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 program Gradbeništvo,
Konstrukcijska smer

Kandidat:
Miha Jaklič

Numerično iskanje optimalne oblike ploskovnih konstrukcij

Diplomska naloga št.: 2988

Mentor:
izr. prof. dr. Boštjan Brank

Somentor:
izr. prof. dr. Marko Kegl

Ljubljana, 19. 12. 2007

IZJAVA O AVTORSTVU

Podpisani **MIHA JAKLIČ** izjavljam, da sem avtor diplomske naloge z naslovom: »**Numerično iskanje optimalne oblike ploskovnih konstrukcij«.**

Izjavljam, da prenašam vse materialne avtorske pravice v zvezi z diplomsko nalogo na UL, Fakulteto za gradbeništvo in geodezijo.

Ljubljana, 4.12.2007

BIBLIOGRAFSKO – DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK: **624.073(043.2)**
Avtor: **Miha Jaklič**
Mentor: **izr. prof. dr. Boštjan Brank**
Somentor: **izr. prof. dr. Marko Kegl**
Naslov: **Numerično iskanje optimalne oblike ploskovnih konstrukcij**
Obseg in oprema: **69 str., 13 pregl., 68 sl., 26 en.**
Ključne besede: **ploskovne konstrukcije, lupine, optimizacija oblike**

Izvleček

Diplomska naloga obravnava določitev optimalne oblike ploskovnih konstrukcij. Podrobno je opisan postopek parametrizacije ploskovne konstrukcije s t.i. Bézierovim telesom, ki ga imenujemo tudi projektno telo. Spremembam parametrov projektnega telesa sledi tudi mreža končnih elementov. Optimizacija vrednosti parametrov poteka z iterativnim postopkom iskanja minimuma namenske funkcije. Uporabljen je raziskovalni računalniški program EKON - program za analizo elastičnih konstrukcij po metodi končnih elementov. Za samo optimizacijo je uporabljen raziskovalni računalniški program iGO, zasnovan na gradientni metodi. Prikazani so primeri numerične optimizacije oblike ploskovnih konstrukcij z uporabo namenske funkcije minimuma deformacijske energije in različnih omejitvenih pogojev (prostornina konstrukcije, pozicija točke konstrukcije, ...). Opisanih je več različnih računskih modelov iste konstrukcije z uporabo različnega števila projektnih elementov in projektnih spremenljivk. Poleg geometrijsko linearnih primerov je za primerjavo prikazana tudi numerična optimizacija geometrijsko nelinearnih primerov. Za kontrolo uklonske nosilnosti optimalne oblike ploskovne konstrukcije je prikazan postopek prenosa geometrije konstrukcije iz programa EKON v komercialni program SAP2000, s katerim je izvedena geometrijsko nelinearna analiza ter analiza uklonske nosilnosti.

BIBLIOGRAPHIC – DOCUMENTALISTIC INFORMATION

UDC: **624.073(043.2)**
Author: **Miha Jaklič**
Supervisor: **assoc. prof. dr. Boštjan Brank**
Co Supervisor: **assoc. prof. dr. Marko Kegl**
Title: **Numerical shape optimization of shell structures**
Notes: **69 p., 13 tab., 68 fig., 26 eq.**
Key words: **shell structures, thin shells, shape optimization**

Abstract

In this graduation thesis a numerical shape optimization of shell structures is considered. Parameterization of the shape of surface construction by a Bézier body as a design element is described in detail. The finite element mesh depends on the changes of parameters of the design element. The optimization consists of iterative minimization of the object function considered. The program EKON for the analysis of elastic constructions by FEM is used. The optimization step is done by the program iGO, which is based upon the gradient method. Special cases of numerical shape optimization of shell structures by minimization of strain energy and various constrains (construction volume, position of a point in a construction, etc.) are shown.

Various models of the same construction with different number of project elements and project variables are described. Besides geometric linear cases, also the numerical optimization of geometrical nonlinear cases are considered and compared with the linear ones. Program SAP2000 is used for the geometric nonlinear analysis and the analysis of the buckling resistance. An algorithm for the export of the geometry of the construction from EKON to SAP2000 is given.

ZAHVALA

Za pomoč pri nastajanju diplomske naloge se zahvaljujem mentorju izr. prof. dr. Boštjanu Branku in somentorju izr. prof. dr. Marku Keglu, ki sta si vedno vzela čas in mi svetovala in omogočila spoznanje s tako zanimivo temo.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

1 UVOD

Optimizacija oblike z metodami matematičnega programiranja je postala že kar obvezni del modernega projektiranja tako izdelkov kot nekaterih konstrukcij. Optimizacija oblike se uporablja v avtomobilski in letalski industriji, še posebej pomembna je pri masovni izdelavi izdelkov (zaradi manjše porabe materiala in s tem nižje cene izdelka).

V konvencionalni optimizaciji konstrukcij so zajete samo količine, neodvisne od same oblike konstrukcije, npr. velikosti prerezov, položaji armature v prerezu, debeline posameznih elementov,...

Pri projektiranju ploskovnih konstrukcij imamo opravka s koncentriranimi silami, z različnimi robnimi pogoji ter neskladnostmi med obliko in debelino konstrukcije. Zato postaja proces iskanja optimalne oblike vse pomembnejši del projektiranja ploskovnih konstrukcij. Ob upoštevanju vseh zgornjih dejstev lahko optimizacijo oblike ploskovnih konstrukcij opišemo kot način iskanja najprimernejše (optimalne) oblike pri:

- omejeni velikosti napetosti in pomikov;
- upoštevanju vseh robnih pogojev in vseh možnih obtežnih primerov;
- upoštevanju uklonskih problemov;
- upoštevanju realnih materialnih karakteristik.

V diplomski nalogi so predstavljeni različni pristopi k reševanju optimizacije oblike lupinastih konstrukcij s poudarkom na pristopu projektnega telesa. Namen naloge je bil: spoznati se z optimizacijo oblike lupinastih konstrukcij, spoznavanje z raziskovalnimi računalniškimi programi, ki omogočajo numerično optimizacijo oblike konstrukcij ter preveriti občutljivost numerične optimizacije oblike na izbiro projektnih teles in oblikovnih parametrov s pripravljanjem večih modelov iste konstrukcije ter različnimi omejitvenimi pogoji.

2 OPTIMIZACIJA OBLIKE PLOSKOVNIH KONSTRUKCIJ

Med ploskovnimi konstrukcijami so najbolj zanimive tanke lupinaste konstrukcije. Lupine so konstrukcijski elementi, ki prenašajo obtežbo na podpore pretežno z membranskim stanjem napetosti. Napetostno stanje, ki ne vsebuje upogibnih in prečnih notranjih statičnih količin, imenujemo membransko stanje napetosti. Lupine so konstrukcije, s katerimi ekonomično premostimo velike razpone in površine (športne dvorane, razstavni prostori, cerkve,...). V gradbeništvu se zelo pogosto pojavljajo v obliki kupol, silosov ter rezervoarjev. V osnovi so lupine konstrukcije, ki so zelo tanke in svojo veliko nosilnost dobijo na osnovi ukrivljenosti. Vzrok za njihovo izredno obnašanje je tako imenovani dvojni ločni efekt, ki v nasprotju z enodimenzionalnim lokom, omogoča prenašanje velikega števila obtežnih primerov skoraj v celoti z membranskimi silami. To pomeni, da so lupinaste konstrukcije v membranskem napetostnem stanju optimalne konstrukcije. Za dosego membranskega napetostnega stanja pa je potrebno določiti najugodnejšo obliko lupinaste konstrukcije.

2.1 Načini optimizacije

Pristope za optimizacijo oblike ploskovnih konstrukcij lahko razdelimo na dve skupini:

- starejši pristop, ki temelji predvsem na eksperimentiranju; npr. z uporabo enostavnega visečega modela (angl. *hanging model*);
- novejši pristop - *numerična optimizacija konstrukcij* (angl. s*tructural optimization*), pri kateri se uporabljajo moderne numerične in računalniške metode.

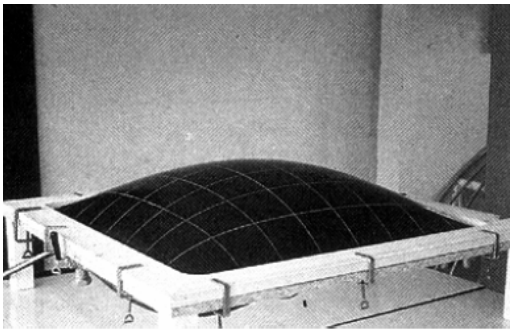
2.1.1 Eksperimenti

Pri iskanju optimalne oblike z eksperimenti je nujna izgradnja velikega števila modelov konstrukcije, kar pa je drago in zamudno. Za vsak model je potrebno predvideti materialne karakteristike, primerno obtežbo (v merilu glede na model) in geometrijske posebnosti konstrukcije. Zato se pri iskanju optimalne oblike uporablja karakterističen obtežni primer, ki je mišljen kot kombinacija najpogostejših obtežb konstrukcije. Realen model se obteži s karakteristično obtežbo (v primernem merilu), pod katero se model deformira. Obrnjena

deformirana oblika je končna (optimalna) oblika lupinaste konstrukcije. Za model je možno uporabiti deformabilno membrano, ki je največkrat kar kos tkanine.

Slika 1: Uporaba tkanine za model - eksperiment Heinza Islerja.

Pri uporabi tkanin se upošteva tudi orientacija vlaken, kar pa se lahko ugotovi le z iterativnim postopkom, da se zadosti mehanskim, funkcionalnim in estetskim pogojem v deformirani obliki. Obstajajo pa tudi druge metode iskanja fizičnega modela, zanimiva je na primer še metoda napihljive membrane (slika 2).



Slika 2: Napihnjena membrana – eksperiment Heinza Islerja.

Proces iskanja optimalne oblike s tem ni končan, saj se lahko nadaljuje z numerično simulacijo z geometrijsko nelinearno teorijo, kar pa že spada v numerično optimizacijo konstrukcij.

Kot zanimivost omenimo še arhitekta Heinza Islerja, ki je v 20. stoletju pri raziskovanju optimalne oblike naredil več kot 1000 različnih fizičnih modelov lupinastih konstrukcij iz betona.

2.1.2 Numerična optimizacija konstrukcij

Pri numerični optimizaciji se uporabljajo računalniški programi, ki omogočajo preverjanje odziva konstrukcije pri velikem številu različnih obtežnih primerov. Računalniški programi uporabljajo metodo končnih elementov.

Pristopov k numerični optimizaciji oblike lupinastih konstrukcij je veliko. Vsak avtor, ki se ukvarja z optimizacijo oblike konstrukcij, predlaga svoj pristop. Naj omenim le nekatere:

- Tehnika izbire robov lupinastih končnih elementov in njihovega premika z uporabo nelinearne optimizacije, kombinirano z uporabo komercialnih programov po metodi končnih elementov (Gates in Accorsi).
- Uporaba optimizacije materialne topologije lupine – poenostavitev oblike lupinaste konstrukcije z namenom dobiti gladko optimalno obliko (Maute in Ramm).
- Posebni pristop optimizacije lupin, ki imajo obliko dežnika (Imam).
- Uporaba optimizacije lupinastih konstrukcij z uporabo parametrizacije obstoječega CAD modela – z uporabo računalniškega modela po metodi končnih elementov (Daud, Camprubi in Bletzinger).
- Postopek optimizacije z uporabo topologije statično obteženih in vibrirajočih lupinastih konstrukcij z uporabo homogenizacijskih in evolucijskih metod (Belblidia in Bulman).
- Pristop k oblikovni in topološki optimizaciji lupinastih konstrukcij z minimizacijo podajnosti konstrukcij (Ansola).
- Pristop strategije diskretne optimizacije enostavnih lupin z ojačitvenimi elementi, z uporabo projektnih spremenljivk, ki opisujejo obliko konstrukcije in ojačitvenih elementov (Lagaros).
- Pristop *projektnega telesa,* s pripravljanjem modela konstrukcije s projektnimi telesi, z uporabo metode končnih elementov (Kegl in Brank).

V nadaljevanju sta opisana dva osnovna pristopa, ki ju lahko uporabimo tudi za bolj zahtevne primere lupinastih konstrukcij. To sta pristop optimizacije mreže končnih elementov in pristop projektnega telesa. V obeh pristopih je numerična optimizacija oblike opisana z minimiziranjem ene ali večih namenskih funkcij z upoštevanjem omejitvenih pogojev, kot osnova pa je računalniški program po metodi končnih elementov. Namenske funkcije

izračunamo z integracijo količin (napetosti, deformacije, prostornine,...) po končnih elementih.

2.2 Namenska funkcija

Za iskanje optimalne oblike ploskovnih konstrukcij je zanimivih več namenskih funkcij: deformacijska energija, enakomerno napetostno stanje, prostornina, teža, uklonska nosilnost.
Za dosego redukcije upogibnih momentov je uporabna namenska funkcija deformacijske energije, ki se izračuna po enačbi

$$f_E = \frac{1}{2}\int_V \sigma \cdot \varepsilon \, dV \,, \tag{2.1}$$

kjer je σ - napetost in ε - deformacija.
Za dosego čimbolj enakomernega napetostnega stanja, s konceptom omejevanja napetosti z vnaprej določeno vrednostjo napetosti σ_A, je enačba te namenske funkcije

$$f_L = \int_V (\sigma - \sigma_A)\, dV \,. \tag{2.2}$$

Za betonske konstrukcije lahko to namensko funkcijo uporabimo za preprečitev nateznih napetosti v konstrukciji.
Uporabimo pa lahko tudi namensko funkcijo prostornine z enačbo

$$f_V = \int_V dV \tag{2.3}$$

ali lastne teže, kjer pri integraciji upoštevamo še gostoto konstrukcije ρ,

$$f_W = \int_V \rho \, dV \,. \tag{2.4}$$

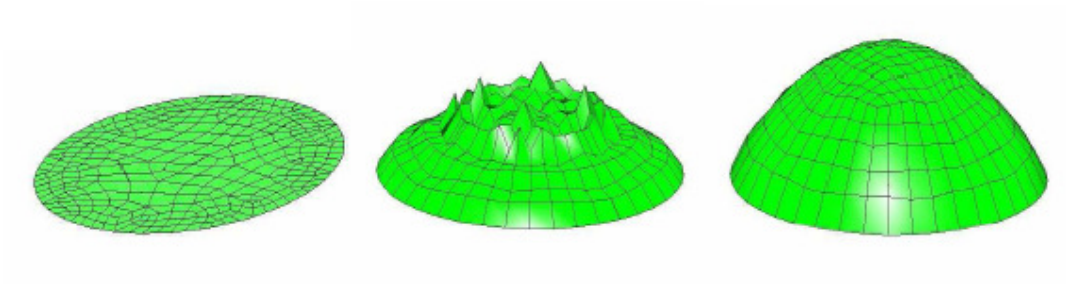
Lupinaste konstrukcije so zelo občutljive na geometrijske nepravilnosti, kar je potrebno upoštevati tudi v modelu, ko računamo maksimalno uklonsko nosilnost λ (ali minimum negativne vrednosti). Tudi to je lahko namenska funkcija z enačbo:

$$f_C = -\lambda \,. \tag{2.5}$$

2.3 Numerična optimizacija oblike konstrukcije z mrežo končnih elementov

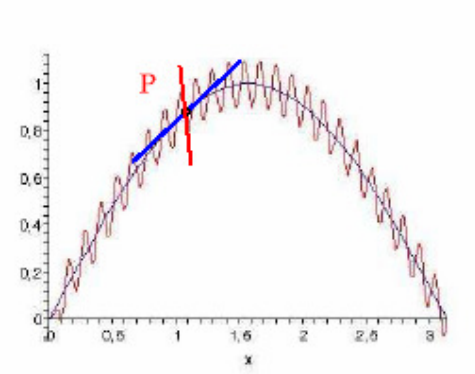
Za projektiranje konstrukcij so v uporabi različni komercialni računalniški programi po metodi končnih elementov, kjer se s postavitvijo mreže končnih elementov naredi

računalniški model konstrukcije. Zato je najbolj enostavno, če se to mrežo uporabi še za optimizacijo oblike konstrukcij. Za optimizacijo oblike je potrebno izbrati projektne spremenljivke, ki opisujejo obliko konstrukcije. V tem pristopu se za projektne spremenljivke uporabi kar koordinate točk mreže končnih elementov. Tak pristop omogoča veliko svobodo pri procesu optimizacije, saj je na izbiro veliko število projektnih spremenljivk (odvisno od števila končnih elementov). Tega principa se ne uporablja, pogosto predvsem zaradi velikega števila projektnih spremenljivk, negladke oblike optimalne konstrukcije in problemov, povezanih z diskretizacijo končnih elementov. Z uporabo zmogljivejših računalnikov in različnih filtrov končne oblike se te pomanjkljivosti počasi odpravljajo.



Slika 3: Primer optimizacije: osnovna oblika, optimalna oblika in filtrirana optimalna oblika.

Za optimizacijo se uporablja metode matematičnega programiranja z uporabo namenskih funkcij (npr. minimum deformacijske energije). V primeru uporabe gradientne metode je zaradi velikega števila projektnih spremenljivk, in zato različnih sprememb gradientov, gladko obliko praktično nemogoče doseči. Ta negladkost je odvisna predvsem od števila končnih elementov in mreženja modela konstrukcije, pa tudi zaradi več lokalnih ekstremov namenske funkcije. Zato je potrebno problem negladkosti oblike optimalne konstrukcije rešiti na drugačen način. Za izboljšanje optimalne oblike se uporabljajo različni filtri. Nazoren prikaz je v primeru prostoležeče okrogle lupine, obtežene z lastno težo (slika 3), kjer je bil uporabljen filter iz akustike – z omejevanjem še dovoljenih odstopanj (valov z visokimi frekvencami) od koordinat globalnega vala (slika 4).



Slika 4: Primer valov z visoko frekvenco (rdeča barva) na globalnem valu (modra barva).

2.4 Numerična optimizacija oblike konstrukcij s projektnim telesom

Na trgu je veliko zelo zmogljivih programov po metodi končnih elementov, v katerih pa optimizacija oblike konstrukcije (zaenkrat) še ni zajeta. Ponavadi se naredi računski model konstrukcije, ki pa ni nujno optimalne oblike. Predpostavi se spremenljive količine (prerezi, debeline, armatura, postavitev armature v prerezu,...) ter se optimira le-te količine, pri čemer ostane mreža končnih elementov nespremenjena skozi ves proces optimizacije – oblika konstrukcije se ne spreminja.

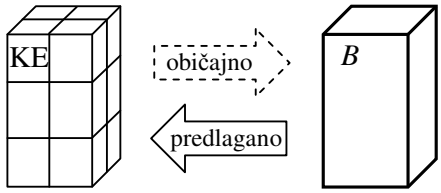
V optimizaciji oblike konstrukcije pa je potrebno pripraviti geometrijske podatke konstrukcije na drugačen način. Mrežo končnih elementov postavimo tako, da se prilega prej definiranemu projektnemu telesu. Projektno telo imenujemo takšno geometrijsko telo, ki je specializirano za oblikovanje meja (robov) telesa. S takšnim pristopom zajamemo lupinaste, skeletne in (polne) masivne konstrukcije.

2.4.1 Primerno projektno telo

V konvencionalnem načinu pripravljanja modela konstrukcije so podatki o obliki pripravljeni neposredno z ustrezno mrežo končnih elementov. Lahko rečemo, da mreža končnih elementov definira telo B (slika 5).

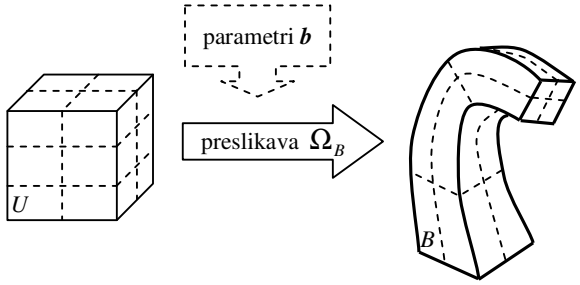
Za potrebe optimizacije oblike pa je potrebno definirati telo B kot neodvisni objekt in mrežo končnih elementov kot odvisno. Z definicijo telesa B kot primerno parametriziranega telesa, je lahko sprememba oblike mreže končnih elementov (in s tem konstrukcije) dosežena na eleganten in učinkovit način.

To se doseže z obravnavo telesa B kot preslikave $\Omega_B: U \times R^{N_B} \to B$, kjer je $U \subset \mathbb{R}^3$ ustrezno definicijsko območje v prostoru (npr. enotska kocka) in N_B skupno število projektnih spremenljivk, zbranih v vektorju $\boldsymbol{b} \in R^{N_B}$.



Slika 5: Prikaz odvisnosti med mrežo KE in projektnim telesom *B*.

Preslikavo Ω_B lahko predstavimo na sledeči način: Ω_B preslika točko iz U v točko v B pod vplivom projektnih spremenljivk. Povedano drugače: projektne spremenljivke so parametri, ki vplivajo na sliko B definicijskega območja U. Na tak način se lahko dobi mreža končnih elementov iz geometrijskih parametrov. Ko je preslikava Ω_B definirana, se lahko iz vrednosti projektnih spremenljivk izračuna poljuben geometrijski podatek mreže končnih elementov.



Slika 6: Definicija projektnega telesa *B* in mreže končnih elementov.

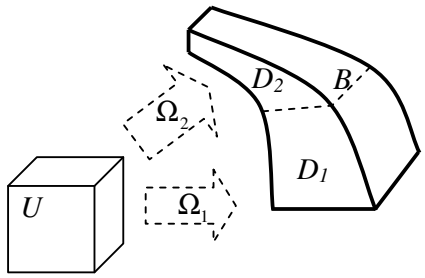
Prva stvar, na katero se je potrebno osredotočiti, je sprememba načina pripravljanja podatkov. Običajno se geometrijski podatki za mrežo končnih elementov preberejo iz vhodne datoteke. V tem primeru je potrebno pripraviti proceduro, ki bo izračunala podatke o poziciji končnih elementov iz telesa B pri znanih vrednostih projektnih spremenljivk.

Drug pomemben vidik se nanaša na parametrizacijo telesa B. Primerna izbira parametrizacije lahko prinese maksimalno fleksibilnost B z minimalnim nizom neodvisnih parametrov (projektnih spremenljivk).

Izkaže se, da je najbolje, če se razdeli B v N_D delčkov D_i imenovanih projektni elementi, tako da velja:

$$B = \bigcup_{i=1}^{N_D} D_i, \quad i = 1, 2, \ldots, N_D. \tag{2.6}$$

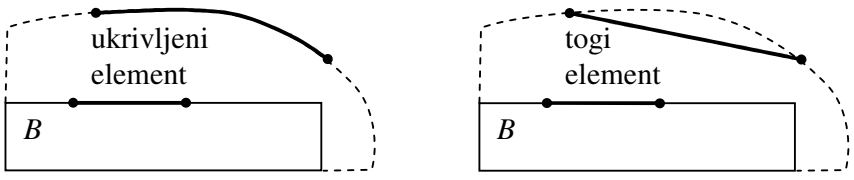
Ob predpostavki, da ima projektni element D_i enostavno obliko, se lahko telo B parametrizira z uporabo primerne preslikave $\Omega_i : U \times R^{N_B} \rightarrow D_i$. Na tak način je enostavno dobiti primerno parametrizacijo celotnega telesa B.



Slika 7: Projektno telo B, sestavljeno iz 2 projektnih elementov.

Tretji vidik, na katerega je potrebno še posebej paziti, pa je kvarjenje oblik mreže končnih elementov pri spreminjanju oblike telesa B. Pri tem pomaga procedura za popravljanje mreže med optimizacijo, ki ima to lastnost, da pri ponovnem pozicioniranju vozlišč obdrži fiksno topologijo. Uporaba tovrstne procedure je možna le ob predpostavki obstoja togih kontaktnih površin, po katerih se vozlišča lahko premikajo.

Z določitvijo leg vozlišč končnih elementov je tako v celoti določena oblika končnih elementov. Take elemente imenujemo *toge* (angl. *non convective*). Obstajajo pa tudi elementi, katerih geometrijo lahko podrobneje opišemo, njihova oblika pa se lahko popolnoma prilega obliki obravnavanega telesa. To so *ukrivljeni* (angl. *convective*) elementi.

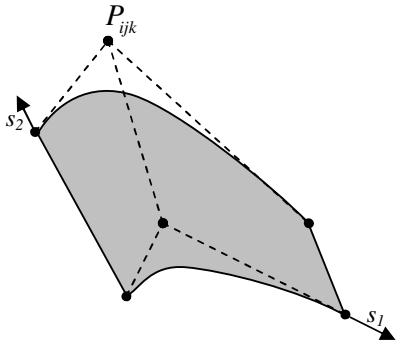


Slika 8: Razlika med ukrivljenim in togim elementom, ko se oblika spremeni iz osnovne (neprekinjena črta) v novo (prekinjena črta).

2.4.2 Bézierovo telo

V raziskovalnem programu EKON, ki sem ga uporabljal za modeliranje in optimizacijo ploskovnih konstrukcij, je za projektno telo izbrano racionalno Bézierovo telo, ki je izboljšano z dodatno spremenljivko – skalarnim poljem. Projektno telo je boljša izbira kot projektna ploskev, saj se lahko obravnava tako ploskovne konstrukcije (lupine) kot prostorske konstrukcije (npr. lupine ojačene s 3-D palično konstrukcijo, polne (ang. *solid*) konstrukcije,... Bézierovo telo je definirano s topološko kvadrasto shemo $N_1 \times N_2 \times N_3$ kontrolnih točk P_{ijk}. Vsaka kontrolna točka P_{ijk} racionalnega Bézierovega telesa ima naslednje podatke: lokacijo q_{ijk} in pripadajočo utež w_{ijk}. V programu EKON pa je bilo dodano še skalarno polje h_{ijk}, s katerim lahko vpeljemo specifične vrednosti končnih elementov – npr. debelino lupinastega končnega elementa.

Zaradi parametrizacije oblike projektnega telesa je potrebno definirati projektne spremenljivke. S predpostavko, da q_{ijk}, w_{ijk} in h_{ijk} niso konstante, temveč odvisne od projektnih spremenljivk $b_1, b_2, \ldots, b_N$, zbranih v vektorju $\boldsymbol{b} \in \boldsymbol{R}^N$; dobimo $q_{ijk} = q_{ijk}(\boldsymbol{b}), w_{ijk} = w_{ijk}(\boldsymbol{b})$ in $h_{ijk} = h_{ijk}(\boldsymbol{b})$.



Slika 9: Vsaka kontrolna točka P_{ijk} je podana s pozicijo q_{ijk}, utežjo w_{ijk} in skalarjem h_{ijk}.

Odvisnosti je možno zapisati na različne načine. V tem primeru je bila uporabljena Fortranova sintaksa: vsako komponento kontrolnih točk, uteži in skalarnega polja lahko zapišemo kot kombinacijo projektnih spremenljivk b_i v obliki: $\alpha_0 + \sum_i \alpha_i \cdot b_i^n$, kjer sta α_0 in α_i skalarja,

$n \in \mathbb{R}$. Tako je enostavno definirati projektne spremenljivke, ki opisujejo obliko konstrukcije ter izračunati potrebne odvode po projektnih spremenljivkah $dq_{ijk}/d\boldsymbol{b}$, $dw_{ijk}/d\boldsymbol{b}$ in $dh_{ijk}/d\boldsymbol{b}$. Bézierovo telo ima krivočrtni lokalni koordinatni sistem s koordinatami s_1, s_2 in s_3, ki tečejo od 0 do 1. Tako krajevni vektor $\boldsymbol{s}=[s_1,s_2,s_3]^T$ predstavlja kar točko na projektnem telesu - *lokalno točko*, definirano relativno na koordinatni sistem projektnega telesa.

Za katerokoli točko $\boldsymbol{s}$ lahko izrazimo pozicijo $\boldsymbol{r}$ v globalnem koordinatnem sistemu (kartezičnem 3-D prostoru) z enačbo:

$$\boldsymbol{r}=\boldsymbol{r}(\boldsymbol{s},\boldsymbol{b})=\frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{k=1}^{N_3} B_i^{N_1}\cdot B_j^{N_2}\cdot B_k^{N_3}\cdot w_{ijk}\cdot q_{ijk}}{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{k=1}^{N_3} B_i^{N_1}\cdot B_j^{N_2}\cdot B_k^{N_3}\cdot w_{ijk}}, \tag{2.7}$$

prav tako pa tudi skalarno količino *t* kot:

$$t=t(\boldsymbol{s},\boldsymbol{b})=\frac{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{k=1}^{N_3} B_i^{N_1}\cdot B_j^{N_2}\cdot B_k^{N_3}\cdot w_{ijk}\cdot h_{ijk}}{\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\sum_{k=1}^{N_3} B_i^{N_1}\cdot B_j^{N_2}\cdot B_k^{N_3}\cdot w_{ijk}}, \tag{2.8}$$

kjer so $B_i^{N_1}=B_i^{N_1}(s_1)$, $B_j^{N_2}=B_j^{N_2}(s_2)$ in $B_k^{N_3}=B_k^{N_3}(s_3)$ Bernsteinovi bazni polinomi stopnje N_1-1, N_2-1 in N_3-1.

2.4.3 Lupinasti končni element

Za modeliranje dela lupinaste konstrukcije je uporabljen štiritočkovni bilinearni izoparametrični lupinasti končni element.

Interpolacijo srednje referenčne ploskve $\boldsymbol{r}$, referenčno smerno polje $\boldsymbol{n}$ in polje premikov $\mathbf{u}=(\mathbf{v},\Delta\mathbf{n})$ se definira z naslednjimi enačbami (glede na izoparametrični koncept):

$$\mathbf{r}(\xi,\eta)=\sum_{n=1}^{4}N_n(\xi,\eta)\mathbf{r}_n,\ \mathbf{n}(\xi,\eta)=\sum_{n=1}^{4}N_n(\xi,\eta)\mathbf{n}_n,\ t(\xi,\eta)=\sum_{n=1}^{4}N_n(\xi,\eta)t_n, \tag{2.9}$$

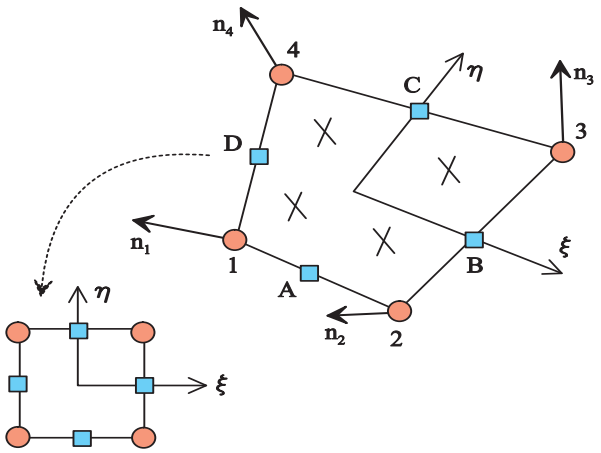
in

$$\mathbf{v}(\xi,\eta)=\sum_{n=1}^{4} N_n(\xi,\eta)\mathbf{v}_n, \qquad \Delta\mathbf{n}(\xi,\eta)=\sum_{n=1}^{4} N_n(\xi,\eta)\Delta\mathbf{n}_n. \tag{2.10}$$

$N_n(\xi,\eta)$ je standardna bilinearna izoparametrična oblikovna funkcija

$$N_n(\xi,\eta)=\frac{1}{4}(1+\xi\xi_n)(1+\eta\eta_n), \qquad n=1,2,3,4, \tag{2.11}$$

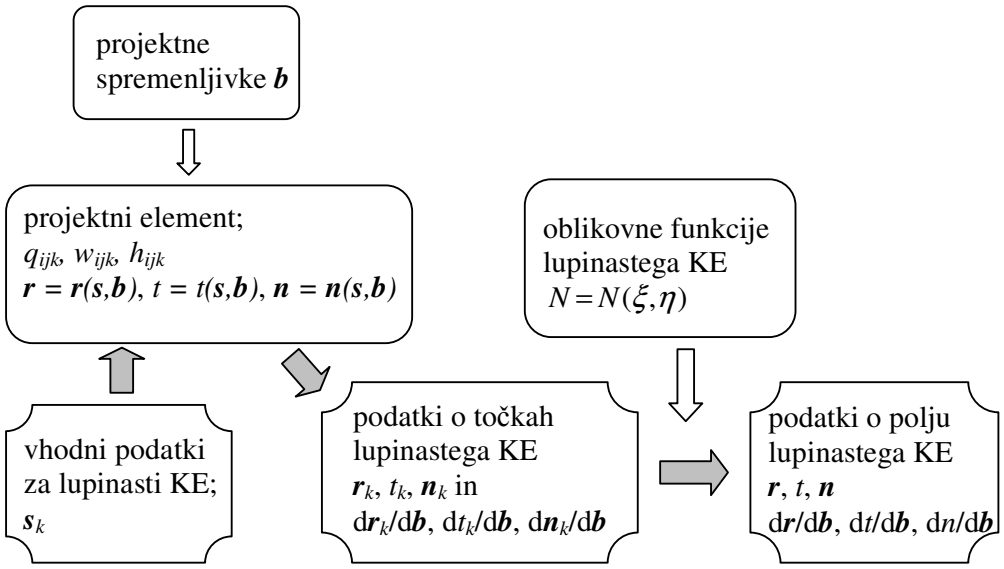
kjer je $(\xi_n,\eta_n)\in\{(-1,-1),(1,-1),(1,1),(-1,1)\}$ in so $(\mathrm{x})_n$ vrednosti posameznih količin v vozliščih končnega elementa.



Slika 10: Štiritočkovni bilinearni izoparametrični lupinasti končni element.

Če so znani vsi podatki projektnega elementa q_{ijk}, w_{ijk} in h_{ijk}, lahko z enačbama (2.7) in (2.8) izračunamo vse geometrijske podatke za končni element samo z njegovo pozicijo v lokalnem koordinatnem sistemu.

Geometrijski podatki lupinastega končnega elementa so popolnoma določeni s štirimi lokalnimi točkami s_k, k = 1,2,3,4. Pozicijo ***r***, debelino t lupinastega končnega elementa in vrednost smernega polja ***n*** katerekoli vozliščne točke lupinastega končnega elementa lahko izračunamo na način, prikazan na diagramu (slika 11).



Slika 11: Proces izračuna geometrijskih podatkov lupinastega končnega elementa.

Projektno telo uporabimo le za pridobitev geometrijskih podatkov točk $r_k = \mathbf{r}(s_k,\mathbf{b})$, $t_k = t(s_k,\mathbf{b})$, $n_k = n(s_k,\mathbf{b})$ in ustreznih odvodov

$$\frac{dr_k}{d\mathbf{b}} = \left.\frac{dr}{d\mathbf{b}}\right|_{s=s_k}, \quad \frac{dt_k}{d\mathbf{b}} = \left.\frac{dt}{d\mathbf{b}}\right|_{s=s_k}, \quad \frac{dn_k}{d\mathbf{b}} = \left.\frac{dn}{d\mathbf{b}}\right|_{s=s_k} \tag{2.12}$$

po projektnih spremenljivkah za vse lupinaste končne elemente.

Ko so podatki o mreži končnih elementov znani, geometrijske podatke, potrebne za integracijo po končnem elementu, izračunamo z oblikovno funkcijo $N_k = N_k(\xi,\eta)$, ki je definirana v uporabljenem končnem elementu.

Tako velja

$$r = \sum_{k=1}^{4} N_k \cdot r_k, \quad t = \sum_{k=1}^{4} N_k \cdot t_k, \quad n = \sum_{k=1}^{4} N_k \cdot n_k \tag{2.13}$$

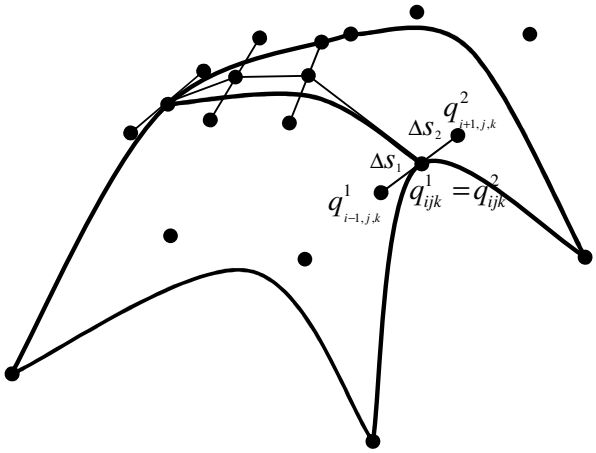
in

$$\frac{dr}{db}=\sum_{k=1}^{4} N_k \cdot \frac{dr_k}{db},\ \ \frac{dt}{db}=\sum_{k=1}^{4} N_k \cdot \frac{dt_k}{db},\ \ \frac{dn}{db}=\sum_{k=1}^{4} N_k \cdot \frac{dn_k}{db}, \tag{2.14}$$

kjer je vsaka definirana količina odvisna od ξ, η, s_i, $i=1,2,3,4$ in $\boldsymbol{b}$. S takšno proceduro je oblika konstrukcije (in mreža končnih elementov) parametrizirana le s projektnimi spremenljivkami $\boldsymbol{b}$.

2.4.4 Gladko sestavljanje Bézierovih teles

Pri sestavljanju dveh ali večih projektnih teles se pojavi na stiku obeh teles problem negladkosti. Zvezno odvedljivost je možno zagotoviti že s primerno postavitvijo kontrolnih točk q_{ijk} dveh teles. Za zadostitev pogoja zveznosti je dovolj že, če se kontrolne točke na skupnih robovih obeh teles ujemajo ($q^1_{ijk}=q^2_{ijk}$). V primeru zvezne odvedljivosti je potrebno zagotoviti pozicije kontrolnih točk na robu obeh teles tako, da so vse trojice točk ($q^1_{i-1,j,k}$, $q^{1,2}_{i,j,k}$, $q^2_{i+1,j,k}$) kolinearne in velja razmerje $\Delta s_1 : \Delta s_2$ med oddaljenostjo sosednjih kontrolnih točk za vse trojice točk, kjer sta $\Delta s_1=\left\|q^1_{i-1}-q^1_i\right\|$ in $\Delta s_2=\left\|q^2_{i+1}-q^2_i\right\|$ razdalji med dvema robnima kontrolnima točkama.



Kontrolne točke $q^1_{i-1,j,k}$, $q^{1ali2}_{i,j,k}$, $q^2_{i+1,j,k}$ ležijo na premici ter velja razmerje razdalj $\Delta s_1 : \Delta s_2 = konst$ za vse i, *j* in *k*.

Slika 12: Zvezno odvedljivo sestavljeno Bézierovo telo (vsako telo ima 4x3x1 kontrolnih točk).

S primerno izbiro kontrolnih točk na robu obeh teles in oddaljenostjo med njimi je tudi sestavljeno telo gladko.

2.5 Formulacija in rešitev optimizacijskega problema

Optimizacijski problem statično obtežene konstrukcije je ponavadi zapisan kot

$$\min(f_0) \tag{2.15}$$

z omejitvenimi pogoji

$$\begin{aligned} &f_i \le 0,\ i = 1,2,\ldots,M \\ &b_i^L \le b_i \le b_i^U,\ i = 1,2,\ldots,N \end{aligned} \tag{2.16}$$

Velikokrat je namenska funkcija $f_0 = f_0(\boldsymbol{b},\boldsymbol{u})$ definirana kot prostornina ali deformacijska energija konstrukcije. Omejene količine $f_i = f_i(\boldsymbol{b},\boldsymbol{u})$ so ponavadi točkovni premiki in rotacije, napetosti, geometrijske, tehnološke omejitve,... Oznaki b_i^L in b_i^U pomenita spodnjo in zgornjo možno vrednost projektne spremenljivke, $\boldsymbol{u}$ pa vektor odzivnih spremenljivk – ponavadi točkovni pomiki v primeru statično obtežene konstrukcije.

Omeniti je potrebno še, da so projektne spremenljivke $\boldsymbol{b}$ neodvisne spremenljivke, odzivne spremenljivke pa odvisne, $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{b})$. Ta odvisnost je uvedena kot naslednja enakost – (vsota notranjih sil je enaka vsoti zunanjih sil)

$$\boldsymbol{F} - \boldsymbol{R} = \boldsymbol{0}, \tag{2.17}$$

kjer sta $\boldsymbol{F} = \boldsymbol{F}(\boldsymbol{b},\boldsymbol{u})$ in $\boldsymbol{R} = \boldsymbol{R}(\boldsymbol{b},\boldsymbol{u})$ vektorja notranjih in zunanjih sil. Notranje sile so očitno odvisne neposredno od $\boldsymbol{b}$ in $\boldsymbol{u}$. Zunanje sile so velikokrat kar konstantne. Če je upoštevan izračun lastne teže, so zunanje sile odvisne tudi od $\boldsymbol{b}$, v primeru elastičnih podpor pa tudi od $\boldsymbol{u}$.

2.6 Optimizacija po gradientni metodi

Ker so projektne spremenljivke zvezne in namenske funkcije odvedljive po projektnih spremenljivkah, je optimizacijski problem možno reševati z uporabo gradientne metode. V tem primeru se rešitev išče po naslednji proceduri:

1. postavi števec $k = 0$, izberi začetne vrednosti $\boldsymbol{b}^{(0)}$,
2. izračunaj f_i, $i = 0,1,...,M$ pri vrednostih $\boldsymbol{b}^{(k)}$ (analiza odziva),
3. izračunaj $\mathrm{d}f_i/\mathrm{d}\boldsymbol{b}$, $i = 1,2,...,M$ pri vrednostih $\boldsymbol{b}^{(k)}$ (občutljivostna analiza),

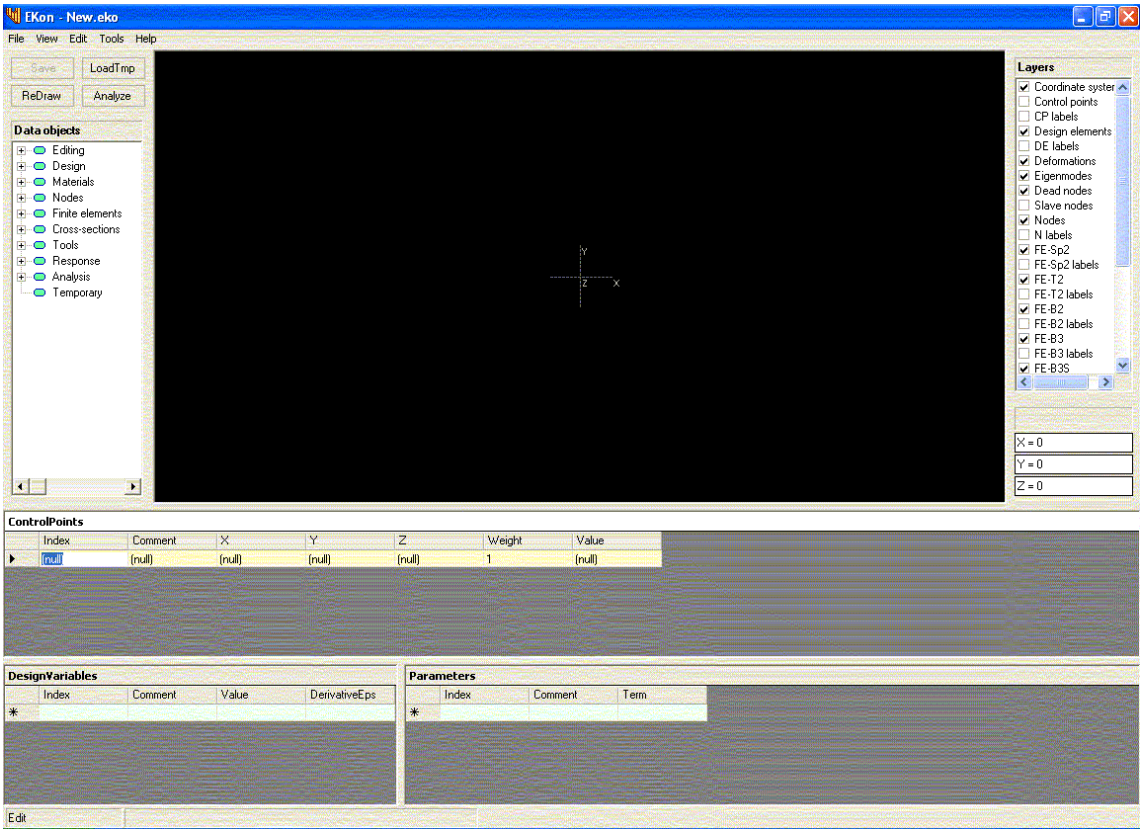
4. izračunane vrednosti podaj optimizatorju, da izračuna popravke $\Delta \boldsymbol{b}^{(k)}$ in izračuna izboljšane vrednosti projektnih spremenljivk $\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)} + \Delta \boldsymbol{b}^{(k)}$,
5. postavi $k = k + 1$ in preveri konvergenčne kriterije – če so v redu, končaj, sicer pojdi na korak 2.

3 PROGRAMA EKON IN IGO

Pri optimizaciji oblike lupinastih konstrukcij sem uporabljal raziskovalna računalniška programa EKON in iGO. Programa razvijajo na Fakulteti za strojništvo Univerze v Mariboru in na Fakulteti za gradbeništvo in geodezijo v Ljubljani. Programa sta sicer povsem neodvisna, vendar je kombinirana uporaba enostavna.

3.1 Program EKON

Program EKON je program za analizo elastičnih konstrukcij po metodi končnih elementov. Vgrajene ima naslednje končne elemente: končni element vzmeti, končni element 3D palice, dvovozliščni 3D nosilec s konstantnim prerezom, trivozliščni 3D nosilec s konstantnim prerezom, končni element za ravninsko napetostno stanje, končni element za 3D telo in končni element za lupine.



Slika 13: Grafični vmesnik programa EKON.

Program za mreženje uporablja projektna telesa, vgrajena so Bézierova telesa. Na sliki (slika 13) je prikaz grafičnega vmesnika, ki je pregleden in zato enostaven za uporabo. V sredini je 3D koordinatni sistem, na katerem se izriše modelirana konstrukcija (projektna telesa, vozlišča, končni elementi, po analizi tudi deformacije, ...) Na desni strani je okno »Layers«, kjer lahko odkrivamo ali skrivamo plasti podatkov, kot so: koordinatni sistem, kontrolne točke, vozlišča, končni elementi, deformacije in označbe posameznih podatkov. Na levi strani je okno »Data Objects«, kjer najdemo vse kategorije podatkov – o materialu, projektnem telesu, obtežbi, uporabljenih končnih elementih, vozliščih, analizi, občutljivostni analizi, drobljenju mreže končnih elementov, ... Ko izberemo kategorijo podatkov, se v trenutno aktivnem oknu (izmed treh) odpre tabela za podajanje podatkov te kategorije. V kolikor pa se šele spoznavamo s programom, si lahko pomagamo tudi s pomočjo (»Help«), kjer sem v sklopu diplomske naloge tudi opisal kategorije podatkov in njihov pomen.

3.1.1 Priprava podatkov

Vrstni red podajanja podatkov je v programu EKON nekoliko specifičen. Da lahko s pomočjo parametrov spreminjamo obliko celotne konstrukcije, je potrebno najprej definirati projektne spremenljivke in parametre. Projektne spremenljivke so količine, ki se spreminjajo v procesu optimizacije s programom iGO (normirana debelina, normirane pozicije kontrolnih točk v x, y ali z smeri, ...). V programu EKON kot spremenljivke uporabljamo parametre, ki so linearno odvisni od projektnih spremenljivk. Definirati moramo projektno telo (Bézierovo telo), kar naredimo s podajanjem kontrolnih točk (»Design/Control Points«), ki so lahko podane kot vrednosti v globalnem koordinatnem sistemu ali pa kot sklic na parameter (ki je lahko odvisen od projektnih spremenljivk). Za pregled podanih kontrolnih točk, ki se sicer izrisujejo v koordinatni sistem (s klikom na gumb »ReDraw«) je potrebno prej definirati analizo konstrukcije (»Analysis/Settings«). Ko imamo podane kontrolne točke, sledi definiranje projektnega telesa (»Design/QuadrangularBézierBody« ali »Design/TrianguralBézierBody«), kjer podamo vrstni red že definiranih kontrolnih točk. Glede na to, da lahko projektna telesa poljubno sestavljamo, je nujna še definicija projektnih elementov (»Design/DesignElements«). Sledi podajanje materialnih karakteristik, nato še vozlišč in končnih elementov glede na lokalni koordinatni sistem projektnega elementa (koordinate tečejo od 0 do 1 v vseh treh smereh). Lahko si pomagamo z vgrajenim orodjem za

avtomatično podajanje vozlišč in/ali avtomatičnega mreženja projektnega telesa z izbranimi končnimi elementi. Nato podamo še obtežbo (vozliščno, ploskovno, lastno težo). Če so pravilno podani vsi podatki, je možna linearna analiza in po kliku na gumb »Analyze« se izriše deformirana oblika podanega modela konstrukcije.

Pri optimizaciji oblike konstrukcije želimo narediti tudi občutljivostno analizo, zato moramo definirati namensko funkcijo in omejitvene pogoje (»ResponseQuantities«). Namensko funkcijo in omejitveni pogoj napišemo s pomočjo šifre (uporabljamo lahko samo tiste, ki so vgrajene v program) in formule, po kateri se izračuna normirana vrednost:

$$R = \frac{G}{N_v} - A_v , \tag{3.1}$$

kjer se G izračuna:

- če je izbrana opcija »Square«: $G = \left(F - C_v\right)^2$;
- če je izbrana opcija »Absolute« $G = \left|F - C_v\right|$.

Glede na možnost izbire operacij (absolutne vrednosti in kvadrata) in izbiro treh konstant (C_v, N_v in A_v), je možno podati tudi še tako kompliciranе omejitvene pogoje.

Pomeni konstant za izračun normirane namenske funkcije, oziroma normiranega omejitvenega pogoja, so v naslednji preglednici (preglednica 1).

Preglednica 1: Pomeni konstant za izračun normirane namenske funkcije.

konstanta	Pomen
R	normirana vrednost namenske funkcije ali omejitvenega pogoja
F	vrednost namenske funkcije (iz podatkov izračuna EKON)
C_v	primerjalna vrednost (compare value)
N_v	normalizacijska vrednost (normalized value)
A_v	dopustna vrednost (allowable value)

3.1.2 Dodatki in spremembe programa EKON

3.1.2.1 Pomoč

V času spoznavanja s programom EKON je bila pomoč (v programu) zelo skopa. V okviru svojega dela sem se zato lotil pisanja potrebnih datotek za dograditev programske pomoči, kar je bilo kasneje tudi vključeno v program EKON (glej PRILOGE).

3.1.2.2 Izboljšava projektnega telesa

V programu EKON je bilo možno podajanje Bézierovega projektnega telesa s 5 kontrolnimi točkami v posamezno smer (največ 5x5x5 – 125 kontrolnih točk na telo). Zaradi natančnejšega podajanja (vsaj možnosti) oblik lupinastih konstrukcij sem dodal kodo v program EKON, tako da sem lahko podajal Bézierovo telo z 10 kontrolnimi točkami v posamezni smeri (največ 10x10x10 – 1000 kontrolnih točk na telo).
Izračunati je bilo potrebno:

1. Bernsteinove bazne polinome za n = 6,7,8,9 in 10;
2. prve odvode Bernsteinovih baznih polinomov za n = 6,7,8,9 in 10;
3. druge odvode Bernsteinovih baznih polinomov za n = 6,7,8,9 in 10.

Bernsteinove bazne polinome, prve in druge odvode sem izračunal s programom Mathematica. Bernsteinovi bazni polinomi so definirani z:

$$B_i^n(x) = \binom{n}{i} \cdot x^i \cdot (1-x)^{n-i}, \tag{3.2}$$

kjer je n stopnja polinoma (število kontrolnih točk v smeri-1), za vse $i \in \{0,1,\ldots,n-1,n\}$.
Njihovi prvi odvodi:

$$\frac{dB_i^n(x)}{dx} = \binom{n}{i} \cdot x^{i-1} \cdot (1-x)^{n-i-1} \cdot (i - n \cdot x), \tag{3.3}$$

drugi odvodi pa:

$$\frac{d^2B_i^n(x)}{dx^2} = \binom{n}{i} \cdot x^{i-2} \cdot (1-x)^{n-i-2} \cdot \left(i^2 + (2 \cdot x \cdot (1-n) - 1) \cdot i + (n-1) \cdot n \cdot x^2\right). \tag{3.4}$$

3.2 Program iGO

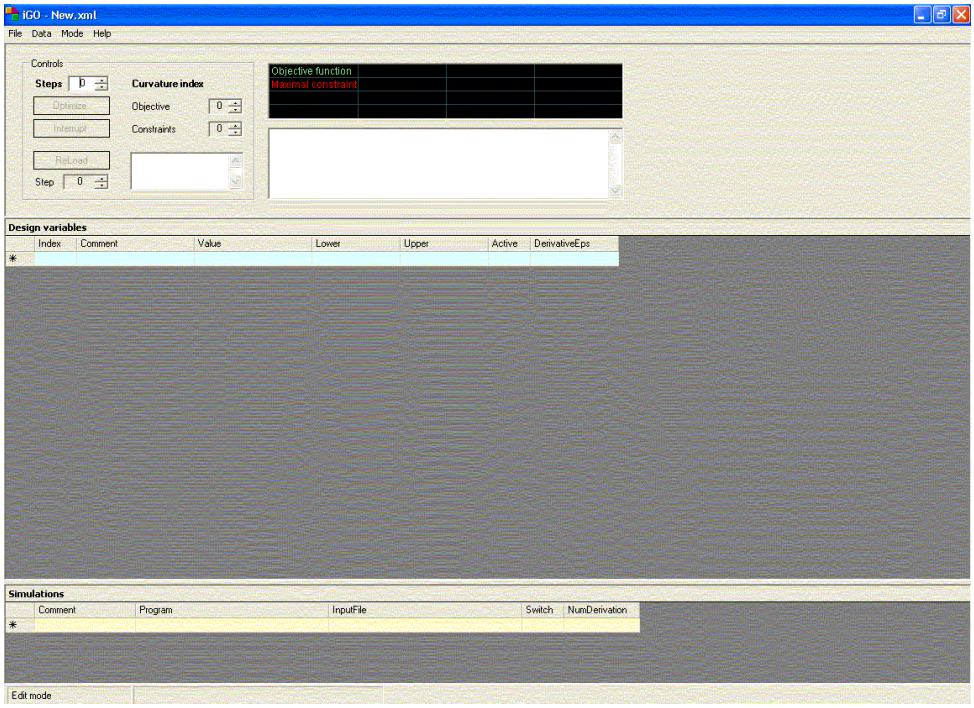
Program iGO omogoča interaktivno gradientno optimizacijo z uporabo zunanjih simulatorjev (npr. EKON).

Program deluje v dveh načinih:

- Urejevalnik (»Edit mode«): način namenjen podajanju in spreminjanju podatkov;
- Optimizacijski način (»Optimization mode«): način pri izvajanju optimizacije.

V tabelo »Design variables« moramo vpisati podatke o projektnih spremenljivkah, ki pa jih lahko enostavno vnesemo z ukazom »Load Design Variables« iz vhodne datoteke za program EKON. Definiramo še območje »Lower« in »Upper«, s čimer omejimo možne vrednosti projektnih spremenljivk in s tem podamo interval, v katerem program iGO poišče optimalno rešitev. Sledi še podajanje simulacij (»Simulations«).

Podati moramo pot do zunanjega simulatorja (programa EKON) in pot do vhodne (EKON-ove) datoteke. Da se ne odpira grafični vmesnik, nastavimo stikalo »/s« (silent mode), in izberemo ali naj se izračunajo tudi odvodi namenske funkcije in omejitvenih funkcij po projektnih spremenljivkah. Ker program EKON to že izračuna, ponovno računanje ni potrebno. Ko se shrani iGO-va vhodna datoteka, lahko začnemo z optimizacijo in iskanjem najoptimalnejše rešitve problema. Za začetek optimizacije je potreben prehod v optimizacijski način, kar naredimo v meniju »Mode«. Ker je metoda interaktivna, lahko nastavimo število iterativnih korakov in omejimo velikost sprememb projektnih spremenljivk s spremembo indeksa »Curvature Index«, ločeno za namensko funkcijo in omejitvene pogoje. Za vsak korak se izračunajo vse namenske funkcije in omejitveni pogoji, njihovi odvodi po vseh projektnih spremenljivkah (EKON) in se zažene optimizator, ki izboljša vrednosti projektnih spremenljivk. Po vsakem koraku se izriše tudi graf namenske funkcije in omejitvenih pogojev, na katerem je možno spremljanje poteka optimizacije.



Slika 14: Grafični vmesnik programa iGO.

Ko se vrednost reducirane namenske funkcije v nekaj zaporednih iteracijah ne spremeni bistveno, je model konstrukcije optimalen (pri prej postavljenih mejah projektnih spremenljivk). Seveda pa si lahko po vsaki posamezni iteraciji pogledamo vmesni rezultat v programu EKON, kjer s pritiskom na gumb »LoadTmp« na zgornji levi strani grafičnega vmesnika programa EKON vnesemo trenutne vrednosti projektnih spremenljivk.

3.3 Prenos geometrije konstrukcije iz EKON-a v SAP2000

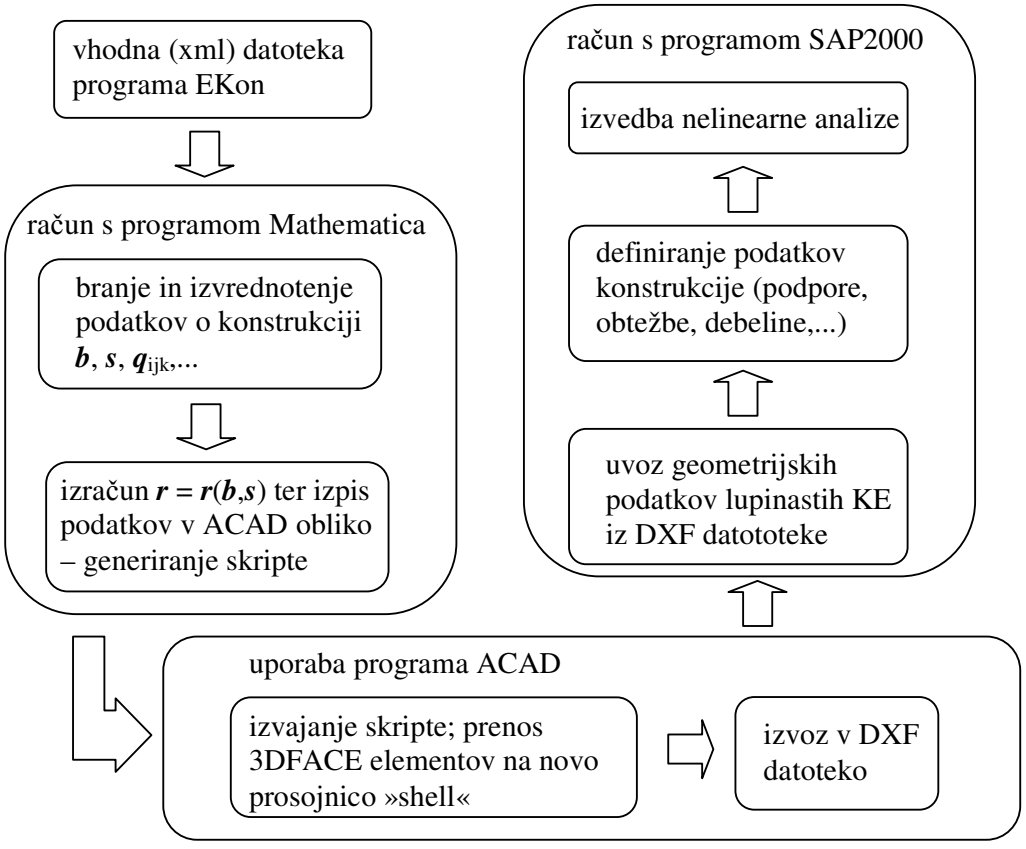
Zaradi kontrole analize odziva EKON-a ter računanja uklonske nosilnosti, sem geometrijo optimalnega modela konstrukcije prenesel v komercialni program SAP2000.

SAP2000 je zelo zmogljiv program po metodi končnih elementov, ki omogoča tako linearno kot nelinearno analizo (geometrijska nelinearnost, uklon, potres,...) konstrukcij. Program ima vključeno tudi dimenzioniranje betonskih in jeklenih elementov.

Zaradi specifičnega podajanja oblike konstrukcije s projektnimi telesi v programu EKON, sem moral koordinate vseh vozlišč končnih elementov iz lokalnega koordinatnega sistema (u,v) preračunati v globalnega (x,y,z), definirati vse lupinaste končne elemente, jih prenesti v program ACAD, vse končne elemente narisati z ukazom 3DFACE in jih postaviti na svojo

prosojnico, narisano nato posneti v DXF datoteko in geometrijo lupinaste konstrukcije tako uvoziti v program SAP2000. S postavitvijo končnih elementov sem imel definirano obliko konstrukcije, sledilo pa je še definiranje podpor, obtežbe, debeline lupin in materialnih karakteristik.

Za branje podatkov in zapis v ACAD skripto sem uporabil program Mathematica, kjer sem sprogramiral proceduro, ki prebere vse potrebne podatke iz vhodne datoteke programa EKON (xml), izračuna vse točke končnih elementov in zapiše podatke o lupinastih končnih elementih v skripto, ki se lahko požene znotraj programa ACAD. Ker za uvoz končnih elementov v program SAP2000 potrebujemo DXF datoteko, sem napisal še funkcijo, ki znotraj Mathematice kliče program ACAD in izvede prej narejeno skripto ter posname dxf datoteko. Tako sem lahko geometrijo modela vsake konstrukcije iz EKON-a enostavno prenesel v komercialni program SAP2000.



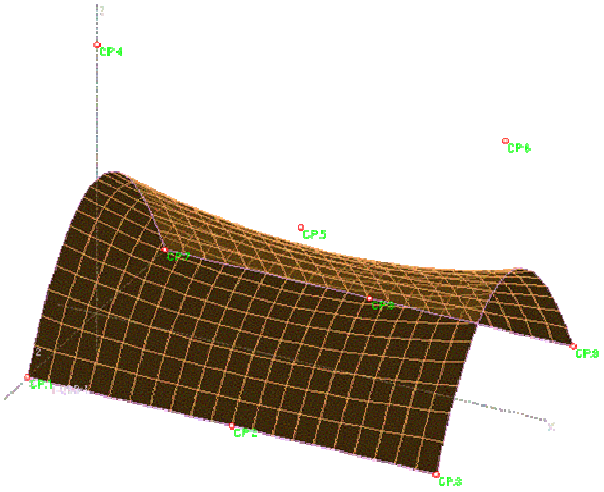
Slika 15: Diagram prenosa geometrijskih podatkov iz programa EKON v program SAP2000.

4 LINEARNI PRIMERI

4.1 Sedlo

4.1.1 Opis primera

Obravnaval sem optimizacijo konstrukcije oblike sedla, geometrija le-te je prikazana na spodnji sliki (Slika 16). Višina na robu (x = 0 m , y = 0 m in x = 30 m, y = 0 m) meri 10 m, v sredini (y = 0 m, x = 15 m) pa 7.5 m, dolžina sedla pa je 30 m. Začetna debelina lupine je h = 15 cm. Materialne karakteristike uporabljenega betona so naslednje: elastični modul $E = 3 \cdot 10^7$ kN/m^2, Poissonov količnik ν = 0.2. Sedlo je nepomično podprto na spodnjih robovih, ter obteženo z lastno težo γ = 25 kN/m^2. Sedlo sem modeliral s 400 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom z 9 kontrolnimi točkami (3x3x1). Za projektne spremenljivke sem izbral z-koordinato kontrolne točke (4 in 6) na robu, ki se lahko spreminja v intervalu [-40 m, 60 m], z-koordinato kontrolne točke (5) v sredini sedla [-17 m, 32 m], ter debelino lupine h [5 cm, 25 cm].



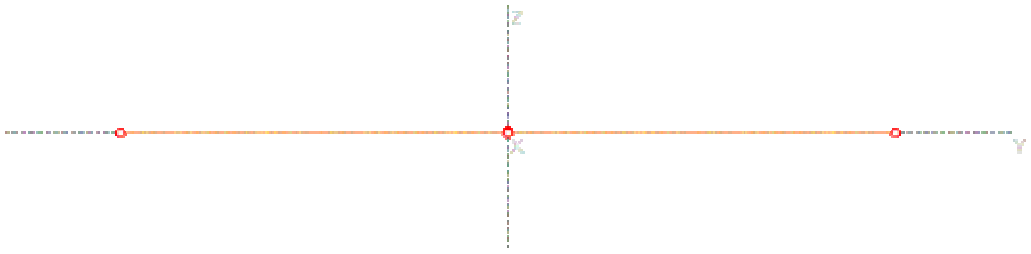
Slika 16: Začetni model konstrukcije – sedla.

4.1.2 Rezultati

Poiskati je potrebno optimalno obliko konstrukcije pri:

(a) minimalni prostornini konstrukcije.

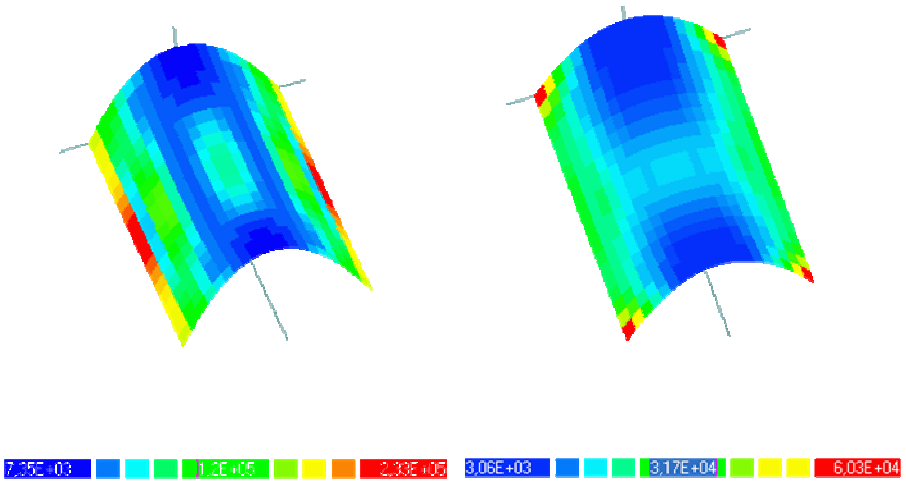
Prostornina začetne oblike konstrukcije je 122 m^3, po optimizaciji pa 30 m^3. Glede na to, da se spreminjajo le višine in debelina lupine, je bilo pričakovati, da se bo debelina lupine zmanjšala na 5 cm, višini pa se bosta zmanjšali. Rezultat optimizacije je viden na spodnji sliki (slika 17), kjer je razvidno, da se je oblika sedla spremenila v ploščo. Glede na dobljeno obliko pridemo do ugotovitve, da bi bilo potrebno omejiti minimalne višine (na robu in v sredini) zaradi uporabnosti takšne lupinaste konstrukcije.



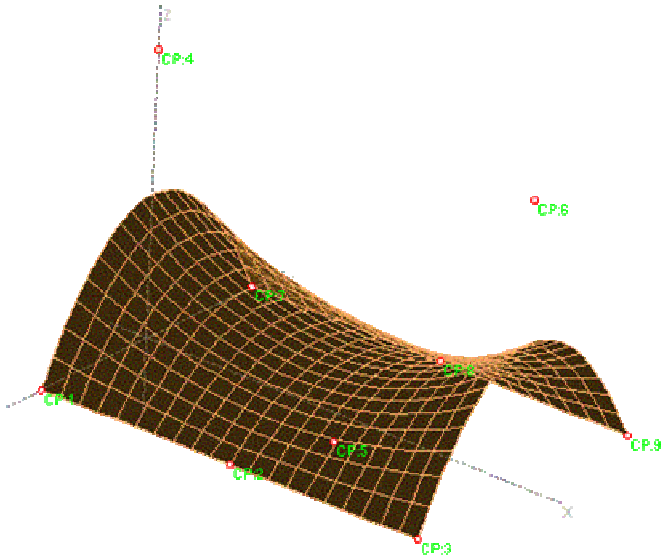
Slika 17: Optimalna oblika konstrukcije sedla – primer a (min.V).

(b) minimalni deformacijski energiji.

Na spodnji levi sliki je prikazana deformacijska energija začetne oblike sedla, ki je znašala $28.4 \cdot 10^6$ kNcm. Po optimizaciji se je deformacijska energija konstrukcije zmanjšala (za 77%) na $6.5 \cdot 10^6$ kNcm (slika 18). Debelina lupine se je zmanjšala na spodnjo mejo 5 cm, notranja višina se je zmanjšala na 4.65 m, poudaril pa se je lok v smeri osi *x* (slika 19).



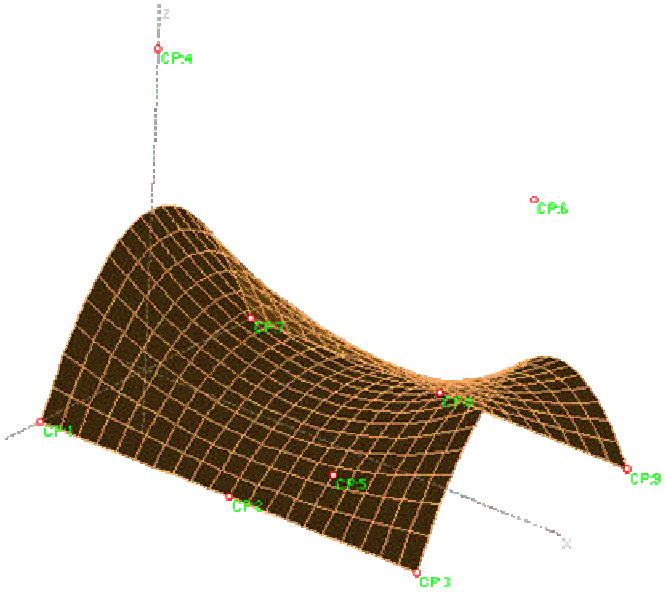
Slika 18: Deformacijske energije končnih elementov sedla pred (levo) in po optimizaciji (desno).



Slika 19: Optimalna oblika sedla – primer b.

(c) minimalni deformacijski energiji pri pogoju, da je prostornina konstrukcije V=50 m^3.

Optimalna oblika lupine (slika 20) je podobna kot v primeru (b). Spremenili sta se višini, zunanja višina se je povečala na 10.9 m, v sredini lupine pa zmanjšala na 5 m. Debelina lupine se je zaradi omejitve prostornine spremenila le na 7.5 cm. Smiselna bi bila omejitev prostornine sedla na manjšo vrednost. Deformacijska energija končnih elementov lupinaste konstrukcije se je zmanjšala (za 72%) z začetnih $28.4 \cdot 10^6$ kNcm na $7.9 \cdot 10^6$ kNcm.



Slika 20: Optimalna oblika sedla –primer c.

4.1.3 Komentar

Pri optimizaciji oblike lupine ob iskanju minimuma deformacijske energije je opazno veliko zmanjšanje deformacijske energije (do 77%). Manjša deformacijska energija pomeni več prenašanja obtežbe z membranskimi silami in manj z upogibom. Najbolje se to opazi pri optimalni obliki lupinastih konstrukcij (primer b in c) z dvojno ukrivljenostjo lupinaste konstrukcije. Pri optimizaciji oblike sedla bi bilo potrebno upoštevati še oblikovne omejitve (npr. višin) zaradi estetskih in funkcionalnih razlogov.

Preglednica 2: Projektne spremenljivke in optimalne vrednosti.

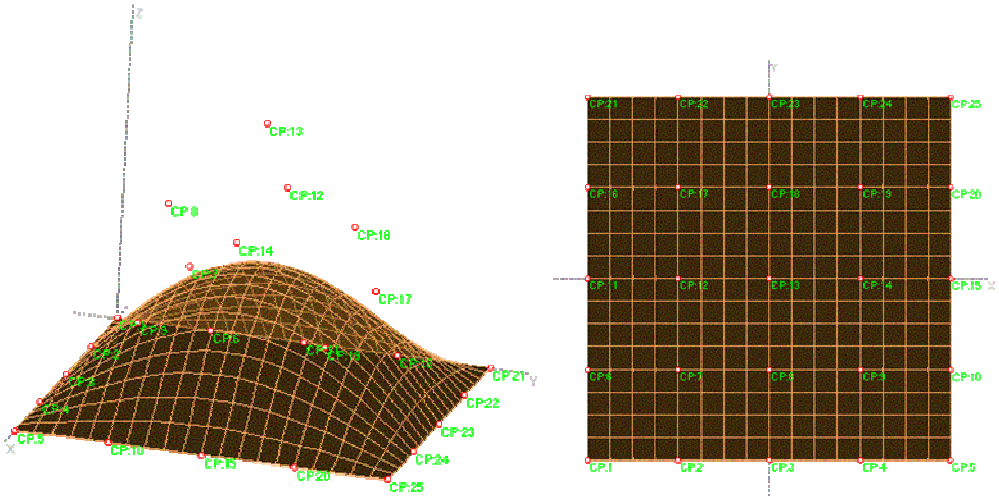
Projektna spremenljivka	Začetna vrednost	Interval	Optimalna vrednost (a)	Optimalna vrednost (b)	Optimalna vrednost (c)
z koordinata KT 4 in 6	10 m	[-40 m, 60 m]	0 m	10.15 m	10.9 m
z koordinata KT 5	7.5 m	[-17 m, 32 m]	0 m	4.65 m	5 m
debelina lupine	15 cm	[5 cm, 25 cm]	5 cm	5 cm	7.5 cm

4.2 AB lupina 1

4.2.1 Model 1

4.2.1.1 Opis primera

Obravnavana lupina ima kvadratni tloris 20x20 m, oblika je prikazana na spodnji sliki (slika 21). Lupina je vrtljivo podprta v vogalih. Višina točke v sredini lupine je 8 m. Začetna debelina h = 50 cm.

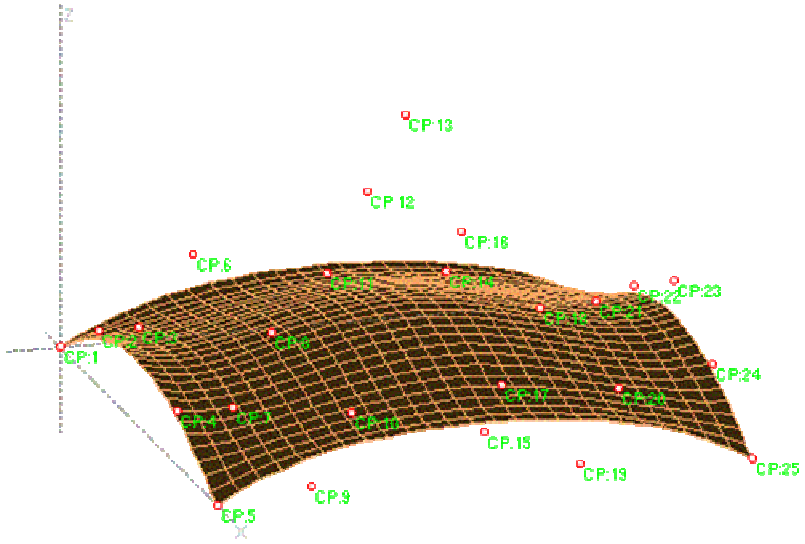


Slika 21: Začetna oblika lupine (levo 3D pogled, desno tloris) – model 1.

Materialne karakteristike uporabljenega betona so: elastični modul E = 30000 MN/m^3, Poissonov količnik ν = 0.3. Lupina je obtežena s težo snega 5000 kN/m^2 (Reitinger & Ramm,1995), ki deluje na tlorisno površino. Upošteval sem tudi lastno težo konstrukcije, ki jo program EKON izračuna glede na dejansko debelino lupine. Lupino sem modeliral s 625 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom s 25 kontrolnimi točkami (5x5x1). Ker sem upošteval simetrijo lupine za projektne spremenljivke, sem izbral samo 13 višin kontrolnih točk ter 13 spremenljivk, ki opisujejo gladko spreminjanje debeline lupine preko celotne konstrukcije. Spreminjanje projektnih spremenljivk sem omejil – debelino lupine na interval [5cm, 50cm], višine kontrolnih točk pa na [-10 m, 10 m].

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna pri pogoju, da je prostornina konstrukcije 60 m^3.

4.2.1.2 Rezultati

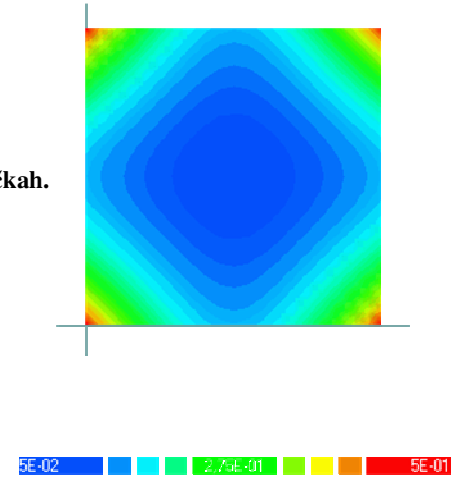


Slika 22: Oblika lupine po optimizaciji.

Po 16-ih iteracijah (slika 22) se je deformacijska energija zmanjšala (za 72%) s 26611 kNm na 7335 kNm, višina na sredini se je zmanjšala na 3,57m. Prostornina se je z začetnih 256 m^3 zmanjšala na zahtevanih 60 m^3. Debelina lupine se gladko spreminja s 50 cm v vogalu na 5 cm na sredini lupine.

Preglednica 3: Vrednosti debeline lupine (v cm) v kontrolnih točkah.

50	10.63473	6.832131	10.63473	50
10.43904	5.508757	5	5.508757	10.43904
6.832131	5.50395	5	5.50395	6.832131
10.43904	5.508757	5	5.508757	10.43904
50	10.63473	6.832131	10.63473	50

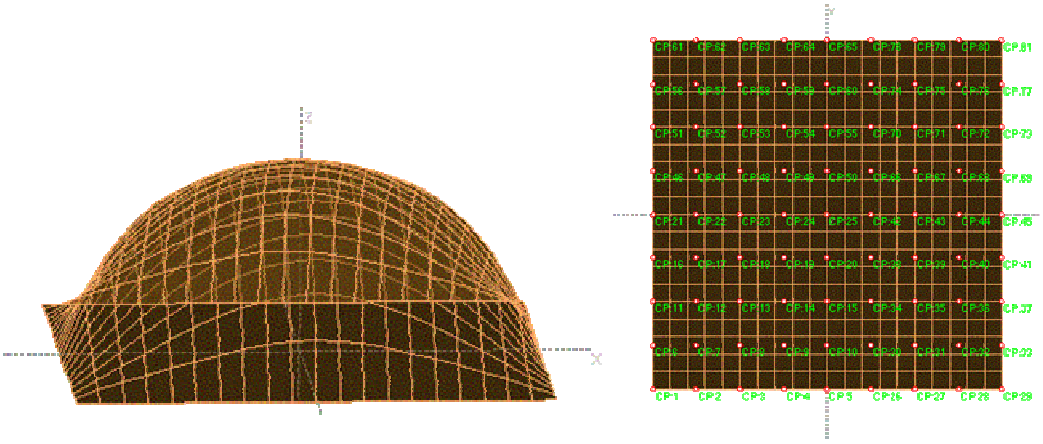


Slika 23: Grafični prikaz spreminjanja debeline lupine po konstrukciji.

4.2.2 Model 2

4.2.2.1 Opis primera

Začetna oblika obravnavane lupine kvadratne tlorisne oblike 20x20 m, vrtljivo podprte v vogalih (slika 24). Višina točke v sredini lupine je 8 m.

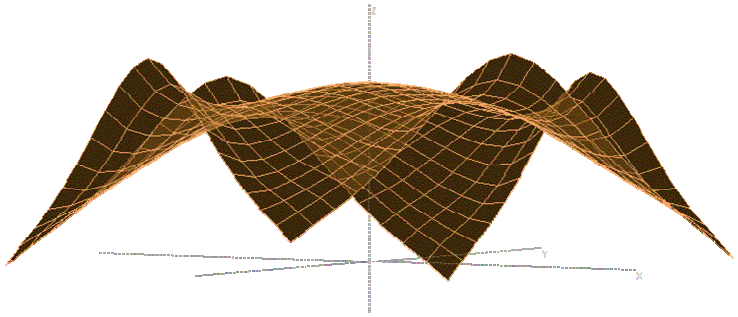


Slika 24: Začetna oblika lupine – model 2.

Materialne karakteristike uporabljenega betona so naslednje: elastični modul E = 30000 MN/m^3, Poissonov količnik ν = 0.3. Lupina je obtežena z lastno težo, ki jo program EKON izračuna glede na dejansko debelino lupine. Lupino sem modeliral s 400 lupinastimi končnimi elementi in s štirimi projektnimi elementi - Bézierovimi telesi s 25 kontrolnimi točkami (5x5x1) – skupaj 81 kontrolnih točk. Upošteval sem simetrijo konstrukcije preko x in y osi. Zaradi uporabe štirih projektnih teles se je pojavil problem zagotavljanja zveznosti in gladkosti celotne lupine. Zato sem to dejstvo upošteval z ustreznim postavljanjem kontrolnih točk. Tako sem uporabil le 11 projektnih spremenljivk – izbral sem višine 10 kontrolnih točk ter debelino lupine. Spreminjanje projektnih spremenljivk sem omejil – debelino lupine na interval [5 cm , 50 cm], višine kontrolnih točk pa na [-15 m, 15 m].

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna pri pogoju, da je prostornina konstrukcije 60 m^3.

4.2.2.2 Rezultati



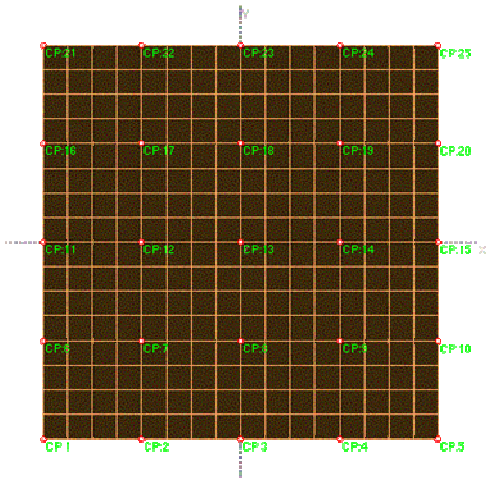
Slika 25: Oblika lupine po optimizaciji – model 2.

Že po šestih iteracijah optimizacije se je deformacijska energija zmanjšala (za 96%) z 19796 kNm na 808 kNm, višina na sredini se je zmanjšala na 6.83 m, na robu (kontrolna točka 21) pa se je povečala na 7.48 m (slika 25). Prostornina se je z začetnih 289 m^3 zmanjšala na zahtevanih 60 m^3. Debelina lupine se je zmanjšala s 50 cm na 12 cm.

4.2.3 Model 3

4.2.3.1 Opis modela

Začetna oblika obravnavane lupine kvadratne tlorisne oblike 20x20 m, vrtljivo podprte v vogalih (slika 26). Začetna oblika lupine je (ravna) plošča v ravnini *xy*.

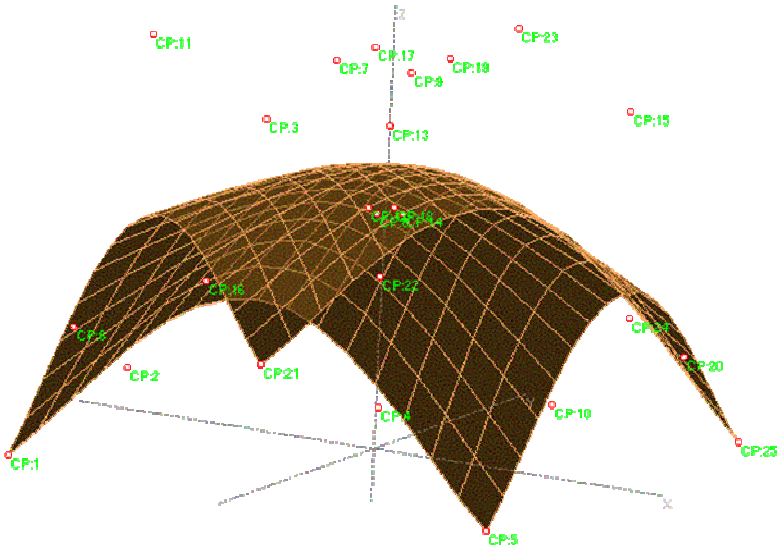


Slika 26: Začetna oblika lupine – model 3.

Materialne karakteristike so naslednje: elastični modul E = 30000 MN/m^3, Poissonov količnik ν = 0.3. Lupina je obteženo s težo snega 5000 kN/m^2 na tlorisno površino. Lupino sem modeliral z 256 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom s 25 kontrolnimi točkami (5x5x1). Za projektne spremenljivke sem izbral 21 višin kontrolnih točk, 20 spremenljivk, ki opisujejo pozicijo kontrolnih točk (v x in y smeri), ter 25 spremenljivk, ki opisujejo gladko spreminjanje debeline lupine. Spreminjanje projektnih spremenljivk sem omejil – debelino lupine na interval [7.5 cm, 50 cm], spremembo pozicije (x in y koordinate) kontrolnih točk na [-5 m, 5 m], ter višine kontrolnih točk (z – koordinate) na [0 m, 40 m].

4.2.3.2 Primer a

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna, pri pogojih, da je prostornina konstrukcije 60 m^3 in višina točke na sredini več kot 8 m.

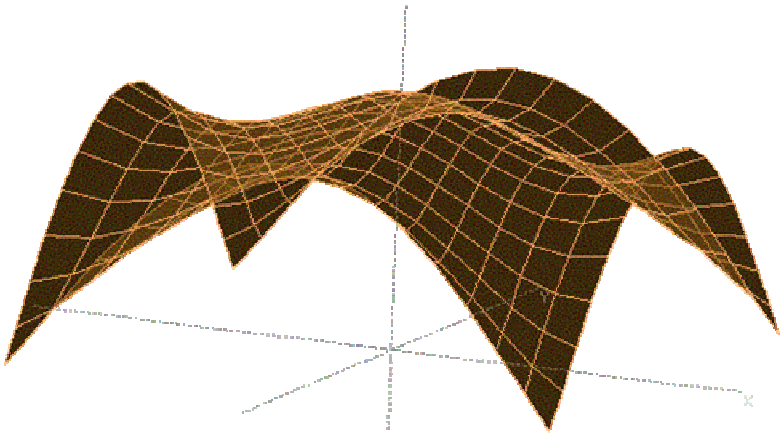


Slika 27: Optimalna oblika lupine – model 3a.

Po 14-ih iteracijah se je deformacijska energija zmanjšala (za 98%) s 42865 kNm na 540 kNm, višina na sredini se je z začetnih 0 m povečala na 10.6 m, na robu med dvema podporama pa se je povečala z začetnih 0 m na 7.3 m. Prostornina se je z začetnih 200 m^3 zmanjšala na zahtevanih 60 m^3. Debelina lupine se gladko spreminja s 25 cm v vogalu na 7.5 cm na sredini lupine.

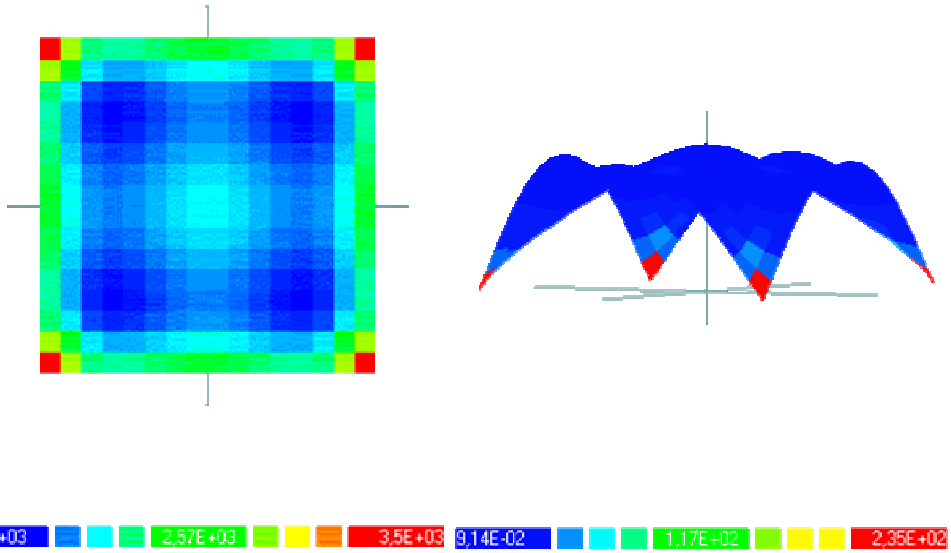
4.2.3.3 Primer b

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna pri pogojih, da je prostornina konstrukcije 60 m^3, višina točke na sredini več kot 8 m in višina točk na robu ($x = 0$, $y = 0$) 8 m.



Slika 28: Optimalna oblika lupine – model 3b.

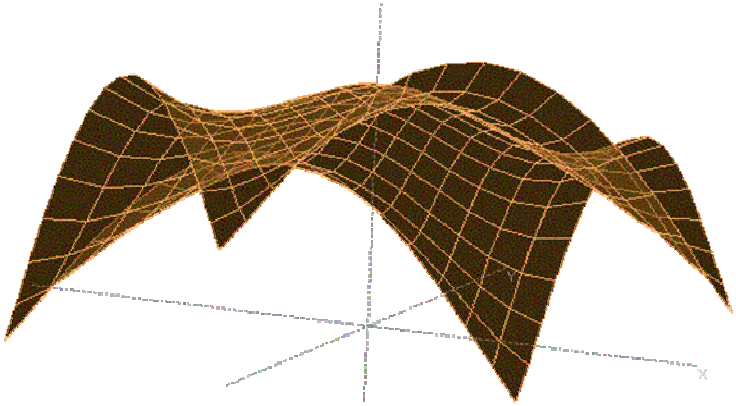
Po 10-ih iteracijah se je deformacijska energija (slika 29) zmanjšala (za 95 %) z 42865 kNm na 2104 kNm, višina na sredini se je z začetnih 0 m povečala na 8.8 m, na robu med podporama pa se je povečala z začetnih 0 m na zahtevanih 8 m. Prostornina se je z začetnih 200 m^3 zmanjšala na zahtevanih 60 m^3. Debelina lupine se gladko spreminja s 35 cm v vogalu na 7.5 cm na sredini lupine.



Slika 29: Deformacijska energija pred (levo) in po optimizaciji (desno) - model 3b.

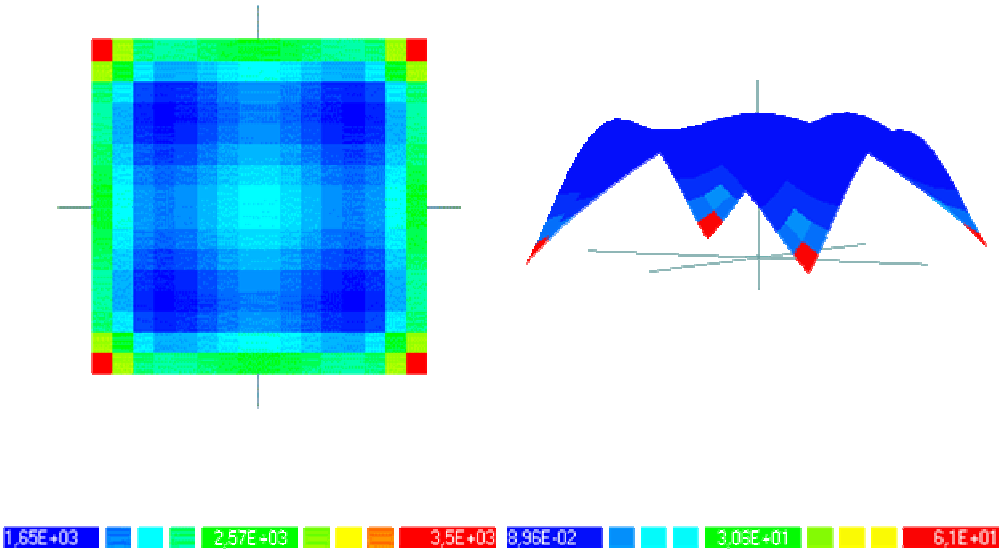
4.2.3.4 Primer c

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna, pri pogojih, da je prostornina konstrukcije 60 m^3, višina točke na sredini je omejena med 8 m in 9 m, ter višina točk na robu je 8 m.



Slika 30: Optimalna oblika lupine – model 3c.

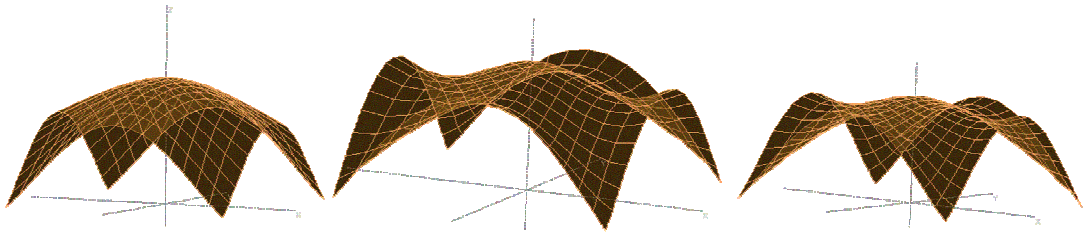
Po 10-ih iteracijah se je deformacijska energija (slika 31) zmanjšala (za 99%) z 42865 kNm na 579 kNm, višina na sredini se je z začetnih 0 m povečala na 8.6 m, na robu pa se je povečala z začetnih 0 m na zahtevanih 8 m. Prostornina se je z začetnih 200 m^3 zmanjšala na zahtevanih 60 m^3. Debelina lupine se gladko spreminja s 35 cm v vogalu na 7.5 cm na sredini lupine.



Slika 31: Deformacijska energija pred (levo) in po optimizaciji (desno) – model 3c.

4.2.4 Komentar rezultatov

Iz navedenih primerov – modelov iste konstrukcije je razvidno, da pristop s fiksnimi kontrolnimi točkami v x in y smeri ni zadosten. Bolje je že na začetku modelirati konstrukcijo z več projektnimi spremenljivkami in jih dobro izbrati ter dopustiti premike kontrolnih točk Bézierovega telesa v vse tri smeri (x, y in z), da se v procesu optimizacije postavijo na najbolj optimalen položaj (v x, y in z smeri). Rezultat – optimalna oblika lupinaste konstrukcije – se zelo spreminja glede na dane omejitvene pogoje. To je lepo vidno v treh primerih modela 3 (slika 32).

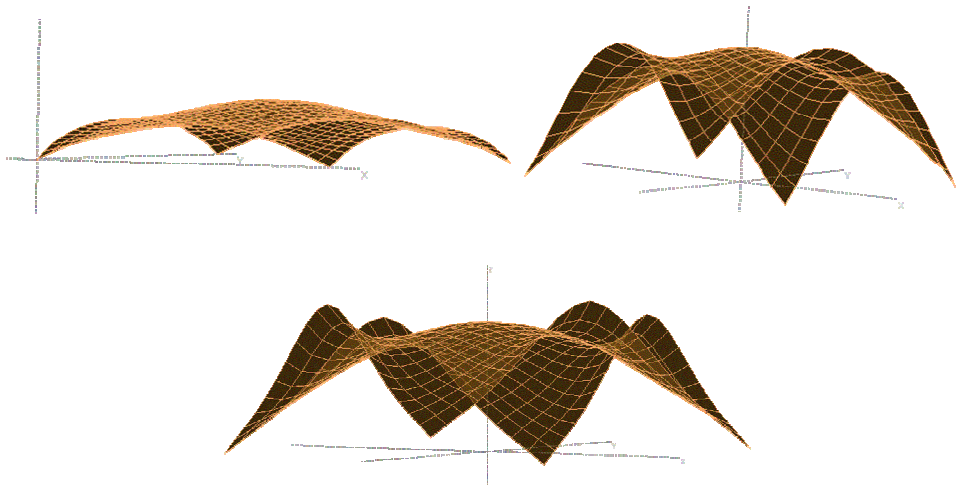


Slika 32: Primerjava rezultatov modela 3 (a, b, in c).

Zaradi simetričnosti lupinaste konstrukcije sem v vseh modelih upošteval simetrijo pri postavljanju projektnih spremenljivk. V prvem modelu ni omejitve višin posameznih točk na lupini, zato ni tako izrazitih lokov.

V drugem modelu so uporabljena štiri Bézierova telesa, a ker program EKON zaenkrat še ne omogoča gladkega sestavljanja Bézierovih teles, sem v modelu uporabil le 11 projektnih spremenljivk - višin kontrolnih točk – (od možnih 81), vse ostale sem potreboval za zadostitev gladkosti celotne lupine.

V tretjem modelu sem z enim projektnim telesom izkoristil vse programske zmogljivosti, povečal število projektnih spremenljivk ter dobil najnatančnejši model. Zmanjšal sem število končnih elementov, saj se čas računanja vsake iteracije močno podaljša s številom projektnih spremenljivk in končnih elementov (zaradi numeričnega računanja odvodov in namenske funkcije). Vse, kar je bilo še potrebno narediti, je bilo postaviti ustrezne oblikovne pogoje (omejitev višin).



Slika 33: Primerjava rezultatov modelov 1, 2 in 3c.

V spodnji preglednici (preglednica 4) so prikazani nekateri podatki o vsakem od modelov lupinaste konstrukcije, ki sem jih uporabil za optimizacijo oblike.

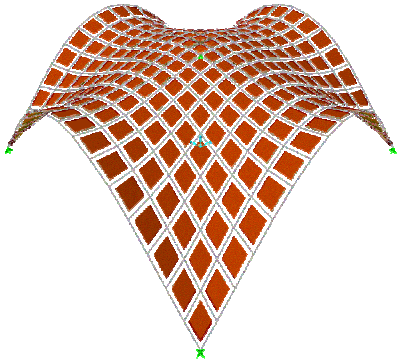
Preglednica 4: Primerjava modelov iste konstrukcije.

	Model 1	Model 2	Model 3a	Model 3b	Model 3c
Začetna oblika	Slika 21	Slika 24	Slika 26	Slika 26	Slika 26
Optimalna oblika	Slika 22	Slika 25	Slika 27	Slika 28	Slika 30
Št. Bézierovih teles	1	4	1	1	1
Št. kontrolnih točk	25	81	25	25	25
Št. lupinastih končnih elementov	625	400	256	256	256
Št. projektnih spremenljivk za višino	13	10	21	21	21
Št. projektnih spremenljivk za *x*, *y* lego	0	0	20	20	20
Št. projektnih spremenljivk za debelino	13	1	25	25	25

4.2.5 Račun modela c s programom SAP2000

S programom SAP2000 sem izvedel nelinearno analizo lupinaste konstrukcije. Za model sem izbral optimalno obliko primera 3c. Obliko konstrukcije sem prenesel iz programa EKON z uporabo programa Mathematica in napisane funkcije. Tako sem lahko analiziral obliko modela popolnoma identično obliki konstrukcije v EKONu. V programu SAP2000 sem

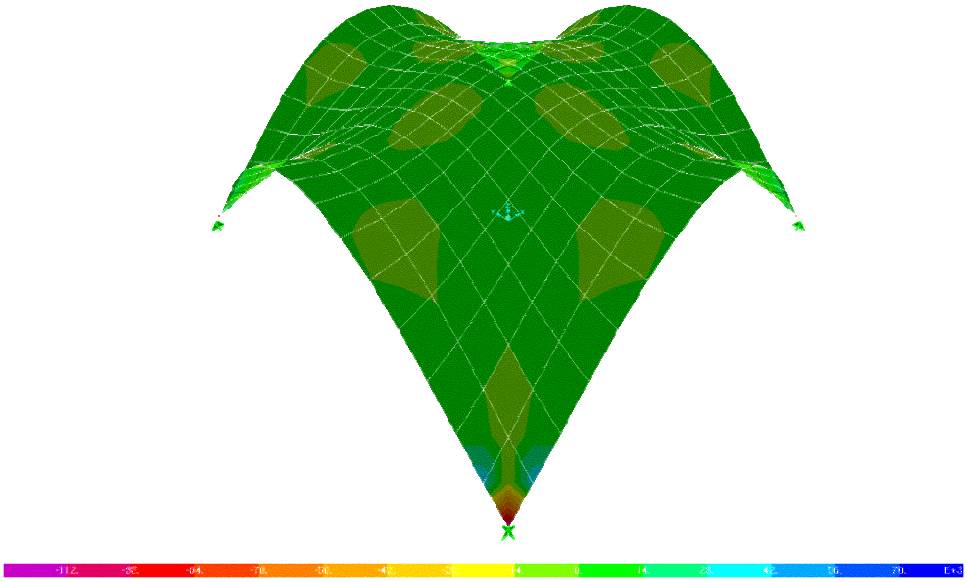
upošteval enake materialne in obtežne karakteristike, upošteval sem tudi spreminjanje debeline konstrukcije. Glede na veliko občutljivost ploskovnih konstrukcij na uklon sem preveril tudi uklonsko nosilnost lupinaste konstrukcije.



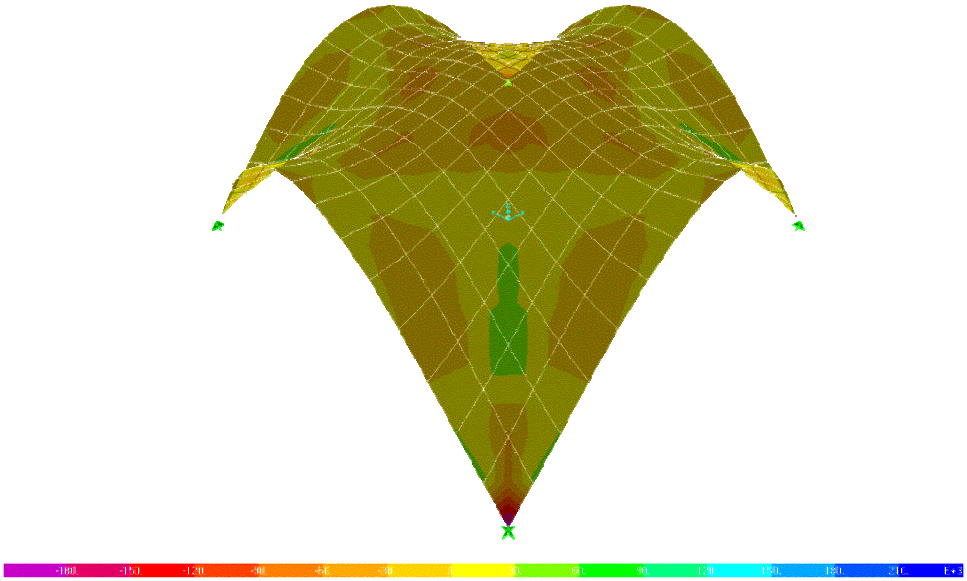
Slika 34: Model lupinaste konstrukcije v programu SAP2000.

4.2.5.1 Rezultati

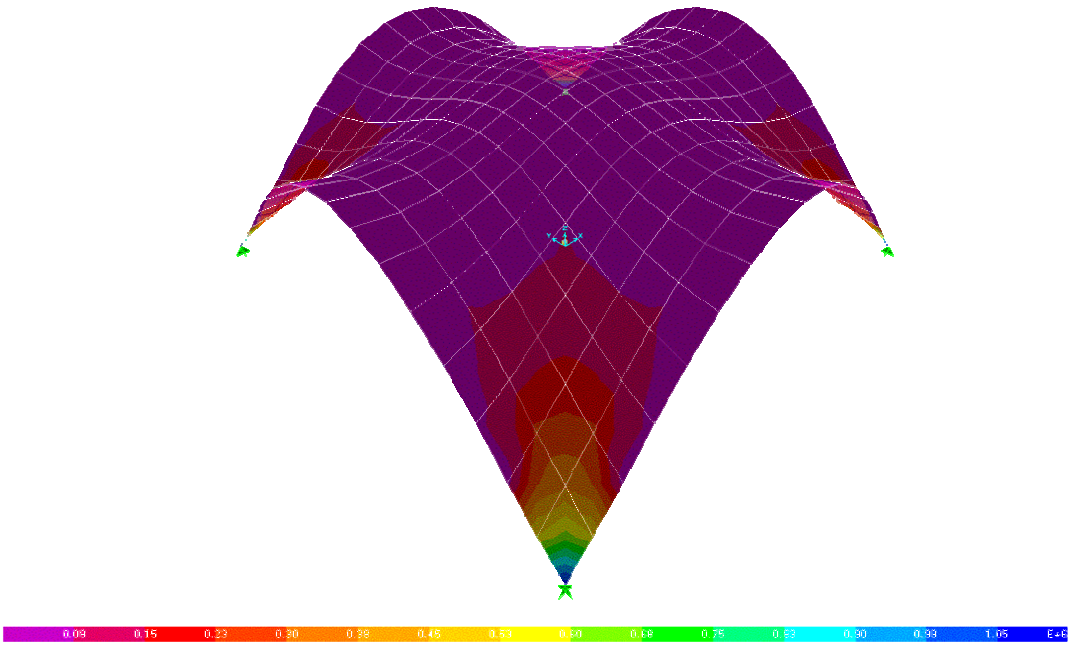
Rezultati linearne analize dane lupinaste konstrukcije so bili pričakovani – majhni upogibni momenti po celotni lupini (red velikosti 1 kNm/m), izjema so le podpore, kjer se pojavijo konice upogibnih momentov (110 kNm/m) ter primerno velike membranske sile po celotni lupini (red velikosti 50 kN/m). Izjema so spet podpore, kjer se pojavijo konice sil (230 kN/m). Podobno je tudi pri prikazu notranjih sil po pogoju von Mises-a, kjer je sila po celotni lupini (60 kN/m), konice pa so pri podporah (1100 kN/m).



Slika 35: Diagram upogibnih momentov MMAX.



Slika 36: Diagram membranskih sil FMAX.



Slika 37: Diagram notranjih sil FVM.

S programom SAP2000 sem izračunal prvih 10 uklonskih oblik in uklonskih faktorjev, ki so navedeni v spodnji tabeli (preglednica 5), pri obtežbi s težo snega (kot v primeru modela 3c).

Preglednica 5: Uklonska nosilnost lupinaste konstrukcije.

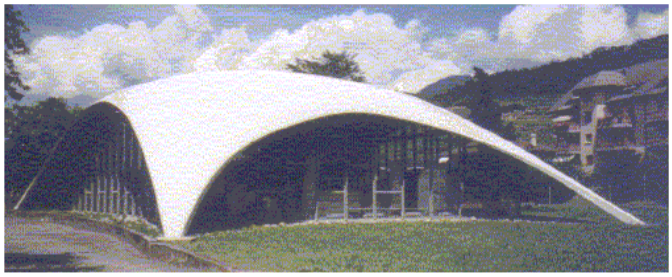
Uklonska oblika	Brezdimenzionalni uklonski faktor
1	11.259066
2	13.020961
3	13.020965
4	17.644421
5	24.434128
6	24.434167
7	34.484470
8	38.666468
9	47.891660
10	47.929918

4.3 AB lupina 2

4.3.1 Model 1

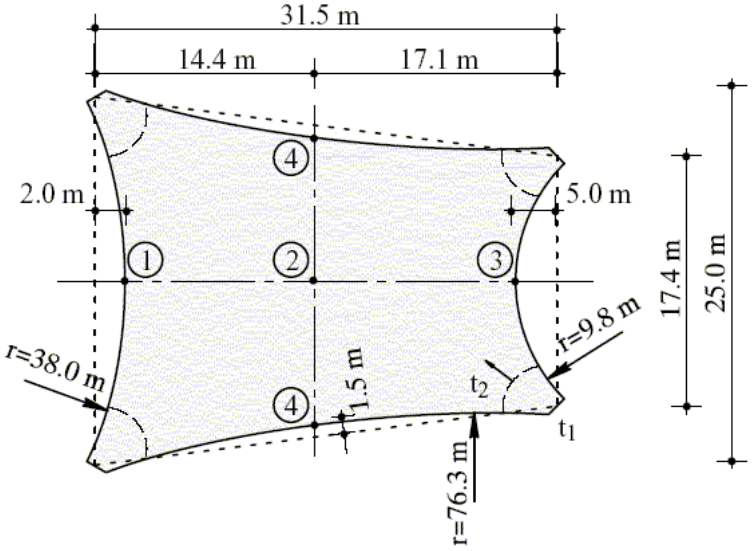
4.3.1.1 Opis primera

Primer sem črpal iz članka (Ramm, Kemmler in Schwarz, 2000). Obravnaval sem primer obstoječe konstrukcije, ki jo je projektiral Heinz Isler leta 1968 (slika 38).



Slika 38: Laboratorij Gips-Union, Bex.

Podatki o obliki lupine so analogni glede na obstoječo konstrukcijo v Bex-u (slika 39).



Točka	Višina točke
1	5.23 m
2	8.34 m
3	5.09 m
4	6.06 m

Preglednica 6: Višine točk na lupini.

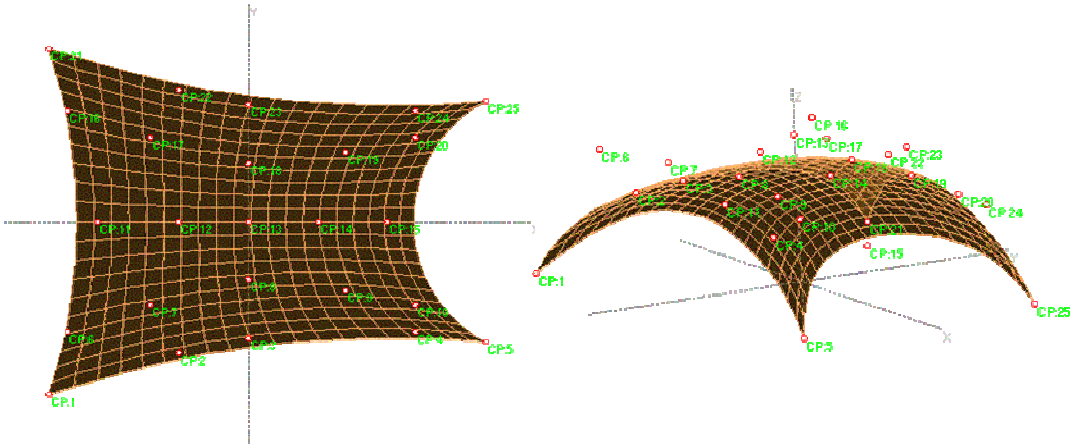
Debelina lupine se spreminja:

- ob podporah: t_1 = 15 cm;
- sicer pa: t_2 = 8 cm.

Slika 39: Podatki o obliki konstrukcije.

Konstrukcija je vpeta v vseh štirih vogalih. Materialne karakteristike uporabljenega betona so naslednje: elastični modul E = 30000 MN/m^3, Poissonov količnik ν = 0.3. Konstrukcija je obtežena z lastno težo – za specifično težo armiranobetonske konstrukcije sem upošteval γ =

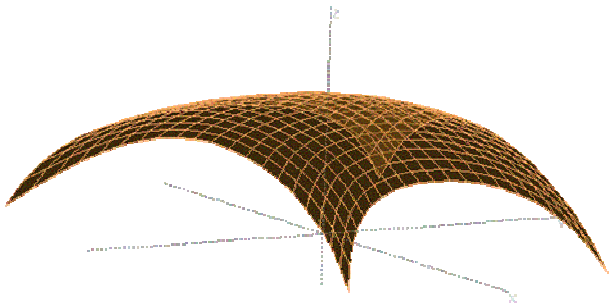
25 kN/m^2, lastno težo pa program EKON izračuna glede na dejansko debelino lupine. Lupino sem modeliral s 400 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom s 25 kontrolnimi točkami (5x5x1). Za projektne spremenljivke sem izbral 13 višin kontrolnih točk (upošteval sem simetrijo lupine glede na os *x*). Začetne višine sem določil s pomočjo Bézierovih krivulj in višin točk 1 do 5 (Preglednica 6), tako da sem že za začetno obliko konstrukcije dobil enakomerno gladko lupino. Glede na to, da so začetne *z* koordinate kontrolnih točk različne, sem omejil spremembe višin na [-10 m, 10 m].



Slika 40: Začetni model konstrukcije, tloris (levo) in 3D pogled (desno) – model 1.

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna.

4.3.1.2 Rezultati



Slika 41: Optimalna oblika lupine - model 1.

Preglednica 7: Višine točk pred in po optimizaciji – model 1.

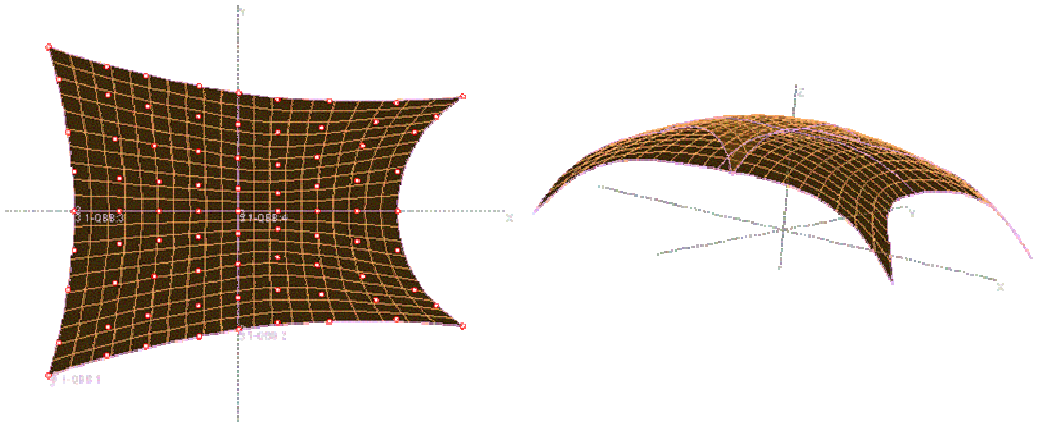
Točka	Višina točke	Višina točke po optimizaciji
1	5.23 m	4.42 m
2	8.34 m	9.39 m
3	5.09 m	5.68 m
4	6.06 m	4.76 m

Večjih sprememb same oblike po optimizaciji ni videti, je pa opazna poudarjena dvojna ukrivljenost lupinaste konstrukcije. Zaradi tega se na robovih zmanjšajo višine, v sredini pa se lupina izboči. Močno se je zmanjšala (za 89 %) deformacijska energija konstrukcije iz začetnih 16066 kNm na 1794 kNm (po 23 iteracijah).

4.3.2 Model 2

4.3.2.1 Opis modela

Geometrijski podatki o obliki lupine so enaki kot pri prvem modelu (slika 39).
Lupino sem modeliral s 400 lupinastimi končnimi elementi in s štirimi projektnimi elementi - Bézierovimi telesi s po 25 kontrolnimi točkami (5x5x1).



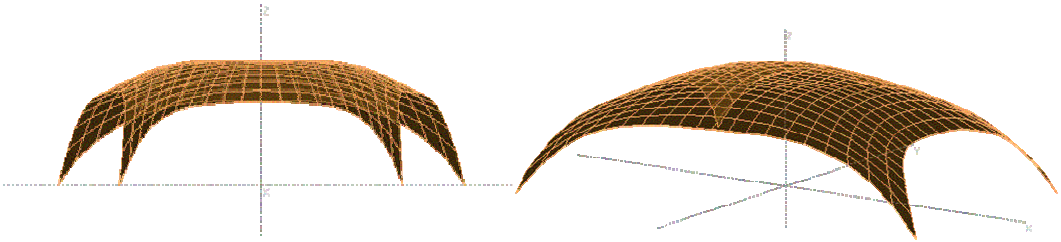
Slika 42: Začetna oblika lupinaste konstrukcije - model 2.

Za projektne spremenljivke sem izbral 45 višin (z koordinat) kontrolnih točk. Zaradi zadostitve gladkosti sestavljene lupine je bila izbira projektnih spremenljivk omejena. Z zadostitvijo gladkosti sem dobil le 14 projektnih spremenljivk (od možnih 49-ih). Začetne višine sem določil s pomočjo Bézierovih krivulj in višin točk 1 do 5 (preglednica 6), tako da

sem za začetno obliko konstrukcije dobil enakomerno gladko lupino. Glede na to, da so začetne *z* koordinate kontrolnih točk različne, sem omejil spremembe višin na [-10 m, 10 m].

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna, pri pogoju, da volumen konstrukcije ostane nespremenjen (76,6 m^3).

4.3.2.2 Rezultati



Slika 43: Lupina po optimizaciji - model 2.

Preglednica 8: Višine točk pred in po optimizaciji – model 2.

Točka	Višina točke	Višina točke po optimizaciji
1	5.23 m	5.20 m
2	8.34 m	7.63 m
3	5.09 m	5.09 m
4	6.06 m	5.72 m

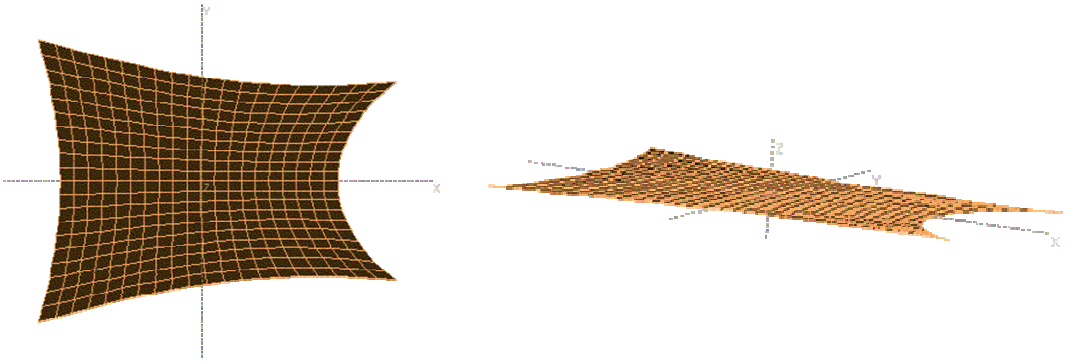
Z omejitvijo prostornine se višine točk po optimizaciji ne spremenijo toliko. Opazna je poudarjena dvojna ukrivljenost lupinaste konstrukcije, ter zaradi spajanja štirih projektnih elementov nekoliko vbočena sredina (notranjih 9 kontrolnih točk ima enako višino) lupine. Zmanjšanje deformacijske energije konstrukcije ni tako izrazito. Deformacijska energija se zmanjša (za 12 %) iz 52171 kNm na 40722 kNm (po 21 iteracijah).

4.3.3 Model 3

4.3.3.1 Opis modela

Podatki o obliki lupine so enaki kot pri prvem modelu (slika 39).

Lupino sem modeliral s 400 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom z 81 kontrolnimi točkami (9x9x1). Za projektne spremenljivke sem izbral 43 višin (z koordinat) kontrolnih točk. Upošteval sem simetrijo konstrukcije preko x osi. z-koordinate kontrolnih točk sem omejil na [-10 m, 10 m]. Začetni model konstrukcije je plošča (slika 44).



Slika 44: Oblika začetnega modela (levo – tloris, desno – 3D pogled) – model 3.

4.3.3.2 Rezultati

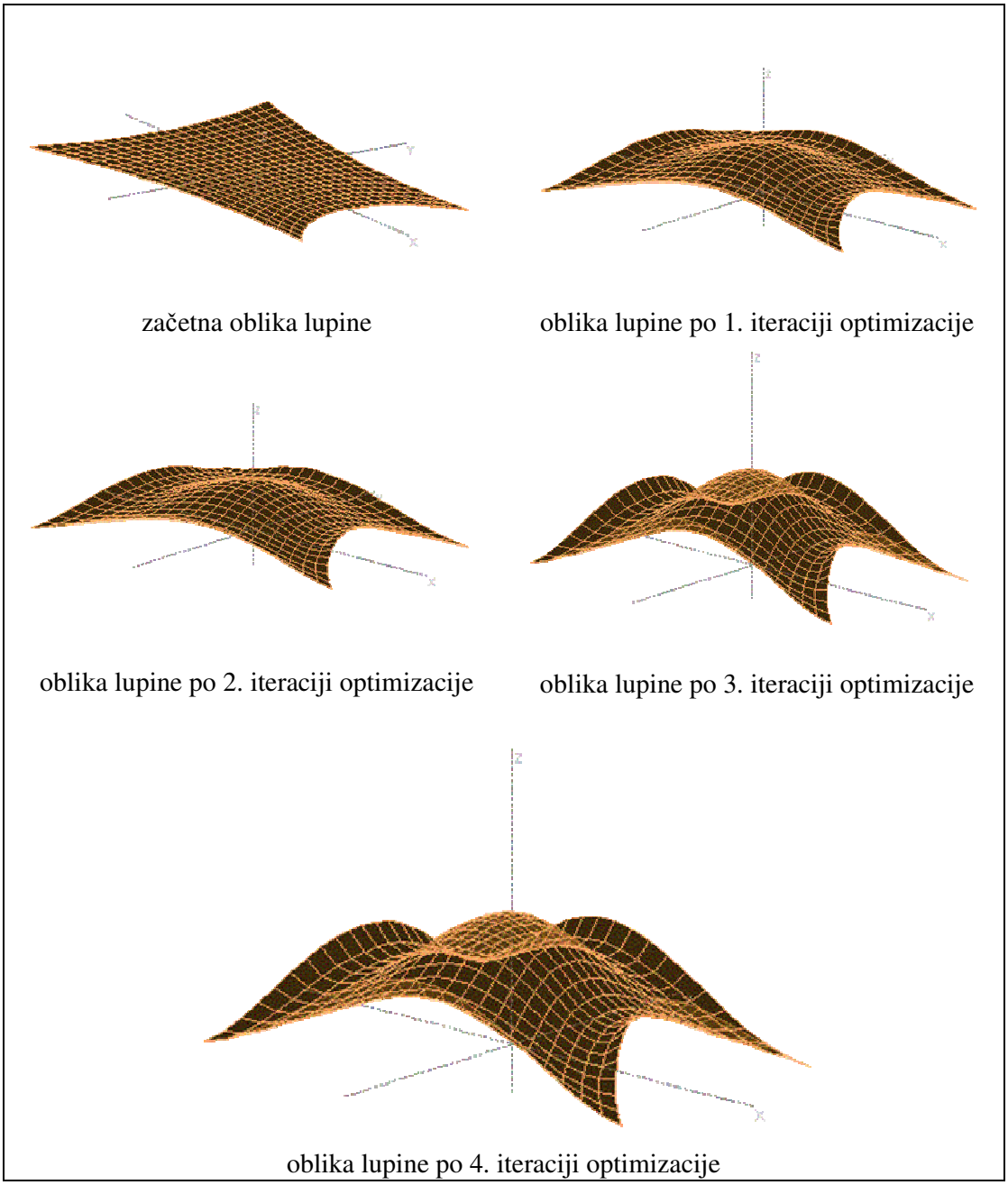
Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna, pri pogojih:

- predpisane so višine robnih točk (1 do 4).

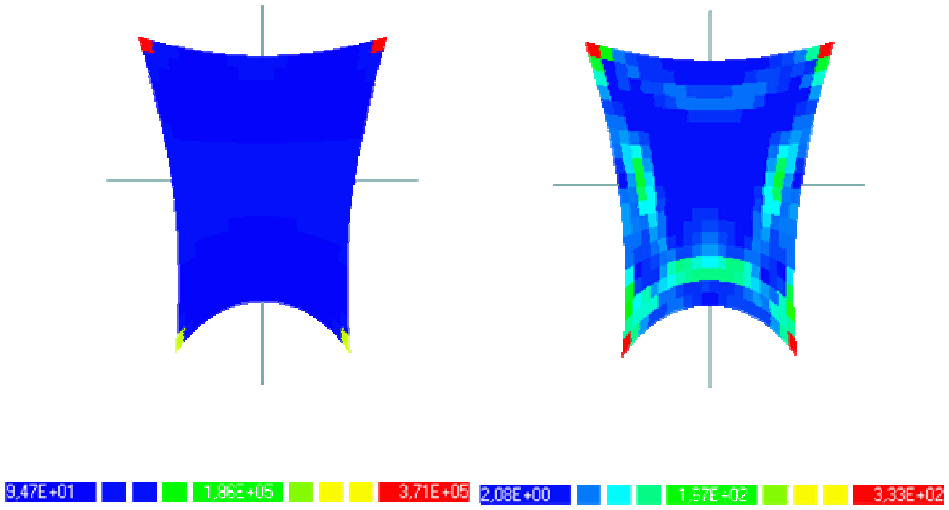
Preglednica 9: Predpisane višine točk na lupini.

Točka	Višina točke
1	5.23 m
2	8.34 m
3	5.09 m
4	6.06 m

Optimalna oblika konstrukcije je precej različna od modela 2. Očitno je možno več optimalnih oblik, velik vpliv na to ima izbira projektnih spremenljivk in izbira začetne oblike lupinaste konstrukcije. Začetna prostornina lupinaste konstrukcije 44.6 m^3 se je povečala na 53.11 m^3. Zaradi poenostavljene začetne oblike in omejitvenih pogojev se tvorijo očitni loki na robovih in opazna je izrazita dvojna ukrivljenost optimalne oblike lupinaste konstrukcije (minimum deformacijske energije). Deformacijska energija konstrukcije se je močno zmanjšala (za 165 krat) iz 2483814 kNm na 15142 kNm (že po samo štirih iteracijah).



Slika 45: Oblike konstrukcije po iteracijah - model 3.



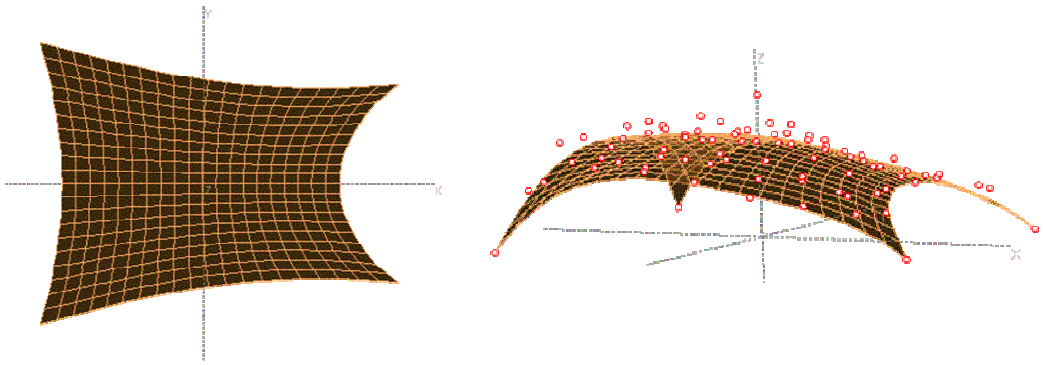
Slika 46: Primerjava deformacijske energije pred (levo) in po (desno) optimizaciji (skala ni enaka).

4.3.4 Model 4

4.3.4.1 Opis modela

Podatki o obliki lupine so enaki kot pri prvem modelu (slika 39).

Lupino sem modeliral s 400 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom z 81 kontrolnimi točkami (9x9x1). Za projektne spremenljivke sem izbral 43 višin (z-koordinat) kontrolnih točk, 24 premikov v x smeri in 21 premikov v y smeri. Skupaj je tako 88 projektnih spremenljivk. Upošteval sem simetrijo preko x osi. Višine z-koordinate kontrolnih točk sem omejil na [-10 m, 10 m], možne premike točk od začetnih vrednosti v x in y smeri pa na [-2 m, 2 m]. Začetni model lupinaste konstrukcije je prikazan na naslednji sliki (slika 47).

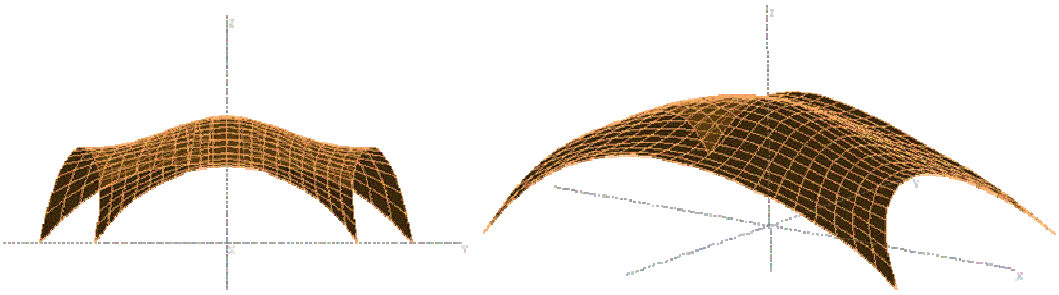


Slika 47: Prikaz oblike začetnega modela (leva slika – tloris, desna slika – 3D pogled) – model 4.

4.3.4.2 Rezultati

Naloga je poiskati optimalno obliko konstrukcije, kjer bo deformacijska energija konstrukcije minimalna, pri pogojih:

- predpisane so višine robnih točk (1 do 4) (preglednica 6).

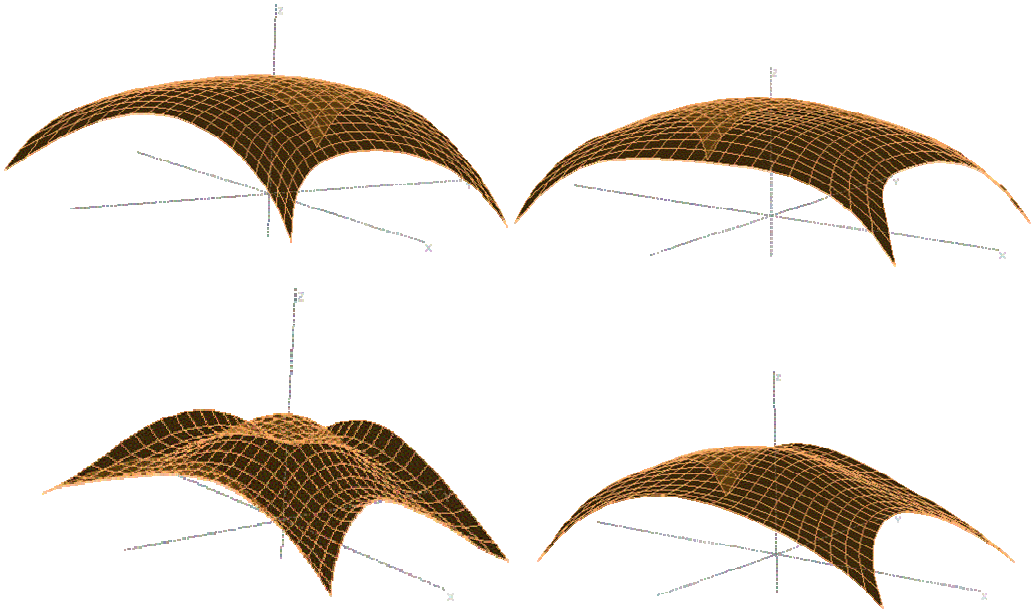


Slika 48: Optimalna oblika lupinaste konstrukcije - model 4.

Zaradi velikih skokov med posameznimi iteracijami (tako deformacijske energije kot omejitvenih pogojev) sem pri optimizaciji uporabil zmanjšane korake (v programu iGO sem uporabil različne »Curvature index«). Optimalna oblika lupinaste konstrukcije je po 8 iteracijah optimizacije zelo podobna obliki pri modelu 2. Glede na dovoljeno premikanje kontrolnih točk v x in y smeri so bili premiki le-teh minimalni (red velikosti 10^{-6} m). Večji pomen je imela začetna oblika lupinaste konstrukcije. Prostornina lupinaste konstrukcije se je minimalno povečala s 47.8 m^3 na 48.6 m^3. Deformacijska energija konstrukcije se je zmanjšala (za približno 2 krat) iz 42529 kNm na 23073 kNm (po 8 iteracijah).

4.3.5 Komentar rezultatov

Iz navedenih primerov – modelov iste konstrukcije, je razvidno, da je možno več optimalnih oblik (slika 49) glede na začetno obliko lupinaste konstrukcije in število projektnih teles in projektnih spremenljivk ter omejitvenih pogojev, saj sem pri vseh modelih za namensko funkcijo uporabil minimum deformacijske energije.



Slika 49: Primerjava rezultatov modelov 1, 2, 3 in 4.

Zaradi simetričnosti lupinaste konstrukcije sem v vseh modelih upošteval simetrijo preko x osi pri postavljanju projektnih spremenljivk. V prvem modelu je začetna oblika že precej podobna optimalni. Zaradi majhnega števila projektnih spremenljivk, in ker nisem postavil omejitvenih pogojev višin petih točk, se je lupina na robu in sredini nekoliko znižala (preglednica 7).

V drugem modelu sem povečal število projektnih teles na štiri in povečal število projektnih spremenljivk. Višin petih točk (preglednica 6) spet nisem podal kot omejitvenih pogojev, a je razlika med višinami optimalne oblike in zahtevanimi višinami že manjša, nekoliko pa se lupina na sredini tudi vboči, kar je posledica združevanja štirih projektnih teles. Zaradi dobrega približka začetne oblike ni velikega zmanjšanja deformacijske energije konstrukcije.

V tretjem modelu sem v enem projektnem telesu povečal število kontrolnih točk na 81 (9x9x1). Projektne spremenljivke so tudi v tem modelu le *z*-koordinate kontrolnih točk. Za omejitvene pogoje sem postavil dejanske višine (Ramm, Kemmler in Schwarz, 2000) in za začetno obliko postavil ploščo v ravnino *xy*. Ravno zaradi izbire začetne oblike se je deformacijska energija optimalne konstrukcije močno zmanjšala, je pa optimalna oblika zelo zanimiva. Na vseh robovih se tvorijo očitni loki, ki obtežbo prenašajo z membranskimi silami, kar je pričakovano glede na izbiro namenske funkcije (deformacijska energija). Tudi slika deformacijske energije po končnih elementih pokaže večje vrednosti ravno pri teh lokih (slika 46). Pričakoval sem obliko, ki bi bila bolj podobna dejansko zgrajeni (slika 38).

V četrtem modelu sem glede na tretji model spremenil začetno obliko lupinaste konstrukcije in omogočil premike kontrolnih točk tudi v *x* in *y* smeri. V samem postopku optimizacije se je oblika lupine v iteracijah močno spreminjala, zato sem omejil velikosti sprememb vrednosti projektnih spremenljivk, in tako dobil rezultat, ki je dejansko najbolj podoben zgrajenemu (kar sicer ni bil moj cilj). Lepo vidna je dvojna ukrivljenost lupine in tudi v tem modelu se tvorijo loki na robovih.

	Model 1	Model 2	Model 3	Model 4
Začetna oblika	Slika 40	Slika 42	Slika 44	Slika 47
Optimalna oblika	Slika 41	Slika 43	Slika 45	Slika 48
Št. Bézierovih teles	1	4	1	1
Št. kontrolnih točk	25(5x5x1)	81(4x25)	81(9x9x1)	81(9x9x1)
Št. lupinastih končnih elementov	400	400	400	400
Št. projektnih spremenljivk za višino	13	14	43	43
Št. projektnih spremenljivk za *x*,*y* lego	0	0	0	45(24+21)
Deformacijska energija optimalne konstrukcije [v kNm]	1794	40722	15142	23073

Preglednica 10: Primerjava modelov.

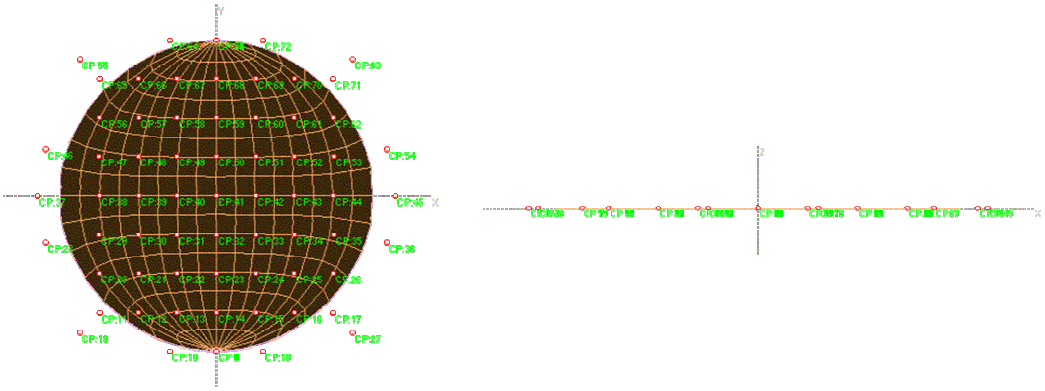
V zgornji tabeli so prikazani podatki o vseh modelih lupinaste konstrukcije, ki sem jih uporabil za optimizacijo oblike (preglednica 10).

4.4 Enostavni primeri

4.4.1 Okrogla prostoležeča plošča

4.4.1.1 Opis konstrukcije

Primer sem črpal iz literature (Camprubi, Bischoff in Bletzinger, 2002). Začetna oblika obravnavane lupine je prostoležeča jeklena okrogla plošča z radijem 10 m, kot je prikazano na spodnji sliki (slika 50). Začetna debelina lupine je h = 1 cm.



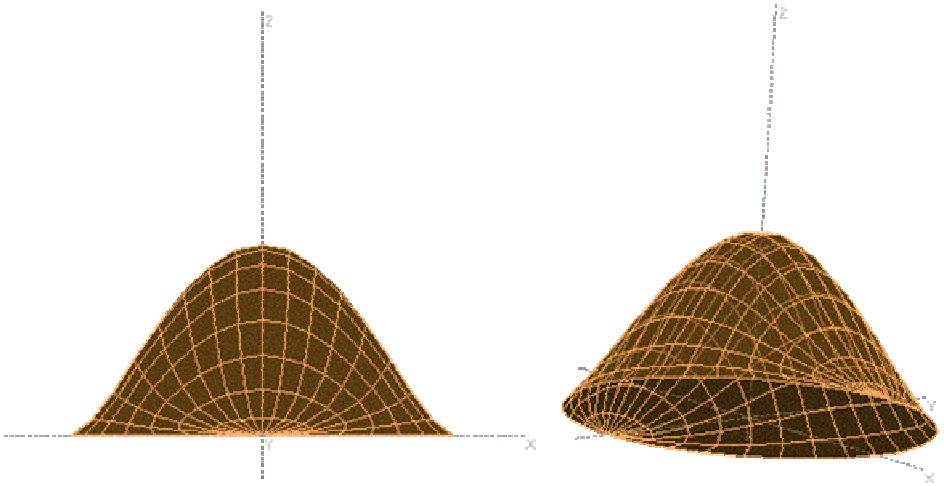
Slika 50: Začetna oblika lupine (levo – tloris, desno – pogled s strani).

Materialne karakteristike uporabljenega jekla so naslednje: elastični modul E = 21000 MN/m^3, Poissonov količnik ν = 0.3. Lupino sem modeliral z 256 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom z 81 kontrolnimi točkami (9x9x1). Glede na simetričnost lupinaste konstrukcije sem upošteval dvojno simetrijo (preko x in y osi) pri definiciji projektnih spremenljivk. Za projektne spremenljivke sem izbral 16 višin kontrolnih točk, 24 (12 + 12) spremenljivk, ki opisujejo dopusten premik kontrolnih točk v x in y smeri ter 1 spremenljivko za spreminjanje debeline lupine. Spreminjanje projektnih spremenljivk sem omejil – debelino lupine na interval [1 cm, 10 cm], spremembe pozicij v x in y smeri na interval [-80 cm , 80 cm] in višine kontrolnih točk na [-20 m, 20 m].

4.4.1.2 Obtežba - lastna teža

Naloga je poiskati optimalno obliko konstrukcije, obtežene le z lastno težo, kjer bo deformacijska energija konstrukcije minimalna, pri pogojih, da je višina v sredini enaka radiju (10 m) in prostornina konstrukcije omejena na 5 m^3.

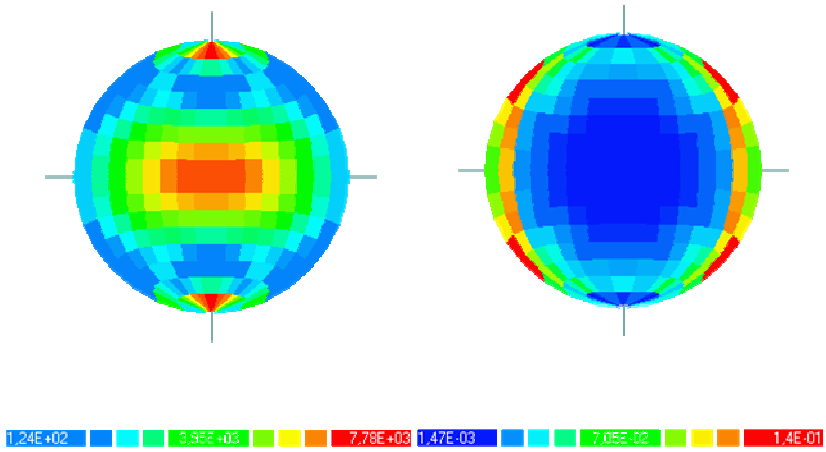
Razlog za postavitev omejitve višine v sredini je, da se je v primeru brez omejitve pri optimizaciji spreminjala le debelina lupine, ne pa višine kontrolnih točk.



Slika 51: Optimalna oblika lupinaste konstrukcije.

Optimalna oblika se približuje obliki zvona – in ne obliki polovice sfere. Opazno je izrazito zmanjšanje deformacijske energije, saj se le-ta zmanjša (za 90000 krat) z začetnih 815993 kNm na 9 kNm. Lupinasta konstrukcija ima tudi po optimizaciji debelino 1 cm.

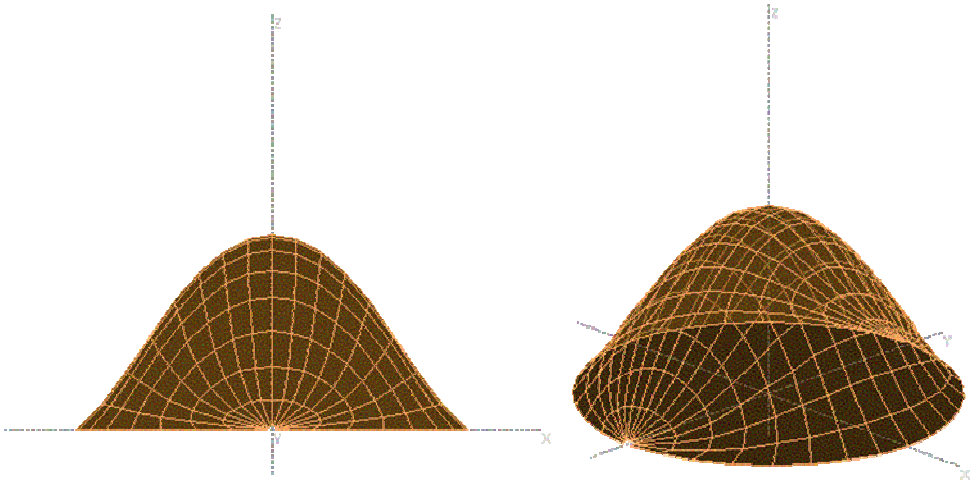
Pomembno je opozoriti še na nepravilnosti deformacijske energije (slika 52) zgoraj in spodaj na robu lupine, kar je posledica definicije deformiranega Bézierovega telesa, saj na vsaki posamezni točki leži 9 kontrolnih točk projektnega telesa. Na tak način sem najenostavneje modeliral okroglo obliko z enim projektnim telesom.



Slika 52: Deformacijska energija (skala ni enaka) – pred (leva slika) in po (desna slika) optimizaciji – lastna teža.

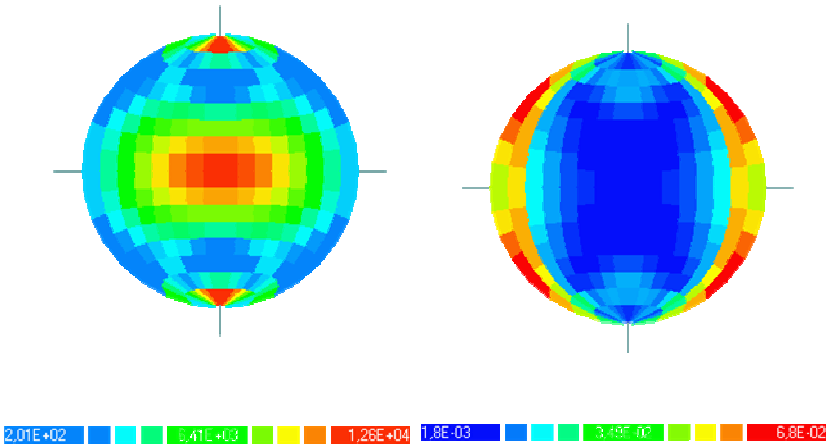
4.4.1.3 Obtežba - sneg

Naloga je poiskati optimalno obliko konstrukcije, obtežene z obtežbo snega 1 kN/m^2, ki deluje na tlorisno površino, kjer bo deformacijska energija konstrukcije minimalna, pri pogojih, da je višina v sredini lupine enaka radiju (10 m) in prostornina konstrukcije omejena na 5 m^3.



Slika 53: Optimalna oblika lupine.

Optimalna oblika lupine je pričakovano podobna prejšnjemu primeru, saj obtežba snega deluje podobno kot lastna teža. Že po 4 iteracijah se je deformacijska energija lupinaste konstrukcije zmanjšala (za 265000 krat) z začetnih 1324180 kNm na 5 kNm.

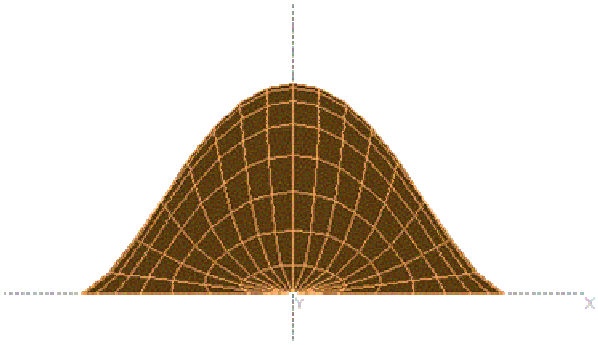


Slika 54: Deformacijska energija (skala ni enaka) pred (leva slika) in po (desna slika) optimizaciji – obtežba snega.

Spet se pojavijo nepravilnosti deformacijske energije (slika 52) zgoraj in spodaj na lupini, kar je posledica definicije deformiranega Bézierovega telesa, saj v posamezni točki leži 9 kontrolnih točk projektnega telesa. Konstrukcija ima tudi po optimizaciji debelino 1 cm.

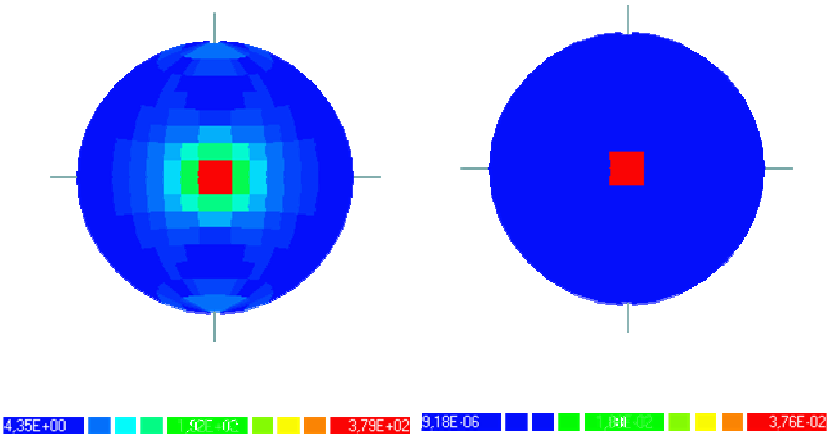
4.4.1.4 Obtežba - koncentrirana sila na sredini plošče

Naloga je poiskati optimalno obliko konstrukcije, obtežene s koncentrirano silo na sredini lupine F = 10 kN, kjer bo deformacijska energija konstrukcije minimalna, pri pogojih, da je višina v sredini konstrukcije enaka radiju (10m) in prostornina konstrukcije omejena na 5 m^3. Optimalna oblika lupine je zelo podobna obrnjeni obliki deformacij zaradi iste obtežbe. Vendar velikih razlik v obliki prejšnjih dveh primerov obtežbe ni. Je pa razlika lepo vidna pri prikazu deformacijske energije po končnih elementih, kjer so maksimalne vrednosti prav na mestu delovanja koncentrirane sile (slika 56).



Slika 55: Optimalna oblika lupine.

Tudi v tem primeru je debelina lupinaste konstrukcije enaka tudi po optimizaciji – 1 cm. Po 7 iteracijah se je deformacijska energija lupinaste konstrukcije zmanjšala (za 42300 krat) z začetnih 8884 kNm na 0.21 kNm.



Slika 56: Deformacijska energija po končnih elementih konstrukcije (skala ni enaka) - pred (leva slika) in po (desna slika) optimizaciji.

4.4.1.5 Komentar rezultatov

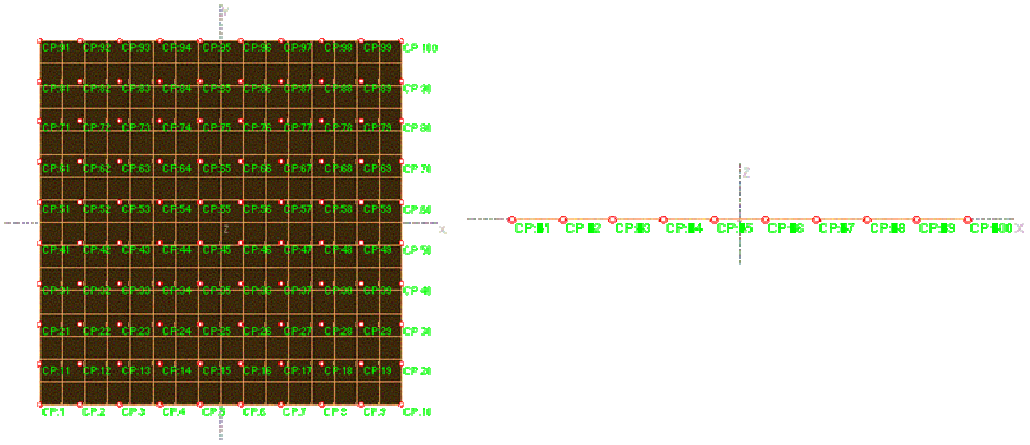
Ker sem uporabil isti model pri vseh obtežnih primerih je najbolj zanimivo dejstvo, da je optimalna oblika pri vseh obtežnih primerih zelo podobna. Razlog za to pa je omejitev prostornine in omejitev pozicije (v z smeri) sredinske točke okrogle lupine. Optimalna oblika se približuje obliki zvona in ne sfere - kar bi marsikdo lahko pričakoval pred postopkom optimizacije. Večje razlike med primeri so vidne le pri prikazu deformacijske energije po končnih elementih konstrukcije. Omeniti je potrebno nesimetričnost deformacijske energije (zgoraj, spodaj) pri začetni obliki, ki je posledica poenostavitve modeliranja okrogle oblike z enim deformiranim štirikotnim Bézierovim telesom.

4.4.2 Štirikotna prostoležeča plošča

4.4.2.1 Opis konstrukcije

Primer sem črpal iz literature (Camprubi, Bischoff in Bletzinger, 2002). Začetna oblika obravnavane lupine (ki je enaka za vse obtežne primere) je prostoležeča jeklena kvadratna

plošča $a = b = 20$ m, kot je prikazano na spodnji sliki (slika 57). Začetna debelina lupine je $h = 5$ cm.

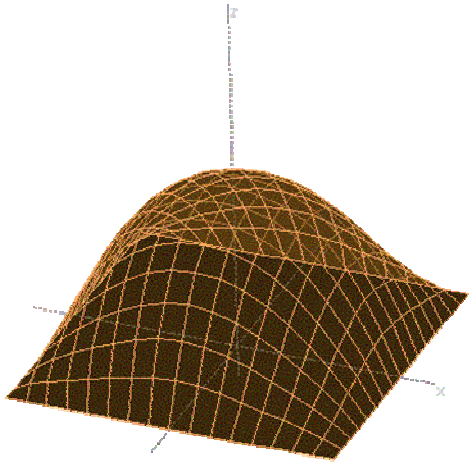


Slika 57: Začetna oblika lupine (levo – tloris, desno – pogled s strani).

Materialne karakteristike uporabljenega jekla so naslednje: elastični modul $E = 21000$ MN/m^3, Poissonov količnik $\nu = 0.3$. Lupino sem modeliral z 256 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom s 100 kontrolnimi točkami (10x10x1). Glede na simetričnost lupinaste konstrukcije in obtežbe sem upošteval dvojno simetrijo (preko x in y osi) pri definiciji projektnih spremenljivk. Za projektne spremenljivke sem izbral 16 višin kontrolnih točk, 32 (16 + 16) spremenljivk, ki opisujejo dopusten premik kontrolnih točk v x in y smeri ter eno spremenljivko za spreminjanje debeline lupine. Spreminjanje projektnih spremenljivk sem omejil – debelino lupine h na interval [1 cm, 10 cm], spremembe pozicij v x in y smeri na interval [-110 cm , 110 cm] in višine kontrolnih točk z na [-20 m, 20 m].

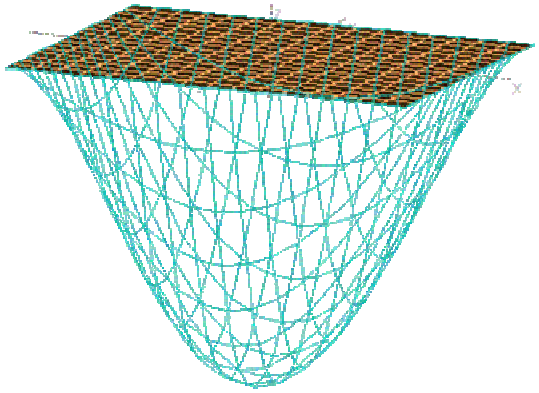
4.4.2.2 Obtežba - lastna teža

Naloga je poiskati optimalno obliko konstrukcije obteženo z lastno težo pri pogojih, da bo deformacijska energija konstrukcije minimalna in višina točke v sredini $a/2$ (10 m).



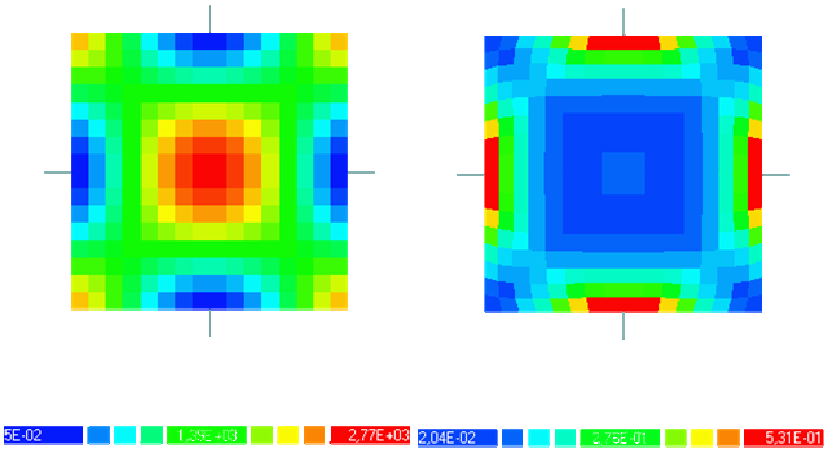
Slika 58: Optimalna oblika – lastna teža.

Optimalna oblika te lupinaste konstrukcije je podobna obrnjeni obliki deformacij zaradi lastne teže (slika 59).



Slika 59: Deformacije začetne oblike lupine zaradi lastne teže.

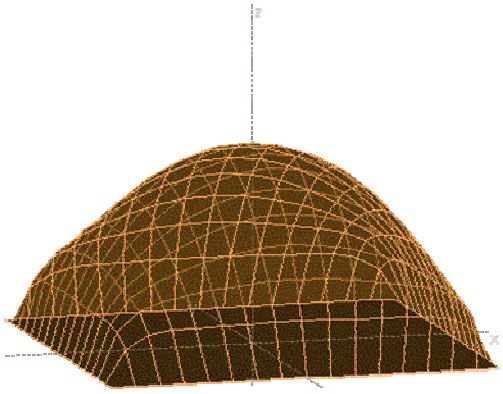
Deformacijska energija konstrukcije se je zmanjšala s 153559018.5 kNm na 42.8 kNm. Prikaz spremembe razporeditve deformacijske energije končnih elementov je na naslednji sliki (slika 60).



Slika 60: Primerjava deformacijske energije (skala ni enaka) pred (leva slika) in po (desna slika) optimizaciji - lastna teža.

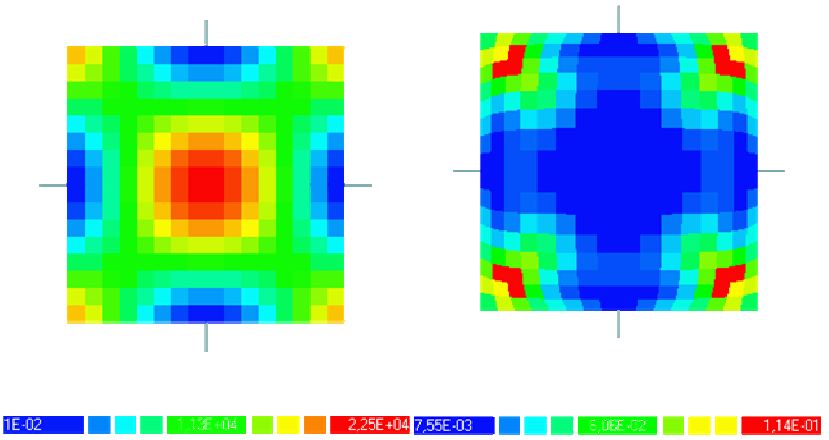
4.4.2.3 Obtežba - sneg

Naloga je poiskati optimalno obliko konstrukcije, obtežene z obtežbo snega 1 kN/m^2 pri pogojih, da bo deformacijska energija konstrukcije minimalna in višina točke v sredini *a*/2 (10 m).



Slika 61: Optimalna oblika lupinaste konstrukcije - obtežba snega.

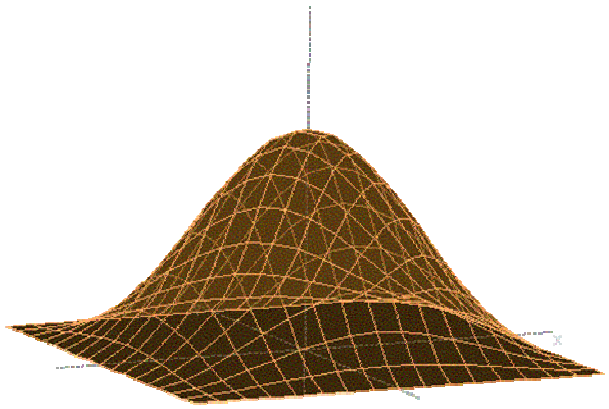
Optimalna oblika je podobna kot v prejšnjem primeru, razlika je le pri podporah, kjer je oblika bolj strma. Debelina lupine *h* se je zmanjšala na 1 cm. Deformacijska energija konstrukcije se je odločno zmanjšala s $3.2 \cdot 10^{13}$ kNm na 8.75 kNm (slika 62).



Slika 62: Primerjava deformacijske energije (skala ni enaka) pred (leva slika) in po (desna slika) optimizaciji – obtežba snega.

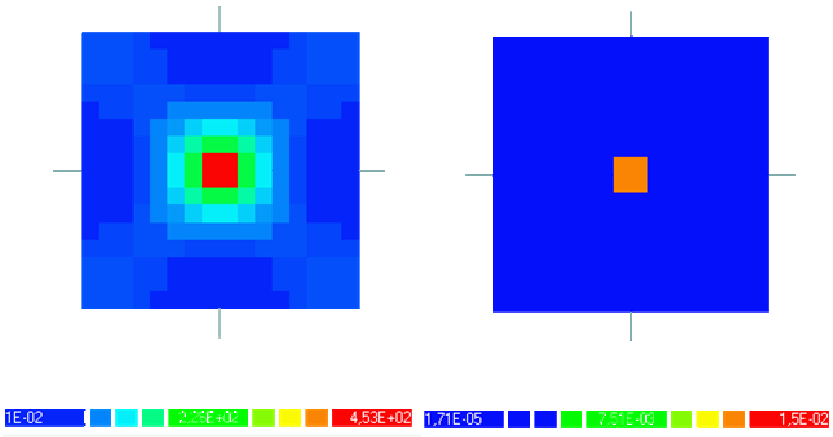
4.4.2.4 Obtežba – koncentrirana točkovna sila v sredini lupine

Naloga je poiskati optimalno obliko konstrukcije, obtežene s točkovno silo v sredini lupine F_z = -10 kN pri pogojih, da bo deformacijska energija konstrukcije minimalna in višina točke v sredini $a/2$ (10 m).



Slika 63: Optimalna oblika konstrukcije - točkovna obtežba.

Na optimalno obliko te optimizacije ima največji vpliv omejitev višine sredinske točke. Pričakoval sem, da bo optimalna oblika podobna kot v primeru okrogle lupine. Debelina lupine h se je zmanjšala na 1 cm. Deformacijska energija konstrukcije se je močno zmanjšala s $4.5 \cdot 10^8$ kNm na 0.10 kNm (Slika 64).



Slika 64: Primerjava deformacijske energije (skala ni enaka) pred (leva slika) in po (desna slika) optimizaciji – točkovna sila.

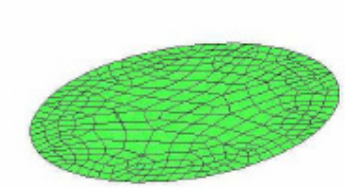
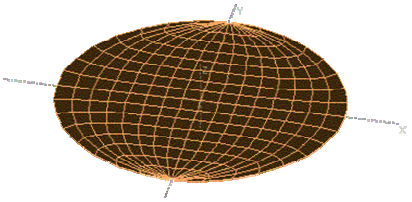
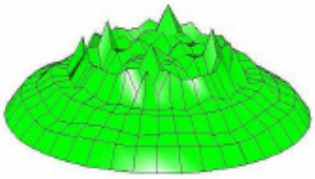
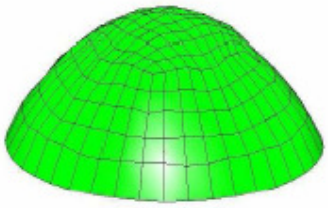
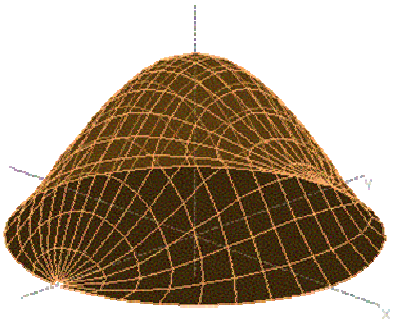
4.4.2.5 Komentar rezultatov

Optimalna oblika pri ploskovnih obtežbah (lastna teža, sneg) se približuje obrnjeni obliki deformacij zaradi obtežbe. Takšno obliko sem pričakoval že pred postopkom optimizacije. Zaradi pogoja višine točke v sredini se optimalna oblika približuje obliki zvona (deformirane zaradi same oblike (okrogla, kvadratna) lupinaste konstrukcije).

Zanimiva je primerjava z rezultati iz literature (Daud, Camprubi in Bletzinger, 2004), kjer so z drugačnim pristopom (KE-pristop) prišli do podobnih rezultatov. Ker sta v članku (Daud, Camprubi in Bletzinger, 2004) prikazana primera okrogle lupine, obtežene z lastno težo in kvadratne lupine, obtežene s koncentrirano silo v sredini, lahko prikažem samo primerjavo teh dveh primerov.

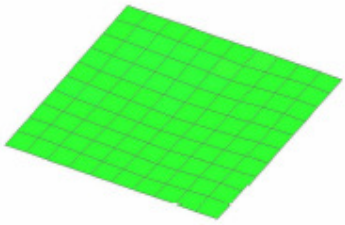
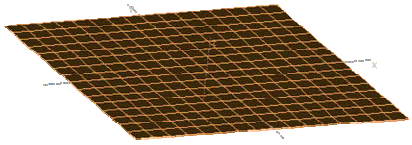
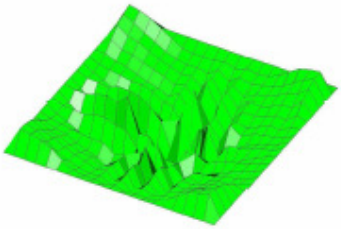
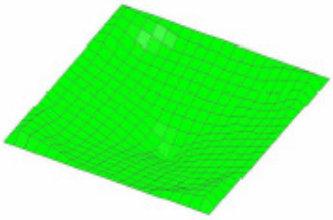
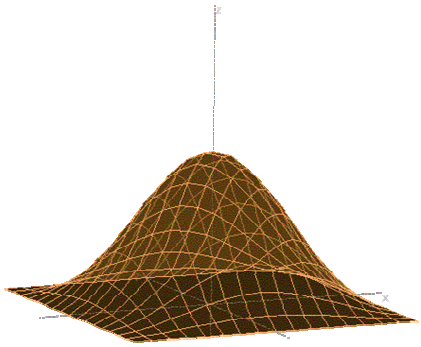
Glede na podobnost omejitev pri obeh modelih uporaba filtra pri primeru iz članka (Daud, Camprubi in Bletzinger, 2004) (omejen glavni val z radijem) in omejitve višine točke na sredini lupine v primeru modela programa EKON (višina te točke enaka radiju) je tudi optimalna oblika za okroglo lupinasto konstrukcijo zelo podobna. Po obeh pristopih je bila uporabljena namenska funkcija minimuma deformacijske energije.

Preglednica 11: Primerjava rezultatov različnih pristopov na okrogli lupini.

	Daud, Camprubi in Bletzinger, 2004	Model programa EKON
Začetni model konstrukcije		
Optimalna oblika konstrukcije	 Pred uporabo filtra  Po uporabi filtra	

Pri primeru štirikotne lupine, obtežene s koncentrirano silo v sredini, je zanimivo, da sta rezultata – optimalni obliki različni. Očitna razlika med optimalnima oblikama obeh modelov je v obliki loka (konkavna ali konveksna oblika), enkrat so membranske sile natezne, drugič tlačne. Vidna pa je tudi razlika med modeloma. Pri metodi projektnega telesa je oblika konstrukcije bolj gladka, ni večjih odstopanj – kar pa je možno pri KE-pristopu. Res pa je, da s čistimi koncentriranimi silami v gradbeništvu nimamo opravka, saj se obtežba vedno razporedi na neko ploskev. (tudi material ne bi prenesel velikih lokalnih napetosti). Glede na primere fizičnih modelov, bi pričakoval optimalno obliko kot obrnjeno obliko deformacij zaradi obtežbe, kar je bilo doseženo z modelom programa EKON.

Preglednica 12: Primerjava rezultatov različnih pristopov na štirikotni lupini.

	Daud, Camprubi in Bletzinger, 2004	Model programa EKON
Začetni model konstrukcije		
Optimalna oblika konstrukcije	 Pred uporabo filtra  Po uporabi filtra	

5 NELINEARNI PRIMERI

Lupinaste konstrukcije so zelo občutljive na uklon in že z zelo majhnimi spremembami oblike lahko precej izboljšamo ali poslabšamo uklonsko nosilnost. V nadaljevanju je prikazanih nekaj primerov optimizacije oblike ploskovnih konstrukcij z upoštevanjem geometrijske nelinearnosti in primerjava rezultatov optimizacije z geometrijsko linearnim modelom.

5.1 AB streha 1

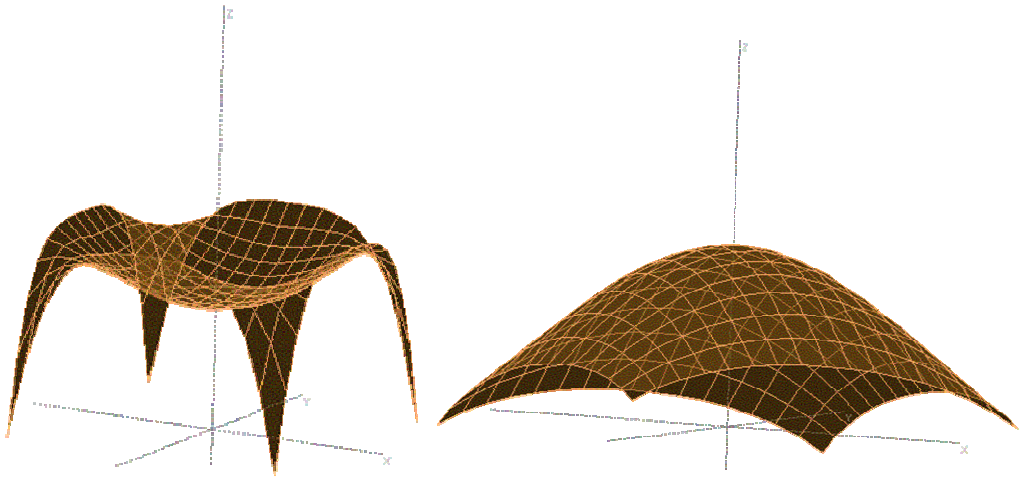
Model lupinaste konstrukcije je opisan v poglavju 4.2 (stran 31). Model je povsem enak, razlika je le v upoštevanju geometrijske nelinearnosti – izvede se nelinearna analiza.
Začetna oblika obravnavane lupine je kvadratne tlorisne oblike 20x20 m, vrtljivo podprte v vogalih. Začetna oblika lupine je (ravna) plošča v ravnini *xy*.
Materialne karakteristike so naslednje: elastični modul E = 30000 MN/m^3, Poissonov količnik ν = 0.3. Lupina je obtežena s težo snega 5000 kN/m^2 na tlorisno površino. Lupino sem modeliral z 256 lupinastimi končnimi elementi in z enim projektnim elementom - Bézierovim telesom s 25 kontrolnimi točkami (5x5x1). Za projektne spremenljivke sem izbral 21 višin kontrolnih točk, 20 spremenljivk, ki opisujejo pozicijo kontrolnih točk (v *x* in *y* smeri) ter 25 spremenljivk, ki opisujejo gladko spreminjanje debeline lupine. Spreminjanje projektnih spremenljivk sem omejil – debelino lupine na interval [7.5 cm, 50 cm], spremembo pozicije (*x* in *y* koordinate) kontrolnih točk na [-5 m, 5 m] ter višine kontrolnih točk na [0 m, 40 m].

5.1.1 Primer a

Naloga je poiskati optimalno obliko konstrukcije z upoštevanjem geometrijske nelinearnosti, kjer bo deformacijska energija konstrukcije minimalna pri pogojih, da je prostornina konstrukcije 60 m^3 in višina točke na sredini več kot 8 m.

Pri postopku optimizacije s programom iGO sem dobil dve obliki. Leva (slika 65) je rezultat optimizacije brez dodatnih posegov, po 10-ih iteracijah ni bilo konvergence tako deformacijske energije kot prostornine lupine. Rezultat pa je vseeno zanimiv, saj je zadoščeno vsem omejitvenim pogojem. Deformacijska energija se je zmanjšala z 42073 Nm na 8971

Nm, prostornina se je z začetnih 200 m^3 zmanjšala na 57.7 m^3, višina točke na sredini se je z začetne vrednosti 0 m povečala na 8.39 m.

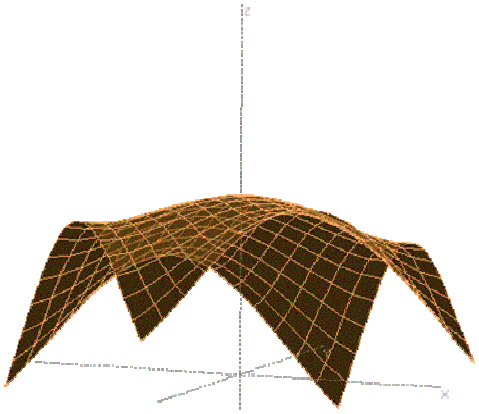


Slika 65: Optimalni obliki lupinaste konstrukcije nelinearne analize.

Pri optimizaciji desne oblike (slika 65) sem omejil velikost sprememb projektnih spremenljivk zaradi namenske funkcije (deformacijske energije) in dobil bolj pričakovano obliko (po 16-ih iteracijah). Deformacijska energija konstrukcije se je zmanjšala z začetnih 42073 Nm na 1204 Nm, prostornina konstrukcije se je zmanjšala na zahtevanih 60 m^3, višina točke na sredini pa se je povečala z začetne vrednosti 0 m na 8.43 m.

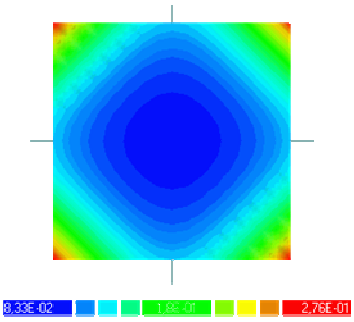
5.1.2 Primer b

Naloga je poiskati optimalno obliko konstrukcije z upoštevanjem geometrijske nelinearnosti, kjer bo deformacijska energija konstrukcije minimalna pri pogojih, da je prostornina konstrukcije 60 m^3, višina točke na sredini več kot 8 m in višina točk na robu (x = 0 m, y = 0 m) 8 m.



Slika 66: Optimalna oblika konstrukcije – nelinearna analiza (primer b).

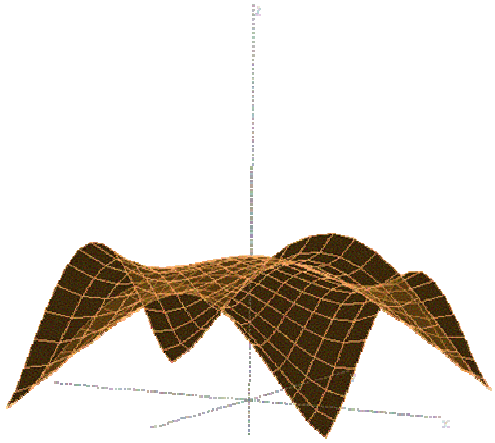
Optimalna oblika konstrukcije je zaradi omejitvenih pogojev precej podobna optimalni obliki linearne analize. Spremembe so opazne pri višini sredinske točke in debelinah končnih lupinastih elementov (ki so manjše), ter manjše deformacijske energije konstrukcije. Višina sredinske točke se je povečala z začetnih 0 m na 9.98 m, višine ostalih točk na robu pa so se spremenile na zahtevanih 8 m, deformacijska energija se je zmanjšala na 1506 Nm, debelina lupine pa se zvezno spreminja s 27.5 cm v kotih na 7.5 cm v sredini lupine.



Slika 67: Spreminjanje debeline končnih elementov - nelinearna analiza (primer b).

5.1.3 Primer c

Naloga je poiskati optimalno obliko konstrukcije z upoštevanjem geometrijske nelinearnosti, kjer bo deformacijska energija konstrukcije minimalna pri pogojih, da je prostornina konstrukcije 60 m^3, višina točke na sredini med 8 m in 9 m, ter višina točk na robu 8 m.



Slika 68: Optimalna oblika konstrukcije - nelinearna analiza (primer c).

Razlike med optimalnima oblikama linearne in nelinearne analize so opazne v višini sredinske točke, spreminjajoče se debeline lupinastih elementov in nekoliko večje deformacijske energije konstrukcije. Višina sredinske točke se je z začetnih 0 m povečala na zahtevanih 8 m, prav tako imajo ostale točke zahtevano višino 8 m. Debelina se je zmanjšala z začetnih 50 cm na zvezno spreminjajočo od 33.6 cm v podporah do 8 cm v sredini lupine. Deformacijska energija konstrukcije se je zmanjšala s 42073 Nm na 808 Nm.

5.1.4 Primerjava rezultatov

Upoštevanje geometrijske nelinearnosti (nelinearna analiza) v tem optimizacijskem problemu (več omejitvenih pogojev – višine robnih točk, omejena prostornina) ni tako opazno. Večjih sprememb oblike ni bilo, razlike so se pojavile pri višinah sredinske točke ter zveznem spreminjanju debeline.

Preglednica 13: Primerjava analize.

	Primer a		Primer b		Primer c	
Analiza	linearna	nelinearna	linearna	nelinearna	linearna	nelinearna
Začetna oblika	slika 26		slika 26		slika 26	
Optimalna oblika	slika 27	slika 65	slika 28	slika 66	slika 30	slika 68
Zač. def. energija	42073 Nm					
Opt. def. energija	540 Nm	1204 Nm	2104 Nm	1506 Nm	580 Nm	808 Nm
Opt. višina sredinske točke	7.3 m	8.43 m	8.8 m	9.98 m	8.6 m	8.0 m
Začetna debelina	50 cm					
Sprememba debeline (od - do)	7.5 cm 25 cm	8 cm 11 cm	7.5 cm 35 cm	7.5 cm 27.5 cm	7.5 cm 35 cm	8 cm 33.6 cm

6 ZAKLJUČEK

Obstaja kar nekaj principov iskanja optimalne oblike ploskovnih konstrukcij, ki se razlikujejo glede na vrsto uporabljenih končnih elementov, njihove diskretizacije, izbiro parametrov v optimizacijskem postopku in izbiro zunanjih dejavnikov. V nalogi je predstavljen iterativni postopek iskanja optimalne oblike ploskovnih konstrukcij po principu projektnega telesa.

V raziskovalnem računalniškem programu EKON je uporabljen princip projektnega telesa, ki zahteva nekoliko drugačen način priprave vhodnih podatkov. Za projektno telo je bilo izbrano Bézierovo telo, pri katerem je parametrizacija precej enostavno določljiva. Konstrukcija je tako predstavljena kot tridimenzionalno telo, ki je (lahko) sestavljeno iz več Bézierovih teles.

V poglavju 3 sta predstavljena računalniška programa EKON in iGO, ki ju razvijajo na Fakulteti za strojništvo Univerze v Mariboru. EKON je program za analizo konstrukcij po metodi končnih elementov, v katerem so vključeni nekateri standardni KE. Poleg analize odziva konstrukcije pa EKON omogoča tudi analizo občutljivosti. iGO je samostojen program za interaktivno gradientno optimizacijo, ki programe za analizo (kot je EKON) uporablja kot zunanje simulatorje. V samem interaktivnem procesu iGO zaganja tak program in njegove rezultate uporabi kot izboljšavo v algoritmu gradientne optimizacije.

V poglavju 4 je predstavljenih več primerov optimizacije oblike lupinastih konstrukcij, kjer je za namensko funkcijo uporabljen minimum deformacijske energije. Glede na rezultate večih različnih modelov iste konstrukcije lahko sklepamo, da je oblika zelo odvisna od začetne oblike, omejitvenih pogojev in obtežbe. V poglavju 5 je predstavljenih nekaj primerov, kjer je bila upoštevana geometrijska nelinearnost ter primerjava rezultatov z geometrijsko linearnimi primeri.

Očitno je, da z uporabo namenske funkcije minimuma deformacijske energije konstrukcije stremijo k membranskem napetostnem stanju, kar je lepo razvidno v primeru, izračunanim tudi s programom SAP2000. Lupine so veliko bolj primerne za prenašanje vertikalnih obremenitev kot plošče, saj je že z minimalno dvojno ukrivljenostjo dosežena veliko manjša debelina konstrukcije. Takšne konstrukcije so tudi estetsko bolj zanimive.

VIRI

Ansola, R., Canales, J., Tarrago, J.A. in Rasmussen, J., 2004, 'Combined shape and reinforcement layout optimization of shell structures', Structural and Multidisciplinary Optimization,Vol. 27, No. 4, 219-227.

Belblidia, F., Bulman, S., 2002, 'A hybrid topology optimization algorithm for static and vibrating shell structures', International Journal for Numerical Methods in Engineering, Vol. 54, No.6, str. 835-852.

Camprubi, N., Bischoff, M. in Bletzinger, K.-U., 2002, 'Shape optimization of shell structures', Technische Univerzität München, Statusseminar.

Daud, F., Camprubi N. in Bletzinger, K.-U., 2004, 'Filtering and regularization techniques in shape optimization with CAD-free parametrization', Proceedings of NATO, str. 441-446, UL -FGG.

Gates, A.A. in Accorsi M.L., 1993, 'Automatic Shape Optimization of Three-Dimensional Shell Structures with Large Shape Changes', Computers & Structures 49, str. 167-178.

Imam, H., 1998, 'Shape optimization of umbrella-shaped concrete shells subjected to self-weight as the dominant load', Computers & Structures, vol.69, str. 513-524.

Jaklič, G., 1999, 'Bézierove krivulje in ploskve', diplomsko delo, Univerza v Ljubljani, FMF.

Kegl, M. in Oblak, M., 1997, 'Optimization of mechanical systems: On non-linear first-order approximation with an additive convex term', Communications in Numerical Methods in Engineering, 13, str. 13-20.

Kegl, M., Butinar, B. J. in Kegl, B., 2002, 'An efficient gradient-based optimization algorithm for mechanical systems', Communications in Numerical Methods in Engineering, vol. 18, str. 363-371.

Kegl, M., Brank, B. in Jaklič, M., 2004, 'Optimiranje oblike lupinastih konstrukcij', Kuhljevi dnevi , str. 149-156.

Kegl, M., 2005, 'Parameterization based shape optimization : theory and practical implementation aspects', Eng. comput., vol. 22, no. 5/6, str. 646-663.

Kegl, M. in Brank, B., 2006, 'Shape optimization of truss-stiffened shell structures with variable thickness', Comput. methods appl. mech. eng.. [Print ed.], vol. 195, iss. 19/22, str. 2611-2634.

Lagaros, N.D., Fragiadakis M.M. in Papadrakakis, M., 2002, Greece, 'Automatic structural optimization of stiffened shells', 4th GRACM Congress on Computational Mechanics.

Maute, K. in Ramm, E., 1995, 'Adaptive topology optimization', Structural Optimization, Vol. 10, str.100-112.

Ohmori H. in Yamamoto, K., 1998, 'Shape optimization of shell and spatial structure for specified stress distribution', Memoirs of the School of Engineering, Nagoya University, Vol. 50, No.1.

Ramm, E., Kemmler R., in Schwarz, S., 2000, Greece, 'Formfinding and Optimization of Shell Structures', Fourth International Colloquium on Computation of Shell & Spatial Structures.

Reitinger, R. in Ramm, E., 1995, 'Buckling and Imperfection Sensitivity in the Optimization of Shell Structures', Thin-Walled Structures 23, str. 159-177.

PRILOGE

Priloga A: Dodana programska koda

V programsko kodo programa EKON sem dodal spremembe, da sem lahko uporabljal do 10x10x10 kontrolnih točk na posameznem projektnem telesu. Ker je program EKON programiran v Fortranu, je programska koda v Fortran obliki. Tukaj so navedene samo procedure (BernPol, BernPolC in BernPolCC) v modulu mkQBB, ki so bile spremenjene.

1. Procedura za izračun Bernsteinovih baznih polinomov

```
!=====================================================================
pure subroutine BernPol (N,X,B)
  implicit none
  integer, intent(in) :: N
  doubleprecision, intent(in) :: X
  doubleprecision, intent(out) :: B(:)

!_Execute
  select case(N)
    case(1)
      B(1) = 1
    case(2)
      B(1) = 1 - X; B(2) = X
    case(3)
      B(1) = (1-X)**2; B(2) = 2*(1-X)*X; B(3) = X**2
    case(4)
      B(1) = (1-X)**3; B(2) = 3*(1-X)**2*X; B(3) = 3*(1-X)*X**2; B(4) = X**3
    case(5)
      B(1) = (1-X)**4; B(2) = 4*(1-X)**3*X; B(3) = 6*(1-X)**2*X**2; B(4) = 4*(1-
X)*X**3; B(5) = X**4
    case(6)
      B(1) = (1-X)**5; B(2) = 5*(1-X)**4*X; B(3) = 10*(1-X)**3*X**2; B(4) = 10*(1-
X)**2*X**3; B(5) = 5*(1-X)*X**4; B(6) = X**5
    case(7)
      B(1) = (1-X)**6; B(2) = 6*(1-X)**5*X; B(3) = 15*(1-X)**4*X**2; B(4) = 20*(1-
X)**3*X**3; B(5) = 15*(1-X)**2*X**4;B(6) = 6*(1-X)*X**5; B(7) = X**6
    case(8)
      B(1) = (1-X)**7; B(2) = 7*(1-X)**6*X; B(3) = 21*(1-X)**5*X**2; B(4) = 35*(1-
X)**4*X**3; B(5) = 35*(1-X)**3*X**4
      B(6) = 21*(1-X)**2*X**5; B(7) = 7*(1-X)*X**6; B(8) = X**7
    case(9)
      B(1) = (1-X)**8; B(2) = 8*(1-X)**7*X; B(3) = 28*(1-X)**6*X**2; B(4) = 56*(1-
X)**5*X**3; B(5) = 70*(1-X)**4*X**4; B(6) = 56*(1-X)**3*X**5; B(7) = 28*(1-
X)**2*X**6; B(8) = 8*(1-X)*X**7; B(9) = X**8
    case(10)
      B(1) = (1-X)**9; B(2) = 9*(1-X)**8*X; B(3) = 36*(1-X)**7*X**2; B(4) = 84*(1-
X)**6*X**3; B(5) = 126*(1-X)**5*X**4; B(6) = 126*(1-X)**4*X**5; B(7) = 84*(1-
X)**3*X**6; B(8) = 36*(1-X)**2*X**7;
B(9) = 9*(1-X)*X**8; B(10) = X**9
  end select

!_End
end subroutine
```

2. Procedura za izračun prvega odvoda Bensteinovih baznih polinomov

```
!=====================================================================
pure subroutine BernPolC (N,X,BC)
  implicit none
  integer, intent(in) :: N
  doubleprecision, intent(in) :: X
  doubleprecision, intent(out) :: BC(:)

!_Execute
  select case(N)
    case(1)
      BC(1) = 0
    case(2)
      BC(1) = -1; BC(2) = 1
    case(3)
      BC(1) = -2*(1-X); BC(2) = 2*(1-X) - 2*X; BC(3) = 2*X
    case(4)
      BC(1) = -3*(1-X)**2; BC(2) = 3*(1-X)**2 - 6*(1-X)*X; BC(3) = 6*(1-
X)*X - 3*X**2; BC(4) = 3*X**2
    case(5)
      BC(1) = -4*(1-X)**3; BC(2) = 4*(1-X)**3 - 12*(1-X)**2*X; BC(3) =
12*(1-X)**2*X - 12*(1-X)*X**2; BC(4) = 12*(1-X)*X**2 - 4*X**3
BC(5) = 4*X**3
    case(6)
      BC(1) = -5*(1-X)**4; BC(2) = 5*(1-X)**4 - 20*(1-X)**3*X; BC(3) =
20*(1-X)**3*X - 30*(1-X)**2*X**2; BC(4) = 30*(1-X)**2*X**2 - 20*(1-X)*X**3;
BC(5) = 20*(1-X)*X**3 - 5*X**4; BC(6) = 5*X**4
    case(7)
      BC(1) = -6*(1-X)**5; BC(2) = 6*(1-X)**5 - 30*(1-X)**4*X; BC(3) =
30*(1-X)**4*X - 60*(1-X)**3*X**2; BC(4) = 60*(1-X)**3*X**2 - 60*(1-
X)**2*X**3; BC(5) = 60*(1-X)**2*X**3 - 30*(1-X)*X**4; BC(6) = 30*(1-X)*X**4
- 6*X**5; BC(7) = 6*X**5
    case(8)
      BC(1) = -7*(1-X)**6; BC(2) = 7*(1-X)**6 - 42*(1-X)**5*X; BC(3) =
42*(1-X)**5*X - 105*(1-X)**4*X**2; BC(4) = 105*(1-X)**4*X**2 - 140*(1-
X)**3*X**3; BC(5) = 140*(1-X)**3*X**3 - 105*(1-X)**2*X**4; BC(6) = 105*(1-
X)**2*X**4 - 42*(1-X)*X**5; BC(7) = 42*(1-X)*X**5 - 7*X**6; BC(8) = 7*X**6
    case(9)
      BC(1) = -8*(1-X)**7; BC(2) = 8*(1-X)**7 - 56*(1-X)**6*X; BC(3) =
56*(1-X)**6*X - 168*(1-X)**5*X**2; BC(4) = 168*(1-X)**5*X**2 - 280*(1-
X)**4*X**3; BC(5) = 280*(1-X)**4*X**3 - 280*(1-X)**3*X**4; BC(6) = 280*(1-
X)**3*X**4 - 168*(1-X)**2*X**5; BC(7) = 168*(1-X)**2*X**5 - 56*(1-X)*X**6;
BC(8) = 56*(1-X)*X**6 - 8*X**7; BC(9) = 8*X**7
    case(10)
      BC(1) = -9*(1-X)**8; BC(2) = 9*(1-X)**8 - 72*(1-X)**7*X; BC(3) =
72*(1-X)**7*X - 252*(1-X)**6*X**2; BC(4) = 252*(1-X)**6*X**2 - 504*(1-
X)**5*X**3; BC(5) = 504*(1-X)**5*X**3 - 630*(1-X)**4*X**4; BC(6) = 630*(1-
X)**4*X**4 - 504*(1-X)**3*X**5; BC(7) = 504*(1-X)**3*X**5 - 252*(1-
X)**2*X**6 ;BC(8) = 252*(1-X)**2*X**6 - 72*(1-X)*X**7; BC(9) = 72*(1-
X)*X**7 - 9*X**8; BC(10) = 9*X**8
  end select

!_End
end subroutine
```

3. Procedura za izračun drugega odvoda Bensteinovih baznih polinomov

```
!=====================================================================
pure subroutine BernPolCC (N,X,BCC)
  implicit none
  integer, intent(in) :: N
  doubleprecision, intent(in) :: X
  doubleprecision, intent(out) :: BCC(:)

!_Execute
  select case(N)
    case(1)
      BCC(1) = 0
    case(2)
      BCC(1) = 0; BCC(2) = 0
    case(3)
      BCC(1) = 2; BCC(2) = -4; BCC(3) = 2
    case(4)
      BCC(1) = 6*(1-X); BCC(2) = -12*(1-X) + 6*X; BCC(3) = 6*(1-X) - 12*X;
BCC(4) = 6*X
    case(5)
      BCC(1) = 12*(1-X)**2; BCC(2) = -24*(1-X)**2 + 24*(1-X)*X; BCC(3) =
12*(1-X)**2 - 48*(1-X)*X + 12*X**2 ; BCC(4) = 24*(1-X)*X - 24*X**2; BCC(5)
= 12*X**2
    case(6)
      BCC(1) = 20*(1-X)**3; BCC(2) = -40*(1-X)**3 + 60*(1-X)**2*X; BCC(3) =
20*(1-X)**3 - 120*(1-X)**2*X + 60*(1-X)*X**2; BCC(4) = 60*(1-X)**2*X -
120*(1-X)*X**2 + 20*X**3; BCC(5) = 60*(1-X)*X**2 - 40*X**3; BCC(6) =
20*X**3
    case(7)
      BCC(1) = 30*(1-X)**4; BCC(2) = -60*(1-X)**4 + 120*(1-X)**3*X; BCC(3)
= 30*(1-X)**4 - 240*(1-X)**3*X + 180*(1-X)**2*X**2; BCC(4) = 120*(1-X)**3*X
- 360*(1-X)**2*X**2 + 120*(1-X)*X**3; BCC(5) = 180*(1-X)**2*X**2 - 240*(1-
X)*X**3 + 30*X**4; BCC(6) = 120*(1-X)*X**3 - 60*X**4; BCC(7) = 30*X**4
    case(8)
      BCC(1) = 42*(1-X)**5; BCC(2) = -84*(1-X)**5 + 210*(1-X)**4*X; BCC(3)
= 42*(1-X)**5 - 420*(1-X)**4*X + 420*(1-X)**3*X**2; BCC(4) = 210*(1-X)**4*X
- 840*(1-X)**3*X**2 + 420*(1-X)**2*X**3; BCC(5) = 420*(1-X)**3*X**2 -
840*(1-X)**2*X**3 + 210*(1-X)*X**4; BCC(6) = 420*(1-X)**2*X**3 - 420*(1-
X)*X**4 + 42*X**5; BCC(7) = 210*(1-X)*X**4 - 84*X**5; BCC(8) = 42*X**5
    case(9)
      BCC(1) = 56*(1-X)**6; BCC(2) = -112*(1-X)**6 + 336*(1-X)**5*X; BCC(3)
= 56*(1-X)**6 - 672*(1-X)**5*X + 840*(1-X)**4*X**2; BCC(4) = 336*(1-X)**5*X
- 1680*(1-X)**4*X**2 + 1120*(1-X)**3*X**3; BCC(5) = 840*(1-X)**4*X**2 -
2240*(1-X)**3*X**3 + 840*(1-X)**2*X**4; BCC(6) = 1120*(1-X)**3*X**3 -
1680*(1-X)**2*X**4 + 336*(1-X)*X**5; BCC(7) = 840*(1-X)**2*X**4 - 672*(1-
X)*X**5 + 56*X**6; BCC(8) = 336*(1-X)*X**5 - 112*X**6; BCC(9) = 56*X**6
    case(10)
      BCC(1) = 72*(1-X)**7; BCC(2) = -144*(1-X)**7 + 504*(1-X)**6*X; BCC(3)
= 72*(1-X)**7 - 1008*(1-X)**6*X + 1512*(1-X)**5*X**2; BCC(4) = 504*(1-
X)**6*X - 3024*(1-X)**5*X**2 + 2520*(1-X)**4*X**3; BCC(5) = 1512*(1-
X)**5*X**2 - 5040*(1-X)**4*X**3 + 2520*(1-X)**3*X**4; BCC(6) = 2520*(1-
X)**4*X**3 - 5040*(1-X)**3*X**4 + 1512*(1-X)**2*X**5; BCC(7) = 2520*(1-
X)**3*X**4 - 3024*(1-X)**2*X**5 + 540*(1-X)*X**6; BCC(8) = 1512*(1-
X)**2*X**5 - 1008*(1-X)*X**6 + 72*X**7; BCC(9) = 504*(1-X)*X**6 - 144*X**7;
BCC(10) = 72*X**7
  end select
```

Priloga B: Pomoč v programu EKON

V sklopu diplomske naloge sem napisal opis vhodnih podatkov in pomene posameznih postavk (»Data objects«), kar je kot pomoč (»Help«) bilo vključeno v program EKON.

1. Design variables

A **design variable** represents a quantity that can be varied by third-party software for optimization purposes. In that case the design variables have to be used in the definition of the **parameters**. The parameters, in turn, are employed in the definition of control point positions, thicknesses of finite elements and so on. By this arrangement the shape of the structure (in the most general sense) can be made dependent on the design variables. Consequently the structure can be optimized by suitable optimization programs like **iGO**.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Value** - the current value of the design variable. Note that *this value will be changed* when running an optimization task.
- **DerivateEps** - the epsilon used when computing numerically the derivatives of the **parameters** with respect to the design variables. If the order of magnitude of the design variable is around *1*, the default value *2**(-12)* is recommended.

Note. In the **term** expression of a **parameter**, the design variable is referenced to by a letter **b**, followed by the index of the design variable. For example, the expression *200 + 10*b3 + b5**2* is a valid definition of a **parameter** (that depends on the design variables 3 and 5).

Note. It is highly recommended to define the **parameters** in such a way, that the order of magnitude of all design variables is around *1*. For example, for a parameter that will vary in the interval from *100* to *140* in dependence of the design variable *1*, it is a good choice to define it by the term *120 + 20*b1*. Here we assumed that *b1* will be varied within the interval *[-1, 1]*.

2. Parameters

A **parameter** represents a scalar-valued quantity. It may be a constant or a function depending on design variables. Parameters are used to define the design dependent quantities and to simplify the data preparation.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.

- **Term** - the mathematical definition of the parameter. May be any valid FORTRAN expression containing the arithmetic operators (+, -, /, *, **), arithmetic intrinsic FORTRAN functions (*sin()*, *cos()*, ...), optionally nested parenthesis, numeric constants in standard FORTRAN notation as well as the symbols: *Pi* and *b1, bn*, representing the design variables.

Note. In order to define any quantity (for example, the X coordinate of a node) by a parameter, you simply input the character **#** followed by the **index** of the corresponding parameter. For example, *#3* would mean that the value of the quantity is defined by the parameter 3. It is also possible to input *-#3*, which means that the value of the quantity equals the **negative** value of parameter 3.

Note. The **Term** may also contain **references** to other parameters as follows:

- **Backward referencing** - to any **previously** defined parameters - is always legal. For example, the expression *100 + #4* is always a valid definition of the parameter number 5 or higher.
- **Forward referencing** - to any **forthcoming** parameters - is legal **only** under the condition that the referenced parameters themselves do not contain any further parameter references. For example, the expression *100 + #4* is a valid definition of the parameter number 3 or lower only if the parameter 4 itself does not references other parameters.

3. Control points

A **control point** represents a geometrical point defined in the real 3D space. Control points are used in the definition of Bézier bodies serving as the design elements. A control point may also hold a scalar quantity that is used to compute a scalar field defined within the design element.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **X** - the X coordinate (constant or parameter) of the control point.
- **Y** - the Y coordinate (constant or parameter) of the control point.
- **Z** - the Z coordinate (constant or parameter) of the control point.
- **Weight** - the weight (constant or parameter) of the control point.
- **Value** - the scalar quantity (constant or parameter) of the control point.

4. Quadrangular Bézier bodies

A **quadrangular Bézier body** is a geometrical object that can be used as the design element.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **NumCP1** - the number of control points in direction 1.
- **NumCP2** - the number of control points in direction 2.
- **NumCP3** - the number of control points in direction 3.

- **Rational** - a switch indicating that the Bézier body is of a rational type. The Bézier body is of a rational type if the weight of any control point differs from 1.
- **ControlPoints** - the record of indexes of control points from the control point list.

Note. The sequence of the specified control points is important. It determines the shape of the quadrangular Bézier body. The indexes of the control points should be separated by space (blank) characters.

Note. The number of control points of the body is defined by NumCP1 x NumCP2 x NumCP3. The maximum number of control points in each direction is limited to 5. Therefore, the quadrangular Bézier body can be determined by a minimum of 1 (1x1x1) and a maximum of 125 (5x5x5) control points. In the former case the body is degenerated to a single point.

5. Triangular Bézier bodies

A **triangular Bézier body** is a geometrical object that can be used as the design element.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **NumCP1** - the number of control points on the triangular plane.
- **NumCP2** - the number of planes.
- **Rational** - a switch indicating that the Bézier body is of a rational type. The Bézier body is of a rational type if the weight of any control point differs from 1.
- **ControlPoints** - the record of indexes of control points from the control point list.

Note. The sequence of the specified control points is important. It determines the shape of the triangular Bézier body. The indexes of the control points should be separated by space (blank) characters.

Note. The number of control points of the body is defined by NumCP1 x NumCP2. The maximum number of control points 15x5. Therefore, the triangular Bézier body can be determined by a minimum of 1 (1x1) and a maximum of 75 (15x5) control points. In the former case the body is degenerated to a single point.

6. Design elements

A **design element** is a geometrical object that may serve as the 'finite element mesh carrier'. This is achieved by defining the positions of the finite element nodes with respect to the design element.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **BodyCode** - the code of the geometrical body:
  - QBB - quadrangular Bézier body.
  - TBB - triangular Bézier body.
- **BodyInx** - the index of the geometrical body.

7. General forces

A **general force** represents a point load (force or moment) that can be applied to a finite element node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Fx** - X component (constant or parameter) of the nodal force.
- **Fy** - Y component (constant or parameter) of the nodal force.
- **Fz** - Z component (constant or parameter) of the nodal force.
- **Mx** - X component (constant or parameter) of the nodal moment.
- **My** - Y component (constant or parameter) of the nodal moment.
- **Mz** - Z component (constant or parameter) of the nodal moment.

8. General pressures

A **general pressure** represents a pressure load (such as snow, wind, hydrostatic pressure) that can be applied to some finite elements. The magnitude of the pressure is calculated as $p = C + Cx*X + Cy*Y + Cz*Z$. In this term X, Y and Z are the coordinates of the point in the real 3D space where the pressure is evaluated.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **DirectionCode** - the code that defines the direction of pressure load:
  - X - the direction of pressure is in the direction of X axis.
  - Y - the direction of pressure is in the direction of Y axis.
  - Z - the direction of pressure is in the direction of Z axis.
  - XN - the direction of pressure is in the direction of X axis but the pressure is multiplied by *cos(alpha)*, where *alpha* is the angle between the X-axis and the normal of the finite element.
  - YN - the direction of pressure is in the direction of Y axis but the pressure is multiplied by *cos(alpha)*, where *alpha* is the angle between the Y-axis and the normal of the finite element.
  - ZN - the direction of pressure is in the direction of Z axis but the pressure is multiplied by *cos(alpha)*, where *alpha* is the angle between the Z-axis and the normal of the finite element.
- **C** - constant of the pressure expression.
- **Cx** - the pressure expression factor to be multiplied by the X coordinate.
- **Cy** - the pressure expression factor to be multiplied by the Y coordinate.
- **Cz** - the pressure expression factor to be multiplied by the Z coordinate.

9. General displacements

A **general displacement** represents a prescribed displacement (translation or rotation) of a finite element node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Ux** - the value of displacement in the direction of X axis.
- **Uy** - the value of displacement in the direction of Y axis.
- **Uz** - the value of displacement in the direction of Z axis.
- **Rx** - the value of rotation around the X axis.
- **Ry** - the value of rotation around the Y axis.
- **Rz** - the value of rotation around the Z axis.

10. General stiffnesses

A **general stiffness** represents an elastic support of a finite element node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Kx** - translational stiffness of the support in the direction of positive X axis.
- **Ky** - translational stiffness of the support in the direction of positive Y axis.
- **Kz** - translational stiffness of the support in the direction of positive Z axis.
- **Tx** - rotational stiffness of the support around the X axis.
- **Ty** - rotational stiffness of the support around the Y axis.
- **Tz** - rotational stiffness of the support around the Z axis.

11. Node types

A **node type** represents an object holding various data for a finite element node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **FixUx** - a switch indicating that the displacement in the X direction is prevented and equal to zero.
- **FixUy** - a switch indicating that the displacement in the Y direction is prevented and equal to zero.
- **FixUz** - a switch indicating that the displacement in the Z direction is prevented and equal to zero.
- **FixRx** - a switch indicating that the rotation around the X axis is prevented and equal to zero.
- **FixRy** - a switch indicating that the rotation around the Y axis is prevented and equal to zero.
- **FixRz** - a switch indicating that the rotation around the Z axis is prevented and equal to zero.
- **ForceInx** - the index of the applied point load from the general forces list.
- **DisplacementInx** - the index of the imposed displacement from the general displacements list.
- **StiffnessInx** - the index of the employed elastic support from the general stiffnesses list.

Note. In EKON there is a predefined node type 0 which defines a fully supported (all displacements/rotations equal to zero) node with 0 degrees of freedom.

12. Nodes

A **node** represents a finite element node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **TypeInx** - the index of the corresponding nodal type from the node types list.
- **MasterNodeInx** - index of the *master* node from the nodes list. If different from *(null)*, the current node is considered to be of the *slave* type. In that case the *fixed* degrees of freedom of the slave node are copied from the master node.
- **DesignElementInx** - the index of the design element from the design elements list. If different from *(null)*, the current node position is considered to be defined relatively to the local coordinate system of the design element.
- **X** - either the X coordinate of the node or its first local coordinate with respect to the design element.
- **Y** - either the Y coordinate of the node or its second local coordinate with respect to the design element.
- **Z** - either the Z coordinate of the node or its third local coordinate with respect to the design element.

Note. The local coordinates of the design elements (QBB, TBB) are running from *0* to *1*. Thus, if the position of the node is defined with respect to the design element, its coordinates (X, Y, Z) have to be within the interval *[0, 1]*.

13. Elastic materials

An **elastic material** represents a data object holding the data of an isotropic and linearly elastic material.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Modulus** - the value of Young's modulus (E).
- **PoissonRatio** - the value of Poisson ratio.
- **Density** - the density of matter.
- **GravityAccX** - the applied acceleration in the direction of X axis.
- **GravityAccY** - the applied acceleration in the direction of Y axis.
- **GravityAccZ** - the applied acceleration in the direction of Z axis.

Note. The applied acceleration may be used to model the gravity (weight) of the structure or any other volume load related linearly to the mass of the structure.

14. Cross sections

A **cross section** represents an object that holds the geometrical data of a cross section.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Code** - the code of the cross section - see the table below.
- **Par1** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par2** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par3** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par4** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par5** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par6** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par7** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par8** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par9** - the value (constant or parameter) of the parameter according to the cross section code.
- **Par10** - the value (constant or parameter) of the parameter according to the cross section code.

Note. The table of Cross-section codes and the required parameters.

15. Spring finite element Sp2

A **spring finite element Sp2** is a cinematically nonlinear 2-node spring finite element with 3 degrees of freedom (3 displacements) per node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Stiffness** - the value of the spring stiffness.
- **PreLoad** - the value of the preload (axial force at zero displacements).
- **Nodes** - a record of 2 indexes of nodes from the nodes list.

Note. The indexes of the nodes should be separated by space (blank) characters.

16. Truss finite element T2

A **truss finite element T2** is a cinematically nonlinear 2-node truss finite element with 3 degrees of freedom (3 displacements) per node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **MaterialInx** - the index of the employed material from the elastic materials list.
- **CrossSecInx** - the index of the employed cross section from the cross section list.
- **Nodes** - a record of 2 indexes of nodes from the nodes list.

Note. The indexes of the nodes should be separated by space (blank) characters.

17. Beam finite element B2

A **beam finite element B2** is a cinematically nonlinear 2-node beam finite element with 6 degrees of freedom (3 displacements and 3 rotations) per node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **MaterialInx** - the index of the employed material from the elastic materials list.
- **CrossSecInx** - the index of the employed cross section from the cross section list.
- **G3Code** - the code that determines the direction of the G3 vector - the third unit vector of the local orthogonal frame of the beam element - see the table below.
- **Nodes** - a record of 2 indexes of nodes from the nodes list.

Note. The sequence of the specified nodes is important. It determines the orientation of the local coordinate system of the element. The indexes of the nodes should be separated by space (blank) characters.

Note. The table of G3 codes.

18. Beam finite element B3

A **beam finite element B3** is a cinematically nonlinear and arbitrarily curved 3-node beam finite element. It has 6 degrees of freedom (3 displacements and 3 rotations) per each end node and 3 degrees of freedom (3 rotations) at the middle node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **MaterialInx** - the index of the employed material from the elastic materials list.
- **CrossSecInx** - the index of the employed cross section from the cross section list.
- **G3Code** - the code that determines the direction of the G3 vector - the third unit vector of the local orthogonal frame of the beam element - see the table below.
- **Nodes** - the record of 3 indexes of nodes from the nodes list in the following order (first end node, second end node, middle node).

Note. The sequence of the specified nodes is important. It determines the orientation of the local coordinate system of the element. The indexes of the nodes should be separated by space (blank) characters. The middle node must not be of the same type as the end nodes since it has only 3 degrees of freedom.

Note. The table of G3 codes.

19. Plane stress finite element PSS8

A **plane stress finite element PSS8** is a cinematically linear 8-node plane stress finite element with 2 degrees of freedom (2 displacements) per node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **MaterialInx** - the index of the employed material from the elastic materials list.
- **Thickness** - the thickness (constant or parameter) of the element.
- **Nodes** - a record of 8 indexes of nodes from the nodes list.

Note. The sequence of the specified nodes is important. It determines the orientation of the local coordinate system of the element. The indexes of the nodes should be separated by space (blank) characters.

20. Solid finite element S20

A **solid finite element S20** cinematically linear 20-node solid finite element with 3 degrees of freedom (3 displacements) per node.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **MaterialInx** - the index of the employed material from the elastic materials list.
- **Nodes** - a record of 20 indexes of nodes from the nodes list.

Note. The sequence of the specified nodes is important. It determines the orientation of the local coordinate system of the element. The indexes of the nodes should be separated by space (blank) characters.

21. Shell finite element S4

A **shell finite element S4** is a cinematically linear 4-node shell finite element with 5 degrees of freedom per node (3 displacements and 2 rotations). The drilling rotation is not taken into account.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **MaterialInx** - the index of the employed material from the elastic materials list.
- **Thickness** - the thickness (constant or parameter) of the element.
- **Nodes** - a record of 4 indexes of nodes from the nodes list.
- **PressureInx** - the index of the pressure load from the general pressures list.

Note. The sequence of the specified nodes is important. It determines the orientation of the local coordinate system (and the positive normal direction) of the element. The indexes of the nodes should be separated by space (blank) characters.

22. Analysis settings

An **analysis setting** is an object holding the analysis settings data.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **LinearAnalysis** - a switch indicating that only linear analysis has to be run.
- **LoadIncrement** - the value of load increment (nonlinear analysis).
- **MaxIteration** - the maximum of allowed iterations.
- **AbsResidualEps** - the absolute residual epsilon (tolerance) employed in measuring the convergence (nonlinear analysis).
- **RelResidualEps** - the relative residual epsilon (tolerance) employed in measuring the convergence (nonlinear analysis).

Note. Although several **analysis settings** objects may be defined, when running the analysis, **only** the object with the index equal to **1** is taken into account.

23. Meshing tools

A **meshing tool** is an object holding the data for the meshing utility.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **DesignElementInx** - the index of the design element from the design elements list.
- **DERange1Min** - the lower boundary of the meshing region in direction 1of the design element.
- **DERange1Max** - the upper boundary of the meshing region in direction 1of the design element.
- **DERange2Min** - the lower boundary of the meshing region in direction 2 of the design element.

- **DERange2Max** - the upper boundary of the meshing region in direction 2 of the design element.
- **DERange3Min** - the lower boundary of the meshing region in direction 3 of the design element.
- **DERange3Max** - the upper boundary of the meshing region in direction 3 of the design element.
- **ObjectComment** - the comment assigned to all new objects (nodes and finite elements).
- **NodeTypeInx** - the index of the nodal type from the node types list. All new nodes will be of the specified type.
- **NumFECells1** - the number of finite elements/cells in the direction 1.
- **NumFECells2** - the number of finite elements/cells in the direction 2.
- **NumFECells3** - the number of finite elements/cells in the direction 3.
- **FinalFECell1** - a switch indicating that the end cells in the direction 1 should be finalized (closed).
- **FinalFECell2** - a switch indicating that the end cells in the direction 2 should be finalized (closed).
- **FinalFECell3** - a switch indicating that the end cells in the direction 3 should be finalized (closed).
- **FECode** - the code of the finite element:
  - T2 - Truss T2 finite element.
  - B2 - Beam B2 finite element *(not yet implemented)*.
  - B3 - Beam B3 finite element *(not yet implemented)*.
  - PSS8 - Plane stress PSS8 finite element.
  - S20 - Solid S20 finite element *(not yet implemented)*.
  - S4 - Shell S4 finite element.
- **FEMaterialInx** - the index of the employed material from the elastic materials list. All new elements will be of the specified material.
- **FECrossSecInx** - the index of the employed cross section from the cross section list. All new elements (that hold cross section references) will have the specified cross section.
- **FEG3Code** - the code that determines the direction of the G3 vector. All new beam elements will have the specified G3 code.
- **FEThickness** - the thickness (value or parameter) of the finite elements. All new elements (that hold thickness references) will have the specified thickness.
- **FEPressureInx** - the index of the pressure load from the general pressures list. All new elements (that hold pressure references) will be loaded by the specified pressure.

Note. The local coordinates of the design elements (QBB, TBB) are running from 0 to 1. Thus, the meshing region boundaries (lower and upper) in all directions must be within the interval [0, 1].

24. Node typing tools

A **node typing tool** is an object holding the data for the node typing utility.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **DesignElementInx** - the index of the design element from the design elements list.
- **DERange1Min** - the lower boundary of the design element in the direction 1.
- **DERange1Max** - the upper boundary of the design element in the direction 1.
- **DERange2Min** - the lower boundary of the design element in the direction 2.
- **DERange2Max** - the upper boundary of the design element in the direction 2.
- **DERange3Min** - the lower boundary of the design element in the direction 3.
- **DERange3Max** - the upper boundary of the design element in the direction 3.
- **Tolerance** - the value of the tolerance used to determine whether some node is within the specified range.

- **NodeComment** - the comment assigned to all modified nodes. If *(null)* no comment is assigned to.
- **NodeTypeInx** - the index of the nodal type from the node types list. This type is assigned to the nodes within the specified range.

Note. The local coordinates of the design elements (QBB, TBB) are running from 0 to 1. Thus, the boundaries (lower and upper) in all directions must be within the interval [0, 1].

25. Node slaving tools

A **node slaving tool** is an object holding the data for the node slaving utility.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Tolerance** - the absolute value of the position tolerance used to determine whether two nodes are coincident.
- **OnNodeComment** - the required comment of the slave node. If the comment of the candidate slave node does *not* match the required comment, *no* slaving is performed. If the required comment is *(null)*, any found slave candidate node is slaved.
- **NodeTypeInx** - the index of the nodal type from the node types list. This type is assigned to all slaved nodes.

26. Response quantities

A **response quantity** is an object holding the response quantity data.

The following data has to be provided:

- **Index** - a unique positive index.
- **Comment** - an optional comment on the meaning of the object.
- **Step** - (Not implemented in EKON)
- **Status** - the status of response quantity:
  - 0 - the response quantity is not calculated nor displayed.
  - 1 - the response quantity represents an objective function.
  - 2 - the response quantity represents a constraint.
- **Code** - the code of the response function *f*. See the table below.
- **CompareValue** - the value *c* to be compared to *f*.
- **Square** - a switch indicating that the term *(f-c)* has to be squared.
- **Absolute** - a switch indicating that the term *(f-c)* has to be multiplied by *-1* if its value is less than zero.
- **NormalizeValue** - the value *n* used to normalize the term *(f-c)*.
- **AllowableValue** - the value *a* to be subtracted from the term *(f-c)/n*.

Note. The response quantity *r* is evaluated as follows:

$r = (g / n) - a$

where

- $g = (f - c)$ if neither the 'Square' nor the 'Absolute' switch is checked,
- $g = (f - c)^2$ if the 'Square' switch is checked,
- $g = ABS(f - c)$ if only the 'Absolute' switch is checked.

Note. The response function *f* is defined by the codes given in the table below.

Priloga C: Programska koda EKON2dxf

Za potrebe računa nelinearne analize, uklonske nosilnosti in prikaza notranje statičnih količin, je bilo potrebno geometrijo konstrukcije prenesti v komercialni program SAP2000, kjer sem izračunal nelinearno analizo. Ker lahko geometrijo konstrukcije v SAP2000 podamo tudi z uvozom ACAD-ove dxf datoteke, sem v programu Mathematica sprogramiral proceduro, ki iz vhodne EKON-ove xml datoteke prebere podatke o projektnih spremenljivkah, osnovnih podatkih konstrukcije in zapiše podatke o legi končnih elementov v dxf obliko, ki jo lahko uvozimo v program SAP2000.

```
Prenos geometrije konstrukcije (lupinastih KE) iz xml datoteke (EKON) v datoteko
dxf (ACAD) za SAP2000

Pomembno je, da se funkcije zaganjajo po vrsti kot so navedene – zaradi uporabe
globalnih  spremenljivk, ki se spreminjajo. Za druge primere se spremeni le ime
datotek in poti, nato pa je najbolje uporabiti funkcijo Kernel / Quit Kernel/ Local
in nato Kernel / Evaluation / EvaluateNotebook.

Definiranje vhodne (*.xml) datoteke, poti do programa ACAD
Definiranje poti in imena vhodne datoteke (brez koncnice –), imena izhodne datoteke
(brez koncnice) in poti do mesta Live3D za prikaz konstrukcije v Internet
Explorerju

pot=SetDirectory["d:\\diploma\\xml2dxf"];
pot3D=SetDirectory["d:\\Live3D"];
potACAD=SetDirectory["c:\\Program files\\AutoCAD 2002"];
potizhod=SetDirectory["d:\\diploma\\xml2dxf"];
ImeVhodneDatoteke="konzola";

Vnos podatkov iz vhodne datoteke

doc=Import[pot<>"\\"<>ImeVhodneDatoteke<>".xml"];

Branje podatkov iz xml datoteke
Branje projektnih spremenljivk (b), parametrov (x), kontrolnih tock (kt), vrstni
red KT, tocke (nodes), lupinastih KE(shell).
Branje projektnih spremenljivk

BeriDesignVariables[x_]:=Module[{i,k,a,b,c},BeriDesignVariables::usage="BeriDesignV
ariables[x] prebere vse projektne spremenljivke iz xml datoteke v vrstnem redu kot
so v EKON-u.";
    k=Cases[x,
        XMLElement["DesignVariables",_,{XMLElement["Index",_,{name_}],_,
              XMLElement["Value",_,{s_}],_}]:>
          {ToExpression[name],ToExpression[s]},Infinity];
    b=Table[0,{Length[k]}];
    For[i=1,i ≤ Length[k],
       {c=k[[i,1]];
              b[[c]]=k[[i,2]]
             };i++];
       Return[b];
```

```
    ]
 b=BeriDesignVariables[doc];

Funkciji za generiranje navodil za zamenjavo bi→b[[i]] in  xi→x[[i]]
 navod[xx_]:=Module[{a,b,c,d,f,i,p,nabo,k={}},
    navod::usage="navod[xx] generira listo navodil za spremembo bi->b[[i]], kjer je
i (dolzina xx) stevilo projektnih spremenljivk.";
    a="b";
    b="[[";
    c="]]";
    d="->";
    p=Length[xx];
    For[i=1,i ≤ p,i++,
      f=ToString[i];
      nabo=StringJoin[a,f,d,a,b,f,c];
      AppendTo[k,ToExpression[nabo]];
      ];
    Return[k];]

 navod1[ff_]:=Module[{a,b,c,d,f,i,p,nabo,k={}},
    navod1::usage="navod1[ff] generira listo navodil za spremembo xi→x[[i]], kjer
je i (dolzina xx) stevilo parametrov.";
    a="x";
    b="[[";
    c="]]";
    d="->";
    p=Length[ff];
    For[i=1,i≤ Length[ff],i++,
      f=ToString[i];
      nabo=StringJoin[a,f,d,a,b,f,c];
      AppendTo[k,ToExpression[nabo]];
      ];
    Return[k];]

Branje parametrov

BeriParameter[xx_]:=Module[{f,a,ll,i,j,t,p,k},
 BeriParameter::usage="BeriParameter[xx] prebere vse parametre iz xml datoteke, jih
izvrednosti in vrne v vrstnem redu kot so v EKON-u.";
    f=Cases[xx,XMLElement["Parameters",_,{XMLElement["Index",_,{name_}],_,
              XMLElement["Term",_,{s_}]}]:→{ToExpression[name],s},Infinity];
    j=Length[f];
    a=Table[{1,""},{j}];
    For[i=1,i ≤ Length[a],i++,
      a[[i,1]]=f[[i,1]];
      a[[i,2]]=ToExpression[StringJoin[Characters[f[[i,2]]]/.{"#"→"x","
"→"","."→"."}]];
      ];
    ll=Table[0,{j}];
    For[i=1,i ≤ Length[ll],i++,
      k=a[[i,1]];
      ll[[k]]=a[[i,2]];
      ];
    navodila=navod[b];
    x=ll/.navodila;
    navodila1=navod1[x];
    f=ll/.navodila;
    k=f//.navodila1;
    Return[k]]
 x=N[BeriParameter[doc]];

Branje kontrolnih tock
```

```
BeriControlPoints[xx_]:=Module[{t,i,a,h,k},
    BeriControlPoints::usage="BeriControlPoints[x] prebere vse kontrolne tocke iz
xml datoteke, jih izvrednoti in vrne v vrstnem redu kot so v EKON-u.";

t=Cases[xx,XMLElement["ControlPoints",_,{XMLElement["Index",_,{name_}],XMLElement["
X",_,{x_}],XMLElement["Y",_,{y_}],XMLElement["Z",_,{z_}],__}]:→{ToExpression[name],x
,y,z},Infinity];
    a=Table[{1,"","",""},{Length[t]}];
    For[i=1,i ≤ Length[t],i++,
      a[[i,1]]=t[[i,1]];
      a[[i,2]]=ToExpression[StringJoin[Characters[t[[i,2]]]/.{"#"→"x"," " → "","","
→ "."}]];
      a[[i,3]]=ToExpression[StringJoin[Characters[t[[i,3]]]/.{"#"→"x"," " → "","","
→ "."}]];
      a[[i,4]]=ToExpression[StringJoin[Characters[t[[i,4]]]/.{"#"→"x"," " → "" ,","
→ "."}]];
      ];
    h=Table[{0,0,0},{Length[a]}];
    For[i=1,i ≤ Length[a],i++,
      k=a[[i,1]];
      h[[k,1]]=a[[i,2]];
      h[[k,2]]=a[[i,3]];
      h[[k,3]]=a[[i,4]];
      ];
    Return[N[h//.navodila1]]
    ]
 kt=BeriControlPoints[doc];
```

Branje vrstnega reda kontrolnih tock

```
 BeriVrstniRedCP[xx_]:=Module[{},
    BeriVrstniRedCP::usage="BeriVrstniRedCP[x] prebere stevilo kontrolnih tock v
vseh treh smereh ter vrstni red kontrolnih
tock.";Cases[xx,XMLElement["QuadrangularBezierBodies",_,{__,XMLElement["NumCP1",_,{
CP1_}],XMLElement["NumCP2",_,{CP2_}],XMLElement["NumCP3",_,{CP3_}],_,XMLElement["Co
ntrolPoints",_,{KT_}]}]:→{{ToExpression[CP1],ToExpression[CP2],ToExpression[CP3]},KT
},Infinity]]
```

Branje tock mreze koncnih elementov

```
 BeriNodes[xx_]:=Module[{i,k,l,c,t},
    BeriNodes::usage="BeriNodes[xx] prebere koordinate vseh tock (s1,s2) v lokalnem
koordinatnem sistemu, ki definirajo koncne elemente.";

t=Cases[xx,XMLElement["EKo_Nodes",_,{XMLElement["Index",_,{index_}],__,XMLElement["
X",_,{x_}],XMLElement["Y",_,{y_}],_}]:→{ToExpression[index],{x,y}},Infinity];
    c=Table[{0,0},{Length[t]}];
    l=Table[{0,{0,0}},{Length[t]}];
    For[i=1,i ≤ Length[t],i++,
      k=t[[i,1]];
      l[[k,1]]=t[[i,1]];
      l[[k,2,1]]=ToExpression[StringJoin[Characters[t[[i,2,1]]]/.{","→"."}]];
      l[[k,2,2]]=ToExpression[StringJoin[Characters[t[[i,2,2]]]/.{","→"."}]];
      c[[k,1]]=l[[k,2,1]];
      c[[k,2]]=l[[k,2,2]];
      ];
    Return[N[c]]
    ]
 nodes=BeriNodes[doc];
```

Branje koncnih elementov (po 4 tocke)

```
 BeriShell[xx_]:=Module[{i,a,c,d,e,ll,t},
```

```
    BeriShell::usage="BeriShell[xx] prebere listo tock (po 4 tocke), ki definirajo
posamezen koncni element.";

t=Cases[xx,XMLElement["EKo_FE_ShellS4",_,{__,XMLElement["Nodes",_,{nodes_}],_}]:→{no
des},Infinity];
    e=Table[{0},{Length[t]}];
    For[i=1,i ≤ Length[t],i++,
      a=StringJoin[Characters[t[[i,1]]]/.{" "→","}];
      ll=StringInsert[a,"{",1];
      c=StringJoin[ll,"}"];
      d=ToExpression[c];
      e[[i,1]]=d;
      ];
    Return[N[Flatten[e,1]]]]
 shell=BeriShell[doc];


Preracun tock iz lokalnega koordinatnega sistema v globalni koordinatni sistem
Potrebni podatki za funkcijo RatBez : (u,v), m,n, razdelitev KT po smereh x,y,z

Racun u,v

Defuu[xx_,i_]:=xx[[i,1]];
Defvv[xx_,i_]:=xx[[i,2]];
u=Table[Defuu[nodes,i],{i,1,Length[nodes]}];
v=Table[Defvv[nodes,i],{i,1,Length[nodes]}];

Razdelitev kontrolnih tock  v tabelo m x n

 CP[xx_]:=Module[{i,a,o,c,d,e},
    CP::usage="CP[xx] razdeli kontrolne tocke v listo glede na stevilo KT v prvi
smeri (m x n).";
i=Cases[xx,XMLElement["QuadrangularBezierBodies",_,{__,XMLElement["NumCP1",_,{CP1_}
],XMLElement["NumCP2",_,{CP2_}],XMLElement["NumCP3",_,{CP3_}],_,XMLElement["Control
Points",_,{KT_}]}]:→{{ToExpression[CP1],ToExpression[CP2],ToExpression[CP3]},KT},Inf
inity];
    a=StringJoin[Characters[i[[1,2]]]/.{" "→","}];
    o=StringInsert[a,"{",1];
    c=StringJoin[o,"}"];
    d=ToExpression[c];
    e=Partition[d,i[[1,1,1]]];
    Return[e];]

Definicija  m in n, ki predstavljata stevilo KT v posameznih smereh

m=BeriVrstniRedCP[doc][[1,1,1]];
n=BeriVrstniRedCP[doc][[1,1,2]];
Razdelitev  KT po koordinatah (kx,ky,kz)
 xkt[x_]:=kt[[x,1]]
ykt[x_]:=kt[[x,2]]
zkt[x_]:=kt[[x,3]]
 kx=Map[xkt,CP[doc]];
ky=Map[ykt,CP[doc]];
kz=Map[zkt,CP[doc]];

Racun koordinat tock v globalnem koordinatnem sistemu
Funkcija DeCast za izracun tocke na Bezierovi krivulji (v eni smeri)

DeCast[n_,b1_,t_]:=Module[{r,i,j,b,p},
    DeCast::usage="Decast[n,b1,t] izracuna vrednost koordinate tocke v globalni
koordinatni sistemu s podatki: lokalno koordinato t∈{0,1}; kontrolne tocke b1 - v
eni smeri, ki definirajo Bezierovo krivuljo n-te stopnje.";
    b=b1;
```

```
    p=b;
    For[r=1,r ≤ n,r++,
      For[i=0,i ≤ n-r,i++,
        For[j=0,j ≤ n-r,j++,
            p[[i+1]]=N[(1-t)*b[[i+1]]+t*b[[i+2]]]
            ];
        ];
      b=Take[p,n+1-r];
      ];
    Return[Flatten[b]];
    ]

Funkcija RatBez za izracun tocke na Bezierovi ploskvi
 RatBez[m_,n_,Bx_,By_,Bz_,u_,v_]:=Module[{i,j,px,py,pz,tabelaxx,tabelayy,tabelazz},
   RatBez::usage="RatBez[m,n,Bx,By,Bz,u,v] izracuna vrednost koordinat tocke na
Bezierovi ploskvi (u[[i]],v[[i]]) pri koordinatah KT v treh smereh Bx, By, Bz, ter
stopnjo m v prvi smeri in stopnje n v drugi smeri.";
    tabelaxx={};
    tabelayy={};
    tabelazz={};
    For[i=1,i ≤ m+1,i++,
      tabelaxx=Append[tabelaxx,DeCast[n,Bx[[i]],u]];
      tabelayy=Append[tabelayy,DeCast[n,By[[i]],u]];
      tabelazz=Append[tabelazz,DeCast[n,Bz[[i]],u]];
      ];
    px=DeCast[m,tabelaxx,v];
    py=DeCast[m,tabelayy,v];
    pz=DeCast[m,tabelazz,v];
    Return[N[Flatten[{px,py,pz}],5]];
    ]

Racun

tocke={};For[i=1,i ≤ Length[nodes], i++,AppendTo[tocke,Chop[RatBez[n-1,m-
1,kx,ky,kz,u[[i]],v[[i]]],10^-5]]];tocke

Zapis koncnih elementov v ACAD obliko - .dxf
Zapis v datoteko .scr
Zapis[tocke_, lica_]:=Module[{i,j,k,str},
    Zapis::usage="Zapis[tocke,lica] zapise vse koncne elemente (lica) v ACAD obliko
(3DFACE).";
    str=OpenWrite[potizhod <>"\\"<>ImeVhodneDatoteke<>".scr"];
    WriteString[str,"-layer"," ","m"," ","shell"," ","  ","\n"];
    For[i=1, i ≤ Length[lica], i++,
      WriteString[str,"3DFACE "];
      For[j=1, j ≤ 4, j++,
        WriteString[str,tocke[[lica[[i,j]],1]]]; (* x koor. *)
        WriteString[str,","];
        WriteString[str,tocke[[lica[[i,j]],2]]]; (* y koor. *)
        WriteString[str,","];
        WriteString[str,tocke[[lica[[i,j]],3]]]; (* z koor. *)
        WriteString[str," "];
        ];
      WriteString[str,"\n"];
      ];
    strdxf=OpenWrite[potizhod <>"\\"<>ImeVhodneDatoteke<>".dxf"];
    WriteString[strdxf," "];
    Close[strdxf];
    strdwg=OpenWrite[potizhod <>"\\"<>ImeVhodneDatoteke<>".dwg"];
    WriteString[strdwg," "];
    Close[strdwg];
    WriteString[str,"zoom"," ","e","\n"];
    WriteString[str,"filedia"," ","0","\n"];
```

```
    WriteString[str,"saveas"," ","dxf"," ","16","
",potizhod<>"\\"<>ImeVhodneDatoteke<>".dxf"," ","y","\n"];
    WriteString[str,"saveas"," ","2000","
",potizhod<>"\\"<>ImeVhodneDatoteke<>".dwg"," ","y","\n"];
    WriteString[str,"filedia"," ","1","\n"];
    WriteString[str,"quit"," "];
    Close[str];
    ]
 Zapis[tocke,shell]

Klicanje Acad.exe in izvajanje narejene skripte, brisanje vmesnih datotek
(.scr,.dwg,.bak)

scr2dxf:=Module[{st},
    st="\""<>potACAD<>"\\acad.exe"<>"\""<>" /nologo"<>" /b "<>ImeVhodneDatoteke;
    Run[st];
DeleteFile[{potizhod<>"\\"<>ImeVhodneDatoteke<>".bak",potizhod<>"\\"<>ImeVhodneDato
teke<>".scr",potizhod<>"\\"<>ImeVhodneDatoteke<>".dwg"}];
     ]
scr2dxf

Graficni prikaz konstrukcije v Mathematici

 tabelatock = Table[Point[N[tocke[[i]],2] ], {i,1,Length[tocke]}] ;
 Show[Graphics3D[tabelatock],Axes→True,Boxed→False]

Prikaz v Internet Explorerju z funkcijo RealTime3D (Live3D poddirektorij!!!)
 <<RealTime3d`
 mmBrowserPath:="c:\\program files\\internet explorer\\iexplore.exe";
(*mmBrowserPath:="C:\\Program
Files\\Netscape\\Communicator\\Program\\netscape.exe";*)
mmIExplorerPath:=StringJoin["\"",mmBrowserPath,"\" "];
Live3DHead:="<HTML>\n<BODY BGCOLOR=#FFFFFF>\n<APPLET
ARCHIVE=\""<>"file:///"<>Live3DDirectory<>"live.jar\" CODEBASE=\".\"
CODE=\"Live.class\" WIDTH=700 HEIGHT=500 ALIGN=LEFT>\n<PARAM NAME=BGCOLOR VALUE
=#FFFFFF>\n<PARAM NAME=MAGNIFICATION VALUE=1.5>\n<PARAM NAME=INPUT_FILE VALUE=\"";
Live3DTail:="\">\n</APPLET>\n</BODY>\n</HTML>";
Live3DDirectory:=StringJoin[pot3D,"\\"];
Live3DFile:="live3df.html";
Live3DMMFile:="mmlive3d.m";

Live3D[b_Graphics3D]:=
  Module[{g,s,dir,str,str2,cmdl},
    g=ToString[InputForm[b]];
    s=StringJoin[Live3DHead,Live3DMMFile,Live3DTail];
    dir=StringJoin[Live3DDirectory,Live3DMMFile];
    str=OpenWrite[dir];
    WriteString[str,g];
    Close[str];
    dir=StringJoin[Live3DDirectory,Live3DFile];
    str=OpenWrite[dir];
    WriteString[str,s];
    Close[str];
    str2=StringJoin["file:///",ToString[str[[1]]]];
    cmdl=StringJoin[mmIExplorerPath,str2];
    Run[cmdl];
    ]
 DefineTocke[ff_]:=Module[{a,b},
    a=Table[{Point[N[kt[[i]],2]]},{i,1,Length[kt]}];
    b=Flatten[a];
    a=Prepend[b,PointSize[ff]];
    Return[Flatten[a]];]
 Live3D[Show[Graphics3D[{tabelatock,DefineTocke[0.015]}],Axes → True,
Boxed→False]];
```