zmožnosti posameznika za obravnavno in kritično presojo vseh generiranih rešitev.

Uporabljen je model plošče podan z algoritmom iz točke 4.1.1 (prostoležeče podprta plošča).

Komponento {Galapagos} povežemo z novima drsnikoma, ki določata debelino plošče in velikost enakomerne obtežbe. Ta dva parametra sedaj predstavljata edini spremenljivki, ki vplivata na končni model plošče, ki se izvozi v SAP2000. Izračun po metodi končnih elementov vrne rezultate, od katerih kot namensko funkcijo izberemo vrednost povprečne (glej razlago tabele 2) manjše vrednosti glavne napetosti na spodnjem robu plošče.

Zagon algoritma z uporabo komponente {Galapagos} glede na spreminjanje osnovnih dveh parametrov, skuša poiskati *najmanjšo* vrednost namenske funkcije. Glavni napetosti sta v izpisu rezultatov SAP2000 urejeni tako, da velja:

$$\sigma_1 \geq \sigma_2 \,, \tag{12}$$

torej iščemo *minimalno vrednost* $\sigma_2 = \sigma_2(h, p)$ *na zgornjem robu plošče*, pri čemer *h* predstavlja debelino plošče in *p* velikost enakomerne površinske obtežbe na zgornji ploskvi v smeri negativne *z* osi. Ostali parametri so tekom analize konstantni.

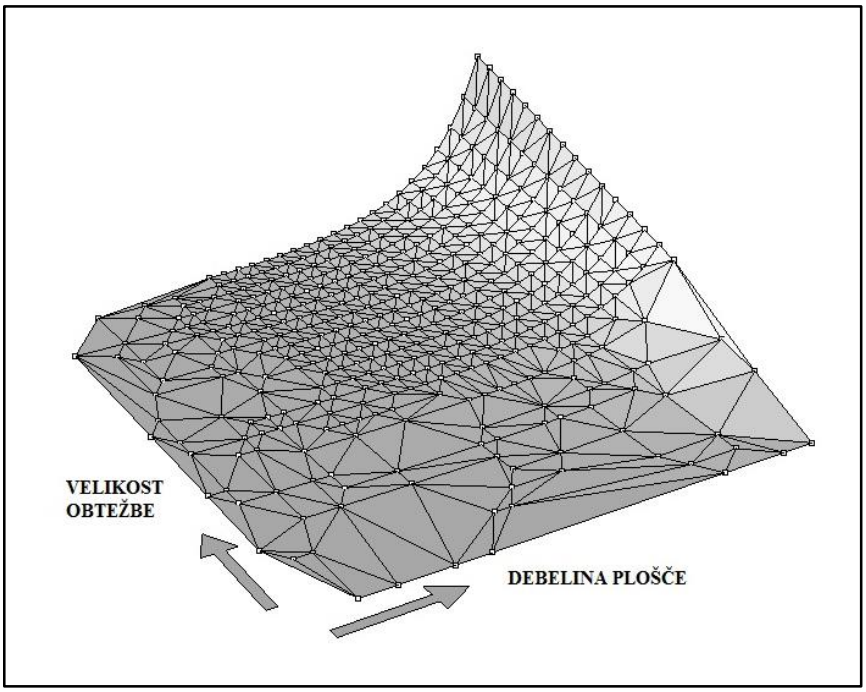
Predvidevamo, da se bodo največje tlačne napetosti ($\sigma_{2,\min}$) na zgornjem robu, pojavile na sredini razpona plošče, in sicer pri najnižji vrednosti parametra A (debelina plošče), ter najvišji vrednosti parametra B (velikost enakomerne obtežbe). Povzetek iskanja optimalne rešitve je podan v tabeli 3.

Tabela 3: Povzetek iskanja optimalne rešitve modela plošče z uporabo namenske funkcije.

KARAKTERISTIKE MODELA		
Konstantne vrednosti:	Dolžina v *x* smeri: Dolžina v *y* smeri: Material: Modul elastičnosti: Poissonov količnik: Maksimalen rob mreže končnih elementov:	6.15m 7.00m BETON C25/30 3100 kN/cm2 0.2 1.0m
Spremenljivka št.1:	DEBELINA PLOŠČE	0.08m
Spremenljivka št.2:	ENAKOMERNA POVRŠINSKA OBTEŽBA	0.050 kN/m^2
Konstantne nastavljene vrednosti {Galapagos} komponente:	ŠTEVILO STAGNIRANIH POPULACIJ (»Max. Stagnant«): ŠTEVILO ZAČETNE POPULACIJE (»Population«): ZAČETNO POVEČANJE (»Initial Boost«): DELEŽ OHRANITVE (»Maintain«): FAKTOR HOMOGAMIJE (»Inbreeding«):	10 30 2 5% -50%
Optimalna namenska funkcija:	PRIPADAJOČA MINIMALNA GLAVNA NAPETOST NA ZGORNJEM ROBU PLOŠČE	-94.64 kN/m^2

4.1.4 Komentar rezultatov

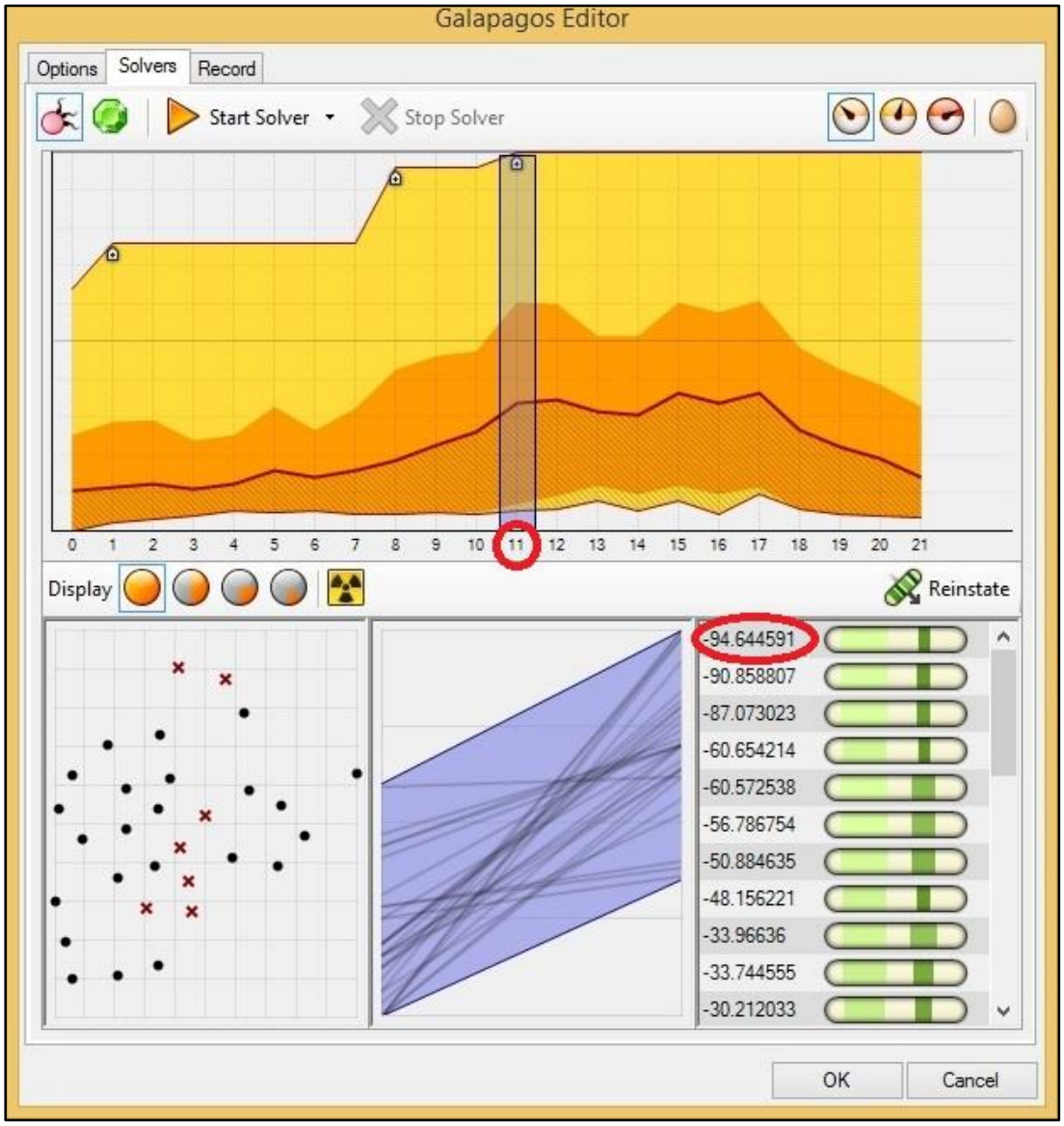
Izkaže se, da je dejanska oblika ploskve namenske funkcije trivialna in skladna s predvidevanji o obnašanju plošče. Na sliki 29 je jasno prikazan najvišji vrh ploskve, ki zaznamuje izvrednoteno namenske funkcijo pri danih parametrih (tabela 3). Celotna ploskev je sestavljena iz 751 točk, kar pomeni, da je algoritem opravil toliko izračunov notranjih statičnih količin po metodi končnih elementov, za različne kombinacije osnovnih parametrov.



Slika 29: Dejanski model prostora namenske funkcije za primer armiranobetonske prosto ležeče plošče iz 4.1.1. Namenska funkcija izvrednotena v 751 točkah, ki so na sliki predstavljene kot vozlišča mreže. Višina posamezne točke (z koordinata) predstavlja uravnoteženo vrednost izračunane minimalne glavne napetosti.

Vrednosti, ki jih zavzameta oba osnovna parametra, sta minimalna (debelina plošče) in maksimalna (velikost enakomerne obtežbe) vrednost, ki ju dotična parametra sploh lahko zavzameta. V samem algoritmu smo namreč nastavili okvirne vrednosti obeh parametrov. Očitno je torej, da je trivialna rešitev, ki jo vrne algoritem pravilna.

Na sliki 30 je prikazan rezultat po posameznih generacijah (izbrano število kombinacij na generacijo je 30) ter pripadajoče vrednosti namenske funkcij.

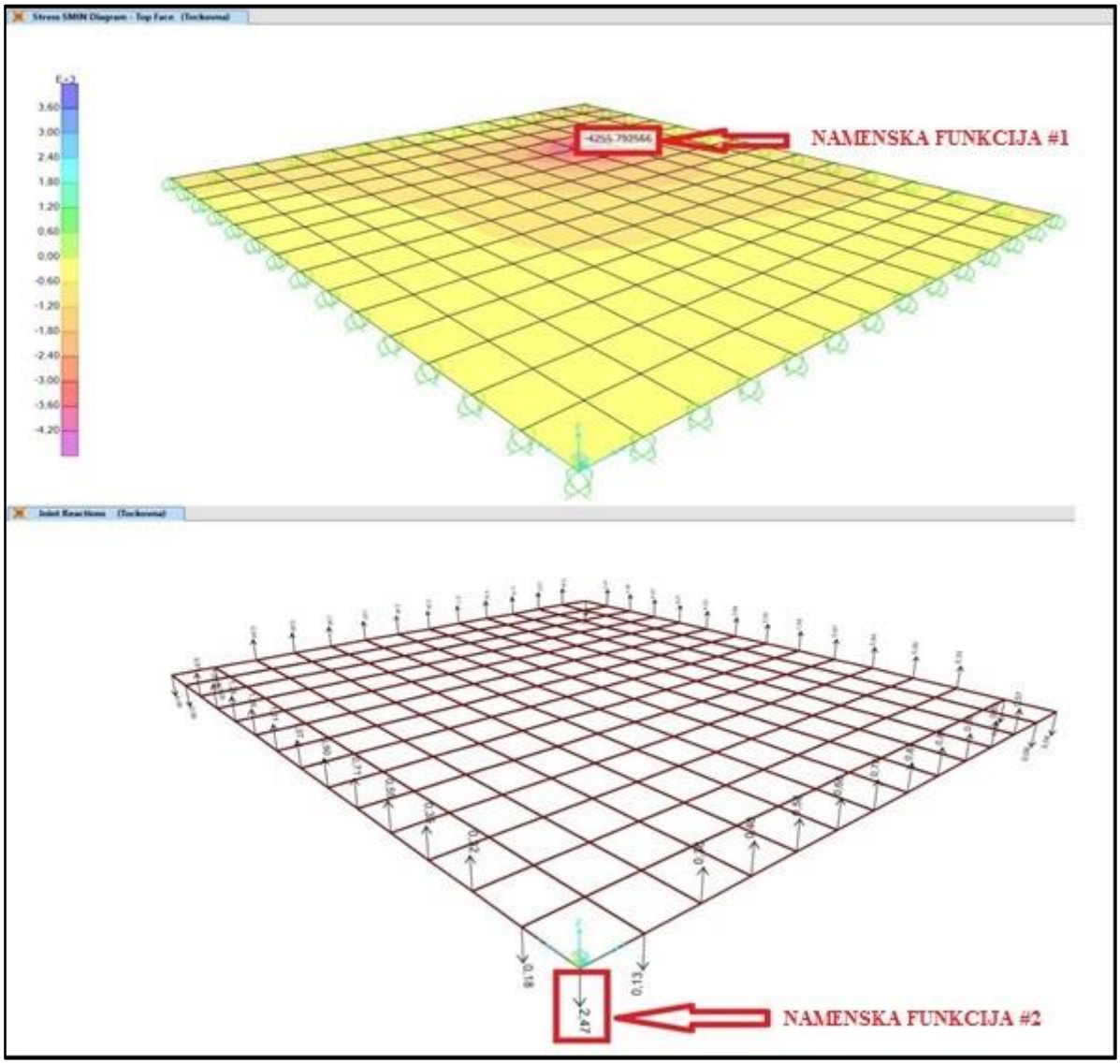


Slika 30: Končni prikaz izračuna namenske funkcije po posameznih generacijah. Končna (optimalna) vrednost (-94.64) je dosežena v 11.generaciji. Ker algoritem v naslednjih desetih (nastavljena vrednost) generacijah ni našel ustreznejše vrednosti, se proces iskanja ustavi.

Pri preizkušanju različnih nastavitev komponente {Galapagos} je do izraza zopet prišlo dejstvo, da dober začetni približek osnovnih parametrov občutno skrajša celoten čas izračuna. V kolikor namreč parametre nastavimo blizu optimalnih vrednosti, algoritem končno (pravilno!) vrednost namenske funkcije najde že v prvih nekaj generacijah. Obenem je zopet smiselno poudariti dejstvo, da evolucijski algoritmi, kljub skrbno izbranim nastavitvam, optimalne rešitve ne najdejo nujno. V našem primeru armiranobetonske plošče je bila le-ta trivialna in zato enostavno preverljiva (z ročnim vnosom parametrov v programu SAP2000).

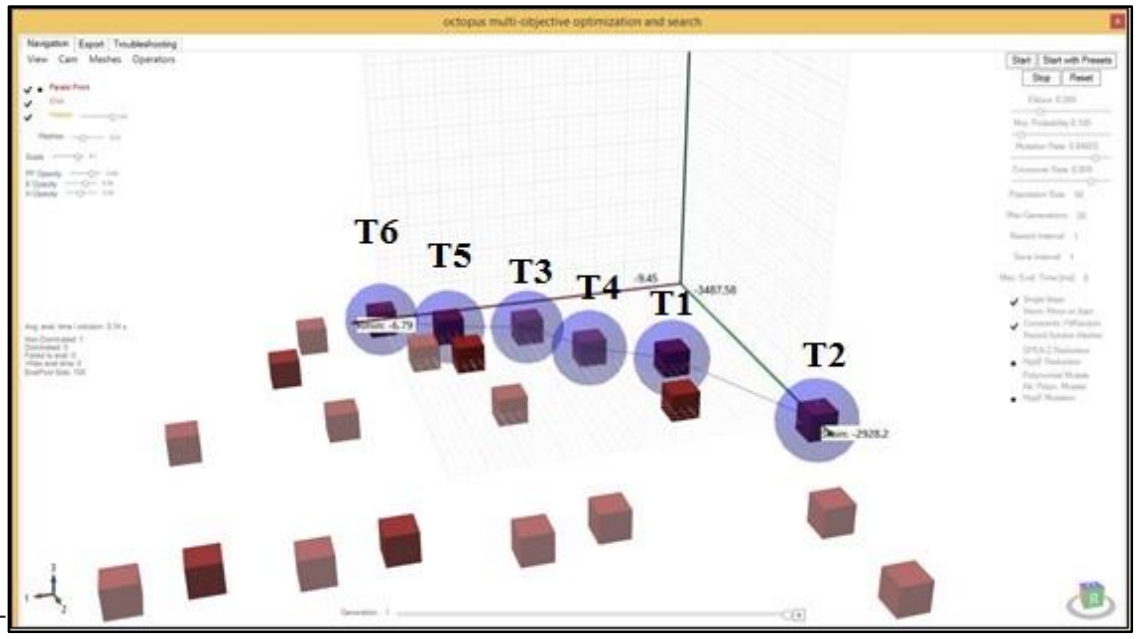
4.1.5 Optimizacija modela plošče z večkratno namensko funkcijo

Uporabljen je model plošče definiran v točki 4.1.1 z izjemo, da tokrat debelina plošče ne predstavlja osnovne spremenljivke, temveč je določena kot h = 0.15m. Obenem je obtežba tokrat točkovna, določena kot P = 50kN, njeno prijemališče (x in y koordinati) pa predstavljata osnovna parametra/spremenljivki v modelu. Ker je plošča podprta prosto ležeče in obtežba deluje v smeri globalne negativne z osi koordinatnega sistema, predvidevamo, da se lahko v primeru, ko točkovna obtežba deluje v polju (ne nad podporami), v vogalih plošče pojavijo reakcijske sile v negativni smeri z osi (kot posledica strižnih momentov na robovih plošče). Vrednosti reakcijskih sil bomo zato vključili v proces optimizacije. Obenem je za obnašanje izbranega modela relevantna tudi vrednost glavne napetosti $\sigma_2 = \sigma_2(p)$ na zgornjem robu plošče. V tem primeru oznaka p predstavlja položaj prijemališča točkovne obtežbe.



Slika 31: Prikaz rezultatov v okolju SAP2000, ki predstavljajo naši večkratni namenski funkciji. Zgoraj so prikazane glavne napetosti (»Stress S_{min} diagram – Top Face«) spodaj pa reakcijske sile na robovih plošče (»Joint Reactions«).

Namenski funkciji torej predstavljata najnižja vrednost glavne napetosti σ_2 (v SAP2000 oznaka S_{min}) na zgornjem robu plošče v celotnem polju, ter vrednost reakcijske sile v vogalih. Predznaki reakcijskih sil so po robovih plošče večinoma pozitivni (pozitivna smer globalne koordinatne *z* osi), razen v bližini vogalov (negativna smer globalne koordinatne *z* osi). Izmed vseh vrednosti reakcijskih sil (vogalnih reakcij) izberemo najmanjšo vrednost, ki predstavlja našo drugo namensko funkcijo (slika 31, namenska funkcija #2). Iščemo minimalni vrednosti glavne napetosti σ_2 in izbrane reakcijske sile z ozirom na različne položaje točkovne obremenitve plošče. Prijemališče točkovne obtežbe se nahaja samo znotraj polja plošče in ne nad podporami.



TT

Slika 32: Grafični vmesnik vtičnika Octopus, s prikazanim prostorskim modelom rešitev za primer plošče. Rešitve (točke od T1 do T6) povezane s črto predstavljajo Pareto optimum.

Tabela 4: Povzetek iskanja optimalne rešitve z uporabo večkratne namenske funkcije.

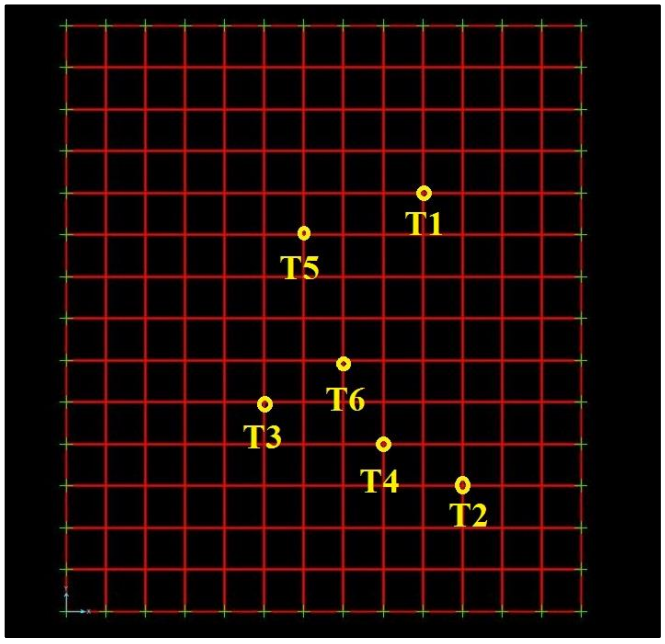
<table>
<tr><td colspan="3">KARAKTERISTIKE MODELA</td></tr>
<tr><td>Konstantne vrednosti:</td><td>Dolžina v x smeri:
Dolžina v y smeri:
Material:
Modul elastičnosti:
Poissonov količnik:
Maksimalen rob mreže končnih elementov:
Debelina plošče:
Velikost točkovne obtežbe:</td><td>6.15m
7.00m
BETON C25/30
3100 kN/cm2
0.2
0.5m
0.15m
50kN</td></tr>
<tr><td rowspan="6">Koordinate prijemališča točkovne obtežbe v globalnem koordinatnem sistemu

1.spremenljivka = x koordinata
2.spremenljivka = y koordinata</td><td colspan="2">T1 (4.26, 5.00, 0.0)</td></tr>
<tr><td colspan="2">T2 (4.73, 1.50, 0.0)</td></tr>
<tr><td colspan="2">T3 (2.37, 2.50, 0.0)</td></tr>
<tr><td colspan="2">T4 (3.79, 2.00, 0.0)</td></tr>
<tr><td colspan="2">T5 (2.84, 4.50, 0.0)</td></tr>
<tr><td colspan="2">T6 (3.31, 3.00, 0.0)</td></tr>
<tr><td rowspan="5">Namenska funkcija št.1:</td><td rowspan="5">PRIPADAJOČA MINIMALNA GLAVNA NAPETOST NA ZGORNJEM ROBU PLOŠČE</td><td>-3192.3 kN/m2</td></tr>
<tr><td>-2928.2 kN/m2</td></tr>
<tr><td>-3381.9 kN/m2</td></tr>
<tr><td>-3269.9 kN/m2</td></tr>
<tr><td>-3422.7 kN/m2</td></tr>
<tr><td>-3487.6 kN/m2</td></tr>
<tr><td rowspan="5">Namenska funkcija št.2:</td><td rowspan="5">PRIPADAJOČA MAKSIMALNA REAKCIJSKA SILA V VOGALU PLOŠČE</td><td>-8.83 kN</td></tr>
<tr><td>-9.44 kN</td></tr>
<tr><td>-7.90 kN</td></tr>
<tr><td>-8.26 kN</td></tr>
<tr><td>-7.28 kN</td></tr>
<tr><td>-6.78 kN</td></tr>
</table>

4.1.6 Komentar rezultatov optimizacije z večkratno namensko funkcijo

Rezultati so prikazani v prostorskem modelu, kjer posamezne namenske funkcije predstavljajo koordinatne osi (v primeru več kot treh namenskih funkcij so vstavljene dodatne vmesne osi). Ustreznejše rešitve so bližje koordinatnemu izhodišču, pri čemer je vsaka rešitev povezana s pripadajočimi vrednostmi namenskih funkcij, končna izbira optimalne rešitve pa je prepuščena uporabniku.

Rezultat analize sta *x in y* koordinati šestih položajev prijemališča točkovne obtežbe, ki predstavljajo nedominantne rešitve. V kolikor so to edine rešitve, ki določajo Pareto optimum, lahko sklepamo, da položaj teh šestih točk zaznamuje edina mesta, na katere lahko postavimo točkovno obtežbo, v kolikor želimo doseči (enakovredno) razmerje med minimalnima vrednostima reakcijske sil v vogalu plošče in vrednostjo glavne napetosti na zgornjem robu plošče.



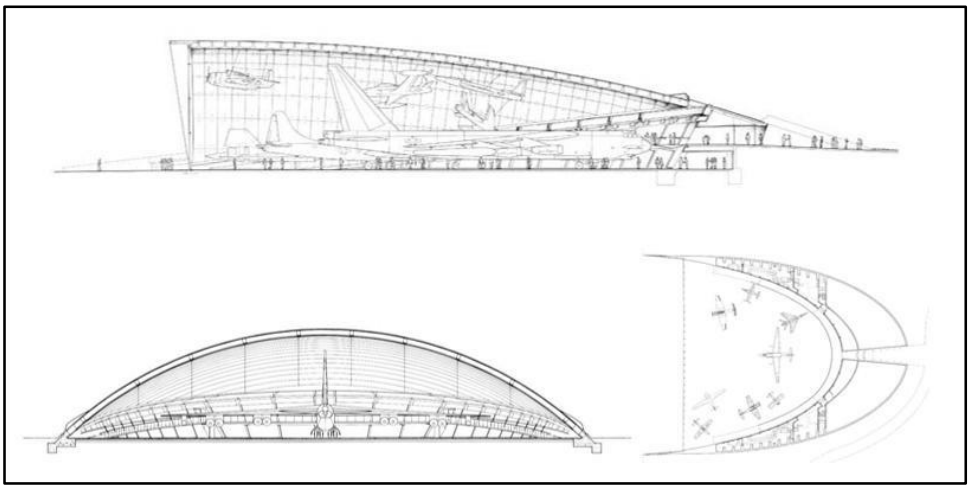
Slika 33: Položaji prijemališč (rumene točke) točkovne obtežbe kot rezultat analize problema plošče z uporabo večkratne namenske funkcije. Koordinate točk in pripadajoči namenski funkciji so podani v tabeli 4.

Na tem mestu poudarimo, da so dobljeni rezultati močno odvisni od nastavitev (število generacij, velikost populacije itd.) komponente {Octopus}, za katero veljajo splošne omejitve (glej poglavje 3.3.1) evolucijskih algoritmov. Ker podrobnejšega delovanja komponente ne poznamo, ter je v postopek izračuna vpletenih veliko programskih vtičnikov, ki so zaenkrat še razmeroma nezanesljivi, je smiselnost (ustreznost) takega rezultata potrebno preveriti še z alternativnimi metodami.

4.2 Lupinasta konstrukcija z dvojno ukrivljenostjo

Prikaz parametrične zasnove dvojno ukrivljene lupinaste konstrukcije sem izvedel na konkretnem primeru obstoječe stavbe American Air Museum v kraju Duxford, Velika Britanija. Stavbo je zasnoval arhitekturni biro Foster and Partners (leto izgradnje 1995). Stavba s svojim 90-metrskim razponom je namenjena stalni razstavi vojnih letal, ki so vpeta v samo lupinasto konstrukcijo (točkovne obremenitve). Podatki o projektu povzeti po [30].

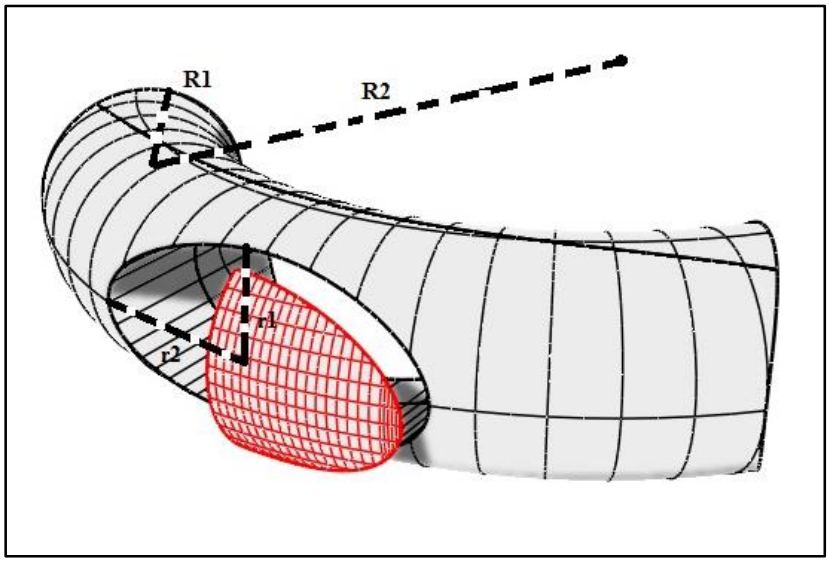
Parametrični model, zasnovan v okolju Grasshopper, ima zaradi načina podajanja algoritma določene poenostavitve oziroma se od dejanskega objekta v določenih aspektih razlikuje. Osnovna ideja oziroma vodilo je prikaz možnosti uporabe programskega okolja Rhinoceros/Grasshopper kot pripomoček pri »idejnih zasnovah« ukrivljenih konstrukcij. Ker gre za izrazito ploskovno konstrukcijo, imamo pri računu notranjih statičnih količin po metodi končnih elementov opravka s ploskovnimi končnimi elementi. Le ti se preko Grasshopper vmesnikov prenašajo v SAP2000 v obliki mrež, zato bo izbrani projekt služil predstavitvi, kreiranju in načinu obdelave ploskovnih elementov v okolju Grasshopper.



Slika 34: Ameriški letalski muzej (American Air Museum), Duxford, Velika Britanija [30].

4.2.1 Zasnova in parametrizacija geometrije konstrukcije

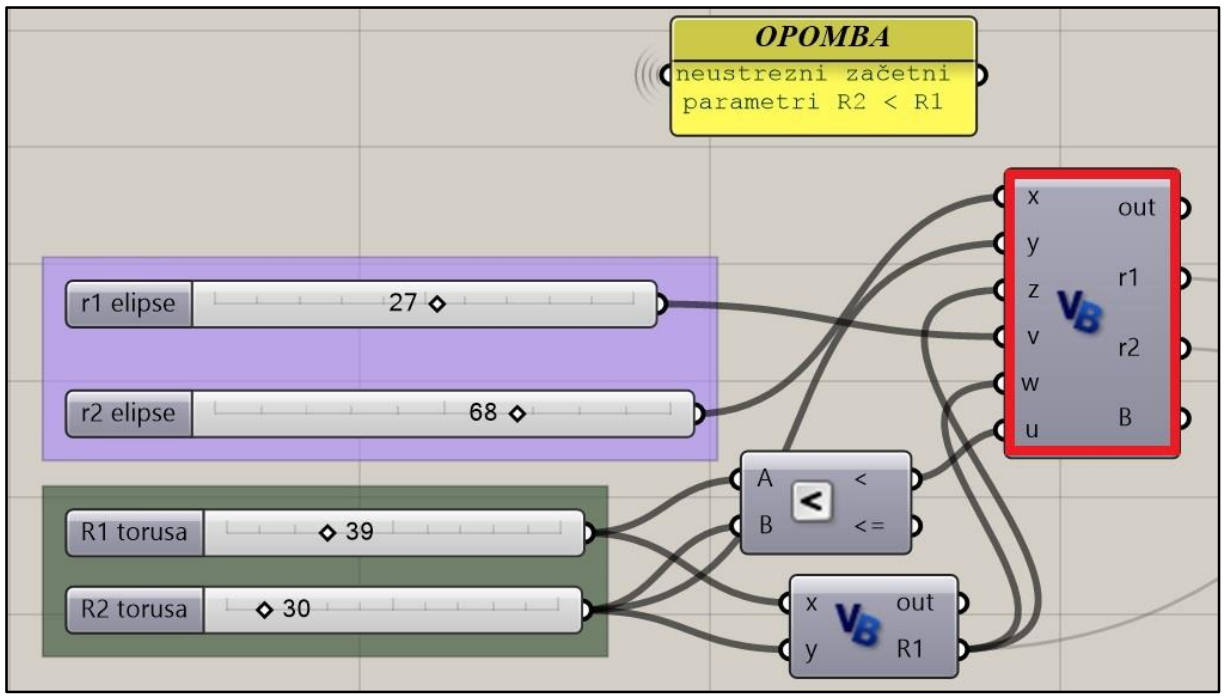
Strešno konstrukcijo predstavlja dvojno ukrivljena prefabricirana armiranobetonska lupina. Dejansko stanje se na tem mestu razlikuje, saj sem v modelu za prerez lupine predpostavil poenostavljen prerez konstantne debeline. Samo obliko lupine dobimo tako, da s pomočjo elipse iz torusa izrežemo del zunanje ploskve, ter jo prepolovimo vzdolž krajšega radija. Izrezana rdeča ploskev (slika 35) predstavlja obliko naše lupinaste konstrukcije. Začetne parametre v modelu predstavljajo izhodiščna radija samega torusa (*R1* in *R2*), ter glavna radija elipse (*r1* in *r2*).



Slika 35: Osnova za izhodiščno obliko lupinaste konstrukcije. Oznaki R1 in R2 predstavljata osnovna radija za določitev oblike torusa, oznaki r1 in r2 pa predstavljata glavna radija elipse, s katero izrežemo končno obliko.

Prav tako kot v primeru modela armiranobetonske plošče iz točke 4.1, tudi tokrat celotno konstrukcijo najprej določimo z osnovnimi geometrijskimi liki. Čim enostavnejša zasnova pomembno vpliva na morebitno kompleksnost celotnega algoritma in s tem povezanim časom izrisa modela. Pri definiranju geometrijskih likov, torusa in elipse, na začetku uvedemo določene omejitve vrednosti glavnih parametrov. Te omejitve so potrebne zaradi pravilnega delovanja določenih komponent v nadaljevanju algoritma.

Grasshopper omogoča tudi implementacijo kode spisane v drugih programskih jezikih, npr. Visual Basic.NET, Python, C# itd., kar mestoma lahko poenostavi samo modeliranje. Namesto vstavljanja in povezovanja klasičnih Grasshopper komponent (matematičnih operatorjev <, > itd.) vstavimo Visual basic komponento. Znotraj le–te definiramo nastopajoče neznanke, ter njihovo medsebojno povezanost. Neznanke lahko predstavljajo objekte iz okolja Rhinoceros/Grasshopper (geometrijska telesa itd.) oziroma se navezujejo na že predhodno definirane parametre/komponente v Grasshopper skripti. Povezava same komponente z osnovnimi drsniki, s katerimi določamo velikosti parametrov je prikazana na sliki 36. Vsebina komponente (na sliki 36 označena z rdečo obrobo) je prikazana v tabeli 5.

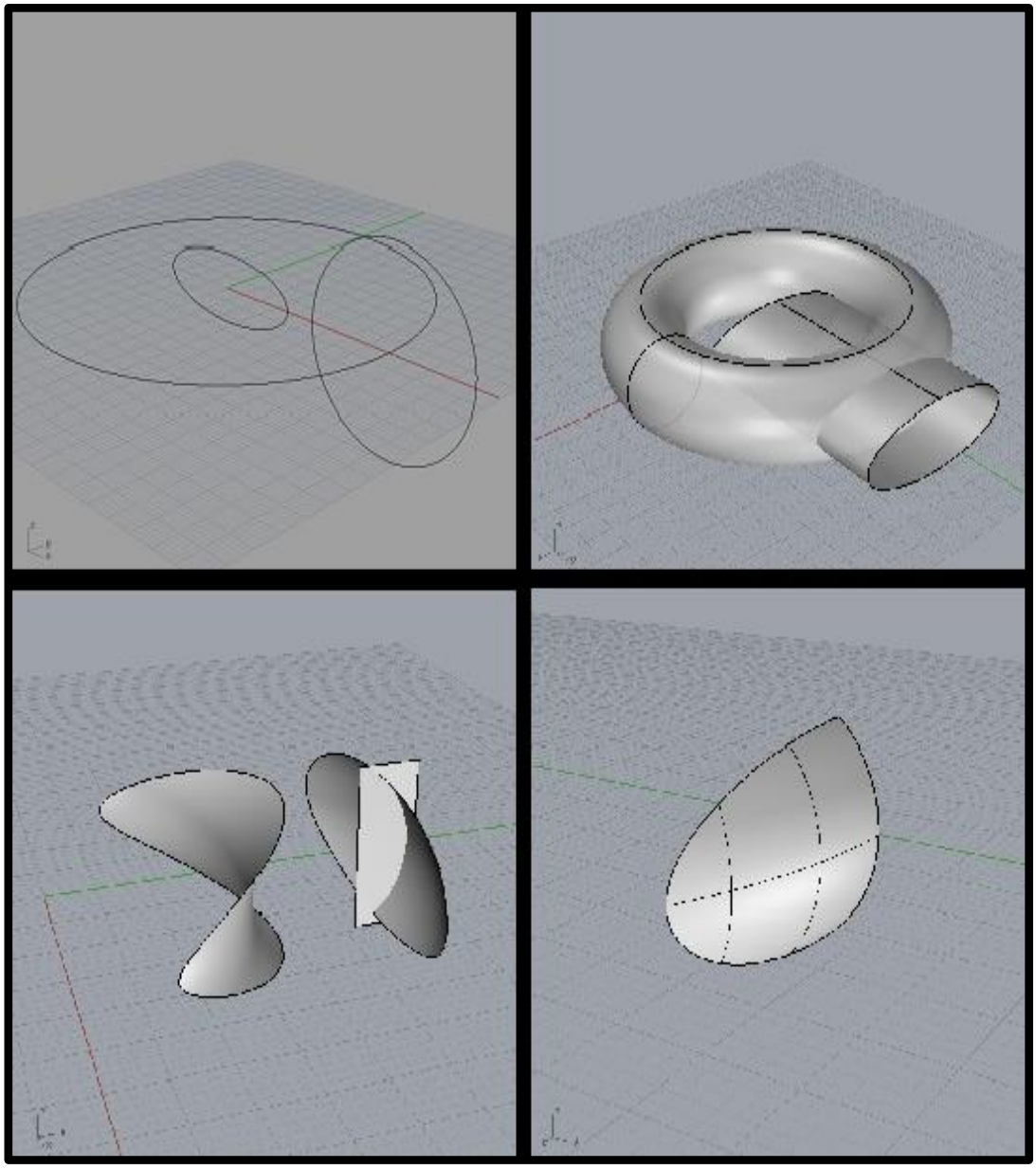


Slika 36: Povezava drsnikov s komponentami, ki vključujejo kodo spisano v programskem jeziku Visual Basic.

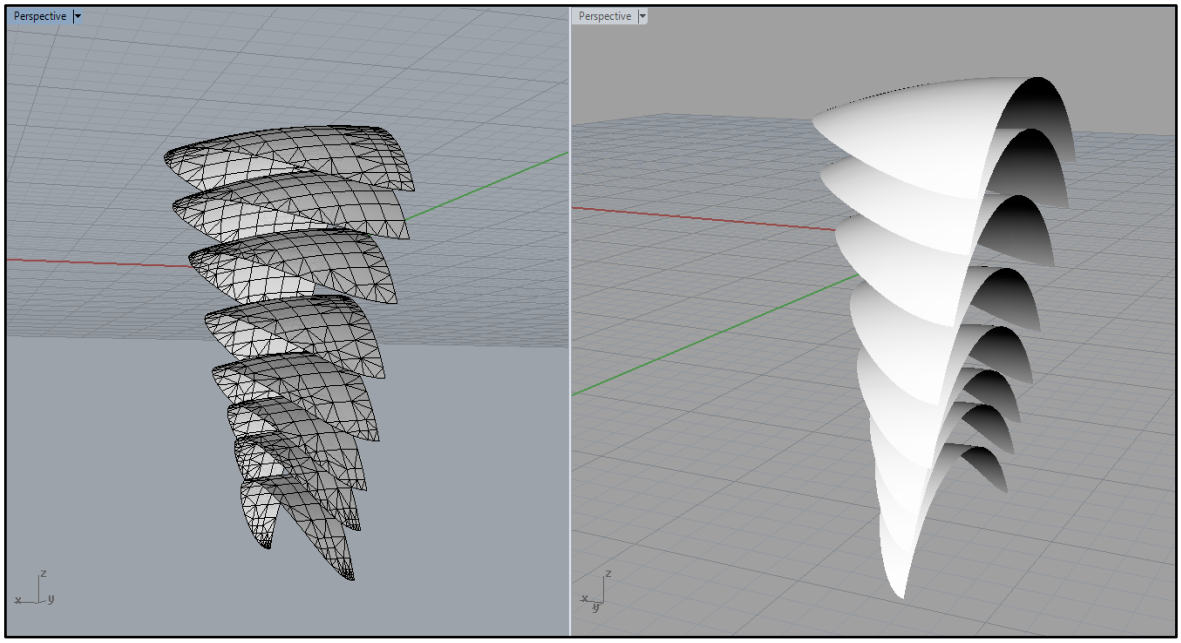
Tabela 5: Implementacija Visual basic skripte v Grasshopper algoritem. Podana skripta določa medsebojne omejitve geometrijskih karakteristik začetnih elementov. V primeru, da so izbrani parametri geometrijsko nekompatibilni, skripta vrne opozorilo.

```
Public Class Script_Instance

  Inherits GH_ScriptInstance

Private Sub

RunScript(ByVal x As Object, ByVal y As Object, ByVal z As Object, ByVal v As Object, ByVal w As Object, ByVal u
As Boolean, ByRef r1 As Object, ByRef r2 As Object, ByRef B As Object)

    If v < w Then r1 = v
    If v >= w Then r1 = w - 1
    If z > y Then z = y
    If x < y - z Then r2 = x
    If x >= y - z Then r2 = y - z - 1
    If u = False Then B = "neustrezni začetni parametri R2 < R1"
 End Sub

End Class
```

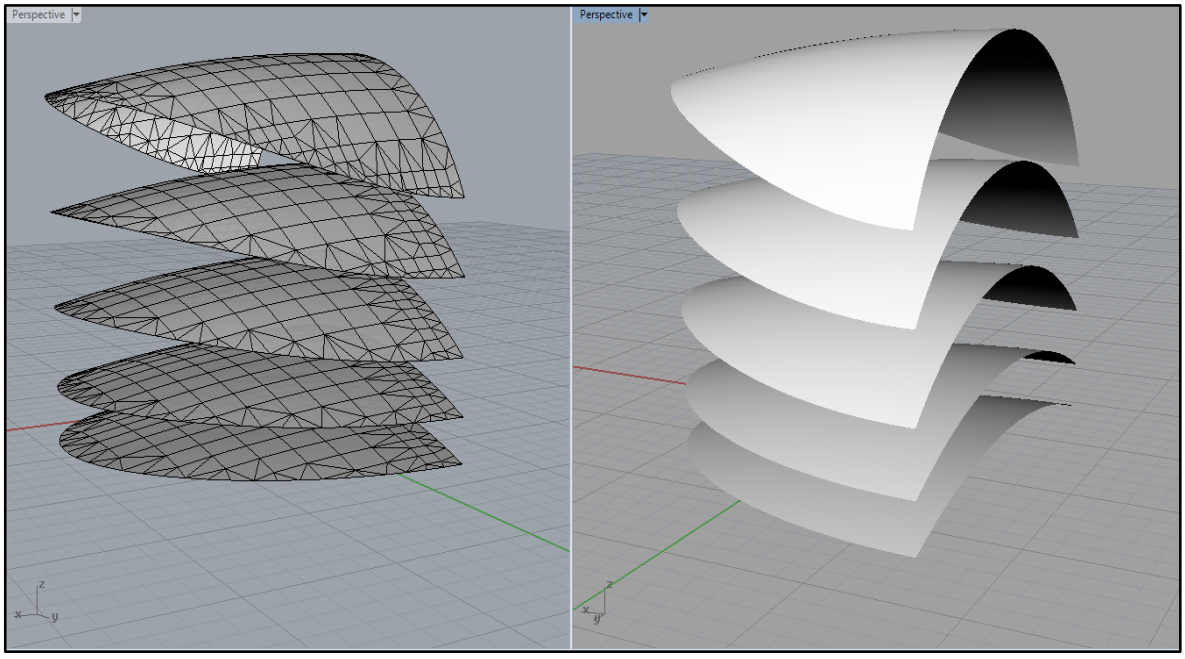
Do končne oblike, ki predstavlja našo lupinasto konstrukcijo, pridemo nato s kombinacijo različnih komponent. Posamezni odseki celotnega algoritma so prikazani v prilogi B, povzetek, ki predstavlja vizualizacijo procesa pa predstavlja slika 37.



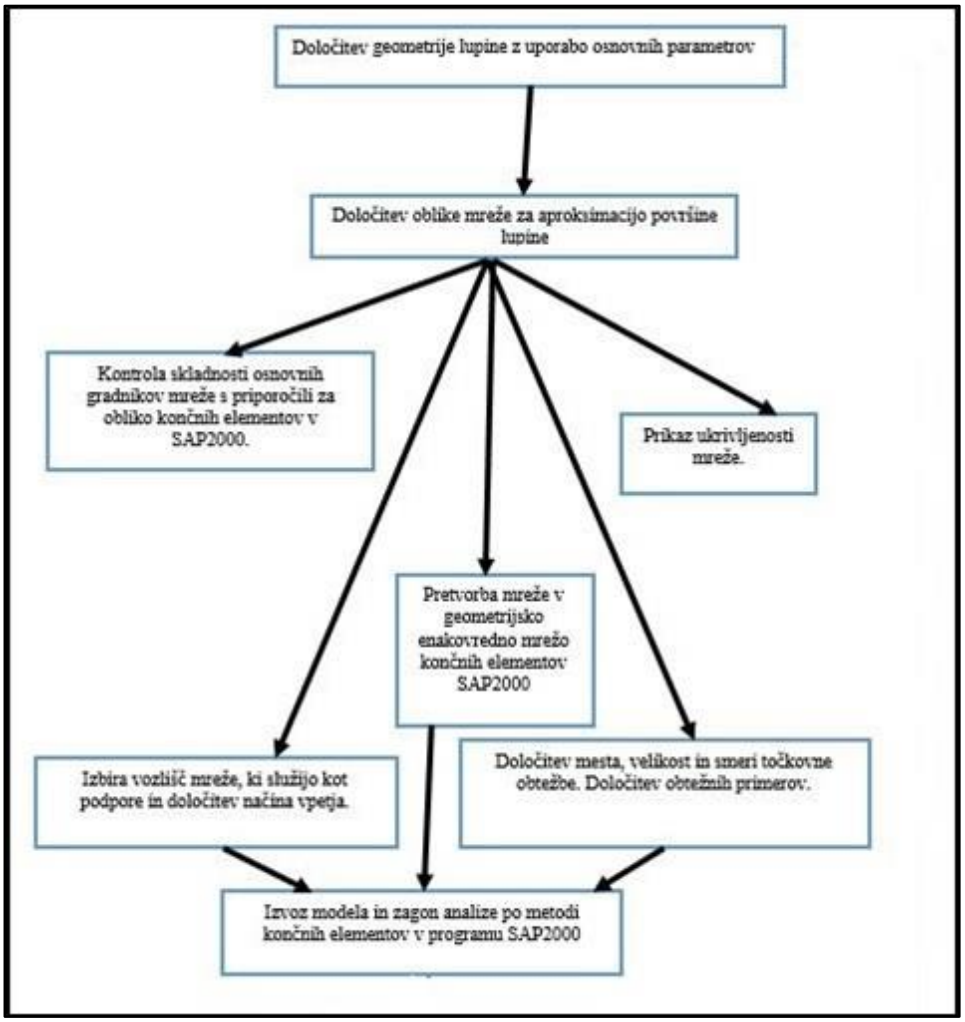
Slika 37: Vizualni prikaz izbire oblike lupinaste konstrukcije po korakih. Kvadrat levo zgoraj vsebuje osnovne parametre, ki jih v algoritmu spreminjamo (elipsa in torus). Desno zgoraj je prikazan celoten torus, ter ekstrudirana elipsa. Levo spodaj so ploskve, ki jih iz torusa izreže ekstrudirana elipsa.



Slika 38: Naključen nabor različnih oblik, ki jih zavzame model lupine pri spreminjanju osnovnega parametra R2.



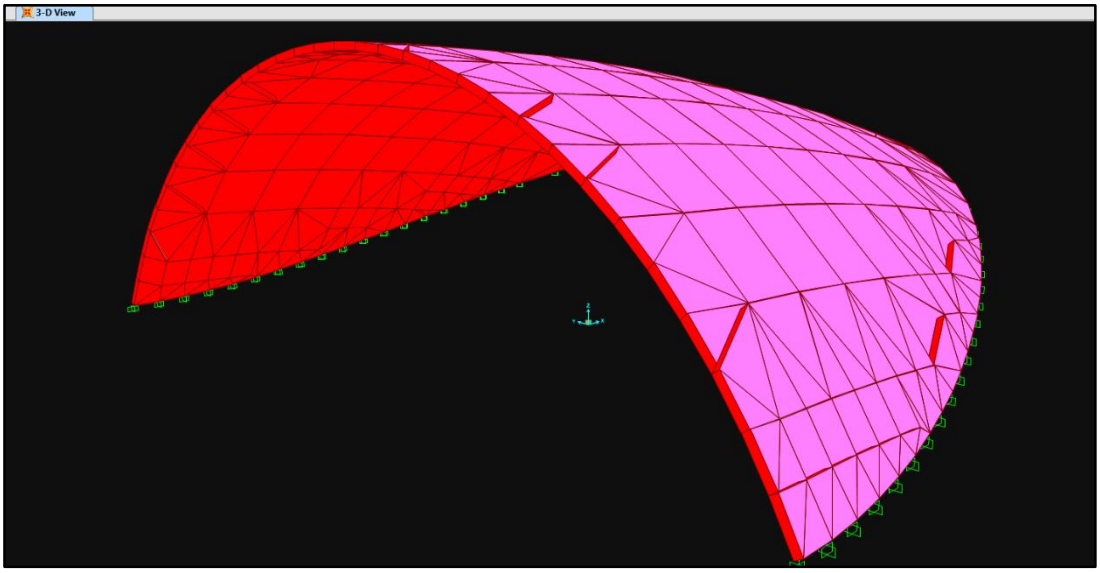
Slika 39: Naključen nabor različnih oblik, ki jih zavzame model lupine pri spreminjanju osnovnega parametra R1.



Slika 40: Diagram poteka algoritma za model lupinaste konstrukcije (hangarja).

4.2.2 Obtežba in podpiranje konstrukcije

Lupina je po obodu podprta s stebri, ki pa jih zaradi poenostavitve modela in izrazito ploskovne narave konstrukcije nadomestimo z linijsko podporo vzdolž celotne stranice lupine, ki je v stiku s tlemi. Model, prenesen v okolje SAP2000, je na vsakem stičišču mreže končnih elementov podprt z podporo, ki preprečuje pomike in rotacije v smereh *x*, *y* in *z* (polno vpetje).

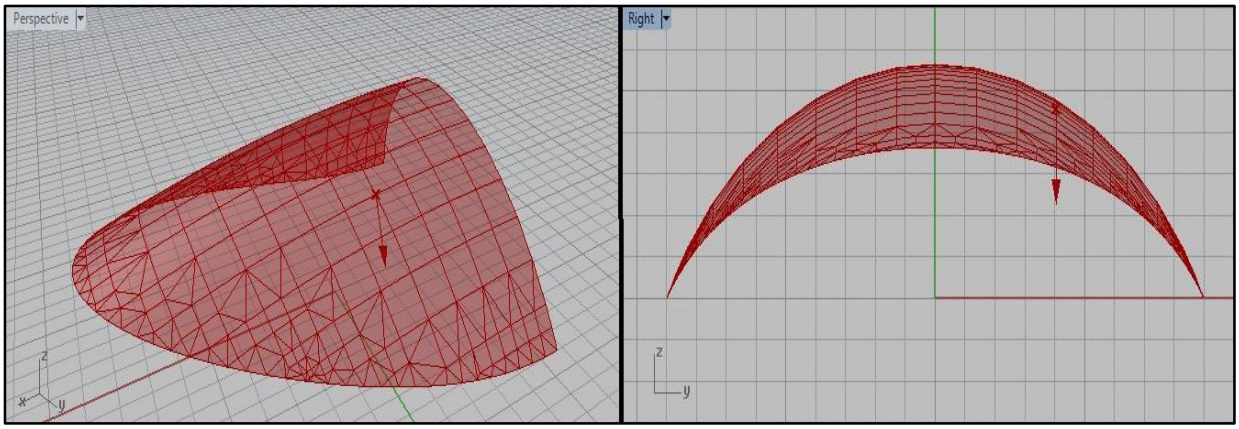


Slika 41: Prenešen model polno vpete lupine v programu SAP2000.

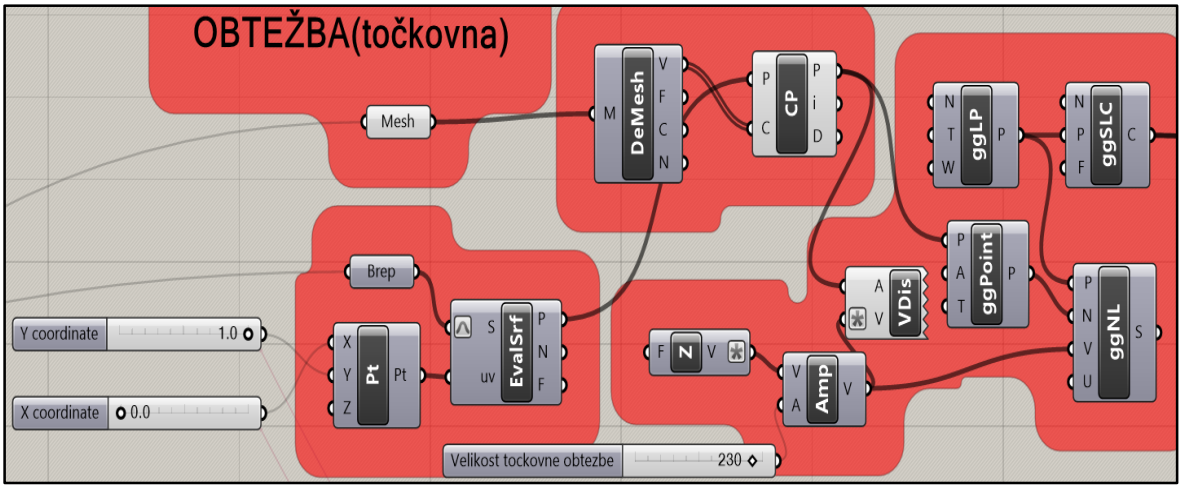
V modelu zaradi poenostavitve nastopata samo lastna teža konstrukcije ter točkovne obtežbe, ki predstavljajo težo razstavnih eksponatov (letal) pripetih na strop. Položaj točkovnih obtežb določimo parametrično, saj se vrednosti notranjih statičnih količin občutno spreminjajo glede na relativno mesto točkovnih obtežb.

V primeru modeliranja ploskovnih konstrukcij, ki se v izbrani program za računanje konstrukcij uvozijo z uporabo mrež, lahko podajanje točkovnih obremenitev izvedemo na dva načina. V obeh primerih je potrebno upoštevati pogoj, da se mesto prijemališča točkovne obtežbe ujema z enim od vozlišč mreže. Ta pogoj je posledica definicije končnih elementov (uporabljen element tipa »shell – thin«), za katere je obtežba določena samo v vozliščih končnih elementov. Ujemanje prijemališča obtežbe in vozlišča mreže lahko podamo sledeče:

1. Prijemališče sile postavimo v enega izmed obstoječih vozlišč vnaprej definirane mreže v okolju Grasshopper. Obliko in velikost mreže določimo že v predhodnih korakih, kar vpliva na sam razpored vozlišč in njihovo medsebojno razdaljo. Posledično to seveda pomeni, da je mesto prijemališča obtežbe vezano na samo mrežo in ni povsem poljubno (slika 42). Velikost in smer obtežbe določimo z Grasshopper komponentami v vektorski obliki (podamo prijemališče, smer in velikost vektorja).



Slika 42: Mreža s prijemališčem točkovne obtežbe (vektor v negativni smeri z osi koordinatnega sistema) v »obstoječem« vozlišču.



Slika 43: Odsek algoritma, ki določa prijemališče točkovne obtežbe v »obstoječem« vozlišču mreže ploskovnih elementov.

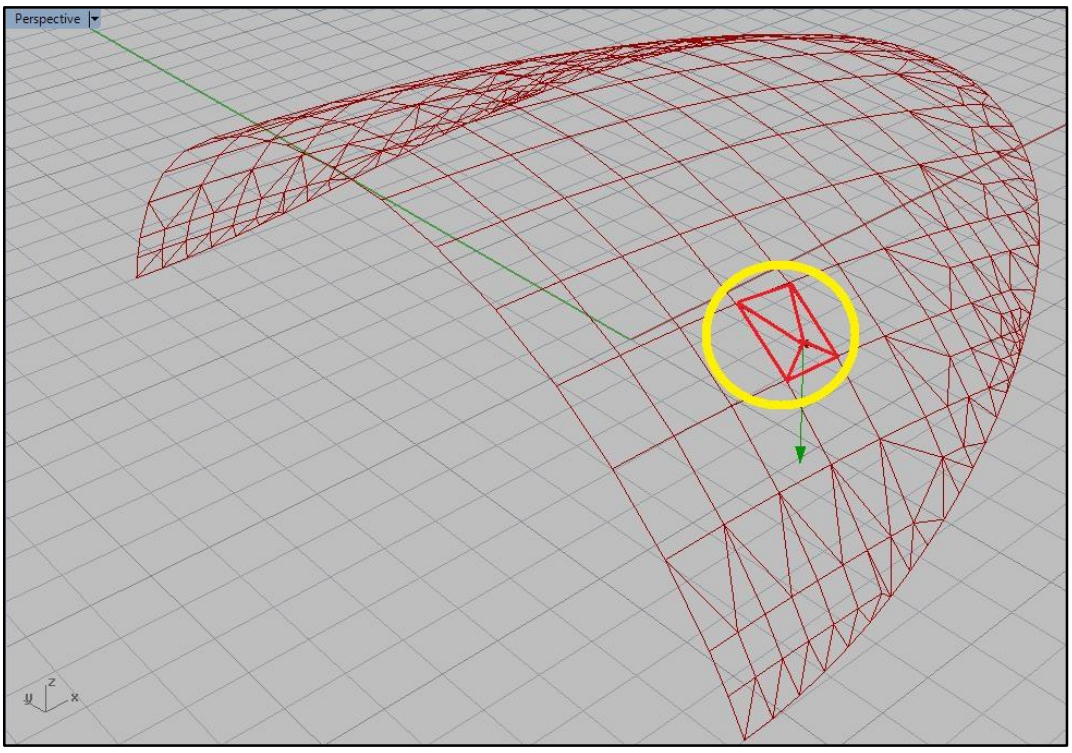
Prijemališče točkovne obtežbe definiramo v obliki točke, ki jo parametrično premikamo po reparametrizirani ploskvi modela lupine. Vsaka ploskev v okolju Rhinoceros je namreč (glej poglavje NURBS) definirana s parametroma u in v, ki sta določena z domenama, omejenima z mejnima vrednostima. V ukazni vrstici okolja Rhinoceros lahko vsak trenutek tudi preverimo (s klikom na površino in ukazom »What«), kakšni sta domeni parametrov u in v, ki določata trenutno površino modela lupine.

Z ukazoma {EvalSrf} in {»Reparametrize«} v Grasshopperju (slika 43) površini dodelimo novi enotski domeni (tj. u in v zavzameta vrednosti med 0 in 1). Točko {Point}, ki je definirana z drsnikoma {X coordinate} in {Y coordinate}, ter povezana z {EvalSrf} sedaj lahko premikamo po celotni površini lupine. S tem lahko (grafično) izberemo naš relativni položaj točkovne obtežbe. Ker pa ima mreža ploskovnih elementov drugačne koordinate vozlišč kot naša izbrana

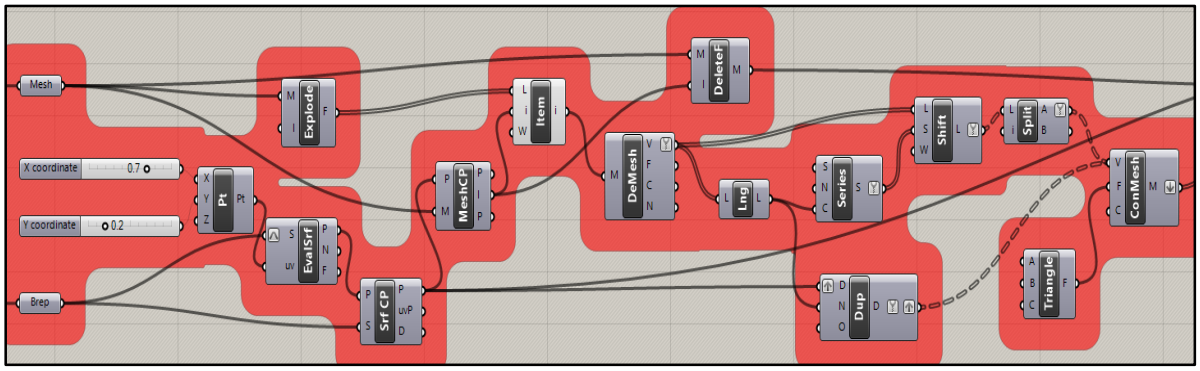
točka (mreža je namreč zgolj aproksimacija površine z ravnimi ploskvami), je potrebno sedaj točko iz površine lupine prestaviti v najbližje vozlišče mreže. Ukaz {DeMesh} razgradi mrežo na elementarne gradnike (vozlišča, osnovne ploskve itd.) ter urejen seznam koordinat vozlišč posreduje naprej. Komponento {CP} uporabimo za določitev najbližjega vozlišča mreže z ozirom na naš relativni položaj točkovne obtežbe.

Velikost in smer obtežbe zopet določata komponenti {Amp} in {Z}, ki obtežbo definirata v vektorski obliki. Ostale komponente služijo za prenos podatkov v SAP2000 preko vmesnika GeometryGym.

2. Prijemališče točkovne sile prestavimo na poljubno mesto, ter ustrezno spremenimo/razdrobimo mrežo ploskovnih elementov. V določenih primerih se pri modeliranju ploskovnih konstrukcij ta korak v programih za analizo konstrukcij izvede samodejno. Na mestu prijemališča točkovne sile se najbližja vozlišča ter ploskev mreže (končnih) elementov spremeni/razdrobi na način, ki ustreza definiciji MKE (prijemališče v vozlišču). Za razliko od prvega načina v tem primeru ne spreminjamo položaja točkovne obtežbe, temveč samo mrežo ploskovnih elementov.

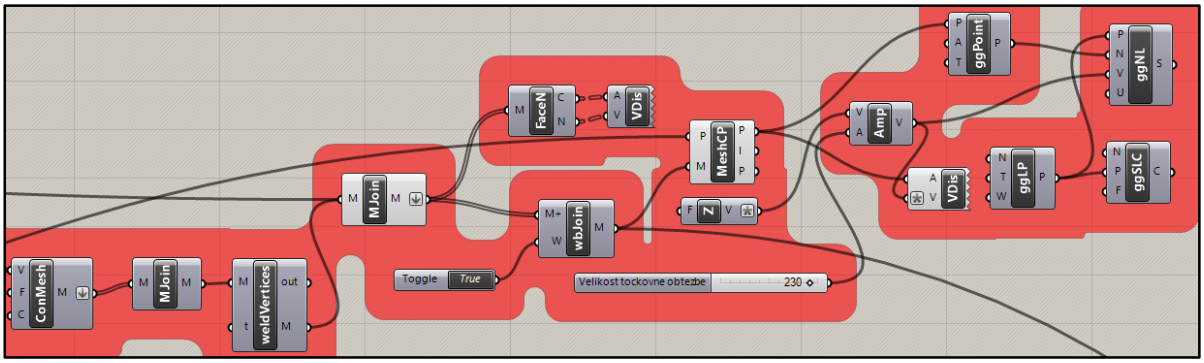


Slika 44: Drobitev mreže ploskovnih elementov na mestu prijemališča točkovne obtežbe.



Slika 45: Prvi del odseka algoritma za drobitev mreže na mestu prijemališča točkovne obtežbe.

.



Slika 46: Drugi del odseka algoritma za drobitev mreže na mestu prijemališča točkovne obtežbe (prva komponenta {ConMesh} je prikazana tudi na sliki 48).

Postopek je, v začetnih korakih, podoben kot v prvem primeru, le da tokrat posamezen odsek (ploskev) mreže, ki je najbližje izbranemu mestu prijemališča točkovne sile, odstranimo iz celotne mreže {DeleteF}. Na mestu prijemališča ustvarimo novo vozlišče, ga dupliciramo *N*-krat (kjer *N* predstavlja število vozlišč izbrisane ploskve) ter ga s kopijami vozlišč izbrisane ploskve povežemo v novo mrežo. Novonastale ploskve, ki si delijo položaje posameznih vozlišč, združimo z {MJoin} in »zvarimo« podvojena vozlišča {weldVertices}, tako da dobimo eno samo mrežo, z enim od vozlišč na mestu prijemališča točkovne sile. Preostalo mrežo in novonastali manjkajoči delu združimo v celoto z enakim postopkom {wbJoin}. Definiramo še velikost in smer (vektorske) obtežbe ter dodamo komponente za prenos vseh elementov v SAP2000.

Podajanje točkovne obtežbe je pri ploskovnih elementih torej močno povezano s samo mrežo (končnih) elementov. Ker gre za parametričen model konstrukcije, je potrebno del algoritma, ki obravnava točkovno obtežbo nastaviti tako, da je aplikativen neodvisno od izbranih parametrov in posledično mreže elementov. Za delujoč model je torej potrebna tudi čim enostavnejša in smiselno urejena mreža (končnih) elementov.

4.2.3 Modeliranje mreže ploskovnih elementov

Za kreiranje in modifikacijo mrež je v okolju Grasshopper na voljo veliko število dodatkov, saj je proces aproksimacije površine z uporabo mrež izredno pomemben tudi na drugih področjih (aeronavtika, biomehanika, avtomobilska industrija itd.). Dejstvo, da se mreža ustvarjena v Grasshopper, direktno prenese v SAP2000 kot mreža končnih elementov, le še dodatno povečuje smiselnost uporabe Grasshopper za parametrično zasnovo in modeliranje ukrivljenih konstrukcij. Vnašanje in spreminjanje mreže končnih elementov v okolju SAP2000 je namreč v primerjavi z možnostmi, ki jih ponuja Grasshopper dokaj robustno. Na samo število in obliko končnih elementov lahko podrobno vplivamo že z uporabo komponent, ki jih kot del osnovnega nabora nudi Grasshopper. Grasshopper namreč že v osnovi nudi 3 načine za aproksimacijo poljubnih površin z uporabo mrež [31]:

- {Simple Mesh}

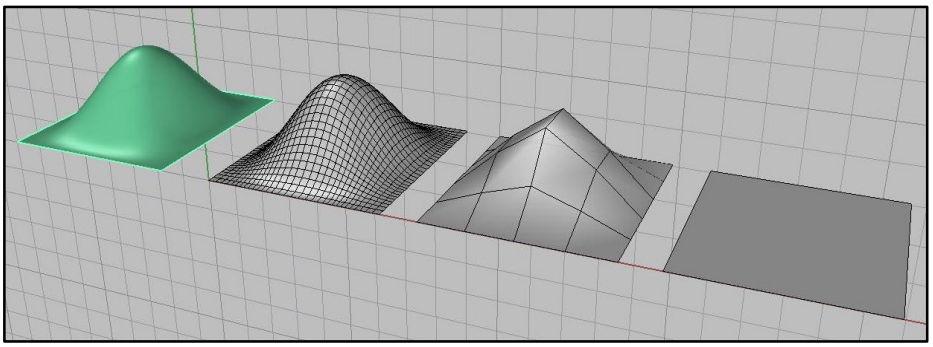
  Predstavlja najenostavnejšo aproksimacijo površin z uporabo ene same tri ali štirivozliščne ploskve, praktično brez ozira na samo točnost aproksimacije. Uporabna zgolj v najpreprostejših primerih ravnih površin. V primeru, da osnovna površina ni »enostavna«, bo za aproksimacijo uporabljeno privzeto orodje iz Rhinoceros,

- {Mesh Surface}

  Poizkus aproksimacije površine s pravokotno mrežo sestavljeno iz poljubnega števila ploskev. Določimo lahko točno število ploskev v obeh smereh pravokotne mreže, ter uporabimo (približno) izenačitev razpona posameznih ploskev.

- {Mesh Brep}

  Uporablja privzeto orodje za izdelavo mrež iz osnovnega okolja Rhinoceros. Omogoča razmeroma natančne nastavitve, ki vplivajo na končno podobo mreže, kot npr. maksimalna deviacija mreže od aproksimirane površine, mejni medsebojni kot, ki ga lahko oklepajo normale posameznih sosednjih ploskev, dolžine in razmerja stranic osnovnih ploskev itd.



Slika 47: Prikaz uporabe različnih komponent za aproksimacijo poljubne površine (zelena barva). Z desne proti levi si sledijo {Simple mesh}, {Mesh surface} in {Mesh Brep}.

4.2.4 Prikaz Gaussove in povprečne ukrivljenosti ploskve

Definicijo ukrivljenosti ploskve povzamemo iz diferencialne geometrije [32]. V poljubni točki na ploskvi obstaja množica krivulj, ki so presek neke normalne ravnine in ploskve. Od teh krivulj predstavljata tisti z največjo in najmanjšo ukrivljenostjo, glavni ukrivljenosti (κ_1 in κ_2), ki sta definirani kot recipročna vrednost polmera pritisnjene krožnice (R_1 in R_2).

$$R_1 = \frac{1}{\kappa_1} \tag{13}$$

$$R_2 = \frac{1}{\kappa_2} \tag{14}$$

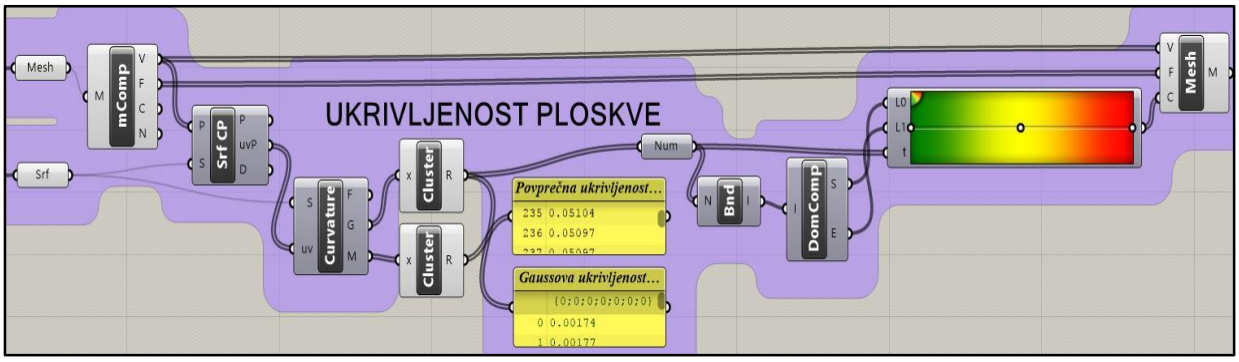
Gaussova ukrivljenost (K) je podana kot:

$$K = \kappa_1 \kappa_2 \tag{15}$$

Povprečna ukrivljenost (H) pa je definirana kot:

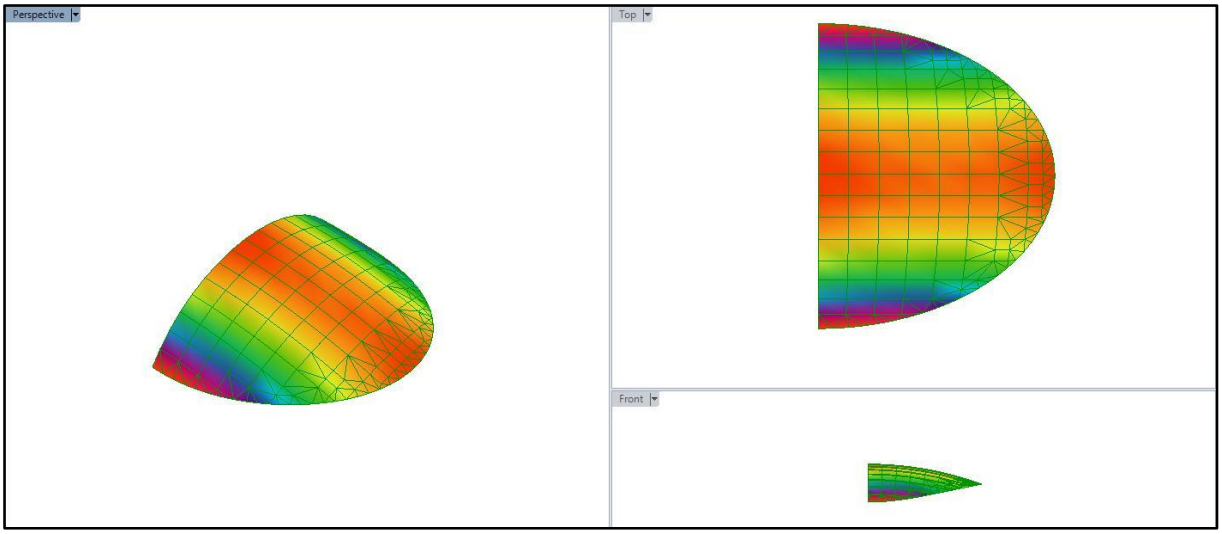
$$H = \frac{1}{2}(\kappa_1 + \kappa_2) \tag{16}$$

V okolju Grasshopper vpeljemo parametra *K* in *H* z uporabo naslednjih komponent (slika 48).

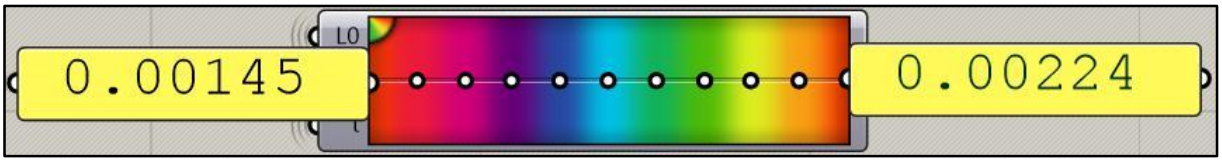


Slika 48: Algoritem za prikaz povprečne in Gaussove ukrivljenosti ploskve.

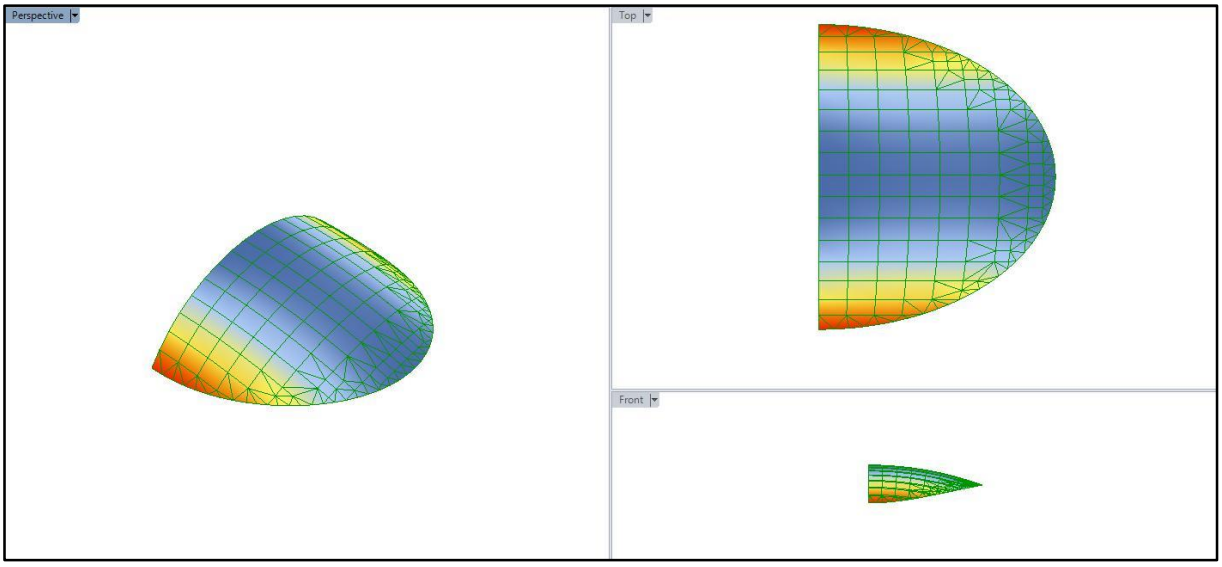
Vhodne podatke za določitev ukrivljenosti predstavljata model ploskve {Srf} in njena aproksimacija z mrežo {Mesh}. Za vsako točko vozlišča mreže najprej poiščemo najbližjo točko na površini ploskve {SrfCP}. S koordinatami teh točk v parametrični obliki (koordinati *u* in *v*) nato komponenta {Curvature} izvrednoti vrednosti Gaussove in povprečne ukrivljenosti. Za lažjo grafično predstavo priredimo barvno skalo, omejeno z relativnimi vrednostmi najmanjše in največje dobljene ukrivljenosti, ter jo apliciramo na obstoječo mrežo (slike 49, 50, 51 in 52).



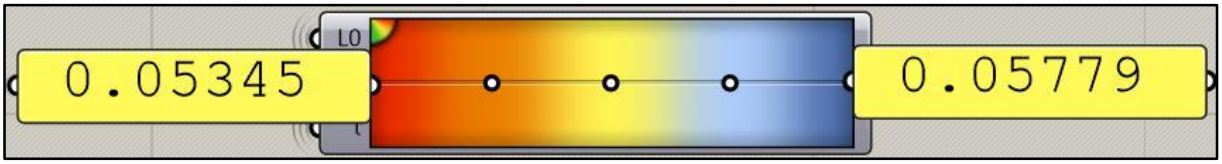
Slika 49: Prikaz spreminjanja Gaussove ukrivljenosti za poljubno izbran model lupine.



Slika 50: Barvna skala za primer lupine na sliki 49. Barvni razdelki predstavljajo 10% celotne domene zavzetih vrednosti omejene s spodnjo (0.00145) in zgornjo vrednostjo (0.00224).



Slika 51: Prikaz spreminjanja povprečne ukrivljenosti za poljubno izbran model lupine.



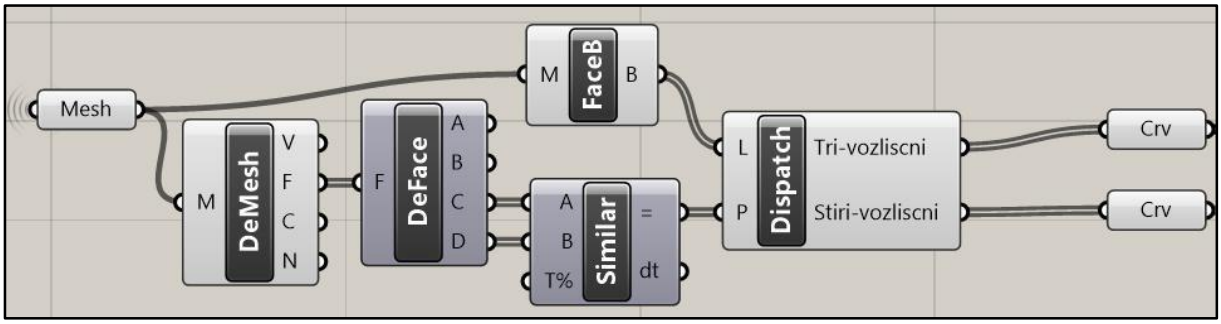
Slika 52: Barvna skala za primer lupine na sliki 51. Barvni razdelki predstavljajo 25% celotne domene zavzetih vrednosti omejene s spodnjo (0.05345) in zgornjo vrednostjo (0.05779).

4.2.5 Drobitev mreže ploskovnih elementov

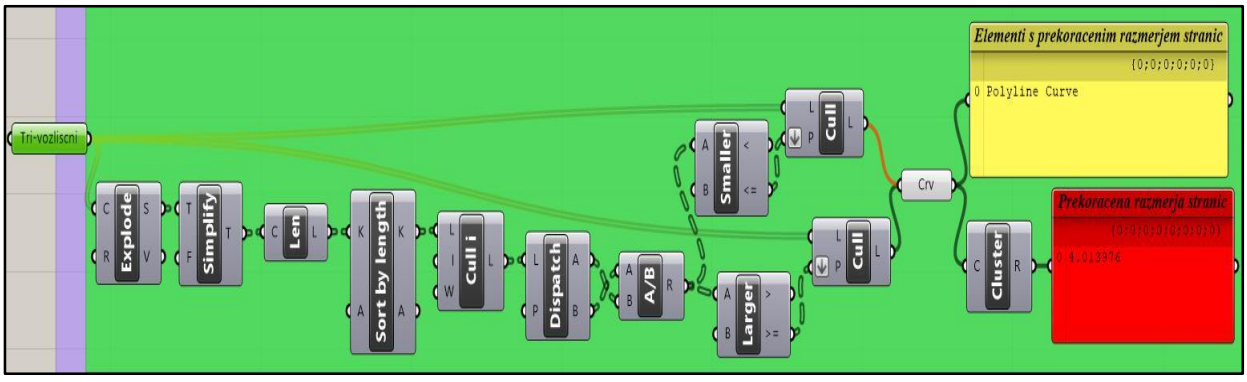
Metoda končnih elementov (MKE) je numerična metoda, ki temelji na razdelitvi (mreženju, diskretizaciji) obravnavane zvezne domene na določeno število medsebojno povezanih poddomen enostavnih geometrijskih oblik, ki jih imenujemo končni elementi. Natančnost numerične rešitve je običajno sorazmerna s številom končnih elementov oziroma stopnjo interpolacijskih funkcij. Na rezultate analize po metodi končnih elementov vpliva tudi izbira tipa in oblike uporabljenih končnih elementov.

Dvojno ukrivljeno lupino modeliramo samo z uporabo končnih elementov tipa »shell«, za katere so v programu SAP2000 izdana sledeča priporočila [33] oziroma smernice:

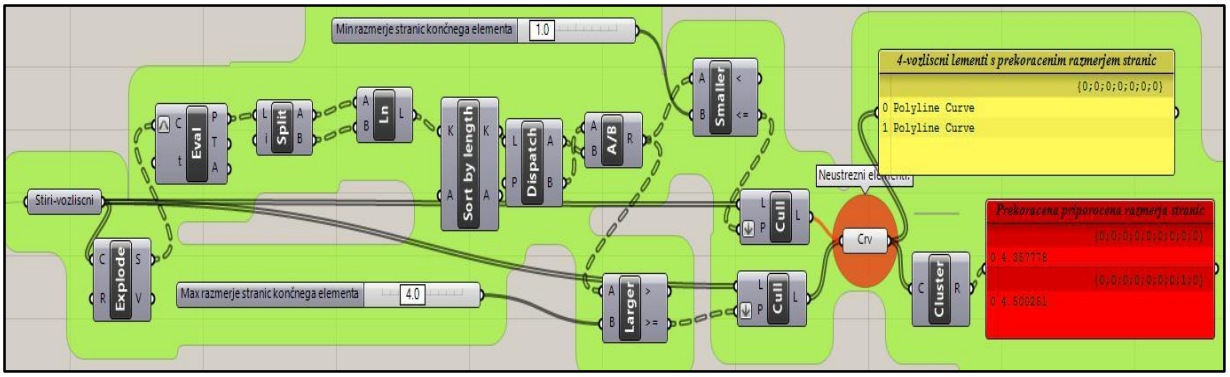
1. Razmerja stranic končnih elementov naj bodo čimbolj enaka (razmerje med 1 in 4). Za trivozliščne elemente razmerje stranic predstavlja dolžino najdaljše stranice, deljeno z dolžino najkrajše stranice (slika 53, 54 in 55). Razmerje stranic štirivozliščnih elementov, pa je določeno kot razmerje simetral parov stranic.



Slika 53: Razdelitev mreže na posamezne ploskve glede na število vozlišč.



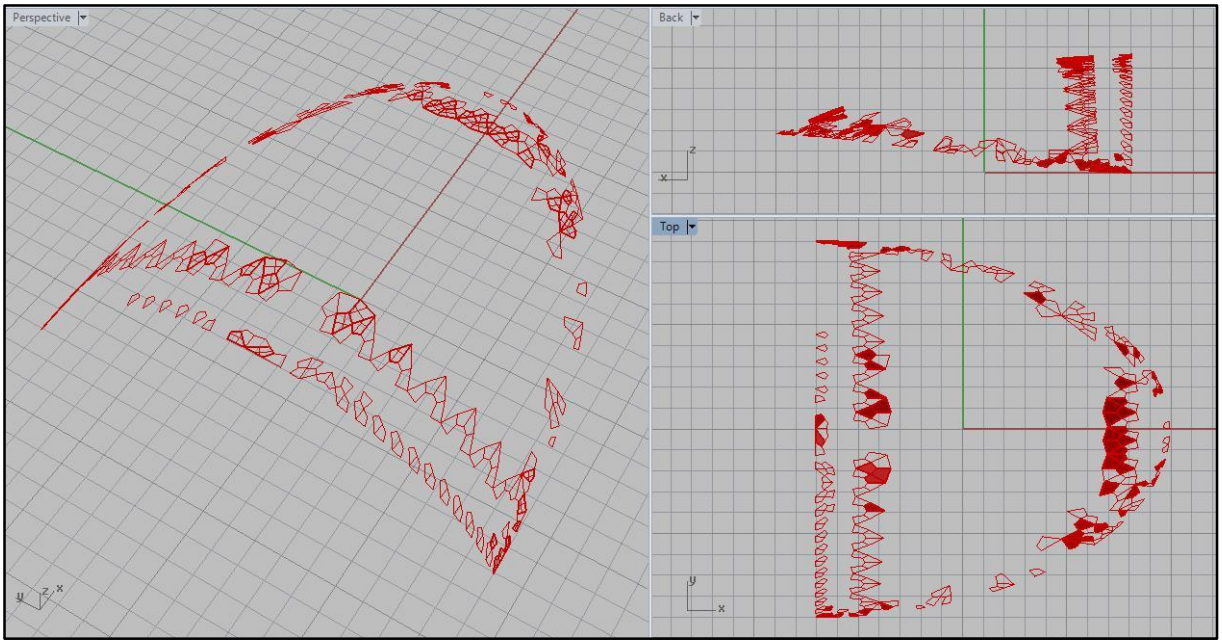
Slika 54: Kontrola razmerja dolžin stranic tri-vozliščnih elementov mreže.



Slika 55: Kontrola razmerja dolžin stranic štiri-vozliščnih elementov. Z nastavitvijo vrednost drsnikov lahko spreminjamo sprejemljivo mejo razmerja stranic posameznega ploskovnega elementa.

2. Položaji vozlišč posameznega štirivozliščnega končnega elementa naj bodo izbrani tako, da bo velikost notranjega kota zavzela vrednosti med 45° in 135°.
3. Za štirivozliščne elemente je dopustno majhno odstopanje vozlišč od srednje ravnine končnega elementa. Priporočena vrednost kota med normalami definiranimi v vozliščih je manjša od 30°.

Podane usmeritve lahko uporabimo v algoritmu za kontrolo mreže ploskovnih elementov, preden jih izvozimo v SAP2000 kot mrežo končnih elementov.



Slika 56: Grafični prikaz ploskev/elementov mreže, ki ne ustrezajo priporočenim geometrijskim karakteristikam podanim v priročniku SAP2000 za končne elemente tipa »shell«. Iz slike je razvidno da se »problematični« elementi pojavljajo v okolici ukrivljenih robov lupine.

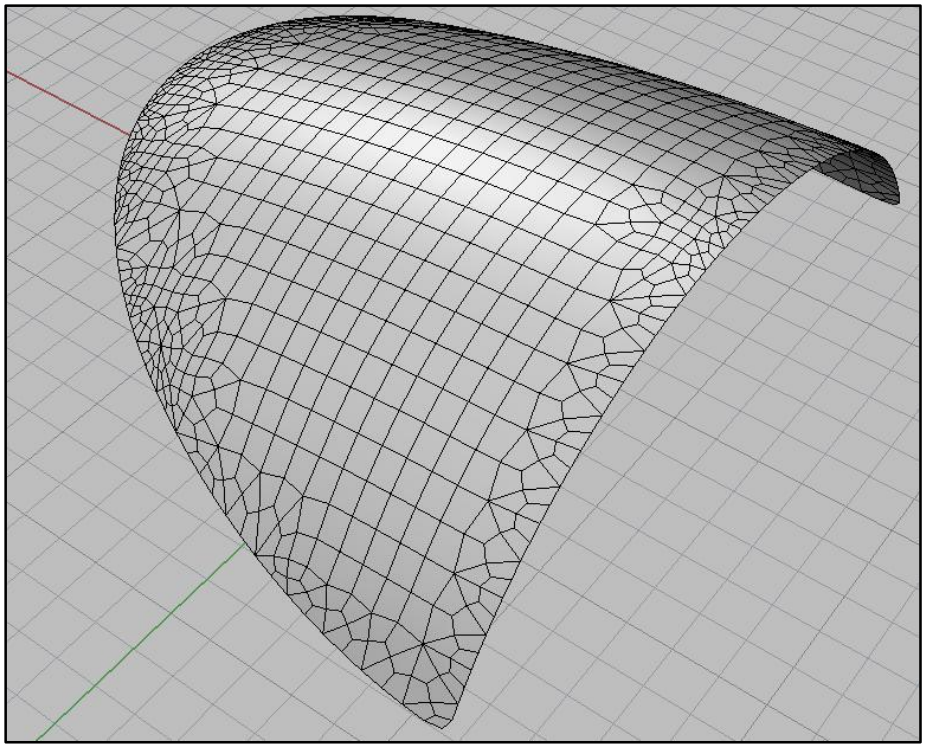
Neustrezne elemente lahko z nadaljnjo drobitvijo mreže priredimo tako da ustrezajo priporočilom.

Poleg geometrijskih omejitev je v priročniku podano tudi splošno priporočilo. Štirivozliščni končni elementi se v splošnem obnašajo bolje kot trivozliščni. Uporaba slednjih se priporoča zgolj za mesta, kjer ne nastopajo velike spremembe napetosti.

Pri zasnovi mreže z uporabo končnega elementa tipa »shell« stremimo k čim bolj urejeni mreži sestavljeni iz pretežno štirivozliščnih končnih elementov, v kolikor je to le mogoče.

Za obdelavo mrež je v okolju Grasshopper na voljo dodatek Weaverbird, ki razširi nabor komponent relevantnih za samo modeliranje oziroma drobitev mreže. V nasprotju s prikazanim postopkom delitve mreže iz druge točke poglavja 4.2.5, ki lahko vodi do precej kompleksnih rešitev in slabega končnega modela, je dodatek Weaverbird namenjen rekonstrukciji izbrane geometrije in apliciranju postopkov za »glajenje« mrež z relativno preprosto uporabo komponent. Uporabljena je metoda Catmull-Clark, ki je podrobneje obravnavana v poglavju 3.4.

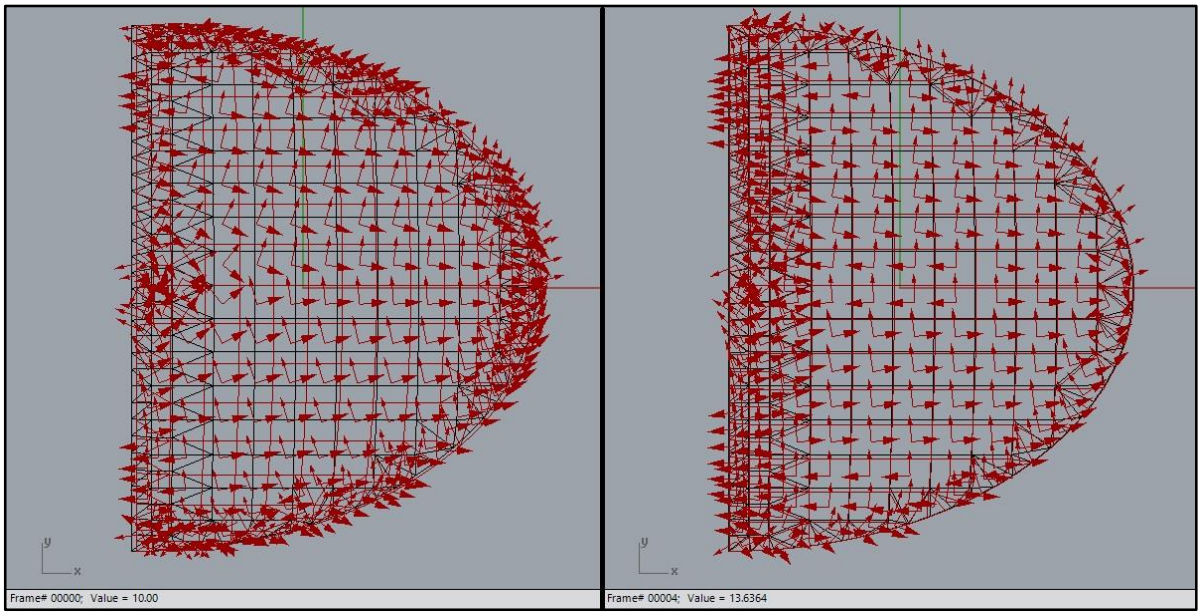
.



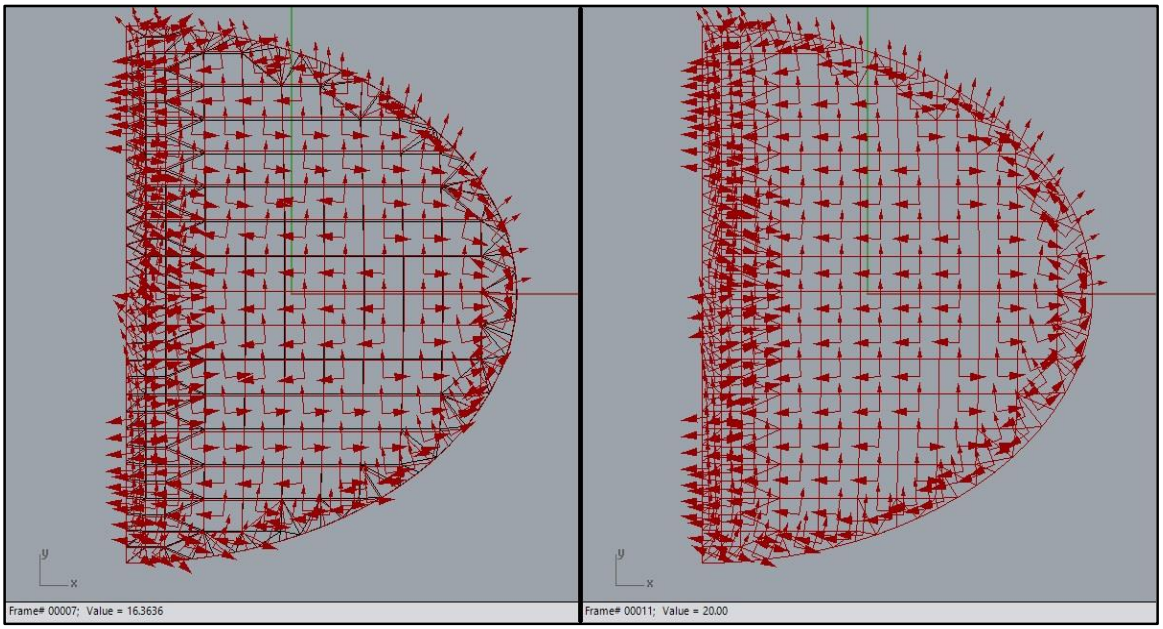
Slika 57: Grafični prikaz delitve mreže ploskovnega modela lupine z uporabo metode Catmull-Clark. Mreža je že po prvi iteraciji sestavljena izključno iz štiri-vozliščnih elementov.

4.2.6 Analiza konstrukcije in optimizacija geometrije

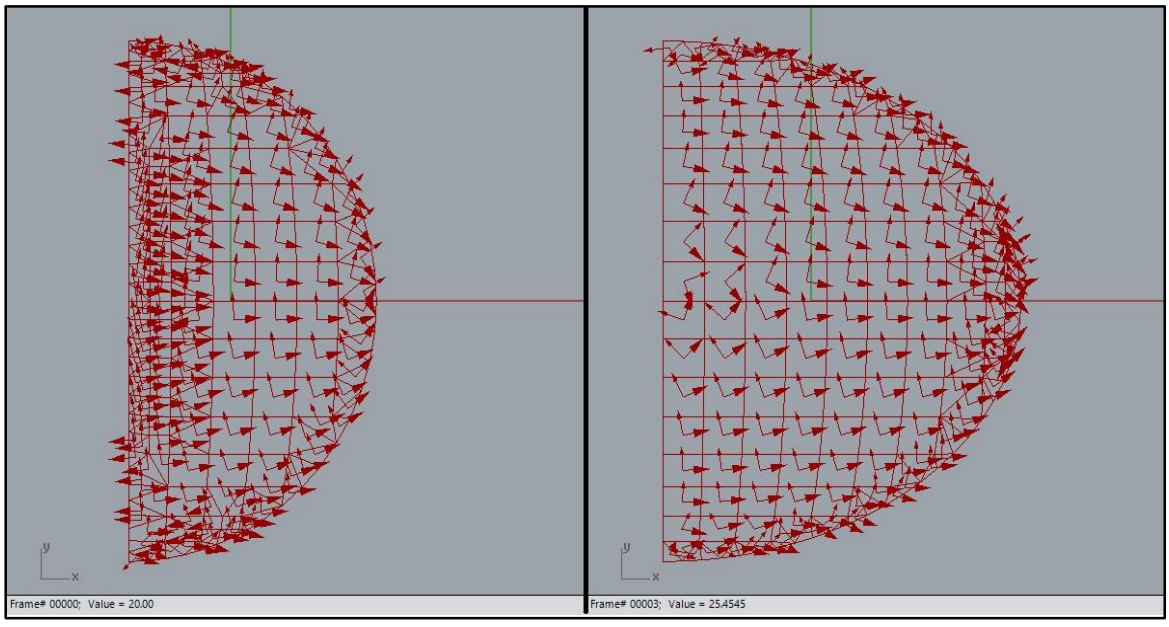
Spremljati želimo potek povprečnih (glej razlago tabele 2) smeri glavnih napetosti ($S_{max\ top}$, $S_{min\ top}$) ter njihovih vrednosti, v odvisnosti od parametrov *R1* in *R2*. Model lupine je v tem primeru obremenjen samo z lastno težo. Zaradi lažjega grafičnega prikaza zopet omejimo vrednosti glavnih parametrov. Na slikah 58, 59, 60 in 61 je prikazano spreminjanje smeri glavnih napetosti (rdeče puščice), za posamezne končne elemente v ravnini *x-y*. Debelejše rdeče puščice predstavljajo smeri večje glavne napetosti (S_{max} na zgornjem robu lupine), tanjše pa smeri manjše glavne napetosti (S_{min} na zgornjem robu lupine).



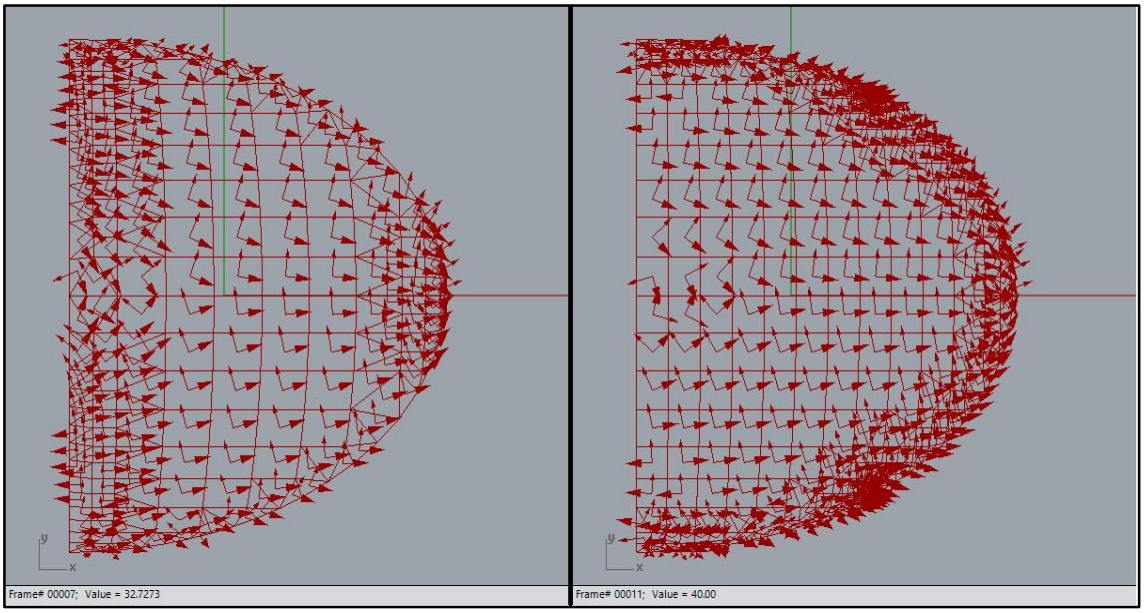
Slika 58: Spreminjanje smeri glavnih napetosti na zgornji ploskvi v odvisnosti od parametra R1 (radij torusa). Levi del prikazuje model lupine pri parametru R1 = 10m, desni del pri vrednosti R1 = 13.6m.



Slika 59: Spreminjanje smeri glavnih napetosti na zgornji ploskvi v odvisnosti od parametra R1 (radij torusa). Levi del prikazuje model lupine pri parametru R1 = 16.3m, desni del pri vrednosti R1 = 20.0m.



Slika 60: Spreminjanje smeri glavnih napetosti v odvisnosti od parametra R2 (radij torusa). Levi del prikazuje model lupine pri parametru R1 = 20.0m, desni del pri vrednosti R1 = 25.4m.



Slika 61: Spreminjanje smeri glavnih napetosti v odvisnosti od parametra R2 (radij torusa). Levi del prikazuje model lupine pri parametru R1 = 32.7m, desni del pri vrednosti R1 = 40.0m.

Ker so rezultati analize konstrukcije odvisni tudi od oblike mreže končnih elementov je v nadaljevanju podan tudi pregled in kratek opis ujemanja karakteristik končnih elementov s priporočili iz SAP2000 priročnika pri zasnovi modela po metodi Catmull-Clark.

Uporabimo primer lupine podane s sledečimi parametri:

R1 = 11.0m,
R2 = 40.0m,
r1 = 12.0m,
r2 = 19.0m.

Osnovna mreža je podana z omejitvijo maksimalne stranice elementa (»Max Edge«) na 2.0m in vključenim pogojem za uporabo minimalnega števila trivozliščnih elementov. Rezultat je privzeta mreža elementov sestavljena iz 264-ih ploskev, ter skupno 247-ih vozlišč. Na dobljeni mreži ploskovnih elementov sedaj uporabimo metodo Catmull-Clark ter obenem beležimo število posameznih elementov, ki v posamezni iteraciji metode, odstopajo od podanih priporočil (poglavje 4.2.5) za modeliranje končnih elementov tipa »shell«.

Tabela 6: Ujemanje geometrijskih lastnosti mreže elementov s priporočili za modeliranje končnih elementov tipa »shell« v programu SAP2000.

DELEŽ ELEMENTOV, KI ODSTOPA OD PODANIH PRIPOROČIL	RAZMERJE STRANIC (PRIPOROČILO ŠT.1)	NOTRANJI KOTI (PRIPOROČILO ŠT.2)	ZASUK ELEMENTOV (PRIPOROČILO ŠT.3)
OSNOVNA MREŽA GENERIRANA Z UPORABO KOMPONENTE »MESH BREP« (98 TRIVOZLIŠČNIH IN 166 ŠTIRIVOZLIŠČNIH ELEMENTOV)	0 *(ujemanje s priporočili)*	0 *(ujemanje s priporočili)*	0 *(ujemanje s priporočili)*
OSNOVNA MREŽA DROBLJENA Z METODO CATMULL-CLARK V PRVI ITERACIJI (*958* ŠTIRIVOZLIŠČNIH ELEMENTOV)	0.1044% *(1 element)*	16.4926% *(158 elementov)*	0 *(ujemanje s priporočili)*
OSNOVNA MREŽA DROBLJENA Z METODO CATMULL-CLARK V DRUGI ITERACIJI (*3832* ŠTIRIVOZLIŠČNIH ELEMENTOV)	0. 0261% *(1 element)*	6.8111% *(261 elementov)*	0 *(ujemanje s priporočili)*
OSNOVNA MREŽA DROBLJENA Z METODO CATMULL-CLARK V TRETJI ITERACIJI (*15328* ŠTIRIVOZLIŠČNIH ELEMENTOV)	0. 0131% *(2 elementa)*	3.2881% *(504 elementov)*	0 *(ujemanje s priporočili)*

4.2.7 Komentar rezultatov

Iz dobljenih podatkov je razvidno, da za uporabljen geometrijski model dvojno ukrivljene lupine ustreznost posameznih elementov (glede na priporočila) narašča skladno z gostoto mreže elementov. Sklepam, da je to posledica boljše aproksimacije površine, ki pa obenem sorazmerno s večanjem natančnosti opisa površine (diskretizacija mreže) povečuje tudi čas izračuna konstrukcije z metodo končnih elementov. Pri izredno gosti mreži končnih elementov lahko v območju podpor konstrukcije dobimo izredno visoke konice napetosti, zato je drobitev mreže smiselna zgolj do neke meje.

Seveda je za določitev mreže sestavljene iz izključno štirivozliščnih elementov, metoda Catmull-Clark zgolj ena izmed možnih. Iskanje oziroma določitev takega algoritma, katerega rezultat bi v celoti ustrezal priporočilom za obliko končnih elementov, je z uporabo programskega okolja Grasshopper seveda možno ustvariti (motivacija za nadaljnje delo na tem področju).

4.2.9 Iskanje optimalne oblike lupine - relaksacija mreže

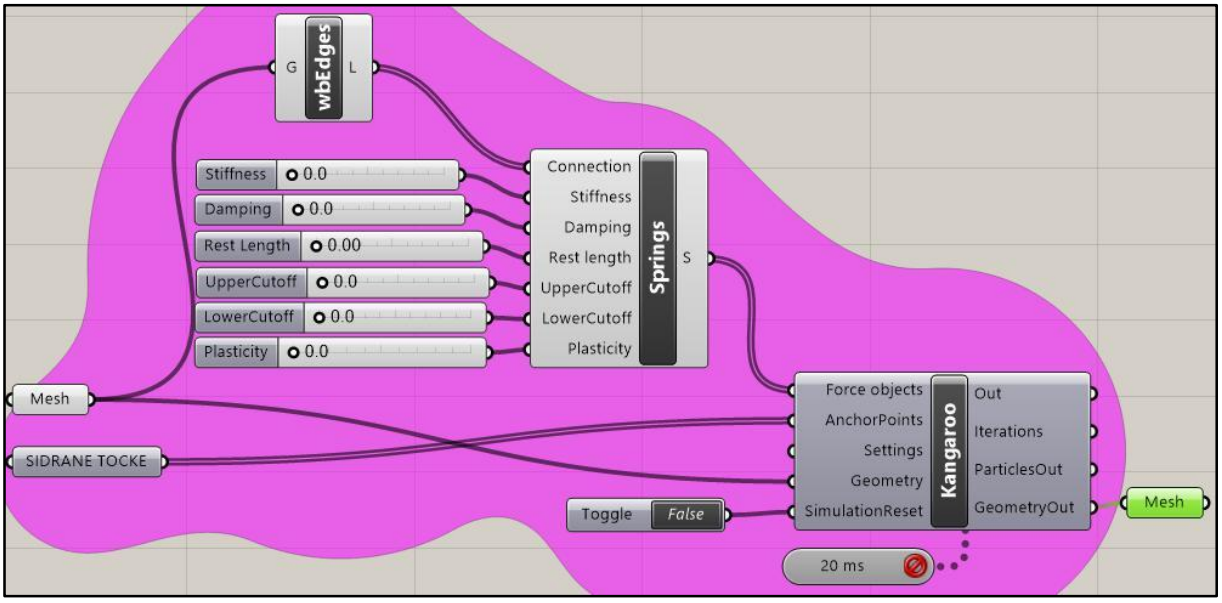
Obnašanje armiranobetonske lupine (v našem primeru hangarja) je v veliki meri odvisno od oblike, ki jo zavzame srednja ploskev modela. V splošnem so bolj primerne oblike lupin, ki so zmožne razviti osne sile ob majhnih upogibnih momentih (membranska teorija plošč) v nasprotju z lupinami, ki obremenitve prenašajo pretežno z upogibnimi momenti. Analiza takih lupin je preprostejša, končni prerezi pa so običajno racionalnejši.

Metode iskanja oblike lupin z računalniškimi algoritmi običajno predstavljajo optimizacijo, ki kot spremenljivke uporabljajo koordinate vozlišč. V zadnjih desetletjih je bilo razvitih precej metod za iskanje optimalne oblike (angl. shape optimization, form finding) za različne tipe konstrukcij (masivne lupine, membrane, paličja itd.)

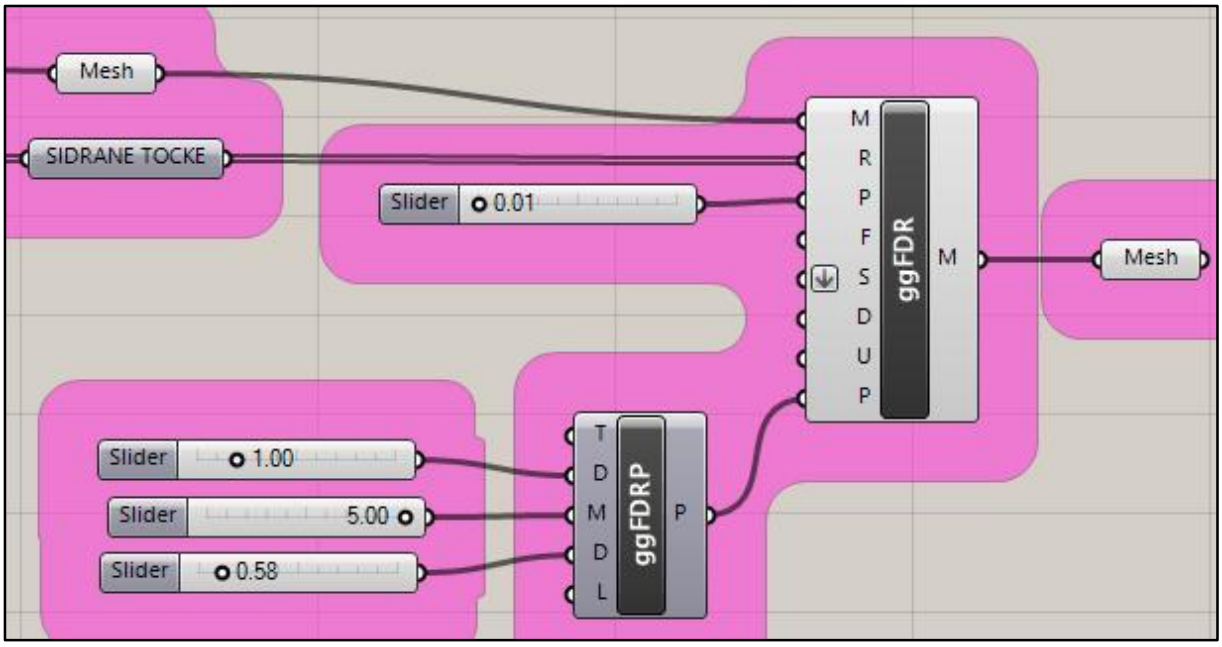
Programska orodja oziroma dodatki za Grasshopper se osredotočajo predvsem na metodo gostote sil (*angl. force density*), kjer srednjo ploskev nadomešča mreža iz vrvi (vzmeti), in metodo dinamične relaksacije (*angl. dynamic relaxation*), kjer se sledi dušenemu gibanju vozlišč zaradi podane obremenitve. Kljub temu, da omenjeni metodi prvotno nista namenjeni iskanju optimalne oblike masivnih lupin, me je zanimal vpliv relaksacije mreže na velikost in potek napetosti lupine hangarja.

V sklopu uporabljenih orodij, so za relaksacijo mreže oziroma iskanje minimalne površine na voljo komponente vtičnika Kangaroo (slika 62) in GeometryGym (slika 63).

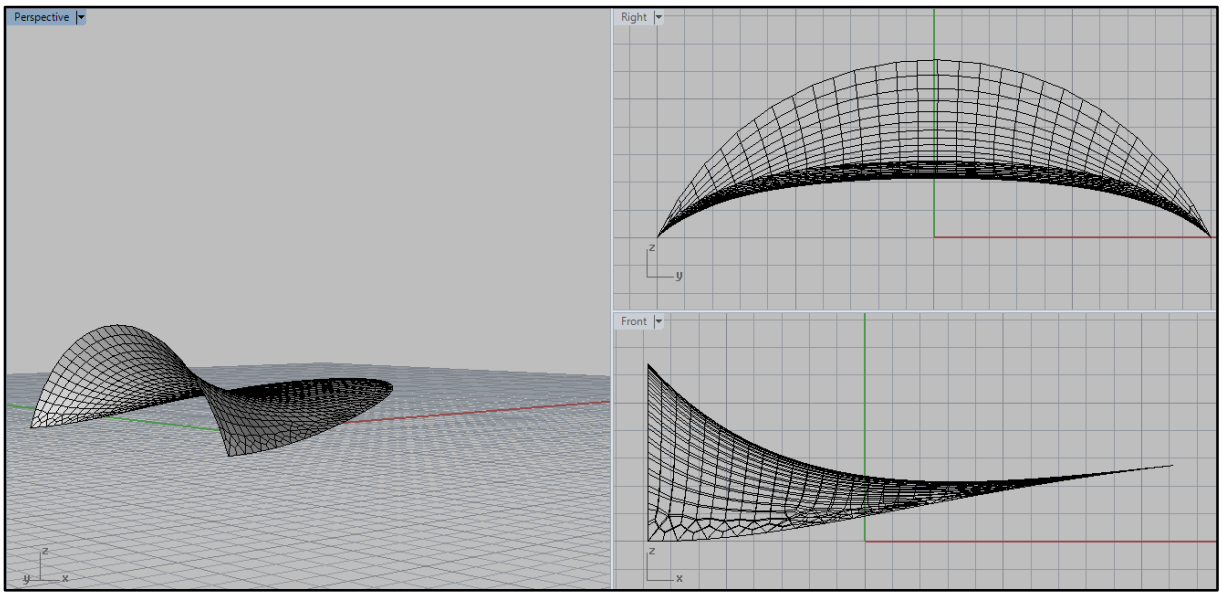
Pri uporabi komponent, katerih ozadja (algoritma) ne poznamo oziroma je težko preverljivo, je nujno potrebna kritična presoja rezultatov, zato je v nadaljevanju prikazan zgolj enostaven primer uporabe komponent, ki služijo relaksaciji mreže.



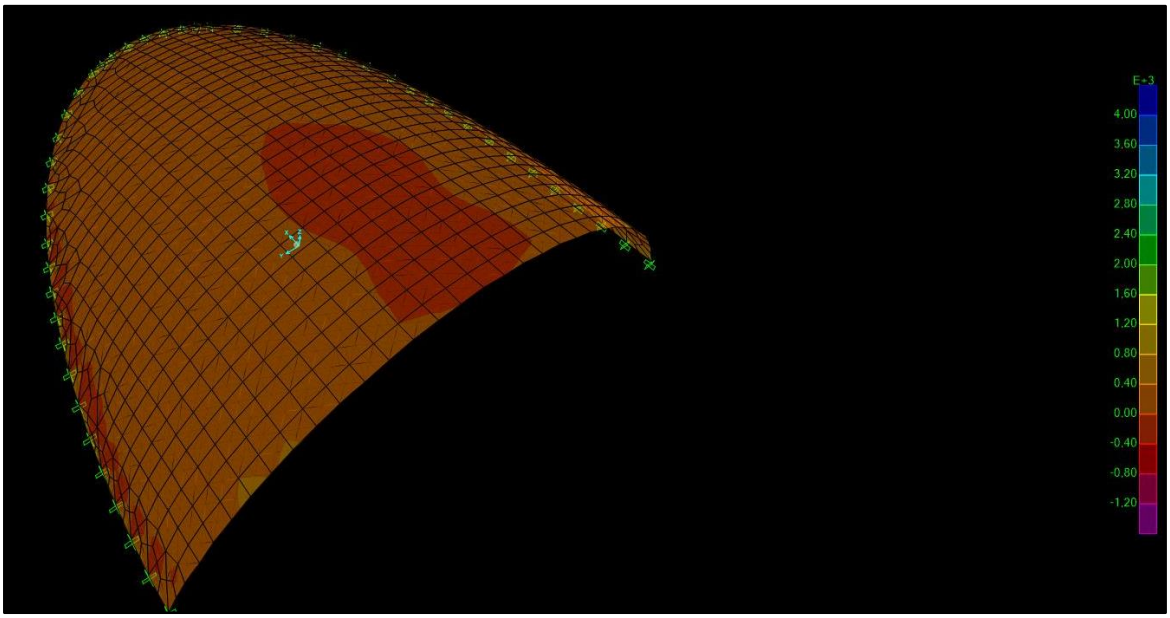
Slika 62: Osnovni sestav algoritma za relaksacijo mreže, z uporabo komponent vtičnika Kangaroo.



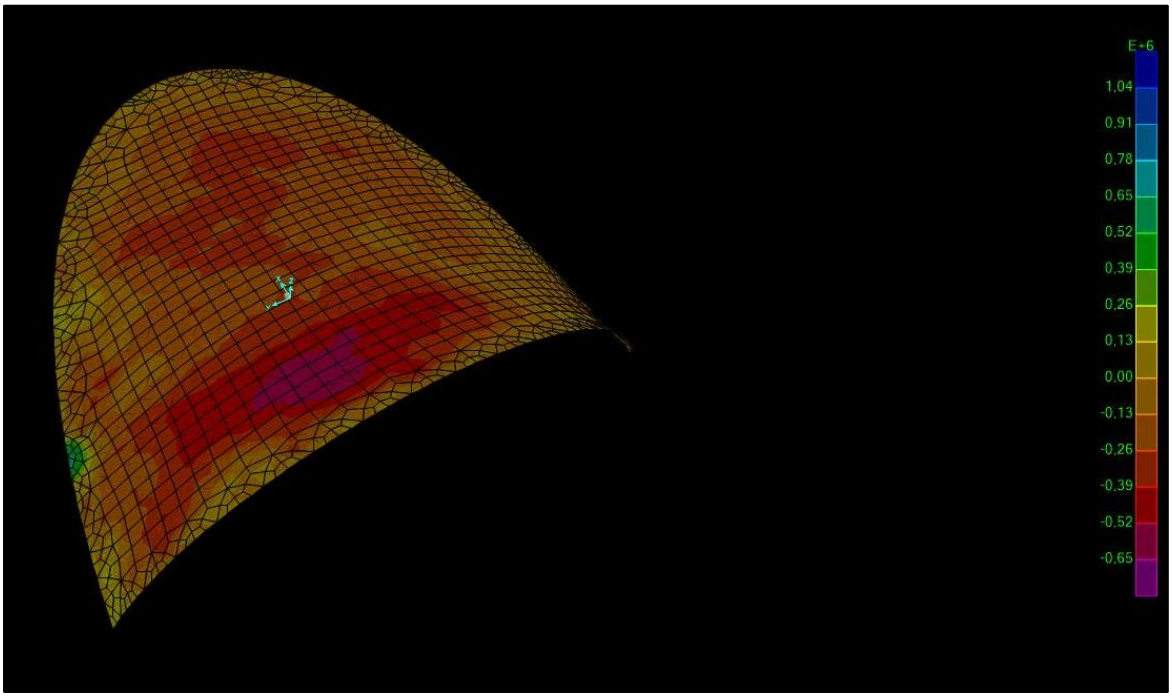
Slika 63: Osnovni sestav algoritma za relaksacijo mreže z uporabo komponent vtičnika GeometryGym.



Slika 64: Prikaz modela lupine z uporabo relaksirane mreže. Relaksacija izvedena s komponentami vtičnika Kangaroo. Točke na robovih lupine so vpete in se med relaksacijo ne premikajo.



Slika 65: Prikaz poteka napetosti na zgornji ploskvi Smax (večja izmed glavnih napetosti) za geometrijo lupine, podane z osnovnimi parametri R1= 11m, R2= 40m, r1= 12m in r2= 19m.



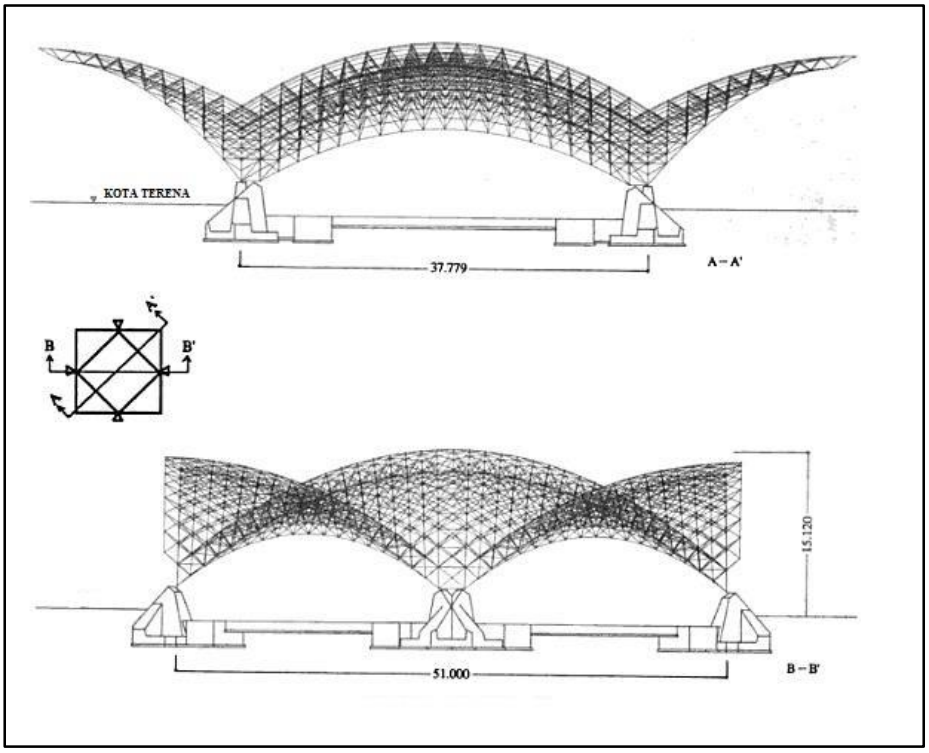
Slika 66: Prikaz napetosti na zgornji ploskvi Smax (večja izmed glavnih napetosti) za geometrijo lupine, podane z osnovnimi parametri R1= 11m, R2= 40m, r1= 12m in r2= 19m in predhodno podvržene relaksaciji (po principih metode gostote sil), z uporabo komponente vtičnika Kangaroo (nastavitve osnovnih parametrov relaksacje so prikazane na sliki 61).

Iz rezultatov (slika 65 in 66) je razvidno, da se napetosti pri predhodno relaksirani mreži končnih elementov občutno povečajo (za faktor 10^3). Za uporabljen primer masivne lupine hangarja lahko sklepam, da relaksacija vpliva neugodno. V kolikor bi bil za lupino izbran drug konstrukcijski sistem (npr. prednapeta membrana), predvidevam, da bi poglavju relaksacije lahko namenil znatno večjo pozornost in ga v diplomski nalogi vključil v večji meri.

4.3 Palična lupinasta konstrukcija

Lupinaste palične konstrukcije so primer konstrukcij, kjer je z uporabo osnovnih gradnikov – palic stikovanih v vozliščih, opisana ukrivljena ploskev. Prednosti takih konstrukcij v primerjavi z monolitnimi lupinami so majhna teža, hitrejša montaža itd. Za prikaz parametrične zasnove palične lupinaste konstrukcije sem uporabil stavbo International Plaza, zasnovano in izdelano samo za potrebe sejma tehnologije v mestu Kobe, Japonska, leta 1981. Za pokritje odprtega razstavišča (paviljona) je bil uporabljen model dvojno ukrivljenega prostorskega okvirja podprtega v samo štirih točkah. Podatki o projektu povzeti po [36].

Osnovno obliko paviljona sestavljajo štiri ploskve v obliki trikotnih listov, povezane s centralno eliptično paraboloidno lupino (slika 67). Celotna pokrita površina je obsegala približno 2600m^2.

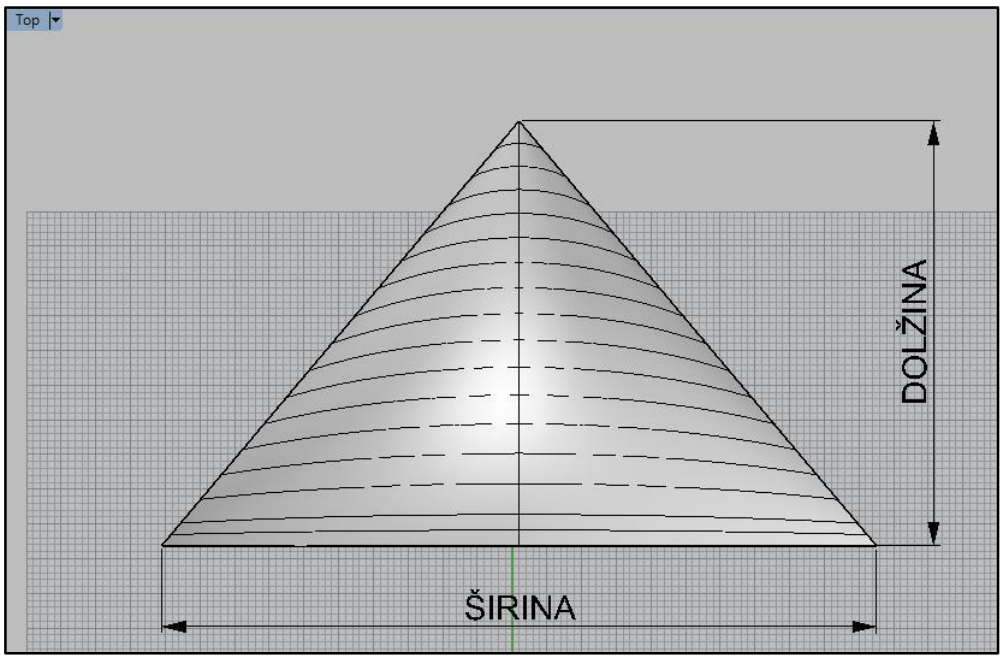


Slika 67: Paviljon International Plaza, Kobe, Japonska.

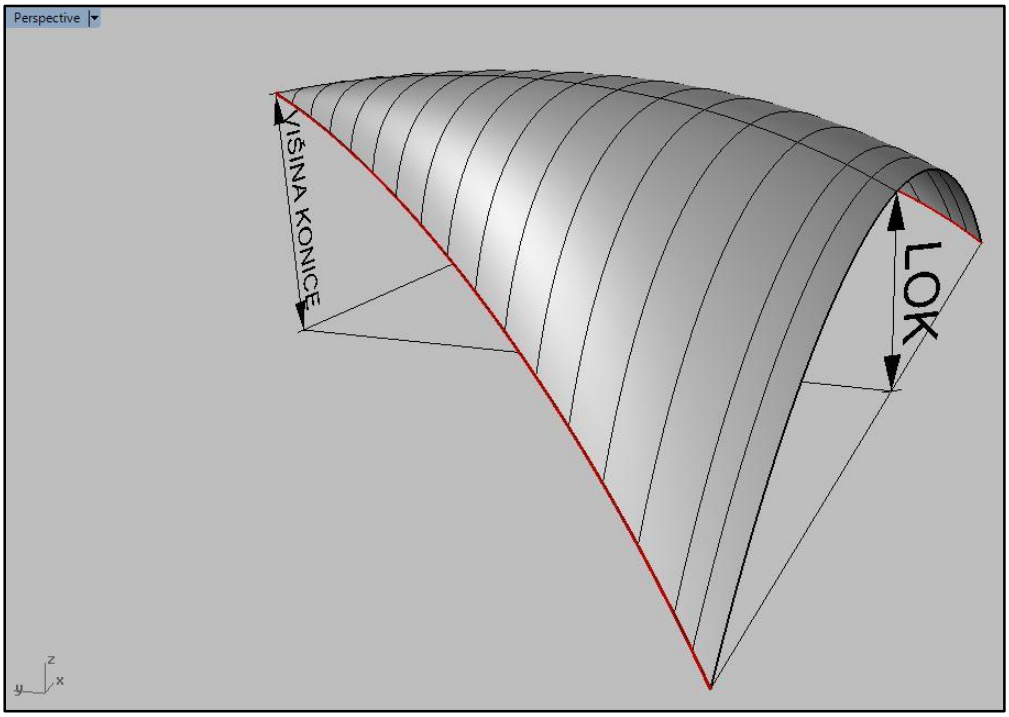
4.3.1 Zasnova in parametrizacija geometrije konstrukcije

Pri zasnovi modela sem tokrat uporabil vtičnik Paneling Tools, z namenom prikazati načina podajanja točkovnih mrež na izbrano ploskev. Mreže, definirane z uporabo Paneling Tools, predstavljajo zgolj logično urejene (»drevesna struktura«) koordinate točk, s katerimi v nadaljevanju algoritma določamo povezave prostorskega paličja.

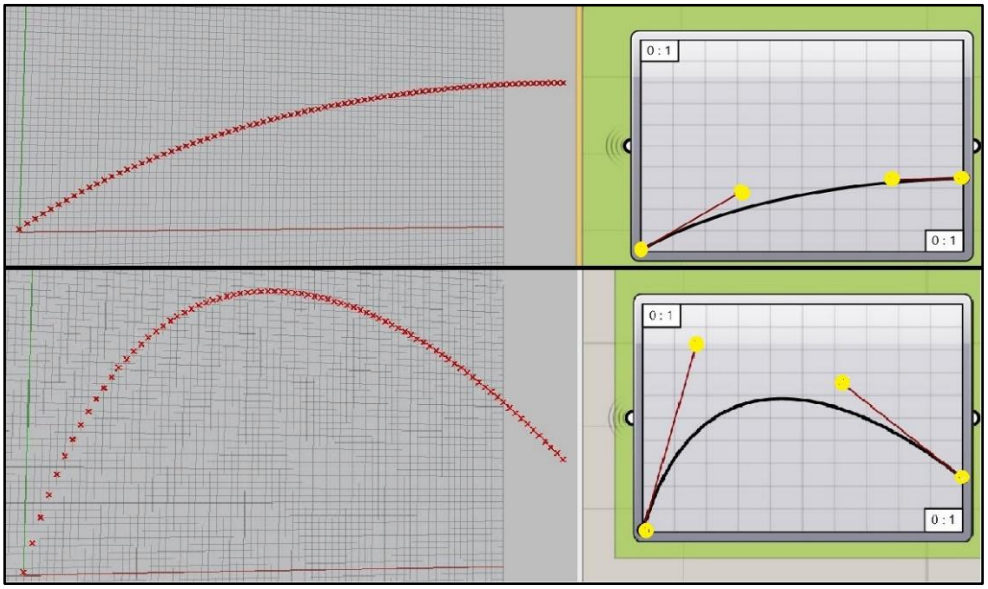
Parametrična zasnova izhaja iz oblike kril, ki so določena z obliko treh robnih krivulj. V modelu določa obliko in velikost kril pet osnovnih parametrov, s katerimi poljubno vplivamo na končno geometrijo kril.



Slika 68: Oznaka parametrov, ki v tlorisu določajo obliko krila (»širina«, »dolžina«).

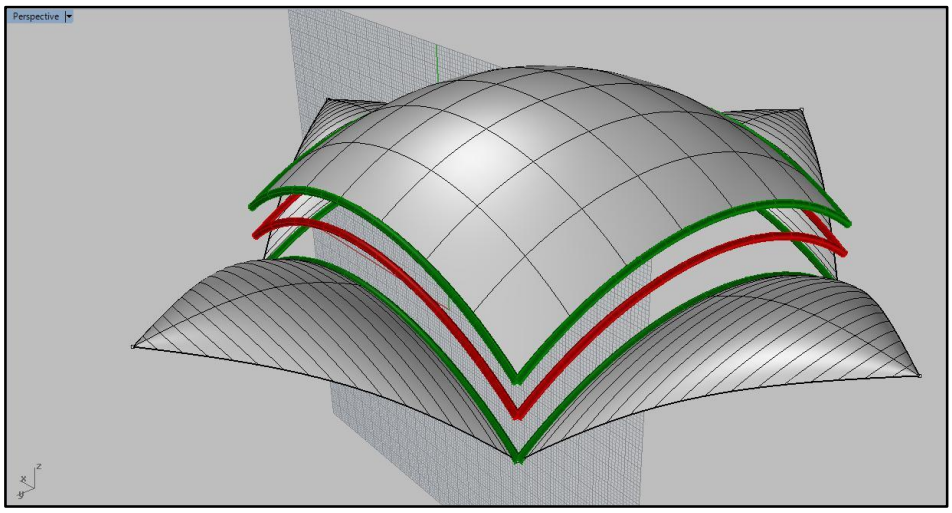


Slika 69: Oznaka parametrov, ki določajo obliko krila (»višina konice«, »lok«). Oblika krivulje označene z rdečo barvo je določena z zadnjim parametrom (slika 71).

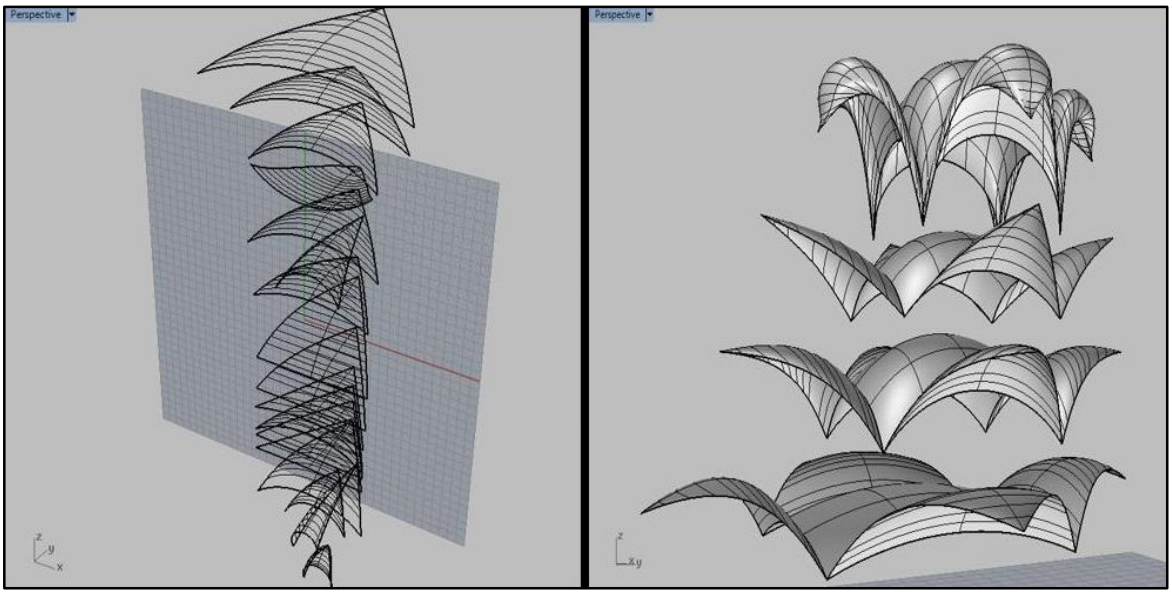


Slika 70: Zadnji parameter, ki določa obliko krila, je vezan na prosti stranici (rdeča oznaka na sliki 69). Spremenljivko tokrat predstavljajo štiri kontrolne točke (desna stran slike – rumene točke), s katerimi določamo obliko stranice (leva stran slike). Izjemoma tokrat parameter ne zavzame numerične vrednosti, a omogoča korenito spreminjanje oblike krila.

Površina, ki določa končno obliko kupole, je določena s štirimi robnimi krivuljami, ki obenem predstavljajo eno izmed robnih krivulj posameznega krila (slika 71). Srednja ploskev kupole je z ukazom {Network surface} torej napeta med robnimi štirimi krivuljami. Posredno s spreminjanjem parametrov, ki določajo krila (»Lok« in »Širina«), vplivamo tudi na obliko kupole.

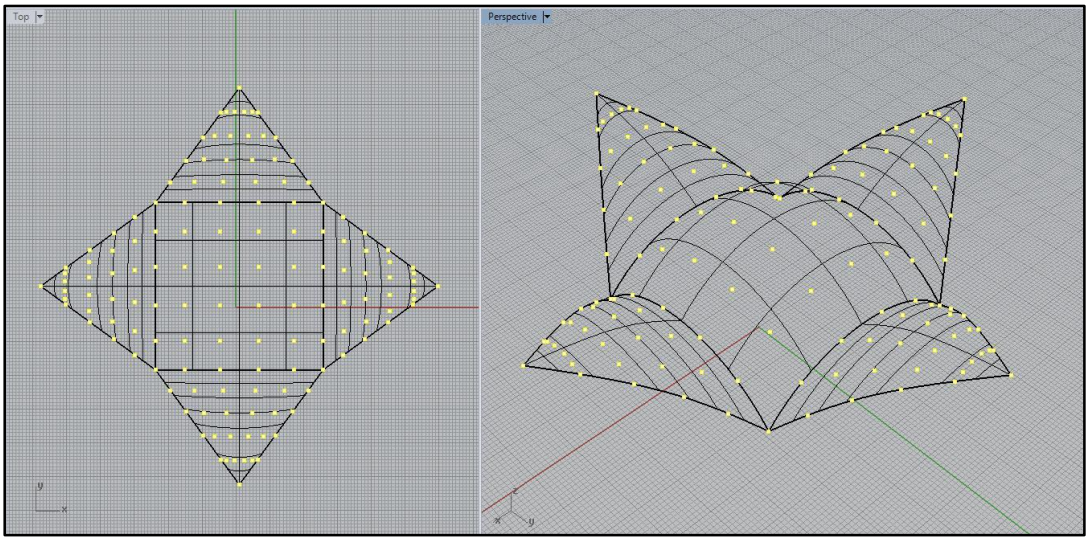


Slika 71: Prikaz dvignjene ploskve osrednje kupole napete med robne štiri krivulje (zelene krivulje za krila in kupolo so enake). Sklenjena rdeča krivulja predstavlja robne krivulje kupole.



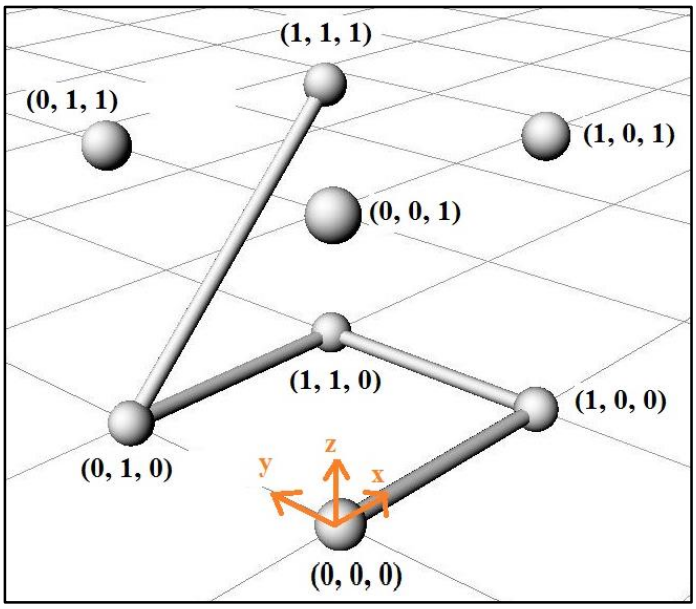
Slika 72: Grafični prikaz modela krila (levo) in celotnega paviljona (desno) za naključno izbrane vrednosti začetnih parametrov.

Parametrično zasnovano obliko paviljona sedaj z uporabo Paneling Tools prekrijemo z urejeno točkovno mrežo (slika 73). Število in medsebojne razdalje točk, določimo znotraj posameznih komponent algoritma.



Slika 73: Pokrivanje izbrane oblike paviljona z mrežo urejenih (Grasshopper podatkovne strukture) točk.

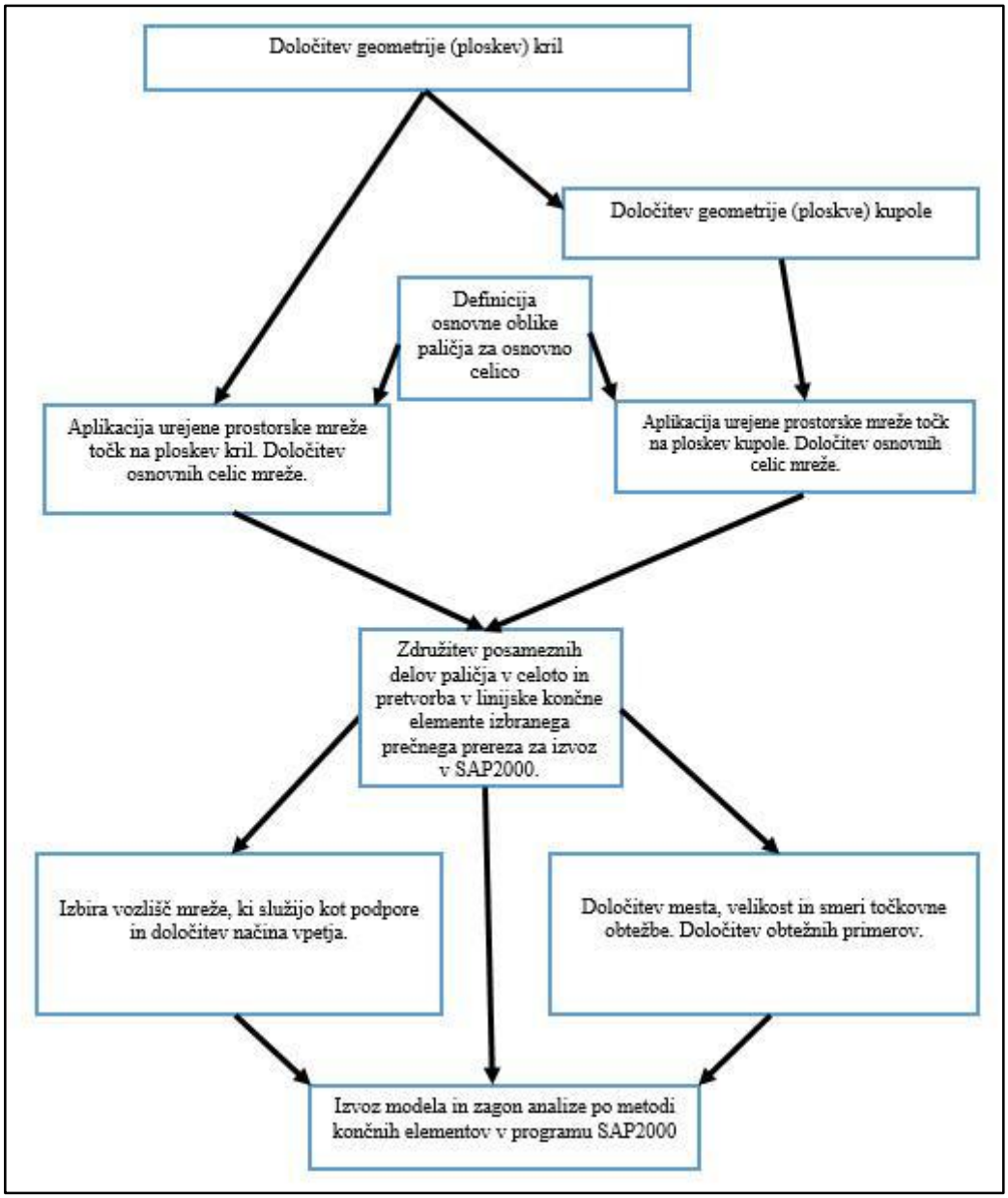
Celotno mrežo nato preslikamo na poljubno višino, s čimer dobimo prostorsko mrežo urejenih točk. Skupine »sosednjih« osmih točk tvorijo osnovne celice, na podlagi katerih nato določamo povezave, ki predstavljajo palice prostorskega paličja (slika 74). V našem primeru spodnje štiri točke osnovne celice nadomestimo z eno samo in s tem nekoliko zmanjšamo samo število palic.



Slika 74: Osnovna (enotska) celica prostorskega paličja, podanega z uporabo komponente {ptMPanel3D} vtičnika Paneling Tools. Povezave med vozlišči podajamo v obliki spremenljivk (angl. pattern string). Povezave/linije med vozlišči na sliki predstavlja zapis (0,0,0)(1,0,0)(1,1,0)(0,1,0)(1,1,1).

Tak način podajanja povezav prostorskega paličja nudi uporabniku ogromno manevrskega prostora pri izbiri osnovne oblike paličja. Dimenzije in število osnovnih celic v algoritmu prav tako podamo parametrično.

Izbira tipa palične kupole je torej določena zgolj z enostavno zasnovanim osnovnim gradnikom (celico) ploskovne ali prostorske mreže ter pripadajočim vzorcem povezav. Pred izvozom modela v SAP2000 vsaki palici določimo prečni prerez in izberemo material ter določimo podpore in obtežbo.



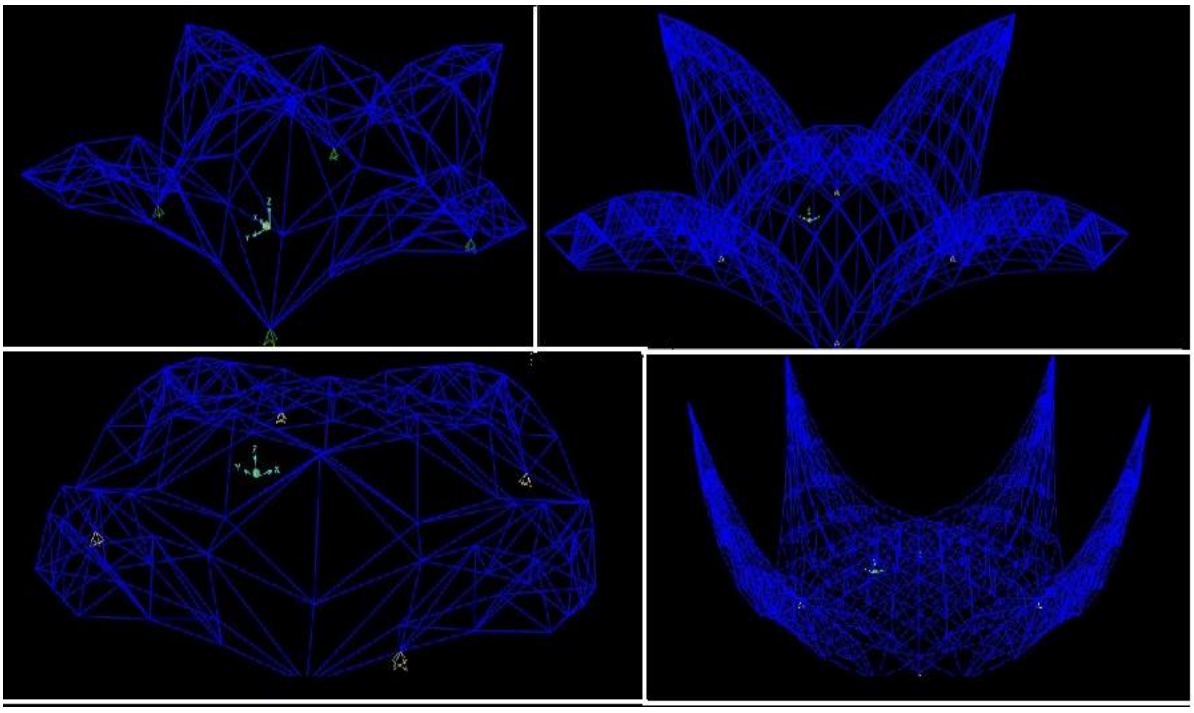
Slika 75: Diagram poteka algoritma za model paviljona.

4.3.2 Obtežba in podpiranje konstrukcije

Modeli konstrukcije, ki jih izvozimo v program SAP2000, so poenostavljeni in zajemajo zgolj osnovne vplive (lastna teža, točkovna obtežba zaradi koristne obtežbe v vozliščih palic). Obtežba zaradi vpliva snega, vetra in obtežne kombinacije, ter kontrola uklona niso zajeti v modelu. Prav tako niso posebej obravnavani stiki palic oziroma sam način izvedbe (vijačenje, varjenje). Konstrukcija je podprta prostoležeče v štirih točkah.

4.3.3 Komentar

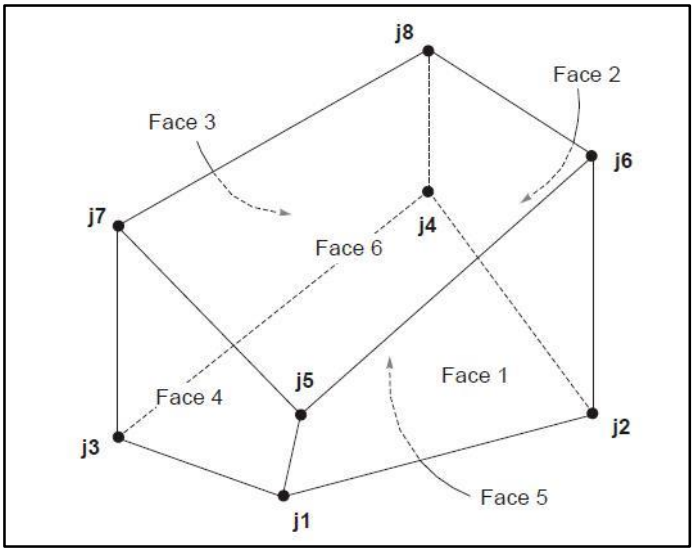
Pri modeliranju izbrane palične konstrukcije v okolju Grasshopper sem želel prikazati dejstvo, da z uporabo premišljene zasnove in izbiro pravih osnovnih parametrov občutno vplivamo na raznovrstnost množice končnih modelov konstrukcije. Parametrični model ima zaradi načina podajanja določene poenostavitve in se od dejanskega objekta seveda razlikuje.



Slika 76: Prikaz raznovrstnosti množice rešitev modela palične lupinaste konstrukcije v programu SAP2000, podane z enim samim algoritmom, definiranim v okolju Rhinoceros – Grasshopper.

4.4 Optimizacija lameliranega nosilca

Zadnji primer parametričnega modeliranja sem prikazal na prosto ležečem nosilcu, sestavljenem iz lepljenih lamel. Za razliko od ostalih primerov so tokrat posamezne lamele modelirane z uporabo končnih elementov tipa »solid« [33], ki ga določajo koordinate osmih vozlišč (slika 77).

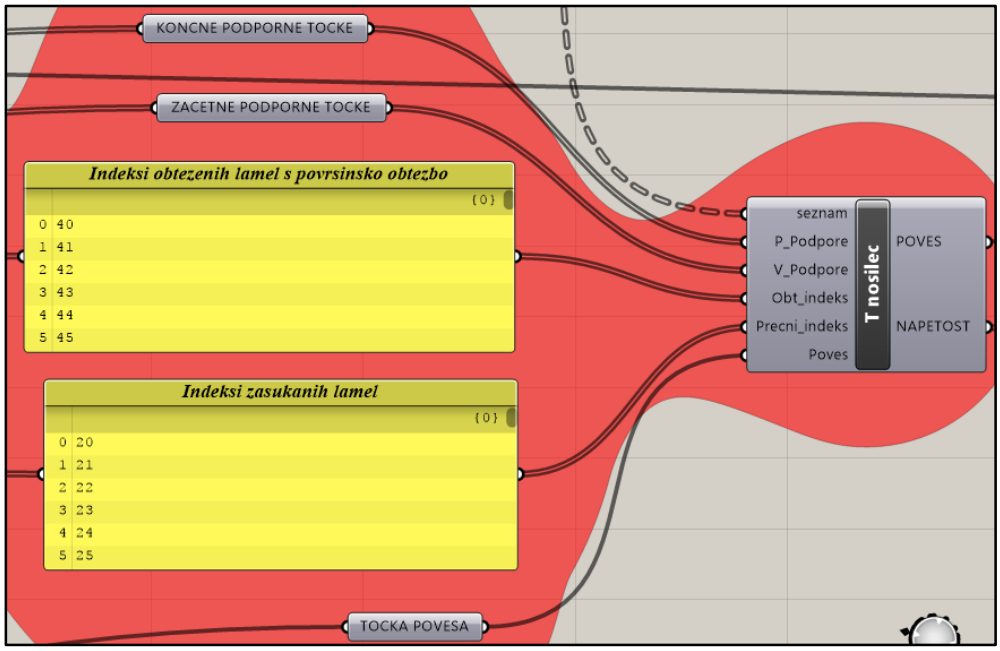


Slika 77: Končni element tipa "solid". Zaporedje označevanja vozlišč (od j1 do j8) je potrebno pri prenosu modela v programsko okolje SAP2000 dosledno upoštevati [33].

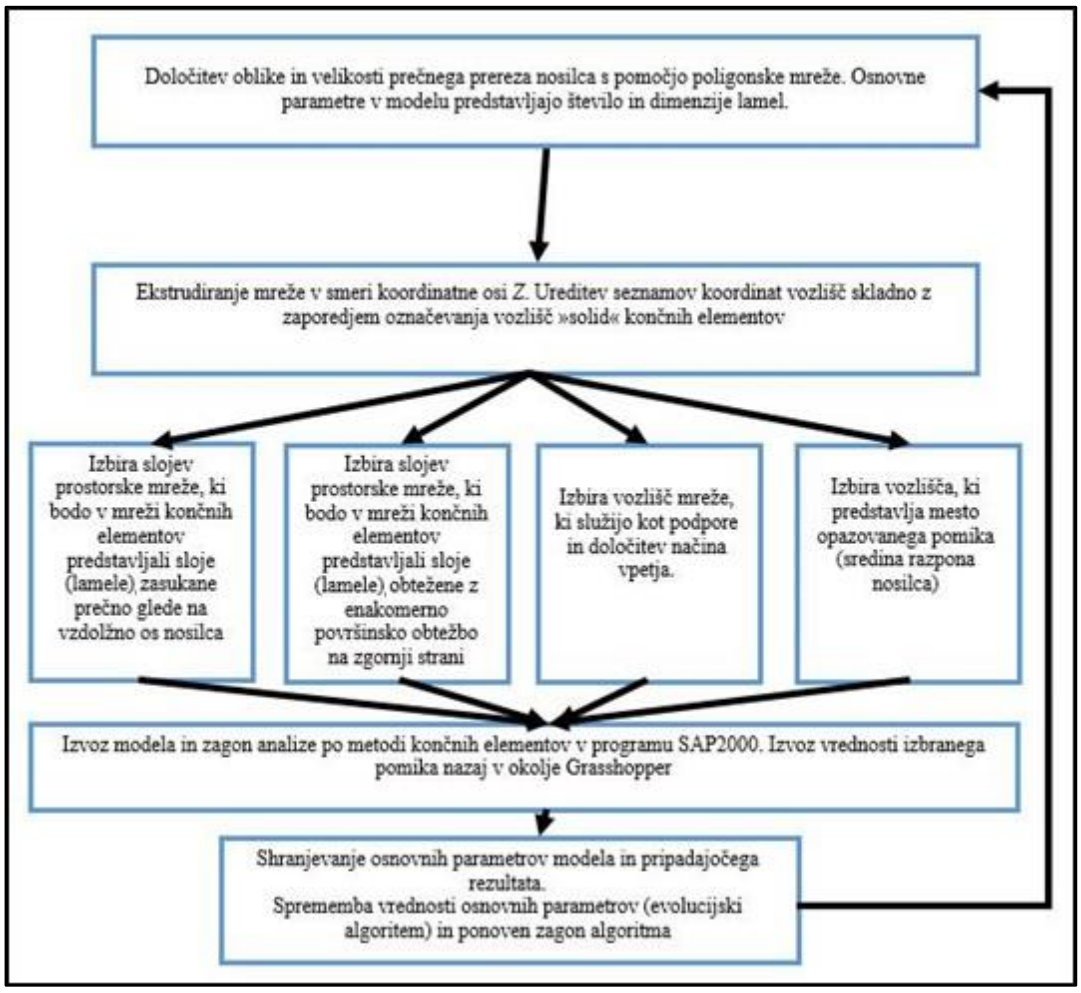
Izhodišče modela je torej urejena mreža kvadrov z ustrezno »orientacijo« vozlišč. Karakteristike uporabljenega materiala podamo z določitvijo koeficientov splošnega anizotropnega materiala znotraj modela. Osnovni cilj take zasnove je določitev optimalne razporeditve posameznih plasti lamel tako, da bo pomik na sredini nosilca minimalen.

Ker vmesnik GeometryGym ne omogoča podajanja vseh potrebnih vhodnih podatkov za analizo tako zasnovanega nosilca, sem za ta primer izdelal namensko komponento z uporabo programskega jezika Visual Basic.NET. Komponenta z imenom {T nosilec} preko OAPI vzpostavi dvosmerno povezavo s SAP2000 in glede na podane parametre (po končani analizi v programu SAP2000) vrne želene rezultate. Vhodne parametre predstavljajo urejene koordinate vozlišč posameznih končnih elementov, koordinate podpor, indeksi obteženih končnih elementov in indeksi tistih končnih elementov, ki v modelu predstavljajo lamele, postavljene prečno na vzdolžno os nosilca. Ostale lamele so orientirane v smeri vzdolžne osi (slika 80).

V prilogi DIII je podan celoten potek Visual Basic.NET skripte, ki definira obliko in delovanje komponente {T_nosilec}.



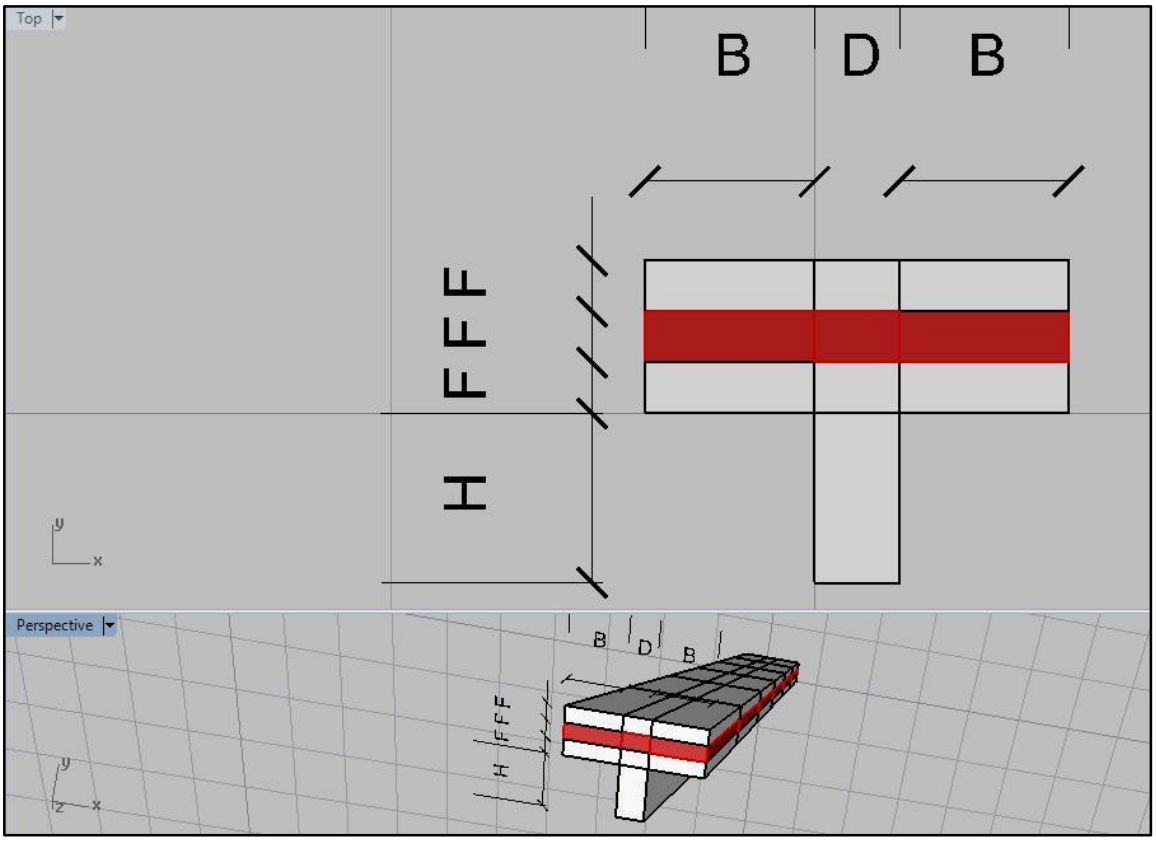
Slika 78: Prikaz namenske komponente {T_nosilec}.



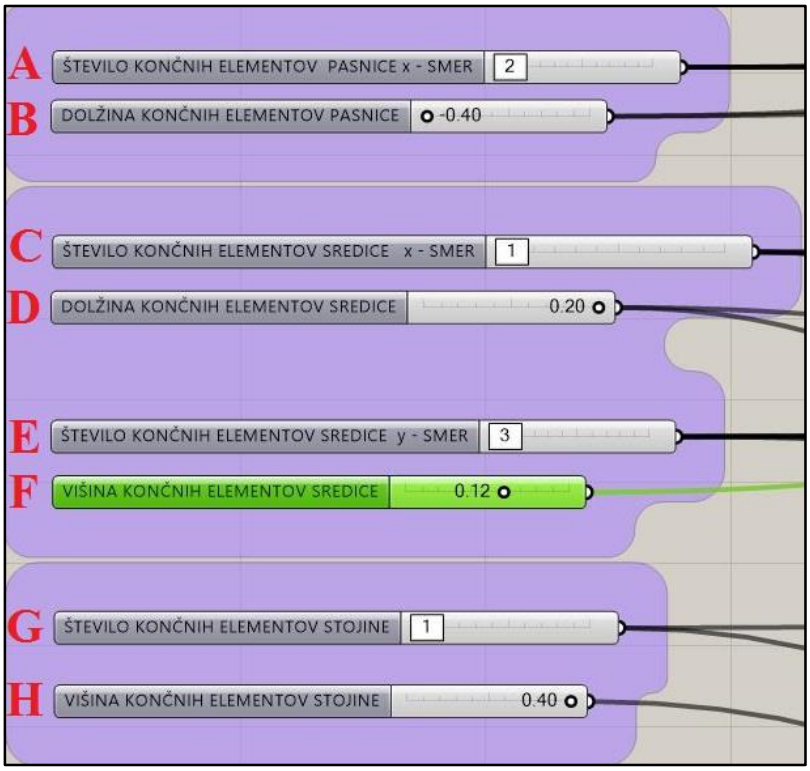
Slika 79: Diagram poteka algoritma za model optimizacije nosilca.

4.4.1 Zasnova in parametrizacija geometrije konstrukcije

Osnovna oblika nosilca izhaja iz enostavne oblike prečnega prereza T-oblike. Ker pa želimo določiti optimalno razporeditev in dimenzije posameznih lamel, je pozornost pri parametrični zasnovi celote potrebno nameniti posameznim lamelam. Število, geometrija, položaj in orientacija le-teh namreč predstavljajo končne neznanke. Prečni prerez je definiran v ravnini globalnih osi *x-y*, vzdolžna os nosilca pa sovpada z globalno koordinatno osjo *z*.

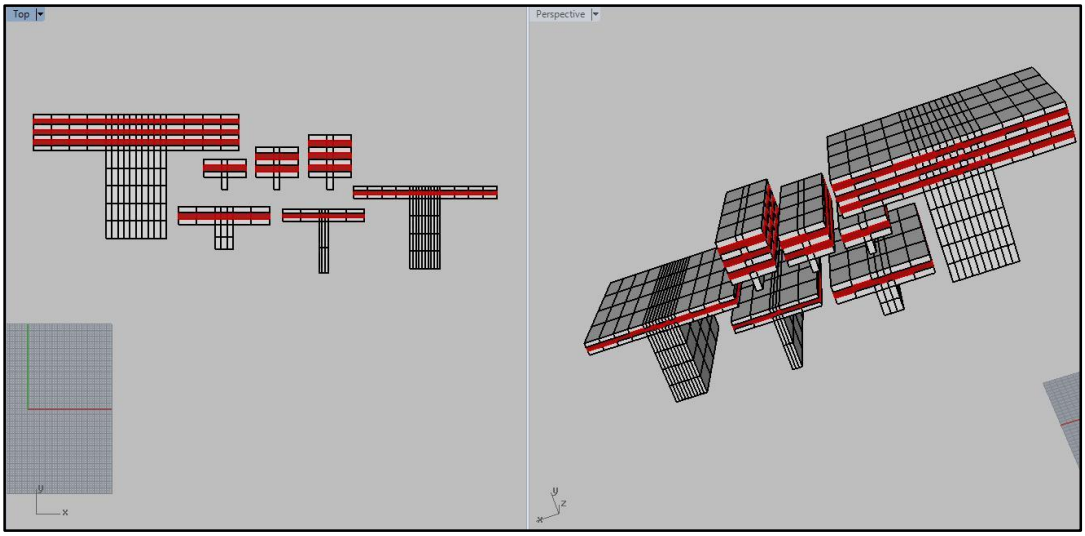


Slika 80: Osnovna oblika nosilca, iz katere izhajamo pri parametrični zasnovi. Pasnica je (po višini) sestavljena iz najmanj treh vrst. Lamele, označene z rdečo barvo so v ravnini x-z zasukane za 90 stopinj glede na vzdolžno os nosilca. Kotirane oznake prikazujejo osnovne parametre (glej sliko 81).



Slika 81: Prikaz drsnikov osnovnih parametrov. Parametri, ki določajo dolžino oziroma višino posameznih končnih elementov, so z istimi oznakami (rdeče črke) prikazani na sliki 80.

Nabor možnih rešitev (slika 82) je smiselno omejiti, saj določene oblike prečnih prerezov preprosto niso primerne (tehnologija izvedbe, nosilnost materiala itd.).



Slika 82: Prikaz raznovrstnosti nekaj možnih rešitev parametrično podanega modela prostoležečega nosilca.

Pri določevanju oblike pasnice nosilca predpostavimo sledeče pogoje:

- Prečni prerez pasnice je po višini vedno sestavljen iz lihega števila lamel (3, 5 ali 7 vrst).
- Vsaka druga vrsta lamel v pasnici je zasukana za 90 stopinj v ravnini pasnice (2., 4. ali 6.vrsta - odvisno od celotnega števila lamel).
- Višina posamezne lamele pasnice je omejena z minimalno (6cm) in maksimalno (16cm) vrednostjo (slika 81, oznaka F).

Geometrijo stojine omejimo na naslednji način:

- Višina posamezne lamele je omejena z minimalno (8cm) in maksimalno (40cm) vrednostjo (slika 81, oznaka H).
- Širina posamezne lamele je omejena z minimalno (1.2cm) in maksimalno (2.0cm) vrednostjo (slika 81, oznaka D).
- Vse lamele v stojini so enako orientirane.

Navedene omejitve geometrije modela (število, širina in višina lamel) vpeljemo v algoritem z uvedbo maksimalnih in minimalnih vrednosti, ki jih drsniki osnovnih parametrov lahko zavzamejo. Zasuk posameznih lamel dosežemo s spremembo faktorja »Material angle«, ki v SAP2000 določa orientacijo uporabljenega materiala za končne elemente skladno z lokalnim koordinatnim. Ostali parametri, potrebni za potek analize po metodi končnih elementov so definirani znotraj programske skripte komponente (priloga DIII) in so tekom parametrične analize konstantni (dolžina nosilca, lastnosti materiala, obtežba itd.).

4.4.2 Obtežba in podpiranje konstrukcije

Nosilec je podprt prosto ležeče in obremenjen s konstantno površinsko obtežbo 10 kN/m^2. Podatki o materialni karakteristikah (elastični moduli, poissonovi količniki, strižni moduli in koeficienti temperaturnega raztezka za posamezne smeri lokalih koordinatnih osi končnih elementov) so podani v prilogi DIII.

4.4.3 Analiza konstrukcije in optimizacija geometrije

Navedene osnovne parametre povežemo s komponento {Galapagos}, namensko funkcijo pa pri tem določimo kot vrednost povesa (pomik v negativni smeri globalne osi *y*) na sredini nosilca. Iščemo minimalno vrednost namenske funkcije (čim manjši poves).

Pred analizo konstrukcije sem se, zaradi ogromnega števila možnih končnih rešitev (cca. 11.059.200 različnih primerov), osredotočil na vrednosti parametrov, ki določajo višino nosilca. Z uporabo pridobljenega znanje mehanike trdnih teles in obnašanja lepljenih lameliranih konstrukcij sklepam, da je za primer prosto ležečega nosilca vrednost povesa obratno sorazmerna z višino prečnega prereza.

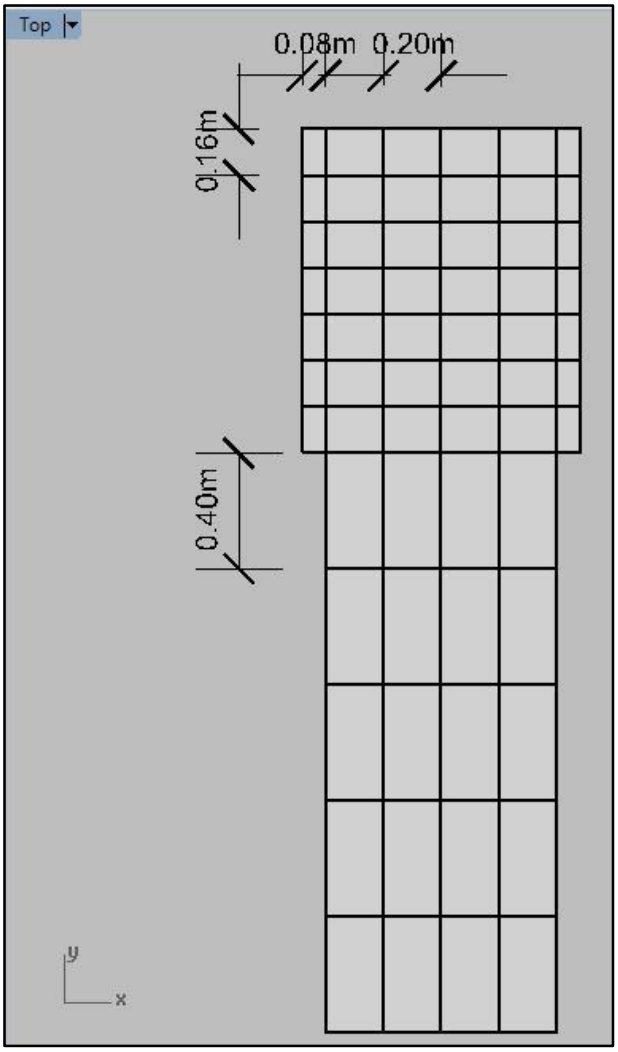
Zaradi skrajšanja časovnega okvira izračuna sem zato parametre, ki vplivajo samo na višino nosilca (slika 81, oznake E, F, G in H) nastavil na vrednosti, ki nosilcu priredijo maksimalno višino prečnega prereza. Parametre modela torej sedaj predstavljajo samo še vrednosti drsnikov A, B, C in D (slika 81). Algoritem iskanja optimalne rešitve je bil s temi spremembami zagnan.

4.4.4 Komentar rezultatov

Končni rezultat, glede na podane omejitve algoritma, je dosežen v 22.generaciji (približno 1100 izračunov), pri čemer je vseh možnih rešitev natanko 2304. Algoritem je potekal približno 19 ur, vendar je potrebno omeniti dejstvo, da so bili modeli nosilca sestavljeni iz velikega števila končnih elementov in je bil čas izračuna posamezne iteracije zato razmeroma dolg (do 30 sekund). Algoritem kot optimalno rešitev vrne poves sredinske točke nosilca za geometrijo, ki jo določajo samo mejne vrednosti drsnikov. Širine lamel so minimalne, medtem ko višine zavzemajo maksimalne (možne) vrednosti. Rezultat je torej razmeroma trivialna rešitev (slika 83).

Tabela 7: Povzetek iskanja optimalne rešitve modela nosilca.

KARAKTERISTIKE MODELA		
Konstantne vrednosti:	Dolžina nosilca v *z* smeri: Število končnih elementov v smeri *z* osi: Površinska obtežba na zgornji ploskvi nosilca: Materialne lastnosti (modul elastičnosti, Poissonov količnik, strižni modul, temperaturni koeficient)	4.0 20 $10kN/m^2$ Glej prilogo DIII
Konstantne nastavljene vrednosti {Galapagos} komponente:	ŠTEVILO STAGNIRANIH POPULACIJ (»Max. Stagnant«): ŠTEVILO ZAČETNE POPULACIJE (»Population«): ZAČETNO POVEČANJE (»Initial Boost«): DELEŽ OHRANITVE (»Maintain«): FAKTOR HOMOGAMIJE (»Inbreeding«):	50 50 2 5% +50%
Spremenljivke/parametri	Število končnih elementov pasnice v *x* –smeri: Dolžina končnih elementov pasnice: Število končnih elementov sredice v *x* –smeri: Dolžina končnih elementov sredice:	2 0.08 4 0.20
Optimalna namenska funkcija:	POMIK NOSILCA NA SREDINI RAZPONA V SMERI NEGATIVNE GLOBALNE OSI *y*	-0.24462



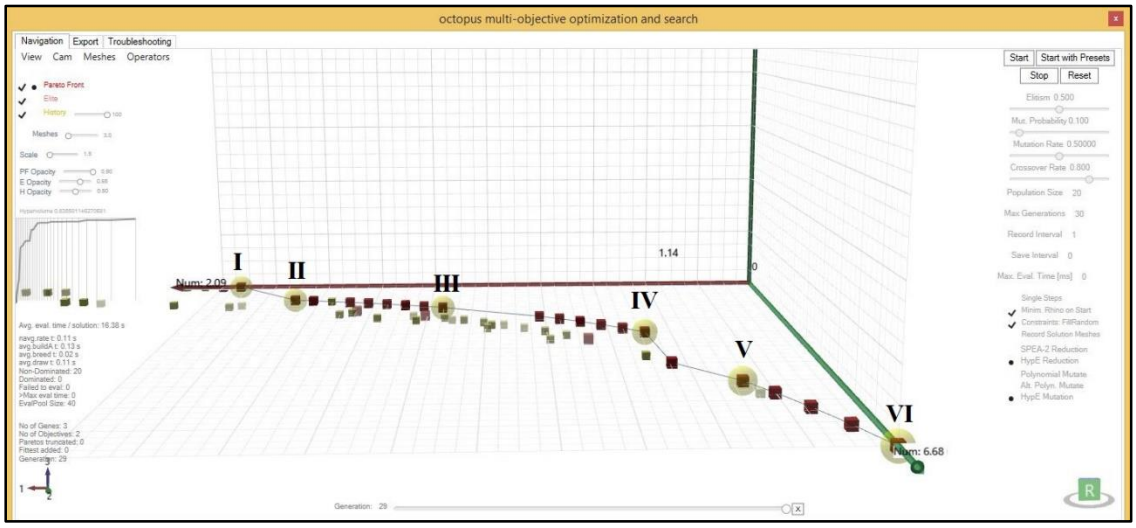
Slika 83: Končna optimalna razporeditev števila in dimenzij lamel za primer prosto ležečega nosilca. Dimenzije sorodnih lamel v pasnici in stojini prečnega prereza so enake.

4.4.4 Analiza konstrukcije in optimizacija geometrije z uvedbo dodatne namenske funkcije

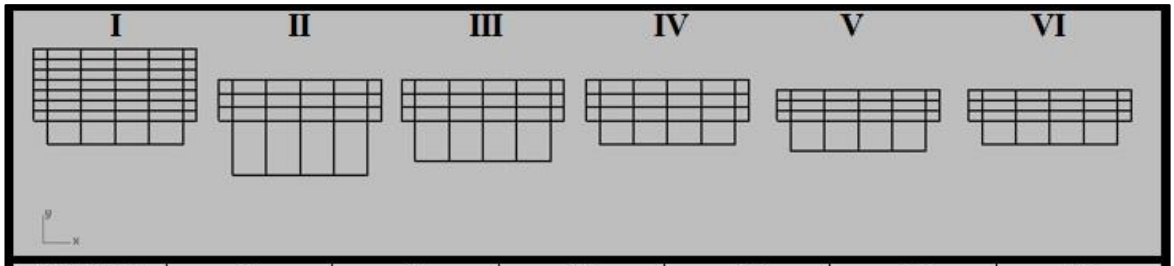
V kolikor želimo, za obravnavni nosilec poiskati rešitev, ki bo ob pogoju minimalnega povesa, obenem zagotavljala tudi minimalno vrednost volumna celotnega nosilca, lahko to poiščemo s komponento {Octopus}.

Izhajamo iz razporeditve lamel iz prejšnjega poglavja, pri čemer tokrat osnovne parametre predstavljajo (slika 81, oznake E, F in H) samo količine, ki vplivajo na višino nosilca.

Rezultat za nastavljenih 30 generacij (posamezna generacija sestavljena iz 20 izračunov/iteracij) je prikazan na sliki 84.



Slika 84: Grafični prikaz razporeditve najustreznejših dobljenih rešitev. Vrednosti na rdeči osi (od cca. 1.14 do 2.09) predstavljajo prvo namensko funkcijo (velikost povesa), vrednost na zeleni osi (od cca. 0 do 6.68) pa drugo namensko funkcijo (volumen nosilca). Rešitve povezane s črto predstavljajo Pareto optimum. Označene rešitve (od I do VI) so podrobneje prikazane na sliki 85.



OZNAKA	I	II	III	IV	V	VI
ŠT. VRST	7	3	3	3	3	3
VIŠINA LAMEL PASNICE	0.06	0.08	0.08	0.08	0.06	0.06
VIŠINA LAMEL STOJINE	0.14	0.32	0.24	0.14	0.18	0.14

Slika 85: Prečni prerezi, ki pripadajo rešitvam označenim na sliki 84. Z leve proti desni si sledijo glede na ustreznost namenskih funkcij (poves in nato volumen nosilca).

4.4.5 Komentar rezultatov optimizacije z večkratno namensko funkcijo

Tudi tokrat se pri uporabi večkratne namenske funkcije pojavi vprašanje, kako vrednotiti dobljene rešitve. Rešitve, ki predstavljajo Pareto optimum (slika 84), so namreč nedominantne. Celotna optimizacija je torej nabor vseh možnih rešitev (2304 različnih primerov) zmanjšala na peščico končnih variant, med katerimi uporabnik lažje izbere »optimalno«. Kriterij po katerem izberemo končno obliko prečnega prereza nosilca, je lahko sedaj odvisen tudi od dejavnikov, ki v optimizaciji niso bili zavzeti (tehnologija izvedbe, cena materiala itd.).

5 ZAKLJUČEK

V diplomski nalogi sem predstavil možnosti uporabe sodobnih računalniških orodij za modeliranje geometrije 3D in implementacijo oziroma združevanje funkcionalnosti le-teh s programom za računanje konstrukcij po metodi končnih elementov.
Uporabil sem orodje Rhinoceros, kjer zapisi geometrije temeljijo na matematičnem zapisu NURBS, ki postopek modeliranja elementov konstrukcij razširijo z možnostmi modeliranja povsem poljubnih (organskih) linij in oblik. Podrobnejši pregled teoretičnih osnov NURBS se je izkazal za zelo koristnega, saj omogoča boljše razumevanje načina delovanja programa in hitrejše uvajanje v delo s programom.

Skupaj s postopkom parametrizacije v okolju Grasshopper (dodatek za Rhinoceros) ter razpoložljivim vmesnikom za povezavo s programom za računanje konstrukcij z metodo končnih elementov celoten proces nekoliko odstopa od ustaljenih postopkov zasnove, modeliranja in analize konstrukcij. Povprečen uporabnik lahko kljub temu hitro osvoji način podajanja geometrije v dokaj intuitivnem vizualnem programskem jeziku. Z razmeroma majhno količino vhodnih podatkov lahko parametrično podani model omogočajo izračun velikega števila geometrij konstrukcij po metodi končnih elementov brez dodatnega (zamudnega) spreminjanja posameznih iteracij (geometrije) modela.

Iskanja optimalne rešitve v množici parametrično podanih modelov/rešitev, sem se lotil z uporabo genetskih algoritmov v okolju Rhinoceros. Potrebno je razumevanje prednosti in pomanjkljivosti takega iskanja rešitev. Že dober začetni približek lahko namreč občutno skrajša celoten postopek iskanja, ki ga glede na število možnih rešitev lahko primerjamo z »iskanjem igle v ogromni bali sena«. Kriterij za določitev optimalne rešitve je lahko po zaslugi dvosmerne povezave med Rhinoceros in SAP2000 določen v obliki poljubnih robnih pogojev notranjih statičnih količin, pomikov ali pa je povsem drugačne narave (izvedba, cena itd.).
Tako zastavljen potek računanja konstrukcij brez dvoma nudi neizmerne možnosti izboljšav nosilnih sistemov konstrukcij.

Celoten delotok sem uporabil na štirih poenostavljenih primerih konstrukcij oziroma posameznih delih konstrukcije, pri čemer so bili primeri izbrani tako, da je tip konstrukcije (linijske in ploskovne konstrukcije) pri primerih služil ponazoritvi zmožnosti takšne parametrične zasnove. Žal sem bil pri izbiri in predstavitvi primerov primoran določene aspekte projektiranja (prekomerno) zapostaviti oziroma v celoti izpustiti. Na področju podajanja obtežbe snega, obtežbe vetra, kontrole uklona konstrukcije in podobnih izredno važnih faktorjev, ki vplivajo na končno obliko konstrukcije, so primeri obravnavani zgolj na konceptualni ravni. Nekatere izmed zgoraj naštetih je za parametrično podane modele namreč izredno težko vključiti v taki meri, kot so podani v standardih Eurocode.

Med pisanjem pisanja naloge sem bil pogosto soočen s »hrošči« in nedelovanjem oziroma kompatibilitetnimi problemi programov, vtičnikov in dodatkov. Razen osnovnih dveh programov Rhinoceros in SAP2000 so namreč vsa uporabljena orodja še v fazi razvoja. Prehajanje med posameznimi verzijami in nedelujoči algoritmi sta zaenkrat dva izmed glavnih razlogov, zakaj je proces parametrične zasnove, modeliranja in analize konstrukcij smiselno uporabiti zgolj na posameznih delih konstrukcije.

Tovrstnim težavam se lahko sicer v določeni meri zoperstavimo z uvedbo lastnih (namenskih) komponent v okolju Grasshopper, ki prav tako omogočajo interaktivno komunikacijo programov Rhinoceros in SAP2000 preko vmesnika OAPI. Tak primer sem skušal ponazoriti s primerom optimizacije lepljenega lameliranega nosilca. Namenske komponente so pravzaprav »ad hoc« rešitve problemov, a imajo to prednost, da vsebino in delovanje algoritma poznamo v celoti (česar pri ostalih uporabljenih orodjih ne moremo trditi).

Uporabljen proces parametrične zasnove, modeliranja in analize konstrukcij s sodobnimi programskih orodji je torej relativno nov pristop in po mojem mnenju zaenkrat še ni preizkušen v taki meri, da bi bili tako dobljeni rezultati direktno uporabni za samo dimenzioniranje elementov. Kontrola rezultatov je obvezna. Obenem pa nedvomno menim, da je tak pristop sestavni del prihodnjih generacij programskih orodij za modeliranje konstrukcij.

SLOVAR MANJ ZNANIH BESED IN TUJK

Hipervolumski indikator – hypervolume indicator znan tudi kot Lebesguova méra. Méra na množici je v matematični analizi sistematični način prireditve števila vsaki njeni ustrezni podmnožici, ki ga intuitivno tolmačimo kot njeno velikost. V tem smislu je mera posplošitev koncepta dolžine, ploščine in prostornine. V okviru evolucijskih algoritmov z večkratnimi namenskimi funkcijami je hipervolumski indikator uporabljen kot méra na množici s katero ocenjujemo ustreznost iskalnih algoritmov in vodimo iskanje. Povzeto po [38] in [39].

Baricenter – masno središče**,** določeno kot uteženo povprečje vseh točk telesa pri čemer je utež sorazmerna masi točke [40].

VIRI

Uporabljeni viri:

[1] Piegl, L., Tiller, W. 1997. The NURBS book, 2nd edition. Berlin idr. Springer: str.1-139. http://diyhpl.us/~bryan/papers2/cad/the-nurbs-book.pdf (Pridobljeno 7. 1. 2014.)

[2] Mesh Edit: vtičnik programa Rhinoceros za razširitev nabora komponent. 2011. http://www.food4rhino.com/project/meshedittools?etx (Pridobljeno 30. 4. 2014.)

[3] Schuhmacher, J. 2013. TT Toolbox: vtičnik programa Rhinoceros za razširitev nabora komponent. http://www.food4rhino.com/project/tttoolbox?etx (Pridobljeno 12. 8. 2014.)

[4] Robert McNeel & Associates. 2011. GHPython: vtičnik programa Rhinoceros za vpeljavo programskega jezika Python. http://www.food4rhino.com/project/ghpython?etx (Pridobljeno 11. 5. 2014.)

[5] Robert McNeel & Associates. 2014. Grasshopper Assembly: knjižnica ukazov za programiranje Grasshopper komponent v različnih programskih jezikih. https://visualstudiogallery.msdn.microsoft.com/9e389515-0719-47b4-a466-04436b491cd6 (Pridobljeno 12. 12. 2014.)

[6] Microsoft. 2013. Visual Studio Community 2013: razvojno okolje namenjeno razvoju aplikacij na ogrodju .NET. http://www.visualstudio.com/ (Pridobljeno 12. 12. 2014.)

[7] Robert McNeel & Associates. 2011. Rhinoceros: računalniški program za tridimenzionalno modeliranje. http://www.rhino3d.com/ (Pridobljeno 09. 12. 2013.)

[8] Rhinoceros 3D. 2011. http://en.wikipedia.org/wiki/Rhinoceros_3D (Pridobljeno 15. 4. 2014.)

[9] Grasshopper: vtičnik programa Rhinoceros za parametrično zasnovo in modeliranje. 2011. http://www.grasshopper3d.com (Pridobljeno 22. 4. 2014.)

[10] Grasshopper primer user guide. 2009. http://www.liftarchitects.com/s/Grasshopper-Primer_Second-Edition_090323-rn88.pdf (Pridobljeno 10. 1. 2014.)

[11] Genetic algorithm. 2014. http://en.wikipedia.org/wiki/Genetic_algorithm (Pridobljeno 10. 6. 2014.)

[12] Evolutionary algorithm. 2014. http://en.wikipedia.org/wiki/Evolutionary_algorithm (Pridobljeno 10. 6. 2014.)

[13] Naravni izbor. 2013. http://sl.wikipedia.org/wiki/Naravni_izbor5 (Pridobljeno 10. 6. 2014.)

[14] Genetski algoritmi - teorija in praksa. 2014. http://www.fmf.uni-lj.si/~skreko/Pouk/ipo_2010-11/Prezentacije/GregorPapa-predstavitev.pdf (Pridobljeno 10. 10. 2014.)

[15] Rutten, D. 2011. Galapagos. David Rutten blog, objavljeno 4.3.2011. http://ieatbugsforbreakfast.wordpress.com/ (Pridobljeno 2. 6. 2014.)

[16] Slika modela prostora namenske funkcije dveh spremenljivk. 2011. https://ieatbugsforbreakfast.wordpress.com/2011/03/04/epatps01/ (Pridobljeno 2. 6. 2014.)

[17] Slika modela prostora namenske funkcije dveh spremenljivk s prikazom populacije izvrednotene namenske funkcije. 2011. https://ieatbugsforbreakfast.wordpress.com/2011/03/04/epatps01/ (Pridobljeno 2. 6. 2014.)

[18] Slika strnjenosti druge generacije izvrednotene namenske funkcije. 2011. https://ieatbugsforbreakfast.wordpress.com/2011/03/04/epatps01/ (Pridobljeno 2. 6. 2014.)

[19] Slika ponazoritve faktorja homogamije. 2011. https://ieatbugsforbreakfast.wordpress.com/2011/03/04/coupling-algorithms/ (Pridobljeno 2. 6. 2014.)

[20] Computers and Structures, Inc. 1996. SAP2000: računalniški program za analizo in zasnovo konstrukcij po metodi končnih elementov. http://www.csiamerica.com/sap2000 (Pridobljeno 30. 02. 2014.)

[21] Mirtschin, J. 2009. GeometryGym: vtičnik programa Rhinoceros za izmenjavo datotek z ostalimi programskimi orodji. https://geometrygym.wordpress.com/downloads/ (Pridobljeno 17. 11. 2014.)

[22] Mirtschin, J. 2014. GeometryGym license&issues. Message to: Soklič, R. 30.1.2014. Osebna komunikacija.

[23] Computers and Structures, Inc. 2011. SAP2000 v16.0.0. CSi_OAPI_Documentation.chm: datoteka v kateri so naštete in opisane funkcije programskega vmesnika SAP2000.

[24] Issa, R. 2011. Paneling Tools: vtičnik programa Rhinoceros za racionalizacijo NURBS površin. http://v5.rhino3d.com/group/panelingtools (Pridobljeno 30. 4. 2014.)

[25] Piacentino, G. 2012. Weaverbird: vtičnik programa Rhinoceros za obravnavo mrež. http://www.giuliopiacentino.com/weaverbird/ (Pridobljeno 30. 4. 2014.)

[25] Piker, D. 2011. Kangaroo physics: vtičnik programa Rhinoceros za simuliranje vzajemnega obnašanja geometrije. http://www.food4rhino.com/project/kangaroo?etx (Pridobljeno 15. 8. 2014.)

[26] Vierling, R. 2014. Octopus: vtičnik programa Rhinoceros za optimiziranje problemov z večkratno sposobnostno funkcijo. http://www.food4rhino.com/project/octopus?etx (Pridobljeno 12. 8. 2014.)

[27] Zitzler, E. et al. 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm http://www.kddresearch.org/Courses/Spring-2007/CIS830/Handouts/P8.pdf (Pridobljeno 27. 12. 2014.)

[28] Bader, J., Zitzler, E. 2008. Hype: An algorithm for fast hypervolume-based many-objective optimization. http://www.tik.ee.ethz.ch/sop/publicationListFiles/bz2008a.pdf (Pridobljeno 27. 12. 2014.)

[29] Pareto optimum. 2014. http://en.wikipedia.org/wiki/Pareto_efficiency (Pridobljeno 20. 12. 2014.)

[30] Farshad, M. 1992. Design and analysis of shell structures. Dordrecht, Kluwer academic publishers: 415 str.

[31] Slika American Air Museum v Duxfordu. 1997. http://www.fosterandpartners.com/projects/american-air-museum/ (Pridobljeno 20. 5. 2014.)

[31] Understanding geometry. 2014. http://www.grasshopper3d.com/forum/topics/understanding-geometry (Pridobljeno 8. 8. 2014.)

[32] Diferencialna geometrija ploskev. 2014. http://en.wikipedia.org/wiki/Differential_geometry_of_surfaces (Pridobljeno 2. 8. 2014.)

[33] Computers and structures, inc. 2013. CSI analysis reference manual: str. 160, 161,162, 218, 219

[34] Catmull, E., Clark, J. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. https://truesculpt.googlecode.com/hg/Doc/CatmullClark_SDSurf.pdf (Pridobljeno 10. 12. 2014.)

[35] Povzetek metode Catmull-Clark. 2014. http://en.wikipedia.org/wiki/Catmull%E2%80%93Clark_subdivision_surface (Pridobljeno 10. 12. 2014.)

[36] Gabriel, J. F (ur.). 1997. Beyond the Cube: The Architecture of Space Frames and Polyhedra, John Wiley, NewYork: str. 211-227.

[37] Bernstein polynomials. 2014. http://www.wolframalpha.com/input/?i=bernstein+polynomial (Pridobljeno 20. 12. 2014.)

[38] Bezier curve. 2014. http://www.wolframalpha.com/input/?i=B%C3%A9zier+curve&lk=1&a=ClashPrefs_*MathWorld.BezierCurve- (Pridobljeno 20. 12. 2014.)

[38] Hypervolume indicator. 2014. http://iridia.ulb.ac.be/~manuel/hypervolume (Pridobljeno 22. 12. 2014.)

[39] Mera na množici. 2014. http://sl.wikipedia.org/wiki/Mera_(matematika) (Pridobljeno 26. 12. 2014.)

[40] Baricenter. 2013. http://sl.wikipedia.org/wiki/Masno_sredi%C5%A1%C4%8De (Pridobljeno 2. 12. 2014.)

Ostali viri:

Khabazi, M. 2009. Algorithmic modeling with Grasshopper. http://download.mcneel.com/s3/mcneel/grasshopper/1.0/docs/en/AlgorithmicModelling.pdf (Pridobljeno 24.4.2014.)

Daniels, J. et al. 2008. Quadrilateral Mesh Simplification. http://www.sci.utah.edu/~csilva/papers/siggraph-asia2008.pdf (Pridobljeno 24.7.2014.)

Stavric, M., Marina, O. 2011. Parametric modeling for advanced architecture http://www.universitypress.org.uk/journals/ami/19-794.pdf (Pridobljeno 9.5.2014.)

Šebenik, Ž., 2013. Testiranje uporabnosti programskega vmesnika SAP2000. Diplomska naloga. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo (samozaložba Ž. Šebenik): 87 str.

Kragelj, A., 2013. Palične lupinaste konstrukcije. Diplomska naloga. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo (samozaložba A. Kragelj): 35 str.

Turk, J., 2013. Armiranobetonske lupinaste konstrukcije. Magistrska naloga. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo (samozaložba J.Turk): 134 str.

Computers and Structures, Inc. 2014. Optimized Modeling and Design of Structures using SAP2000. https://wiki.csiamerica.com/display/doc/Optimized+Modeling+and+Design+of+Structures+using+SAP2000+-+ARCHIVAL (Pridobljeno 23. 2. 2014.)

Cook, R.D. et al. 2002. Concepts and applications of finite element analysis, 4th edition, J. Wiley: 733 str.

Sitler, B. 2010. Guide to creating custom Grasshopper 0.6.X components. http://www.grasshopper3d.com/forum/attachment/download?id=2985220%3AUploadedFi38%3A57100 (Pridobljeno 22. 12. 2014.)

Robert McNeel & Associates. 2007. RhinoScript 101. http://download.mcneel.com/download.asp?id=RhinoScript101&language= (Pridobljeno 1. 4. 2014.)

Schneider, P.J. 2010. NURB Curves: A Guide for the Uninitiated. http://www.mactech.com/articles/develop/issue_25/schneider.html (Pridobljeno 28. 12. 2013.)

Senske, N. 2012. Serija video navodil za uporabo programa Grasshopper. https://www.youtube.com/user/nsenske?feature=watch (Pridobljeno 30. 12. 2013.)

Rutten, D. 2014. Navigating multi-dimensional landscapes in foggy weather as an analogy for generic problem solving. https://ieatbugsforbreakfast.files.wordpress.com/2014/08/manuscript-david-rutten.pdf (Pridobljeno 29. 11. 2014.)

Bronstein, I.N. et al. 2009. Matematični priročnik, popravljena izd. 1.natis, Tehniška založba Slovenije: str. 757

SEZNAM PRILOG

PRILOGA A

A.I Seznam vseh uporabljenih komponent za primer modela plošče iz poglavja 4.1

ŠT.:	OZNAKA KOMPONENTE:	PRIPADA:
1	Rectangle	Grasshopper
2	Boolean Toggle (Toggle)	
3	Number Slider ()	
4	Settings Custom (Custom Mesh Settings)	
5	Mesh Brep (Mesh)	
6	Deconstruct Mesh (DeMesh)	
7	Mesh NakedEdge (NakedEdge)	Mesh Edit
8	Merge ()	Grasshopper
9	List Item (Item)	
10	Create Set (CSet)	
11	Construct Point (Pt)	
12	Evaluate Surface (EvalSrf)	
13	Amplitude (Amp)	
14	Unit Z (Z)	
15	Closest Point (CP)	
16	Vector Display (VDis)	
17	Panel ()	
18	ggSAPCreateOrFindPoint(ggPoint)	GeometryGym
19	ggDSPPointAtts (ggSA)	
20	ggSAPPointRestraint (ggNR)	
21	ggSAPMeshConvertFiniteElements (ggMFE)	
22	ggSAPCreateShellProp (ggAP)	
23	ggSAPCreateConcrete (ggConc)	
24	ggSAPLoad2dFace (ggFL)	
25	ggSAPStaticLoadCase (ggSLC)	
26	ggSAPLoadPattern (ggLP)	
27	ggSAPLoadNode (ggNL)	
28	ggSAPQueryNodeReactionForce (ggQueryNodeReactF)	
29	ggSAPQueryResult2doption (ggQueryResult2dOpt)	
30	ggSAPQueryPrincipalStress (ggQueryPrinStrss)	
31	ggSAPSolver (ggS)	
32	ggSAPResults2dPrincipalStressDecompose (ggResPrincStressDecompose)	
33	ggSAPResultNodeSetDecompose (ggResNodeSetDcomp)	
34	Cull index (Cull i)	Grasshopper
35	Deconstruct Vector (DeVec)	
36	Sort List (Sort)	
37	Bounds (Bnd)	
38	Deconstruct Domain (DeDomain)	
39	Find similar member (FSim)	
40	List Item (Item)	
41	Vector XYZ (Vec)	
42	Number (Num)	
43	Cull Pattern (Cull)	
44	Dispatch ()	
45	Galapagos Listener (GalapList)	TT Toolbox
46	Octopus (Octopus)	Octopus
47	Galapagos Evolutionary Solver	Grasshopper
48	Patch ()	

PRILOGA B

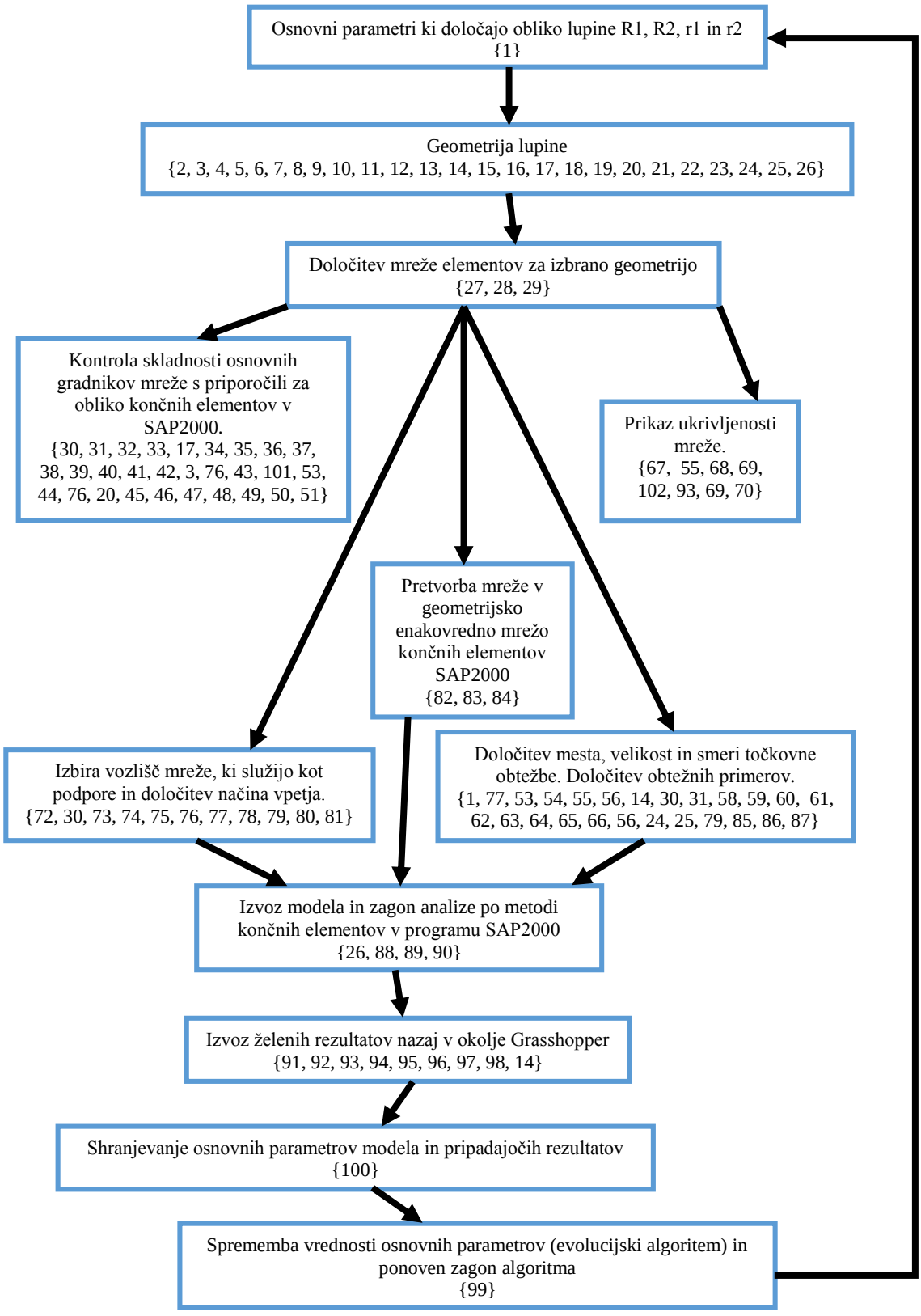
B.I Seznam vseh uporabljenih komponent za primer modela dvojno ukrivljene lupine iz poglavja 4.2.

ŠT.:	OZNAKA KOMPONENTE:	PRIPADA:
1	Number Slider ()	Grasshopper
2	VB Script (VB)	
3	Smaller Than (Smaller)	
4	Circle (Cir)	
5	Ellipse ()	
6	Evaluate Curve (Eval)	
7	Sweep 1 (Swp1)	
8	Area ()	
9	Extrude (Extr)	
10	Unit Y (Y)	
11	Split Brep (Split)	
12	Distance (Dist)	
13	Sort List (Sort)	
14	List Item (Item)	
15	YZ Plane (YZ)	
16	Plane Through Shape (PxS)	
17	Dispatch ()	
18	Brep ()	
19	Move ()	
20	Vector 2Pt (Vec2Pt)	
21	Rotate ()	
22	Deconstruct Brep (DeBrep)	
23	Deconstruct (pDecon)	
24	Unit Z (Z)	
25	Amplitude (Amp)	
26	Boolean Toggle (Toggle)	
27	Settings Custom (Custom Mesh Settings)	
28	Mesh Brep (Mesh)	
29	Weaverbird's Catmull-Clark Subdivision (wbCatmullClark)	Weaverbird
30	Deconstruct Mesh (DeMesh)	Grasshopper
31	Deconstruct Face (DeFace)	
32	Face Boundaries (FaceB)	
33	Similarity (Similar)	
34	Curve()	
35	Explode()	
36	Evaluate Curve (Eval)	
37	Split List (Split)	
38	Line (Ln)	
39	Sort List (Sort by length)	
40	Division (A/B)	
41	Larger Than (Larger)	
42	List Length (Lng)	
43	Simplify Tree (Simplify)	
44	End Points (End)	
45	Construct Plane (Pl 3)	
46	Deconstruct Plane (DePlane)	
47	Angle ()	
48	Degrees (Deg)	

49	Vector Display Ex (VDisEx)	Grasshopper
50	Mass Addition (MA)	
51	Equality (Equals)	
52	Construct Point (Pt)	
53	Mesh Explode (Explode)	
54	Evaluate Surface (EvalSrf)	
55	Surface Closest Point (Srf CP)	
56	Mesh Closest Point (MeshCP)	
57	Delete Faces (DeleteF)	
58	Series ()	
59	Duplicate Data (Dup)	
60	Shift List (Shift)	
61	Mesh Triangle (Triangle)	
62	Construct Mesh (ConMesh)	
63	Mesh Join (MJoin)	
64	Mesh WeldVertices (weldVertices)	
65	Face Normals (FaceN)	
66	Weaverbird's Join Meshes and Weld (wbJoin)	Weaverbird
67	Decompose (mComp)	Grasshopper
68	Surface Curvature (Curvature)	
69	Domain Components (DomComp)	
70	Gradient ()	
71	Mesh ()	
72	Mesh NakedEdge (NakedEdge)	Mesh Edit
73	Merge ()	Grasshopper
74	Create Set (CSet)	
75	Planar ()	
76	Cull Pattern (Cull)	
77	Point (Pt)	
78	ggProximity Points (ggPP)	GeometryGym
79	ggSAPCreateOrFindPoint(ggPoint)	
80	ggDSPPointAtts (ggSA)	
81	ggSAPPointRestraint (ggNR)	
82	ggSAPMeshConvertFiniteElements (ggMFE)	
83	ggSAPCreateShellProp (ggAP)	
84	ggSAPCreateConcrete (ggConc)	
85	ggSAPStaticLoadCase (ggSLC)	
86	ggSAPLoadPattern (ggLP)	
87	ggSAPLoadNode (ggNL)	
88	ggSAPQueryResult2doption (ggQueryResult2dOpt)	
89	ggSAPQueryPrincipalStress (ggQueryPrinStrss)	
90	ggSAPSolver (ggS)	
91	ggSAPResults2dPrincipalStressDecompose (ggResPrincStressDecompose)	
92	Cull index (Cull i)	Grasshopper
93	Bounds (Bnd)	
94	Deconstruct Domain (DeDomain)	
95	Find similar member (FSim)	
96	Face Normals (FaceN)	
97	Vector Display (VDis)	
98	Number (Num)	
99	Galapagos Evolutionary Solver ()	

100	Galapagos Listener (GalapList)	TT Toolbox
101	Panel ()	Grasshopper
102	Expression ()	

B.II Potek algoritma za model lupine iz poglavja 4.2. Števila v zavitih oklepajih predstavljajo zaporedne številke uporabljenih komponent (B1, B2, B3).



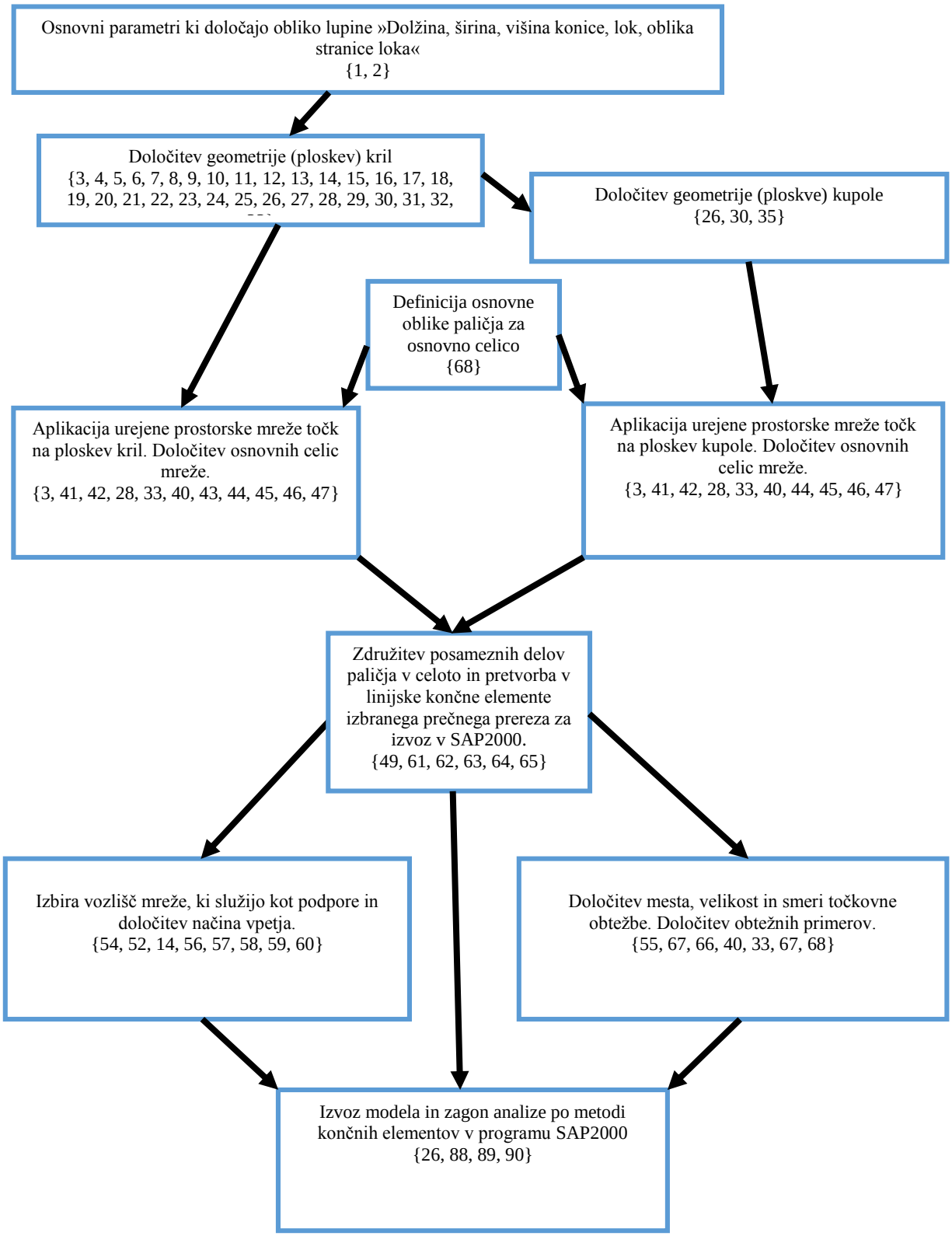
PRILOGA C

C.I Seznam vseh uporabljenih komponent za primer modela paviljona iz poglavja 4.3

ŠT.:	OZNAKA KOMPONENTE:	PRIPADA:
1	Number Slider ()	Grasshopper
2	Graph Mapper (Graph)	
3	Range ()	
4	Multiplication (AxB)	
5	Series()	
6	Construct Point (Pt)	
7	Interpolate (IntCrv)	
8	Line SDL (Line)	
9	Mirror ()	
10	XZ Plane (XZ)	
11	Rotate Axis (RotAz)	
12	End Points (End)	
13	Join Curves (Join)	
14	Deconstruct (pDecon)	
15	Minimum (Min)	
16	Arc 3Pt (Arc)	
17	Sweep2 (Swp2)	
18	Polar Array (ArrPolar)	
19	Radians (Rad)	
20	Area	
21	Vector 2Pt (Vec2Pt)	
22	XY Plane (XY)	
23	Sphere (Sph)	
24	Box Corners	
25	Evaluate Curve (Eval)	
26	Deconstruct Vector (DeVec)	
27	Addition (A+B)	
28	Move()	
29	Rotate()	
30	List item (Item)	
31	Line (Ln)	
32	Evaluate Length (Eval)	
33	Amplitude (Amp)	
34	Deconstruct Brep (DeBrep)	
35	Network Surface (NetSurf)	
36	Brep Edges (Edges)	
37	Solid Union (SUnion)	
38	Unit Z (Z)	
39	Center Grid (ptCenter)	Paneling Tools
40	Surface Parameter (ptSrfParam)	
41	Cellulate (ptCell)	
42	Panel 3D Grid (ptMPanel3D)	
43	Length (Len)	Grasshopper
44	Equality (Equals)	
45	Dispatch()	
46	Pipe()	
47	removeDuplicateLines (dupLn)	Kangaroo
48	Remove Duplicate Points (CullDupPoints)	Grasshopper
49	Null Item (Null)	

50	Cull Pattern (Cull)	
51	ggProximity Points (ggPP)	GeometryGym
52	Extremes (X-tremez)	Grasshopper
53	Cull Duplicates (CullPt)	
54	Sort List (Sort)	
55	Sub List (SubSet)	
56	ggSAPCreateOrFindPoint(ggPoint)	GeometryGym
57	ggDSPPointAtts (ggSA)	
58	ggSAPPointRestraint (ggNR)	
59	ggHenerateCHS (ggCHS)	
60	ggSAPCreate Steel (ggSteel)	
61	ggSAPCreateSectionProp (ggSP)	
62	ggSAPCreateFrame (ggFrame)	
63	ggSAPFrameAtts (ggBA)	
64	ggSAPStaticLoadCase (ggSLC)	
65	ggSAPLoadPattern (ggLP)	
66	ggSAPLoadNode (ggNL)	
67	ggSAPBakeModel (ggBake)	
68	Panel ()	Grasshopper

C.II Potek algoritma za model paviljona iz poglavja 4.3. Števila v zavitih oklepajih predstavljajo zaporedne številke uporabljenih komponent (C1, C2).

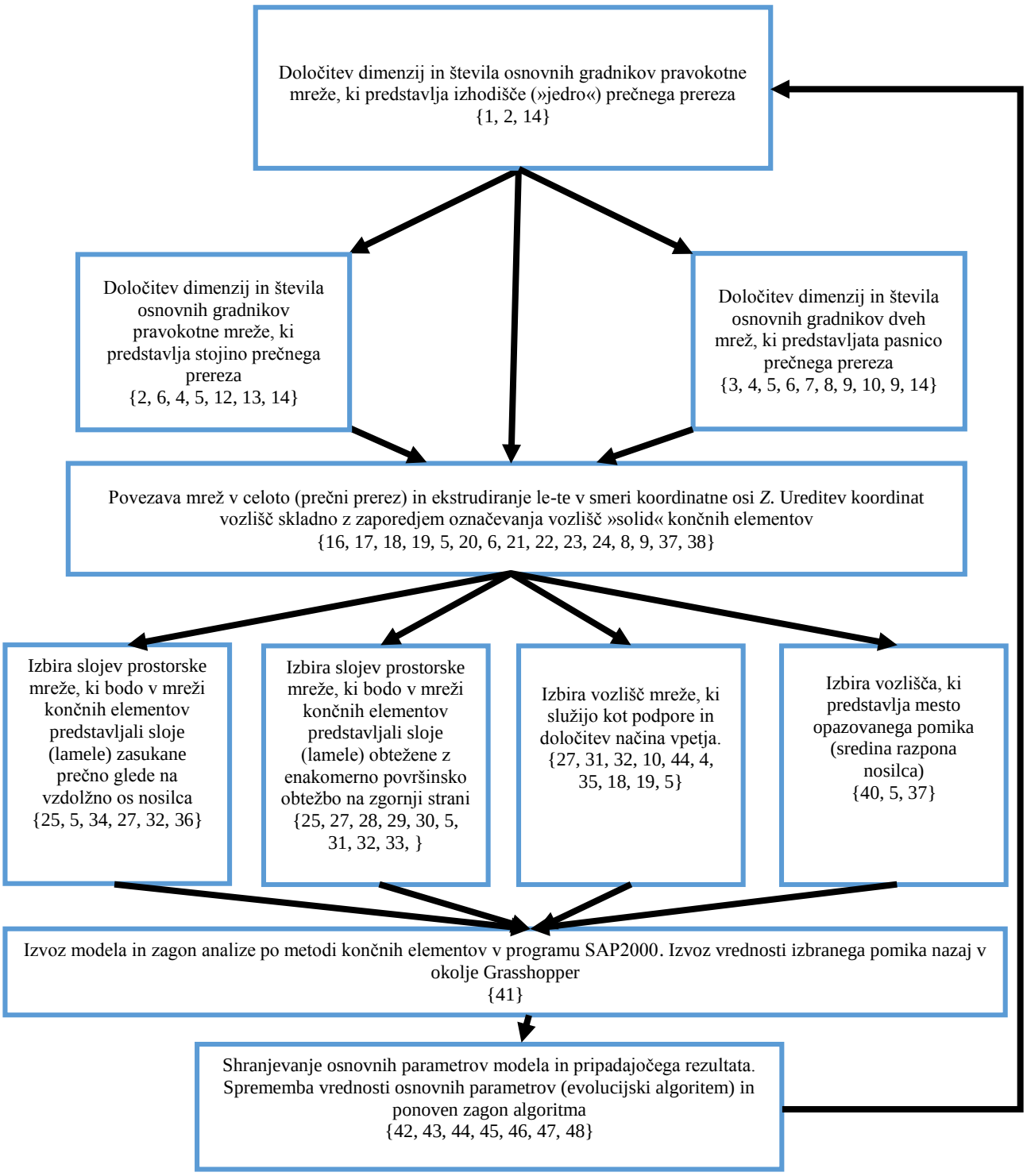


PRILOGA D

D.I Seznam vseh uporabljenih komponent za primer modela optimizacije nosilca iz točke 4.4

ŠT.:	OZNAKA KOMPONENTE:	PRIPADA:
1	Number Slider ()	Grasshopper
2	Rectangular (RecGrid)	
3	Tree Statistics (TStat)	
4	List Length (Lng)	
5	List Item (Item)	
6	Tree Branch (Branch)	
7	Unit X (X)	
8	Amplitude (Amp)	
9	Linear Array (ArrLinear)	
10	Sort Points (Sort Pt)	
11	Tree Item (Item)	
12	Vector 2Pt (Vec2Pt)	
13	Move ()	
14	Mesh FromPoints (FromPoints)	
15	Mesh UnifyNormals (UnifyNormals)	
16	Mesh WeldVertices (weldVertices)	Mesh Edit
17	Mesh Join (MJoin)	Grasshopper
18	Mesh Explode (Explode)	
19	Deconstruct Mesh (DeMesh)	
20	Rectangle 2Pt (Rec 2Pt)	
21	Deconstruct Rectangle (DRec)	
22	Domain Box (Box)	
23	Construct Domain (Dom)	
24	Unit Z (Z)	
25	Volume ()	
26	Mass Addition (MA)	
27	Deconstruct (pDecon)	
28	Bounds (Bnd)	
29	Deconstruct Domain (DeDomain)	
30	Find similar member (FSim)	
31	Equality (Equals)	
32	Cull Pattern (Cull)	
33	Item Index (Index)	
34	Split List (Split)	
35	Remove Duplicate Points (CullDupPoints)	
36	Member Index (MIndex)	
37	Evaluate Curve (Eval)	
38	Deconstruct Brep (DeBrep)	
39	Replace Items (Replace)	
40	Line ()	
41	»T nosilec ()«	*»GLEJ PRILOGO E«*
42	Galapagos ()	Grasshopper
43	Galapagos Listener (GalapList)	TT Toolbox
44	Boolean Toggle ()	Grasshopper
45	Sort List (Sort)	
46	Panel ()	
47	Absolute (Abs)	
48	Octopus ()	Octopus

D.II Potek algoritma za primer modela optimizacije nosilca iz točke 4.4. Števila v zavitih oklepajih predstavljajo zaporedne številke uporabljenih komponent iz priloge D.I.



D.III Skripta v programskem jeziku Visual Basic.NET za interaktivni prenos modela iz okolja Grasshopper v SAP2000 preko komponente »T_nosilec«.

Programirano v programskem okolju Visual Studio Community 2013 s souporabo »Grasshopper Assembly« vtičnika.

```
Imports System.Collections.Generic
Imports SAP2000v16
Imports Grasshopper.Kernel
Imports Rhino.Geometry
Imports Grasshopper.Kernel.Types
Imports Grasshopper.Kernel.Data


Public Class TnosilecComponent
  Inherits GH_Component
  ''' <summary>
  ''' Each implementation of GH_Component must provide a public
  ''' constructor without any arguments.
  ''' Category represents the Tab in which the component will appear,
  ''' Subcategory the panel. If you use non-existing tab or panel names,
  ''' new tabs/panels will automatically be created.
  ''' </summary>
  Public Sub New()
        MyBase.New("T nosilec", "T nosilec", _
            "Description", _
            "Category", "Subcategory")
  End Sub

  ''' <summary>
  ''' Registers all the input parameters for this component.
  ''' </summary>
    Protected Overrides Sub RegisterInputParams(pManager As GH_Component.GH_InputPa-
ramManager)
        pManager.AddGenericParameter("Mreza elementov", "seznam", "seznam vozlisc
koncnih elementov", GH_ParamAccess.tree)
        pManager.AddGenericParameter("Koordinate drsno podprtih tock", "P_Podpore",
"seznam drsnih podpornih tock", GH_ParamAccess.tree)
        pManager.AddGenericParameter("Koordinate vpeto podprtih tock", "V_Podpore",
"seznam vpetih podpornih tock", GH_ParamAccess.tree)
        pManager.AddGenericParameter("Indeksi obtezenih elementov", "Obt_indeks",
"seznam indeksov obtezenih elementov", GH_ParamAccess.tree)
        pManager.AddGenericParameter("Indeksi zasukanih elementov", "Precni_indeks",
"seznam indeksov zasukanih elementov", GH_ParamAccess.tree)
        pManager.AddGenericParameter("Sredinska tocka", "Poves", "koordinate tocke
kjer kontroliramo poves", GH_ParamAccess.tree)
    End Sub


    ''' <summary>
    ''' Registers all the output parameters for this component.
    ''' </summary>
  Protected Overrides Sub RegisterOutputParams(pManager As GH_Component.GH_OutputPa-
ramManager)
        pManager.AddGenericParameter("Izbran pomik", "POVES", "pomik izbrane tocke",
GH_ParamAccess.tree)
        pManager.AddGenericParameter("Izbrane napetosti", "NAPETOST", "vrednosti nape-
tosti v vozliscih elementa", GH_ParamAccess.tree)

    End Sub
```

```
    ''' <summary>
    ''' This is the method that actually does the work.
    ''' </summary>
    ''' <param name="DA">The DA object can be used to retrieve data from input parame-
ters and
    ''' to store data in output parameters.</param>
    Protected Overrides Sub SolveInstance(DA As IGH_DataAccess)

        Dim seznam As New GH_Structure(Of IGH_Goo)
        Dim Ppodpore As New GH_Structure(Of IGH_Goo)
        Dim Vpodpore As New GH_Structure(Of IGH_Goo)
        Dim Obt_indeks As New GH_Structure(Of IGH_Goo)
        Dim Precni_indeks As New GH_Structure(Of IGH_Goo)
        Dim Poves As New GH_Structure(Of IGH_Goo)

        DA.GetDataTree(Of IGH_Goo)(0, seznam)
        DA.GetDataTree(Of IGH_Goo)(1, Ppodpore)
        DA.GetDataTree(Of IGH_Goo)(2, Vpodpore)
        DA.GetDataTree(Of IGH_Goo)(3, Obt_indeks)
        DA.GetDataTree(Of IGH_Goo)(4, Precni_indeks)
        DA.GetDataTree(Of IGH_Goo)(5, Poves)


        Dim SapObject As New SAP2000v16.SapObject
        Dim SapModel As SAP2000v16.cSapModel
        Dim ret As Long

        'create Sap2000 object
        SapObject = New SAP2000v16.SapObject

        'create SapModel object
        SapModel = SapObject.SapModel


        'start Sap2000 application
        SapObject.ApplicationStart(SAP2000v16.eUnits.kN_m_C, True)


        'initialize model
        ret = SapModel.InitializeNewModel(SAP2000v16.eUnits.kN_m_C)

        'create new blank model
        ret = SapModel.File.NewBlank
```

```
'ODZNACIMO VSE
        ret = SapModel.SelectObj.ClearSelection

        'DODAMO MATERIAL
        ret = SapModel.PropMaterial.SetMaterial("Les", SAP2000v16.eMatType.MATE-
RIAL_NODESIGN)

        'DOLOCIMO ANIZOTROPNE LASTNOSTI MATERIALA
        Dim MyE() As Double
        Dim MyU() As Double
        Dim MyA() As Double
        Dim MyG() As Double

        ReDim MyE(2)
        ReDim MyU(14)
        ReDim MyA(6)
        ReDim MyG(2)
        MyE(0) = 30000
        MyE(1) = 10000
        MyE(2) = 2000
        MyU(0) = 0.2
        MyU(1) = 0.05
        MyU(2) = 0.1
        MyU(3) = 0
        MyU(4) = 0
        MyU(5) = 0
        MyU(6) = 0
        MyU(7) = 0
        MyU(8) = 0.01
        MyU(9) = 0
        MyU(10) = 0
        MyU(11) = 0
        MyU(12) = 0
        MyU(13) = 0
        MyU(14) = 0
        MyA(0) = 0.0000065
        MyA(1) = 0.0000065
        MyA(2) = 0.0000065
        MyA(3) = 0.0000065
        MyA(4) = 0.0000065
        MyA(5) = 0.0000065
        MyG(0) = 1500
        MyG(1) = 2500
        MyG(2) = 8700
        ret = SapModel.PropMaterial.SetMPAnisotropic("Les", MyE, MyU, MyA, MyG)

        'DOLOCIMO LASTNOSTI VZDOLZNIH SOLID ELEMENTOV - material in "materialni koti"
        ret = SapModel.PropSolid.SetProp("Vzdolzno", "Les", 0, 0, 0, True)

        'DOLOCIMO LASTNOSTI PRECNIH SOLID ELEMENTOV - material in "materialni koti"
        ret = SapModel.PropSolid.SetProp("Precno", "Les", 90, 0, 0, True)

        'DODAMO SOLID ELEMENTE IZ SEZNAMA POVZETEGA IZ OKOLJA GRASSHOPPER
        Dim tocka(0) As GH_Point
        Dim x() As Double
        Dim y() As Double
        Dim z() As Double


        For i = 0 To seznam.Branches.Count - 1 'pregled cez vse branche
            ReDim x(7)
            ReDim y(7)
```

```
            ReDim z(7)
            ReDim tocka(seznam.Branch(i).Count)
            For j = 0 To seznam.Branch(i).Count - 1 'pregled cez vse tocke v enem
        branchu
                tocka(j) = seznam.Branch(i).Item(j)
                x(j) = tocka(j).Value.X
                y(j) = tocka(j).Value.Y
                z(j) = tocka(j).Value.Z
                ret = SapModel.SolidObj.AddByCoord(x, y, z, "name")
            Next
        Next
        'DODAMO SAP2000 GROUP Z IMENOM  - SOLIDS
        ret = SapModel.GroupDef.SetGroup("SOLIDS")

        'VSE SOLIDE POSTAVI V GROUP - SOLIDS
        ret = SapModel.SelectObj.All(False)
        ret = SapModel.SolidObj.SetGroupAssign("name", "SOLIDS", False,
SAP2000v16.eItemType.SelectedObjects)

        'POVEZEMO DOLOCENE LASTNOSTI Z GROUP  - SOLIDS
        ret = SapModel.SolidObj.SetProperty("SOLIDS", "Vzdolzno", SAP2000v16.eItem-
Type.Group)


        'SPREMENIMO OZNAKE (label) SOLID ELEMENTOV
        For i = 0 To seznam.DataCount - 1
            ret = SapModel.SolidObj.ChangeName((8 + i * 8).ToString, i.ToString)
        Next

        'DODAMO SAP2000 GROUP - OBT_ELEMENTI
        ret = SapModel.GroupDef.SetGroup("OBT_ELEMENTI")

        'DODAMO SAP2000 GROUP - PRECNI_ELEMENTI
        ret = SapModel.GroupDef.SetGroup("PRECNI_ELEMENTI")

        'iz grasshopperja preberemo indekse obteženih elementov (elementi na zgornji
ploskvi nosilca)
        For i = 0 To Obt_indeks.DataCount - 1
            ret = SapModel.SolidObj.SetGroupAssign(Obt_indeks.DataItem(i).ToString,
"OBT_ELEMENTI")

        Next
        'iz grasshopperja preberemo indekse zasukanih elementov
        For i = 0 To Precni_indeks.DataCount - 1
            ret = SapModel.SolidObj.SetGroupAssign(Precni_indeks.DataItem(i).ToString,
"PRECNI_ELEMENTI")

        Next

        'POVEZEMO DOLOCENE LASTNOSTI Z GROUP  - PRECNI_ELEMENTI
        ret = SapModel.SolidObj.SetProperty("PRECNI_ELEMENTI", "Precno",
SAP2000v16.eItemType.Group)

        'PROSTOLEZECE PODPORE
        ReDim tocka(0)
        Dim Ppodpiranje(5) As Boolean
        For i = 0 To 2
            Ppodpiranje(i) = True
        Next
        For i = 3 To 5
            Ppodpiranje(i) = False
```

```
        Next

        For i = 0 To Ppodpore.DataCount - 1
            tocka(0) = Ppodpore.DataItem(i)
            ret = SapModel.PointObj.AddCartesian(tocka(0).Value.X, tocka(0).Value.Y,
tocka(0).Value.Z, "name", "P" & CStr(i), , False)

            ret = SapModel.PointObj.SetRestraint("P" & CStr(i), Ppodpiranje)
            ret = SapModel.PointObj.SetSpecialPoint("P" & CStr(i), True)
        Next

        'VPETE PODPORE
        ReDim tocka(0)
        Dim Vpodpiranje(5) As Boolean
        For i = 0 To 1
            Vpodpiranje(i) = True
        Next
        For i = 2 To 5
            Vpodpiranje(i) = False
        Next

        For i = 0 To Vpodpore.DataCount - 1
            tocka(0) = Vpodpore.DataItem(i)
            ret = SapModel.PointObj.AddCartesian(tocka(0).Value.X, tocka(0).Value.Y,
tocka(0).Value.Z, "name", "V" & CStr(i), , False)

            ret = SapModel.PointObj.SetRestraint("V" & CStr(i), Vpodpiranje)
            ret = SapModel.PointObj.SetSpecialPoint("V" & CStr(i), True)
        Next


        'TOCKA POMIKA
        ReDim tocka(0)
        tocka(0) = Poves.DataItem(0)
        ret = SapModel.PointObj.AddCartesian(tocka(0).Value.X, tocka(0).Value.Y,
tocka(0).Value.Z, "name", "Poves", , False)
        ret = SapModel.PointObj.SetSpecialPoint("Poves", True)

        'OBTEZBA DODAMO LOAD PATTERN -Povrsinska
        ret = SapModel.LoadPatterns.Add("Povrsinska", SAP2000v16.eLoadPattern-
Type.LTYPE_LIVE, 0.0)


        'OBTEZBA - dodamo povrsinsko obtezbo na zgornji strani nosilca(face 3)
        ret = SapModel.SolidObj.SetLoadSurfacePressure("OBT_ELEMENTI", "Povrsinska",
3, 10, , , SAP2000v16.eItemType.Group)


        'SHRANIMO IN POZENEMO ANALIZO
        ret = SapModel.Analyze.SetRunCaseFlag("MODAL", False)
        ret = SapModel.Analyze.SetRunCaseFlag("DEAD", False)
        ret = SapModel.File.Save("C:\Users\Rok\Documents\SAP2000\SAPapi\T_nosilec\Tno-
silec.sdb")
        ret = SapModel.Analyze.RunAnalysis

        'REZULTATI()
        'ODZNACIMO VSE OBTEZNE PRIMERE IN KOMBINACIJE
        ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput

        'DOLOCIMO OBTEZNI PRIMER
        ret = SapModel.Results.Setup.SetCaseSelectedForOutput("Povrsinska")
```

```
        'Dim NumberResults As Long
        Dim PointElm() As String
        Dim LoadCase() As String
        Dim StepType() As String
        Dim StepNum() As Double
        Dim Obj() As String
        Dim Elm() As String
        Dim NumberResults As Long
        Dim U1() As Double
        Dim U2() As Double
        Dim U3() As Double
        Dim R1() As Double
        Dim R2() As Double
        Dim R3() As Double



        ' IZPISI REZULTATE
        ret = SapModel.Results.JointDispl("Poves", SAP2000v16.eItemTypeElm.ObjectElm,
NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)

        DA.SetDataList(0, U2) 'IZPISI REZULTATE (pomik U2 za tocko z imenom Poves) V
GRASSHOPPER


        Dim S11() As Double
        Dim S22() As Double
        Dim S33() As Double
        Dim S12() As Double
        Dim S13() As Double
        Dim S23() As Double
        Dim SMax() As Double
        Dim SMid() As Double
        Dim SMin() As Double
        Dim SVM() As Double
        Dim DirCosMax1() As Double
        Dim DirCosMax2() As Double
        Dim DirCosMax3() As Double
        Dim DirCosMid1() As Double
        Dim DirCosMid2() As Double
        Dim DirCosMid3() As Double
        Dim DirCosMin1() As Double
        Dim DirCosMin2() As Double
        Dim DirCosMin3() As Double

        ret = SapModel.Results.SolidStress("0", SAP2000v16.eItemTypeElm.Element,
NumberResults, Obj, Elm, PointElm, LoadCase, StepType, StepNum, S11, S22, S33, S12,
S13, S23, SMax, SMid, SMin, SVM, DirCosMax1, DirCosMax2, DirCosMax3, DirCosMid1, Dir-
CosMid2, DirCosMid3, DirCosMin1, DirCosMin2, DirCosMin3)

        DA.SetDataList(1, S11) 'IZPISI REZULTATE (napetosti S11 za solid element z
oznako/label 0) V GRASSHOPPER

        'ODKLENEMO MODEL
        ret = SapModel.SetModelIsLocked(False)
```

```
        'prekini povezavo
        SapObject.ApplicationExit(False)
        SapModel = Nothing
        SapObject = Nothing

    End Sub


    ''' <summary>
    ''' Provides an Icon for every component that will be visible in the User Inter-
face.
    ''' Icons need to be 24x24 pixels.
    ''' </summary>
    Protected Overrides ReadOnly Property Icon() As System.Drawing.Bitmap
        Get
            'You can add image files to your project resources and access them like
this:
            ' return Resources.IconForThisComponent;
            Return Nothing
        End Get
    End Property

    ''' <summary>
    ''' Each component must have a unique Guid to identify it.
    ''' It is vital this Guid doesn't change otherwise old ghx files
    ''' that use the old ID will partially fail during loading.
    ''' </summary>
    Public Overrides ReadOnly Property ComponentGuid() As Guid
        Get
            Return New Guid("{4DFDA3B2-A319-4507-BE7B-FE50E8A469F7}")
        End Get
    End Property
End Class
```

»Ta stran je namenoma prazna«