Objektno orijentirane biblioteke za realizaciju metode konačnih elemenata s primjenom u brodogradnji

Zečević, Tadej-Ivan

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Tadej-Ivan Zečević

OBJEKTNO ORIJENTIRANE BIBLIOTEKE ZA REALIZACIJU METODE KONAČNIH ELEMENATA S PRIMJENOM U BRODOGRADNJI

Diplomski rad

Voditelj rada:
prof.dr.sc Luka Grubišić

Zagreb, srpanj, 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Posvećujem Bogu, roditeljima, sestrama i bratu te najdražoj Izabeli

Sadržaj

Sadržaj **iv**

Uvod **3**

1 Metoda konačnih elemenata **5**
1.1 Uvod 5
1.2 Matrica krutosti 9
1.3 Osnovni pravokutni element 18
1.4 Jacobijeva matrica i prirodne koordinate 21
1.5 Uvjeti kvalitete mreže 23

2 GMSH **25**
2.1 Uvod 25
2.2 Geometrija 26
2.3 Gmsh algoritmi 27

3 Praktični zadatak **37**
3.1 Pygmsh 37
3.2 Zadatak 37
3.3 Histogrami i grafovi 41
3.4 Diskusija provedenih eksperimenata 51
3.5 Dodatak(kodovi) 52

Bibliografija **65**

Uvod

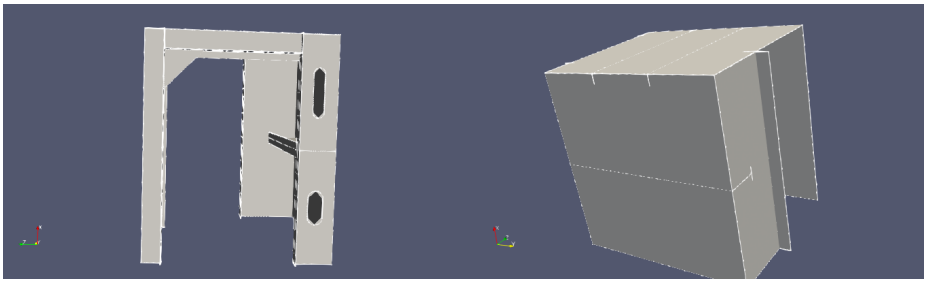
Objektno orijentirano programiranje (OOP) je programska paradigma temeljena na konceptu ”objekata”, programskih konstrukcija koje mogu sadržavati podatke u poljima, poznatih u terminologiji kao atributi. Također mogu sadržavati i kod, u obliku funkcija, u terminološkom nazivu metode. Značajka objekata je da metode nekog objekta mogu pristupiti i često mijenjati podatkovna polja objekta s kojim su povezani. U OOP-u, računalni programi su dizajnirani iz objekata koji međusobno ostvaruju interakciju. Postoji značajna raznolikost OOP jezika, ali najpopularniji su oni koji su bazirani na klasama. Klasa u OOP-u je proširivi programski kod- služi kao predložak za stvaranje objekata, pridruživanje početnih vrijednosti varijablama i implentacija metoda. Takva građa omogućuje dijeljenje objekta među aplikacijama. Ostvarenje tog zadatka vrši se kroz klijent-server okruženje kreirano tako da postoje objekti definirani klasama koje su poznate i klijentu i serveru. U praksi se takav način komunikacije ostvaruje objektno orijentiranim bibliotekama. Svaki simulacijski kod koji koristi metodu konačnih elemenata sastoji se od tri komponente: generiranje mreže, sklapanje matrica i linearna algebra te vizualizacija. Razvoj svake od ovih komponenti je poseban zadatak. Štovise, rješenja se često koriste za više različitih aplikacija. Objektno orjentirani pristup omogućava kombiniranje onih djelova svake od aplikacija koji je potreban za rješavanje specifičnog problema kojega ćemo promatrati unutar diplomskog rada. Iz razloga što smo u ovom diplomskom radu trebali koristiti objekte koji već postoje u određenim aplikacijama odlučili smo se na objektno orijentirani pristup u rješavanju problema. Korištenje pythona je logičan pristup u takvoj problematici zbog fleksibilnosti, jednostavnosti kodiranja, besplatne nabavke, ugrađenih datoteka koje omogućuju široku primjenu u rješavanju različitih problema, prilagođenosti objektno orijentiranom, funkcionalnom i mnogim ostalim paradigmama programiranja. U ovom diplomskom radu koristit ćemo se prvenstveno jednom od njegovih objektno orijentiranih biblioteka i njenoj realizaciji metode konačnih elemenata unutar pythona. Preko te biblioteke spajat ćemo se na GMSH-3D generator mreže konačnih elemenata. Biblioteka nosi upućujući naziv pygmsh i nastala je kao odgovor na složenost direktnog programiranja u GMSHov jeziku. U svome radu omogućuje jednostavnije zadavanje geometrije u ravnini i prostoru te do punog kapaciteta koristi prilagodljivost programiranja unutar pythona. Praktični zadatak koji ćemo rješavati unutar diplomskog rada je primjer problema particije

plohe u uniju međusobno disjunktnih trokuta (simpleksa). Proces generiranja takve particije nazivamo triangulacija plohe (općenito govorimo o teselaciji ili popločavanju plohe). Triangulacija je proces podjele plohe na uniju trokutnih (općenito simplicijalnih) podskupova. Uređeni par elementa triangulacije i konačno dimenzionalnog prostora funkcija (najčešće polinoma ograničenog stupnja) definiranih na danom simpleksu zovemo konačni element. U skupu triangulacija posebno će nas interesirati one triangulacija koje zadovoljavaju i Delaunayevo svojstvo [14]. Takve triangulacije ćemo nazivati Delaunayeve triangulacije. Njihovu važnost u kreiranju kvalitetne mreže konačnih elemenata ćemo spomenuti unutar drugog poglavlja. Za skup točaka P u Euklidskom prostoru Delaunay triangulacija je triangulacija u kojoj niti jedna točka iz P nije unutar opisane hipersfere bilo kojeg simpleksa unutar Delaunay triangulacije. Pojam vrijedi i za neeuklidske norme, ali onda triangulacija ne mora biti jedinstvena (unutar drugog poglavlja proučavat ćemo beskonačnu normu u tom kontekstu). U ovom radu bavit ćemo se konačnim elementima s polinomijalnim stupnjevima slobode gdje svaki konačni element triangulacije smije imati najviše polinomni pomak fiksnog stupnja (npr. u ovom radu stupnja 2). Dakle, triangulacija je skup trokuta, a mreža je skup vrhova i veza među njima (dva vrha su povezana ako se nalaze na istom bridu elementa triangulacije). U tekstu ćemo slobodno koristiti pojam mreže konačnih elemenata, unatoč tome što ne odgovara direktnoj definiciji pojma. Koristit ćemo također inženjerski pristup rješavanju problema jer omogućuje najkraću vezu između problema teselacije ploha i generiranog diskretnog modela promatranog kontinuiranog objekta. Također koristit ćemo i inženjersku terminologiju zbog lakše interpretacije i validacije realiziranog rješenja.

U prvom dijelu rada bit će općenito opisana metoda konačnih elemenata te ćemo objasniti zašto je jedna od najpopularnijih numeričkih metoda za rješavanje parcijalnih diferencijalnih jednadžbi. Zatim ćemo prezentirati osnovni algoritam sklapanja matrice krutosti za nekoliko osnovnih klasa konačnih elemenata te ćemo istaknuti njihova svojstva. Na temelju ovih razmatranja prezentirat ćemo nekoliko mjera kvalitete generirane mreže konačnih elemenata. Unutar drugog poglavlja opisat ćemo GMSH generator mreža, neke osnovne karakteristike algoritama koje smo koristili za dobivanje mreže te geometrijske naredbe koje smo koristili unutar GMSH programiranja. Primjetit ćemo da svi algoritmi kreiraju trianguliranu mrežu (mrežu podijeljenu na uniju trokutnih konačnih elemenata). U našem radu je izrazito važno imati mrežu četverokuta jer su formalni zahtjevi analize konstrukcije takvi da se moraju koristiti četverokutni elementi (koji odgovaraju fizičkim pločama od kojih se sklapa konstrukcija). Tu nam treba pojam rekombinacije- dobivanja mreže četverokuta iz mreže trokuta. To se ostvaruje spajanjem susjednih trokuta u mreži trokutnih konačnih elemenata. Postavit ćemo karakteristike najkvalitetnijeg pristupa u kreaciji mreže za dobru rekombinaciju koji je baziran na mješavini Frontal-Delaunay algoritmu. Time se ostvaruje dobar uvod u poglavlje o praktičnom zadatku koji je zaprimljen u suradnji s Fakultetom strojarstva i brodogradnje Sveučilišta u Zagrebu. Osvrnut ćemo se na pygmsh python bi-

blioteku koja nam je bila potrebna za analizu podataka o mreži. Preko python biblioteke i GMSH generatora ćemo u praksi vršiti kreiranje mreže trokuta i navedenu rekombinaciju. Unutar našeg diplomskog rada zadatak je bio dobiti što bliže ispunjenje sljedeća 4 kriterija za mrežu: omjer najveće i najmanje stranice u četverokutu mora biti između 1:4 (prihvatljivo) i 1 (idealno), mreža mora biti sastavljena većinom od četverokuta, a ako neki trokut "preživi" rekombinaciju omjer njegovih "kateta" treba biti između 1:2 i 1 te na kraju najmanji kut unutar četverokuta ne smije biti manji od 60 stupnjeva. Tu vidimo važnost spominjanja osnovnih trokutnih i četverokutnih elemenata unutar prve cjeline jer kvaliteta tih elemenata utječe na kvalitetu i točnost same metode. Nakon što smo naveli uvjete, analiziramo koliko dobiveni podaci o mreži zadovoljavaju iste te uvjete (prilikom kreiranja mreže testiramo šest Gmsh algoritama na oglednom primjeru). Usporedit ćemo kvalitetu mreže dobivenu automatskim kreiranjem geometrije s nadziranim pristupom popravljanja geometrije u iteracijama. Nadzirani pristup ostavlja otvoreni prostor za mogućnost buduće automatizacije tog pristupa. Rad završava dodatkom s kodovima koji su bili napisani za potrebe praktičnog zadatka.

Prvo poglavlje se bazira na knjizi Jurice Sorića: Metoda konačnih elemenata(odakle su preuzete i slike) [3] , drugo na dostupnu dokumentaciju o GMSH generatoru [2] i na znanstveni članak preuzet s interneta koji obrađuje problematiku pristupa kreaciji mreže preko Frontal-Delaunay metode upotrebom beskonačne norme(iz tog znanstvenog članka su sve slike u tom poglavlju) [1]. Treće i završno poglavlje se bazira dijelom na pygmsh dokumentaciji [4] i navedenom članku, a većim dijelom na samostalnom radu i rješavanju problemskog zadatka. Na slici ispod možemo vidjeti brodsku konstrukciju na kojoj trebamo kreirati kvalitetnu mrežu. Konstrukcija je došla iz CAD (Computer-aided Design) sustava u obliku .stp datoteke. Sadržava određene greške i poteškoće u obliku ne zatvorenih linijskih petlji (line loop- vidjeti drugu cjelinu) i nepovezanih utora s ostatkom brodske konstrukcije.



CAD model dijela palubne konstrukcije

Poglavlje 1

Metoda konačnih elemenata

1.1 Uvod

Metoda konačnih elemenata danas je veoma bitan i nezaobilazan faktor u inženjerskim proračunima. Velik broj programa koji su kreirani na temelju metode konačnih elemenata omogućuju analizu zadanih konstrukcija bez razmatranja složenih teorija u pozadini problema. Princip rada tih programa sastoji se od zadavanja ulaznih podataka prema uvjetima problema, a izlazni podaci su najčešće grafički prikazi. Mana ovakvog (Black-Box) prikaza je u tome što zanemarujemo skrivene složene teorije u rješavanju inženjerskih problema. Nadalje takav pristup može dovesti do pogrešne procjene stanja u zadanoj konstrukciji što opet može ugroziti njezinu čvrstoću i stabilnost. Moramo imati na umu da je metoda konačnih elemenata aproksimativna numerička metoda. To znači da su dobivena rješenja približna, približavanje realnim vrijednostima moguće je jedino uz dobar odabir konačnih elemenata i proračunskog modela. Ovdje vidimo da je za put prema realnim vrijednostima potrebno razumijevanje i teorijske osnove konačnih elemenata, ali i fizikalnog ponašanja konstrukcije koju proučavamo. Uz puno znanstvenog rada i kritičkog osvrta već ostvarenog postiže se mogućnost dohvaćanja realnijih rezultatskih vrijednosti.

Problemima kontinuiranih sustava pristupa se rješavanjem, često puta složenih, diferencijalnih jednadžbi. Takav pristup zapinje na složenijim proračunskim modelima jer je točno analitičko rješenje diferencijalnih jednadžbi moguće dobiti samo za jednostavnije modele. Općenito, veoma je teško dobiti rješenje koje zadovoljava diferencijalnu jednadžbu izvan lokalnih područja, tj. na čitavom području promatranja. U takvim slučajevima pribjegava se upotrebi numeričkih metoda koje se temelje na diskretizaciji kontinuiranog sustava. Veoma često to je uspješan pristup jer se događa zamjena diferencijalnih jednadžbi sa sustavom algebarskih jednadžbi.

Metoda konačnih elemenata spada u takvu vrstu numeričkih metoda i temelji se na fizičkoj diskretizaciji kontinuiranog prostora. Promatrani kontinuirani prostor s beskonačno

stupnjeva slobode gibanja (beskonačno nezavisnih čimbenika koji utječu na sam kontinuirani prostor) upotrebom metode konačnih elemenata, zamjenjuje se s diskretnim modelom međusobno povezanih elemenata s ograničenim stupnjevima slobode funkcije pomaka (u ovom radu se koristi za probleme u mehanici, ali vrijedi i za rješavanje ostalih parcijalno diferencijalnih jednadžbi na ograničenim prostorima, npr. problem kretanja temperature). Detaljnije, područje kontinuuma razbija se na konačan broj potprostora koji se nazivaju konačni elementi. Skup takvih međusobno povezanih konačnih elemenata se naziva mreža. Točke u kojima se konačni elementi povezuju nazivaju se čvorovi, a samo stanje u svakom elementu- koje može varirati od fizikalnih pojava, pomicanja u prostoru, naprezanja do svih ostalih veličina u polju opisano je pomoću interpolacijskih funkcija. Veoma je bitno da spomenute interpolacijske funkcije zadovoljavaju uvjete koji omogućuje diskretiziranom modelu što bolji opis kontinuiranog sustava. Kada se konačni elementi pravilno formuliraju, uz njihovu količinu raste kvaliteta točnosti rješenja. Izrada algebarskih jednadžbi za disretizirani model kreće od diferencijalnih jednadžbi koje opisuju stanje u elementu ili kroz upotrebu varijacijske formulacije. Sada smo u posjedu jednadžbi za konačan element, gdje su nepoznanice neovisne varijable u čvorovima. Sljedeći korak je odgovarajućim postupcima izvesti opće jednadžbe za diskretizirani model. Pomažu nam izračunate čvorne veličine jer iz njih pomoću poznatih teorijskih relacija možemo odrediti sve veličine potrebne za opis kontinuiranog sustava. Što je konstrukcija složenija, zahtjeva diskretizaciju s većim brojem konačnih elemenata što opet povlači veliki sustav algebarskih jednadžbi koji je veoma teško riješiti bez upotrebe računala. To nas dovodi pred činjenicu da je rješavanje problema s metodom konačnih elemenata potrebno imati odgovarajuće računalne programe. Metoda konačnih elemenata ima široko područje primjene kao što su rješavanje statičkih i dinamičkih problema, proračuni temperaturnih polja, strujanja te analiza elktromagnetskog polja. Izvođenje sustava algebarskih jednadžbi je analogno za sva navedena područja.

Pristup izvođenja jednadžbe individualnog konačnog elementa, koji se temelji na pronalasku rješenja diferencijalnih jednadžbi, naziva se metoda težinskog reziduala. Postupak je sljedeći: pretpostavljamo približno rješenje, to je najčešće funkcija neovisnih parametara u čvorovima elementa, te uvrštavamo to rješenje unutar diferencijalne jednadžbe. Ako je ostatak koji se pojavio unutar izračuna jednak nuli, rješenje diferencijalne jednadžbe je točno. Ostatak zovemo rezidual. U protivnom slučaju metodama minimalizacije reziduala dolazimo do sustava algebarskih jednadžbi čije nepoznanice tvore parametri u čvorovima. Opisani postupak se primjenjuje kada je varijacijska formula presložena, kada ne postoji funkcional koji opisuje zadani problem ili kada rješenje nije prikazivo zatvorenom formulom.

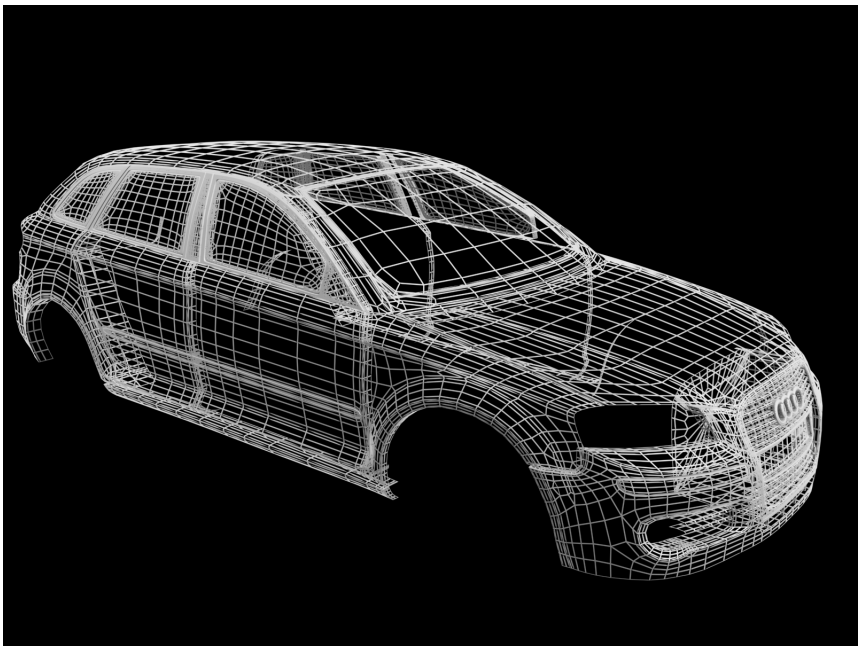
Drugačiji pristup za dobivanje jednadžbe konačnog elementa temelji se na varijacijskoj formulaciji. Najčešće se ovakav pristup koristi u rješavanju problema unutar mehanike deformabilnih tijela i u ovisnosti o neovisnim varijablama koje sudjeluju u varijaciji. Imamo nekoliko različitih principa u pristupu : princip virtualnih pomaka, princip mini-

muma ukupne potencijalne energije gdje su neovisne varijable pomaci, princip virtualnih sila i princip minimuma komplementarne energije gdje su neovisne varijable sile. Veoma česte su upotrebe i tzv. proširenih varijacijskih principa s više neovisnih polja (naprezanja, deformacije i pomaci). Metoda pomaka je najčešće u upotrebi, unutar nje su nepoznate veličine pomaci u čvorovima konačnih elemenata. Zadane sile i pomaci povezuju se preko zadane matrice krutosti koju ćemo kasnije definirati. Osim ovdje spomenutih metoda kod rješavanja problema metodom konačnih elemenata pojavljuju se i mješovite formulacije, temeljene na više različitih polja (najčešće dva) neovisnih varijabli. Mješovite metode pridonose rješavanju i boljem razumijevanju rubnih uvjeta na način da se promatra unutarnji dio konačnog elementa i pomaci i naprezanja na tom dijelu konačnog elementa te se promatra (neovisno o unutarnjem dijelu) raspodjela pomaka i naprezanja na rubnim dijelovima konačnog elementa. Početak kreiranja metode konačnih elemenata teško je ustvrditi. Ideja o opisu kontinuranog sustava s konačno mnogo diskretnih elemenata potječe iz 40-tih godina 20. stoljeća. Kao pionire korištenja metode konačnih elemenata ćemo istaknuti: J.H Argyris [5], Richard Courant [6], Gilbert Strang [7]. U njihovim radovima vidimo zanimljive ideje i rješenja, posebno se ističu podjelom elastičnog kontinuuma na manje dijelove. Sam otac pojma konačan element je američki inženjer Ray William Clough [8]. Pojam je uveden u upotrebu 1960. godine , otprilike u isto vrijeme kada kreće upotreba računala u rješavanju inženjerskih problema. R.W. Clough je u svojem radu iz 1965. godine uspio dokazati mogućnost upotrebe varijacijske Rayleigh-Ritzove metode u izvođenju jednadžbe konačnog elementa. Ovdje je veoma bitno spomenuti matematičara Richarda Couranta, na čijem se radu iz 1943. zasniva matematička interpretacija metode konačnih elemenata. Na taj način je metoda konačnih elemenata dobila svoju matematičku formulaciju što je pokrenulu široku upotrebu istoimene metode. Metoda napušta isključivu primjenu na području teorije elastičnosti i kreće mini revolucija. Upotreba se širi- prvo na područje mehanike fluida, prijelaza topline,a u novije vrijeme i na područje zrakoplovstva, projektiranja, konstruiranja, automobilske industrije te na posebno zanimljivom području biomehanike. Pregledom obavljenih istraživanja na području metode konačnih elemenata, dolazimo do podatka da su moderna istraživanja u velikoj mjeri usmjerena k rješavanju nelinearnih problema. Općenito, na svakom polju istraživanja ideja je približiti se što više realnim procesima koje promatramo i istražujemo. To se postiže uz što veću numeričku učinkovitost, odnosno u kraćem vremenu postići što više točnih rješenja. Postiže se i napredak u razvoju novih elemenata koji mogu lakše opisivati sve složenije konstrukcije modernoga vremena. Bitno je napomenuti postojanje algoritama za automatsko generiranje mreže (Adaptive Finite Element Refinement) koji služe za procjenu greške pri postupku diskretizacije. Automatizirana kontrola omogućuje promjenu veličine elemenata u cilju postizanja točnije mreže. Taj postupak dobivanja točnosti mreže kroz veličinu konačnih elemenata se naziva $h-postupak$. Imamo još i $p-metodu$ koja koristi metodu povećanja stupnja polinoma interpolacijske funkcije pri čemu su veličine elemenata nepromjenjene.

Kombinirana metoda je mješavina h i p metode te se naziva $h/p - metoda$. Kod kombinirane metode valja istaknuti rad američko-češkog matematičara Ive Babuške [15]. U modernim vremenima metoda konačnih elemenata upotrebljava se i u procesu (i optimizaciji) konstruiranja.

Postoje različiti tipovi konačnih elemenata. Oblik konačnih elemenata koji ćemo upotrijebiti u ovisnosti je o parametrima u čvorovima. Složenost tih parametara utječe i na složenost interpolacijske funkcije - što je veći broj nepoznanica koje koristimo veća je složenost. Najjednostavniji konačni elementi služe za rješavanje jednodimenzionalnih problema (štapni i gredni elementi). Od njih je osnovni štapni element koji posjeduje dva stupnja slobode dok gredni element ima 4 stupnja slobode. Uz pomoć konačnih elemenata za dvodimenzionalnu analizu moguće je opisati stanje naprezanja i deformacije u ravnini. Tijekom tog procesa dvije komponente pomaka su nepoznati parametri u čvorovima. Trodimenzionalne konačne elemente poput prizmi raznih oblika koristimo za trodimenzijsku analizu i tim slučajevima imamo čvorove s tri komponente pomaka u smjeru Kartezijevih koordinatnih osi. Prstenaste elemente služe pri analizi osnosimetričnih tijela te posjeduju stupnjeve slobode u obliku radijalnih i osnih pomaka. Pri rješavanju problema savijanja ploča koristimo piramidalne i kvadralne oblike gdje je jedna od dimenzija skoro zanemarive veličine. Također posjeduju po tri stupnja slobode u svakom čvoru i to su redom: pomaci u pravcu normale na srednju plohu ploče te kutevi zakreta normale oko dvije osi u ravnini ploče. Postoje konačni elementi za analizu ljuskastih konstrukcija. Često puta posjeduju i po 6 stupnjeva slobode usvakom čvoru, a same elemente znamo dobivati superpozicijom osnovnog elementa za probleme savijanja ploča i elementa za dvodimenzionalnu analizu. Sve navedene elemente moguće je proširiti dodavanjem neovisnih veličina u svakom čvoru ili dodavanjem čvorova (i na bridovima i na površini elementa). Osim spomenutih elemenata danas su u upotrebi još mnogi specijalni konačni elementi. Oni se koriste za analize mehaničkih problema. Postoje singularni elementi za probleme mehanike loma, neskonačni elementi za diskretizaciju beskonačnih ili polubeskonačnih područja. Specijalni konačni elementi, nazvani superelementi, koriste se i za analizu problema kod složenih konstrukcija zrakoplova, plovnih objekata,... Oni nastaju uparivanjem više konačnih elemenata gdje se elminiraju unutarnji čvorovi, a zadržavaju samo najvažniji vanjski- u litiraturi nazivani superčvorovi. Kod problema projektiranja, gdje često imamo potrebu smanjiti broj stupnjeva slobode, najbolje je koristiti složenije konačne elemente koji se zovu makroelementi. U tu skupinu spadaju element orebrene ljuske ili ploče, trodimenzionalni element tankostjene kutije te drugi slični konačni elementi. Na fotografiji ispod prikazana je kreacija mreže konačnih elemenata na konstrukciji automobila. Unutar automobilske industrije često se koriste raznovrsne kombinacije koje vode rješenju problema- ravni i zakrivljeni ljuskasti elementi u kombinaciji s grednim elementima, trodimenzionalni i ljuskasti elementi, elementi za analizu ploča i ljusaka,... Uvod o metodi konačnih elemenata nas navodi na zaključak o korisnoj i širokoj upotrebi navedene metode. Velika količina

modernih konstrukcija upotrebljava metodu konačnih elemenata u barem jednom dijelu svojega razvoja i kreacije.



slika 0
(preuzeto sa stranice:
http://unit66alex.blogspot.com/2014/01/ha4-task-4-mesh-construction.html)

U sljedećim potpoglavljima bavit ćemo se sklapanjem matrice krutosti, osvrtom na neke uvjete kvalitete mreže te odgovoriti na pitanje zašto je bitno koristiti prirodne koordinate za konačne elemente.

1.2 Matrica krutosti

Direktno sklapanje matrice krutosti

Tijekom rješavanja problema mehanike deformabilnih tijela upotrebom metode konačnih elemenata, nužnost je izvesti jednadžbe koje povezuju ovisnost sila u čvorovima konačnog elementa i pripadnih pomaka. Ključan dio je upravo matrica krutosti koja povezuje sile u čvorovima s pomacima preko relacija teorije elastičnosti. Postupak izvođenja treba raspoređiti u više koraka:

$$\|y - y_1\|_2 = a \; i \; \|y - y_2\|_2 = a\sqrt{2}$$

- Raspoređenost pomaka u elementu prikazuje se ovisno o pomacima u čvorovima:
  u=Nv
  Ovdje **N** označava matricu funkcija koje opisuju oblik raspodjele pomaka. Pomaci u čvorovima v označavaju stupnjeve slobode konačnog elementa.

- Kinematičke relacije su ključne za pronalazak veze između deformacije u elementu i pomaka u čvorovima:
  $\boldsymbol{\epsilon = D_k u}$
  gdje **u** označava raspodjelu pomaka u elementu,$\boldsymbol{\epsilon}$ označava komponente tenzora (poopćenje skalara i vektora) deformacije koje se određuju deriviranjem komponenti pomaka, $\boldsymbol{D_k}$ označava kinematički diferencijalni operator koji je definiran na sljedeći način:

$$\boldsymbol{D_e = \begin{bmatrix} \partial x & 0 & 0 & \partial y & 0 & \partial z \\ 0 & \partial y & 0 & \partial x & \partial z & 0 \\ 0 & 0 & \partial z & 0 & \partial y & \partial x \end{bmatrix}}$$

  $\boldsymbol{D_e}$ označava diferencijalni operator koji koristimo za rješavanje prostornih problema teorije elastičnosti. $\boldsymbol{\partial x}$,$\boldsymbol{\partial y}$ i $\boldsymbol{\partial z}$ označavaju parijcalne derivacije po kartezijevim koordinatama. Naš kinematički operator $\boldsymbol{D_k}$ je ustvari transponirana matrica elastičnog diferencijalnog operatora. $\boldsymbol{D_k = D_e^T}$. U prvoj točki smo definirali **u=Nv** pa sa dodatnom definicijom iz druge točke $\boldsymbol{\epsilon = D_k u}$ zaključujemo sljedeće:
  $\boldsymbol{\epsilon = D_k u = D_k N v = B v \Rrightarrow B = D_k N}$
  Matrica $\boldsymbol{B}$ označava matricu ovisnosti deformacije u elementu i pomaka u čvorovima (drugi naziv je matrica funkcija oblika za deformacije).

- Relacija povezanosti naprazenja u elementu i pomaka u čvorovima:
  $\boldsymbol{\rho = D\epsilon}$
  $\boldsymbol{\epsilon = Bv}$
  zaključak: $\boldsymbol{\rho = DBv}$ Ovdje $\boldsymbol{D}$ označava matricu elastičnosti čiji članovi različiti od nule ovise o elastičnim konstantama materijala.

- Sva naprezanja na rubovima konačnog elementa treba zamijeniti sa statički ekvivalentnim silama u čvorovima, što nas navodi na sljedeću relaciju:
  $\boldsymbol{F = A\rho}$
  gdje je $\boldsymbol{A}$ matrica ovisnosti naprezanja i čvornih sila

- Zaključak svih navedenih relacija izvodi matricu krutosti $\boldsymbol{K}$:
  $\boldsymbol{F = A\rho}$
  $\boldsymbol{\rho = DBv}$

$$\boldsymbol{F = ADBv = Kv \Rightarrow K = ADB}$$

Kod izvođenja matrice krutosti nismo upotrebljavali relacije koje opisuju uvjete ravnoteže unutar elementa iz čega izazi da takvi uvjeti u općenitom slučaju ne trebaju biti zadovoljeni. Također, u općem slučaju, konstruirana matrica krutosti nije simetrična. Pretpostavimo suprotno, $\boldsymbol{K = K^\intercal}$

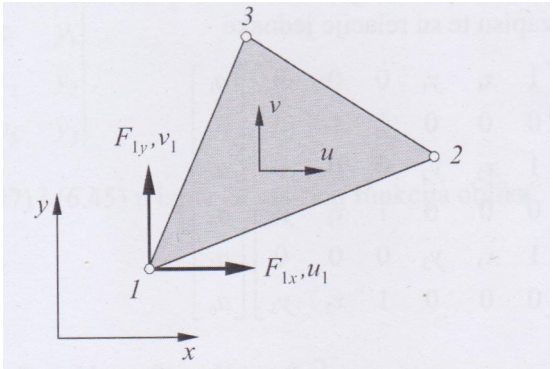
$\Rightarrow \boldsymbol{ADB = (ADB)^\intercal}$
$\Rightarrow \boldsymbol{A = B^\intercal}$ (matrica $\boldsymbol{D}$ je simetrična)

tvrdnja $\boldsymbol{A = B^\intercal}$ ne vrijedi u općem slučaju što nas navodi na kontradikciju.

Problem leži u tome što iz direktne formulacije nije moguće doći do potrebnih rubnih uvjeta, već je za te informacije potrebno fizikalno promatrati problem. Čim je složenije opterećenje (problemi s početnim deformacijama, stabilnosti i vibracije) nemoguće je primjeniti fizikalno promatranje, što nas navodi na zaključak da je direktnu metodu moguće primijeniti za izvođenje matrice krutosti samo kod jednostavnijih konačnih elemenata. Direktna metoda zbog svoje simplističnosti pridonosi razumijevanju metode konačnih elemenata i to joj je definitivno veliki plus u odnosu na ostale metode.

Matrica krutosti za osnovni trokutni element

Osnovni konačni element koji smo koristili u ovom radu (za kreiranje mreže) je bio trokut. Izabrani GMSH algoritmi su kreirali mrežu trokuta, a onda smo iz te mreže rekombinacijom više trokuta stvarali četverokutne konačne elemente i mrežu koja je većim dijelom bila četverokutna. U ovom potpoglavlju ćemo upotrijebiti znanje navedeno u prethodnom potpoglavlju i direktnom metodom izvesti matricu krutosti za trokutasti konačni element.



slika 1.

Definirat ćemo trokut kao konačni element s tri čvora i šest stupnjeva slobode, proizvoljne orijentacije u koordinatnom sustavu x,y (slika (1)). Uz pomoć linearnih polinoma možemo opisati polje pomaka:

$$u = a_1 + a_2 x + a_3 y$$

$$v = a_4 + a_5 x + a_6 y$$

To možemo zapisati i matrično:

$$\begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}$$

i simbolički:

$$u = \alpha a$$

simboli predstavljaju sljedeće vrijednosti:

$$u^{\mathsf{T}} = \begin{bmatrix} u & v \end{bmatrix}$$

$$\alpha = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix}$$

$$a^{\mathsf{T}} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{bmatrix}$$

Vidimo da je broj nepoznatih parametara istovrstan broju stupnjeva slobode konačnog elementa. Kod našeg konačnog elementa imamo tri čvora koji su određeni $x = x_i$ i $y = y_i$ koordinatama te komponente pomaka $u = u_i$ $v = v_i$ gdje je i=1,2,3.

Ako uvrstimo tu činjenicu u gornju jednadžbu dobivamo sljedeće:

$$u_1 = a_1 + a_2x_1 + a_3y_1$$
$$v_1 = a_4 + a_5x_1 + a_6y_1$$
$$u_2 = a_1 + a_2x_2 + a_3y_2$$
$$v_2 = a_4 + a_5x_2 + a_6y_2$$
$$u_3 = a_1 + a_2x_3 + a_3y_3$$
$$v_3 = a_4 + a_5x_3 + a_6y_3$$

Tj. u matričnom zapisu:

$$\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 1 & x_2 & y_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ 1 & x_3 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}$$

skraćeno: $v = Ca$

Ovdje je v vektor pomaka u čvorovima elementa:

$$v^\intercal = \begin{bmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 \end{bmatrix}$$

matrica C predstavlja izraz:

$$C = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 1 & x_2 & y_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ 1 & x_3 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_3 & y_3 \end{bmatrix}$$

Vektor a ,koji predstavlja nepoznate paramatre, dobije se iz relacije $a = C^{-1}v$, gdje je inverzna matrica C jednaka:

$$C^{-1} = \frac{1}{2A} \begin{bmatrix} x_2y_3 - x_3y_2 & 0 & x_3y_1 - x_1y_3 & 0 & x_1y_2 - x_2y_1 & 0 \\ y_2 - y_3 & 0 & y_3 - y_2 & 0 & y_1 - y_2 & 0 \\ x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 & 0 \\ 0 & x_2y_3 - x_3y_2 & 0 & x_3y_1 - x_1y_3 & 0 & x_1y_2 - x_2y_1 \\ 0 & y_2 - y_3 & 0 & y_3 - y_1 & 0 & y_1 - y_2 \\ 0 & x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 \end{bmatrix}$$

A ovdje predstavlja površinu trokutnog elementa koja se može prikazati pomoću sljedećeg izraza:

$$A = \frac{1}{2}\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}$$

Sada definirajmo matricu funkcija N kao izraz $N = \alpha C^{-1}$.

Traži se:

$$N = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix}$$

gdje svaka od funkcija unutar matrice N ima sljedeći oblik:

$$N_1 = \frac{1}{2A}[x_2y_3 - x_3y_2 + x(y_2 - y_3) + y(x_3 - x_2)]$$

$$N_2 = \frac{1}{2A}[x_3y_1 - x_1y_3 + x(y_3 - y_1) + y(x_1 - x_3)]$$

$$N_3 = \frac{1}{2A}[x_1y_2 - x_2y_1 + x(y_1 - y_2) + y(x_2 - x_1)]$$

Ovdje koristi uvesti supstituciju u ovisnosti o koordinatama svakog čvora(1.1):

$$\alpha_1 = x_2y_3 - x_3y_2,\ \alpha_2 = x_3y_1 - x_1y_3,\ \alpha_3 = x_1y_2 - x_2y_1$$

$$\beta_1 = y_2 - y_3,\ \beta_2 = y_3 - y_1,\ \beta_3 = y_1 - y_2$$

$$\gamma_1 = x_3 - x_2,\ \gamma_2 = x_1 - x_3,\ \gamma_3 = x_2 - x_1$$

Te supstitucije smo uveli radi mogućnosti kraćeg zapisa:

$$N_i = \frac{1}{2A}(\alpha_i + x\beta_i + y\gamma_i), i = 1, 2, 3$$

Važno je uočiti činjenicu da je raspodjela funkcija oblika po samom konačnom elementu definitivno linearna.

Sada smo postigli da se pomaci u konačnom elementu mogu prikazati kao sljedeći izraz u ovisnosti o pomacima u čvorovima:

$$u = Nv$$

što se može rastaviti na sljedeći oblik (uzimajući u obzir ranije definirane vektore *u* i *v*):

$$u = N_1u_1 + N_2u_2 + N_3u_3$$

$$v = N_1v_1 + N_2v_2 + N_3v_3$$

Podsjetimo se kinematičkog diferencijalnog operatora D_k i matrice opisa raspodjele deformacija B, pojmova opisanih u prethodnom potpoglavlju i relacije koja ih povezuje:
$B = D_kN$
Ako uvrstimo kinematički operator i pomnožimo ga s matricom N dobijemo da je matrica B sljedeći izraz:

$$B = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} \end{bmatrix}$$

Kako smo ranije definirali funkcije N_1,N_2 i N_3, sada možemo izračunati parcijalne derivacije i uvrstiti ih u gornji izraz(1.2):

$$B = \frac{1}{2A}\begin{bmatrix} y_2 - y_3 & 0 & y_3 - y_1 & 0 & y_1 - y_2 & 0 \\ 0 & x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 \\ x_3 - x_2 & y_2 - y_3 & x_1 - x_3 & y_3 - y_1 & x_2 - x_1 & y_1 - y_2 \end{bmatrix}$$

Korištenjem ranije definirane supstitucije(1.1) konačno dobijemo(1.3):

$$B = \frac{1}{2A}\begin{bmatrix} \beta_1 & 0 & \beta_2 & 0 & \beta_3 & 0 \\ 0 & \gamma_1 & 0 & \gamma_2 & 0 & \gamma_3 \\ \gamma_1 & \beta_1 & \gamma_2 & \beta_2 & \gamma_3 & \beta_3 \end{bmatrix}$$

Vidimo iz izraza (1.2) i (1.3) da je polje deformacije konstantno iz čega slijedi da je i polje naprezanja također konstantno. Ako pretpostavimo da imamo ravninsko stanje naprezanja matricu elastičnosti D možemo definirati na sljedeći način:

$$D = \frac{E}{1 - v^2}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix}$$

Unutar ovog izraza E predstavlja Youngov model elastičnosti dok je v Poissonov koeficijent.

Da bismo dobili matricu A potrebno je komponente naprezanja duž rubova opisati silama. U tom slučaju sila u npr. prvom čvoru trokutastog konačnog elementa rastavlja se na komponente ovog oblika:

$$F_{1x} = \frac{h}{2}[\rho_x(y_2 - y_3) + \tau_{xy}(x_3 - x_2)]$$

$$F_{1y} = \frac{h}{2}[\rho_y(x_3 - x_2) + \tau_{xy}(y_2 - y_3)]$$

Ovdje je h debljina trokutastog elementa, ρ_x i ρ_y su normalne komponente naprezanja u Kartezijevom koordinatnom sustavu dok je τ_{xy} posmična komponenta naprezanja u Kartezijevom koordinatnom sustavu.

Kada izračunamo sile u ostalim čvorovima, možemo napisati relaciju povezivanja komponenta čvornih sila s komponentama naprezanja:

$$\begin{bmatrix} F_{1x} \\ F_{1y} \\ F_{2x} \\ F_{2y} \\ F_{3x} \\ F_{3y} \end{bmatrix} = \frac{h}{2} \begin{bmatrix} y_2 - y_3 & 0 & x_3 - x_2 \\ 0 & x_3 - x_2 & y_2 - y_3 \\ y_3 - y_1 & 0 & x_1 - x_3 \\ 0 & x_1 - x_3 & y_3 - y_1 \\ y_1 - y_2 & 0 & x_2 - x_1 \\ 0 & x_2 - x_1 & y_1 - y_2 \end{bmatrix}$$

Ovdje primjećujemo da opet možemo unijeti supstituciju (1.1) pa dobivamo da je matrica A :

$$A = \frac{h}{2} \begin{bmatrix} \beta_1 & 0 & \gamma_1 \\ 0 & \gamma_1 & \beta_1 \\ \beta_2 & 0 & \gamma_2 \\ 0 & \gamma_2 & \beta_2 \\ \beta_3 & 0 & \gamma_3 \\ 0 & \gamma_3 & \beta_3 \end{bmatrix}$$

Sada imamo definirano matrice A, B i D i iz toga možemo dobiti matricu krutosti K. Sjetimo se izraza:

$$K = ADB$$

Množenjem matrica dobivamo da je matrica krutosti jednaka:

$$
\boldsymbol{K} = \begin{bmatrix}
\beta_1^2 + \frac{1}{2}(1-\nu)\gamma_1^2 & \nu\gamma_1\beta_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_1 & \nu\beta_1\gamma_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_1 & \beta_1\beta_2 + \frac{1}{2}(1-\nu)\gamma_1\gamma_2 & \nu\beta_1\gamma_2 + \frac{1}{2}(1-\nu)\gamma_1\beta_2 & \beta_1\beta_3 + \frac{1}{2}(1-\nu)\gamma_1\gamma_3 & \nu\beta_1\gamma_3 + \frac{1}{2}(1-\nu)\gamma_1\beta_3 \\
\nu\gamma_1\beta_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_1 & \gamma_1^2 + \frac{1}{2}(1-\nu)\beta_1^2 & \nu\beta_2\gamma_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_2 & \nu\beta_2\gamma_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_2 & \gamma_1\gamma_2 + \frac{1}{2}(1-\nu)\beta_1\beta_2 & \nu\beta_3\gamma_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_3 & \gamma_1\gamma_3 + \frac{1}{2}(1-\nu)\beta_1\beta_3 \\
\beta_1\beta_2 + \frac{1}{2}(1-\nu)\gamma_1\gamma_2 & \nu\beta_2\gamma_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_2 & \beta_2^2 + \frac{1}{2}(1-\nu)\gamma_2^2 & \nu\beta_2\gamma_2 + \frac{1}{2}(1-\nu)\beta_2\gamma_2 & \nu\beta_2\gamma_2 + \frac{1}{2}(1-\nu)\beta_2\gamma_2 & \beta_2\beta_3 + \frac{1}{2}(1-\nu)\gamma_2\gamma_3 & \nu\beta_2\gamma_3 + \frac{1}{2}(1-\nu)\beta_2\gamma_3 \\
\nu\beta_1\gamma_2 + \frac{1}{2}(1-\nu)\beta_2\gamma_1 & \gamma_1\gamma_2 + \frac{1}{2}(1-\nu)\beta_1\beta_2 & \nu\beta_2\gamma_2 + \frac{1}{2}(1-\nu)\beta_2\gamma_2 & \gamma_2^2 + \frac{1}{2}(1-\nu)\beta_2^2 & \nu\beta_3\gamma_2 + \frac{1}{2}(1-\nu)\beta_2\gamma_3 & \nu\beta_3\gamma_2 + \frac{1}{2}(1-\nu)\beta_2\gamma_3 & \gamma_2\gamma_3 + \frac{1}{2}(1-\nu)\beta_2\beta_3 \\
\beta_1\beta_3 + \frac{1}{2}(1-\nu)\gamma_1\gamma_3 & \nu\beta_3\gamma_1 + \frac{1}{2}(1-\nu)\beta_1\gamma_3 & \nu\beta_2\gamma_3 + \frac{1}{2}(1-\nu)\gamma_2\beta_3 & \nu\beta_3\gamma_2 + \frac{1}{2}(1-\nu)\beta_2\gamma_3 & \beta_3^2 + \frac{1}{2}(1-\nu)\gamma_3^2 & \nu\beta_3\gamma_3 + \frac{1}{2}(1-\nu)\beta_3\gamma_3 & \gamma_3^2 + \frac{1}{2}(1-\nu)\beta_3^2 \\
\nu\beta_1\gamma_3 + \frac{1}{2}(1-\nu)\beta_3\gamma_1 & \gamma_1\gamma_3 + \frac{1}{2}(1-\nu)\beta_1\beta_3 & \nu\beta_2\gamma_3 + \frac{1}{2}(1-\nu)\gamma_2\beta_3 & \nu\beta_2\gamma_3 + \frac{1}{2}(1-\nu)\beta_2\beta_3 & \gamma_2\gamma_3 + \frac{1}{2}(1-\nu)\beta_2\beta_3 & \nu\beta_3\gamma_3 + \frac{1}{2}(1-\nu)\beta_3\gamma_3 & \gamma_3^2 + \frac{1}{2}(1-\nu)\beta_3^2
\end{bmatrix}
$$

1.3 Osnovni pravokutni element

Unutar ovog rada, nakon rekombinacije mreže trokuta dobivamo većim dijelom kvadrate, a manjim pravokutnike kao elemente mreže. Poradi toga, osvrnut ćemo se i na osnovni pravokutni element u našoj problematici. Analogija opisa pravokutnog konačnog elementa je veoma slična kao i raniji opis kojim smo opisali trokut. Koristimo nepotpune polinome drugog stupnja da bi opisali polje pomaka:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & x & y & xy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x & y & xy \end{bmatrix}$$

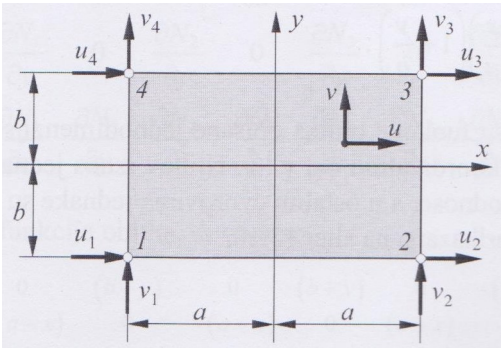
Uvrštavanjem i korištenjem već poznate relacije $N = \alpha C^{-1}$ dobivamo matricu funkcija oblika za pravokutni konačni element:

$$N = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix}$$

Funkcije oblika izgledaju na sljedeći način:

$$N_1 = \frac{1}{4}\left(1 - \frac{x}{a}\right)\left(1 - \frac{y}{b}\right)$$
$$N_2 = \frac{1}{4}\left(1 + \frac{x}{a}\right)\left(1 - \frac{y}{b}\right)$$
$$N_3 = \frac{1}{4}\left(1 + \frac{x}{a}\right)\left(1 + \frac{y}{b}\right)$$
$$N_4 = \frac{1}{4}\left(1 - \frac{x}{a}\right)\left(1 + \frac{y}{b}\right)$$

gdje su a i b polovina veličina stranice osnovnog pravokutnog elementa kojeg možemo vidjeti na slici(2).



slika 2.

Jednostavno je primjetiti da su funkcije opisane linearnim polinomima u smjeru koordinatnih osi x i y. Kako se svaka odnosi na jedan čvor osnovnog pravokutnog konačnog elementa njihov iznos jednak je jedan u čvorovima za koje se odnose, a u drugim čvorovima iznose nula. Već viđenom formulom $\boldsymbol{B} = \boldsymbol{D_k N}$ dobivamo matricu međusobne ovisnosti deformacije u elementu i pomaka u čvorovima:

$$\boldsymbol{B} = \frac{\mathbf{1}}{\mathbf{4}\boldsymbol{ab}} \begin{bmatrix} -(b-y) & 0 & (b-y) & 0 & (b+y) & 0 & -(b+y) & 0 \\ 0 & -(a-x) & 0 & -(a+x) & 0 & (a+x) & 0 & (a-x) \\ -(a-x) & -(b-y) & -(a+x) & (b-y) & (a+x) & (b+y) & (a-x) & -(b+y) \end{bmatrix}$$

Iz ovih podataka i ranijih podataka o matrici elastičnosti $\boldsymbol{D}$ možemo krenuti u izračun matrice krutosti $\boldsymbol{K}$:

$$\boldsymbol{K} = \int_{-a}^{a} \int_{-b}^{b} \boldsymbol{B}^{\intercal} \boldsymbol{D} \boldsymbol{B} \boldsymbol{dx} \boldsymbol{dy}$$

Konačnim izračunom dobivamo matricu krutosti za pravokutni konačni element (matrica je radi praktičnosti na sljedećoj stranici):

$$
\boldsymbol{K} = \frac{Eh}{12(1-\nu^2)}
\begin{bmatrix}
4\frac{b}{a} + (1-\nu)\frac{a}{b} & \frac{3}{2}(1+\nu) & -4\frac{b}{a} + (1-\nu)\frac{a}{b} & \frac{3}{2}(1+\nu) & -2\frac{b}{a} - (1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) & 2\frac{b}{a} - 2(1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) \\
\frac{3}{2}(1+\nu) & 4\frac{a}{b} + 2(1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & 2\frac{a}{b} - (1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & -2\frac{a}{b} - (1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & -4\frac{a}{b} + (1-\nu)\frac{b}{a} \\
-4\frac{b}{a} + (1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) & 4\frac{b}{a} + 2(1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) & 2\frac{b}{a} - 2(1-\nu)\frac{a}{b} & \frac{3}{2}(1+\nu) & -2\frac{b}{a} - (1-\nu)\frac{a}{b} & \frac{3}{2}(1+\nu) \\
-\frac{3}{2}(1+\nu) & 2\frac{a}{b} - (1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & 4\frac{a}{b} + 2(1-\nu)\frac{b}{a} & \frac{3}{2}(1+\nu) & -4\frac{a}{b} + (1-\nu)\frac{b}{a} & \frac{3}{2}(1+\nu) & 2\frac{a}{b} - 2(1-\nu)\frac{b}{a} \\
-2\frac{b}{a} - (1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) & 2\frac{b}{a} - 2(1-\nu)\frac{a}{b} & \frac{3}{2}(1+\nu) & 4\frac{b}{a} + 2(1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) & -4\frac{b}{a} + (1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) \\
-\frac{3}{2}(1+\nu) & -2\frac{a}{b} - (1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & -4\frac{a}{b} + (1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & 4\frac{a}{b} + 2(1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & 2\frac{a}{b} - (1-\nu)\frac{b}{a} \\
2\frac{b}{a} - 2(1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) & -2\frac{b}{a} - (1-\nu)\frac{a}{b} & -\frac{3}{2}(1+\nu) & -4\frac{b}{a} + (1-\nu)\frac{a}{b} & \frac{3}{2}(1+\nu) & 4\frac{b}{a} + 2(1-\nu)\frac{a}{b} & \frac{3}{2}(1+\nu) \\
-\frac{3}{2}(1+\nu) & -4\frac{a}{b} + (1-\nu)\frac{b}{a} & \frac{3}{2}(1+\nu) & -2\frac{a}{b} - (1-\nu)\frac{b}{a} & -\frac{3}{2}(1+\nu) & 2\frac{a}{b} - 2(1-\nu)\frac{b}{a} & \frac{3}{2}(1+\nu) & 4\frac{a}{b} + 2(1-\nu)\frac{b}{a}
\end{bmatrix}
$$

U promatranoj matrici krutosti $\boldsymbol{E}$ je Youngov modul elastičnosti,a $\boldsymbol{h}$ debljina konačnog elementa (ostale oznake su analogne kao i za matricu krutosti osnovnog trokutnog elementa). Primjećujemo da je ključan omjer unutar matrice krutosti $\boldsymbol{b/a}$ i $\boldsymbol{a/b}$. To nas navodi na zaključak da ako imamo element koji ima kombinaciju izrazito velike i male stranice omjer će biti veoma loš i utjecat će na globalnu matricu krutosti koja nastaje sumiranjem lokalnih matrica krutosti za osnovni element. Primjećujemo da s nekoliko takvih elemenata sama točnost metode dolazi u pitanje. Kako to kontrolirati? Saznat ćemo u sljedećem poglavlju. Prije toga ćemo napomenuti da neće biti dovoljno upotrijebiti samo osnovne elemente zbog toga što se u ovom radu susrećemo s kompliciranom geometrijom. Trebamo svaki konačni element koji upotrebljavamo transformirati u osnovne elemente te ih od tamo vratiti transformacijom nazad. Iz toga razloga trebamo uvesti zamjenu koordinata između koordinata Kartezijevog koordinatnog sustava i prirodnih koordinata (bezdimenzijskih koordinata). Sam prijelaz mora biti dobar te transformacija stabilna. Taj postupak ćemo opisati u sljedećem potpoglavlju.

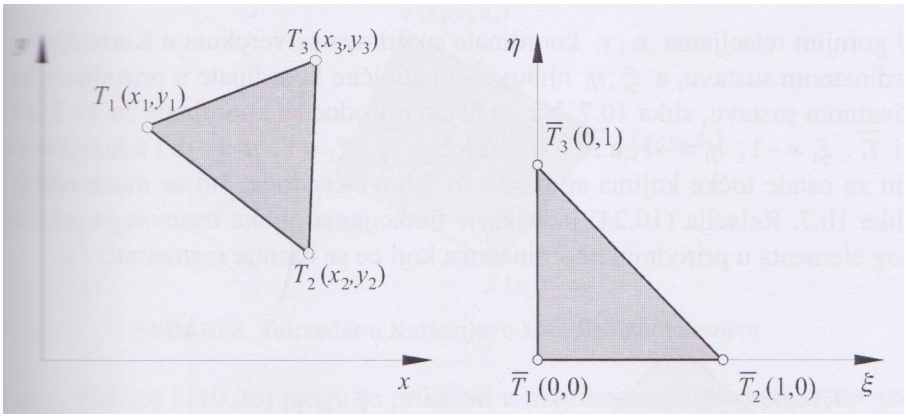
1.4 Jacobijeva matrica i prirodne koordinate

Kao što je gore istaknuto , Kartezijev koordinatni sustav nailazi relativno često na određene probleme u opisivanju konačnih elemenata te zato uvodimo lokalne bezdimenzijske koordinate(prirodne koordinate) koje su definirane geometrijskim veličinama elementa. Nove prirodne koordinate su uvedene kao funkcije Kartezijevih koordinata. Često se prirodni koordinatni sustavi mogu povezati različitim relacijama, a mi ćemo opisati $\boldsymbol{\zeta}$ $\boldsymbol{\eta}$ prirodni koordinatni sustav. Navedeni sustav se koristi jer je prilikom izračunavanja matrice krutosti jednostavnije integrirati po površini pravokutnog jediničnog trokuta (koji je osnovni konačni element s kojim zamjenjujemo svoje trokute u Kartezijevom koordinatnom sustavu) ovog koordinatnog sustava nego po proizvoljnim trokutnim površinama Kartezijevog koordinatnog sustava. Navedeni pravokutni trokut s jediničnim katetama u ovom prirodnom koordinatnom sustavu dobivamo preslikavanjem proizvoljnog trokuta iz Kartezijevog sustava x,y na sljedeći način:

$$\boldsymbol{x = x_1 + (x_2 - x_1)\zeta + (x_3 - x_1)\eta}$$

$$\boldsymbol{y = y_1 + (y_2 - y_1)\zeta + (y_3 - y_1)\eta}$$

Gdje su $\boldsymbol{x_1, y_1, x_2, y_2, x_3, y_3}$ koordinate vrhova trokuta u Kartezijevom koordinatnom sustavu (vrhovi su numerirani u smjeru suprotnom od gibanja kazaljke na satu).



slika 3.

Trokut u Kartezijevom koordinatnom sustavu te transformirani trokut možemo vidjeti na slici(3). Poveznica između elementarne površine u koordinatnom sustavu x,y i sustavu ζ,η jednaka je:

$$dxdy = detJd\zeta d\eta$$

Ovdje je $detJ$ determinanta Jacobijeve matrice:

$$detJ = \begin{vmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix}$$

Rezultat gornje determinante jednak je dvostrukoj površini trokuta u Kartezijevom koordinatnom sustavu. Kada se susretnemo s četverokutnim konačnim elementom također ga je moguće preslikati u kvadrat s jediničnim koordinatama u prirodnom sustavu ζ,η. Izrazi za preslikavanje su sljedeći:

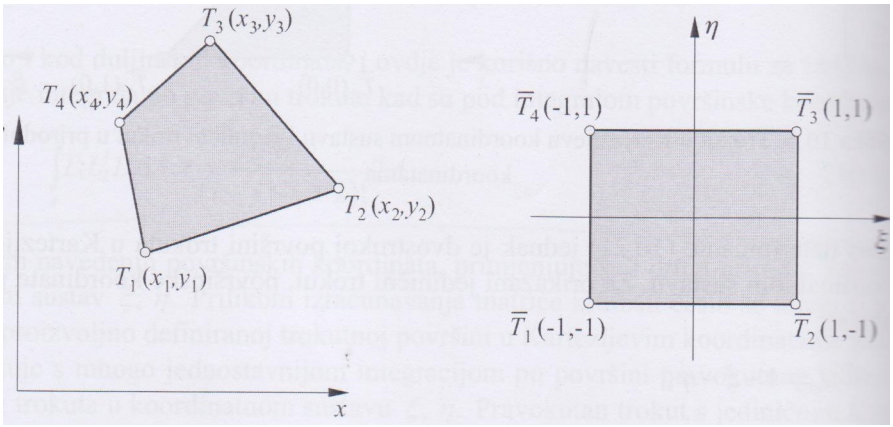
$$x = \sum_{i=1}^{4} N_i(\zeta,\eta)x_i$$

$$y = \sum_{i=1}^{4} N_i(\zeta,\eta)y_i$$

$$N_i(\zeta,\eta) = \frac{1}{4}(1+\zeta_i\zeta)(1+\eta_i\eta)$$

Gdje x_i i y_i predstavljaju koordinate vrhova četverokuta u Kartezijevom koordinatnom sustavu, ζ_i i η_i su njihove jedinične koordinate u prirodnom koordinatnom sustavu. Grafički

prikaz možemo vidjeti na slici(4)
.



slika 4.

1.5 Uvjeti kvalitete mreže

Ovdje ćemo spomenuti nekoliko uvjeta za kvalitetnu mrežu:

- **(Mjera kvalitete za četverokutni oblik)**
  Neka je $\boldsymbol{q}$ četverokutni konačni element u našoj mreži. Njegovi unutarnji kutevi su redom $\boldsymbol{\alpha_k}$, k=1,2,3,4. Definiramo kvalitetu $\boldsymbol{\eta(q)}$ četverokuta $\boldsymbol{q}$ kao sljedeći izraz:

  $$\boldsymbol{\eta(q) = \max(1 - \frac{2}{\pi} \max_k(|\frac{\pi}{2} - \alpha_k|), 0)}$$

  Ova mjera kvalitete iznosi 1 ako je element kvadrat, a 0 ako četverokut ima barem jedan kut koji je manji od 0 ili veći od $\boldsymbol{\pi}$. U ovom kontekstu važno nam je napomenuti mjeru koja označava prosječnu kvalitetu mreže $\bar{\boldsymbol{\eta}}$ te element najgore kvalitete u mreži $\boldsymbol{\eta_w}$. Te veličine su bitne za statističku ocjenu kvalitete cjelokupne mreže.

- **Index efikasnosti mreže**
  Bezdimenzionalna duljina vektora $\boldsymbol{y = b - a}$ unutar $\boldsymbol{R^3}$ u kontekstu ne uniformnog polja veličine $\boldsymbol{\delta(x)}$ definirana je s:

  $$\boldsymbol{l = \|y\| \int_0^1 \frac{1}{\delta(a + ty)} dt}$$

Optimalna mreža u kontekstu veličine je mreža za koju vrijedi da je bezdimenzionalna veličina svakog brida l_i jednaka jedan. Naravno, nemoguće je ostvariti takvu mrežu u praksi. Zato uvodimo sljedeću veličinu- index efikasnosti mreže τ je definiran na sljedeći način:

$$\tau[\%] = 100e^{\frac{1}{n_e}\sum_1^{n_e} d_i}$$

gdje je n_e broj bridova u mreži, a d_i razlika između duljine brida l_i i jedan:

ako je $l_i < 1 : d_i = l_i - 1$ ili ako je $l_i > 1 : d_i = \frac{1}{l_i} - 1$

Kvalitetni algoritmi za produkciju mreže proizvode mrežu trokuta za koju vrijedi da je τ otprilike 85% s bezdimenzionalnom veličinom svakog brida između $\frac{1}{\sqrt{2}} \leq l_i \leq \sqrt{2}$

- **Topološki kriterij za kvalitetnu mrežu**
  Kvalitetne mreže četverokuta su napravljene od mnogih četverokuta koji se obično dodiruju ortogonalno jedni s drugima. Optimalna topologija za vrh u takvoj mreži je vrh koji ima četiri četverokuta oko sebe. Parametar d_4 je postotak vrhova koji imaju četiri susjedna četverokuta. Ovdje u obračun nisu uzeti granični vrhovi. Konačno $d_{\max}$ je maksimalan broj četverokuta koji su susjedni jednom vrhu.

Zaključno, prelazak na prirodne koordinate (i preko njih na jedinične konačne elemente: jedinični pravokutni trokut i kvadrat) je veoma bitan jer se na taj način omogućuje puno lakša integracija na konačnim elementima potrebna za računanje matrice krutosti i površine elemenata te ostale proračune bitne za fizikalni element. U srcu svega toga jest Jacobijeva matrica koja nam govori o stabilnosti prelaska iz Kartezijevog u prirodni koordinatni sustav. Precizno odabiranje konačnih elemenata s naglaskom na kvalitetne omjere stranica jest veoma bitno radi zadržavanja preciznosti matrice krutosti za lokalni konačni element. To je bitno jer se preko sumiranja lokalnih matrica dobiva globalna matrica krutosti koja je ključna za preciznost i točnost same metode konačnih elemenata. U trećem poglavlju ćemo se baviti uvjetima za stranice i kuteve unutar konačnih elemenata, koje smo dobili za rješavanje konkretnog problema kreiranja mreže na brodskoj konstrukciji. Sljedeće poglavlje je posvećeno gmsh generatoru mreža i njegovim algoritmima za kreiranje kvalitetne mreže.

Poglavlje 2

GMSH

2.1 Uvod

Gmsh je 3D generator mreže konačnih elemenata s ugrađenim CAD softverom. Također GMSH je brz i jednostavan alat za kreiranje mreža s parametarskim unosima koji posjeduje naprednu mogućnost vizualizacije. Gmsh u svojoj srži pristupa problemima kroz 4 modula:

- postavljanje geometrije
- kreiranje mreže
- solver
- naknadna obrada podataka

Specifikacija ulaza u bilo koji od ovih modula izvršava se interaktivno preko grafičkog korisničkog sučelja te preko ASCII tekstualnih datoteka koristeći Gmshov vlastiti skriptni jezik (.geo datoteka) ili pomoću C++, C ili Python biblioteka. Velika prednost Gmsh softwarea je što je izdan s GNU javnom licencom što znači da je besplatan softver. Također od verzije 3.0 podržava mnoga svojstva geometrije temeljena na Open Cascade tehnologiji [9]. Gmsh je modificiran za integriranje sa SwiftCompom [10] (višenamjenski simulacijski softver koji brzo i jednostavno isporučuje točnost 3D prikaza konačnih elemenata). Modificirana verzija, nazvana Gmsh4SC je sastavljana na Composites design and Manufacturing Hub [11] (cdmHUB, https://cdmhub.org/) - platformi koja se bavi simuliranjem proizvodnje i raznovrsnih ideja. Osim toga, potpuno integrirano Matlab Gui Gmsh [12] sučelje je jednostavno za korištenje i dostupno je s FEATool Multiphysics [13] (simulacija fizičkih oblika i konačnih elemenata za Matlab).

U sljedećim potpoglavljima bavit ćemo se korištenim geometrijskim elementima i algoritmima za kreiranje mreže unutar GMSH-a.

2.2 Geometrija

U ovom potpoglavlju ćemo kratko opisati geometrijske elemente koje smo koristili unutar rada i kodiranja, a koji pripremaju teren za kreiranje mreže konačnih elemenata. Osnovna baza koju sam koristio u gmsh programiranju(unutar .geo datoteke) je sljedeća: kreirali smo točke u euklidskom prostoru koje smo zatim pridružili Spline krivuljama, te krivulje linijskim petljama. Linijske petlje su stvorile zatvoreni prostor na kojem sam mogao definirati površinu. Pobliže, koristili smo sljedeće naredbe:

- ***Point*(*expression*) = (*expression*, *expression*, *expression* <, *expression* >);** Ova naredba stvara osnovnu točku. Izraz unutar zagrada je identifikacijski broj točke, a prva tri izraza s desne strane daju X, Y i Z koordinate točke u trodimenzionalnom Euklidskom prostoru.Zadnji izraz koji je optimalan postavlja veličinu elementa mreže u toj točki.

- ***Spline*(*expression*) = (*expression* − *list*);** Naredba kreira Spline krivulju koja prolazi kroz sve zadane točke u desnom dijelu izraza. U lijevoj zagradi je identifikacijski broj Spline krivulje. Točke su zadane svojim identifikacijskim brojevima.

- ***LineLoop*(*expression*) = (*expression* − *list*);** Kreira usmjerenu linijsku petlju. Lijeva zagrada sadrži identifikacijski broj linijske petlje. Desna strana sadži identifikacijske brojeve svih elementarnih linija koje tvore linijsku petlju (u našem slučaju to su Spline krivulje). Linijska petlja(line loop) mora biti zatvorena petlja, a elementarne linije pravilno uređene i orijentirane. Ako je kojim slučajem orijentacija ispravna, ali redosljed neispravan Gmsh će sam napraviti ispravan redosljed i kreirati zatvorenu petlju. Linijska petlja se stvara kao temelj za kreiranje površina.

- ***PlaneSurface*(*expression*) = (*expression* − *list*);** Izraz kreira beskonačnu 2D ravninu na kojoj je definirana površina uređena s zadanim linijskim petljama čije identifikatore možemo naći u desnom dijelu izraza. Lijeva zagrada sadržava identifikacijski broj površine koju zadajemo. U desnoj zagradi prva linijska petlja definira vanjske granice površine,a sve ostale ostale linijske petlje definiraju eventualne rupe na zadanoj površini. Petlja koja definira rupu na površini ne bi smjela imati nikakve dodirne linije s vanjskom petljom. Isto tako petlja koja definira rupu na površini ne smije imati dodirnih linija s nekom drugom petljom koja eventualno definira rupu na istoj površini.

2.3 GMSH algoritmi

Unutar ovoga rada kreirali smo mrežu na dijelu brodske konstrukcije koristeći sljedeće gmsh algoritme: MeshAdapt, Automatic, Delaunay, Frontal, BAMG, DelQuad [2]. U ranijim izdanjima Gmsh generatora (u vrijeme pisanja ovog rada u opticaju je verzija 3.0) Gmsh je omogućavao upotrebu tri 2D nestrukturirana algoritma: MeshAdapt, Frontal ,Delaunay. Valja napomenuti da je četvrti algoritam, Automatic, nastao kao svojevrsna kombinacija Delaunay i MeshAdapt algoritma. U ranijim verzijama Gmsh-a (prije verzije 2.8) Automatic algoritam je tražio najbolji algoritam za svaku posebnu površinu na modelu kojeg promatramo. Poslije 2.8 izdanja , Automatic koristi Delaunay za površine nastale naredbom *plane surface* te MeshAdapt algoritam za sve ostale površine. U kasnijim izdanjima dodani su još drugi algoritmi kao što su Delquad i BAMG (osnovan na radu Bidimensional Anisotropic Mesh Generator koji je dio FreeFem++ softwarea). Bitno je napomenuti da su svi navedeni algoritmi triangularni te stvaraju mrežu trokutastih konačnih elemenata. Algoritam koji se automatski koristi upotrebom naredbe "Recombine Surface" te kreira četverokutnu mrežu- naziva se Blossom. Osnovna tri algoritma (MeshAdapt, Frontal, Delaunay) imaju svoje prednosti i mane. Za sva tri algoritma prvi korak u kreiranju mreže je sljedeći: automatski se stvara Delaunay mreža koja sadrži sve čvorove jednodimenzionalne mreže, pri tome se koristi "podijeli i vladaj" algoritam. Rubni dijelovi koji nedostaju se obnavljaju upotrebom "edge swap" algoritmom. Nakon ovih inicijalnih koraka kreće sama upotreba algoritama i stvaranje finalne mreže:

- MeshAdapt algoritam se temelji na lokalnim promjenama unutar mreže. Dugački rubni dijelove se razbijaju na manje, kratki rubni dijelovi se uništavaju, a dolazi do zamjene rubnih dijelova ako se time postiže bolja geometrijska konfiguracija.

- Delaunay algoritam je inspiriran radom GAMMA grupe na INRIA (inventors for the digital world) web stranici. Nove točke se ubacuju uzastopce u središte kružnice opisane elementu i to takve koja ima najveći radijus. Mreža se zatim ponovno spaja i kreira pomoću anizotropnog Delaunay kriterija.

- Frontal algoritam je inspiriran radom S.Rabaya. Koristi se kada je važna visoka kvaliteta konačnog elementa.

Algoritme možemo ocijeniti u tri kategorije: robusnost, izvedba i kvaliteta konačnog elementa kreiranog tim algoritmom. Najbolji u kategoriji robusnosti je MeshAdapt, Delaunay pa Frontal. Delaunay je najbolji u izvedbi, Frontal drugi te MeshAdapt najgori. Što se tiče kvalitete konačnog elementa tu je najbolji Frontal, a ostala dva su podjednako kvalitetna. Ocjene bi mogli objasniti na ovaj način: za veoma složene zakrivljene površine najbolje je upotrijebiti MeshAdapt, za velike mreže ravnih površina potrebno je upotri-

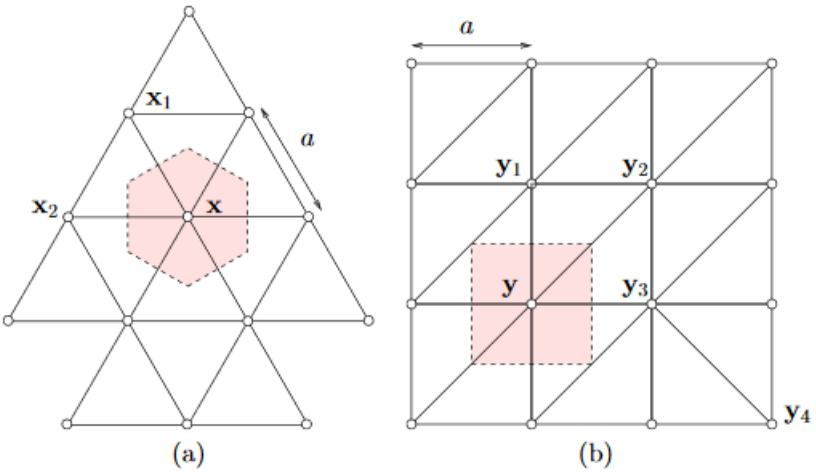
jebiti Delaunay zbog njegove brzine. Ako u zadatku tražimo kvalitetu elemenata mreže upotrijebit ćemo Frontal.

Kratko ćemo se osvrnuti i na dva algoritma za 3D probleme. Delaunay za 3D slučajeve je podijeljen u dva odvojena koraka. Prvo, inicijalna mreža nastaje spajanjem svih volumena u modelu- to se izvodi pomoću H. Tetgenovog algoritma. Zatim se primjenjuje trodimenzionalna verzija 2D Delaunay algoritma opisanog iznad. 3D Frontal algoritam koristi J. Schoeberl Netgenov algoritam kao svoju osnovu. Delaunay za 3D je narobusniji i najbrži algoritam i jedini koji podržava specifikaciju veličine elemenata. Međutim, ovaj algoritam će ponekad modificirati mrežu na određenoj površini i stoga nije prikladan za proizvodnju hibridnih strukturiranih/nestrukturiranih rešetki. U tom slučaju preferira se frontalni algoritam. Kvaliteta elemenata proizvedenih u oba algoritma je približno jednaka.

Najnovije metode za rekombinaciju (prelazak iz trokutastog oblika mreže u četverokutni) u svojoj osnovi imaju poznati algoritam teorije grafova- Blossom algoritam čije izvođenje traje u polinomijalnom vremenu. Ključno pitanje prilikom kreiranja četverokutne mreže je kako postaviti gore navedene algoritme da prilikom rekombinacije stvore što kvalitetniju mrežu četverokuta. Opisat ćemo metodu koja to ostvaruje. Ključan dio metode je činjenica da se za izračunavanje udaljenosti, prilikom kreiranja mreže, upotrebljava beskonačna norma umjesto klasične euklidske. U srži metode se nalazi tkz. *Frontal Delaunay* procedura unutar koje je poravnanje elemenata kontrolirano križnim poljem definiranim na domeni. Mreže konstruirane na ovaj način imaju čvorove poravnate sa smjerom križnog polja. Kada spojimo Frontal Delaunay metodu i upotrebu beskonačne norme s Blossom algoritmom za rekombinaciju stvara se mreža četverokuta odlične kvalitete.

Za sada postoje dva bitna pristupa u automatskom kreiranju mreže četverokuta. Kod direktnog pristupa četverokuti su kreirani odjednom i to najčešće ili upotrebom naprednih frontalnih tehnika ili tehnikama koje se baziraju na rešetkama. Suprotno od njih, neizravni pristup u kreiranju mreže četverokuta služi se početnom mrežom trokuta te zatim koristi tehnike spajanja da bi kreirao mrežu četverokuta iz mreže trokuta. Postoje još sofisticirane i napredne metode kreiranja mreže četverokuta i one su kombinacija frontalnih tehnika i tehnika spajanja i rekombinacije. Prednost neizravnih metoda je u tome što se oslanjaju na algoritme za kreiranje mreže trokuta koji su relativno jednostavni za implementaciju i imaju matematička svojstva koja omogućuju brzi razvoj robusnih mreža. U ranijim istraživanjima pokazano je da je moguće kreirati mrežu četverokuta počevši od bilo koje mreže trokuta koja sadrži parni broj trokuta. Jedna od verzija Blossom algoritma (Blossom-quad algoritam) omogućuje učinkovitu rekombinaciju bilo koje mreže trokuta.

Do nedavno su programeri, s raznim metodama kod kreiranja mreže trokuta, ciljali proizvesti što više jednakostraničnih trokuta unutar mreže. Nažalost, skup jednakostraničnih trokuta nije optimalna startna pozicija za dobivanje kvalitetne mreže četverokuta nakon rekombinacije. Da bi uvidjeli problematiku takvog pristupa s jednakostraničnim trokutima možemo promatrati sliku (5).



Slika 5.

Slika (5) lijevo prikazuje mrežu trokuta u $\boldsymbol{R^2}$ gdje su svi elementi jednakostranični trokutovi s duljinom stranice jednakom $\boldsymbol{a}$. Ova mreža se može smatrati savršenom u smislu optimalnih kriterija veličine i oblika. U ovom slučaju Voronoijeva ćelija za svaki vrh trokuta $\boldsymbol{x}$ je šesterokut površine $\boldsymbol{a\sqrt{3}/2}$, a broj točaka po jedinici površine iznosi $\boldsymbol{2/a\sqrt{3}}$. Ako usporedimo ove rezultate s mrežom pravokutnih trokutova koju vidimo na slici (5) desno vidimo da su Voronoijeve ćelije sada kvadrati površina $\boldsymbol{a^2}$. Popunjavanje $\boldsymbol{R^2}$ s jednakostraničnim trokutima zahtijeva $\boldsymbol{2/\sqrt{3}}$ puta više vrhova nego popunjavanje $\boldsymbol{R^2}$ s pravokutnim trokutima (to je oko 15 posto više). Iako se mreža četverokuta može postići iz bilo koje mreže trokuta primjećujemo da tradicionalne mreže trokuta s jednakostraničnim trokutima nisu dobra polazna točka iz razloga što sadrže 15 posto više vrhova nego mreže s pravokutnim trokutima. Cilj ovog potpoglavlja je dakle predstaviti metodu koja postavlja mrežu trokuta koji su opet sposobni izvršiti stabilnu i preciznu rekombinaciju u mrežu četverokuta. Ako ponovno promotrimo sliku (5) i upotrijebimo Euklidsku normu dolazimo do sljedeće činjenice: trokuti na slici (5) imaju bridove *različitih duljina*.

Ovi trokuti sadrže duži brid na 45 stupnjeva od osi i ostala dva kraća brida poravnata s osima. To i objašnjava zašto elementi na slici (5) desno imaju manji broj čvorova od elemenata na slici (5) lijevo. Problem u ovakvom pristupu što će prilikom rekombinacije dugački bridovi biti eliminirani i stvorit će se četverokut sa svim stranicama duljine a. Ovim pristupom gubimo razliku u broju čvorova koje bi smo postigli pravilnim korištenjem pravokutnih trokuta. Tom problemu ćemo doskočiti na način da stavimo u upotrebu beskonačnu normu:

Kod beskonačne norme vidimo da trokuti na slici (5) lijevo nisu više jednakostranični:

$$||x - x_2||_\infty = a$$
$$||x - x_1||_\infty = a\sqrt{2}$$

Upravo suprotno sada su trokuti sa slike (5) desno postali jednakostranični.

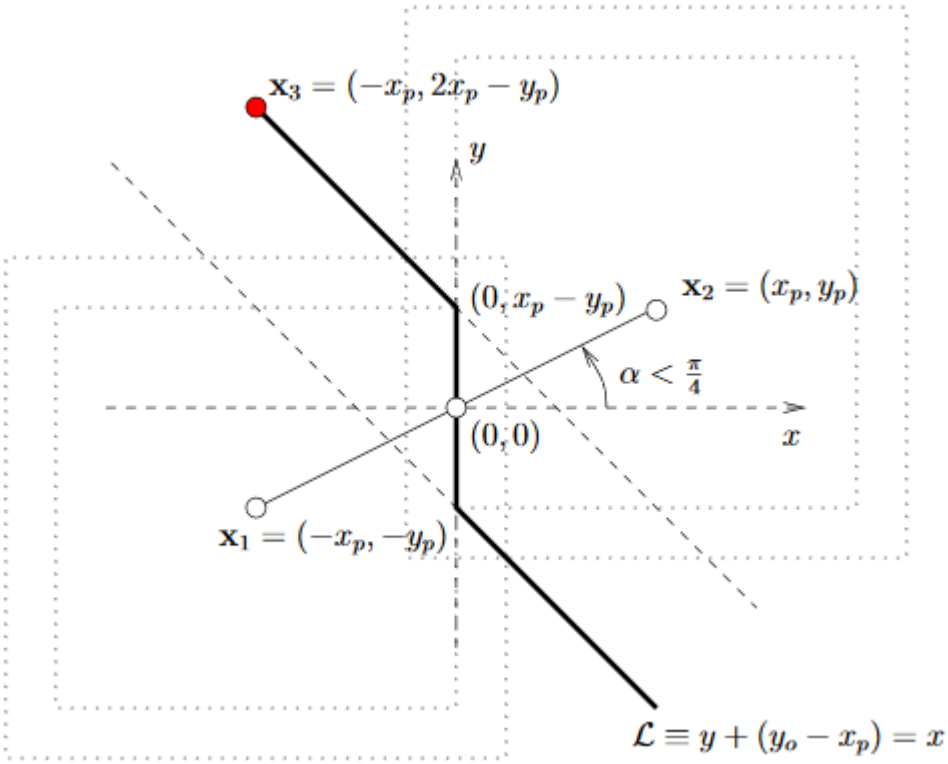
$$||y - y_1||_\infty = ||y - y_2||_\infty = a$$

S ovim saznanjem ključni sastojak naše metode , frontal-Delanauy algoritam se može prilagoditi radu s beskonačnim normama. S time će postići kreiranje mreže trokuta koja će dobar broj čvorova i proizvesti kvalitetne četverokute nakon rekombinacije. Na neki način smo spojili moderni pristup mreži preko pravokutnih trokuta te tradicionalan pristup preko jednakostraničnih trokuta. Trebamo razriješiti pitanje kako kreirati jednakostranični(pravokutni) trokut unutar beskonačne norme. Da bi to odgovorili prvo ćemo definirati beskonačnu normu i simetralu dužine unutar beskonačne norme. Unutar $\boldsymbol{R^2}$ ravnine udaljenost između dvije točke $\boldsymbol{A(x_1, y_1)}$ i $\boldsymbol{B(x_2, y_2)}$ putem beskonačne norme definirana je na sljedeći način:

$$||A - B||_\infty = \max(|x_2 - x_1|, |y_2 - y_1|)$$

Korisna stvar koja će nam trebati kasnije jest da je jedinični krug u beskonačnoj normi ustvari kvadrat s duljinom stranice jednakom 2. Simetralu dužine između točaka $\boldsymbol{P_1(x_p, y_p)}$ i $\boldsymbol{P_2(-x_p, -y_p)}$ definiramo kao skup točaka $\boldsymbol{X(x, y)}$ koje su jednako udaljene od točaka $\boldsymbol{A}$ i $\boldsymbol{B}$ (koristeći beskonačnu normu). Taj skup točaka je ustvari unija presjeka kružnica s centrima u A i B koje imaju jednaki radijus. Za beskonačnu normu simetrala dužine između $\boldsymbol{P_1}$ i $\boldsymbol{P_2}$ je generalno razlomljeni pravac. Taj skup možemo prikazati na sljedeći način: bez smanjenja općenitosti neka je $\boldsymbol{x_p \geq y_p}$:

$$\{P = (x, y), max(|x - x_p|, |y - y_p|) = max(|x + x_p|, |y + y_p|)\}$$



slika 6.

Generalni prikaz možemo vidjeti na slici (6). Iz ove pozicije kreiranja simetrale između $\boldsymbol{P_1}$ i $\boldsymbol{P_2}$ možemo doći do kreiranja jednakostraničnog trokuta u beskonačnoj normi

$$\boldsymbol{T(P_1, P_2, P_3)}$$

t.d. vrijedi:

$$\boldsymbol{\|P_1 - P_2\|_\infty = \|P_2 - P_3\|_\infty = \|P_1 - P_3\|_\infty}$$

Kako bismo dobili jednakostranični trokut iskoristimo naše zadane točke $\boldsymbol{P_1(x_p, y_p)}$ i $\boldsymbol{P_2(-x_p, -y_p)}$ te potražimo točku na simetrali udaljenu za $\boldsymbol{2x_p}$ od krajnjih točaka dužine $\boldsymbol{P_1P_2}$. To je točka $\boldsymbol{P_3(-x_p, 2x_p - y_p)}$ Postoje dva zanimljiva rubna slučaja ovakvog odabira točke $\boldsymbol{P_3}$. Prvi kada je dužina $\boldsymbol{P_1P_2}$ poravnata s x osi tj. $\boldsymbol{y_p = 0}$. U tom slučaju treći vrh trokuta je bilo koja točka između točaka $\boldsymbol{(-x_p, 2x_p)}$i $\boldsymbol{(x_p, 2x_p)}$. Drugi slučaj jest kada je $\boldsymbol{x_p = y_p}$. Tada je treći vrh trokuta točka $\boldsymbol{P_3(-x_p, x_p)}$. Na ovaj način smo dobili jednakostraničan trokut u beskonačnoj normi koji je u stvarnosti pravokutni trokut jednak polovini

kvadrata. Još jedan bitan pojam za Frontal-Delanauy algoritam jest pojam trokutu opisane kružnice(u našem slučaju trokutu opisani kvadrat). Središte trokutu opisane kružnice našeg trokuta

$$\boldsymbol{T(P_1, P_2, P_3)}$$

nalazi se u točki $\boldsymbol{(x_c, y_c)}$ te u beskonačnoj normi vrijedi:

$$\boldsymbol{||P_1 - S||_\infty = ||P_2 - S||_\infty = ||P_1 - S||_\infty}$$

U beskonačnoj normi naše središte S nalazi se u presjeku simetrala dužina u trokutu

$$\boldsymbol{(P_1P_2, P_3P_2, P_1P_3)}$$

Trokutu opisana kružnica je sada najmanji kvadrat koji ima središte u točki S i opisuje sva tri vrha trokuta. Radijus naše kružnice-kvadrata je udaljenost između središta S i svih ostalih vrhova u trokutu $\boldsymbol{T(P_1(x_1, y_1), P_2(x_2, y_2), P_3(x_3, y_3))}$. Definiran je na sljedeći način:

$$\boldsymbol{R_\infty = \frac{1}{2} max((max(x_1, x_2, x_3) - min(x_1, x_2, x_3)), (max(y_1, y_2, y_3) - min(y_1, y_2, y_3)))}$$

Važno je napomenuti da zbog beskonačne norme centar trokutu opisane kružnice-kvadrata nije jedinstven pa iz toga slijedi da i radijus nije jedinstven. Pozitivno je što je računanje radijusa i centara kružnice-kvadrata je veoma stabilna numerička operacija u beskonačnoj normi.

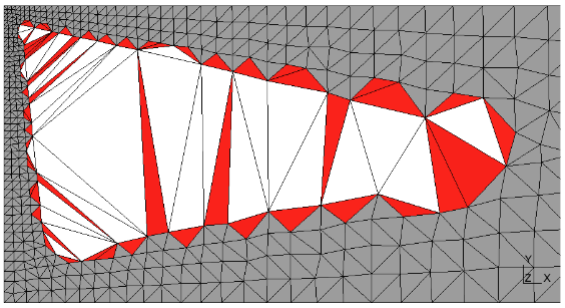
Za sada smo pokazali kako dobiti kvalitetan trokut pogodan za mrežnu rekombinaciju. Sljedećih nekoliko redaka ćemo posvetiti tome kako kreirati mrežu takvih trokuta koristeći Frontal-Delaunay algoritam na zadanoj površini. Frontal-Delaunay pristup omogućuje upotrebu najboljih stvari iz frontal i Delanaunay algoritma. Delanauy dio omogućuje da se mreža kvalitetno sačuva u svakom trenutku (štp je glavna odlika tog algoritma), ali ta činjenica tjera algoritam da prilikom svakog ubacivanja točke na površini dolazi do kreacija i brisanja većeg broja trokuta. Poradi tog razloga s tim dijelom se bavi frontal algoritam. Taj algoritam kreira skup bridova zatvorene površine na kojoj kreiramo mrežu (skup se naziva front). Kada se zatim kreira trokut unutar površine, taj trokut koristi jedan od zadanih bridova unutar front skupa kao svoju bazu. Ostala dva njegova brida se dodaju u front skup i postupak se ponavlja. Glavna prednost front pristupa je da on generira trokute i točke u isto vrijeme te je zbog toga moguće dobiti optimalne veličine konačnih elemenata. Algoritam traje dokle god u front skupu ima neiskorištenih bridova i dokle god nije cijela površina(domena) pokrivena s trokutima. Jedan od pristupa u frontal-Delanauy je sljedeći. Neka je S površina na kojoj definiramo mrežu, veličina mrežnog polja za točku u $\boldsymbol{\delta(u)}$, križno polje (cross field) $\boldsymbol{\theta(u)}$ veličina koja označava preferiranu orijentaciju mreže za svaku točku u unutar domene (u našem slučaju se radi o kutu), metrika $\boldsymbol{M(u)}$ koja se bavi

s udaljenošću točaka. Početna mreža τ_0 je kreirana u parametarskoj ravnini te sadrži samo točke nastale diskretizacijom granica površine na kojoj kreiramo mrežu (pristup prisutan u front algoritmu). Definirajmo mrežnu veličinu h_i (koja za udaljenosti koristi beskonačnu normu) koja je definirana za svaki trokut T_i od inicijalne mreže τ_0:

$$h_i = \frac{R_\infty(T_i, \theta(u))}{\delta(u)|det(M(u))|^{\frac{1}{4}}} = \frac{R_\infty(T_i, \theta(u))}{\delta'(u)}$$

Nakon toga, trokuti su podijeljeni u tri grupe:

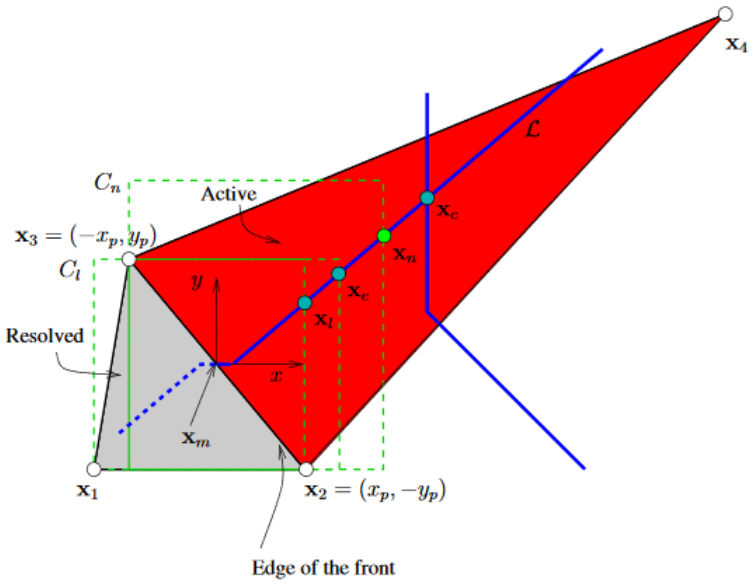
- trokut je riješen ako za njega vrijedi $h_i \leq h_{max}$
- trokut je aktivan ako za njega vrijedi $h_i > h_{max}$ i ako je jedan od njegova tri susjeda riješen ili je jedna od njegovih stranica na granici domene
- trokut je u fazi čekanja ako nije niti riješen niti aktivan



Slika 7. Crveno su aktivni trokuti, sivo riješeni, a bijeli čekajući.

Standardni izbor za h_{max} je $\frac{4}{3}$. Na slici (7) možemo vidjeti kako je algoritam podijelio trokute u tri kategorije.

Skup unutar front algoritma je definiran kao skup aktivnih trokuta(koji su sortirani po vrijednostima h_i). Bridovi unutar front skupa su definirani kao bridovi koji dijele aktivne od riješenih trokuta. Frontalni algoritam ubacuje novu točku tako da definira optimalni trokut s bridom koji odgovara najvećem aktivnom trokutu.



Slika 8. Ubacivanje optimalnog trokuta u mrežu frontal-Delanauy pristupom

Na slici (8) možemo primjetiti dužinu $\boldsymbol{x_2x_3}$ te pretpostavimo da odgovara najvećem aktivnom trokutu unutar mreže(crveni trokut). Za potrebe ovog primjere također pretpostavimo da je centralna točka koordinatnog sustava $\boldsymbol{x_m = 1/2(x_2, x_3)}$ te je lokalno križno polje definirano s kutem $\boldsymbol{\theta(x_m)}$ koji opisuje željenu orijentaciju.

Pozicija nove točke $\boldsymbol{x_n}$ na $\boldsymbol{L_\infty}$ simetrali od $\boldsymbol{x_2x_3}$ je odabrana tako da ispuni $\boldsymbol{\delta(x_m)}$ veličinu mrežnog polja za točku $\boldsymbol{x_m}$. Da bismo dobili trokut $\boldsymbol{T_i(x_2, x_3, x_n)}$ t.d. vrijedi $\boldsymbol{R_\infty(T_i, \theta) = \delta'(x_m)}$ trebamo smjestiti $\boldsymbol{x_n}$ na mjesto presjeka simetrale od dužine $\boldsymbol{x_2x_3}$ i kvadrata $\boldsymbol{C_n}$ koji prolazi točkama $\boldsymbol{x_2}$i $\boldsymbol{x_3}$ (kao što možemo vidjeti na slici (8)). Moramo paziti na sljedeće stvari:

- Nova točka ne smije biti smještena ispod $\boldsymbol{x_c}$ koja je centar aktivnog trokuta opisane kružnice-kvadrata (crveni trokut na našoj slici(8)) jer će to stvoriti trokut s premalim $\boldsymbol{x_nx_4}$ bridom.

- Nova točka ne smije biti smještena ispod točke $\boldsymbol{x_l}$, sjecišta između simetrije dužine $\boldsymbol{x_2x_3}$ i kvadrata $\boldsymbol{C_l}$ opisanog trokutu $\boldsymbol{T(x_1, x_2, x_3)}$. Ubacivanje točke unutar kvadrata $\boldsymbol{C_l}$ bi riješeni trokut napravila neispravnim po Delanauy kriterijima.

- Ako vrijedi $\boldsymbol{\delta'(x_m) = ||x_3 - x_2||_\infty}$ tada je najbolja točka $\boldsymbol{x_n = x_e}$. Ta točka odgovara najveće trokutu $\boldsymbol{T_i(x_e, x_2, x_3)}$ koji zadovoljava izraz $\boldsymbol{R_\infty = \delta'(x_m)}$

U ovom poglavlju smo opisali gmsh generator mreža, spomenuli sve algoritme koje on koristi te opisali metodu koja za kreiranje mreže trokuta podložne kasnijoj kvalitetnoj rekombinaciji koristi beskonačnu normu i najbolje od frontal i Delanauy algoritma. U

sljedećem poglavlju bavit ćemo se matematičkim uvjetima za kvalitetnu mrežu te konkretnim problemom kreiranja mreže na dijelu brodske konstrukcije.

Poglavlje 3

Praktični zadatak

3.1 Pygmsh

Cilj pygmsh biblioteke je kombinirati snagu Gmsh-a sa svestranošću Python-a i pružiti korisne apstrakcijske objekte iz Gmsh skriptnog jezika kako bi se lakše stvorile složene geometrije. U Gmsh-u, korisnik mora ručno unijeti jedinstveni ID za svaku točku, krivulju, volumen. To može postati veoma zahtjevno u trenucima kada se stvori puno entiteta i nije jasno koji su ID-ovi već u uporabi. Neke Gmsh naredbe čak stvaraju nove entitete i ”tiho” rezerviraju ID na taj način. Ova biblioteka pokušava zaobići to pružajući rutine u stilu add_point(x) funkcije te jednostavnih funkcija koje u nastavku mogu vratiti ID unesene točke x i obaviti potrebne operacije. Unutar ovog rada koristili smo pygmsh biblioteku kao pomoć u python programiranju te posebno kao pomoć u procjeni kvalitete dobivenih mreža korištenjem različitih algoritama.

3.2 Zadatak

Praktični zadatak tijekom izrade ovog diplomskog rada bio je kreirati mrežu konačnih elemenata na brodskoj konstrukciji što bliže sljedećim uvjetima:

- Koristiti četverokutne konačne elemente (četiri čvora u svakom kutu konačnog elementa), trokutne konačne elemente (tri čvora u svakom kutu konačnog elementa) koristiti iznimno u slučaju nužde.
- Omjeri najdulje i najkraće stranice četverokutnih konačnih elementa: Idealno 1:1; Prihvatljivo do 1:2; Iznimno do 1:4 (izbjegavati osim u slučaju nužde).
- Najmanji kut između bilo koje dvije susjedne stranice četverokutnog konačnog elementa: 60 stupnjeva.

- Omjeri ”kateta” trokutnih konačnih elementa: Idealno 1:1; Prihvatljivo do 1:2

Vidimo da postavljeni uvjeti doista dobro slijede ideju što preciznijeg dobivanja globalne matrice krutosti(spomenuto u prvoj cjelini) te približavanja oblika konačnih elemenata njihovim jediničnim verzijama u prirodnim koordinatama(također opisano u prvoj cjelini). Ako budemo slijedili uvjete što se tiče trokuta i omjera ”kateta” vidimo da bi kreiranje pravokutnih trokuta bilo veoma dobro rješenje i za uvjete i za postavljanje kvalitetnih uvjeta za rekombinaciju. Uočavamo da će zadovoljavanje ovih uvjeta također zadovoljiti i uvjete iz prve cjeline o kvalitetnoj mreži konačnih elemenata.

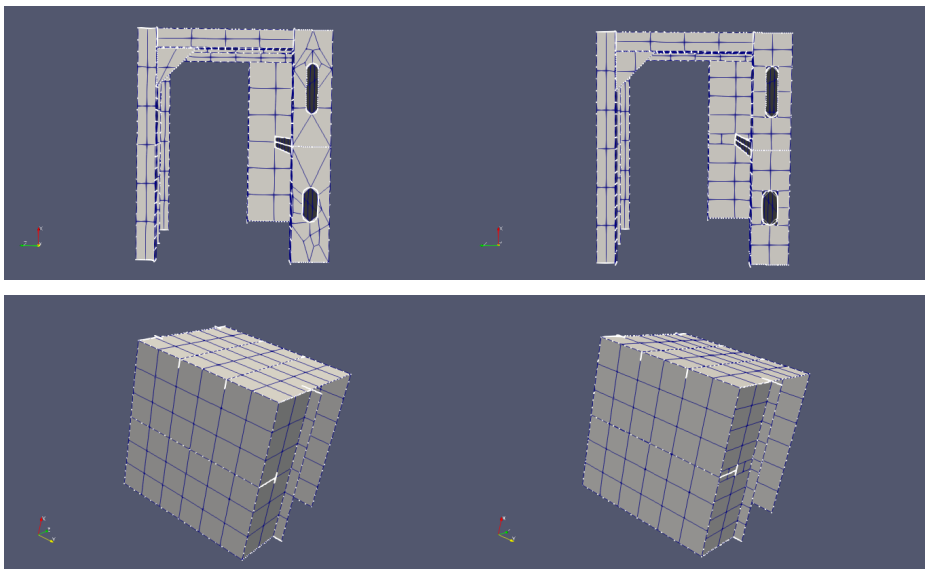
Pristup problemu je imao dvije glavne grane koje su imale isti python/pygmsh pristup završnoj obradi podataka. Prvi pristup je bio sljedeći: automatski je prebačena geometrija iz .stp file-a (u kojem je bila smještena brodska konstrukcija) unutar .geo file koji koristi gmsh generator za editiranje geometrije. Najveći problem su nam zadavale površine s prekidima (”rupama”) i površine s utorima na koje će se poslije nastavljati dijelovi brodske konstrukcije. Unutar python koda moralo se odgonetnuti kako prepoznati te površine. Iako se uspjela kreirati mreža na problematičnim dijelovima, ostalo je prostora za napredak u kvaliteti mreže te se upotrijebio drugi pristup. Kod drugog nadziranog pristupa, cilj je bio promijeniti .geo file i natjerati gmsh generator direktno da kreira kvalitetnu mrežu na problematičnim dijelovima. U nadziranom generiranju unutar Gmsh-ove geometrije koristili smo strukture opisane u drugoj cjelini(Geometrija) te smo dodali još kreiranje jednostavnih dužina (Line (expression) = (expression, expression)-izraz u lijevoj zagradi označava identifikacijski broj dužine, a u desnoj identifikacijski broj točaka- krajeva dužina). Python i pygmsh biblioteka je korištena kako bismo iz automatske (ili napisane-nadzirani pristup) geometrije, nakon kreiranja mreže, izvukli podatke o konačnim elementima i statistički ih opisali kroz histograme i grafove (posebno za svaki korišteni algoritam iz druge cjeline).

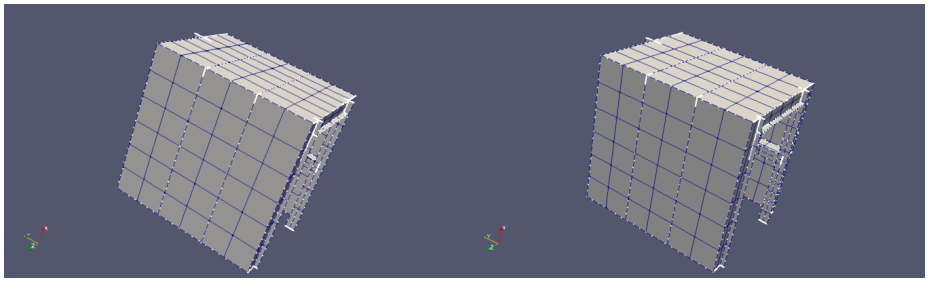
Skraćenu verziju automatskog koda u .geo file-u vidimo u sljedećem potpoglavlju koje sadrži samo kodove (Gmsh kod). Kod je ukupno imao 1776 linija i slijedio je upotrebu geometrijskih elemenata po jednostavnoj logici: $\boldsymbol{Point- > Spline- > LineLoop- > Surface}$ opisanu u drugoj cjelini. Unutar nadziranog pristupa popravljani su dijelovi koji se tiču površina s ”rupama” i utorima te je omogućeno kreiranje mreže po čitavom brodskom modelu i kasnija lakša rekombinacija (dijelove popravka možemo vidjeti unutar Gmsh kod2 u dodatku). I poslije automatskog i poslije nadziranog generiranja geometrije, podaci o geometriji su završili unutar tekstualne datoteke proba02.txt preko koje je krenula analiza mreže unutar pythona.

U dodatku je prikazan i dio koda (Python kod-unošenje geometrije,rekombinacija i spremanje mreže) kojim se ubacuje geometrija u python sustav, pronalaze se problematične površine s prekidima na sebi te se svi konačni elementi(bazične mreže trokuta) rekombiniraju. Također primjećujemo da se elementi mreže spremaju i mogu se vizualno prikazati u uspjeh.vtu datoteci. Veoma je važna naredba $\boldsymbol{geom.add\_raw\_code('Mesh.Algorithm = 8;')}$ kojom odabiremo algoritam s kojim ćemo kreirati početnu mrežu trokuta. Algori-

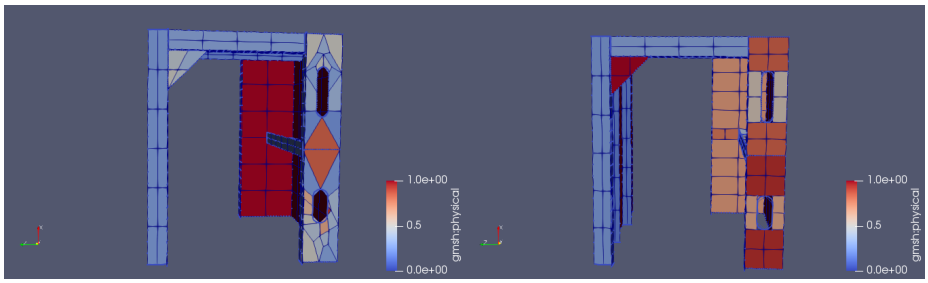
tam se bira rednim brojem, u ovom slučaju to je 8 i označava DelQuad algoritam. Cijeli taj koncept omogućuje pygmsh biblioteka. Nakon kreiranja mreže bitno je odrediti kvalitetu te iste mreže u ovisnosti o algoritmima te uvjetima kvalitete mreže koje smo naveli u ovom poglavlju. Za takve stvari koristit ćemo dodatne funkcije koje koriste liste čvorova konačnih elemenata te riječnike koji unutar sebe sadrže popis svih konačnih elemenata u mreži (u dodatku pod imenom Python-pomoćne funkcije za otkrivanje kvalitete elemenata mreže). Dodatne funkcije su korištene za izračune omjera stranica, izračun kuteva te podijelu na dobre i loše konačne elemente unutar mreže (u dodatku-Python podjela na dobre i loše elemente). Nakon spremanja svih vrijednosti potrebnih za kvalitetnu ocjenu mreže slijedi prikazivanje podataka preko histograma i grafova u kojim pobliže analiziramo svaki pojedini konačni element u mreži. Prije nego krenemo u analizu statistike pogledajmo prvo obojanu vizualizaciju naše mreže na brodskoj konstrukciji. S lijeve strane vidimo kreiranu mrežu nakon automatski generirane geometrije iz .geo file-a dok s desne strane vidimo kreiranu mrežu nakon nadziranog popravljanja i generiranja geometrije unutar .geo file-a. Obojenja unutar vizualizacije su ostvarena preko dodjeljivanja funkcijske vrijednosti svakoj ćeliji mreže unutar pythonovog koda. Svaki konačni element obojan je u skladu zadovoljenja naših kriterija o kvaliteti mreže. S lijeve strane možemo promatrati kvalitetu mreže dobivenu automatskim generiranjem, dok s desne strane primjećujemo kvalitetu mreže dobivene nadziranim preinakama geometrijskom file-a. Nadzirano kreiranje je upotrebljeno prvenstveno radi popravljanja problematičnih površina s prorezima i utorima.

- **Prikaz automatske i nadzirane mreže bez obojenja**

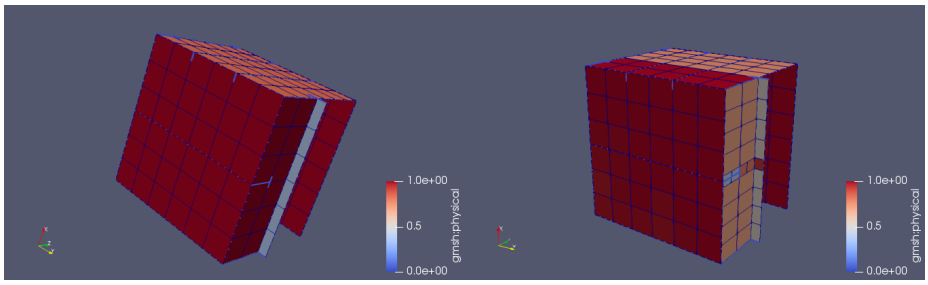




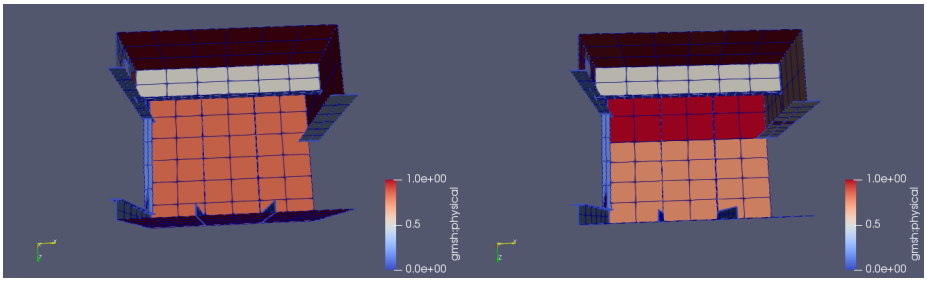
- **Kriteriji: omjer stranica**
  Kriteriji kvalitete omjera su poredani od tamno plave(loše) do tamno crvene boje (savršeno). Glavna problematika kvalitetnije mreže po pitanjima omjera stranica leži u rubnim površinama s pravokutnim konačnim elementima. (Lijevo su elementi dobiveni automatskim pristupom, desno nadziranim)



Na prvoj slici možemo primjetiti problematičnu površinu s ”rupama” na desnom dijelu brodske konstrukcije. Nadziranim pristupom smo ostvarili bolje rezultate.

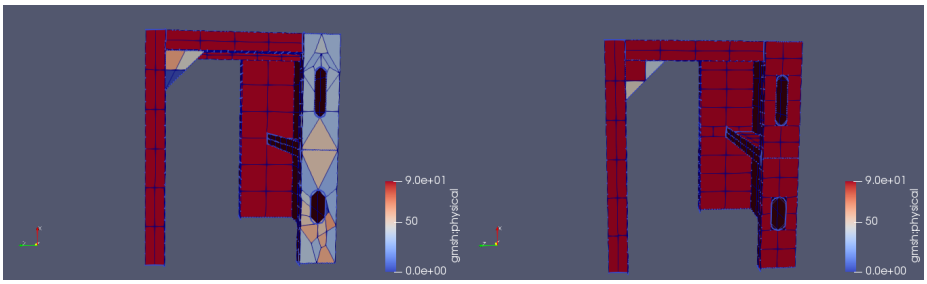


Druga problematična površina s T utorom prikazana je na drugoj slici. Ovdje smo kod nadziranog pristupa ”žrtvovali” nijansu kvalitete konačnih elemenata radi povezivanja krajnjih točaka T utora sa susjednom gornjom i desnom poprečnom površinom. Kvaliteta dijela gornje susjedne površine se poboljšala te je također desna poprečna površina postigla zadovoljavajuću kvalitetu.

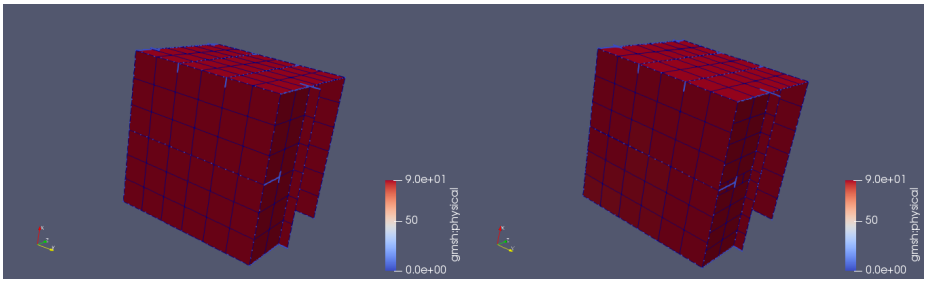


Na trećoj slici vidimo male razlike u kvaliteti u korist nadziranog pristupa.

- **Kriteriji: veličina najmanjeg kuta kod četverokutnih elemenata**
  U ovom kriteriju promatramo najmanji kut unutar svakog četverokutnog elementa. Opet je kvaliteta označena s bojama od tamno plave (loše) do tamno crvene (savršeno). (Lijevo su elementi dobiveni automatskim pristupom, desno nadziranim)



Vidimo da skoro svi četverokuti i kod automatskog i kod nadziranog pristupa zadovoljavaju naše postavljene uvjete. Razlika je jedino u četverokutima koji se nalaze na problematičnoj površini s ”rupama”. Nadziranimm pristupom geometriji i taj problem nestaje te dobivamo maksimalno ispunjenje uvjeta.



3.3 Histogrami i grafovi

Unutar ovog potpoglavlja prikazani su histogrami i grafovi dobivenih rezultata za kvalitetu mreže. Histogrami i grafovi će biti prikazani po algoritmima koje smo koristili. S lijeve strane su smješteni rezultati automatskog pristupa generiranju mreže, a s desne rezultati

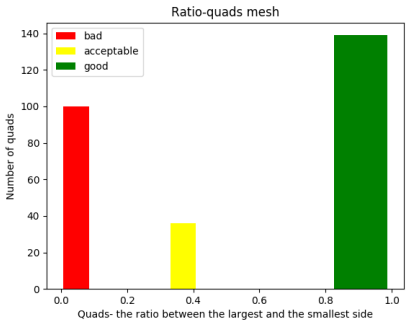
nadziranog pristupa generiranju mreže.

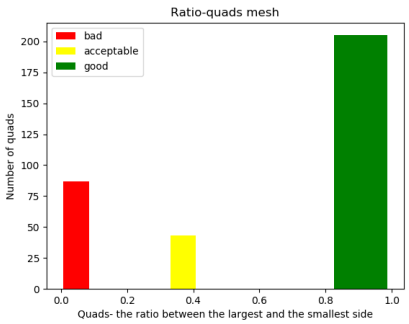
1) ***Koristiti četverokutne konačne elemente***
Kao što ćemo vidjeti iz histograma koji slijede uvjet je ispunjen. Kod svakog algoritma mreža ima preko 90% četverokutnih elemenata.

2) ***Omjeri najdulje i najkraće stranice četverokutnih konačnih elementa***

Kao što vidimo na ovom uvjetu ima dosta četverokuta koji ne zadovoljavaju kriterije. Možemo primjetiti (na slikama iznad) da su najveći problem pravokutnici na rubnim dijelovima i oni zahtijevaju najveću preinaku u nekom od sljedećih radova koji se nastavljaju na ovu temu. Primjetit ćemo da nadziranom obradom geometrije broj kvalitetnih četverokuta osjetno raste (iznimno kod BAMG algoritma), te nadovezujući radovi na ovaj projekt trebaju omogućiti automatsko postizanje sličnih rezultata. Crvena boja označava loše četverokute, žuta prihvatljive, a zelena veoma dobre. Lijevo su rezultati algoritama uz automatski pristup, desno uz nadzirani.

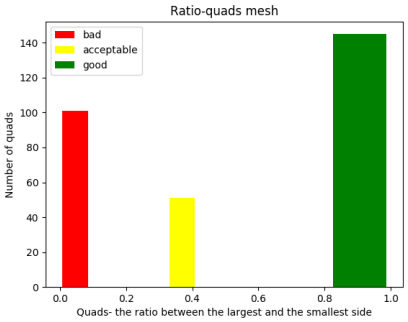
- **Mesh Adapt algoritam:**

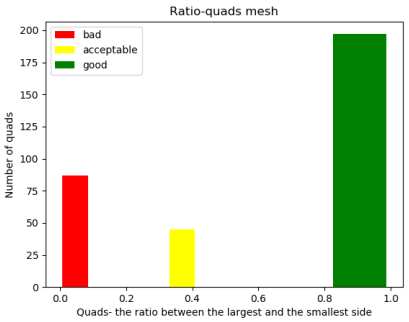




Vidimo pad u lošim četverokutima kada uz algoritme upotrijebimo nadzirani pristup. Pada i broj prihvatljivih, a raste broj veoma dobrih. U automatskom pristupu crveni stupac je približno svugdje jednak (drugi stupci se izrazitije mijenjaju jer su algoritmi različiti, a triangulacija u beskonačnoj normi nije jedinstvena). To upućuje da je potrebno koristiti nadzirani pristup za smanjivanje crvenog i žutog stupca.

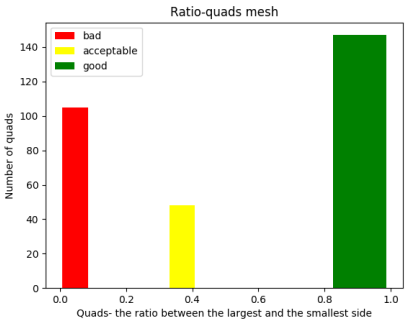
- **Mesh Automatic algoritam:**

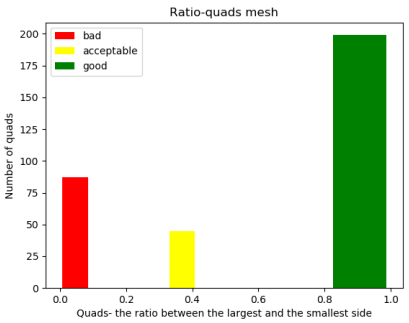




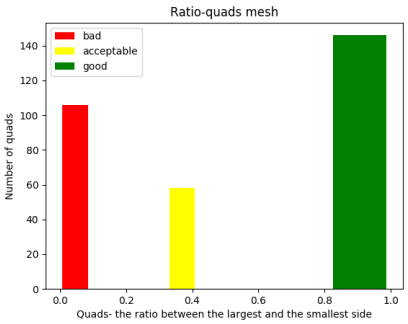
Adapt, Automatic i Frontal algoritmi su nadziranim pristupom dali dobre rezultate, otprilike jednake kvalitete.

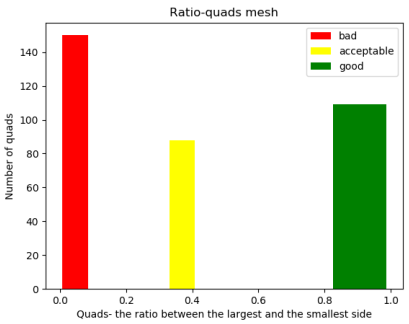
- **Mesh Frontal algoritam:**





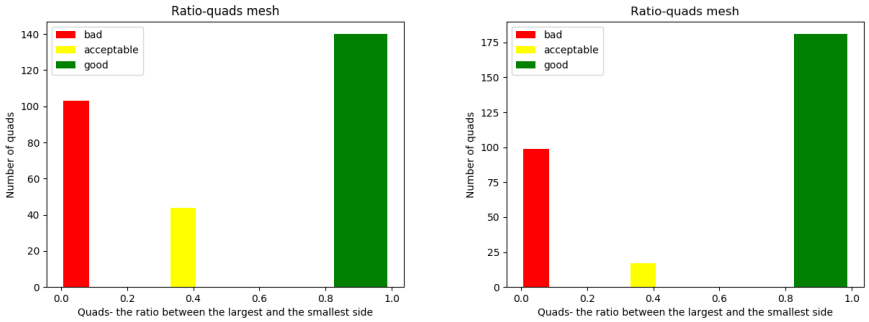
- **Mesh BAMG algoritam:**





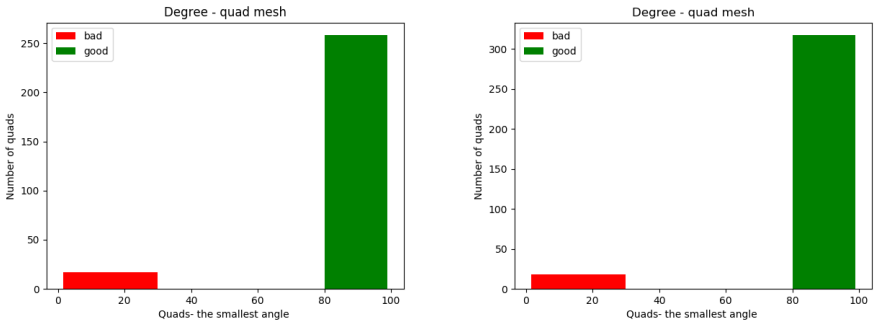
BAMG algoritam je jedini dao bolje rezultate ako se upotrijebio na automatski generiranoj mreži.

- **Mesh DelQuad algoritam:**



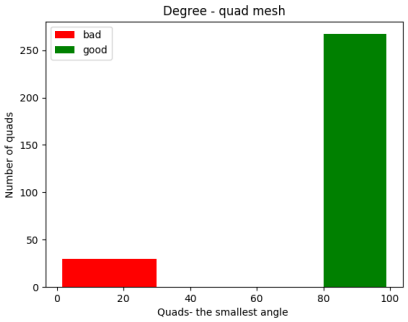
3) ***Najmanji kut između bilo koje dvije susjedne stranice četverokutnog konačnog elementa: 60 stupnjeva***
Kao što možemo vidjeti iz priloženih histograma, svi algoritmi ispunjavaju zadatak s uspjehom od preko 80%. Crvena boja označava loše četverokute, a zelena veoma dobre. Lijevo su rezultati algoritama uz automatski pristup, desno uz nadzirani.

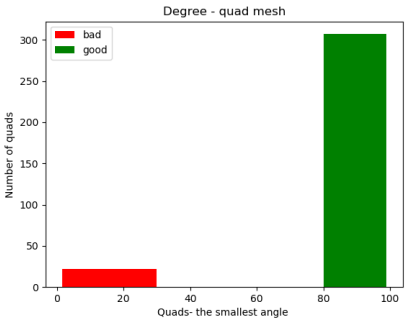
- **Mesh Adapt algoritam:**



Vidimo da je Mesh Adapt dao približno jednake rezultate i za automatski pristup i za nadzirani pristup. To nimalo ne čudi jer je on algoritam s najboljim rezultatima kod automatskog pristupa generiranja mreže.

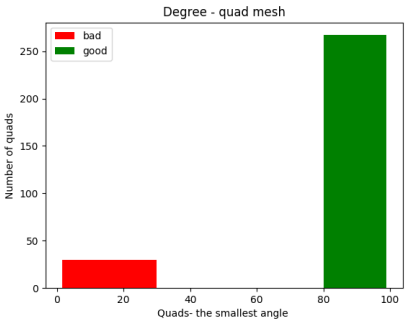
- **Mesh Automatic algoritam:**

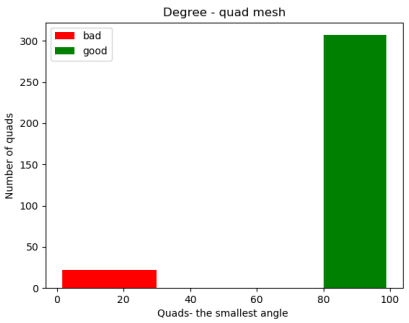




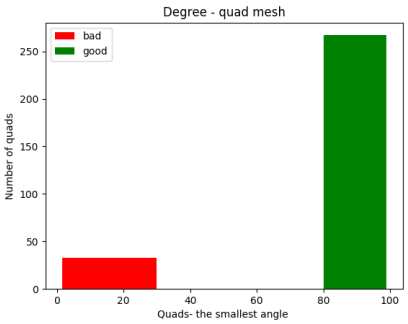
Nadzirani pristup daje otprilike dvostruko manje loših četverokuta kod Automatic i Delaunay algoritma.

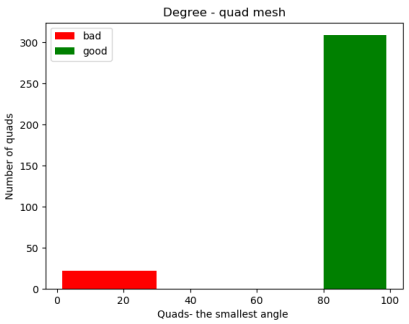
- **Mesh Delaunay algoritam:**





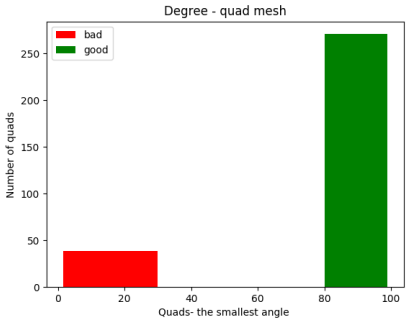
- **Mesh Frontal algoritam:**

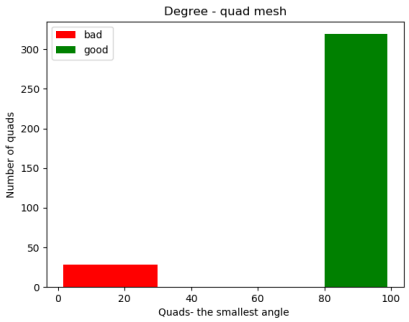




Frontal i BAMG algoritmi su otprilike jednake kvalitete i za automatski i za nadzirani pristup generiranju mreže.

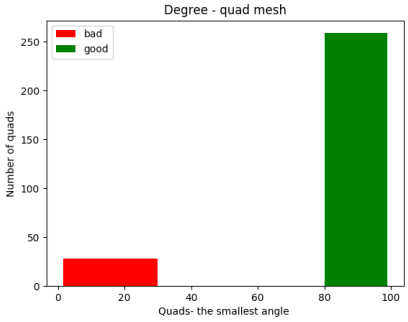
- **Mesh BAMG algoritam:**

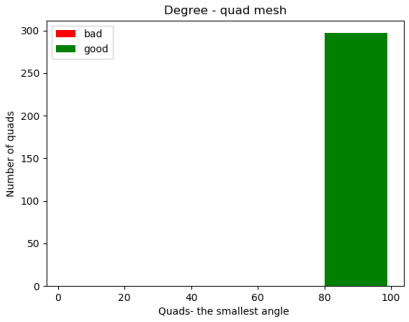




DelQuad algoritam kod nadziranog pristupa generiranju mreže daje 100% rezultat u korist kvalitete čime je najbolji rezultat u ovoj kategoriji kvalitete mreže.

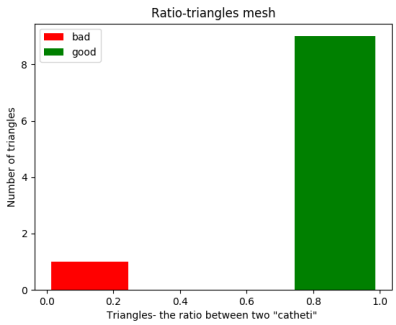
- **Mesh DelQuad algoritam:**





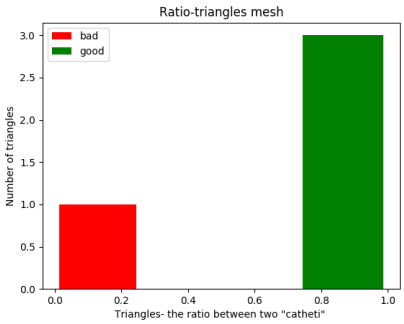
4) ***Omjeri 'kateta' trokutnih konačnih elementa: Idealno 1:1; Prihvatljivo do 1:2***
Što se tiče trokuta unutar mreže, kod većine algoritama ih ukupno ima veoma malo. Kako ih ima malo ne igraju veliku ulogu unutar same mreže, ali kod većine algoritama ih je više dobrih nego loših. Crvena boja označava loše četverokute, a zelena veoma dobre. Lijevo su rezultati algoritama uz automatski pristup, desno uz nadzirani.

- **Mesh Adapt algoritam:**

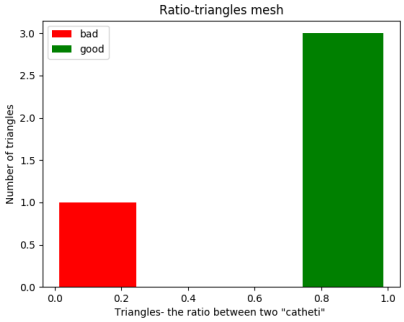


Odnos je u korist kvalitetnih trokuta kod Mesh Adapt algoritma (automatski pristup)koji kreira najveći broj trokuta unutar mreže.

- **Mesh Automatic algoritam:**

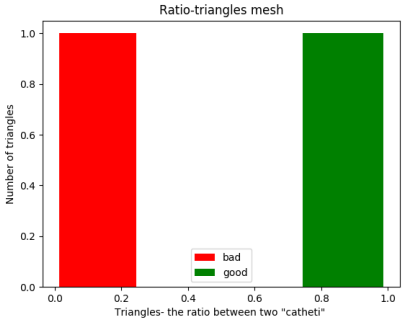


- **Mesh Delaunay algoritam:**



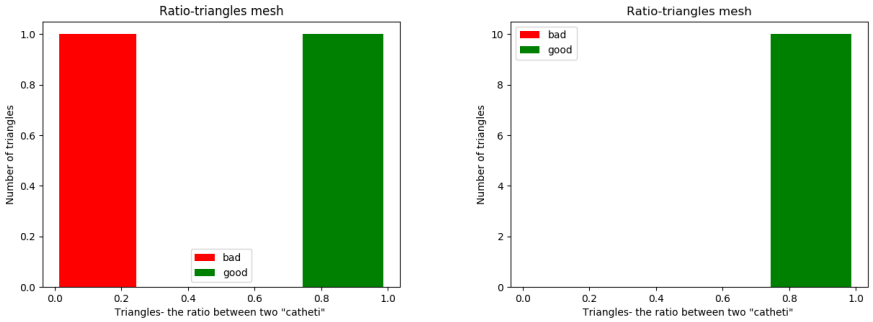
Kod Automatic, Delaunay i Frontal algoritma imamo zanemarivo malo trokuta.

- **Mesh Frontal algoritam:**



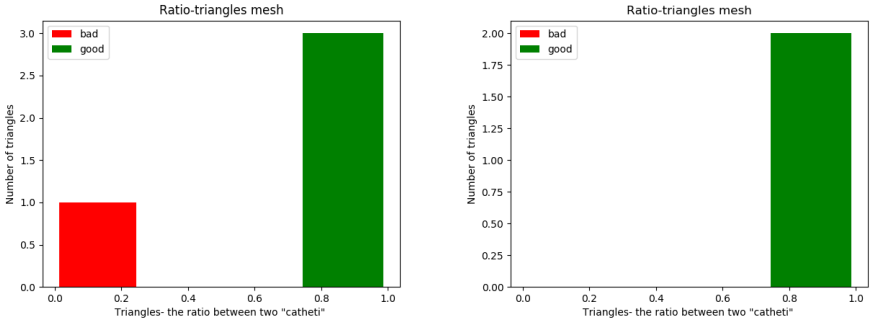
BAMG algoritam (nadzirani pristup) je dao najkvalitetnije rezultate omjera ”kateta” trokuta. Ima ukupno 10 trokuta u mreži i svi su kvalitetni.

- **Mesh BAMG algoritam:**



Unutar nadziranog pristupa trokuti se pojavljuju samo kod BAMG i Delquad algoritma.

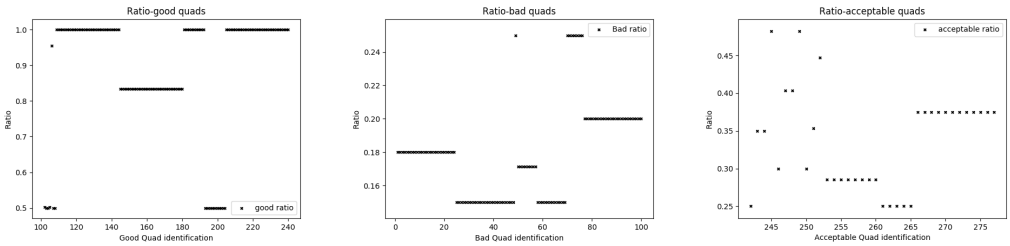
- **Mesh DelQuad algoritam:**



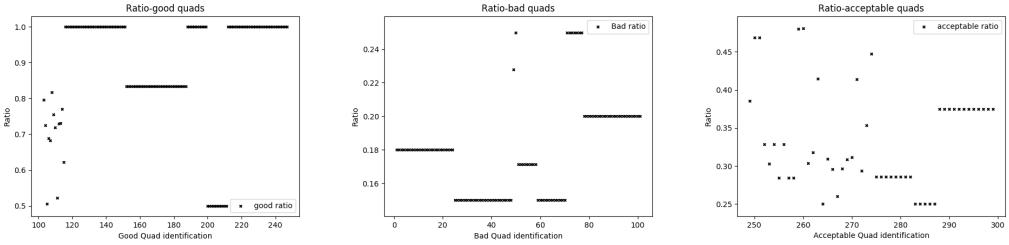
5) ***Vrijednosni prikaz omjera stranica***

Ovdje ćemo prikazati grafove koji uzimaju svaki konačni element i prikažu njegov omjer najveće i najmanje stranice. Na x osi su identifikacijski brojevi za svaki konačni element. Svaki algoritam će imati graf za dobre,loše i prihvatljive četverokute. Prvo su prikazani rezultati automatskog pristupa pa zatim rezultati nadziranog pristupa. Kao što možemo vidjeti iz priloženoga ostvarili smo veliki broj kvadrata u mreži te je glavni problem ostao u većem broju pravokutnika na rubnim i prijelaznim dijelovima brodske konstrukcije.

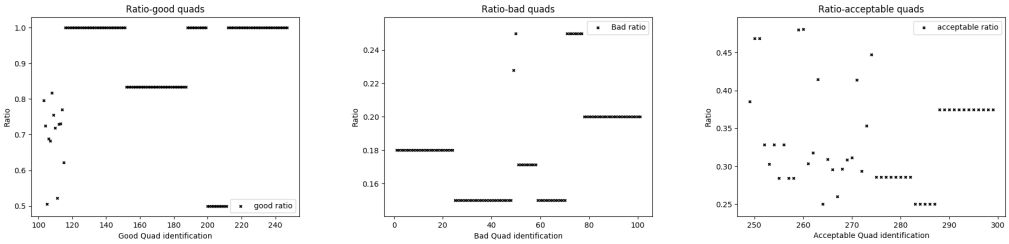
- **Mesh Adapt algoritam:**



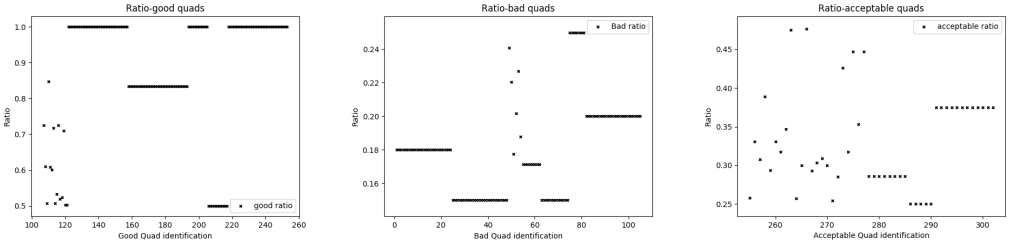
- **Mesh Automatic algoritam:**



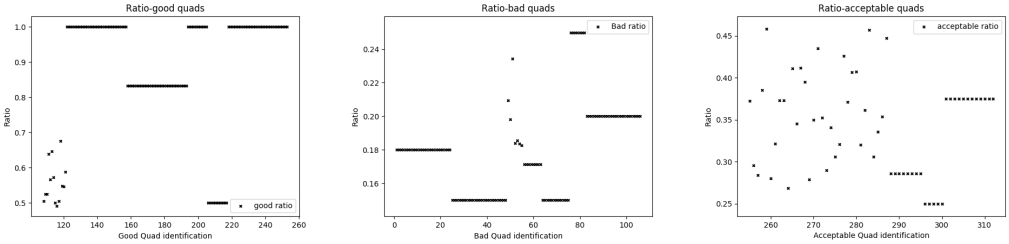
- **Mesh Delaunay algoritam:**



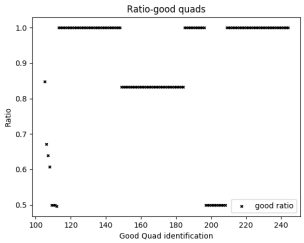
- **Mesh Frontal algoritam:**

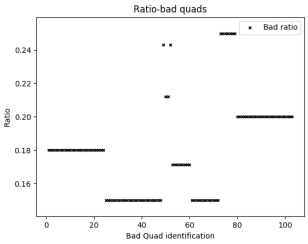


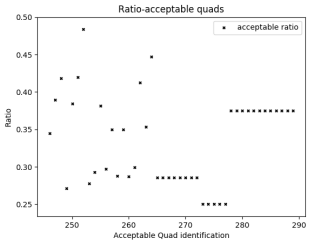
- **Mesh BAMG algoritam:**



- **Mesh DelQuad algoritam:**

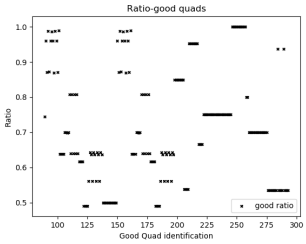


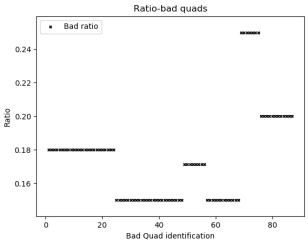


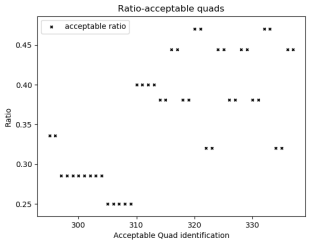


Vidimo da su algoritmi kod automatskog kreiranja mreže stvorili uvjete za veći broj kvadrata nego kod nadziranog pristupa, ali kod nadziranog pristupa je veći broj četverokuta ušao u kategoriju veoma dobrih.

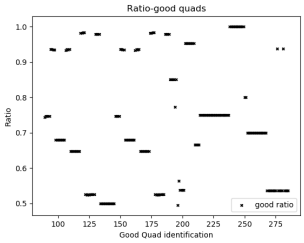
- **Mesh Adapt algoritam-nadzirani pristup:**

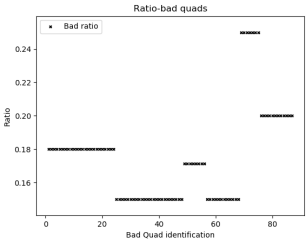


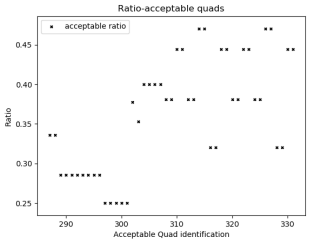




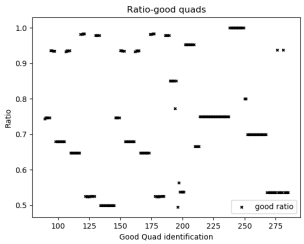
- **Mesh Automatic algoritam-nadzirani pristup:**

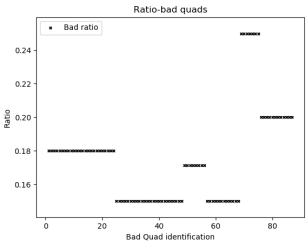


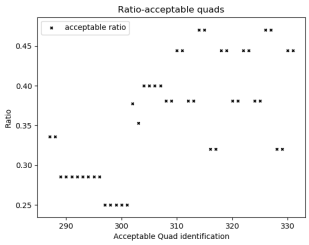




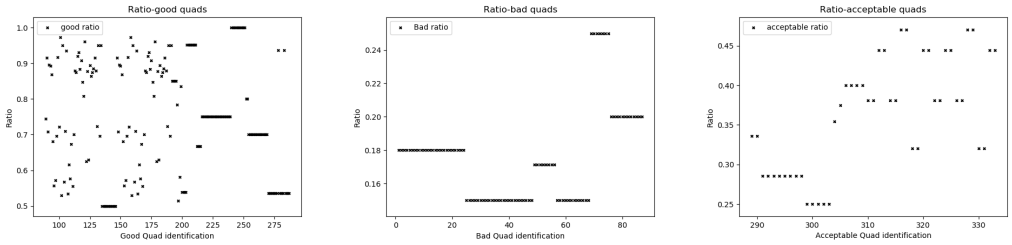
- **Mesh Delaunay algoritam-nadzirani pristup:**



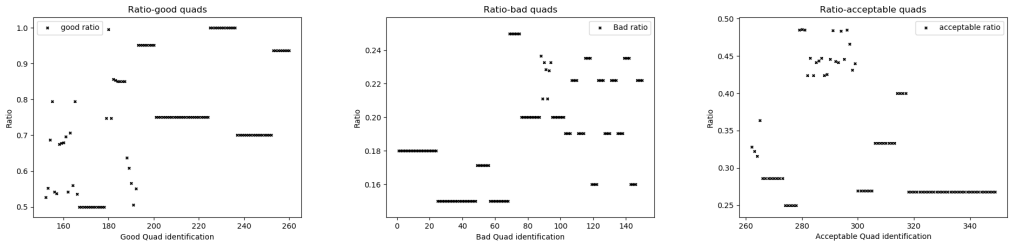




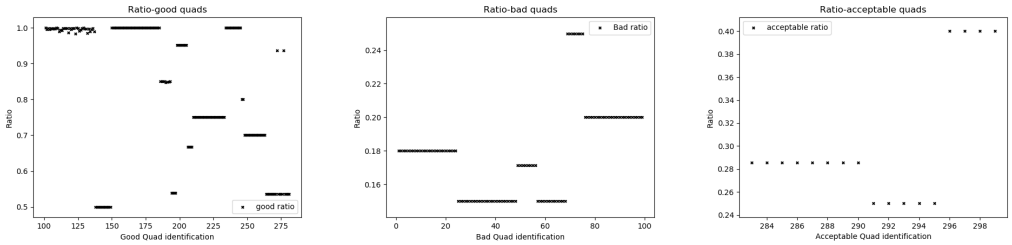
- **Mesh Frontal algoritam-nadzirani pristup:**



- **Mesh BAMG algoritam-nadzirani pristup:**



- **Mesh DelQuad algoritam- nadzirani pristup:**



3.4 Diskusija provedenih eksperimenata

Iz priloženih histograma i grafova vidimo da je u kreiranju kvalitetne mreže uspješniji nadzirani pristup. U uvjetu kvalitetnog omjera najveće i najmanje stranice u četverokutu vidjeli smo veći pomak u kvaliteti kada su se algoritmi koristili nakon nadziranog pristupa. Zanimljivost je da smo automatskim pristupom dobili veći broj kvadrata među veoma dobrim četverokutima, ali manji broj kvalitetnih četverokuta općenito. Najveći problem za kriterij kvalitete omjera ostaju rubni pravokutnici. Pronalazak koda koji bi ih automatski preraspodijelio u oblike bliže kvadratima znatno bi popravio ovaj uvjet kvalitete. Omjeri ”kateta” kod trokutnih elemenata su ili veoma dobri ili je broj trokuta u mreži zanemariv. U oba slučaja je ispunjen uvjet o pretežno četverokutnoj mreži. Uvjet kvalitetnog najmanjeg kuta ispunjen je u većoj mjeri i kod automatskog pristupa, a kod DelQuad algoritma nadziranim

pristupom je ostvaren stopostotni uspjeh. U budućim radovima koji se nadovezuju na ovaj rad i temu trebalo bi kroz iteracije automatskog i nadziranog pristupa otkriti kako generalno pronaći nezgodne dijelove na konstrukcijama. Automatizacijom nadziranog pristupa poboljšava se kvaliteta mreže te ubrzava njena kreacija.

3.5 Dodatak(kodovi)

- **Gmsh kod**

```
cl__1 = 1e+22;
Point(1) = {1200, 399.999999999999, 0, cl__1};
Point(2) = {1200, 400, 1000, cl__1};
...
Point(78) = {1200, 1200, 1000, cl__1};
Point(79) = {0, 1200, 1000, cl__1};
p1 = newp;
Point(p1 + 1) = {1200, 399.999999999999, 50.00000000000004};
Point(p1 + 2) = {1200, 399.9999999999991, 99.99999999999997};
Point(p1 + 3) = {1200, 399.9999999999991, 150};
Point(p1 + 4) = {1200, 399.9999999999992, 199.9999999999999};
...
Point(p1 + 19) = {1200, 399.9999999999999, 950};
Spline(1) = {1, p1 + 1, p1 + 2, p1 + 3, p1 + 4, p1 + 5, p1 + 6, p1
    + 7, p1 + 8, p1 + 9, p1 + 10, p1 + 11, p1 + 12, p1 + 13, p1 +
    14, p1 + 15, p1 + 16, p1 + 17, p1 + 18, p1 + 19, 2};
p2 = newp;
Point(p2 + 1) = {1197, 400, 1000};
Point(p2 + 2) = {1194, 400, 1000};
...
Point(p2 + 19) = {1143, 400, 1000};
Spline(2) = {2, p2 + 1, p2 + 2, p2 + 3, p2 + 4, p2 + 5, p2 + 6, p2
    + 7, p2 + 8, p2 + 9, p2 + 10, p2 + 11, p2 + 12, p2 + 13, p2 +
    14, p2 + 15, p2 + 16, p2 + 17, p2 + 18, p2 + 19, 3};
p3 = newp;

....

Spline(79) = {79, p79 + 1, p79 + 2, p79 + 3, p79 + 4, p79 + 5, p79
    + 6, p79 + 7, p79 + 8, p79 + 9, p79 + 10, p79 + 11, p79 + 12,
    p79 + 13, p79 + 14, p79 + 15, p79 + 16, p79 + 17, p79 + 18, p79
    + 19, 76};
```

```
Line Loop(1) = {1, 2, 3, 4};
Plane Surface(1) = {1};
Line Loop(2) = {5, 6, 7, 8};
Plane Surface(2) = {2};
Line Loop(3) = {9, 10, 11, 12};
Plane Surface(3) = {3};
...
Line Loop(17) = {72, 73, 74, 75};
Plane Surface(17) = {17};
Line Loop(18) = {76, 77, 78, 79};
Plane Surface(18) = {18};
```

- **Gmsh kod 2**

```
...
Point(p80+9)={640,1200,200};
Point(p80+10)={640,1200,0};
Point(p80+11)={600,1200,0};
Point(p80+12)={600,1200,200};

Line(3113)={p80+9,p80+10};
Line(3114)={p80+10,p80+11};
Line(3115)={p80+11,p80+12};
Line(3116)={p80+12,p80+9};
Line Loop(34)={3113,3114,3115,3116};
Plane Surface(34)={34};

Point(p80+13)={560,1200,200};
Point(p80+14)={560,1200,0};

Line(3117)={p80+13,p80+14};
Line(3118)={p80+14,p80+11};
Line(3119)={p80+11,p80+12};
Line(3120)={p80+12,p80+13};
Line Loop(35)={3117,3118,3119,3120};
Plane Surface(35)={35};

Point(p80+15)={640,1200,400};
Point(p80+16)={560,1200,400};
```

```
Line(3121)={p80+9,p80+15};
Line(3122)={p80+15,p80+16};
Line(3123)={p80+16,p80+13};
Line(3124)={p80+13,p80+9};

Line Loop(36)={3121,3122,3123,3124};
Plane Surface(36)={36};

Point(p80+17)={1200,1200,400};
Point(p80+18)={1200,1200,0,cl__1 };

Line(3125)={p80+10,p80+15};
Line(3126)={p80+15,p80+17};
Line(3127)={p80+17,p80+18};
Line(3128)={p80+18,p80+10};

Line Loop(37)={3125,3126,3127,3128};
Plane Surface(37)={37};

Point(p80+19)={0,1200,400,cl__1 };
Point(p80+20)={0,1200,0,cl__1 };

Line(3129)={p80+14,p80+16};
Line(3130)={p80+16,p80+19};
Line(3131)={p80+19,p80+20};
Line(3132)={p80+20,p80+14};

Line Loop(38)={3129,3130,3131,3132};
Plane Surface(38)={38};


Transfinite Surface {32};
//+
Transfinite Surface {31};


Transfinite Surface{34};
Transfinite Surface{35};
Transfinite Surface{36};
Transfinite Surface{37};
Transfinite Surface{38};
```

```
Recombine Surface {32,31,32};//+
Recombine Surface{34,35,36,37,38};



p81=newp;

Point(p81)={640,1200,400};
Point(p81+1)={640,1275,400};
Point(p81+2)={1200,1275,400};
Point(p81+3)={1200,1200,400};

Line(811)={p81,p81+1};
Line(812)={p81+1,p81+2};
Line(813)={p81+2,p81+3};
Line(814)={p81+3,p81};

Line Loop(40)={811,812,813,814};
Plane Surface(40)={40};
Transfinite Surface{40};
Recombine Surface{40};

p82=newp;

Point(p82)={560,1200,400};
Point(p82+1)={560,1275,400};
Point(p82+2)={0,1275,400};
Point(p82+3)={0,1200,400};

Line(821)={p82,p82+1};
Line(822)={p82+1,p82+2};
Line(823)={p82+2,p82+3};
Line(824)={p82+3,p82};

Line Loop(41)={821,822,823,824};
Plane Surface(41)={41};
Transfinite Surface{41};
Recombine Surface{41};

p83=newp;
Point(p83)={640,1200,400};
Point(p83+1)={560,1200,400};
```

```
Point(p83+2)={560,1275,400};
Point(p83+3)={640,1275,400};

Line(831)={p83,p83+1};
Line(832)={p83+1,p83+2};
Line(833)={p83+2,p83+3};
Line(834)={p83+3,p83};

Line Loop(42)={831,832,833,834};
Plane Surface(42)={42};
Transfinite Surface{42};
Recombine Surface{42};


p84=newp;
Point(p84)={0,1200,400};
Point(p84+1)={0,1125,400};
Point(p84+2)={560,1125,400};
Point(p84+3)={560,1200,400};

Line(841)={p84,p84+1};
Line(842)={p84+1,p84+2};
Line(843)={p84+2,p84+3};
Line(844)={p84+3,p84};

Line Loop(44)={841,842,843,844};
Plane Surface(44)={44};
Transfinite Surface{44};
Recombine Surface{44};

p85=newp;
Point(p85)={560,1200,400};
Point(p85+1)={560,1125,400};
Point(p85+2)={640,1125,400};
Point(p85+3)={640,1200,400};

Line(851)={p85,p85+1};
Line(852)={p85+1,p85+2};
Line(853)={p85+2,p85+3};
Line(854)={p85+3,p85};

Line Loop(45)={851,852,853,854};
```

```
Plane Surface(45)={45};
Transfinite Surface{45};
Recombine Surface{45};


p86=newp;
Point(p86)={640,1200,400};
Point(p86+1)={640,1125,400};
Point(p86+2)={1200,1125,400};
Point(p86+3)={1200,1200,400};

Line(861)={p86,p86+1};
Line(862)={p86+1,p86+2};
Line(863)={p86+2,p86+3};
Line(864)={p86+3,p86};

Line Loop(46)={861,862,863,864};
Plane Surface(46)={46};
Transfinite Surface{46};
Recombine Surface{46};
```

- **Python kod-unošenje geometrije,rekombinacija i spremanje mreže**

```
...
geom = pygmsh.built_in.Geometry()

with open('proba02.txt', 'r') as myfile:
for line in myfile:
if 'Plane Surface' in line:
Surface.append(find_id(line))
geom.add_raw_code(line)

elif 'Line Loop' in line:
number=line.count(',')
if number>=4:
Line_loop.append(find_Line_id(line))
Splines_id.append(solve_problem_spline_id(line))

geom.add_raw_code(line)

else:
```

```
geom.add_raw_code(line)

#print(Surface)
myfile.close()

Points_id.append(solve_problem_points_id(Splines_id))
Points1.append(find_cor(Points_id))

#print(Points)

geom.add_raw_code('Mesh.Algorithm=8;')


for i in Surface:
if i not in Line_loop:
geom.add_raw_code('Transfinite Surface {%s};' %i)
geom.add_raw_code('Recombine Surface{%s}=0;' %i)

for i in Line_loop:
geom.add_raw_code('Recombine Surface{%s}=0;' %i)

points, cells, point_data, cell_data, field_data =
    pygmsh.generate_mesh(geom)

meshio.write('uspjeh.vtu', points, cells, cell_data=cell_data)
...
```

- **Python-pomoćne funkcije za otkrivanje kvalitete elemenata mreže**

```
def ratio_tr(tr,points):
point1=points[tr[0]]
point2=points[tr[1]]
point3=points[tr[2]]

po=[]


d1=distance_between_points(point1,point2)
d2=distance_between_points(point2,point3)
d3=distance_between_points(point3,point1)

po.append(d1)
```

```
po.append(d2)
po.append(d3)

po.sort()

cathetus1=po[0]
cathetus2=po[1]

if cathetus1/cathetus2<=1 and cathetus1/cathetus2>=0.49:
return 1
else: return 0

def ratio(quad,points):


point1=points[quad[0]]
point2=points[quad[1]]
point3=points[quad[2]]
point4=points[quad[3]]

po=[]


d1=distance_between_points(point1,point2)
d2=distance_between_points(point2,point3)
d3=distance_between_points(point3,point4)
d4=distance_between_points(point4,point1)

po.append(d1)
po.append(d2)
po.append(d3)
po.append(d4)

po.sort()

if po[0]/po[3]<=1 and po[0]/po[3]>=0.49 :
return 1

elif po[0]/po[3]<0.49 and po[0]/po[3]>=0.25:
return 2
else: return 0
```

```
def degree(quad,points):

point1=points[quad[0]]
point2=points[quad[1]]
point3=points[quad[2]]
point4=points[quad[3]]

po=[]

#vectors
p1=[]
p2=[]
p3=[]
p4=[]

p1.append(point1[0]-point2[0])
p1.append(point1[1]-point2[1])
p1.append(point1[2]-point2[2])

p2.append(point3[0]-point2[0])
p2.append(point3[1]-point2[1])
p2.append(point3[2]-point2[2])

p3.append(point4[0]-point3[0])
p3.append(point4[1]-point3[1])
p3.append(point4[2]-point3[2])

p4.append(point1[0]-point4[0])
p4.append(point1[1]-point4[1])
p4.append(point1[2]-point4[2])



d1=degree_between_vectors(p1,p2)

p2[0]=-p2[0]
p2[1]=-p2[1]
p2[2]=-p2[2]

d2=degree_between_vectors(p2,p3)
```

```
p3[0]=-p3[0]
p3[1]=-p3[1]
p3[2]=-p3[2]

d3=degree_between_vectors(p3,p4)

p1[0]=-p1[0]
p1[1]=-p1[1]
p1[2]=-p1[2]

p4[0]=-p4[0]
p4[1]=-p4[1]
p4[2]=-p4[2]


d4=degree_between_vectors(p1,p4)

po.append(d1)
po.append(d2)
po.append(d3)
po.append(d4)

for p in po:
if p<60: return 0
return 1
```

- **Python podjela na dobre i loše elemente**

```
br=0
for quad in cells["quad"]:
br+=1
if ratio(quad,points)==1:
quads_with_good_ratio.append(quad)
values_quad_ratio.append(v(quad))
elif ratio(quad,points)==0:
quads_with_problematic_ratio.append(quad)
values_quad_ratio.append(v(quad))
else:
quads_025_ratio.append(quad)
values_quad_ratio.append(v(quad))

for quad in cells["quad"]:
```

```
if degree(quad,points)==1:
quads_good_degree.append(quad)
values_degree.append(v_d(quad))
else:
quads_problematic_degree.append(quad)
values_degree.append(v_d(quad))

for tr in cells["triangle"]:

if ratio_tr(tr,points)==1:
triangles_good_ratio.append(tr)
values_triangle_ratio.append(v_t(tr))
else:
triangles_problematic_ratio.append(tr)
values_triangle_ratio.append(v_t(tr))

bad=[]
good=[]
acceptable=[]
for i in values_quad_ratio:
if i<0.25:
bad.append(i)
elif i>=0.25 and i<0.49:
acceptable.append(i)
else: good.append(i)

bad_t=[]
good_t=[]


for i in values_triangle_ratio:
if i<0.49:
bad_t.append(i)

else: good_t.append(i)

good_degree=[]
bad_degree=[]


for i in values_degree:
```

```
if i <60:
bad_degree.append(i)
else: good_degree.append(i)
```

Bibliografija

[1] Remacle, J.-F. and Henrotte, F. and Carrier-Baudouin, T. and Bechet, E. and Marchandise, E. and Geuzaine,C. and Mouton, T. *A frontal Delaunay quad mesh generator using the L^∞ norm*, dostupno na: `http://gmsh.info/doc/preprints/gmsh_quad2_preprint.pdf` (2002.)

[2] Remacle, J.-F. and Geuzaine,C. , *Gmsh - Reference manual* , dostupno na: `http://gmsh.info/doc/texinfo/gmsh.html`, (1997.-2018.)

[3] Sorić Jurica, *Metoda konačnih elemenata*, Golden marketing-Tehnička knjiga, 2004.

[4] Schlömer, N, *pygmsh's documentation*, dostupno na: `https://pygmsh.readthedocs.io/en/latest/`

[5] Tenek,Lazarus Teneketzis and Argyris,J.*Finite element analysis for composite structures*, Kluwer Academic Publishers Group, Dordrecht, 1998.

[6] Richard Courant,*Variational methods for the solution of problems of equilibrium and vibrations*, Bull. Amer. Math. Soc. 49, (1943.)

[7] Strang, G. and Fix, G.*An analysis of the finite element method. Second edition*, Wellesley-Cambridge Press, Wellesley, MA, 2008.

[8] Ray William Clough and Joseph Penzien*Dynamics of structures*, 1975.

[9] Open Cascade Information Technology Company, Capgemini, dostupno na: https://www.opencascade.com/

[10] SwiftComp, dostupno na: http://analyswift.com/swiftcomp-vamuch-micromechanics-modeling-of-heterogeneous-materials/

[11] Composites design and Manufacturing Hub, dostupno na: https://cdmhub.org/

[12] MathWorks, dostupno na: https://www.mathworks.com/discovery/matlab-gui.html

[13] FEATool Multiphysics, dostupno na: https://www.featool.com/

[14] Delaunay triangulation, dostupno na: https://en.wikipedia.org/wiki/Delaunay_triangulation

[15] I. Babuska, M. Griebel and J. Pitkaranta, *The problem of selecting the shape functions for a p-type finite element*, Internat. J. Numer. Methods Engrg. (1989), pp. 1891–1908

Sažetak

U ovom radu opisana je metoda konačnih elemenata, generatori, alati i algoritmi koji se koriste da bi se ona ostvarila u praksi te praktičan zadatak kreiranja mreže konačnih elemenata na brodskoj konstrukciji. Na početku rada smo sklopili matricu krutosti i opisali osnovne elemente koji su bitni za jednostavnije integriranje po površini konačnih elemenata te za ispunjenje uvjeta kvalitete mreže. Opisali smo glavne algoritme gmsh generatora te prikazali njihove rezultate u dobivanju mreže na brodskom modelu. Objasnili smo zašto je važno napraviti dobru triangulaciju prije rekombinacije te naveli Frontal-Delaunay triangulaciju kao pozitivan primjer. Definirali smo pozitivne strane dobivene mreže, opisali prostor za napredak u kvaliteti mreže i kako to postići.

Summary

This paper describes a finite element method together with the generators, tools and algorithms needed for its use. It also provides an example of generating the mesh of finite elements for a ship structure. The first part of the paper assembles the stiffness matrix and describes basic finite elements necessary for simple integration over the surface of finite elements as well as for meeting the conditions of mesh quality. (In the second part), the main algorithms of the gmsh mesh generator are described and these outcomes are further used for making the mesh of the ship structure. We explained why it was important to make good triangulation before recombination. Frontal-Delaunay triangulation was indicated as a positive example. The advantages of the generated mesh are clearly explained. The paper also indicates possible improvements in mesh quality as well as the methods of achieving these improvements.

Životopis

Rođen sam 05.07.1991. u Zagrebu gdje sam završio osnovnu školu Trnsko i V. prirodoslovno matematičku gimnaziju. 2011. Upisujem preddiplomski studiji matematike na Matematičkom odsjeku PMF-u u Zagrebu koji završavam 2015. godine. Diplomski studiji matematike i računarstva upisujem 2015. i završavam ga 2018. godine pod mentorstvom prof.dr.sc. Luke Grubišića.