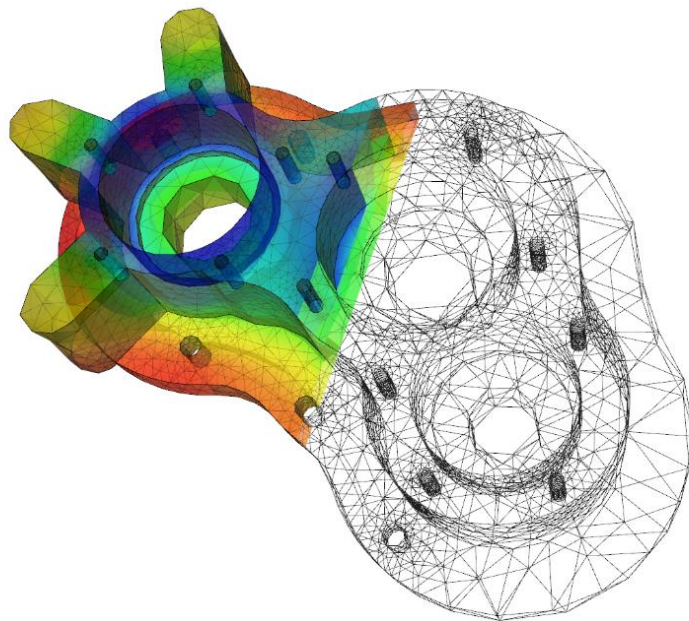


# Inženirske računalniške simulacije v konstrukterstvu

Univerzitetni učbenik



Avtorji:  
Zoran Ren  
Miran Ulbin  
Matej Vesenjāk

Maribor, 2018





Univerza v Mariboru

---

Fakulteta za strojništvo

# Inženirske računalniške simulacije v konstrukterstvu

Univerzitetni učbenik

Avtorji:  
Zoran Ren  
Miran Ulbin  
Matej Vesenjaka

Maribor, marec 2018

Naslov: Inženirske računalniške simulacije v konstrukterstvu

Podnaslov: Univerzitetni učbenik

Avtorji: red. prof. dr. Zoran Ren (Univerza v Mariboru, Fakulteta za strojništvo),  
izr. prof. dr. Miran Ulbin (Univerza v Mariboru, Fakulteta za strojništvo),  
izr. prof. dr. Matej Vesenjajk (Univerza v Mariboru, Fakulteta za strojništvo).

Strokovna recenzija: izr. prof. dr. Janez Kramberger (Univerza v Mariboru, Fakulteta za strojništvo),  
red. prof. dr. Jernej Klemenc (Univerza v Ljubljani, Fakulteta za strojništvo).

Jezikovna recenzija: Nataša Belšak, univ. dipl. ling.

Grafične priloge: Avtorji.

Slike na prvi strani: Elmer-pump-heatequation.png, CC BY-SA 3.0, naloženo: 02.6.2010;  
<https://de.wikipedia.org/wiki/Finite-Elemente-Methode>

Izdajateljica:

Univerza v Mariboru, Fakulteta za strojništvo  
Smetanova ulica 17, 2000 Maribor, Slovenija  
<http://www.fs.um.si>, [fs@um.si](mailto:fs@um.si)

Založnik:

Univerzitetna založba Univerze v Mariboru  
Slomškov trg 15, 2000 Maribor, Slovenija  
<http://press.um.si>, [zalozba@um.si](mailto:zalozba@um.si)

Izdaja: Prva izdaja.

Vrsta publikacije: E-publikacija

Dostopno na: <http://press.um.si/index.php/ump/catalog/book/321>

Izid: Maribor, marec 2018

© Univerzitetna založba Univerze v Mariboru

Vse pravice pridržane. Brez pisnega dovoljenja založnika je prepovedano reproduciranje, distribuiranje, predelava ali druga uporaba tega dela ali njegovih delov v kakršnemkoli obsegu ali postopku, vključno s fotokopiranjem, tiskanjem ali shranjevanjem v elektronski obliki.

CIP - Kataložni zapis o publikaciji  
Univerzitetna knjižnica Maribor

004.942:621-11(075.8)(0.034.2)

REN, Zoran

Inženirske računalniške simulacije v konstrukterstvu [Elektronski vir] :  
univerzitetni učbenik / avtorji Zoran Ren, Miran Ulbin, Matej Vesenjajk. - 1. izd. - Maribor  
: Univerzitetna založba Univerze, 2018

Način dostopa (URL): <http://press.um.si/index.php/ump/catalog/book/321>

ISBN 978-961-286-151-3 (pdf)

doi: 10.18690/978-961-286-140-7

1. Ulbin, Miran 2. Vesenjajk, Matej

COBISS.SI-ID [94242817](https://www.cobiss.si/id/94242817)

ISBN: 978-961-286-151-3 (PDF)  
978-961-286-140-7 (Tiskana izdaja)

DOI: <https://doi.org/10.18690/978-961-286-140-7>

Cena: Brezplačen izvod.

Odgovorna oseba založnika: red. prof. dr. Žan Jan Oplotnik, prorektor Univerze v Mariboru

DOI <https://doi.org/10.18690/978-961-286-140-7>

ISBN 978-961-286-151-3

© 2018 Univerzitetna založba Univerze v Mariboru

Dostopno na: <http://press.um.si>

# Inženirske računalniške simulacije v konstrukterstvu

ZORAN REN, MIRAN ULBIN & MATEJ VESENJAK

Povzetek Učbenik s področja inženirskih računalniških simulacij je namenjen kot študijski pripomoček pri izvedbi predavanj predmetov Inženirske računalniške simulacije in Računalniške simulacije v konstrukterstvu ter posredno pri predmetih Inženirska orodja 2 in Numerična mehanika trdnin. Vsebuje razlago celotne snovi, ki jo morajo študentje pri teh predmetih osvojiti, in je skladen z učnimi načrtom omenjenih predmetov.

Ključne besede: • računalniške simulacije • mehanika trdnih teles • metoda končnih elementov • konstruiranje • numerične metode •

---

NASLOVI AVTORJEV: dr. Zoran Ren, redni profesor, Univerza v Mariboru, Fakulteta za strojništvo, Smetanova ulica 17, 2000 Maribor, Slovenija, e-pošta: zoran.ren@um.si. dr. Miran Ulbin, izredni profesor, Univerza v Mariboru, Fakulteta za strojništvo, Smetanova ulica 17, 2000 Maribor, Slovenija, e-pošta: miran.ulbin@um.si. dr. Matej Vesenjāk, izredni profesor, Univerza v Mariboru, Fakulteta za strojništvo, Smetanova ulica 17, 2000 Maribor, Slovenija, e-pošta: matej.vesenjāk@um.si.

DOI <https://doi.org/10.18690/978-961-286-140-7>  
© 2018 Univerzitetna založba Univerze v Mariboru  
Dostopno na: <http://press.um.si>

ISBN 978-961-286-151-3

## Vsebina

1. Računalniško podprto inženirstvo .....	1
1.1 Analitični ali standardni preračuni .....	1
1.2 Računalniško podprto inženirstvo in metoda končnih elementov .....	4
1.2.1 Predprocesiranje numeričnih modelov .....	5
1.2.2 Procesiranje numeričnih modelov .....	6
1.2.3 Poprocesiranje numeričnih modelov .....	7
1.3 CAE-programi .....	7
1.4 Merski sistemi v CAE .....	9
1.5 Literatura .....	11
2. Zgodovinski pregled računalniških simulacij .....	13
2.1 Matematika in fizika .....	13
2.2 Računalniki in programiranje .....	18
2.3 Programska oprema za inženirske simulacije .....	24
2.4 Literatura .....	29
3. Osnove metode končnih elementov .....	35
3.1 Osnove mehanike elastičnih trdnih teles v MKE .....	37
3.2 Princip virtualnega dela .....	40
3.3 Osnovni aproksimacijski nastavek MKE .....	41
3.4 Enačbe na nivoju končnega elementa in celotnega telesa .....	42
3.5 Tipi končnih elementov .....	44
3.6 Interpolacijske funkcije končnih elementov .....	46
3.7 Interpolacijske funkcije v naravnih koordinatah .....	49
3.8 Izoparametrični končni elementi .....	54
3.9 Numerična integracija togostne matrike in porazdeljenih obremenitev .....	55
3.10 Osnovne enačbe MKE na primeru linijskega končnega elementa .....	57
3.11 Napake pri delu z metodo končnih elementov .....	61
3.12 Literatura .....	64
4. Geometrijsko modeliranje .....	65
4.1 Linijski modeli .....	65
4.2 Volumetrični modeli .....	67
4.3 Prostorski modeli .....	71
4.4 Literatura .....	75

5. Materialne lastnosti v MKE.....	77
5.1 Modul elastičnosti in Poissonovo razmerje .....	77
5.2 Materialni podatki v zahtevnejših simulacijah.....	80
5.3 Določanje materialnih podatkov v programih MKE .....	82
5.4 Literatura .....	83
6. Fizikalne lastnosti numeričnih modelov .....	85
6.1 Linijske fizikalne lastnosti .....	85
6.2 Ravninske fizikalne lastnosti .....	87
6.2.1 Ravninsko napetostno stanje (RNS) .....	87
6.2.2 Ravninsko deformacijsko stanje (RDS).....	89
6.2.3 Osno-simetrično stanje .....	91
6.3 Plošče, lupine in membrane.....	92
6.4 Prostorski fizikalni modeli.....	93
6.5 Določanje fizikalnih veličin v programih MKE .....	93
6.6 Literatura .....	94
7. Sestava numeričnih modelov.....	95
7.1 Koordinatni sistemi.....	95
7.2 Transformacije .....	98
7.3 3D-pogledi in projekcije.....	101
7.4 Transformacije v programu ABAQUS.....	103
7.5 Literatura .....	105
8. Robni pogoji .....	107
8.1 Podpore v eni dimenziji.....	107
8.2 Prostostne stopnje.....	110
8.3 Določanje robnih pogojev .....	115
8.4 Določanje robnih pogojev v programu ABAQUS.....	117
8.5 Literatura .....	122
9. Obremenitve numeričnih modelov.....	123
9.1 Vrste obremenitev .....	123
9.2 Točkovne obremenitve .....	125
9.3 Porazdeljene obremenitve.....	125
9.4 Predpisovanje obremenitev v programu ABAQUS .....	127
9.5 Literatura .....	130
10. Mreženje s končnimi elementi .....	131
10.1 Uvoz geometrije modela .....	131
10.2 Gostota mreže končnih elementov .....	132

10.3 Samodejno mreženje s končnimi elementi .....	135
10.4 Kvaliteta končnih elementov .....	137
10.5 Mreženje v programu ABAQUS .....	138
10.6 Literatura.....	140
11. Reševanje sistema enačb .....	141
11.1 Direktno reševanje sistema enačb .....	143
11.2 Iterativno reševanje sistema enačb .....	145
11.3 Reševanje sistema enačb v ABAQUS-u .....	146
11.4 Literatura.....	148
12. Poprocesiranje numeričnih rezultatov .....	149
12.1 Kontrola numeričnega modela in rešitev.....	149
12.2 Številčni rezultati in preglednice.....	150
12.3 Diagrami .....	151
12.4 Slike .....	151
12.4.1 Pomiki .....	151
12.4.2 Specifične deformacije.....	153
12.4.3 Napetosti .....	153
12.5 Animacije .....	158
12.6 Poprocesiranje v programu ABAQUS.....	158
12.7 Literatura.....	159
13. Poročilo o izvedeni računalniški simulaciji.....	161
13.1 Vsebina poročila .....	161
13.1.1 Naslovna stran in uvodne strani .....	161
13.1.2 Opis in opredelitev problema .....	161
13.1.3 Opis numeričnega modela .....	162
13.1.4 Opis izvedbe računalniške simulacije.....	164
13.1.5 Rezultati.....	164
13.1.6 Zaključki in strokovno mnenje .....	166
13.1.7 Seznam uporabljenih virov.....	166
13.1.8 Priloge .....	166
13.2 Oblika poročila .....	166
13.3 Samodejno pripravljena poročila.....	166
13.4 Literatura.....	167



# 1. Računalniško podprto inženirstvo

Kratica CAE (angl. *Computer Aided Engineering*) – računalniško podprto inženirstvo označuje programsko opremo za računalniške simulacije in testiranje. Sem spadajo tako različne analize z uporabo numeričnih metod kot tudi različni analitični preračuni, računalniški simulatorji delovanja in eksperimentalne metode za določanje vzdržljivosti izdelkov. Ta učbenik obravnava le metodologijo in programsko opremo za simulacije in preračune. Kljub temu pa je treba vedeti, da je rezultate simulacij vedno treba preveriti s preizkusi. Programska oprema CAE je običajno preverjana na testnih primerih, vendar je vsak uporabnik dolžan rezultate simulacij potrditi s primernimi preizkusi.

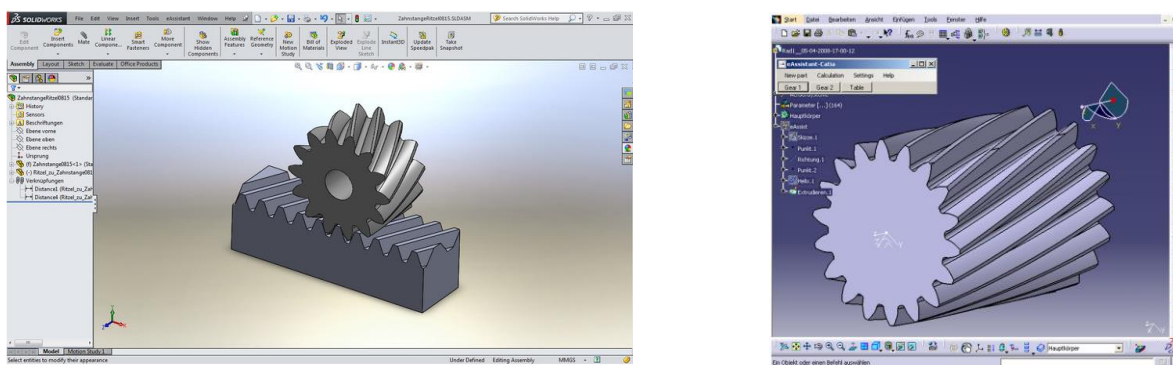
## 1.1 Analitični ali standardni preračuni

Računalniki so primerni za različne inženirske preračune. Dolgo je bilo pomično logaritemsko računalo osnovni pripomoček inženirja. Leta 1970 je to postal kalkulator, ki je še danes, v nekoliko bolj sofisticirani obliki, vedno na delovni mizi inženirja. Ker pa je na njegovi delovni mizi dandanes običajno precej bolj zmogljiv računalnik, ki je nadomestil tudi risalno desko, se tudi kalkulatorji počasi umikajo v predale in omare. Namesto tega je v uporabi še zmogljivejši virtualni kalkulator v obliki programske preglednice, kot je npr. Microsoft Excel [1.1] ali OpenOffice Calc [1.2]. Pri teh programih je celoten ekran preglednica in vsaka celica preglednice je zaslon kalkulatorja. Pri tem je možno uporabljati v posameznem računu rezultate drugih računov v isti oz. kakšni drugi preglednici. Možne so kompleksne operacije na veliki količini podatkov. Hkrati je mogoče podatke grafično predstaviti v obliki različnih grafov. Ti programi omogočajo tudi programiranje v obliki makro programov – tako lahko uporabnik izdela celoten preračun v preglednici, vključno s prijaznim uporabniškim vmesnikom. V ta namen Microsoft uporablja programski jezik basic oz. VBA (angl. *Visual Basic for Application*) [1.3]. Ta programski jezik se zelo pogosto uporablja tudi v programih za računalniško podprto konstruiranje (angl. *CAD – Computer Aided Design*), npr. Catia, Solidworks, Autocad. Pogosto je mogoče povezati preglednico s programom za računalniško podprto konstruiranje in tako lahko npr. program za preračun strojnega elementa samodejno določi dimenzije tega strojnega elementa glede na predvidene obremenitve.

Poleg preračunov v običajnih preglednicah je veliko takšnih preračunov na voljo tudi v obliki spletnih aplikacij, npr. Craftsmanspace [1.4]. Ti preračuni so lahko prosto dostopni (angl. *freeware*), na voljo za uporabo le določen čas (angl. *shareware*) ali plačljivi. Pogosto zahteva izdelava naprednih in obsežnih preračunov veliko časa in veliko vložnega dela, zato pogosto avtorji za kvaliteten izdelek pričakujejo primerno plačilo.

Takšne preračune lahko integriramo v obstoječe programe za računalniško podprto konstruiranje kot vtičnike (angl. *plugin*), saj se po namestitvi obnašajo kot dodaten modul obstoječega programa. Najbolj znan vtičnik je MITCalc [1.5], ki je na voljo za programe Autocad Inventor, Solidworks, Solidedge, Creo in omogoča vključitev več različnih preračunov strojnih elementov neposredno v program za računalniško podprto konstruiranje. Podoben vtičnik je tudi Eassistant [1.6], ki je na voljo za programe Solidworks, Solidedge, Autodesk

Inventor, Catia, NX, Creo. Na sliki 1.1 je prikazan primer delovanja vtičnika v programih Solidworks in Catia. Ker preglednica vse makro programe izvaja v obliki interpreterja (sproti prevaja in izvaja ukaze), so zahtevne aplikacije v preglednicah zelo počasne. Včasih pa so programi, potrebni za preračun, precej bolj kompleksni in vsebujejo tudi veliko podatkov, potrebnih za preračun. Zato v teh primerih za inženirske preračune uporabljamo namensko izdelano programsko opremo.



Slika 1.1 – Eassistant v programih Solidworks in Catia

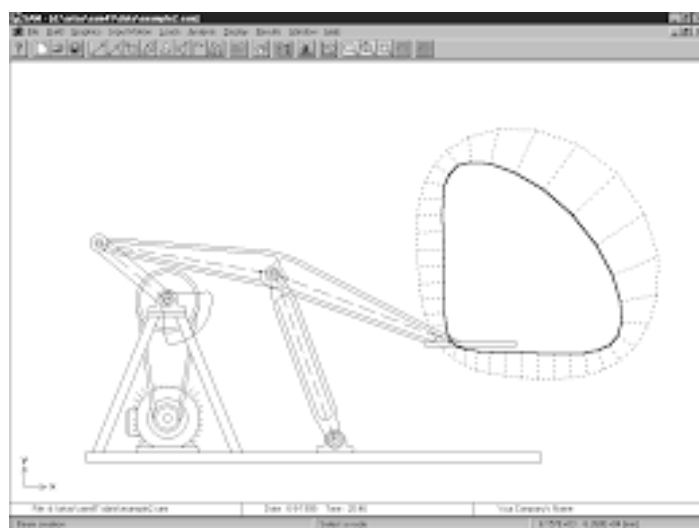
Pogosto za kompleksne strojne elemente, kot so zobniki, uporabljamo samostojno programsko opremo. Takšno programsko orodje je na primer Kisssoft [1.7], ki omogoča celovito podporo za preračun in dimenzioniranje vseh vrst strojnih elementov. Za preračun valjastih zobnikov smo tudi na Fakulteti za strojništvo Univerze v Mariboru razvili programski študijski pripomoček za dimenzioniranje zobniških gonil Zobnik (slika 1.2).

OZNAKA	OPIS	ENOTA	ZOBNIKI	ZOBNIKI2
Mot	$P_1, P_2$	kW	1,57	1,488
Preravnano razmerje	$i$		3,529	3,529
Višina klanca	$h_a, h_b$	mm	2,060	42,5
Širina zob	$b_1, b_2$	mm	17	60
Nominalni modul	$m_n$	mm	1,25	1,25
Širina zob	$b_1, b_2$	mm	10	8
Medosaj	$a$	mm	48,125	48,125
Ukrajni kot na razdelni krogu	$\alpha_n$	°	20	20
Ukrajni kot na klančnem krogu	$\alpha_k$	°	20	20
Kot bočati na razdelni krogu	$\beta$	°	0	0
Koeficient profilnega premika	$x_1, x_2$	mm	0,1862	-0,1862
Premer razdelnega kroga	$d_a, d_b$	mm	21,25	75,0
Premer klančnega kroga	$d_{ka}, d_{kb}$	mm	24,25	79,04
Premer razdelnega kroga	$d_f, d_g$	mm	18,91	71,408
Premer klančnega kroga	$d_{fa}, d_{fb}$	mm	21,25	75,0
Premer osnovnega kroga	$d_{o1}, d_{o2}$	mm	19,968	70,477
Stopnja profilnega skrajšanja	$\gamma_p$		1,61	1,61
Stopnja bočnega skrajšanja	$\gamma_b$		0,0	0,0
Višina klanca odloščitelne sile	$h_{a1}, h_{a2}$	mm	1,563	1,563
Zaščitni klanec odloščitelne sile	$s_{a1}, s_{a2}$	mm	1,563	1,563
Kritična osnaja	$\delta$	°	6	6

Slika 1.2 – Program Zobnik za preračun valjastih zobnikov

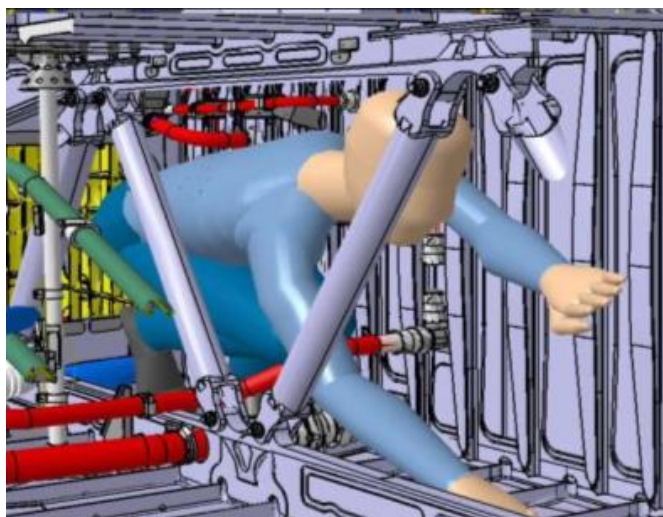
Podobnih namenskih programskih orodij za simulacijo in preračun točno določenih izdelkov je veliko. Če želimo konstruirati poljubne izdelke, pa potrebujemo bolj splošno programsko opremo. Pri simulacijah običajno hitro pomislimo na gibanje in film, ki prikazuje kinematiko

gibanja izdelka. Enostaven program za simulacijo mehanizmov je večkrat nagrajen program SAM podjetja Artas [1.8], prikazan na sliki 1.3. Vodilni program za simulacijo mehanizmov in dinamične analize je ADAMS, ki ga trži firma MSC [1.9]. Če želimo simulirati samo kinematiko izdelka, lahko naredimo takšno simulacijo skoraj v vsakem modelirniku, v katerem smo izdelali sestavo. Takšno simulacijo lahko potem shranimo v obliki videoposnetka, lahko pa tudi analiziramo celoten mehanizem, da se ne pojavi trk ali prekrivanje med sestavnimi deli. V ta okvir simulacij spadajo tudi simulacije delovanja mehanizma, s katerim simuliramo delovni proces našega izdelka in ga nato prikažemo v obliki videoposnetka. Pri tem se seveda postavlja vprašanje, kdaj gre za simulacijo in kdaj za animacijo, saj je končni izdelek identičen. Animacija je definirana tako, da prikazujemo zaporedje kreiranih sličic, medtem ko pri simulaciji na podlagi fizikalnih zakonov ustvarimo posamezne slike videoposnetka. Danes je meja med obema terminoma precej zabrisana, saj se tudi pri kreiranju računalniških iger in risanih filmov vedno pogosteje uporabljajo simulacije, in ne animacije.



Slika 1.3 – Program za konstruiranje in simulacijo mehanizmov SAM [1.8]

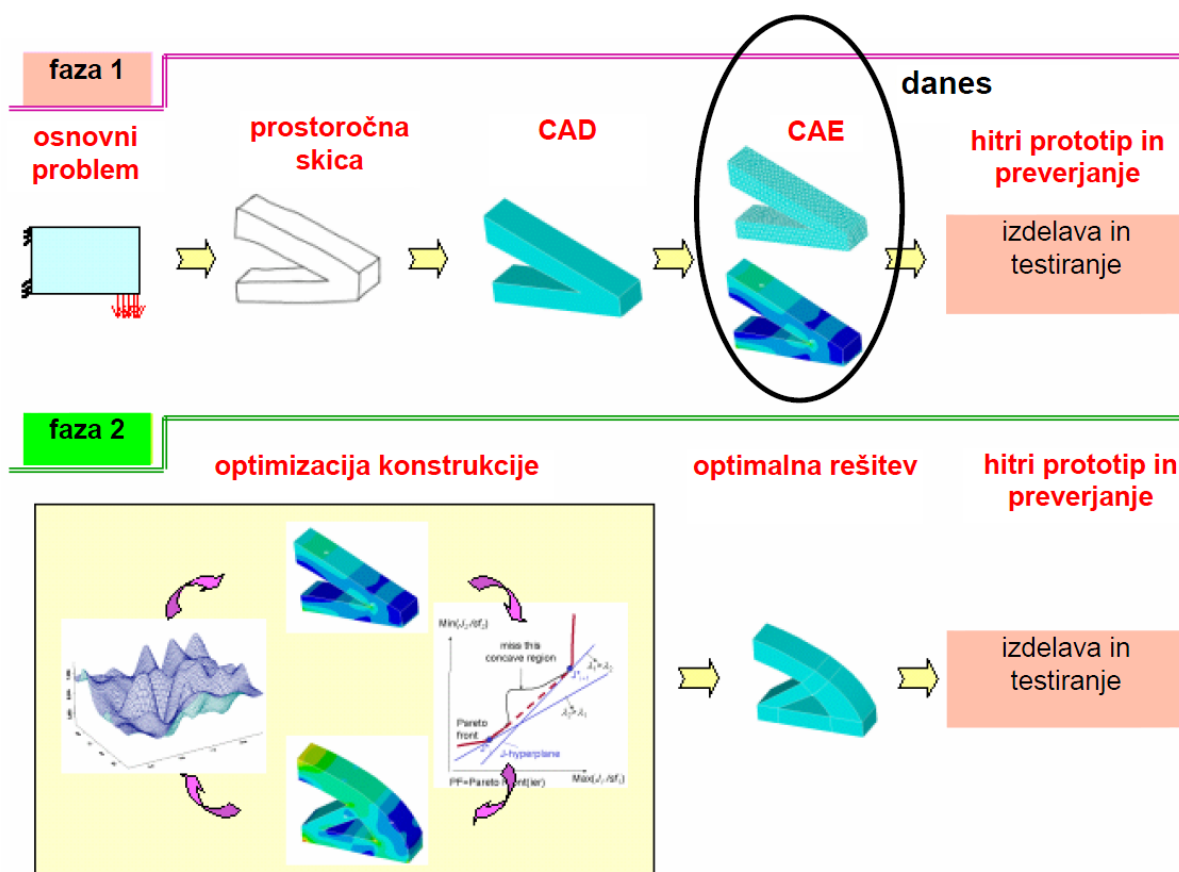
V zadnjem času so postale zelo aktualne tudi ergonomске simulacije. Običajno imamo v ta namen na voljo lutke (angl. *dummy*), ki jih postavljamo na različna mesta in v različnih pozah glede na naš izdelek. Simulacija tako poteka ročno s premikom lutke in njenih udov, seveda pa je mogoče tudi v tem primeru celoten delovni proces shraniti v obliki videoposnetka. Na tak način simuliramo in analiziramo ergonomijo izdelka, lahko pa tudi simuliramo različna vzdrževalna dela, kot je prikazano na sliki 1.4.



Slika 1.4 – Simulacija vzdrževanja v programu Catia [1.10]

## 1.2 Računalniško podprto inženirstvo in metoda končnih elementov

Večina simulacij je namenjena konstruiranju izdelka. Pri konstruiranju izdelka lahko navadno ločimo dve fazi simulacij. V prvi fazi mora izdelek zadoščati predpisanim zahtevam, tudi če je bistveno predimenzioniran. V preteklosti so inženirji svoje izdelke preračunavali z analitičnimi preračuni, ki so vsebovali visoke koeficiente varnosti tudi zato, ker niso bile znane natančne lastnosti materialov in obremenitev. Takšni predimenzionirani izdelki so seveda zadoščali predpisanim zahtevam in uspešno opravljali svojo funkcijo dalj časa. Niso pa bili ti izdelki optimalni. V sodobnem svetu masovne proizvodnje ni več cilj dolga doba trajanja izdelka, ampak čim nižja cena izdelka. Kljub temu mora izdelek za določen čas zadoščati predpisanim zahtevam. Zato je vsak izdelek treba najprej optimirati in šele nato izdelati prototip. To je mogoče ob natančnem poznavanju materialnih lastnosti in z uporabo sodobnih programskih orodij CAE. Kljub temu da postopek še vedno delimo na dve fazi, prikazani na sliki 1.5, se v maloserijski proizvodnji še danes pogosto zadovoljimo le s prvo fazo.



Slika 1.5 – Postopek konstruiranja izdelka in uporaba CAE

Programi CAE temeljijo na razvitih numeričnih metodah za računalniško podprte analize. Najbolj pogosto uporabljena je metoda končnih elementov ali MKE (angl. *FEM* – *Finite Element Method*) [1.11]. Metoda omogoča iskanje približne rešitve parcialne diferencialne enačbe za problem robnih vrednosti. To rešitev poiščemo tako, da celotno področje problema razdelimo na manjše delčke, ki jih imenujemo končni elementi, v katerih z aproksimacijsko metodo poiščemo približno rešitev. Poenostavljeno gledano, metoda končnih elementov deluje podobno kot numerična integracija. Namesto natančnega določanja površine pod dano funkcijo pri analitičnem reševanju, pri numerični integraciji razdelimo interval na manjše delčke, v katerih določamo delne približne površine. Podobno kot pri numerični integraciji tudi pri metodi končnih elementov nikoli ne dobimo natančnega rezultata, ampak zgolj približek, ki je odvisen od števila končnih elementov in stopnje aproksimacije.

### 1.2.1 Predprocesiranje numeričnih modelov

V fazi predprocesiranja uporabnik pripravi numerični model problema za analizo. V to fazo običajno spada tudi modeliranje izdelka, čeprav je pogosto geometrijski model izdelan pred simulacijami na podlagi idejnih skic, predlogov oblikovalcev ali kot rezultat konstruiranja na osnovi preprostih preračunov. Velikokrat je treba obstoječ geometrijski model spreminjati in prilagajati za namene simulacije, včasih pa ga je treba v ta namen znova zgraditi. Tako imamo

lahko na primer na voljo prostorski CAD-model paličja, ki pa ga veliko bolj učinkovito analiziramo z uporabo paličnih končnih elementov, zato je treba celotno prostorsko geometrijo pretvoriti v palični ali žični model.

Pomembna je izbira vrste numerične analize, ki je odvisna od narave procesa, pogojev obratovanja in želenih rezultatov. Izberemo lahko statične ali dinamične analize, ki so lahko linearne ali nelinearne. Pri tem upoštevamo različne pojave med deli izdelka (npr. kontakti) ali v materialu izdelka (npr. razpoke). Od izbire analize je pogosto odvisen tudi izbor materialnih podatkov, vrste končnih elementov in vseh drugih podatkov analize.

Naslednje opravilo je običajno določitev materialnih podatkov. To je lahko zelo preprosto in zahteva vnos le nekaj konstant, lahko pa je precej kompleksno z vnosom številnih konstant, spremenljivk in tabelaričnih podatkov. Včasih je treba definirati posebne materialne modele v obliki posebej razvitih uporabniških programov, ki aktivno sodelujejo z glavnim programom za analizo po metodi končnih elementov.

Izbiri materiala tesno sledi izbor fizikalnih lastnosti modela. Običajno je treba določiti dimenzijo numeričnega problema in izbrati ustrezen tip končnega elementa, ki bo učinkovito modeliral želeni fizikalni model. Izbor fizikalnega 1D- ali 2D-modela je običajno bolj zahteven kot izbor 3D-modela.

Nato je treba določiti še robne pogoje. To pomeni določanje podpor, obremenitev modela in morebitne interakcije med posameznimi deli modela. Kot bomo videli v nadaljevanju, je pomembno, da določimo robne pogoje še pred določanjem mreže končnih elementov. Mreža končnih elementov se lahko spreminja med izvajanjem analiz, zato določitev robnih pogojev neposredno na končnih elementih zahteva dobro poznavanje teoretične zasnove končnega elementa, kot je prikazano v naslednjih poglavjih.

Glavnina dela v fazi predprocesiranja je izdelava mreže končnih elementov. Sem v prvi vrsti spada izbor vrste in velikosti končnih elementov. Čeprav je izdelava mreže končnih elementov danes skoraj popolnoma samodejna, je treba vložiti veliko dela, če želimo izdelati kvaliteten numeričen model, ki bo zagotavljal dovolj natančne rezultate. Pogosto je treba izdelati več mrež končnih elementov, da lahko določimo konvergenco rešitve problema. Gradnja mreže zahteva veliko napornega dela, ki obsega delitev geometrije na ustrezne dele za zagotavljanje kvalitetnejše mreže in pozneje tudi popraviljanje in prilaganje mreže glede na prve rezultate analiz. Gradnja mreže je delovno in časovno najbolj zahtevna operacija priprave analize po metodi končnih elementov.

### 1.2.2 Procesiranje numeričnih modelov

Ta faza je namenjena numerični analizi. Zaradi razvoja hitrih algoritmov za reševanje sistema enačb in hitrosti sodobnih računalnikov je ta faza časovno najkrajša, razen v primeru zahtevnih nelinearnih analiz. V tej fazi uporabnik praktično nima neposrednega dela, razen v primeru dolgotrajnih analiz, ko lahko spremlja potek le-teh, in v primeru, če ugotovi nepravilnosti (divergenca rezultatov, ugotovljene napake), analizo prekine.

### 1.2.3 Poprocesiranje numeričnih modelov

Rezultati numeričnih analiz so običajno zgolj številke. S programi za poprocesiranje jih lahko pretvorimo v slike in grafe. Na osnovi številskih in grafičnih rezultatov je treba izdelati poročilo, iz katerega so razvidni rezultati analize. Poprocesiranje zato obsega delo s programi za poprocesiranje in proučevanje številskih rezultatov, iz katerih je treba izluščiti bistvene informacije in znanje za rešitev problema, zaradi katerega je bila potrebna numerična analiza.

V tej fazi je tudi treba ugotoviti, kakšna je natančnost rezultatov in ali so dobljeni rezultati dejanska rešitev problema ali ne. Analize po metodi končnih elementov se lahko le približajo dejanskim rezultatom, saj rešujejo problem z aproksimacijo in vsebujejo številne možnosti napak v sistemu in uporabniških napak.

## 1.3 CAE-programi

Danes je na trgu veliko komercialnih in odprtokodnih programov za simulacije po metodi končnih elementov. Večina teh programov je tesno povezanih z grafičnimi programi za generiranje geometrije modela in orodji za pred- ter poprocesiranje. Nekateri programi so vključeni v kompleksne PLM-sisteme (angl. *Product Lifecycle Management*), nekateri pa so samostojna kompleksna orodja za analize po metodi končnih elementov.

### MSC NASTRAN

MSC (MacNeal-Schwendler Corporation) je podjetje, ki je že v šestdesetih letih dvajsetega stoletja razvijalo programsko opremo za NASO. MSC NASTRAN je produkt, namenjen za numerične analize po metodi končnih elementov. Njihov program za pred- in poprocesiranje se imenuje PATRAN. Sicer pa podjetje pokriva številna področja analiz, kot je razvidno z njihovih spletnih strani [1.12]. Ob tradicionalni kvaliteti strukturnih analiz, na kateri temelji program Nastran, velja izpostaviti tudi program za analizo dinamike teles, imenovan ADAMS in program MARC za zahtevne nelinearne analize.

### NX Nastran

Podjetje Siemens je lastnik PLM-sistema z imenom NX. Leta 2002 so po privatizaciji prejeli izvorno kodo programa Nastran, ki jo od takrat razvijajo pod imenom NX Nastran. Programski paket vključuje pred- in poprocesor, ki je nastal iz programa Femap. Le-tega je prvotno razvilo podjetje SDRC. Po združitvi podjetij SDRC in Unigraphics so pod okrilje Siemens prešli tudi številni drugi programi za numerične analize. Zato v podjetju ponujajo širok izbor programskih rešitev za simulacije [1.13].

### Nei Nastran

Nei Nastran [1.14] je še tretji program, ki je leta 2002 nastal iz izvorne kode programa Nastran. Program deluje v operacijskem sistemu Linux in uporablja različico programa Femap za pred- in poprocesiranje.

### ANSYS

Potem ko so Nastran razdelili na tri produkte, je postal ANSYS praktično vodilni programski paket za simulacije. V podjetju pokrivajo večino področij simulacij [1.15]. Program je organiziran kot enoten simulacijski sistem, v katerem lahko uporabljamo različna simulacijska

orodja. Čeprav so v vseh numeričnih analizah v svetovnem vrhu, je njihov paradni konj dinamična analiza tekočin ali CFD (angl. *Computational Fluid Dynamics*).

### ABAQUS

Program je precej razširjen v akademskih institucijah, vendar postaja vedno bolj priljubljen tudi v industriji. Potem ko je podjetje prešlo pod okrilje Dassault Systemes in se zdaj programski paket imenuje SIMULIA [1.16], je program vedno bolj prisoten tudi v industrijskem okolju. S programskim paketom SIMULIA je mogoče izvajati številne analize. Orodja so bodisi samostojna ali vključena v programski paket Catia.

### Solidworks Simulation

Čeprav ima Dassault Systemes že paket SIMULIA, pa v Solidworksu ponujajo program za simulacije, ki se je nekoč imenoval Cosmos in je bil tudi samostojen program za numerične analize. Danes je, kot eno izmed orodij, Simulation [1.17] vključen neposredno v Solidworks. Kljub temu je možno izvajati široko paleto numeričnih analiz s trdninami in tekočinami.

### Creo Simulate

Tudi podjetje PTC (Parametric Technology Corporation) ima svoj paket za računalniške simulacije. Včasih, ko se je njihov glavni programski paket imenoval ProEngineer, se je orodje za simulacije imenovalo Mechanica. Danes so njihovi izdelki poimenovani z besedo Creo in zato simulacije izvajamo v Creo Simulate [1.18].

### Autodesk Simulation

Tudi eno od največjih podjetij na področju računalniško podprtega konstruiranja ima svoj program za numerične simulacije, ki se imenuje Autodesk Simulation [1.19]. Pravzaprav gre za več izdelkov, ki so namenjeni računalniškim simulacijam trdnin in tekočin. Izpostaviti velja izdelek Moldflow, ki se uporablja za simulacijo brizganja in ohlajevanja oz. strjevanja plastike. Ker je veliko današnjih izdelkov iz takšnih materialov, so te simulacije izredno priljubljene.

### LS-Dyna

Program LS-Dyna [1.20] podjetja Livermore Software Technology Corporation v nasprotju z zgoraj predstavljenimi orodji ni namenjen za poljubne vrste numeričnih analiz, ampak je namenjen predvsem dinamičnim analizam. LS-Dyna vsebuje številne materialne modele in orodja za simulacijo časovno odvisnih dinamičnih pojavov v trdninah in tekočinah. Program je najbolj znan zaradi uporabe v analizah trkov (angl. *crash test*).

### Esi-group

Na začetku je bil razvit program, namenjen za simulacije trka letala v jedrsko elektrarno. Kmalu je iz tega programa nastal program za simulacije trkov v avtomobilski industriji, imenovan PAM-CRASH. Danes podjetje ponuja tudi orodja za simulacijo izdelave, kjer posebej izstopa program PAM-STAMP. Poleg tega so na voljo številna druga orodja za simulacijo proizvodnje in uporabe izdelkov [1.21].

### Code\_Aster

Program Code\_Aster je odprtokodni program, ki temelji na metodi končnih elementov. Program je bil prvotno razvit za potrebe francoskega elektrogospodarstva, ki pa ga je prenehalo razvijati in ga je dalo na voljo skupnosti uporabnikov, ki nadaljuje razvoj programa. Glavna težava uporabe programa je razumevanje francoskega jezika v programu. Program je združljiv



z drugimi odprtokodnimi programi, vendar za uspešno namestitvev in uporabo zahteva kar nekaj spretnosti uporabnika. Program Code\_Aster [1.22] je predvsem uporaben za analizo trdnin.

### SimScale

Program SimScale [1.23] je spletni program, ki omogoča pred- in poprocesiranje, prav tako pa tudi izvedbo računalniških simulacij, ki temeljijo na metodi končnih elementov. V modul za predprocesiranje je možno uvoziti različne zapise geometrij ali pa uporabiti kar spletni modelirnik, npr. OnShape [1.24]. Poglavitna prednost programa je njegova uporaba v spletnem brskalniku, brez kakršnekoli namestitve programske opreme, kar omogoča dostop do numeričnega modela tako rekoč kjerkoli. Tudi simulacija se izvede na lokalnih strežnikih podjetja, ki ponuja SimScale. Program je primeren je za izvedbo trdnostnih analiz ter analiz mehanike tekočin in prenosa toplote.

Zgoraj so naštetih nekateri najbolj priljubljeni komercialni in en odprtokodni program za simulacije po metodi končnih elementov. Teh programov je seveda še veliko več in jih je mogoče najti v seznamu [1.25].

## 1.4 Merski sistemi v CAE

V PLM-sistemih, ki vključujejo programe za računalniške simulacije, se merske enote določijo na začetku dela, ko uporabnik izbere globalni merski sistem. V takem sistemu je vse usklajeno z merskim sistemom konstruiranja, za konsistentnost enot skrbi računalniški program. Večji problem je v samostojnih programih za numerične simulacije, ki so neodvisni od merskega sistema enot. Uporabnik mora sam poskrbeti, da uporablja konsistenten sistem enot. V programu so vnesene samo številke za dolžine, obremenitve, materialne podatke, čas itd. Merske enote rezultatov so potem odvisne od merskih enot vnesenih podatkov. V takšnih programih zato nikoli ni podatka o uporabljenih enotah in lahko vidimo samo številčne vrednosti brez enot. Šele uporabnik v poročilu predstavi, v kakšnih enotah so zapisani podatki in rezultati.

Danes je v uporabi več merskih sistemov. Pri nas uporabljamo metrični ali mednarodni sistem enot SI [1.26], ki ga je prva uvedla francoska akademija znanosti že leta 1791. Kljub temu pa številne države vztrajajo na starih merskih sistemih, ki so velikokrat nekonsistentni, vendar se tradicionalno uporabljajo v določenih okoljih. Zato se pri interpretaciji podatkov lahko pojavljajo težave, saj lahko iste številke razumemo drugače v različnih merskih enotah. Ena izmed večjih napak se je zaradi tega zgodila leta 1999, ko je NASA izgubila vesoljsko plovilo, vredno 125 milijonov USD, saj so inženirji v podjetju Lockheed Martin uporabljali imperialni merski sistem, inženirji v agenciji NASA pa SI-merski sistem [1.27]. V preglednici 1.1 je prikazana primerjava dveh najpogosteje uporabljenih merskih sistemov. Seveda se v svetu uporabljajo še nekateri drugi merski sistemi, vendar je le SI-merski sistem konsistentno urejen in v skladu s fizikalnimi zakoni. Zato je dolgoročno mogoče pričakovati, da bo ta merski sistem nadomestil druge merske sisteme po celotnem svetu. Vendar je proces dolgotrajen in traja že od osemnajstega stoletja, saj pogosto prevlada tradicionalno mišljenje namesto racionalnega.

Preglednica 1.1 – SI- in imperialni merski sistem [1.27]

	SI	Imperialni sistem	Razmerje
Dolžina	m (mm)	ft (in)	1 ft = 0,3048 m
Čas	s	s	1 s = 1 s
Masa	kg (t)	slug (lbf-s <sup>2</sup> /in)	1 slug = 14,5939 kg
Sila	N	lbf	1 lbf = 4,448222 N
Temperatura	C	F	1 F = 5/9 °C
Hitrost	m/s (mm/s)	ft/s (in/s)	1 ft/s = 0,3048 m/s
Pospšek	m/s <sup>2</sup> (mm/s <sup>2</sup> )	ft/s <sup>2</sup> (in/s <sup>2</sup> )	1 ft/s <sup>2</sup> = 0,3048 m/s <sup>2</sup>
Kotni pospešek	rad/s	rad/s	1 rad/s = 1 rad/s
Gostota	kg/m <sup>3</sup> (t/mm <sup>3</sup> )	slug/ft <sup>3</sup> (lbf-s <sup>2</sup> /in <sup>4</sup> )	1 slug/ft <sup>3</sup> = 515,378 kg/m <sup>3</sup>
Moment	Nm (Nmm)	ft-lbf (in-lbf)	1 ft-lbf = 1,355818 Nm
Napetost, tlak, modul elastičnosti	Pa (MPa)	psi (psf)	1 psi = 6894,757 Pa
Energija, delo	J (Nm)	ft-lbf (in-lbf)	1 ft-lbf = 1,355818 J

Kljub uporabi SI-sistema enot pa to ne pomeni, da je uporabnik brez težav pri določanju enot. SI-merski sistem enot določa skladne enote, ki so v preglednici 1.1 navedene v prvi vrstici, npr. m, N in Pa. Pogosto pa uporabljamo tudi manjše ali večje enote. V strojništvu pogosto za dolžinske enote uporabljamo mm, v gradbeništvu ali meteorologiji pa km. Tako uporabljene enote niso usklajene in izvedene veličine imajo lahko posledično različne vrednosti, kot to prikazuje preglednica 1.2, kjer so za primer prikazane različne kombinacije uporabljenih enot in materialnih podatkov za jeklo.

Preglednica 1.2 – Kombinacija enot in materialnih podatkov [1.28]

Masa	Dolžina	Čas	Sila	Napetost	Energija	Gostota	Modul elastičnosti	Gravitacijski pospešek
kg	m	s	N	Pa	J	7,83e+03	2,07e+11	9,806
kg	cm	s	1,0e-02 N			7,83e-03	2,07e+09	9,806e+02
kg	cm	ms	1,0e+04 N			7,83e-03	2,07e+03	9,806e-04
kg	cm	μs	1,0e+10 N			7,83e-03	2,07e-03	9,806e-10
kg	mm	ms	kN	GPa	kN-mm	7,83e-06	2,07e+02	9,806e-03
g	cm	s	dyne	dyne/cm <sup>2</sup>	erg	7,83e+00	2,07e+12	9,806e+02
g	cm	μs	1,0e+07 N	Mbar	1,0e+07 Ncm	7,83e+00	2,07e+00	9,806e-10
g	mm	s	1,0e-06 N	Pa		7,83e-03	2,07e+11	9,806e+03
g	mm	ms	N	MPa	N-mm	7,83e-03	2,07e+05	9,806e-03
tona	mm	s	N	MPa	N-mm	7,83e-09	2,07e+05	9,806e+03
lbf-s <sup>2</sup> /in	in	s	lbf	psi	lbf-in	7,33e-04	3,00e+07	386
slug	ft	s	lbf	psf	lbf-ft	1,52e+01	4,32e+09	32,17
kgf-s <sup>2</sup> /mm	mm	s	kgf	kgf/mm <sup>2</sup>	kgf-mm	7,98e-10	2,11e+04	9,806e+03
kg	mm	s	mN	1,0e+03 Pa		7,83e-06	2,07e+08	9,806e+03
g	cm	ms	1,0e+1 N	1,0e+05 Pa		7,83e+00	2,07e+06	9,806e-04

## 1.5 Literatura

- [1.1] Excel 2016 [splet], Dosegljivo: <https://products.office.com/sl-si/excel>. [Datum dostopa 5. 12. 2017].
- [1.2] OpenOffice [splet], Dosegljivo: <http://www.openoffice.us.com/openoffice-calc.php>. [Datum dostopa 5. 12. 2017].
- [1.3] Visual Basic for Applications [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Visual\\_Basic\\_for\\_Applications](https://en.wikipedia.org/wiki/Visual_Basic_for_Applications). [Datum dostopa 5. 12. 2017].
- [1.4] Online technical calculators [splet], Dosegljivo: <http://www.craftsmanspace.com/free-software/online-technical-calculators.html>. [Datum dostopa 5. 12. 2017].
- [1.5] Mechanical, Industrial and Technical Calculations [splet], Dosegljivo: <http://www.mitcalc.com/index.htm>. [Datum dostopa 5. 12. 2017].
- [1.6] Web-based calculation software for mechanical engineering [splet], Dosegljivo: <http://www.eassistant.eu/en/home.html>. [Datum dostopa 5. 12. 2017].
- [1.7] Design software for mechanical engineering applications [splet], Dosegljivo: <http://www.kisssoft.ch/english/home/index.php>. [Datum dostopa 5. 12. 2017].
- [1.8] SAM Mechanismen-ontwerp door Artas Engineering Software [splet], Dosegljivo: <http://www.artas.nl/nl/>. [Datum dostopa 5. 12. 2017].
- [1.9] Adams – The Multibody Dynamics Simulation Solution [splet], Dosegljivo: <http://www.mscsoftware.com/product/adams>. [Datum dostopa 5. 12. 2017].
- [1.10] Information Technology Model for Product Lifecycle Engineering [splet], Dosegljivo: <http://airccj.org/CSCP/vol3/csit3652.pdf>. [Datum dostopa 5. 12. 2017].
- [1.11] Finite element method [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Finite\\_element\\_method](https://en.wikipedia.org/wiki/Finite_element_method). [Datum dostopa 5. 12. 2017].
- [1.12] MSC Software Applications [splet], Dosegljivo: <http://www.mscsoftware.com/applications>. [Datum dostopa 5. 12. 2017].
- [1.13] NX CAE [splet], Dosegljivo: [http://www.plm.automation.siemens.com/en\\_us/products/nx/for-simulation/](http://www.plm.automation.siemens.com/en_us/products/nx/for-simulation/). [Datum dostopa 5. 12. 2017].
- [1.14] NEi Nastran [splet], Dosegljivo: [https://en.wikipedia.org/wiki/NEi\\_Nastran](https://en.wikipedia.org/wiki/NEi_Nastran). [Datum dostopa 5. 12. 2017].
- [1.15] ANSYS [splet], Dosegljivo: <http://www.ansys.com/Products>. [Datum dostopa 5. 12. 2017].
- [1.16] SIMULIA [splet], Dosegljivo: <http://www.3ds.com/products-services/simulia/products/>. [Datum dostopa 5. 12. 2017].
- [1.17] SOLIDWORKS [splet], Dosegljivo: <http://www.solidworks.com/sw/products/simulation/packages.htm>. [Datum dostopa 5. 12. 2017].

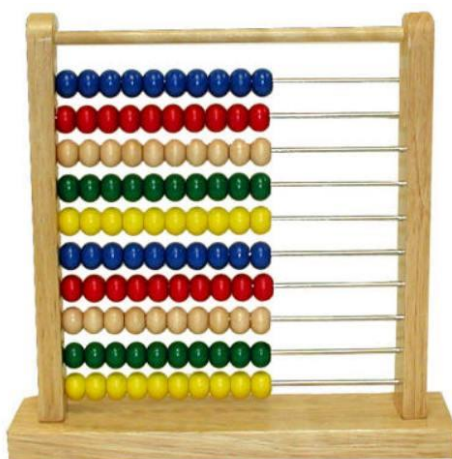
- 
- [1.18] PTC – CAD Software Solutions [splet], Dosegljivo: <https://www.ptc.com/en/products/cad>. [Datum dostopa 5. 12. 2017].
- [1.19] AUTODESK [splet], Dosegljivo: <http://www.autodesk.com/products/simulation/overview>. [Datum dostopa 5. 12. 2017].
- [1.20] LS-DYNA [splet], Dosegljivo: <http://www.lstc.com/products/ls-dyna>. [Datum dostopa 5. 12. 2017].
- [1.21] ESI – Software Solutions [splet], Dosegljivo: <https://www.esi-group.com/software-services>. [Datum dostopa 5. 12. 2017].
- [1.22] Code Aster[splet], Dosegljivo: <http://www.code-aster.org/spip.php?rubrique2>. [Datum dostopa 5. 12. 2017].
- [1.23] SIMSCALE [splet], Dosegljivo: <https://www.simscale.com/>. [Datum dostopa 5. 12. 2017].
- [1.24] Onshape [splet], Dosegljivo: <https://www.onshape.com/>. [Datum dostopa 5. 12. 2017].
- [1.25] List of finite element software packages [splet], Dosegljivo: [https://en.wikipedia.org/wiki/List\\_of\\_finite\\_element\\_software\\_packages](https://en.wikipedia.org/wiki/List_of_finite_element_software_packages). [Datum dostopa 5. 12. 2017].
- [1.26] International System of Units [splet], Dosegljivo: [https://en.wikipedia.org/wiki/International\\_System\\_of\\_Units](https://en.wikipedia.org/wiki/International_System_of_Units). [Datum dostopa 5. 12. 2017].
- [1.27] Tips & Tricks for Consistent FEA Units [splet], Dosegljivo: <https://caesai.com/blog/tips-tricks-consistent-fea-units>. [Datum dostopa 5. 12. 2017].
- [1.28] Consistent units [splet], Dosegljivo: <http://www.dynasupport.com/howtos/general/consistent-units>. [Datum dostopa 5. 12. 2017].

## 2. Zgodovinski pregled računalniških simulacij

V procesu evolucije se je pri sesalcih razvil del možganov, ki ga imenujemo neokorteks [2.1]. Ta del je odgovoren za višje funkcije delovanja organizma, med katere spadajo tudi simulacije. Simulacije imajo posebej pomembno vlogo pri človeku, kjer neokorteks obsega kar 80 % možganov. Skoraj pri vsakem opravilu, ki ga počnemo prvič, najprej opravimo simulacijo v mislih in opravilo navidezno opravimo. Po lastnih izkušnjah vemo, da rezultati simulacije niso vedno enaki kot pri resničnem opravilu, vendar nas simulacija bolje pripravi na odzive v realnosti. Bolj kompleksna opravila zahtevajo seveda bolj kompleksne simulacije in hitro pridemo do opravil, ki jih ni mogoče simulirati samo v glavi. Seveda je mogoče opravilo opraviti tudi s preizkusi, vendar so ti zelo dragi. Zato so ljudje razvili pripomočke, ki omogočajo boljše simulacije realnega sveta.

### 2.1 Matematika in fizika

Osnovno orodje za simulacije je matematika oz. računanje. Zgodovinsko je bil prvi številski sistem kvazišestdesetiški [2.2] in ta številski sistem je prisoten še danes v načinu beleženja časa. Vsaka ura ima 60 minut, vsaka minuta 60 sekund. Tak številski sistem in takšno merjenje časa sta zabeležena pri Sumerjih, kjer so tudi leto razdelili na 360 dni. Zakaj ravno šestdesetiški sistem? Vsi vemo, da je najlažje šteti s prsti, ki jih je 10. Nekateri omenjajo štetje s palcem, pri katerem lahko na eni roki preštejemo dvanajst členkov. Vendar je bolj verjetno dejstvo, da ima število šestdeset natančno dvanajst deliteljev. Dan je zato razdeljen na dvanajst ur (šest ur dopoldan in šest ur popoldan) in noč na dvanajst ur. Število šestdeset je imelo pri Sumerjih poseben pomen in verjetno je ta pomen še starejši, saj je bil ta številski sistem že v prazgodovini globalen in so ga uporabljali tudi Kitajci. Glavna slabost tega številskega sistema je, da ne vsebuje ničle kar zavira razvoj matematike in onemogoča zapis velikih števil.



Slika 2.1 – Desetiški abakus

Seveda je tudi težko šteti samo s prsti, zato so začeli šteti in računati s kamenčki in že Sumerci so izumili primitiven abakus [2.3], ki je omogočal seštevanje in odštevanje v šestdesetiškem sistemu. V zgodovini se je abakus prilagajal različnim številskim sistemom in danes si je najlažje predstavljati delo z abakusom na sliki 2.1, ki deluje v desetiškem številskem sistemu.

Pri tem lahko spodnja vrstica predstavlja enice, ena vrstica više desetice, nato stotice itd. Seštevamo in odštevamo s premikom kamenčkov, in ko se zapolni celotna vrstica, jo premaknemo v izhodiščni položaj ter premaknemo en kamenček v zgornji vrstici. Abakus predstavlja osnovno računalno že tisoče let in se še vedno uporablja v nekaterih delih sveta. Zaradi masovne proizvodnje je danes cena elektronskega kalkulatorja lahko nižja od abakusa, zato ga pogosteje najdemo med otroškimi igračami kot med učnimi pripomočki.

Prsti, kamenčki in abakus omogočajo začasen zapis števil, vendar je veliko bolj uporaben trajen zapis, ki je bil možen ob razvoju pisave. Sumerci so znali zapisati vsa števila od 1 do 59 [2.2], simbol za število 60 pa je bil enak simbolu za število 1. Sicer so vsi simboli za števila od 1 do 59 sestavljeni samo iz dveh osnovnih simbolov. Zato je bil sumerski zapis števil precej nejasen in dvoumen. Egipčani [2.4] so uporabljali precej več simbolov za zapis števil, ki je bil zato precej bolj pregleden. Poleg tega so zapisovali tudi racionalna števila. Število  $\pi$ , danes znano kot Ludolfovo število, je imelo v starem Egiptu vrednost  $4 \cdot (8/9)^2$  [2.5], vendar se to ni razširilo in so še dolgo uporabljali celo število 3 za izračun površine in obsega kroga.

V stari Grčiji so povzeli znanje Sumercev in Egipta ter razvili matematiko in fiziko na področjih geometrije, algebre, trigonometrije, astronomije in fizike. V tistem času so uspeli izračunati velikost Zemlje in določiti razdaljo med Zemljo in Soncem [2.6]. Izračuni z večjo natančnostjo so bili mogoči šele z razvojem moderne znanosti po renesansi. Arheološke najdbe mehanizmov z zobniki nakazujejo, da so v tistem času imeli naprave, s katerimi so simulirali gibanje zvezd za napovedovanje različnih astronomskih pojavov.

Rimljani so prevzeli grško znanost in izpopolnili inženirske dosežke. Rimski zapis števil je bil še vedno brez ničle in brez decimalk in ni omogočal zahtevnih izračunov. Obseg imperija in učinkovita administracija sta imela velik vpliv na razvoj znanosti do današnjih dni. Znana je analiza vpliva širine zadnjice rimskega konja na velikost vesoljskega čolnička [2.7]. Rimljani so standardizirali medkolesno razdaljo na kočijah glede na povprečno širino konjskih zadnjic in vse kočije so zato imele enak kolotek. To se je ohranilo vse do 19. stoletja, ko je širina železniških tirov ostala enaka kot kolotek kočij. Rakete na trdo gorivo, ki so bile nameščene na vsaki strani vesoljskega čolnička, so prevažali z železnico in zato je bila njihova širina določena s širino tirov.

Brahmi	↓		—	=	≡	+	∞	∞	∞	∞	∞
Hindu	↓	०	१	२	३	४	५	६	७	८	९
Arabic	↓	•	١	٢	٣	٤	٥	٦	٧	٨	٩
Medieval	↓	0	1	2	3	4	5	6	7	8	9
Modern		0	1	2	3	4	5	6	7	8	9

Slika 2.2 – Evolucija decimalnega številkega sistema

Arabci, ki so zasedli Bližnji vzhod, so imeli trgovske stike z Indijo in so od tam prevzeli nov decimalni številski sistem. Za razliko od rimskega, ki je številke zapisoval s sedmimi črkami (I, V, X, L, C, D, M), je arabski sistem obsegal 10 znakov in je bil pravi desetiški (decimalni) sistem, prikazan na sliki 2.2. Bistven element tega sistema je ničla, ki omogoča lažji zapis velikih števil in omogoča razvoj matematičnih zakonitosti.

Okoli leta 1200 je Leonardo Fibonacci [2.8] v svoji knjigi uporabil arabske številke in opisal številne matematične operacije in zakonitosti (najbolj znano je Fibonaccijevo zaporedje). Decimalen številski sistem omogoča, da položaj znaka določa potenco števila 10. To omogoča zapis enic, desetic, stotic itd. od desne proti levi in desetink, stotink, tisočink itd. od decimalne vejice proti desni. Decimalna števila za vrednosti manj od 1 so prvi izumili na Kitajskem in okoli 10. stoletja se je pojavil prvi prevod kitajskega dela z decimalnimi števili v arabski literaturi. V Evropi so nato povzeli in izumili decimalni zapis vrednosti na desni strani vejice, kot ga poznamo danes, vendar enotnega zapisa še do danes ni. Že v začetku 17. stoletja je bilo veliko različnih oblik zapisa, kot je prikazano v tabeli 2.1. John Napier, ki je prvi uporabil zapis s piko in vejico, je v isti knjigi uporabil oboje. Ker so kasneje v Evropi uporabljali piko za množenje, je ostala vejica decimalno ločilo. V državah, kjer kot decimalno ločilo uporabljajo piko (.), pa za množenje uporabljajo drug simbol ( $\times$  ali  $*$ ) [2.9].

Preglednica 2.1 – Zapis realnega števila z decimalnim ločilom

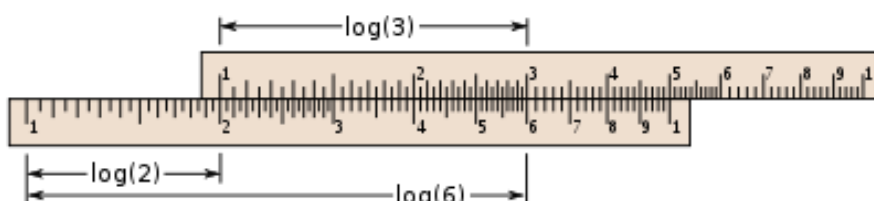
Avtor	Leto	Zapis
Prej		$37 \frac{245}{1000}$
Simon Stevin	1585	$37 2^{(1)} 4^{(2)} 5^{(3)}$
Franciscus Viete	1600	$37 _{245}$
Johannes Kepler	1616	$37(245$
John Napier	1617	$37,2^I 4^{II} 5^{III}$ $37.2^I 4^{II} 5^{III}$
Henry Briggs	1624	$37^{245}$
William Ouhtred	1631	$37 \underline{245}$
Richard Balam	1653	$37:245$
Sodoben zapis	1880–	$37.245$ (Združene države) $37,245$ (Evropa)

Najpomembnejši izum Johna Napierja [2.10] pa so logaritmi, ki omogočajo enostavnejše in hitrejše računanje (npr. seštevanje namesto množenja). Izdelal je tudi napravo za računanje, imenovano Napierjeve kosti [2.11]; prikazano je na sliki 2.3.



Slika 2.3 – Napierjeve kosti

Edmund Gunter [2.12] je izkoristil lastnosti logaritmov in okoli leta 1624 prvi izdelal logaritemsko računalno [2.13]. Logaritemsko računalno ali pomično računalno (angl. *sliderule*, *slipstick*; nem. *Rechenschieber*) omogoča enostavno množenje in deljenje, saj temelji na logaritemskih tabelah. Zahtevnejše aplikacije omogočajo tudi korenjenje, potence in trigonometrične operacije. Na sliki 2.4 je prikazan enostaven princip množenja ( $2 \cdot 3 = 6$ ). Uporaba računalala vseeno ni tako preprosta, vendar so z nekaj vaje uporabniki računali mnogo hitreje kot s pisnim računanjem.

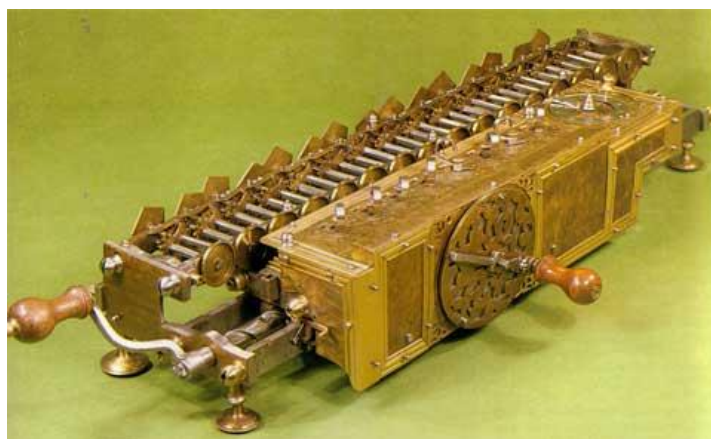


Slika 2.4 – Množenje z logaritemskim računalom

Logaritemsko računalno je bilo nato 300 let osnovno orodje vsakega navigatorja in inženirja. Okoli leta 1970, ko se pojavijo prvi elektronski kalkulatorji, pa je logaritemsko računalno postopoma izginilo iz uporabe in danes ga znajo uporabljati le redki posamezniki. Zanimiv vpogled v pomembnost logaritemskih računal nam omogoča branje zgodnjih znanstveno fantastičnih del, ko so glavni junaki na medzvezdnem potovanju računali trajektorije z logaritemskim računalom. Logaritemsko računalno je danes razmeroma težko najti, zato pa je na voljo virtualno logaritemsko računalno [2.14], na katerem je mogoče preizkusiti način delovanja.



Že leta 1623 je Wilhelm Schickard [2.15] izdelal mehanski kalkulator oz. računsko uro. Blaise Pascal [2.16] je nato leta 1645 predstavil svoj kalkulator Pascaline (pred tem je izdelal okoli 50 prototipov). Za razliko od logaritemskega računalja je ta kalkulator dejansko omogočal samodejno seštevanje in odštevanje. Leta 1675 je Gottfried Wilhelm Leibniz [2.17] izdelal nov računski stroj, ki je ob seštevanju in odštevanju omogočal tudi avtomatsko množenje in deljenje. Leibniz, ki je poleg tega pomagal razviti infinitezimalni račun (limito, odvod, integral), je bil zelo zavzet za razvoj računskega stroja in je za izdelavo strojev potrošil ogromno denarja. Verjetno je bilo izdelanih vsaj 10 strojev, ki pa niso nikoli uspeli v praksi, saj je bilo računanje z logaritmskimi računalji veliko hitrejše in bolj priročno. Leibnizev stroj, prikazan na sliki 2.5, je bil precej manj mobilan in zelo zahteven za uporabo.



Slika 2.5 – Replika Leibnizevega kalkulatorja

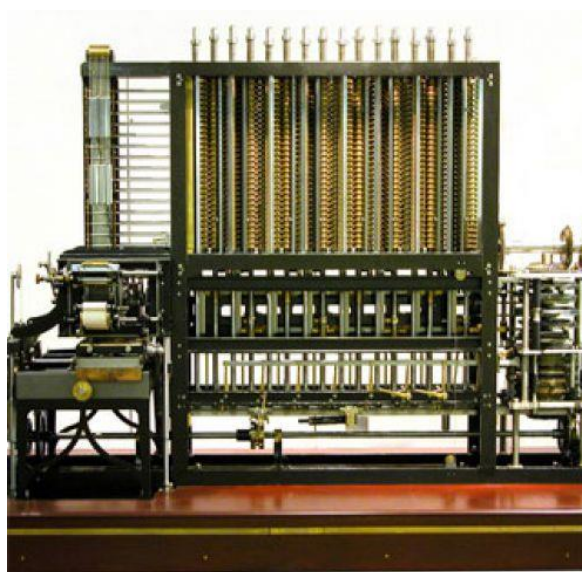
Problem računskih strojev v 17. stoletju je bil le delno problem matematike in bolj problem inženirstva, ki ni bilo doraslo potrebam računskih strojev. Inženirstvo se je v renesansi v 15. stoletju začelo razvijati z Leonardom da Vincijem [2.18] in je sprva razmeroma počasi napredovalo, saj je bilo treba odkriti ali znova odkriti osnovne fizikalne zakone. Čeprav je bil Leonardo da Vinci s svojimi izumi verjetno res prvi inženir nove dobe, pa je dejansko znanstvene temelje fizike v novi dobi postavil Galileo Galilei [2.19]. Galileo je eksperimentalno določil zakone prostega pada in metov teles ter zakon o vztrajnosti. Do takrat je veljala Aristotelova domneva, da telesa z večjo maso padajo hitreje. Galileo je to ovrgel s preprostim poskusom dveh teles, povezanih z vrvico.

Že nekaj let po Galilejevi smrti je leta 1660 Robert Hooke [2.20] zapisal zakon, ki določa odnos med silo in pomikom uteži na vzmeti. Ta zakon je danes znan kot Hookov zakon proporcionalnosti. Praktično vse simulacije elastičnih teles temeljijo na tem zakonu. Njegov sodobnik je bil Isaac Newton [2.21], ki je definiral osnovne zakone gibanja in temelje klasične mehanike, t. i. Newtonove fizike. Njegovi prispevki na področju fizike in matematike so še danes zelo aktualni in zato v računalniških simulacijah pogosto uporabljamo njegove metode. Pogosto zasledimo njegov priimek skupaj s priimkom Euler. Leonhard Euler [2.22] je bil v osnovi mehanik in je v 18. stoletju napisal številna znanstvena dela, ki se v veliki meri uporabljajo v sodobnih metodah za računalniške simulacije. Najbolj znane so njegove enačbe za analitično

ugotavljanje uklona stebra in upogib nosilca. Enačbo za upogib nosilca je razvil skupaj z Danielom Bernoullijem [2.23], ki je poleg številnih del na področju hidrodinamike razvil tudi teorijo virtualnega dela, na kateri prav tako temelji metoda končnih elementov. Euler je razvil tudi idejo o modulu elastičnosti, ki jo je formalno prvi dokumentiral Thomas Young [2.24] leta 1807. Od takrat je bil v uporabi modul elastičnosti ali Youngov modul; inženirji so prvič imeli na voljo vrednost, ki je določala razmerje med napetostjo in specifično deformacijo ter je bila odvisna samo od materiala. Mnogo numeričnih metod v simulacijah je v 19. stoletju izdelal Johann Carl Friedrich Gauss [2.25]. Najpomembnejša je seveda Gaussova integracija, ki je zaradi velike učinkovitosti vgrajena v sodobne računalniške programe za simulacije. Leta 1827 je Simeon Denise Poisson [2.26] ugotovil, da se material pod vplivom zunanje obremenitve različno deformira v smeri pravokotno glede na smer obremenitve [2.27]; razmerje teh deformacij danes imenujemo Poissonovo razmerje.

## 2.2 Računalniki in programiranje

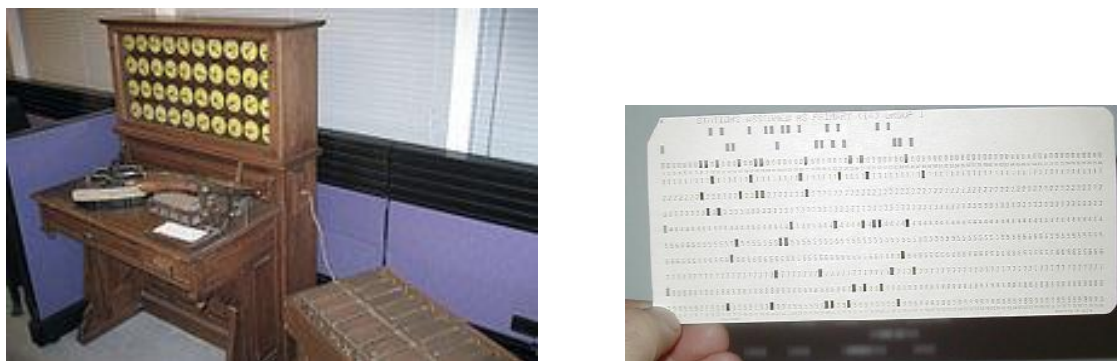
V 19. stoletju industrijske revolucije sta znanost in tehnika zelo hitro napredovali. Charles Babbage [2.28] je razvil parni diferencialni stroj za izdelavo matematičnih tabel. Načrtoval je tudi analitični stroj, ki bi bil prvi dejanski računalnik, če bi ga bilo možno izdelati. Kvaliteta izdelave pa je bila v 19. stoletju še vedno nezadostna za izdelavo mehanskega računalnika. Kljub temu je bilo izdelanih kar nekaj različic diferencialnih strojev (slika 2.6), ki so skupaj z načrti za analitični stroj imeli velik vpliv na razvoj elektronskih računalnikov v 20. stoletju. Razvoj elektronike je seveda omogočal veliko bolj racionalne rešitve, zato danes niso več smiselni mehanski računalniki razen v okolju, kjer je preveč motenj za njihovo delovanje (visoka temperatura, radiacija). Zato od leta 2011 poteka projekt izdelave analitičnega stroja oz. mehanskega računalnika po izboljšanih Babbagevih načrtih, ki bi naj bil končan leta 2021.



Slika 2.6 – Replika diferencialnega stroja

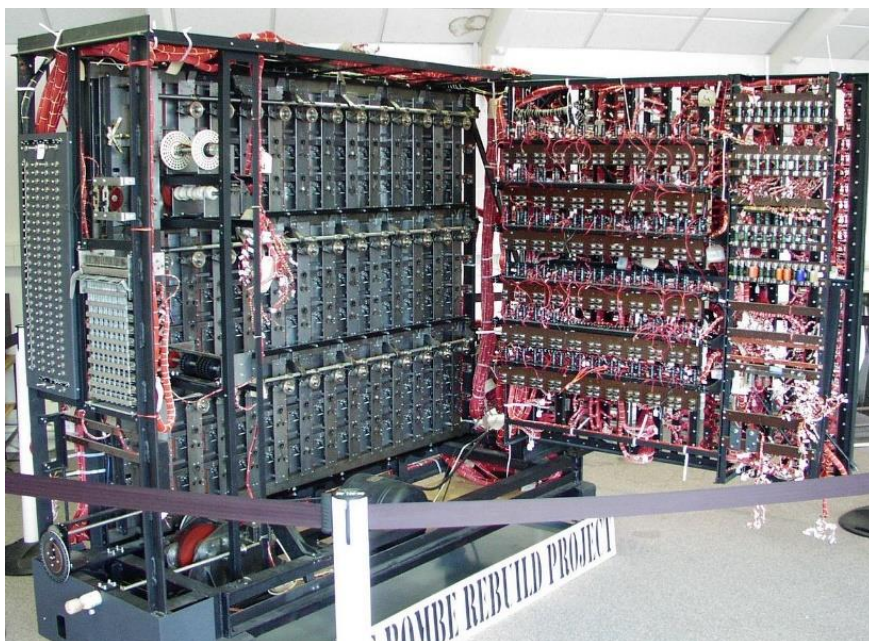
Poleg izdelave prvega računalnika je Babbage postavil tudi temelje za računalniške programe, saj je stroj moral delovati po navodilih uporabnika. Verjetno je Babbage napisal prve računalniške programe za njegov diferencialni in analitični stroj, vendar je prvi opis teh in še nekaterih programov napisala v članku Ada Lovelace [2.29], ki tako velja za prvo programerko.

Podatki in programi so se v stroj vnašali v obliki luknjanih kartic, ki so jih v tekstilni industriji uporabljali že v 18. stoletju. Leta 1889 je Herman Hollerith [2.30] patentiral stroj za električno obdelavo podatkov (slika 2.7). Stroj je deloval s Hollerithovimi luknjanimi karticami, ki so se nato uporabljale v računalnikih skoraj do konca 20. stoletja. S strojem so leta 1890 opravili popis prebivalstva v ZDA v enem letu namesto v osmih letih, kolikor je trajal popis prebivalstva desetletje prej. Po uspehu stroja je Hollerith leta 1911 ustanovil podjetje CTR (Computing Tabulating Recording), ki se je leta 1924 preimenovalo v IBM (International Business Machines).



Slika 2.7 – Električni tabelarni stroj in Hollerithova luknjana kartica

V drugi svetovni vojni so Nemci za kodiranje sporočil uporabljali napravo, imenovano Enigma. Naprava je omogočala  $10^{19}$  oz.  $10^{22}$  različnih nastavitvev za kodiranje sporočila. Nastavitve kodirne naprave so Nemci spremenili vsak dan. Zato je bilo ročno dešifriranje sporočil praktično nemogoče. Alan Turing [2.31], ki se je že pred vojno ukvarjal z zasnovo in gradnjo računalnikov, je za dešifriranje Enigme izdelal elektromehanski (releji) računalnik, imenovan Bombe (slika 2.8). Računalnik je uspel dekodirati šifro v nekaj urah in zavezniki so lahko od leta 1942 pa do konca vojne dešifrirali vsa nemška sporočila. Turing je že leta 1936 zasnoval koncept Turingovega stroja [2.32]. Turingov stroj v teoriji izvaja ukaze na neskončnem traku. Ukazi so običajno matematične operacije, vendar mora stroj omogočati tudi premike po traku in odločitve. Bombe je bil namenski računalnik za dešifriranje Enigme in ni bil Turingov stroj ali računalnik, kot jih poznamo danes.



Slika 2.8 – Replika Turingovega računalnika za dešifriranje



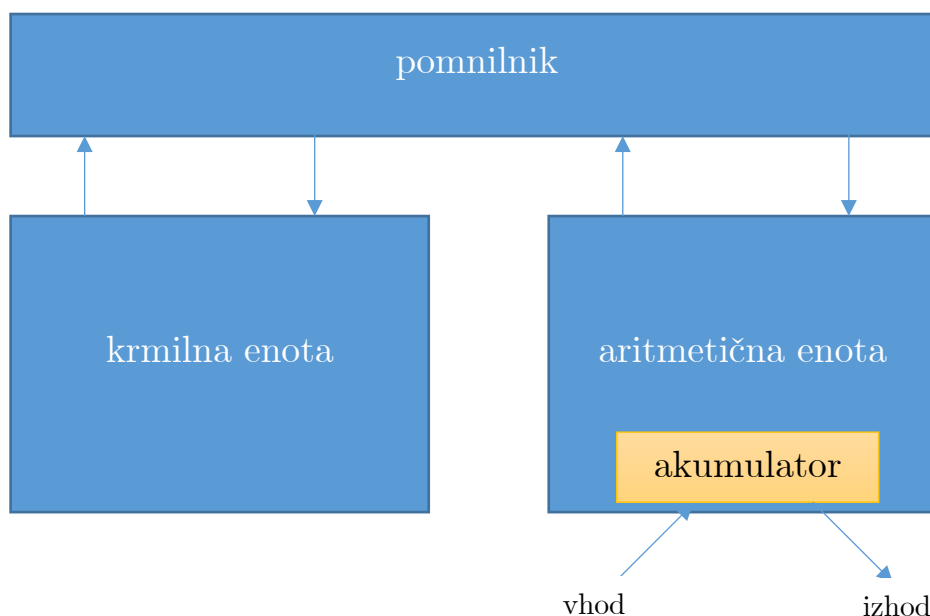
Slika 2.9 – Replika računalnika Z3

Prvi Turingov stroj je bil elektromehanski računalnik Z3, ki ga je izdelal Konrad Zuse [2.33] leta 1941 v Nemčiji. Leta 1945 je izdelal še verzijo Z4. Po vojni je vse avtorske pravice odkupilo podjetje IBM. Na sliki 2.9 je prikazana replika računalnika Z3, ki so ga izdelali po vojni, saj je bil originalni računalnik uničen med bombardiranjem Berlina.

Leta 1944 so v ZDA izdelali elektromehanski računalnik Harvard Mark 1 – IBM ASCC. Problema elektromehanskih računalnikov sta bila glasnost delovanja in velika poraba energije. Poleg tega se je zgodilo, da so med kontakte relejev prišli hrošči, ki so preprečili stik. To je povzročalo napake v delovanju, ki jih pri računalnikih še danes označujemo z istim imenom (napaka = hrošč oz. angl. *bug*). Zato je bilo pri elektromehanskih računalnikih potrebno razhroščevanje (angl. *debugging*) in tudi ta izraz se je ohranil do danes.

Kmalu so elektromehanske računalnike zamenjali elektronski, ki so namesto relejev vsebovali elektrone. Čeprav je bilo obdobje elektronk v računalniku dolgo le dobrih 10 let in so jih že po letu 1955 zamenjali tranzistorji ter kasneje silikonske rezine (angl. *chip*), še vedno uporabljamo izraz elektronski računalniki.

Leta 1946 je Alan Turing objavil tudi članek s konceptom računalnika, ki vsebuje programsko kodo in podatke v pomnilniku, vendar je že leta 1945 podoben članek (vendar manj natančen) objavil John von Neumann [2.34], ko je opisoval koncept elektronskega računalnika EDVAC. Danes vsi računalniki temeljijo na konceptu von Neumana (slika 2.10), delno tudi zato, ker je objava njegovega članka onemogočila patentno zaščito arhitekture računalnika EDVAC.



Slika 2.10 – Von Neumanova arhitektura računalnika

Razvoj strojne računalniške opreme se je nato bliskovito nadaljeval, in sicer najprej zaradi vojaških potreb v hladni vojni, nato pa zaradi konkurence na tržišču. Leta 1965 je Gordon Moore, soustanovitelj podjetja Intel, zapisal Mooreov zakon [2.35], ki pravi, da se zmogljivost računalnikov podvoji vsako leto oz. vsaki dve leti, kot se je Moore kasneje sam popravil. Rast se trenutno upočasnjuje. Leta 2015 je bila podvojitvev zmogljivosti zabeležena na vsaki dve leti in pol.

Zmogljivost računalnika ni odvisna le od strojne opreme, ampak tudi od programske opreme. Alan Turing je v petdesetih letih napisal program za igranje šaha, vendar ni bilo računalnika, na katerem bi se lahko program izvajal. Zato je program izvajal ročno, da je preizkusil delovanje. Program je deloval, vendar je za vsako potezo potreboval približno uro in pol računanja. Danes je situacija obratna, saj so računalniki precej zmogljivi, programska oprema pa ne zmore izkoriščati vseh sposobnosti strojne opreme.

Programsko opremo prvih računalnikov so predstavljali direktni ukazi za strojno opremo. Danes temu rečemo strojni jezik, ki je sestavljen iz števil, ki označujejo ukaze, in števil, ki označujejo podatke. Večina računalnikov temelji na binarnem številskem sistemu (le redki računalniki v zgodovini so bili desetiški), zato so bile številke običajno zapisane v binarnem številskem sistemu. Programerji so uporabljali tudi osmiški in šestnajstiški zapis, ker so bile računalniške besede dolge štiri ali osem bitov. Seveda je takšno kodiranje primerno le za posebej usposobljene programerje. Poleg tega je vsak izdelan računalnik uporabljal drugačne ukaze in je bilo nemogoče prenesti program iz enega računalnika na drugega brez ponovnega kodiranja. Številčne nize so kmalu zamenjali nizi črk, ki so označevali določen ukaz. Tak jezik se je imenoval zbirni jezik (angl. *assembler*) in pred izvajanjem je bilo treba ukaze pretvoriti v strojno kodo.

Zaradi tega so bili zelo hitro izdelani prvi programski jeziki, ki so omogočali programiranje na bolj abstraktnem nivoju. Programski jezik ima, podobno kot naravni jezik, določen nabor besed in pravopisna pravila. Programski jeziki so izdelani tako, da je mogoče zapisati algoritem problema tako, da ga bo mogoče učinkovito rešiti z računalnikom. Orodje, ki se imenuje prevajalnik, nato zapis programskega jezika prevede v strojno kodo računalnika, kjer se program potem izvaja. Na ta način je mogoče napisati samo en program, ki se nato prevede na več različnih računalnikih. Seveda še danes obstaja veliko specifičnih značilnosti za različno strojno opremo, zato so še vedno potrebne prilagoditve pri prenosu programske opreme na različne platforme strojne opreme.

Prvi prevajalnik je bil izdelan za programski jezik FORTRAN v podjetju IBM z namenom neposrednega prevajanja matematičnih formul v računalniški program (angl. *FORmula TRANslating system*). Temu je sledila prava poplava različnih jezikov, ki se razlikujejo v ukazih in obliki zapisa [2.36]. Tudi danes je FORTRAN še vedno prisoten programski jezik, čeprav bolj dominirajo programski jeziki, ki temeljijo na programskem jeziku C (C, C++, C#). Razvoj programskih jezikov še zdaleč ni končan, saj se programski jeziki spreminjajo in prilagajajo tako kot naravni jeziki.

To je omogočilo programiranje računalnikov na vseh področjih, kjer so se uporabljali računalniki. Večina računalnikov v prvem obdobju je bila namenjena neposredno ali posredno za vojaške potrebe. Zato ni čudno, da so prvi programi za simulacije nastali v letalski industriji [2.37]. Leta 1956 je bil prvič omenjen termin metoda končnih elementov (angl. *FEM – Finite Element Method*) v članku [2.38]. Avtorji članka so delali v podjetju Boeing in niso izumili metode, ampak so samo poimenovali metodo in jo uporabili na takratnih računalnikih. Tako so se simulacije na računalnikih uveljavile zaradi inženirskih potreb po simulacijah in zaradi računalniške strojne opreme, ki je bila na voljo.

V naslednjih desetletjih so številni raziskovalci postavili prave znanstvene temelje metode končnih elementov [2.39] in jo izpopolnili. Seveda razvoj še ni končan in metoda se ves čas izboljšuje in dopolnjuje. Poleg te metode so bile razvite še številne druge metode za simulacije. Tukaj velja omeniti metodo robnih elementov [2.40], metodo končnih volumnov [2.41] in brez mrežne metode [2.42]. Kljub temu se metoda končnih elementov zaradi svoje zrelosti še vedno največ uporablja za računalniške simulacije naravnih pojavov.

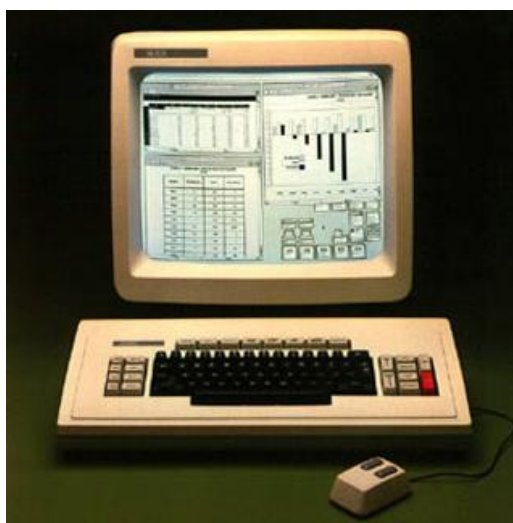
Po drugi svetovni vojni se je začela atomska era in v obdobju hladne vojne je bilo treba razviti sistem za hitro odkrivanje sovražnih letal ali raket, ki bi lahko začele napad z atomskim orožjem. To je sprožilo masovno proizvodnjo računalnikov in razvoj grafičnih sistemov za opazovanje zračnega prostora. Razvoj je potekal v okviru tajnih vojaških projektov, delno pa tudi na javnih univerzah in inštitutih. Tako je leta 1963 Ivan Sutherland [2.43] na MIT izdelal program Sketchpad [2.44] v okviru svojega doktorskega dela. Sketchpad je bil program za 2D-/3D-risanje na vektorskem zaslonu z uporabo svetlobnega peresa in namenskih gumbov. Sutherland je bil pozneje profesor na Univerzi Utah, kjer so postavili temelje sodobne računalniške grafike; ustanovil je podjetje Evans & Sutherland [2.45].

Douglas Engelbart [2.46] je bil zaposlen na Standfordskem raziskovalnem inštitutu, ko je pridobil financiranje projekta Vojaške razvojne agencije (DARPA – Defense Advanced Research Projects Agency). V okviru tega projekta je bila leta 1964 izdelana in leta 1967 patentirana računalniška miška. To je bilo veliko pred časom, ko je miška postala nepogrešljiv pripomoček za delo. Miška na sliki 2.11 je bila predstavljena v filmu znanstvene fantastike (The Mother of all Demos [2.47]), vendar so njeno vrednost spoznali šele čez približno 20 let na osebnih računalnikih.



Slika 2.11 – Prva miška

Predstavitev miške je spodbudila raziskovalce v vojaškem inštitutu PARC (Palo Alto Research Center) [2.48], da so v začetku sedemdesetih let 20. stoletja začeli razvijati grafični operacijski sistem z okni. Tako je leta 1981 nastal računalnik Xerox Star [2.49], prikazan na sliki 2.12. Xerox Star ni bil preveč uspešen, vendar so tako v podjetju Apple kot pri Microsoftu spoznali potencial tega operacijskega sistema. Obe podjetji sta zaposlili razvijalce računalnika Xerox Star in razvili svoja grafična operacijska sistema.



Slika 2.12 – Računalnik Xerox Star

### 2.3 Programska oprema za inženirske simulacije

Čeprav si danes ne moremo predstavljati simulacijskega programa brez računalniške grafike in grafičnega uporabniškega vmesnika, pa so prvi programi za simulacije bili popolnoma brez nje. Simulacije so se uporabljale za reševanje problemov v vojaški industriji, letalstvu, vesoljski tehniki ali pri gradnji in obratovanju jedrskih reaktorjev. Programi so bili običajno shranjeni na magnetnih trakovih, numerični modeli pa so se vnašali v obliki luknjanih kartic. Zato imajo še danes vsi programi za simulacijo osnovni vnos podatkov v obliki besedilne datoteke, v kateri so podatki numeričnega modela zapisani v vrsticah, podobno kot nekoč na karticah.

V agenciji NASA so leta 1964 ugotovili, da uporabljajo številne simulacijske programe. Veliko lažje bi bilo, če bi imeli na voljo celovit program, ki bi reševal številne probleme. Zato so se lotili razvoja programa NASTRAN (NAsa STRucture ANalysis) [2.50]. Leta 1968 je bilo ustanovljeno podjetje MSC (MacNeal-Schwendler Corporation), ki je razvijalo in tržilo program NASTRAN. Potem ko je podjetje MSC kupilo nekaj konkurenčnih podjetij, so sprožili protimonopolni postopek in zato so si leta 2002 izvorno kodo programa NASTRAN razdelila tri podjetja. Od takrat se razvijajo tri ločene verzije: MSC Nastran, Nei Nastran in NX Nastran.

Leta 1970 je John Swanson ustanovil podjetje, ki se je sprva imenovalo SASi (Swanson Analysis Systems Inc.) z namenom razvoja računalniškega programa za simulacije z imenom ANSYS [2.51]. Podjetje je bilo zelo uspešno z rastjo od 10 % do 20 % na leto. Leta 1994 je firmo SASi kupil investicijski sklad TA Associates in jo preimenoval v ANSYS Inc. ter preoblikoval v delniško družbo. ANSYS je danes eden od vodilnih programskih paketov za inženirske računalniške simulacije.

Leta 1976 je John O. Hallquist na Lawrence Livermore National Laboratory razvil program DYNA3D [2.52] za simulacijo dinamičnega trka atomske bombe FUFO (Full Fusing Option), ki je bila predvidena za spuščanje tako z višine 18 km pri nadzvočni hitrosti kot z višine 30 m pri podzvočni hitrosti ameriškega bombnika B-1A. Razvoj bombe so kasneje ustavili, ostal pa



je program za simulacijo, ki je bil uporaben za številne dinamične probleme. Program je bil sprva zelo poenostavljen zaradi nezadostnih računalniških kapacitet. Leta 1984 se je Hallquistu pridružil David Benson in program je postal precej bolj kompleksen ter sčasoma prirejen za različne računalniške platforme. Zaradi uspešnosti programa je bilo leta 1988 ustanovljeno podjetje LSTC (Livermore Software Technology Corporation). Program se je preimenoval v LS-Dyna in je postal veliko bolj univerzalen, namenjen reševanju številnih zahtevnih dinamičnih problemov.

Leta 1971 so raziskovalci Univerze Brown ustanovili prvo komercialno podjetje za analize po metodi končnih elementov, imenovano Marc (Marc Analysis Research Corporation). Program je danes v lasti podjetja MSC in se imenuje MSC Marc. Eden prvih zaposlenih v podjetju MARC je bil Dave Hibbitt, ki je skupaj z Bengom Karlssonom in Paulom Sorensenom leta 1978 ustanovil podjetje HKS (Hibbitt Karlsson Sorensen). To podjetje je izdelalo program za simulacije po metodi končnih elementov ABAQUS [2.53]. Podjetje se je potem preimenovalo v Abaqus Inc. in leta 2005 ga je kupilo podjetje Dassault Systemes ter ga preimenovalo v Simulia Corp.

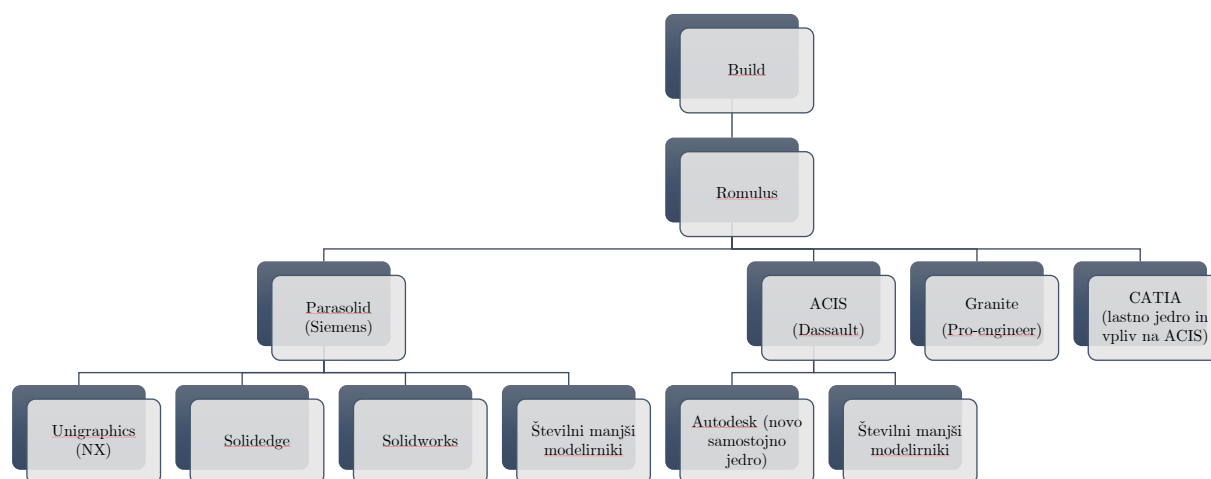
Na Fakulteti za strojništvo Univerze v Mariboru smo sodelovali pri razvoju programa za analize po metodi končnih elementov BERSAFE [2.54]. Program je razvil Trevor Hellen za potrebe britanskega elektrogospodarstva (Central Electricity Generating Board, Berkeley). Program se je uporabljal za analize različnih konstrukcijskih elementov v jedrskih elektrarnah, posebej za proučevanje širjenja razpok z metodo VCE (Virtual Crack Extension). Na Fakulteti za strojništvo v Mariboru je Janez Čep okoli leta 1990 program priredil za delo na mikroročunalnikih in je bil takrat eden prvih programov na osnovi metode končnih elementov na svetu, ki so delovali tudi na osebni računalnikih. Izdelana je bila tudi lastna grafična knjižnica in lastni pred- in poprocesorji za obdelavo podatkov ter rezultatov analiz po metodi končnih elementov.

V zadnjem desetletju 20. stoletja so računalniki končno dobili sliko. Do takrat je vse potekalo samo z izpisovanjem črk in števil. Za prikazovanje slik so bili na voljo grafični terminali, ki so se obnašali bolj kot tiskalniki, in ne toliko kot zaslone, ki jih poznamo danes. Delno je za spremembo poskrbel razvoj strojne opreme. Od prvih grafičnih programov je minilo skoraj 30 let, ko so se začeli vse bolj uveljavljati računalniki, ki so imeli integrirano grafiko – podobno kot računalnik, ki ga je leta 1962 uporabljal Ivan Sutherland. Računalnik ni več zavzel celotne sobe, ampak je bil nameščen kar na delovno mizo. Največji del delovnega prostora je navadno zasedal ekran s katodno cevjo, na katerem je bila prikazana slika. Bistveno pa je bilo ustvarjanje slike v računalniku na grafični kartici, kar je omogočalo hitre spremembe slike in s tem interaktivno delo z grafiko. To je bilo najprej mogoče na precej dragih delovnih postajah, pozneje pa na vedno bolj zmogljivih osebni računalnikih.

V letih med 1960 in 1980 so bili razviti vsi osnovni algoritmi računalniške grafike: prikazovanje grafike, matematični zapis krivulj in površin, odstranjevanje skritih robov, senčenje površin itd. Večina računalniških programov je do leta 1980 prikazovala zgolj 2D-računalniško grafiko. Na voljo pa so bili že številni paketi za računalniško podprto tehniško dokumentacijo.

Že leta 1965 so na Univerzi Cambridge začeli razvoj 3D-površinskega modelirnika, imenovanega BUILD [2.55]. Delo je potekalo v okviru računalniškega laboratorija CAD Group pod vodstvom

Charlesa Langa, ki sta se mu pridružila Ian Braid in kasneje še Alan Grayer. Delo so opravljali v okviru doktorskega študija in leta 1974 so ustanovili podjetje Shape Data, čeprav so vsi trije ostali zaposleni v CAD Group, kjer so razvili modelirnik BUILD 2. V podjetju Shape Data so na osnovi svojega dela leta 1978 razvili modelirnik Romulus, ki so ga začeli tržiti. Na osnovi modelirnika Romulus je nastalo veliko modelirnikov, med katerimi je bil najuspešnejši ME30 podjetja Hewlett Packard. Podjetje Shape Data Ltd. je kupilo podjetje Evans & Sutherland in modelirnik Romulus so preoblikovali v programsko knjižnico Parasolid [2.56]. Kasneje je podjetje Shape Data in z njo knjižnico Parasolid kupil Unigraphics, ki je danes v lasti Siemens. Leta 1985 so Ian Braid, Charles Lang in Alan Grayer zapustili podjetje Shape Data in se pridružili podjetju Spatial Technology, kjer so razvili podobno knjižnico imenovano ACIS (Alan Charles Ian System) [2.57]. Podjetje Spatial technology je potem kupilo podjetje Dassault Systemes, ki je danes lastnik knjižnice ACIS. Kot je razvidno s slike 2.13, skoraj vsi današnji modelirniki temeljijo na knjižnici Parasolid ali knjižnici ACIS, zato je med modelirniki toliko podobnosti.



Slika 2.13 – Razvoj 3D-modelirnikov

Leta 1971 je Patrick Hanratty [2.58], znan tudi kot oče CAD/CAM, ustanovil podjetje MCS in izdelal program ADAM (Automated Drafting and Machining). Različice izvorne koda tega programa je prodal različnim razvijalcem programske opreme, zato po današnji oceni vsebuje vsaj 70 % vseh programov CAD/CAM njegovo kodo. Sam je razvil modelirnik ANVIL-4000, ki je bil pred leti priljubljen tudi v Sloveniji. Njegovo kodo so kupili tudi v podjetju UNIAPT in na osnovi tega izdelali program Unigraphics [2.59]. UNIAPT je deloval pod okriljem McDonell-Douglassa, ki ga je nato prodal podjetju EDS takrat v lasti podjetja General Motors. S tem je Unigraphics postal uraden program takrat največjega proizvajalca avtomobilov. Po velikih uspehih na trgu so kupili še močnega konkurenta – podjetje SDRC, ki je takrat tržilo modelirnik I-DEAS, ki je bil hišni modelirnik proizvajalca avtomobilov Ford. Leta 2002 so oba modelirnika združili v nov program, imenovan NX. NX je od leta 2007 v lasti Siemens.

Ford, ki bi moral uporabljati enak modelirnik kot njegov konkurent GM, pa je raje začel uporabljati modelirnik CATIA [2.60]. Leta 1977 so v podjetju Avions Marcel Dassault začeli za svoje potrebe pri razvoju letala Mirage razvijati računalniški program CATI (Conception Assistée Tridimensionnelle Interactive). Leta 1981 so površinski modelirnik začeli tržiti pod imenom CATIA, in sicer skupaj s podjetjem IBM.

Že leta 1965 so v letalskem podjetju Lockheed začeli razvijati program, ki so ga pozneje imenovali CADAM [2.61]. Leta 1974 so prodali prve verzije programa zunanjim kupcem. Med temi je bil tudi Avions Marcel Dassault. Da bi kupci lažje prilagodili program svojim potrebam, so jim ponudili kar izvorno kodo programa. Ker je program običajno deloval na IBM-ovi strojni opremi, je program leta 1989 kupilo podjetje IBM, ki je tako tržilo dva programa: programa CADAM in CATIA. Leta 1992 je Dassault Systemes kupil CADAM Inc. in združil oba v naslednjih različicah programa CATIA. Kmalu je sledila tudi podpora za različne vrste strojne opreme in resnično sodobna verzija programa CATIA V5.

V zadnjem desetletju drugega tisočletja so se delovne postaje komaj uveljavile, ko so začeli prodirati osebni računalniki Windows. To je bilo obdobje, ko so propadla številna ugledna podjetja in so se pojavila nova podjetja, ki so hitro osvojila trg. Težko si je predstavljati, da sta programa CADAM in CATIA tekmovala za nekaj tisoč uporabnikov, ko gre pri osebnih računalnikih za milijone uporabnikov.

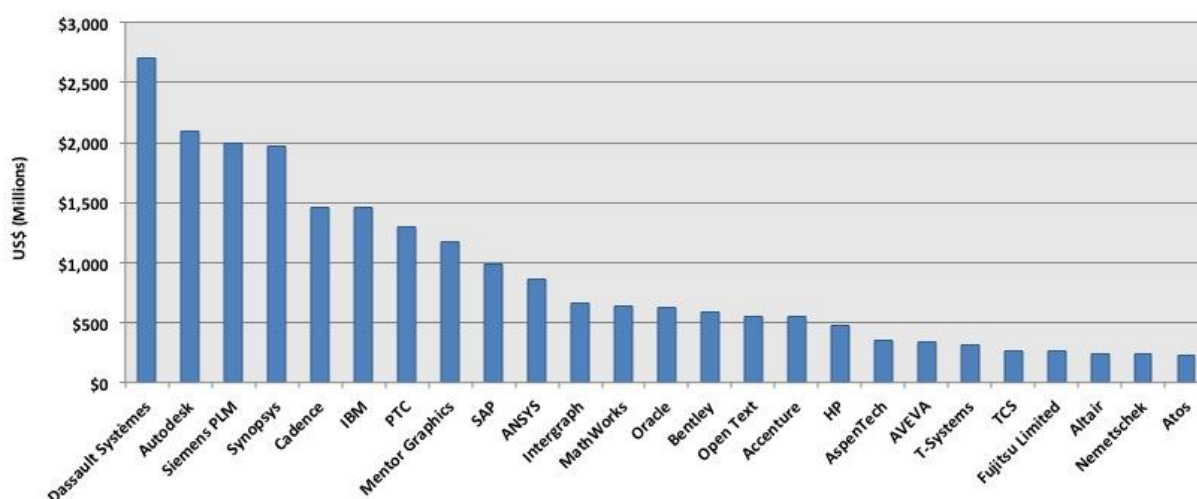
Leta 1974 je Samuel Geisberg, profesor matematike na Univerzi v Leningradu, emigriral v Združene države Amerike. Zaposlil se je v podjetjih Computervision in Applicon, kjer je delal na razvoju CAD-programov. Predlagal je razvoj radikalno drugačnega programa, ki bi temeljil na prostorski geometriji z modeliranjem z značilkami in kjer bi bile dimenzije podane parametrično. Ker nobeno od podjetij ni želelo podpreti njegovega razvoja, je leta 1985 ob pomoči vlagateljev ustanovil podjetje PTC (Parametric Technology Corporation) [2.62]. Leta 1987 je bil izdelan program Pro-Engineer in prvi kupec je bilo podjetje Deere & Company. Pro-Engineer je zabeležil izjemen uspeh predvsem zaradi koncepta, da mora program delovati na vseh platformah enako. Program je delal na različnih delovnih postajah in tudi na osebnih računalnikih, ko so le-ti postali dovolj zmogljivi. Zaradi tega je že v nekaj letih PTC zrasel v vodilnega proizvajalca programske opreme na področju CAD/CAM. Leta 1998 pa je sledil padec priljubljenosti, saj so v podjetju zanemarili nenehen razvoj programa. Zato so programi Unigraphics in CATIA vključili vse, kar je odlikovalo Pro-Engineer in še več. Po drugi strani pa so začeli prodirati novi programi iz spodnjega dela trga, kot so Solidworks in Autocad. Podjetje PTC se je v nekaj letih spet ujelo in danes velja za tretjega proizvajalca opreme CAD/CAM, ki se je preimenovala v serijo izdelkov Creo.

Leta 1993 je Jon Hirschtick zapustil ComputerVision in nekaj časa delal kar doma. Skupaj s prijatelji so izdelovali nov modelirnik. Leta 1994 se jim je priključil Michael Payne, ki je bil pred tem namestnik vodje razvoja v podjetju PTC. Ustanovili so podjetje z imenom Winchester Design Systems in v letu 1995 dokončali prvo različico programa Winchester Design, ki se je kasneje preimenoval v Solidworks [2.63].

Program Solidworks je sprva temeljil na knjižnici ACIS, potem pa so se zaradi boljšega delovanja in nižje cene odločili za knjižnico Parasolid. Solidworks je deloval zgolj v okolju Windows in je bil zelo dobro sprejet, saj je že prva verzija programa delovala brezhibno. Poleg

tega je cena programa znašala le nekoliko več kot cena zmogljivega računalnika, medtem ko so bili konkurenčni izdelki veliko dražji. Potem ko so prodali zgolj 6000 kopij Solidworksa, je podjetje leta 1997 kupilo podjetje Dassault Systemes. Kljub temu se Solidworks še danes samostojno razvija in zavzema segment tržišča manjših podjetij. Glavni produkt podjetja Dassault Systemes je CATIA, ki je namenjena za bolj zahtevno delo in jo uporabljajo večja podjetja.

Velike zasluge na področju tehnologij CAx [2.64] pa danes dosega tudi podjetje Autodesk [2.65], saj proda veliko programov poceni programske opreme. Že v začetku leta 1982 je 17 ljudi, ki so že prej sodelovali pri manjši projekti, ustanovilo podjetje Marin Software Partners. Vsak je vložil v podjetje nekaj denarja in vsak je razvijal svojo programsko opremo, ki so jo potem skupaj tržili na različnih sejmih in razstavah. Podjetje je vodil John Walker, Mike Riddle pa je izdelal program, ki se je najprej imenoval Interact, nato pa MicroCAD, zato je bilo dogovorjeno, da prejema 10 % zasluga tega programa. Po uspehu na trgu se je podjetje preimenovalo v Autodesk po programu za organizacijo namizja, ki ga je prodajalo. Ker se je medtem pojavil na trgu drug program z imenom MicroCAD, so svoj program preimenovali v AutoCAD. AutoCAD je beležil izjemne uspehe, saj so že leta 1983 prodali 1000 kopij po ceni 1000 USD in zaslužili prvi milijon dolarjev. Zato so vso energijo usmerili v razvoj tega programa. Zaradi velikega uspeha programa je bilo kmalu nevzdržno plačevati 10 % Riddlu za vsak izvod AutoCAD-a, zato je Riddle zapustil podjetje, leta 1992 pa so se pogodili za 11 milijonov dolarjev namesto 10 % od vsakega izvoda. Autodesk je kmalu začel razvijati sorodne proizvode. Leta 1999 je izdelal prostorski modelirnik, imenovan Inventor, ki je temeljil na knjižnici ACIS. Potem ko je Dassault Systemes postal lastnik knjižnice ACIS, je Autodesk na osnovi izvorne kode ACIS razvil tudi svojo različico knjižnice. Zaradi velikega tržnega uspeha je Autodesk kupil različna podjetja na področju CAD/CAM in računalniških simulacij. Tako so leta 2008 kupili podjetje Moldflow [2.66] in s tem zelo pomembno programsko opremo za simulacijo brizganja plastičnih mas.



Slika 2.14 – Zasluge CAx podjetij v letu 2014 [2.67]

Čeprav velja na trgu mnenje, da sta dva največja oz. najboljša sistema za CAD/CAM NX podjetja Siemens in CATIA podjetja Dassault Systemes (PTC pa se običajno navaja kot tretje podjetje), pa je dejstvo, da Autodesk še vedno prodaja veliko kosov programske opreme, kar ga uvršča na drugo mesto po zaslužku [2.67]. Podjetja Synopsys, Cadence in Mentor Graphics izdelujejo programsko opremo za načrtovanje in simulacijo elektronskih vezij in zato zgoraj niso bila omenjena, čeprav so po zaslužku zelo visoko.

## 2.4 Literatura

- [2.1] Neocortex [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Neocortex>. [Datum dostopa 5. 12. 2017].
- [2.2] Sexagesimal [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Sexagesimal>. [Datum dostopa 5. 12. 2017].
- [2.3] 10 Facts About Abacus [splet], Dosegljivo: <http://factfile.org/10-facts-about-abacus>. [Datum dostopa 5. 12. 2017].
- [2.4] Egyptian numerals [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Egyptian\\_numerals](https://en.wikipedia.org/wiki/Egyptian_numerals). [Datum dostopa 5. 12. 2017].
- [2.5] Pre computer calculations of  $\pi$  [splet], Dosegljivo: [http://www-history.mcs.st-andrews.ac.uk/history/HistTopics/Pi\\_chronology.html](http://www-history.mcs.st-andrews.ac.uk/history/HistTopics/Pi_chronology.html). [Datum dostopa 5. 12. 2017].
- [2.6] Astronomical unit [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Astronomical\\_unit](https://en.wikipedia.org/wiki/Astronomical_unit). [Datum dostopa 5. 12. 2017].
- [2.7] The Space Shuttle and the Horse's Rear End [splet], Dosegljivo: <http://www.astrodigital.org/space/stshorse.html>. [Datum dostopa 5. 12. 2017].
- [2.8] Leonardo Fibonacci [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Leonardo\\_Fibonacci](https://sl.wikipedia.org/wiki/Leonardo_Fibonacci). [Datum dostopa 5. 12. 2017].
- [2.9] Period or Comma? Decimal Styles over Time and Place [splet], Dosegljivo: <http://www.councilscienceeditors.org/wp-content/uploads/v31n2p042-043.pdf>. [Datum dostopa 5. 12. 2017].
- [2.10] John Napier [splet], Dosegljivo: [https://en.wikipedia.org/wiki/John\\_Napier](https://en.wikipedia.org/wiki/John_Napier). [Datum dostopa 5. 12. 2017].
- [2.11] Napier's bones [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Napier%27s\\_bones](https://en.wikipedia.org/wiki/Napier%27s_bones). [Datum dostopa 5. 12. 2017].
- [2.12] Edmund Gunter [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Edmund\\_Gunter](https://en.wikipedia.org/wiki/Edmund_Gunter). [Datum dostopa 5. 12. 2017].
- [2.13] Slide rule [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Slide\\_rule](https://en.wikipedia.org/wiki/Slide_rule). [Datum dostopa 5. 12. 2017].

- [2.14] Simulated Pickett N4-ES Slide Rule [splet], Dosegljivo: <http://www.antiquark.com/sliderule/sim/n4es/virtual-n4es.html>. [Datum dostopa 5. 12. 2017].
- [2.15] Wilhelm Schickard [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Wilhelm\\_Schickard](https://en.wikipedia.org/wiki/Wilhelm_Schickard). [Datum dostopa 5. 12. 2017].
- [2.16] Pascal's calculator [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Pascal%27s\\_calculator](https://en.wikipedia.org/wiki/Pascal%27s_calculator). [Datum dostopa 5. 12. 2017].
- [2.17] The Stepped Reckoner of Gottfried Leibniz [splet], Dosegljivo: <http://history-computer.com/MechanicalCalculators/Pioneers/Lebniz.html>. [Datum dostopa 5. 12. 2017].
- [2.18] Leonardo da Vinci [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Leonardo\\_da\\_Vinci](https://sl.wikipedia.org/wiki/Leonardo_da_Vinci). [Datum dostopa 5. 12. 2017].
- [2.19] Galileo Galilei [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Galileo\\_Galilei](https://sl.wikipedia.org/wiki/Galileo_Galilei). [Datum dostopa 5. 12. 2017].
- [2.20] Robert Hooke [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Robert\\_Hooke](https://en.wikipedia.org/wiki/Robert_Hooke). [Datum dostopa 5. 12. 2017].
- [2.21] Isaac Newton [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Isaac\\_Newton](https://sl.wikipedia.org/wiki/Isaac_Newton). [Datum dostopa 5. 12. 2017].
- [2.22] Leonhard Euler [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Leonhard\\_Euler](https://sl.wikipedia.org/wiki/Leonhard_Euler). [Datum dostopa 5. 12. 2017].
- [2.23] Daniel Bernoulli [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Daniel\\_Bernoulli](https://en.wikipedia.org/wiki/Daniel_Bernoulli). [Datum dostopa 5. 12. 2017].
- [2.24] Thomas Young [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Thomas\\_Young\\_\(scientist\)](https://en.wikipedia.org/wiki/Thomas_Young_(scientist)). [Datum dostopa 5. 12. 2017].
- [2.25] Carl Friedrich Gauss [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Carl\\_Friedrich\\_Gauss](https://en.wikipedia.org/wiki/Carl_Friedrich_Gauss). [Datum dostopa 5. 12. 2017].
- [2.26] Siméon Denis Poisson [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Sim%C3%A9on\\_Denis\\_Poisson](https://en.wikipedia.org/wiki/Sim%C3%A9on_Denis_Poisson). [Datum dostopa 5. 12. 2017].
- [2.27] Poisson's ratio and modern materials [splet], Dosegljivo: <https://www.nature.com/articles/nmat3134>. [Datum dostopa 5. 12. 2017].
- [2.28] Charles Babbage [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Charles\\_Babbage](https://en.wikipedia.org/wiki/Charles_Babbage). [Datum dostopa 5. 12. 2017].
- [2.29] Ada Lovelace [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Ada\\_Lovelace](https://en.wikipedia.org/wiki/Ada_Lovelace). [Datum dostopa 5. 12. 2017].
- [2.30] Herman Hollerith [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Herman\\_Hollerith](https://en.wikipedia.org/wiki/Herman_Hollerith). [Datum dostopa 5. 12. 2017].

- [2.31] Alan Turing [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Alan\\_Turing](https://en.wikipedia.org/wiki/Alan_Turing). [Datum dostopa 5. 12. 2017].
- [2.32] Turing machine [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Turing\\_machine](https://en.wikipedia.org/wiki/Turing_machine). [Datum dostopa 5. 12. 2017].
- [2.33] Konrad Zuse [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Konrad\\_Zuse](https://en.wikipedia.org/wiki/Konrad_Zuse). [Datum dostopa 5. 12. 2017].
- [2.34] John von Neumann [splet], Dosegljivo: [https://en.wikipedia.org/wiki/John\\_von\\_Neumann](https://en.wikipedia.org/wiki/John_von_Neumann). [Datum dostopa 5. 12. 2017].
- [2.35] Moore's law [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Moore%27s\\_law](https://en.wikipedia.org/wiki/Moore%27s_law). [Datum dostopa 5. 12. 2017].
- [2.36] History of programming languages [splet], Dosegljivo: [https://en.wikipedia.org/wiki/History\\_of\\_programming\\_languages](https://en.wikipedia.org/wiki/History_of_programming_languages). [Datum dostopa 5. 12. 2017].
- [2.37] The Origins of the Finite Element Method [splet], Dosegljivo: [http://home.iitk.ac.in/~mohite/History\\_of\\_FEM.pdf](http://home.iitk.ac.in/~mohite/History_of_FEM.pdf). [Datum dostopa 5. 12. 2017].
- [2.38] M. J. Turner, R. W. Clough, H. C. Martin, L. J. Topp, "Stiffness and Deflection Analysis of Complex Structures," *Journal of the Aeronautical Sciences*, let. 23, št. 9, str. 805–823, 1956.
- [2.39] Finite element method [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Finite\\_element\\_method](https://en.wikipedia.org/wiki/Finite_element_method). [Datum dostopa 5. 12. 2017].
- [2.40] Boundary element method [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Boundary\\_element\\_method](https://en.wikipedia.org/wiki/Boundary_element_method). [Datum dostopa 5. 12. 2017].
- [2.41] Finite volume method [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Finite\\_volume\\_method](https://en.wikipedia.org/wiki/Finite_volume_method). [Datum dostopa 5. 12. 2017].
- [2.42] Meshfree methods [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Meshfree\\_methods](https://en.wikipedia.org/wiki/Meshfree_methods). [Datum dostopa 5. 12. 2017].
- [2.43] Ivan Sutherland [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Ivan\\_Sutherland](https://en.wikipedia.org/wiki/Ivan_Sutherland). [Datum dostopa 5. 12. 2017].
- [2.44] Sketchpad [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Sketchpad>. [Datum dostopa 5. 12. 2017].
- [2.45] Evans & Sutherland [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Evans\\_%26\\_Sutherland](https://en.wikipedia.org/wiki/Evans_%26_Sutherland). [Datum dostopa 5. 12. 2017].
- [2.46] Douglas Engelbart [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Douglas\\_Engelbart](https://en.wikipedia.org/wiki/Douglas_Engelbart). [Datum dostopa 5. 12. 2017].
- [2.47] Douglas Engelbart : The Mother of All Demos [splet], Dosegljivo: <https://www.youtube.com/watch?v=JfIgzSoTMOs>. [Datum dostopa 5. 12. 2017].

- [2.48] PARC [splet], Dosegljivo: [https://en.wikipedia.org/wiki/PARC\\_\(company\)](https://en.wikipedia.org/wiki/PARC_(company)) . [Datum dostopa 5. 12. 2017].
- [2.49] Xerox Star [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Xerox\\_Star](https://en.wikipedia.org/wiki/Xerox_Star). [Datum dostopa 5. 12. 2017].
- [2.50] Nastran [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Nastran>. [Datum dostopa 5. 12. 2017].
- [2.51] Ansys [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Ansys>. [Datum dostopa 5. 12. 2017].
- [2.52] LS-DYNA [splet], Dosegljivo: <https://en.wikipedia.org/wiki/LS-DYNA>. [Datum dostopa 5. 12. 2017].
- [2.53] Abaqus [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Abaqus>. [Datum dostopa 5. 12. 2017].
- [2.54] T. K. Hellen, S. J. Protheroe, "The BERSAFE finite element system," *Computer-Aided Design*, let. 6, št. 1, str. 15–24, 1974.
- [2.55] Solid Modeling [splet], Dosegljivo: <http://solidmodeling.org/bezier-award/i-braid-a-grayer-and-c-lang/>. [Datum dostopa 5. 12. 2017].
- [2.56] Parasolid [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Parasolid>. [Datum dostopa 5. 12. 2017].
- [2.57] ACIS [splet], Dosegljivo: <https://en.wikipedia.org/wiki/ACIS>. [Datum dostopa 5. 12. 2017].
- [2.58] Patrick J. Hanratty [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Patrick\\_J.\\_Hanratty](https://en.wikipedia.org/wiki/Patrick_J._Hanratty). [Datum dostopa 5. 12. 2017].
- [2.59] Siemens NX [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Siemens\\_NX](https://en.wikipedia.org/wiki/Siemens_NX). [Datum dostopa 5. 12. 2017].
- [2.60] CATIA [splet], Dosegljivo: <https://en.wikipedia.org/wiki/CATIA>. [Datum dostopa 5. 12. 2017].
- [2.61] IBM, Lockheed and Dassault Systèmes [splet], Dosegljivo: <http://www.cadhistory.net/13%20IBM,%20Lockheed%20and%20Dassault.pdf>. [Datum dostopa 5. 12. 2017].
- [2.62] Parametric Technology [splet], Dosegljivo: <http://www.cadhistory.net/16%20Parametric%20Technology.pdf>. [Datum dostopa 5. 12. 2017].
- [2.63] SolidWorks [splet], Dosegljivo: <http://www.cadhistory.net/18%20SolidWorks.pdf>. [Datum dostopa 5. 12. 2017].
- [2.64] List of CAx companies [splet], Dosegljivo: [https://en.wikipedia.org/wiki/List\\_of\\_CAx\\_companies](https://en.wikipedia.org/wiki/List_of_CAx_companies). [Datum dostopa 5. 12. 2017].
- [2.65] Autodesk and AutoCAD [splet], Dosegljivo: <http://www.cadhistory.net/08%20Autodesk%20and%20AutoCAD.pdf>. [Datum dostopa 5. 12. 2017].



- [2.66] Moldflow [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Moldflow>. [Datum dostopa 5. 12. 2017].
- [2.67] CIMdata Publishes PLM Market and Solution Provider Report [splet], Dosegljivo: <http://www.cimdata.com/en/news/item/1878-cimdata-publishes-plm-market-and-solution-provider-report>. [Datum dostopa 5. 12. 2017].

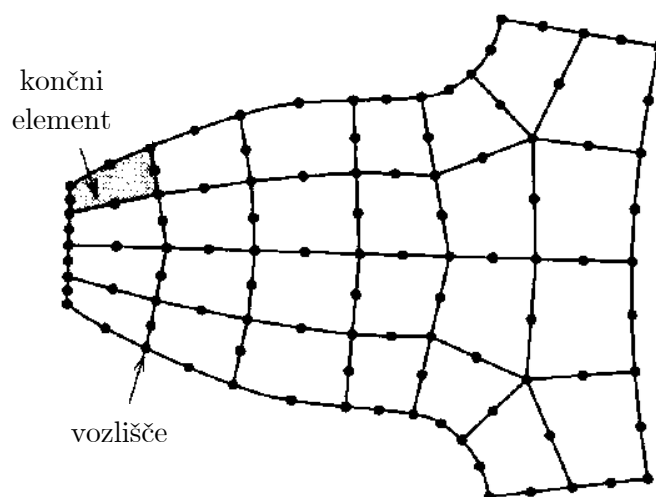


### 3. Osnove metode končnih elementov

Metoda končnih elementov ali MKE (angl. *FEM – Finite Element Method*) omogoča iskanje približne rešitve parcialne diferencialne enačbe za problem robnih vrednosti [3.1]. Približno rešitev poiščemo tako, da območje problema razdelimo na manjše dele enostavnih geometrijskih oblik, ki jih imenujemo končni elementi (slika 3.1).

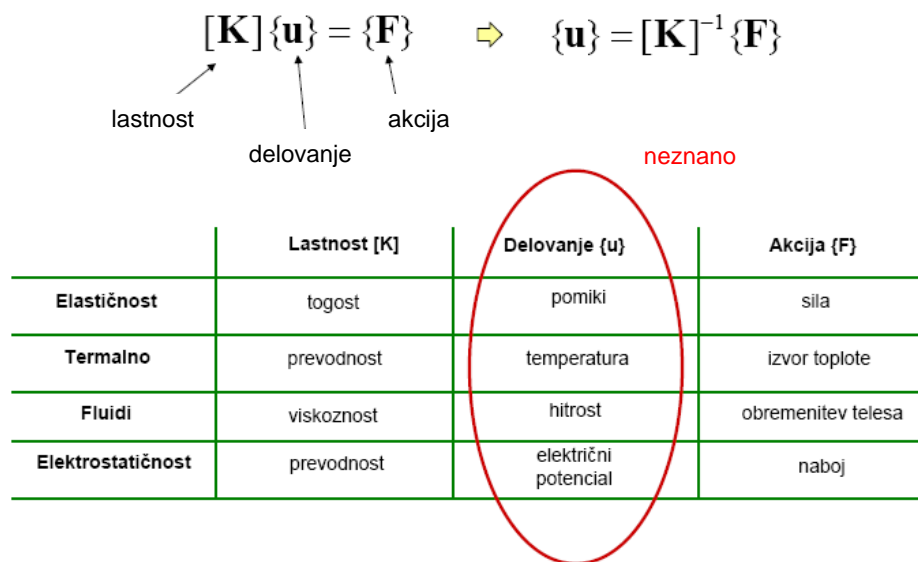
Končni elementi so določeni z oglišči. V oglišča namestimo vozlišča, to so točke v katerih dogovorno iščemo diskretne rešitve osnovnih spremenljivk problema v izbranih prostostnih stopnjah (*DOF – angl. Degrees Of Freedom*). Prostostna stopnja vozlišča določa možno smer pomika oz. zasuka v posameznem vozlišču. Z aproksimacijskim interpolacijskim nastavkom pa predpostavimo potek spremembe osnovnih spremenljivk po volumnu končnega elementa med vrednostmi v njegovih vozliščih. Ob uporabi višjerednih interpolacijskih funkcij imajo končni elementi vozlišča razporejena tudi vzdolž svojih stranic. V ravnini so povezana vozlišča podobna ribiški mreži, zato uporabljamo termin mreža končnih elementov (slika 3.1). Termin mreža končnih elementov uporabljamo tako za končne elemente v ravnini kot v prostoru in tudi za linijske končne elemente. Vozlišča so določena s številko vozlišča in s koordinatami v prostoru. Vsak element je določen s številko elementa, vrsto elementa in seznamom številk vozlišč, ki ga določajo. Končna elementa sta soseda, če sta določena z istim vozliščem. Tip končnega elementa označuje osnovno teorijo (1D, 2D ali 3D), ki je podlaga za popis končnega elementa. Tip določa tudi vrsto vozlišč in način računanja rezultatov. Zato je število prostostnih stopenj vozlišča vedno določeno s tipom končnega elementa.

Ker je natančnost rešitve problema odvisna od števila končnih elementov ter njihovih izbranih interpolacijskih funkcij, pri metodi končnih elementov nikoli ne dobimo povsem natančnega rezultata, ampak zgolj približek. Več elementov in višjeredne interpolacijske funkcije uporabimo, natančnejša je rešitev.



Slika 3.1 – Mreža končnih elementov

Ko obravnavamo trdnine, nas običajno zanima, kakšni so njihovi pomiki, specifične deformacije in napetosti, ko nanje deluje zunanja obremenitev. Poenostavljeno gledano, opazujemo razmerja med pomiki, specifičnimi deformacijami in napetostmi v odvisnosti od togosti izdelka in velikosti zunanje obremenitve. Podobno velja tudi za druge fizikalne pojave, ki jih rešujemo na enak način. Zato lahko splošno rečemo, da nas zanima odziv (delovanje) obravnavanega problema v odvisnosti od njegovih snovskih lastnosti in akcije, kot je to prikazano na sliki 3.2.



Slika 3.2 – Osnovna razmerja za reševanje problemov z MKE

V enačbi na sliki 3.2 je  $K$  matrika, v kateri vsaka vrstica oz. stolpec predstavlja lastnost ene prostostne stopnje numeričnega modela. Vektor  $F$  vsebuje akcije, ki so porazdeljene po nekaterih prostostnih stopnjah, medtem ko iščemo neznan odziv (delovanje) vektorja  $u$  za vsako prostostno stopnjo. Sistem razpoložljivih enačb je tako določen s številom prostostnih stopenj problema.

Ker so prostostne stopnje opredeljene v vozliščih, vendar definirane v končnih elementih, je najprej treba vse snovne lastnosti problema določiti za vsako vozlišče oz. prostostno stopnjo končnega elementa posebej. Nato pa se prispevki vseh končnih elementov po prostostnih stopnjah seštevajo v globalni matriki snovnih lastnosti.

Kratka zgodovina MKE:

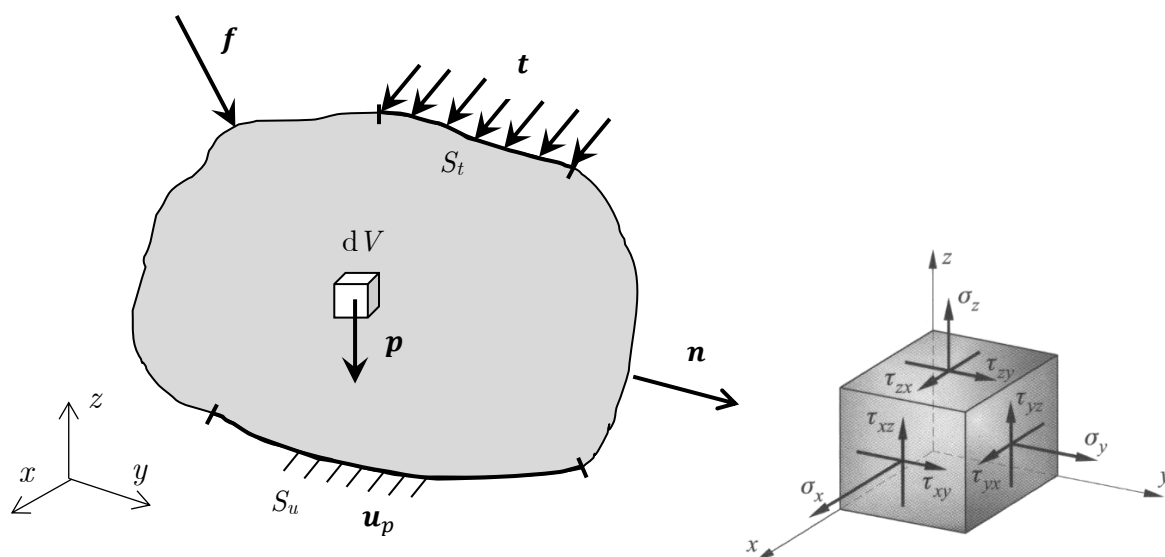
- 1956 – prvič predstavljena tehnika reševanja inženirskih problemov z uporabo končnih elementov [3.2],
- 1960 – prvič uporabljen termin metoda končnih elementov (angl. *Finite Element Method*) [3.3],
- 1965 – prikaz uporabe MKE za reševanje drugih območnih problemov, kot so prenos toplote, tok tekočin ipd. z objavo članka [3.4],
- 1968 – vpeljava koncepta izoparametričnih končnih elementov [3.5],
- do danes – izjemen razvoj MKE v smislu tehnične sofisticiranosti metode in njene uporabnosti za reševanje širokega spektra inženirskih problemov [3.6].

V začetku uporabe metode končnih elementov praktično ni bilo razlike med razvijalcem in uporabnikom metode. Danes pa uporabnik le redko razvija metodo reševanja problema in večinoma pripravlja podatke za numerično analizo (predprocesiranje), zaganja preračune po metodi končnih elementov (procesiranje) in analizira rezultate preračunov (procesiranje).

### 3.1 Osnove mehanike elastičnih trdnih teles v MKE

Slika 3.3 prikazuje ravnotežno stanje trdnega telesa v trirazsežnem prostoru s koordinatnimi smermi  $x$ ,  $y$  in  $z$ , ki je obremenjeno s telesnimi silami  $\mathbf{p}$  (gravitacija, centrifugalne sile, elektromagnetne sile), specifičnimi obremenitvami na enoto površine  $\mathbf{t}$  (tlak, kontakt, trenje ipd.) in točkovnimi silami  $\mathbf{f}$  ter podprto na delu površine  $S_u$ , na katerem poznamo pomike  $\mathbf{u} = \mathbf{u}_p$ , pri čemer so vse navedene veličine izražene v vektorski obliki kot:

$$\mathbf{p} = \begin{Bmatrix} p_x \\ p_y \\ p_z \end{Bmatrix}, \quad \mathbf{t} = \begin{Bmatrix} t_x \\ t_y \\ t_z \end{Bmatrix}, \quad \mathbf{f} = \begin{Bmatrix} f_x \\ f_y \\ f_z \end{Bmatrix} \quad \text{in} \quad \mathbf{u} = \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix} \quad \text{oziroma} \quad \mathbf{u}_p = \begin{Bmatrix} u_{px} \\ u_{py} \\ u_{pz} \end{Bmatrix}.$$



Slika 3.3 – Ravnotežje trdnega telesa pod obremenitvijo

Ravnotežna enačba diferencialno majhnega dela trdnega telesa  $dV$  (slika 3.3) je zapisana kot:

$$\begin{aligned} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + p_x &= 0 \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + p_y &= 0 \\ \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + p_z &= 0 \end{aligned} \Rightarrow (\text{div} \underline{\boldsymbol{\sigma}})^T + \mathbf{p} = 0 \quad (3.1)$$

kjer je Cauchyev napetostni tenzor  $\underline{\boldsymbol{\sigma}}$  definiran kot:

$$\underline{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix} \quad \tau_{ij} = \tau_{ji}$$

Če zapišemo zgolj 6 neodvisnih členov napetostnega tenzorja kot napetostni vektor:

$$\boldsymbol{\sigma}^T = \{\sigma_{xx} \ \sigma_{yy} \ \sigma_{zz} \ \tau_{xy} \ \tau_{yz} \ \tau_{zx}\} \quad (3.2)$$

lahko ravnotežno enačbo (3.1) zapišemo poenostavljeno kot:

$$(\text{div} \underline{\boldsymbol{\sigma}})^T + \mathbf{p} = 0 \quad \Rightarrow \quad \mathbf{L}^T \cdot \boldsymbol{\sigma} + \mathbf{p} = 0 \quad (3.3)$$

kjer je diferencialni operator  $\mathbf{L}$  določen kot:

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \quad (3.4)$$

Rešitev ravnotežne enačbe je mogoče najti le ob znanih robnih pogojih:

na delu površine  $S_u$ , kjer so poznani pomiki  
(Neumannov robni pogoj):

$$\mathbf{u}_{Su} = \mathbf{u}_p$$

na delu površine  $S_t$ , kjer delujejo zunanje  
obremenitve (Dirichletov robni pogoj):

$$\underline{\boldsymbol{\sigma}} \cdot \mathbf{n} = \mathbf{t}$$

(3.5)

kjer je  $\mathbf{n}^T = \{n_x, n_y, n_z\}$  normalni vektor površine.

Ker 3 ravnotežne enačbe (3.1) oziroma (3.3) (v vsaki koordinatni smeri:  $x$ ,  $y$  in  $z$ ) vsebujejo 6 neznanih neodvisnih napetostnih komponent  $\boldsymbol{\sigma}$ , potrebujemo za rešitev problema še dodatne enačbe. Tako najprej uvedemo *konstitutivne (materialne) enačbe*, ki povezujejo napetosti  $\boldsymbol{\sigma}$  in specifične deformacije  $\boldsymbol{\varepsilon}$ . Ob upoštevanju linearno elastične teorije majhnih deformacij lahko konstitutivno zvezo zapišemo v obliki razširjenega Hookovega zakona ( $\boldsymbol{\sigma} = \mathbf{E} \cdot \boldsymbol{\varepsilon}$ ) kot:

$$\boldsymbol{\sigma} = \mathbf{D} \cdot \boldsymbol{\varepsilon} \quad (3.6)$$

kjer je  $\mathbf{D}$  konstitutivna matrika elastičnosti:

$$\mathbf{D} = \begin{bmatrix} L + 2G & L & L & 0 & 0 & 0 \\ L & L + 2G & L & 0 & 0 & 0 \\ L & L & L + 2G & 0 & 0 & 0 \\ 0 & 0 & 0 & G & 0 & 0 \\ 0 & 0 & 0 & 0 & G & 0 \\ 0 & 0 & 0 & 0 & 0 & G \end{bmatrix} \quad (3.7)$$

zapisana z Lamejevima konstantama  $L = \frac{\nu E}{(1+\nu)(1-2\nu)}$  in  $G = \frac{E}{2(1+\nu)}$ , kjer sta  $E$  modul elastičnosti in  $\nu$  Poissonovo razmerje ter  $\boldsymbol{\varepsilon}$  vektor specifičnih deformacij:

$$\boldsymbol{\varepsilon}^T = \{\varepsilon_{xx} \ \varepsilon_{yy} \ \varepsilon_{zz} \ \gamma_{xy} \ \gamma_{yz} \ \gamma_{zx}\} \quad (3.8)$$

ki vsebuje zgolj neodvisne člene tenzorja specifičnih deformacij, določenega kot:

$$\underline{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} & \gamma_{xz} \\ \gamma_{yx} & \varepsilon_{yy} & \gamma_{yz} \\ \gamma_{zx} & \gamma_{zy} & \varepsilon_{zz} \end{bmatrix} \quad \gamma_{ij} = \gamma_{ji} , \quad (3.9)$$

Konstitutivne enačbe (3.6) uvajajo v opis problema 6 novih enačb (za vsako komponento napetosti), vendar uvedejo tudi novih 6 neznanih komponent vektorja specifičnih deformacij  $\boldsymbol{\varepsilon}$ . Zato uvedemo še dodatne *kinematične (kompatibilitetne) enačbe*, ki povezujejo specifične deformacije in pomike:

$$\boldsymbol{\varepsilon} = \mathbf{L} \cdot \mathbf{u} \quad (3.10)$$

Ob upoštevanju teorije majhnih deformacij so komponente vektorja oziroma tenzorja specifičnih deformacij določene kot:

$$\begin{aligned} \text{dolžinske (normalne) spec. deformacije: } \quad \varepsilon_{ii} &= \frac{\partial u_i}{\partial x_i} \quad i = \langle x|y|z \rangle \\ \text{kotne (strižne) spec. deformacije: } \quad \gamma_{ij} &= \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \quad i, j = \langle x|y|z \rangle \end{aligned} \quad (3.11)$$

pri čemer so v enačbi (3.10) posamezna diferencialna razmerja priročno zapisana z diferencialnim operatorjem  $\mathbf{L}$ , določenim v enačbi (3.4).

Enačba (3.10) nam tako uvede 6 novih enačb (za vsako komponento specifične deformacije) ter zgolj 3 nove neznane komponente vektorja pomikov  $\mathbf{u}$ . Tako imamo zdaj na voljo skupno 15 enačb (3 ravnotežne (3.1) oziroma (3.3), 6 konstitutivnih (3.6) in 6 kinematičnih (3.10)) za določitev 15 neznank (3 komponente pomikov  $\mathbf{u}$ , 6 komponent specifičnih deformacij  $\boldsymbol{\varepsilon}$  in 6 komponent napetosti  $\boldsymbol{\sigma}$ ), s čimer je ob upoštevanju robnih pogojev

$$\begin{aligned} &\text{na delu površine } S_u, \text{ kjer so poznani pomiki} \\ &\text{(Neumannov robni pogoj):} \quad \mathbf{u}_{Su} = \mathbf{u}_p \\ &\text{na delu površine } S_t, \text{ kjer delujejo zunanje} \\ &\text{obremenitve (Dirichletov robni pogoj):} \quad \underline{\boldsymbol{\sigma}} \cdot \mathbf{n} = \mathbf{t} \end{aligned} \quad (3.5)$$

problem trdnega telesa enolično določljiv.

Vendar pa morajo biti navedene vodilne enačbe natančno izpolnjene v prav vsaki točki obravnavanega problema (stroga formulacija problema), kar je mogoče zagotoviti zgolj z analitično rešitvijo. Ustrezne analitične rešitve je mogoče najti le v primeru enostavnih geometrij in robnih pogojev obravnavanega problema.

### 3.2 Princip virtualnega dela

Za splošno reševanje problema trdnega telesa zato poiščemo približno rešitev z uporabo t. i. šibke formulacije problema. Eden od načinov je z uporabo principa virtualnega dela, ki pravi, da je elastično telo v ravnotežju z zunanji delujočimi obremenitvami, če je v ravnotežju opravljeno delo zunanjih in notranjih sil zaradi virtualnih pomikov:

$$\delta W_n = \delta W_z \quad (3.12)$$

Virtualni pomiki  $\delta \mathbf{u}$  so diferencialno majhne spremembe koordinat telesa ob upoštevanju robnih pogojev.

Virtualno delo notranjih sil je določeno z razmerjem notranjih napetosti in specifičnih deformacij, integriranih po volumnu telesa, ki so posledica virtualnih pomikov  $\delta \mathbf{u}$ . Ker specifične deformacije izhajajo iz spremembe pomikov, so pri virtualnih pomikih  $\delta \mathbf{u}$  tudi specifične deformacije virtualne:

$$\delta \boldsymbol{\varepsilon} = \mathbf{L} \cdot \delta \mathbf{u} \quad (3.13)$$

in lahko tako virtualno delo notranjih sil zapišemo kot:

$$\delta W_n = \int_V \delta \boldsymbol{\varepsilon}^T \cdot \boldsymbol{\sigma} \cdot dV = \int_V \delta \mathbf{u}^T \cdot \mathbf{L}^T \cdot \boldsymbol{\sigma} \cdot dV \quad (3.14)$$

Virtualno delo zunanjih sil je določeno z majhnimi virtualnimi pomiki zaradi zunanjih obremenitev ob upoštevanju robnih pogojev. Zunanje mehanske obremenitve so lahko točkovna sila  $\mathbf{f}$ , porazdeljena sila  $\mathbf{t}$  po površini telesa  $S_t$  (tlak, kontakt ipd.) ali telesna sila  $\mathbf{p}$ , porazdeljena po volumnu telesa  $V$  (gravitacija, centrifugalna sila, elektromagnetna sila ipd.). Virtualno delo zunanjih sil lahko tako zapišemo kot:

$$\delta W_z = \delta \mathbf{u}^T \cdot \mathbf{f} + \int_{S_t} \delta \mathbf{u}^T \cdot \mathbf{t} \cdot dS_t + \int_V \delta \mathbf{u}^T \cdot \mathbf{p} \cdot dV \quad (3.15)$$

Ob upoštevanju enačbe 3.12 lahko zdaj zapišemo ravnotežno enačbo virtualnega dela v obliki:

$$\int_V \delta \mathbf{u}^T \cdot \mathbf{L}^T \cdot \boldsymbol{\sigma} \cdot dV = \delta \mathbf{u}^T \cdot \mathbf{f} + \int_{S_t} \delta \mathbf{u}^T \cdot \mathbf{t} \cdot dS_t + \int_V \delta \mathbf{u}^T \cdot \mathbf{p} \cdot dV \quad (3.16)$$

To je integralna enačba ravnotežja v t. i. šibki obliki. Bistvena razlika med šibko (enačba (3.16)) in strogo formulacijo problema (enačbe (3.3), (3.6) in (3.10)) je v tem, da v primeru prve ne zahtevamo izpolnitev vodilnih enačb v prav vsaki točki problema, ampak zgolj na obsegu celotnega obravnavanega problema. Poleg tega obstajajo načini za numerično (približno) reševanje kakršnekoli integralne enačbe, medtem ko so načini reševanja diferencialnih enačb omejeni zgolj na probleme z enostavnimi geometrijami in robnimi pogoji.

Ker tudi te integralne enačbe v splošnem ni mogoče rešiti analitično, nadomestimo osnovno spremenljivko (pomike) z njeno aproksimativno rešitvijo po metodi končnih elementov, kot je prikazano v nadaljevanju.



### 3.3 Osnovni aproksimacijski nastavek MKE

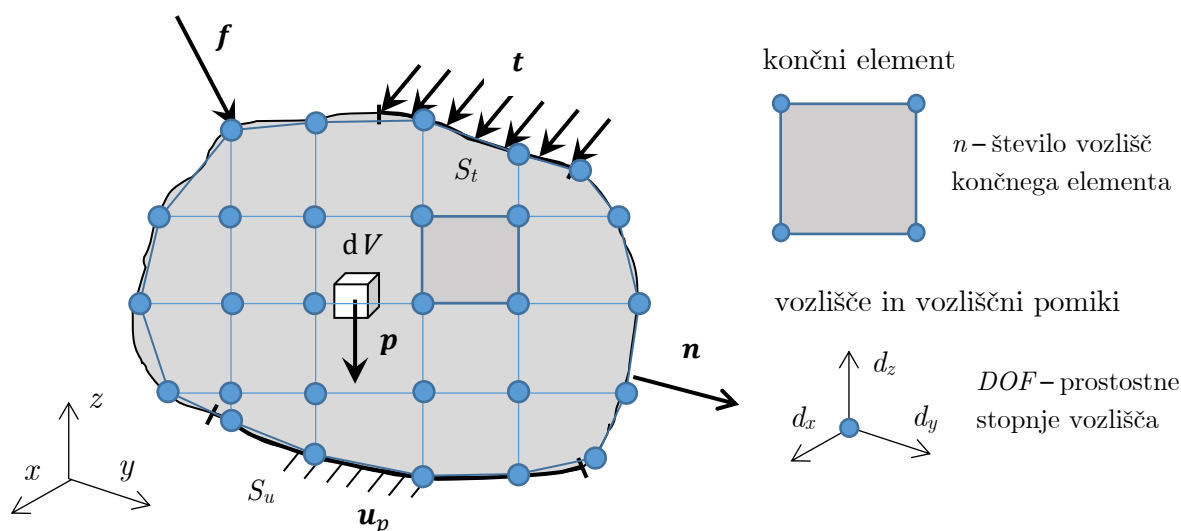
Kot že omenjeno v uvodu, obravnavano trdno telo razdelimo na manjše, geometrijsko enostavne dele, ki jih imenujemo končni elementi (slika 3.4). Postopku razdelitve problema na končne elemente pravimo tudi prostorska diskretizacija. Tipičen končni element je določen z vozlišči, pri čemer predpostavimo, da nas zgoj v vozliščih zanima natančna rešitev pomikov. Predpostavimo, da lahko pomike kjerkoli znotraj končnega elementa med vozlišči poiščemo s funkcijsko aproksimacijo (interpolacijo) v odvisnosti od vozliščnih vrednosti pomikov  $\mathbf{d}_e$  in izbranih interpolacijskih funkcij  $\mathbf{N}_e$ . Če torej poznamo pomike  $\mathbf{d}_e$  v vseh  $n$  vozliščih končnega elementa, lahko določimo pomike  $u$  v katerikoli točki končnega elementa z enačbo:

$$\mathbf{u}_e(x, y, z) = \mathbf{N}_e(x, y, z) \cdot \mathbf{d}_e \quad (3.17)$$

kjer je  $\mathbf{N}_e$  matrika interpolacijskih funkcij končnega elementa:

$$\mathbf{N}_e = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \dots & N_n & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \dots & 0 & N_n & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \dots & 0 & 0 & N_n \end{bmatrix} \quad (3.18)$$

v kateri vsakemu vozlišču pripada lastna interpolacijska funkcija  $N_i(x, y, z)$ .



Slika 3.4 – Razdelitev (diskretizacija) telesa na končne elemente

Vektor vseh vozliščnih komponent pomikov končnega elementa  $\mathbf{d}_e$  je določen kot:

$$\mathbf{d}_e^T = (d_{1x}, d_{1y}, d_{1z}, d_{2x}, d_{2y}, d_{2z} \dots d_{nx}, d_{ny}, d_{nz}) \quad (3.19)$$

in vsebuje zaporedoma vse komponente pomikov ( $DOF$ ) v vseh vozliščih končnega elementa.

Z uporabo nastavka (3.17) lahko zdaj zapišemo razmerje med pomiki in specifičnimi deformacijami končnega elementa (kinematične enačbe (3.10)) v obliki:

$$\boldsymbol{\varepsilon}_e(x, y, z) = \mathbf{L} \cdot \mathbf{u}_e(x, y, z) = \mathbf{L} \cdot \mathbf{N}_e(x, y, z) \cdot \mathbf{d}_e = \mathbf{B}_e(x, y, z) \cdot \mathbf{d}_e \quad (3.20)$$

kje je gradientna matrika končnega elementa  $\mathbf{B}_e$  določena kot:

$$\mathbf{B}_e(x, y, z) = \mathbf{L} \cdot \mathbf{N}_e(x, y, z) \quad (3.21)$$

Konstitutivne enačbe (3.6) pa lahko nato zapišemo na nivoju končnega elementa kot:

$$\boldsymbol{\sigma}_e(x, y, z) = \mathbf{D}_e \cdot \boldsymbol{\varepsilon}_e(x, y, z) = \mathbf{D}_e \cdot \mathbf{B}_e(x, y, z) \cdot \mathbf{d}_e \quad (3.22)$$

### 3.4 Enačbe na nivoju končnega elementa in celotnega telesa

Integralsko ravnotežno enačbo virtualnega dela (3.16) lahko z upoštevanjem MKE-nastavka (3.17), ki velja tudi za virtualne pomike, izrazimo kot:

$$\delta \mathbf{u}_e = \mathbf{N}_e \cdot \delta \mathbf{d}_e \quad (3.23)$$

Posledične zamenjave:

$$\begin{aligned} \delta \mathbf{u}_e^T &= \delta \mathbf{d}_e^T \cdot \mathbf{N}_e^T \\ \mathbf{B}_e^T &= \mathbf{N}_e^T \cdot \mathbf{L}^T \end{aligned} \quad (3.24)$$

zapišemo z upoštevanjem konstitutivne enačbe (3.22) na nivoju končnega elementa kot:

$$\begin{aligned} &\int_{V_e} \delta \mathbf{d}_e^T \cdot \mathbf{B}_e^T \cdot \mathbf{D}_e \cdot \mathbf{B}_e \cdot \mathbf{d}_e \cdot dV \\ &= \delta \mathbf{d}_e^T \cdot \mathbf{N}_e^T \cdot \mathbf{f}_e + \int_{S_{te}} \delta \mathbf{d}_e^T \cdot \mathbf{N}_e^T \cdot \mathbf{t}_e \cdot dS_{te} + \int_{V_e} \delta \mathbf{d}_e^T \cdot \mathbf{N}_e^T \cdot \mathbf{p}_e \cdot dV \end{aligned} \quad (3.25)$$

Vektorja vozliščnih pomikov  $\delta \mathbf{d}_e$  in  $\mathbf{d}_e$  vsebujeta neznanne spremenljivke v znanih točkah (vozliščih) končnega elementa, ki so neodvisne od integralov in ju lahko izstavimo iz integralov. Posledično lahko krajšamo virtualne vozliščne pomike  $\delta \mathbf{d}_e$ . Po ureditvi tako dobimo končno integralsko ravnotežno enačbo virtualnega dela na nivoju končnega elementa v obliki:

$$\int_{V_e} \mathbf{B}_e^T \cdot \mathbf{D}_e \cdot \mathbf{B}_e \cdot dV \cdot \mathbf{d}_e = \mathbf{N}_e^T \cdot \mathbf{f}_e + \int_{S_{te}} \mathbf{N}_e^T \cdot \mathbf{t}_e \cdot dS_{te} + \int_{V_e} \mathbf{N}_e^T \cdot \mathbf{p}_e \cdot dV \quad (3.26)$$

Ob uvedbi zamenjav:

$$\mathbf{K}_e = \int_{V_e} \mathbf{B}_e^T \cdot \mathbf{D}_e \cdot \mathbf{B}_e \cdot dV \quad (3.27)$$

in

$$\mathbf{F}_e = \mathbf{N}_e^T \cdot \mathbf{f}_e + \int_{S_{te}} \mathbf{N}_e^T \cdot \mathbf{t}_e \cdot dS_{te} + \int_{V_e} \mathbf{N}_e^T \cdot \mathbf{p}_e \cdot dV \quad (3.28)$$

kjer je  $\mathbf{K}_e$  togostna matrika končnega elementa in  $\mathbf{F}_e$  vektor delujočih sil v vozliščih končnega elementa, lahko enačbo 3.26 zapišemo v končni matrični obliki na nivoju končnega telesa kot:

$$\mathbf{K}_e \cdot \mathbf{d}_e = \mathbf{F}_e \quad (3.29)$$

Na nivoju celotnega telesa, razdeljenega na  $N$  končnih elementov določenih s skupno  $I$  vozlišči, lahko nato ob seštevanju vozliščnih prispevkov vseh končnih elementov ob upoštevanju njihove lokalno-globalne topologije, zapišemo enako matrično enačbo v obliki:

$$\mathbf{K} \cdot \mathbf{d} = \mathbf{F} \quad (3.30)$$

kjer je  $\mathbf{K}$  globalna togostna matrika celotnega obravnavanega telesa velikosti  $(I \cdot DOF) \times (I \cdot DOF)$ , ki vsebuje prispevke togostnih matrik  $\mathbf{K}_e$  vseh  $N$  končnih elementov,  $\mathbf{d}$  je globalni vektor vseh pomikov v vseh vozliščih celotnega obravnavanega telesa velikosti  $(I \cdot DOF)$  in  $\mathbf{F}$  globalni vektor vseh delujočih vozliščnih sil v vseh vozliščih celotnega obravnavanega telesa velikosti  $(I \cdot DOF)$ , ki vsebuje prispevke vseh zunanjih sil  $\mathbf{F}_e$  vseh  $N$  končnih elementov. Pri tem je  $I$  število vseh vozlišč diskretiziranega telesa in  $DOF$  število prostostnih stopenj posameznega vozlišča.

Vektor vozliščnih pomikov vsebuje neznane pomike vozlišč  $\mathbf{d}_n$  in tudi znane pomike v vozliščih na površini  $S_u$ , na kateri poznamo pomike  $\mathbf{d}_p$ , zato lahko vektor vozliščnih pomikov telesa razdelimo na:

$$\mathbf{d} = \begin{Bmatrix} \mathbf{d}_n \\ \mathbf{d}_p \end{Bmatrix} \quad (3.31)$$

Enako vektor vozliščnih sil vsebuje znane delujoče sile  $\mathbf{F}_p$  (enake 0 ali različne od 0) v vozliščih z neznanimi pomiki  $\mathbf{d}_n$  in neznane reakcijske sile  $\mathbf{F}_n$  v vozliščih z znanimi pomiki  $\mathbf{d}_p$ , zato lahko vektor vozliščnih sil telesa razdelimo na:

$$\mathbf{F} = \begin{Bmatrix} \mathbf{F}_p \\ \mathbf{F}_n \end{Bmatrix} \quad (3.32)$$

Posledično lahko izvedemo dekompozicijo matrične enačbe (3.30) kot:

$$\begin{bmatrix} \mathbf{K}_{np} & \mathbf{K}_{nn} \\ \mathbf{K}_{pp} & \mathbf{K}_{pn} \end{bmatrix} \cdot \begin{Bmatrix} \mathbf{d}_n \\ \mathbf{d}_p \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_p \\ \mathbf{F}_n \end{Bmatrix} \quad (3.33)$$

kar se izkaže kot učinkovita pohitritev pri reševanju sistema enačb, saj iskane neznane pomike določimo iz produkta prve vrstice dekomponirane togostne matrike z vektorjem neznanih vozliščnih pomikov  $\mathbf{d}_n$ , ki je enak vektorju poznanih vozliščnih sil  $\mathbf{F}_p$  oziroma:

$$\mathbf{K}_{np} \cdot \mathbf{d}_n + \mathbf{K}_{nn} \cdot \mathbf{d}_p = \mathbf{F}_p \quad \Rightarrow \quad \mathbf{d}_n = \mathbf{K}_{np}^{-1} \cdot (\mathbf{F}_p - \mathbf{K}_{nn} \cdot \mathbf{d}_p) \quad (3.34)$$

Ko tako določimo vektor neznanih vozliščnih pomikov  $\mathbf{d}_n$ , lahko nato iz produkta druge vrstice dekomponirane togostne matrike z vektorjem znanih vozliščnih pomikov  $\mathbf{d}_p$  določimo vektor neznanih vozliščnih sil  $\mathbf{F}_n$  kot:

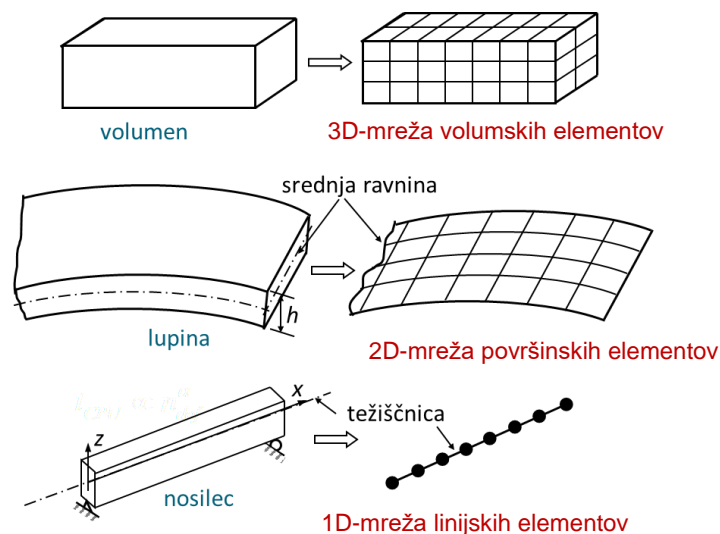
$$\mathbf{F}_n = \mathbf{K}_{pp} \cdot \mathbf{d}_n + \mathbf{K}_{pn} \cdot \mathbf{d}_p \quad (3.35)$$

Po določitvi vseh vozliščnih pomikov  $\mathbf{d}$  in sil  $\mathbf{F}$  lahko nato po posameznih končnih elementih določimo še pripadajoče specifične deformacije  $\boldsymbol{\varepsilon}_e$  (enačba (3.20)) in napetosti  $\boldsymbol{\sigma}_e$  (enačba (3.22)).

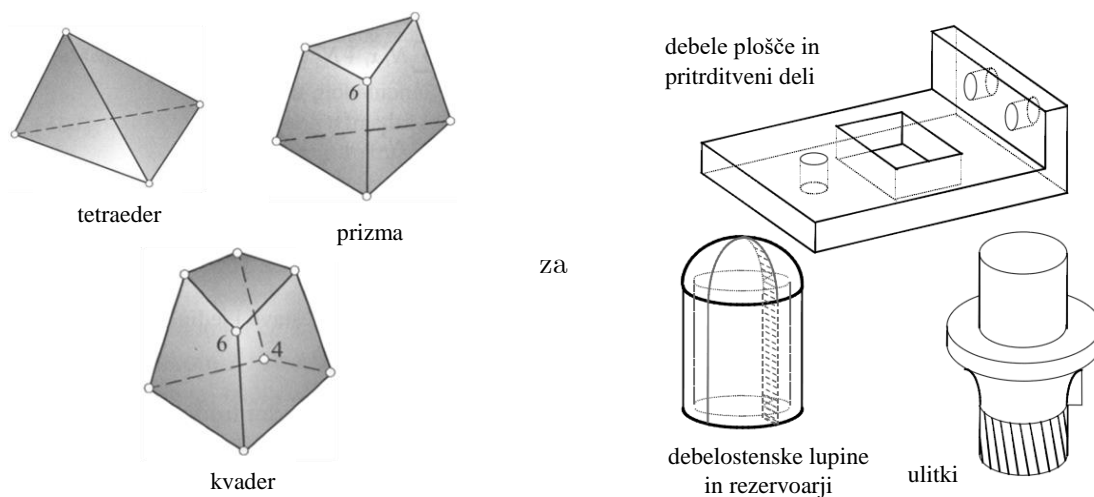
### 3.5 Tipi končnih elementov

Glede na geometrijske značilnosti obravnavanega problema razlikujemo prostorske (ali 3D), površinske (ali 2D) in linijske (ali 1D) končne elemente (slika 3.5).

Kadar so vse tri dimenzije strojnega dela ali v splošnem domene problema približno enakovredne, za prostorsko diskretizacijo uporabimo prostorske (volumske) končne elemente, slika 3.6. Volumski končni elementi so običajno enostavnih prizmatičnih oblik, pri čemer v praksi najpogosteje uporabljamo tetraedre in kvadre.



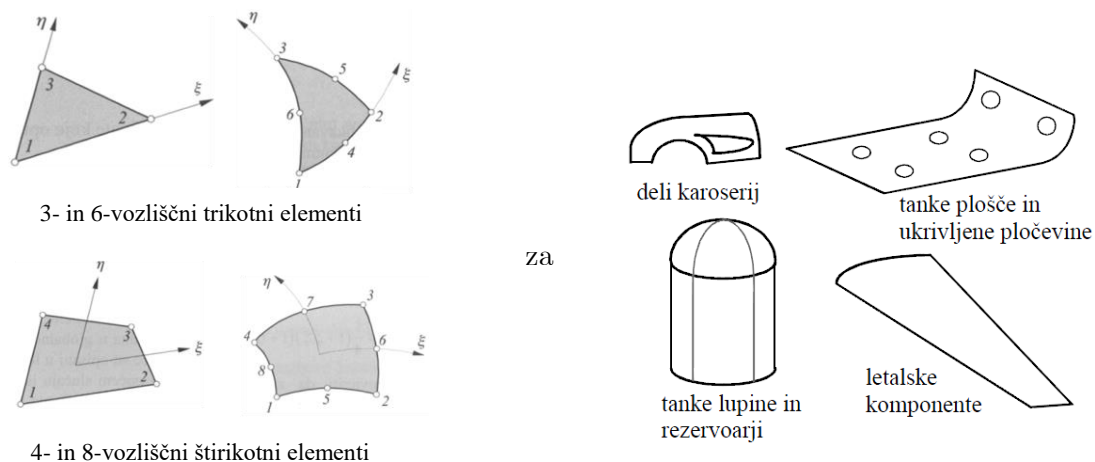
Slika 3.5 – Uporaba končnih elementov različnih dimenzij



Slika 3.6 – Volumski končni elementi

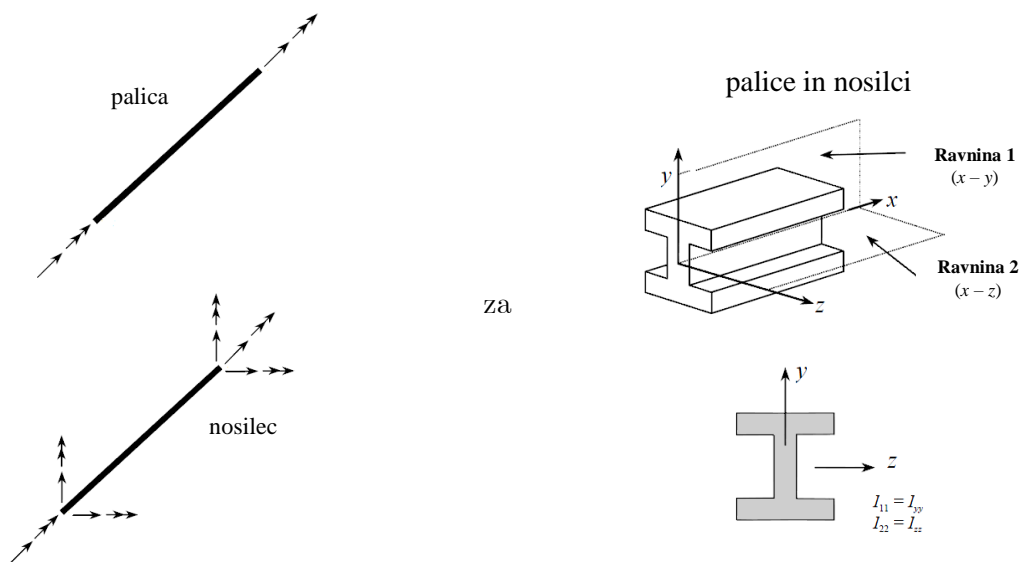
Za prostorsko diskretizacijo strojnih delov ali domen, ki imajo eno dimenzijo mnogo manjšo od drugih dveh dimenzij (plošče, stene, lupine) uporabljamo površinske končne elemente, ki jih

razvrstimo po srednji ravnini volumnov (slika 3.7). Površinski elementi imajo običajno trikotno ali štirikotno obliko in so manjše matematične kompleksnosti kot volumski končni elementi.



Slika 3.7 – Površinski končni elementi

Pri površinskih končnih elementih običajno predpostavimo določena napetostno-deformacijska stanja (ravninsko deformacijsko stanje, ravninsko napetostno stanje, osna simetričnost, ločene membranske in upogibne napetosti), ki omogočijo preprostejšo obravnavo problema v zgolj dveh dimenzijah srednje ravnine. Za določitev napetostno-deformacijskih stanj zunaj srednje ravnine (npr. upogibna napetost na zunanji površini lupine, obodna napetost pri osni simetriji ipd.) pa uporabimo privzete teoretične nastavke za izračun teh vrednosti.



Slika 3.8 – Linijski končni elementi

Pri analizah strojnih delov ali domen, ki imajo eno dimenzijo bistveno daljšo kot drugi dve dimenziji (palice, nosilci), uporabljamo linijske končne elemente, ki jih razvrstimo po srednji liniji oziroma težiščnici delov (slika 3.8). Linijski končni elementi so matematično najmanj kompleksni, saj tudi pri njih običajno predpostavimo določena linijska napetostno-deformacijska stanja (pomiki, odvisni od normalnih sil – palice, pomiki, odvisni od normalnih in prečnih sil, ter zasuki, odvisni od upogibnih ali vzvojnih momentov – nosilci), ki omogočijo preprostejšo obravnavo problema v zgolj eni dimenziji (dolžini) srednje linije. Pri teh končnih elementih predpostavimo, da se prečni prerezi zaradi obremenitve ne deformirajo in imajo tako konstantne karakteristike prerezov (vztrajnostni momenti). Za določitev napetostno-deformacijskih stanj zunaj srednje linije (npr. upogibna napetost na zunanji površini nosilca) uporabimo privzete teoretične nastavke za izračun teh vrednosti.

Za vse vrste končnih elementov je karakteristično, da imajo vozlišča razporejena vedno v vseh skrajnih koncih (linijski na začetku in koncu, površinski v vseh vogalih, volumski v vseh ogliščih). Če so vozlišča končnega elementa določena zgolj v skrajnih koncih, potem je to znak, da imajo opredeljene linearne interpolacijske funkcije osnovne spremenljivke (enačba (3.17)), saj sta za enolično določitev linearne premice dovolj dve točki, v tem primeru končni vozlišči. V primeru uporabe interpolacijskih funkcij višjega reda (kvadratne, kubične ipd.) so za enolično določitev interpolacijske funkcije potrebna, poleg končnih vozlišč, tudi vmesna vozlišča (eno, dve ipd.), kot je prikazano na sliki 3.7. Vsa vozlišča posameznega končnega elementa so običajno lokalno številčena od 1 do  $n$  po določenem pravilu (od leve proti desni, v smeri obratno od urnega kazalca ipd.).

### 3.6 Interpolacijske funkcije končnih elementov

Interpolacijske funkcije končnih elementov so običajno polinomi različnih stopenj, ki jih je mogoče najlažje odvajati in računsko vrednotiti. Polinom v eni dimenziji je v splošnem zapisan kot:

$$P_n(x) = \sum_{i=0}^n a_i x^i \quad (3.36)$$

kjer je število členov polinoma enako  $n + 1$ . V dveh dimenzijah se kompleksnost polinoma poveča:

$$P_n(x, y) = \sum_{k=0}^n a_k x^i y^j \quad i + j \leq k \quad (3.37)$$

kjer  $i$  in  $j$  zavzameta vse možne kombinacije in vrednosti glede na podano omejitev. Število členov polinoma  $n$ -te stopnje je v dveh dimenzijah enako  $(n + 1)(n + 2)/2$ .

Priročno si lahko predstavljamo vse člene dvodimenzionalnega polinoma s Pascalovim trikotnikom, prikazanim na sliki 3.9.

Stopnja in ime polinoma	Členi polinoma	Število členov
$n = 0$ – konstanta	1	1
$n = 1$ – linearni	$x \quad y$	3
$n = 2$ – kvadratni	$x^2 \quad xy \quad y^2$	6
$n = 3$ – kubični	$x^3 \quad x^2y \quad xy^2 \quad y^3$	10
$n = 4$ – kvartični	$x^4 \quad x^3y \quad x^2y^2 \quad xy^3 \quad y^4$	15
$n = 5$ – kvintični	$x^5 \quad x^4y \quad x^3y^2 \quad x^2y^3 \quad xy^4 \quad y^5$	21
$n = 6$ – sekstični	$x^6 \quad x^5y \quad x^4y^2 \quad x^3y^3 \quad x^2y^4 \quad xy^5 \quad y^6$	28

Slika 3.9 – Pascalov trikotnik dvodimenzionalnih polinomov

V treh dimenzijah so polinomi določeni z naslednjim zapisom:

$$P_n(x, y, z) = \sum_{m=0}^n a_m x^i y^j z^k \quad i + j + k \leq m \quad (3.38)$$

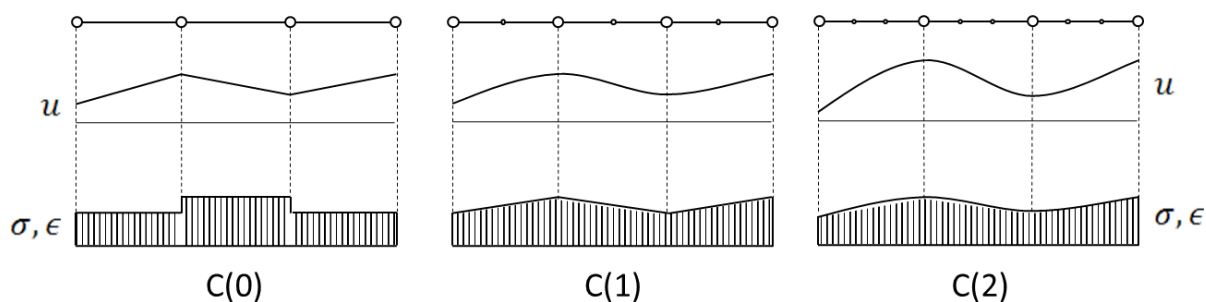
kjer  $i, j$  in  $k$  zavzamejo vse možne kombinacije in vrednosti glede na podano omejitev. Število členov polinoma  $n$ -te stopnje je v treh dimenzijah enako  $(n + 1)(n + 2)(n + 3)/6$ .

V praksi najpogosteje uporabljamo končne elemente z linearnimi interpolacijskimi funkcijami (polinomi stopnje  $n = 1$ ), ki so v splošnem zapisane kot:

$$\begin{aligned} 1D: \quad & N_i(x) = A_i + B_i \cdot x \\ 2D: \quad & N_i(x, y) = A_i + B_i \cdot x + C_i \cdot y \\ 3D: \quad & N_i(x, y, z) = A_i + B_i \cdot x + C_i \cdot y + D_i \cdot z \end{aligned} \quad (3.39)$$

pri čemer so konstante  $A_i, B_i, C_i$  in  $D_i$  odvisne od geometrije končnega elementa in položaja vozlišča, ki mu pripada posamezna interpolacijska funkcija. Takšni elementi imajo vozlišča zgolj v ogliščih končnih elementov in so matematično najmanj zahtevni.

Linearne interpolacijske funkcije omogočajo linearno aproksimacijo poteka osnovne neznane veličine po končnem elementu (pomika) in konstantno aproksimacijo poteka izvedenih veličin (specifične deformacije in napetosti), kar zagotavlja zgolj osnovno  $C(0)$  kompatibilnost rešitev (slika 3.10). Pomiki so po obravnavanem problemu opisani zvezno, vendar odsekovno linearno, torej ne gladko. Specifične deformacije in napetosti so opisane nezvezno, v vsakem končnem elementu s povprečno konstantno vrednostjo. Za dober približek natančni rešitvi običajno potrebujemo večje število teh elementov v območju večjih gradientov izračunanih veličin.



Slika 3.10 – Vrste kompatibilnosti končnih elementov v odvisnosti od stopnje uporabljenih interpolacijskih funkcij

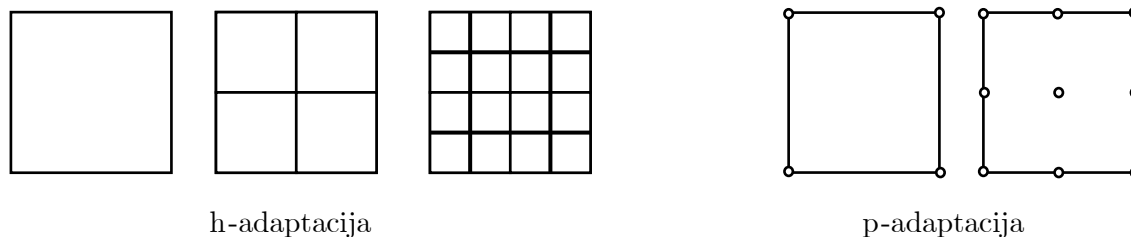
Višjo stopnjo kompatibilnosti rešitev  $C(1)$  zagotavljajo končni elementi s kvadratnimi (ali tudi paraboličnimi) interpolacijskimi funkcijami (polinomi stopnje  $n = 2$ ), zapisanimi v splošnem kot:

$$\begin{aligned}
 1D: \quad & N_i(x) = A_i + B_i \cdot x + C_i \cdot x^2 \\
 2D: \quad & N_i(x, y) = A_i + B_i \cdot x + C_i \cdot x^2 + D_i \cdot y + E_i \cdot y^2 + F_i \cdot x \cdot y \\
 3D: \quad & N_i(x, y, z) = A_i + B_i \cdot x + C_i \cdot x^2 + D_i \cdot y + E_i \cdot y^2 + F_i \cdot x \cdot y \\
 & \quad \quad \quad + G_i \cdot z + H_i \cdot z^2 + J_i \cdot x \cdot z + K_i \cdot y \cdot z
 \end{aligned} \tag{3.40}$$

kjer so konstante  $A_i \dots K_i$  zopet odvisne od geometrije končnega elementa in položaja vozlišča, ki mu pripada posamezna interpolacijska funkcija. Kompleksnost funkcij je večja in posledično je višja tudi matematična zahtevnost njihovega reševanja. Takšni elementi imajo vozlišča v ogliščih končnih elementov in tudi vmesna vozlišča med njimi, saj je za enolično določitev kvadratne funkcije potrebno poznavanje njihove vrednosti v treh točkah. Kvadratne interpolacijske funkcije omogočajo kvadratno aproksimacijo poteka osnovne spremenljivke po končnem elementu (pomika) in linearno aproksimacijo poteka izvedenih veličin (specifične deformacije in napetosti), kot prikazuje slika 3.10. Pomiki so po obravnavanem problemu opisani zvezno in gladko, medtem ko so specifične deformacije in napetosti opisane zvezno, vendar odsekovno linearno, torej ne gladko. Za dober približek natančni rešitvi potrebujemo tako manjše število kvadratnih končnih elementov v območju večjih gradientov izračunanih veličin, vendar je za to potrebno večje število zahtevnejših računskih operacij.

Zvezen in gladek opis tako osnovnih (pomikov) kot tudi izvedenih veličin (specifičnih deformacij in napetosti) in s tem  $C(2)$ -kompatibilnost omogočajo šele polinomi tretjega in višjega reda (polinomi stopnje  $n \geq 3$ ), kot prikazuje slika 3.10. Končni elementi s takšnimi interpolacijskimi funkcijami višjega reda imajo poleg vozlišč v ogliščih končnih elementov med njimi še  $n - 1$  vmesnih vozlišč, saj je za enolično določitev  $n$ -tega interpolacijskega polinoma potrebno poznavanje njegove vrednosti v  $n + 1$  točkah. Kompleksnost funkcij je večja in posledično je višja tudi matematična zahtevnost njihovega reševanja. Za dober približek natančni rešitvi potrebujemo z višanjem reda interpolacijskega polinoma vedno manjše število končnih elementov, vendar se s tem povečuje tudi število zahtevnejših računskih operacij.





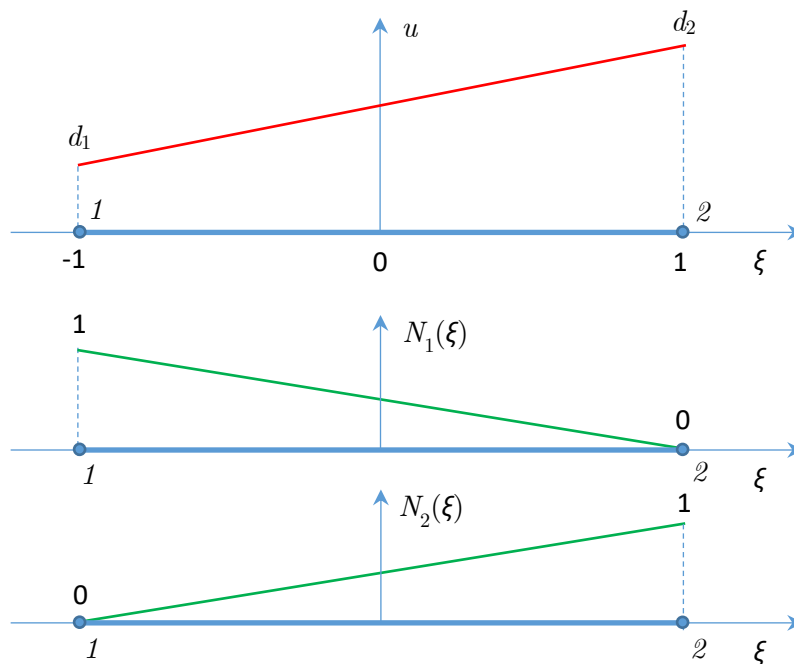
Slika 3.11 – Pristopi k izboljšanju kvalitete numeričnih rezultatov

S spremembo reda polinoma interpolacijskih funkcij tako enostavno izboljšamo kvaliteto rezultatov, čeprav se osnovna mreža končnih elementov pri tem sploh ne spremeni. Takšen način izboljšave mreže končnih elementov imenujemo  $p$ -adaptacija, kjer je  $p$  sinonim za večanje stopnje interpolacijskega polinoma (slika 3.11). Bolj običajna metoda za izboljšanje rezultatov računalniških simulacij je  $h$ -adaptacija, ki temelji na povečanju števila enostavnih končnih elementov z linearno interpolacijo, ki se pri tem zmanjšujejo –  $h$  je tako sinonim za velikost končnega elementa. Vendar lahko veliko število majhnih elementov privede do velike numerične napake rezultatov. Izkaže se, da numerični rezultati hitreje konvergirajo k natančni rešitvi pri  $p$ -adaptaciji mreže končnih elementov [3.7]. V večini primerov za izboljšanje rezultatov zadostuje uporaba končnih elementov s kvadratnimi interpolacijskimi funkcijami in le redki računalniški programi vsebujejo definicije za interpolacijskih funkcij s polinomi reda, višjega od 3.

### 3.7 Interpolacijske funkcije v naravnih koordinatah

Interpolacijske funkcije končnih elementov so običajno zapisane v naravnih koordinatah  $\langle 0,1 \rangle$  oziroma priročneje v območju  $\langle -1, +1 \rangle$ , ki jih poimenujemo tudi lokalne koordinate. Takšen zapis je ugodnejši, saj omogoča osnovni zapis relacij končnih elementov in izvedbo osnovnih matematičnih operacij (integracija) v vedno enakem, enotskem območju.

Najenostavnejši linijski končni element z linearno interpolacijsko funkcijo pomikov je prikazan na sliki 3.12, kjer je koordinata lokalnega sistema v naravnih koordinatah običajno označena z grško črko  $\xi$  (ksi). V naravnem koordinatnem sistemu je element določen tako, da ima prvo vozlišče koordinato  $\xi_1 = -1$ , drugo pa  $\xi_2 = +1$ , torej je skupna dolžina končnega elementa  $l = \xi_2 - \xi_1 = 2$ .



Slika 3.12 – Linijski končni element z linearno interpolacijsko funkcijo pomikov

Vsakemu vozlišču končnega elementa pripadata linearni interpolacijski funkciji, ki ju zapišemo kot:

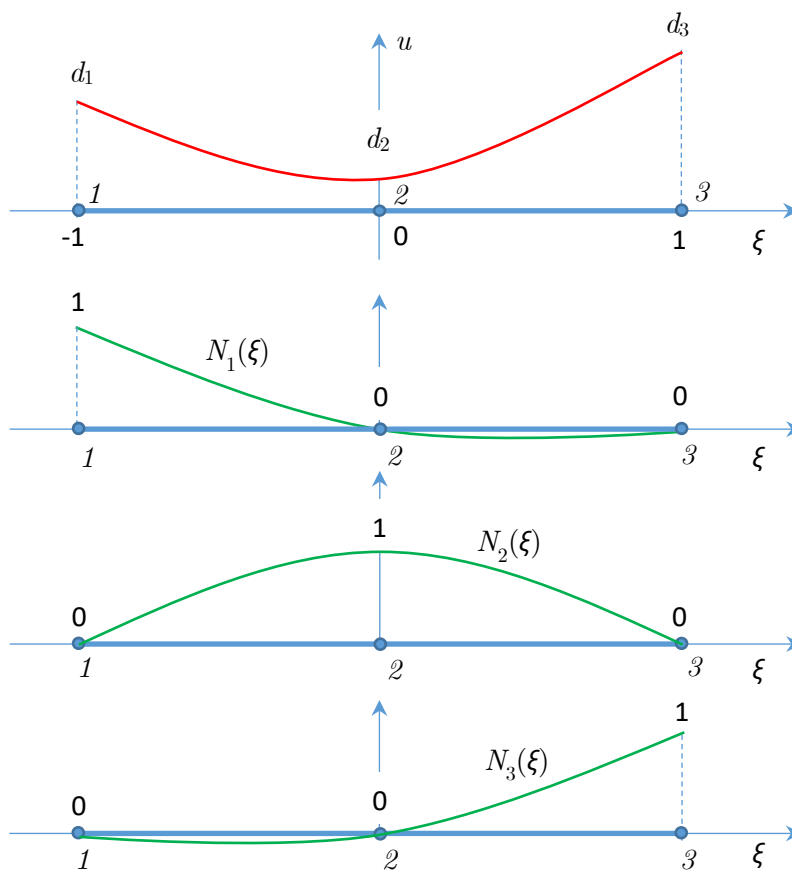
$$N_1(\xi) = \frac{1 - \xi}{2} \quad \text{in} \quad N_2(\xi) = \frac{1 + \xi}{2} \quad (3.41)$$

kjer imata interpolacijski funkciji vrednost 1 v vozlišču, ki mu pripadata, in vrednost 0 v drugem vozlišču.

Interpolacijske funkcije linijskega končnega elementa s kvadratno interpolacijsko funkcijo (slika 3.13) so zapisane kot:

$$N_1(\xi) = \frac{1}{2}\xi \cdot (\xi - 1), \quad N_2(\xi) = (1 + \xi) \cdot (1 - \xi) \quad \text{in} \quad N_3(\xi) = \frac{1}{2}\xi \cdot (\xi + 1) \quad (3.42)$$

kjer imajo interpolacijske funkcije vrednost 1 v vozlišču, ki mu pripadajo, in vrednost 0 v preostalih dveh vozliščih.



Slika 3.13 – Linijski končni element s kvadratno interpolacijo pomikov

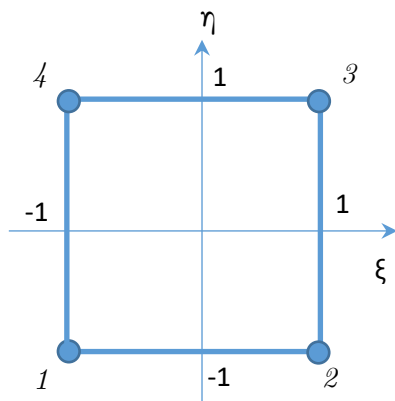
Interpolacijske funkcije ravninskih in prostorskih končnih elementov so podobno določene v vsakem vozlišču končnega elementa, s tem da z vsako dimenzijo pridobimo eno koordinato lokalnega koordinatnega sistema. Teoretični ravninski končni element je določen z dvema koordinatama  $\xi$  (ksi) in  $\eta$  (eta) v območju  $(-1, +1)$ , kot prikazuje slika 3.14. Interpolacijske funkcije so določene za vsako vozlišče in so linearne v vsaki smeri elementa (slika 3.15). Zato jih imenujemo tudi bilinearne interpolacijske funkcije. Zapisane so v naslednji obliki:

$$\begin{aligned} N_1(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 - \eta) & N_2(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 + \eta) \\ N_3(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 + \eta) & N_4(\xi, \eta) &= \frac{1}{4}(1 + \xi)(1 - \eta) \end{aligned} \quad (3.43)$$

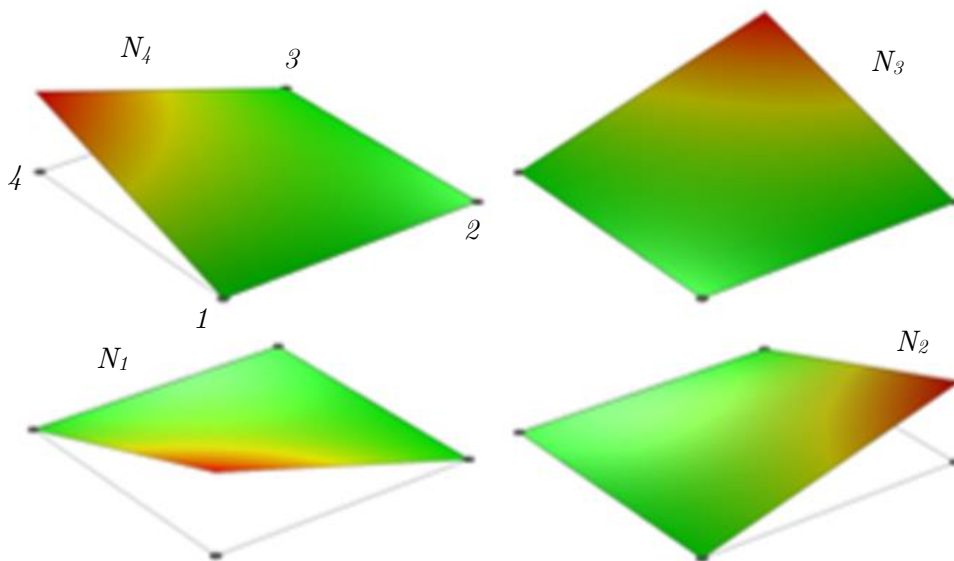
kjer imajo interpolacijske funkcije vrednost 1 v vozlišču, ki mu pripadajo, in vrednost 0 v preostalih vozliščih.

Kvadratne ali parabolne končne elemente dobimo tako, da na vsaki stranici vstavimo dodatno vmesno vozlišče. V vsakem vozlišču je določena interpolacijska funkcija in pomiki se po elementu spreminjajo po parabolni površini. Grafični prikaz posameznih interpolacijskih funkcij kvadratnega končnega elementa je podan na sliki 3.16. Na sliki vidimo, da je prikazanih devet interpolacijskih funkcij, osem za vsa robna vozlišča in ena za vozlišče v sredini končnega elementa. Pri metodi končnih elementov se namreč uporabljata dve družini višjerednih končnih elementov. Lagrangeevi končni elementi imajo vmesna vozlišča na vseh robovih in v središču

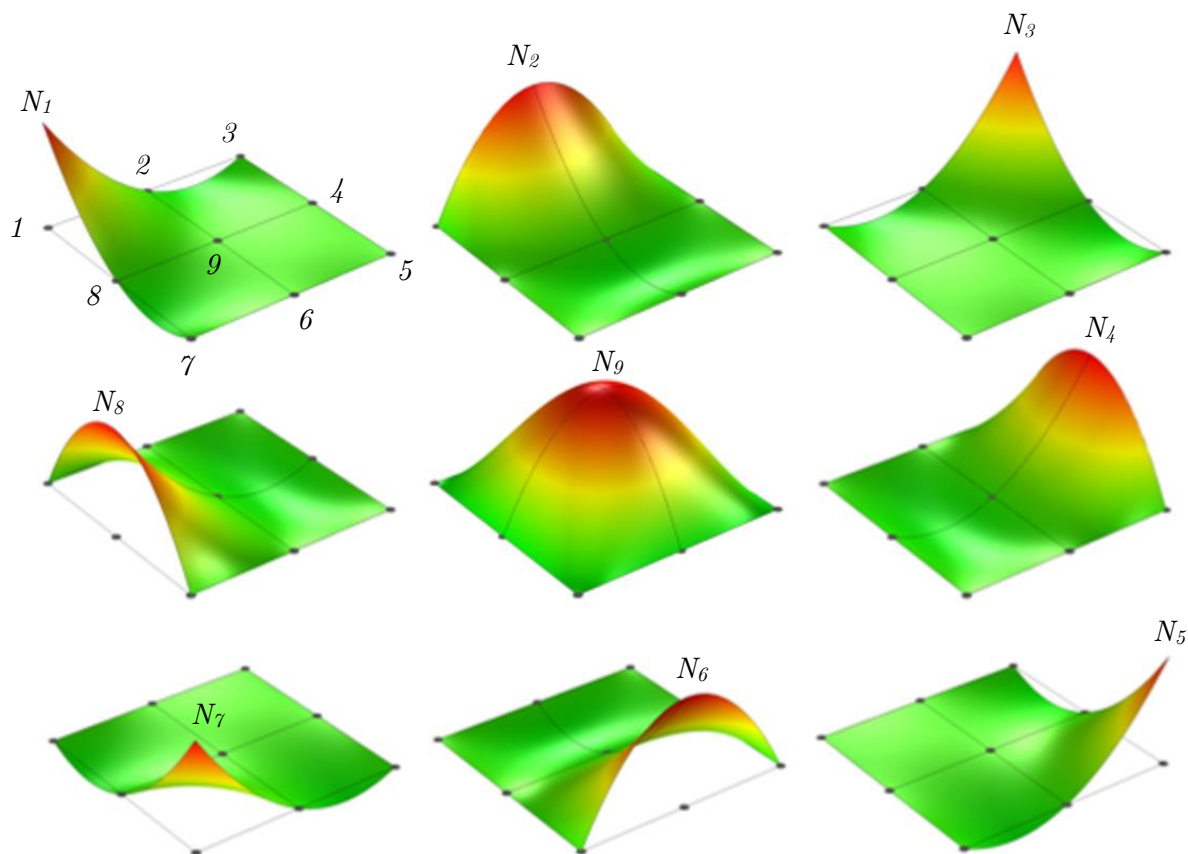
končnega elementa in so opisani s popolnimi interpolacijskimi polinomi ustrezne stopnje, zato tudi vmesno vozlišče na sredini. Končni elementi brez vozlišč na sredini pa se imenujejo Serendipity, kjer so interpolacijske funkcije zapisane z nepopolnimi polinomi pri čemer za njihovo določitev zadoščajo zgolj vozlišča na robovih končnih elementov. Lagrangeovi končni elementi so natančnejši, vendar računsko precej zahtevnejši od končnih elementov Serendipity. V praksi najpogosteje uporabljamo končne elemente Serendipity.



Slika 3.14 – Ravninski končni element v lokalnem koordinatnem sistemu



Slika 3.15 – Grafični prikaz linearnih interpolacijskih funkcij ravninskih končnih elementov [3.8]



Slika 3.16 – Grafični prikaz kvadratnih interpolacijskih funkcije površinskih končnih elementov [3.8]

Podobno so določene interpolacijske funkcije v treh dimenzijah, kjer je teoretičnemu površinskemu elementu dodana še tretja razsežnost  $\zeta$  (zeta). Tako dobimo lokalni koordinatni sistem  $(\xi, \eta, \zeta)$ , kjer ima vsaka koordinata vrednosti v območju  $(-1, +1)$ . Interpolacijske funkcije so predpisane v vseh vozliščih končnega elementa in določene s polinomi različnih stopenj. Če so ravninski linearni končni elementi dejansko bilinearni, so prostorski trilinearni, saj je v vsakem vozlišču določen polinom s tremi spremenljivkami. Linearne interpolacijske funkcije heksaedričnega končnega elementa so zapisane kot [3.9]:

$$\begin{aligned}
 N_1(\xi, \eta, \zeta) &= \frac{1}{8}(1 - \xi)(1 - \eta)(1 - \zeta) & N_2(\xi, \eta, \zeta) &= \frac{1}{8}(1 + \xi)(1 - \eta)(1 - \zeta) \\
 N_3(\xi, \eta, \zeta) &= \frac{1}{8}(1 + \xi)(1 + \eta)(1 - \zeta) & N_4(\xi, \eta, \zeta) &= \frac{1}{8}(1 - \xi)(1 + \eta)(1 - \zeta) \\
 N_5(\xi, \eta, \zeta) &= \frac{1}{8}(1 - \xi)(1 - \eta)(1 + \zeta) & N_6(\xi, \eta, \zeta) &= \frac{1}{8}(1 + \xi)(1 - \eta)(1 + \zeta) \\
 N_7(\xi, \eta, \zeta) &= \frac{1}{8}(1 + \xi)(1 + \eta)(1 + \zeta) & N_8(\xi, \eta, \zeta) &= \frac{1}{8}(1 - \xi)(1 + \eta)(1 + \zeta)
 \end{aligned} \tag{3.44}$$

Z dodajanjem vmesnih vozlišč in povečevanjem stopnje polinoma interpolacijskih funkcij lahko natančnost tudi teh elementov izboljšamo.

Za monotono konvergenco rešitve morajo interpolacijske funkcije zagotavljati konformnost končnih elementov, kar pomeni da izpolnjujejo naslednje zahteve:

- omogočati morajo opis gibanja togega telesa (sposobne opisati polje  $\boldsymbol{\sigma}, \boldsymbol{\varepsilon} = \mathbf{0}$ );
- sposobni morajo biti opisati polje konstantnih specifičnih deformacij  $\boldsymbol{\varepsilon} = \text{konst.}$ ;
- interpolacijske funkcije morajo biti po elementu zvezne in gladke ter
- zagotovljena mora biti kompatibilnost interpoliranih veličin preko robov končnega elementa.

Za izpolnjevanje prvih dveh zahtev mora biti izpolnjen pogoj, da je vsota vseh interpolacijskih funkcij kjerkoli po končnem elementu enaka 1 ali:

$$\sum_{i=1}^n N_i(\xi, \eta, \zeta) = 1 \quad (3.45)$$

### 3.8 Izoparametrični končni elementi

Osnovni princip izoparametričnih elementov temelji na uporabi interpolacijskih funkcij končnih elementov za preslikavo idealne geometrije končnih elementov v naravnih koordinatah v končne elemente v realnih koordinatah, kot prikazuje slika 3.17 [3.5]. Funkcije za preslikavo geometrije imenujemo tudi oblikovne funkcije in v primeru izoparametričnih elementov so le-te enake interpolacijskim funkcijam končnih elementov.

Tako lahko za izoparametrične končne elemente zapišemo zvezo med razsežnostmi končnega elementa v naravnih koordinatah  $\boldsymbol{\xi}_e = (\xi, \eta, \zeta)$  in realnih koordinatah  $\boldsymbol{x}_e = (x, y, z)$  kot:

$$\boldsymbol{x}_e(x, y, z) = \mathbf{N}_e(\xi, \eta, \zeta) \cdot \mathbf{X}_e \quad (3.46)$$

kjer je  $\boldsymbol{x}_e(x, y, z)$  pripadajoča pozicija realnega končnega elementa,  $\mathbf{N}_e(\xi, \eta, \zeta)$  matrika oblikovnih (interpolacijskih) funkcij (enačba (3.18)), ovrednotena v naravnih koordinatah, ter  $\mathbf{X}_e$  vektor vozliščnih pozicij končnega elementa v realnih koordinatah, zapisan kot:

$$\mathbf{X}_e^T = (x_1, y_1, z_1, x_2, y_2, z_2 \cdots x_n, y_n, z_n) \quad (3.47)$$

Izoparametrična formulacija končnih elementov [3.5] omogoča generiranje nepravokotnih končnih elementov, ki lahko imajo tudi ukrivljene stranice (slika 3.17). S tem je omogočena natančna diskretizacija robov lukenj, saj lahko robove izoparametričnih elementov preslikamo v poljubno ukrivljenost, ki jih omogočajo stopnje uporabljenih oblikovnih funkcij. Tako lahko na primer z uporabo končnih elementov s kvadratnimi interpolacijskimi (oblikovnimi) funkcijami natančno geometrijsko opišemo rob okrogle luknje. Zaradi navedene prednosti so številni komercialni računalniški programi za analize po metodi končnih elementov privzeli izoparametrično formulacijo vsebovanih končnih elementov.

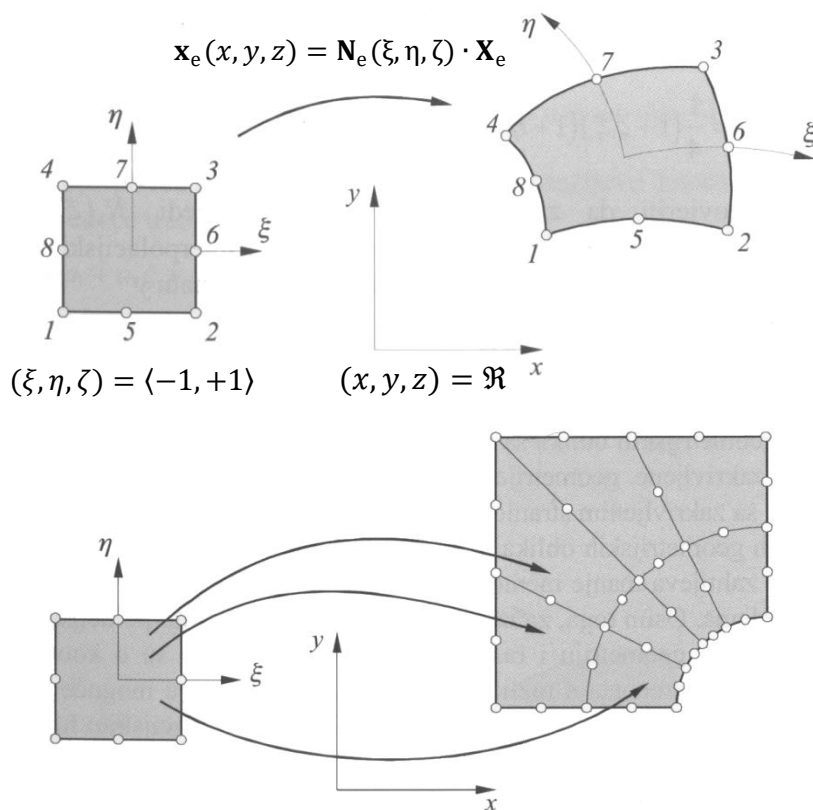
Za preslikavo variacije diferencialnih veličin pa uporabljamo Jacobijevo transformacijsko matriko  $\mathbf{J}$ , ki določa razmerja med razsežnostmi dveh prostorov  $d\boldsymbol{x}_e = \mathbf{J} \cdot d\boldsymbol{\xi}_e$  in jo v splošnem zapišemo kot:

$$\mathbf{J} = \frac{d\mathbf{x}_e}{d\xi_e} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \quad (3.48)$$

Tako lahko zapišemo razmerje med enakovrednima diferencialno majhnima deloma volumna v naravnih in realnih koordinatah kot:

$$dV = dx \cdot dy \cdot dz = \det \mathbf{J} \cdot d\xi \cdot d\eta \cdot d\zeta \quad (3.49)$$

Kar pomeni, da lahko vse osnovne matematične operacije (izračun togostne matrike, porazdelitve obremenitve ipd.) izvedemo v enotskem prostoru naravnih koordinat in nato rezultate preslikamo v ustrezni realni prostor z uporabo Jacobijeve transformacijske matrike  $\mathbf{J}$  oziroma njene determinante.



Slika 3.17 – Koncept preslikave izoparametričnih končnih elementov in idealnega prostora naravnih koordinat v realni prostor

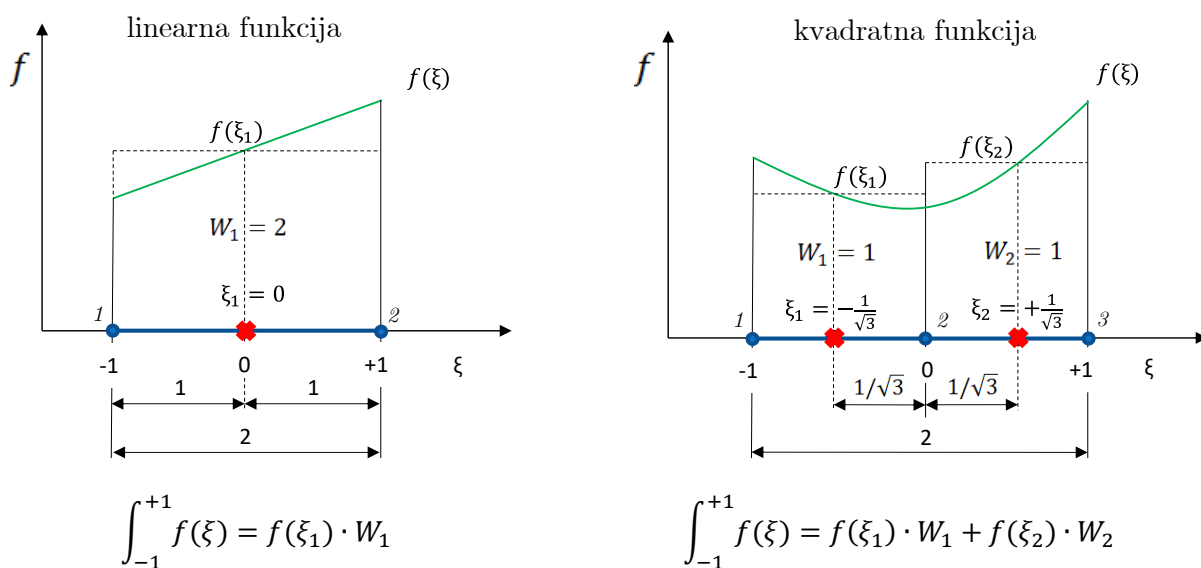
### 3.9 Numerična integracija togostne matrike in porazdeljenih obremenitev

Z metodami numerične integracije je mogoče izračunati vrednost integrala poljubne funkcije vzdolž poljubno dolgega intervala na osnovi ustrezno izbrane aproksimacije. Polinome običajno dokaj natančno numerično integriramo z uporabo metode Gaussove integracije [3.10] [3.11], kot

prikazuje slika 3.18, kjer integral polinoma ovrednotimo kot uteženo vsoto vrednosti polinoma  $f(\xi)$  v točno določenih integracijskih točkah  $i = 1 \dots n$  na integracijskem intervalu  $\langle -1, +1 \rangle$  kot:

$$\int_{-1}^{+1} f(\xi) d\xi = \sum_{i=1}^n f(\xi_i) \cdot W_i \quad (3.50)$$

kjer so  $W_i$  integracijske uteži, ki pripadajo vsaki integracijski točki  $i = 1 \dots n$ . Potrebno število integracijskih točk je enako stopnji interpolacijskega polinoma  $n$ , pri čemer najdemo integracijska pravila v smislu pozicij integracijskih točk in pripadajočih integracijskih uteži za posamezno stopnjo in razsežnost interpolacijskih polinomov v ustrezni literaturi [3.10].



Slika 3.18 – Numerična integracija linijskega končnega elementa z linearno in kvadratno interpolacijsko funkcijo

Numerično integracijo togostne matrike izoparametričnega končnega elementa v naravnih koordinatah s preslikavo v realne koordinate ob upoštevanju enačb (3.27), (3.49) in (3.50) izvedemo kot:

$$\begin{aligned} \mathbf{K}_e &= \int_{V_e} \mathbf{B}_e^T \cdot \mathbf{D}_e \cdot \mathbf{B}_e \cdot dV = \int_{V_e} \mathbf{B}_e^T(x, y, z) \cdot \mathbf{D}_e \cdot \mathbf{B}_e(x, y, z) \cdot dx \cdot dy \cdot dz \\ &= \int_{V_e} \mathbf{B}_e^T(\xi, \eta, \zeta) \cdot \mathbf{D}_e \cdot \mathbf{B}_e(\xi, \eta, \zeta) \cdot \det \mathbf{J} \cdot d\xi \cdot d\eta \cdot d\zeta \\ &= \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \mathbf{B}_e^T(\xi_{i,j,k}, \eta_{i,j,k}, \zeta_{i,j,k}) \cdot \mathbf{D}_e \cdot \mathbf{B}_e(\xi_{i,j,k}, \eta_{i,j,k}, \zeta_{i,j,k}) \cdot W_{i,j,k} \cdot \det \mathbf{J} \end{aligned} \quad (3.51)$$

kjer argument integrala  $\mathbf{B}_e^T \cdot \mathbf{D}_e \cdot \mathbf{B}_e$  ovrednotimo v vseh integracijskih točkah končnega elementa  $i, j, k$  v naravnih koordinatah in pomnožimo s pripadajočimi integracijskimi utežmi  $W_{i,j,k}$  ter nato vse prispevke seštejemo in pomnožimo še z  $\det \mathbf{J}$ .



Na podoben način numerično integriramo tudi porazdeljene specifične obremenitve  $\mathbf{t}_e$  po površini končnega elementa  $S_{te}$  :

$$\begin{aligned} \mathbf{F}_{S_{te}} &= \int_{S_{te}} \mathbf{N}_e^T \cdot \mathbf{t}_e \cdot dS_{te} = \int_{S_{te}} \mathbf{N}_e^T(x, y, z) \cdot \mathbf{t}_e(x, y, z) \cdot dx \cdot dy \cdot dz \\ &= \int_{S_{te}} \mathbf{N}_e^T(\xi, \eta) \cdot \mathbf{t}_e(\xi, \eta) \cdot \det \mathbf{J} \cdot d\xi \cdot d\eta \\ &= \sum_{i=1}^n \sum_{j=1}^n \mathbf{N}_e^T(\xi_{i,j}, \eta_{i,j}) \cdot \mathbf{t}_e(\xi_{i,j}, \eta_{i,j}) \cdot W_{i,j} \cdot \det \mathbf{J} \end{aligned} \quad (3.52)$$

in specifične telesne sile  $\mathbf{p}_e$  po volumnu končnega elementa  $V_e$ :

$$\begin{aligned} \mathbf{F}_{V_e} &= \int_{V_e} \mathbf{N}_e^T \cdot \mathbf{p}_e \cdot dV = \int_{V_e} \mathbf{N}_e^T(x, y, z) \cdot \mathbf{p}_e(x, y, z) \cdot dx \cdot dy \cdot dz \\ &= \int_{V_e} \mathbf{N}_e^T(\xi, \eta, \zeta) \cdot \mathbf{p}_e(\xi, \eta, \zeta) \cdot \det \mathbf{J} \cdot d\xi \cdot d\eta \cdot d\zeta \\ &= \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \mathbf{N}_e^T(\xi_{i,j,k}, \eta_{i,j,k}, \zeta_{i,j,k}) \cdot \mathbf{p}_e(\xi_{i,j,k}, \eta_{i,j,k}, \zeta_{i,j,k}) \cdot W_{i,j,k} \cdot \det \mathbf{J} \end{aligned} \quad (3.53)$$

Interpolacijske polinomske funkcije končnih elementov različnih stopenj so eno-, dvo- ali tro-razsežne, zato je temu sorazmerno tudi potrebno določeno število integracijskih točk, preglednica 3.1.

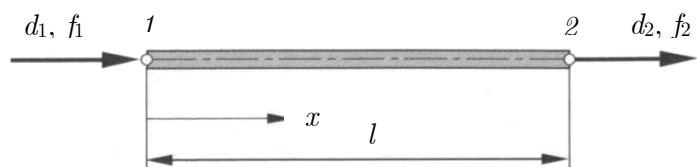
Preglednica 3.1 – Število potrebnih integracijskih točk v odvisnosti od stopnje interpolacijskega polinoma in razsežnosti končnega elementa

Stopnja interpolacijskega polinoma	Potrebno število integracijskih točk		
	1D	2D	3D
$n = 1$ – linearni	1	1	1
$n = 2$ – kvadratni	2	4	8
$n = 3$ – kubični	3	9	27

### 3.10 Osnovne enačbe MKE na primeru linijskega končnega elementa

Najpreprostejši končni element je linijski palični element s konstantnim prerezom  $A$ , z eno prostostno stopnjo (pomik vzdolž dolžine palice) in linearno interpolacijo ter dvema vozliščema na obeh koncih palice (slika 3.12). Takšen palični končni element lahko prenaša zgolj vzdolžne, normalne obremenitve, pri čemer se posledično v elementu pojavijo konstantne specifične deformacije in napetosti (slika 3.19). Vektorja vozliščnih pomikov in vozliščnih sil sta enaka:

$$\mathbf{d}_e = \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} \quad \text{in} \quad \mathbf{f}_e = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \quad (3.54)$$



Slika 3.19 – Palični končni element z linearno interpolacijo pomikov

Palični končni element z linearno interpolacijo ima glede na MKE-nastavek (3.17) pomik  $\mathbf{u}_e(x) = u_x$  opisan z naslednjo aproksimacijo funkcijo:

$$\mathbf{u}_e(x) = u_x = \mathbf{N}_e(x) \cdot \mathbf{d}_e \quad (3.55)$$

pri čemer je matrika interpolacijskih funkcij končnega elementa zapisana kot:

$$\mathbf{N}_e(x) = [N_1(x) \quad N_2(x)] \quad (3.56)$$

Interpolacijske funkcije za posamezno vozlišče imajo obliko polinoma prve stopnje  $N_i(x) = A_i + B_i \cdot x$  in ju zapišemo kot:

$$\begin{aligned} N_1(x) &= A_1 + B_1 \cdot x \\ N_2(x) &= A_2 + B_2 \cdot x \end{aligned} \quad (3.57)$$

Konstante interpolacijskih polinomov določimo glede na potrebni pogoj izračuna ustreznega pomika v pripadajočem vozlišču z aproksimacijsko funkcijo (3.55), ki jo v razviti obliki zapišemo kot:

$$u_x = \mathbf{N}_e(x) \cdot \mathbf{d}_e = [N_1(x) \quad N_2(x)] \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = N_1(x) \cdot d_1 + N_2(x) \cdot d_2 \quad (3.58)$$

Tako lahko za posamezno vozlišče zapišemo naslednje pogoje:

$$\begin{aligned} \text{vozlišče 1: } x_1 = 0, \quad u_{x_1} = d_1 &\quad \Rightarrow \quad N_1(x_1) = 1 \quad \text{in} \quad N_2(x_1) = 0 \\ \text{vozlišče 2: } x_2 = l, \quad u_{x_2} = d_2 &\quad \Rightarrow \quad N_1(x_2) = 0 \quad \text{in} \quad N_2(x_2) = 1 \end{aligned} \quad (3.59)$$

na osnovi katerih lahko izpeljemo konstante interpolacijskih polinomov. Interpolacijske polinome linijskega paličnega elementa v naravnih koordinatah zapišemo kot:

$$\begin{aligned} N_1(x) &= \frac{l-x}{l} \\ N_2(x) &= \frac{x}{l} \end{aligned} \quad (3.60)$$

oziroma matriko interpolacijskih funkcij končnega elementa kot:

$$\mathbf{N}_e(x) = [N_1(x) \quad N_2(x)] = \left[ \frac{l-x}{l} \quad \frac{x}{l} \right] \quad (3.61)$$

Aproksimacijsko funkcijo poteka pomika  $u_x$  po dolžini palice  $x = \langle 0, l \rangle$  lahko tako zapišemo:

$$u_x = N_1(x) \cdot d_1 + N_2(x) \cdot d_2 = \frac{l-x}{l} \cdot d_1 + \frac{x}{l} \cdot d_2 = d_1 + \frac{d_2 - d_1}{l} \cdot x \quad (3.62)$$

Ker vsebuje matrika diferencialnega operatorja (3.4) zgolj en člen  $\mathbf{L} = \left[ \frac{\partial}{\partial x} \right]$ , lahko gradientno matriko (3.21) zapišemo kot:

$$\mathbf{B}_e(x) = \mathbf{L} \cdot \mathbf{N}_e(x) = \left[ \frac{\partial}{\partial x} \right] \cdot \left[ \frac{l-x}{l} \quad \frac{x}{l} \right] = \left[ -\frac{1}{l} \quad +\frac{1}{l} \right] \quad (3.63)$$

ter posledično specifično deformacijo končnega elementa (enačba (3.20)):

$$\boldsymbol{\varepsilon}_e(x) = \varepsilon_x = \mathbf{B}_e(x) \cdot \mathbf{d}_e = \left[ -\frac{1}{l} \quad +\frac{1}{l} \right] \cdot \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \frac{d_2 - d_1}{l} = \text{konst.} \quad (3.64)$$

Ker imamo v primeru linijskega paličnega nosilca zgolj eno prostostno stopnjo (vzdolžni pomik  $u_x$ ), konstitutivna matrika elastičnosti  $\mathbf{D}_e$  posledično vsebuje zgolj en člen  $\mathbf{D}_e = [E]$  in lahko konstitutivno enačbo (3.22) zapišemo kot:

$$\boldsymbol{\sigma}_e(x) = \sigma_x = \mathbf{D}_e \cdot \boldsymbol{\varepsilon}_e(x) = [E] \cdot \left\{ \frac{d_2 - d_1}{l} \right\} = \frac{E}{l} (d_2 - d_1) = \text{konst.} \quad (3.65)$$

Specifične deformacije in posledično napetosti dobimo tako, da z diferencialnim operatorjem odvajamo linearno funkcijo spremembe pomikov, zato so za to vrsto končnega elementa deformacije in napetosti konstantne po celotnem končnem elementu.

Togostno matriko paličnega končnega elementa z linearno interpolacijo lahko nato določimo po enačbi (3.27) ob upoštevanju, da je volumen končnega elementa enak  $V_e = l \cdot A$ , pri čemer je zaradi konstantnega preseka palice  $A$  po dolžini končnega elementa mogoče volumski integral pretvoriti v linijskega kot:

$$\mathbf{K}_e = \int_{V_e} \mathbf{B}_e^T \cdot \mathbf{D}_e \cdot \mathbf{B}_e \cdot dV = A \cdot \int_0^l \begin{bmatrix} -\frac{1}{l} \\ \frac{1}{l} \end{bmatrix} \cdot [E] \cdot \begin{bmatrix} -\frac{1}{l} & +\frac{1}{l} \end{bmatrix} \cdot dx = \frac{E \cdot A}{l} \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \quad (3.66)$$

Končno matrično ravnotežno enačbo končnega elementa (3.29) lahko nato zapišemo kot:

$$\mathbf{K}_e \cdot \mathbf{d}_e = \mathbf{f}_e \quad (3.67)$$

$$\frac{E \cdot A}{l} \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \cdot \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix}$$

Če zgornji zapis uporabimo za rešitev preprostega problema palice, ki je na eni strani (vozlišče 1) fiksno vpeta  $d_1 = 0$ , na drugem koncu (vozlišče 2) pa jo obremenimo s silo  $f_2 = F$ , dobimo:

$$\frac{E \cdot A}{l} \begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \cdot \begin{Bmatrix} 0 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} R \\ F \end{Bmatrix} \Rightarrow d_2 = \frac{l \cdot F}{E \cdot A} \text{ in } R = -F \quad (3.68)$$

ter nato še:  $\varepsilon_x = \frac{F}{E \cdot A}$  in  $\sigma_x = \frac{F}{A}$

kar natanko sovпада z analitično rešitvijo, saj je bila tako linearna variacija pomika po dolžini paličnega končnega elementa (enačba (3.62)) kot tudi konstantna vrednost specifične deformacije (enačba (3.64)) in napetosti (enačba (3.65)) v numerični rešitvi aproksimirana s povsem skladnimi interpolacijskimi funkcijami, ki ustrezajo tudi poteku analitične oziroma realne rešitve natezno ali tlačno obremenjene palice.

Vendar je z linearnimi končnimi elementi pogosto težko simulirati realne razmere. Najpreprosteje lahko to prikažemo na problemu upogibne obremenitve konzole konstantnega prereza po dolžini na sliki 3.20. Analitična rešitev za potek povesov  $u_x$  (prečnih pomikov) po dolžini konzole je podana v obliki:

$$u_x = \frac{Fl^3}{3EI} \cdot \left( 1 - \frac{3x}{2l} + \frac{1}{2} \frac{x^3}{l^3} \right) \quad (3.69)$$

in za največji poves na koncu konzole pri  $x = l$  kot:

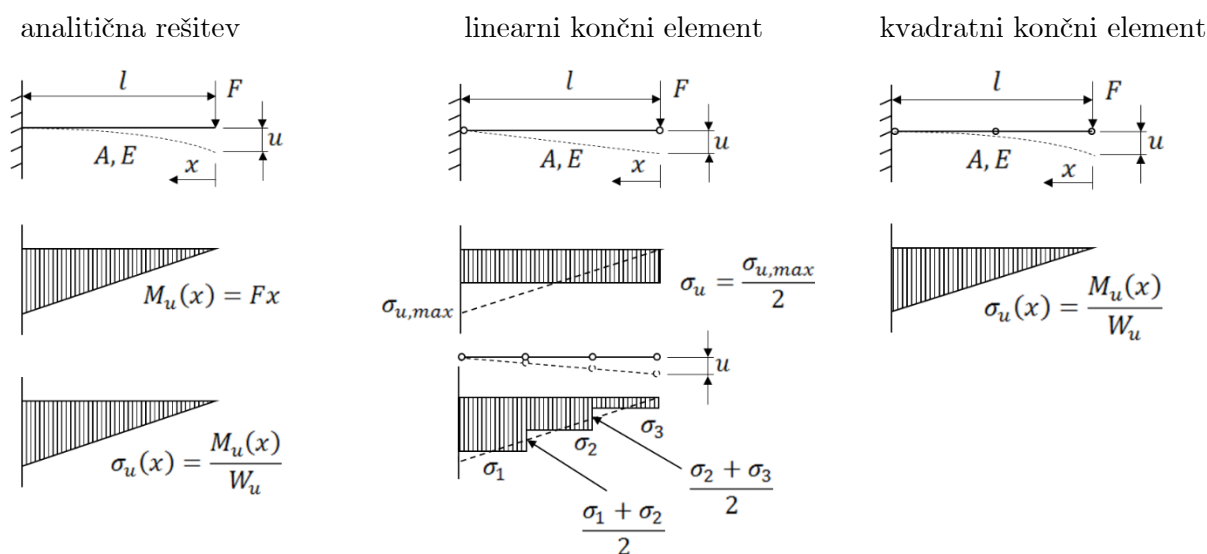
$$u = \frac{Fl^3}{3EI} \quad (3.70)$$

kjer je  $E$  modul elastičnosti in  $I$  vztrajnostni moment prereza konzole. Enačba variacije povesa konzolnega nosilca po dolžini je torej opisana s polinomom tretje, kubične stopnje. Variacija upogibne napetosti pa je opisana z linearno odvisnostjo kot:

$$\sigma_u = \frac{F \cdot x}{W} \quad (3.71)$$

kjer je  $W$  odpornostni upogibni moment prereza konzole.

V primeru diskretizacije upogibno obremenjenega nosilca z linijskimi končnimi elementi z linearno interpolacijsko funkcijo bi bila numerična rešitev sposobna opisati le linearno spremembo pomikov in zgolj konstantno, povprečno vrednost upogibne napetosti po dolžini končnega elementa (slika 3.20). Boljšo približno rešitev bi tako dobili z uporabo večjega števila linearnih končnih elementov po dolžini konzole, na osnovi katerih bi dobili natančnejše rešitve povesov konzole in napetosti v vozliščih mreže končnih elementov. Slednje moramo v vozliščih povprečiti, saj so vrednosti v sosednjih končnih elementih različne (po posameznem končnem elementu) povprečene. Z večanjem števila linearnih končnih elementov povečujemo natančnost rešitve ( $h$ -adaptacija).



Slika 3.20 – Primerjava rešitev upogibno obremenjene konzole s končnimi elementi z linearno in kvadratno interpolacijsko funkcijo

Najboljši približek natančni rešitvi dobimo z uporabo linijskega končnega elementa s kvadratno interpolacijsko funkcijo, ki nam dovolj dobro opiše kubično variacijo povesev in povsem natančno linearno variacijo upogibne napetosti po dolžini končnega elementa.

Vendar v splošnem z metodo končnih elementov numerično rešujemo probleme kompleksnih geometrijskih oblik, ki so izpostavljene kombiniranemu kolektivu statičnih ali dinamičnih obremenitev ob istočasnem upoštevanju geometrijskih, materialnih in strukturnih nelinearnosti. V tem primeru vedno uravnotežimo število končnih elementov ter potrebne razsežnosti in stopnje interpolacijske funkcije, s katerimi prostorsko diskreditiramo obravnavani problem, z razpoložljivimi računalniškimi zmogljivostmi in zahtevano natančnostjo rešitve. V splošnem velja, da uporabljamo končne elemente z linearnimi interpolacijskimi funkcijami za analize problemov, ko nas primarno zanimajo deformacije (pomiki) obravnavanega problema ali ko imamo na voljo zgolj omejene računalniške zmogljivosti za izvršitev računalniške simulacije. Končne elemente s kvadratnimi oziroma višjerednimi interpolacijskimi funkcijami pa uporabljamo, kadar nas posebej zanimajo poteki oziroma koncentracije izvedenih veličin, tj. v primeru trdnih teles specifične deformacije in napetosti.

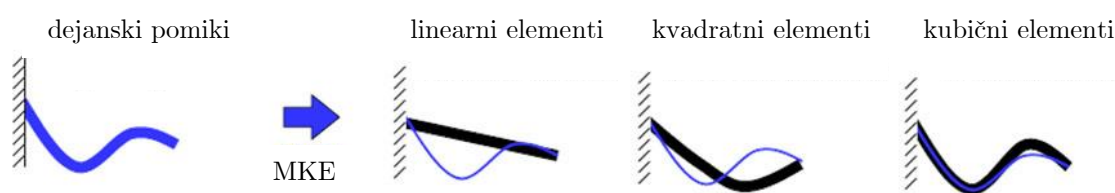
### 3.11 Napake pri delu z metodo končnih elementov

Pri metodi končnih elementov lahko naletimo na dve osnovni vrsti napak, in sicer sistemske napake in napake uporabnika. Sistemske napake so napake, ki so vgrajene v metodo končnih elementov ali so posledica delovanja računalnikov.

Glavna sistemska napaka vseh računalniških numeričnih metod je končna aritmetika. Decimalna števila so običajno shranjena v obliki, ki jo imenujemo enojna natančnost (angl. *single precision*). To pomeni, da so takšna decimalna števila shranjena v dvojiški obliki mantise in eksponenta in obsegajo 32 bitov. V tej obliki je možno število napisati z največ sedmimi desetiški decimalnimi števili. Med analizo po metodi končnih elementov števila večkrat zaporedoma množimo in delimo in pogosto postanejo pomembne neobstoječe številke za sedmim decimalnim mestom. Zaradi tega je treba upoštevati tudi napake zaradi zaokroževanja (angl. *roundoff error*). Če je ta napaka prevelika, se lahko odločimo za števila zapisana z dvojno natančnostjo (angl. *double precision*) in število obsega 64 bitov, ko je natančnih 16 desetiških decimalnih mest. V ekstremnih primerih analiz je mogoče to natančnost še povečati z uporabo 80- ali 128-bitnih zapisov števil. Večji zapisi števil običajno pomenijo daljše čase računanja, vendar danes prehod med 32 biti in 64 biti ni tako problematičen, saj je računalniška arhitektura že v osnovi 64-bitna.

Sistemska napaka se lahko pojavi pri pretvorbi dejanske geometrije problema v numerični model. Čeprav lahko geometrijo na računalniku opišemo izredno natančno, se lahko pojavijo razlike med CAD-geometrijo izdelka in mrežo končnih elementov. Končni elementi imajo namreč načeloma preproste geometrijske oblike, zato jih včasih ne moremo natančno prilagoditi geometriji izdelka. Odstopanja so pogosta pri različnih zaokrožitvah, kjer stranice elementov niso popolnoma enake radijem zaokrožitvev. Še večja odstopanja nastopajo pri kompleksnih oblikah površin sodobnih izdelkov, ki jih nikakor ne moremo popolnoma natančno opisati s končnimi elementi.

Druga napaka se pojavi zaradi uporabljene predpostavljene variacije rezultatov simulacij (pomikov, specifičnih deformacij, napetosti ipd.) po končnih elementih. Dejanski pomiki v realnih strukturah so lahko načeloma poljubnih variacij in včasih zelo kompleksnih oblik, medtem ko končni elementi vnaprej predvidevajo funkcijsko variacijo pomikov z uporabljeno stopnjo interpolacijskega polinoma. Linearni končni elementi tako simulirajo zgolj linearne spremembe pomikov, zato bodo rezultati pomikov vedno linearizirani, čeprav so dejanski pomiki lahko tudi višjega reda. Tudi če izdelek diskretiziramo z večjim številom elementov in bo odziv veliko bolj primerljiv z dejanskimi pomiki, bodo pomiki znotraj posameznega elementa še vedno linearni. Končni elementi višjega reda (kvadratni, kubični) simulirajo pomike višjega reda, zato so rezultati, dobljeni s takšnimi elementi, veliko bolj natančni, kot je prikazano na sliki 3.20 in poenostavljeno na sliki 3.21 .



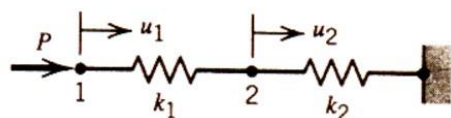
Slika 3.21 – Napake zaradi izbrane stopnje interpolacije končnih elementov

Poleg tega je v metodi končnih elementov običajno uporabljena Gaussova integracija [3.10] [3.11], kjer je integral določen z izborom specifične točke, v kateri lahko natančno določimo integral. Integriranje je izvedeno v t. i. Gaussovih točkah in število teh točk določa natančnost integriranja. Če imamo  $n$  točk, metoda zagotavlja natančnost integrala funkcije stopnje  $2n - 1$ . Zaradi hitrejšega izvajanja analize se pogosto integrira z uporabo manjšega števila točk in zaradi tega se pojavljajo manjša odstopanja rezultatov. Za določene končne elemente pa se je izkazalo, da reducirana integracija daje stabilnejše rezultate kot polna integracija.

Uporabniške napake so pogosto posledica neusklajenih enot. Večina programov, ki temeljijo na metodi končnih elementov, namreč nima vnaprej predpisanih enot in uporabnik mora sam skrbeti za usklajenost med enotami dolžine in časa, materialnimi podatki ter enoto obremenitev. Pogosto lahko uporabniki naredijo tudi napako pri določanju robnih pogojev, kjer se zaradi tega lahko pojavi mehanizem, ko se model začne premikati oz. vrteti, čeprav bi moral biti statičen. Napačna postavitve ali velikost obremenitve lahko vpliva na lokalne ali globalne obremenitvene razmere na modelu.

Nekatere napake so nekje vmes, med sistemskimi in uporabniškimi napakami. Tako npr. niso vsi končni elementi primerni za vsako analizo in če uporabnik izbere napačen tip elementa, bodo tudi rezultati napačni. Včasih tudi samodejna generacija mreže končnih elementov povzroči popačenost končnih elementov, ki dajejo slabše rezultate. Takšne napake pogosto sistemi za analizo po metodi končnih elementov ugotovijo in sporočijo uporabniku v obliki opozorila ali napake. Določenih napak pa ni mogoče tako hitro odkriti. Napaka, prikazana na sliki 3.22, se pojavi, ko imata sosednja končna elementa veliko razliko v togosti. V tem primeru

se pojavi praktično deljenje z ničlo in za rezultat dobimo ekstremne vrednosti veličin na takšnih mestih modela.

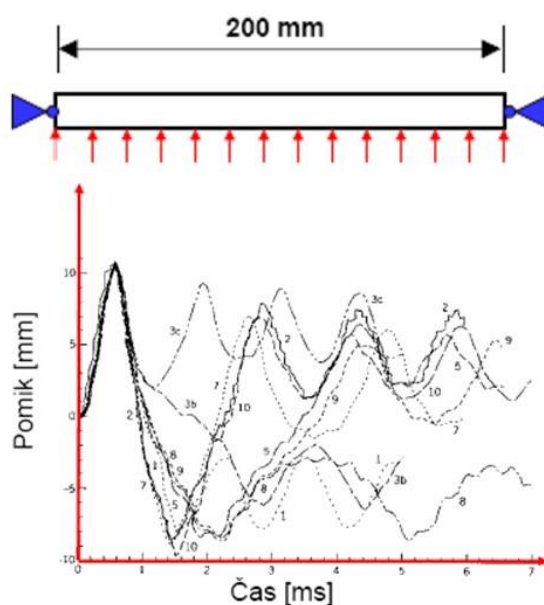


$$k_1 \gg k_2, \quad k_2 \approx 0$$

$$[(k_1 + k_2) - k_2]u_2 = P \Rightarrow u_2 = \frac{P}{k_2} \approx \frac{P}{0}$$

Slika 3.22 – Napaka zaradi razlike v togosti

V vsakem primeru je uporabnik odgovoren za natančnost numeričnih rezultatov. Računalniški programi zagotavljajo pravilnost rezultatov za testne primere (angl. *benchmark test*), za numerične analize poljubnih problemov pa proizvajalci programske opreme ne prevzemajo odgovornosti. Zato mora vsak uporabnik sam preveriti, ali so rezultati ustrezni ali ne. To je mogoče preveriti na različne načine, pri čemer je vedno najbolj zanesljiv dejanski preizkus na izdelanem fizičnem prototipu.



Slika 3.23 – Rezultati dinamičnih analiz istega primera, vendar različnih uporabnikov [3.12]

Za enostavne statične analize je možna tudi primerjava z analitičnimi preračuni ali podobnimi numeričnimi analizami. V primeru dinamičnih analiz so takšne primerjave redko mogoče, kar kaže tudi raziskava [3.12], katere rezultat je prikazan na sliki 3.23. V tej raziskavi so različni

uporabniki analizirali nosilec s členkastimi podporami s pulzno obremenitvijo 1 ms in so morali določiti pomik na sredini nosilca v odvisnosti od časa. Analizo je izvajalo deset različnih izkušenih uporabnikov z različnimi programskimi paketi. Rezultati se dobro ujemajo samo v obremenitveni fazi (1 ms), potem pa se rezultati uporabnikov bistveno razlikujejo.

### 3.12 Literatura

- [3.1] Finite element method [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Finite\\_element\\_method](https://en.wikipedia.org/wiki/Finite_element_method). [Datum dostopa 5. 12. 2017].
- [3.2] M. J. Turner, R. W. Clough, H. C. Martin, L. J. Topp, "Stiffness and Deflection Analysis of Complex Structures," *Journal of the Aeronautical Sciences*, let. 23, št. 9, str. 805–823, 1956.
- [3.3] R. W. Clough, "The Finite Element Method in Plane Stress Analysis," v 2nd ASCE Conference on Electronic Computation. Pittsburgh, PA, 1960.
- [3.4] O. C. Zienkiewicz, Y. K. Cheung, "Finite Element in the Solution of Field Problems," *The Engineer*, let. 220, str. 507-510, 1965.
- [3.5] B. M. Irons, O. C. Zienkiewicz, "The Isoparametric Finite Element System - A New Concept in Finite Element Analysis," v Conf. Recent Advances in Stress Analysis. Royal Aeronautical Society, 1968.
- [3.6] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 6. izdaja. Burlington: Elsevier Butterworth-Heinemann, 2005.
- [3.7] P-FEM [splet], Dosegljivo: <https://en.wikipedia.org/wiki/P-FEM>. [Datum dostopa 5. 12. 2017].
- [3.8] Keeping Track of Element Order in Multiphysics Models [splet], Dosegljivo: <https://www.comsol.com/blogs/keeping-track-of-element-order-in-multiphysics-models/>. [Datum dostopa 5. 12. 2017].
- [3.9] The 8-Node Hexahedron [splet], Dosegljivo: <http://www.colorado.edu/engineering/CAS/courses.d/AFEM.d/AFEM.Ch11.d/AFEM.Ch11.pdf>. [Datum dostopa 5. 12. 2017].
- [3.10] Gaussian quadrature [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Gaussian\\_quadrature](https://en.wikipedia.org/wiki/Gaussian_quadrature). [Datum dostopa 5. 12. 2017].
- [3.11] Gaussian Quadrature [splet], Dosegljivo: <http://mathworld.wolfram.com/GaussianQuadrature.html>. [Datum dostopa 5. 12. 2017].
- [3.12] R. D. Cook, *Finite Element Modeling for Stress Analysis*. New York: John Wiley & Sons, 1995.



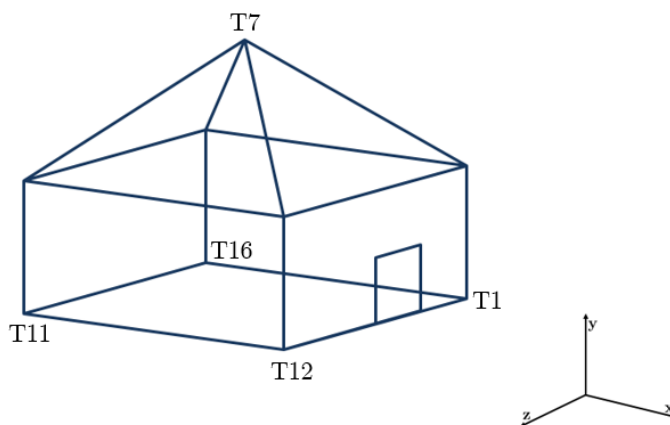
## 4. Geometrijsko modeliranje

Priprava geometrije modela je običajno prvi korak v računalniških simulacijah. Pri tem ni pomembno, za kakšno vrsto simulacije gre, saj lahko simuliramo razmere v trdinah, tekočinah ali tudi procese. Modeliranje geometrije je danes običajno povezano z računalniško grafiko, včasih pa je to pomenilo določitev numeričnih podatkov na osnovi koordinatah točk modela. Zahtevnost modeliranja geometrije je odvisna od razsežnosti modela. Linijski modeli v eni dimenziji so zelo preprosti, ravninski modeli v dveh dimenzijah so dokaj enostavni, prostorski modeli v treh dimenzijah pa so lahko precej kompleksni. Razvoj računalniške opreme danes omogoča razmeroma preprosto modeliranje tudi kompleksnih prostorskih modelov. Kljub temu pa se lahko lotimo modeliranja geometrije na različne načine v odvisnosti od podatkov modela in analize, s katero bomo izvedli simulacijo.

### 4.1 Linijski modeli

S stališča računalniške tehnologije so linijski modeli (angl. *wire-frame model*) [4.1] najpreprostejši, saj je celoten model sestavljen iz povezanih linij, ki so na računalniškem zaslonu prikazane s črtami. Linije ali črte so lahko določene neposredno ali posredno s koordinatami točk.

Točka	x	y	z
T1	1	0	0
T2	1	1	0
T3	0	1	0
T4	0.5	2	0.5
T5	1	1	1
T6	1	1	0
T7	0.5	2	0.5
T8	0	1	1
T9	0	1	0
T10	0	0	0
T11	0	0	1
T12	0	1	1
T13	1	1	1
T14	1	0	1
T15	1	0	0
T16	0	0	0
T17	0	0	1
T18	1	0	1
T19	1	0	0.25
T20	1	0.5	0.25
T21	1	0.5	0.5
T22	1	0	0.5

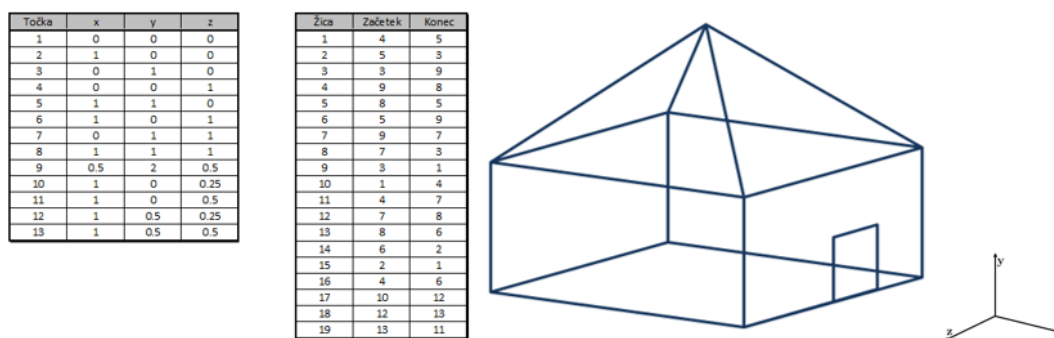


Slika 4.1 – Neposredna predstavitev linijskega modela

Na sliki 4.1 je prikazan linijski model, ki je določen na podlagi zaporedja koordinat. Risanje linij poteka zaporedno in zato je včasih treba ponoviti risanje posamezne linije, da lahko nadaljujemo gradnjo modela.

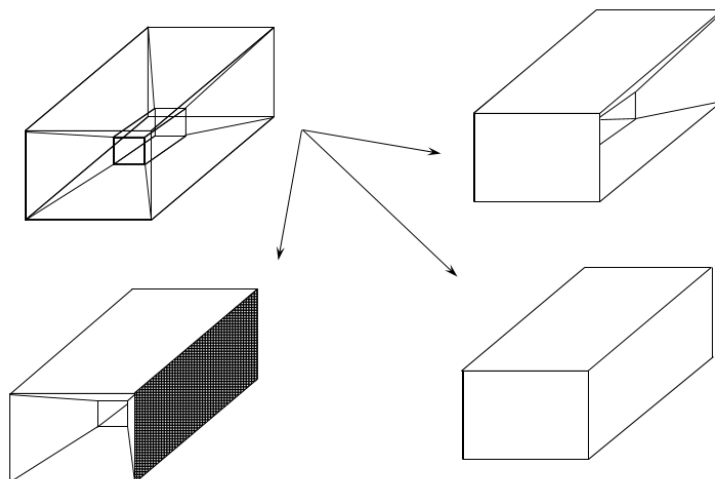
Običajno pa je izdelan seznam točk v prostoru, kjer ima vsaka točka, ki je določena s prostorskimi koordinatami, enolično oznako (običajno je to kar zaporedna številka). Linije pa so potem določene posredno, kot povezava med dvema oštevilčenima točkama. Takšen posreden

način modeliranja, prikazan na sliki 4.2, omogoča modeliranje vsake linije neodvisno od vrstnega reda linij v modelu.



Slika 4.2 – Posredna predstavitev linijskega modela

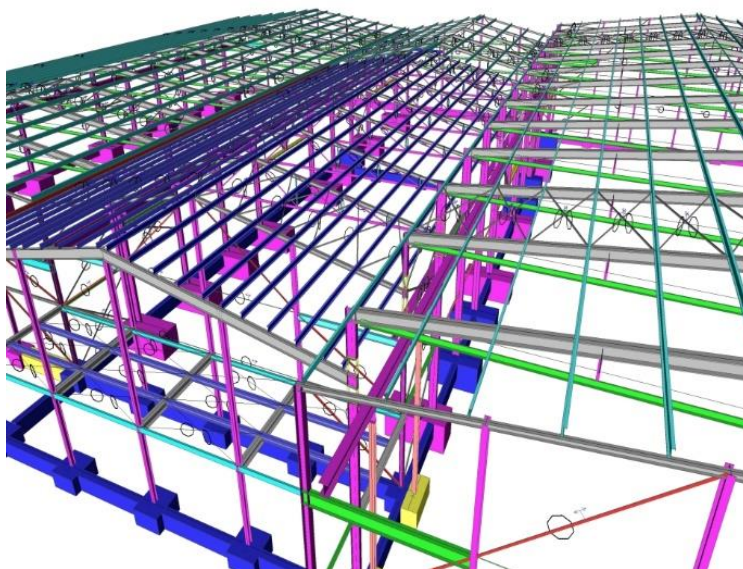
Modeliranje kompleksnih linijskih modelov je lahko zelo zahtevno in delovno intenzivno. Zaradi tega je linijsko modeliranje danes aktualno le za modeliranje in simuliranje paličnih konstrukcij. Za druge vrste modelov postanejo linijski modeli hitro nepregledni in dvoumni, saj ni nobene informacije o ploskvah med robovi, ki ga določajo linije modela. Na sliki 4.3 je prikazana dvoumnost predstavitve z linijskim modelom.



Slika 4.3 – Dvoumnost linijskega modela

Kljub temu je z drugimi načini modeliranja, ki bodo predstavljeni v nadaljevanju, težko modelirati palične konstrukcije, ki so v strojearhitekturi precej pogoste. Tako pri modeliranju kot pri simulacijah zato z linijskimi elementi modeliramo srednjo linijo palic, ki ji določimo ustrezen presek elementa, s katerim gradimo model. Model sestavljamo na podoben način, kot je prikazano pri zgornjem primeru linijskega modeliranja, le da imamo na voljo uporabniku

prijaznejše ukaze za pripravo, kopiranje, premikanje in popravljanje palic. V paličnih konstrukcijah lahko pogosto uporabimo že naprej sestavljene elemente, ki jih tvorijo med seboj povezane palice različnih prerezov. Primer takšnega modelirnika, ki je namenjen modeliranju in simulacijam paličnih konstrukcij, je SCIA Engineer [4.2]. Na sliki 4.4 je prikazana slika modela iz programa SCIA Engineer, kjer vidimo, da so prerezi palic prikazani realistično, čeprav uporabnik modelira le linijski model.

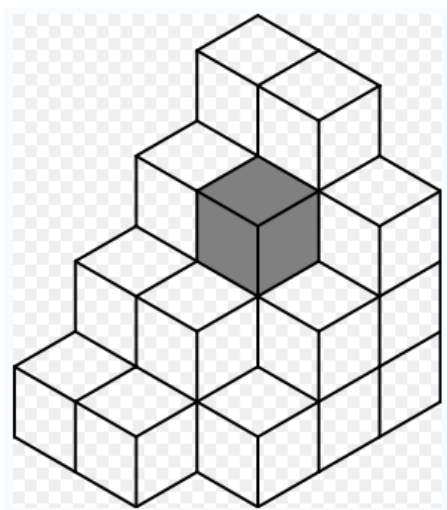


Slika 4.4 – Žični model v modelirniku SCIA Engineer [4.3]

Podobne module vključujejo tudi nekateri CAD-sistemi, pri katerih je že na začetku treba izbrati način modeliranja, s katerim gradimo linijski prostorski model. Tudi programi za simulacije omogočajo podobno predstavitev, kot je prikazana na sliki 4.4.

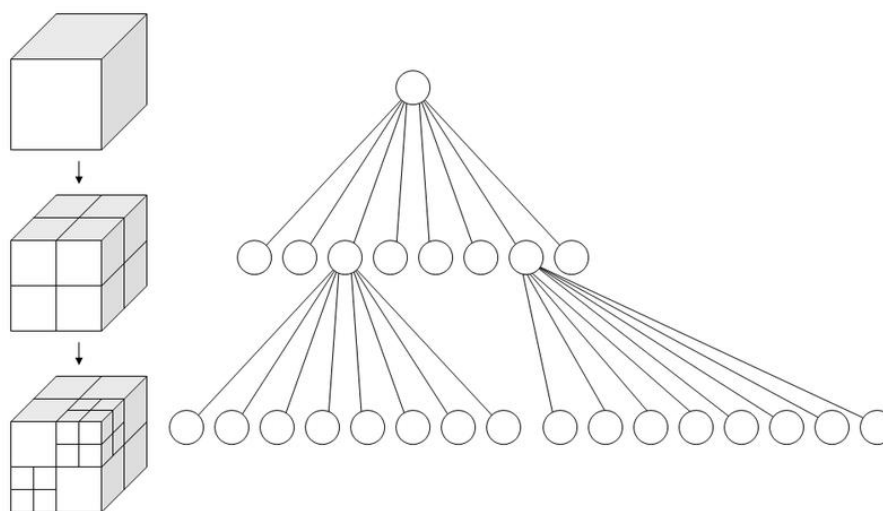
## 4.2 Volumetrični modeli

Ideja volumetričnega modeliranja skuša posneti realen svet, ki ga sestavljajo gradniki, kot so molekule in atomi. Ti gradniki so seveda mnogo premajhni za učinkovito uporabo v sedanjih računalnikih, zato je osnoven element običajno kocka makroskopskih dimenzij. Takšen osnoven element imenujemo voksel (angl. *voxel* – *volume element*) [4.4]. Ime izhaja iz termina slikovna točka (angl. *pixel* – *picture element*), ki se uporablja za prikazovanje slike na računalniških zaslonih. Velikost voksla določa natančnost predstavitve volumetričnega modela.



Slika 4.5 – Volumetrični model, sestavljen iz vokslov [4.4]

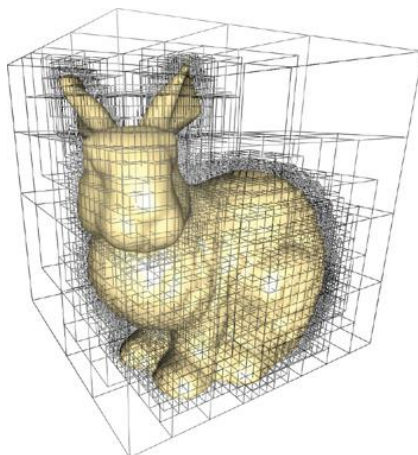
Osnovni problem volumetričnih modelirnikov je pomnilnik, ki je potreben za predstavitev kompleksnega modela. Za predstavitev modela z ločljivostjo  $1024 \times 1024 \times 1024$  v naravnih barvah potrebujemo kar 4 GB pomnilnika. To je še danes velika zahteva za pomnilnik, pred leti, ko so nastajali prvi volumetrični modelirniki, pa je bila nemogoča uporaba takšne količine pomnilnika. Zato so našli rešitev v obliki predstavitve modela kot osmiškega drevesa (angl. *octree*) [4.5], ki je prikazan na sliki 4.6.



Slika 4.6 – Osmiško drevo [4.5]

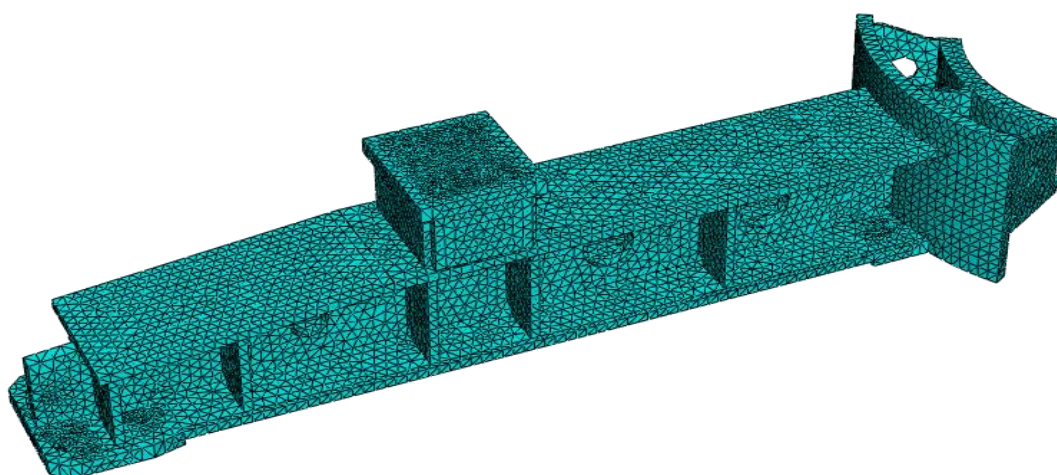
Celoten volumen razdelimo na osem kvadrantov. Vsak kvadrant lahko vsebuje material ali prazen prostor. Če je celoten kvadrant poln ali prazen, ga ni več treba razdeliti. V nasprotnem primeru pa kvadrant nadalje razdelimo na osem kvadrantov in postopek nadaljujemo, dokler ni prostor razdeljen tako, da je v vsakem prostoru material ali prazen prostor. Pri tej predstavitvi so voksli različnih velikosti, vendar so še vedno vsi voksli kocke. Seveda pa za

model ni več potrebno toliko pomnilnika kot pri osnovnem volumetričnem modelu. Primer takšnega modela je prikazan na sliki 4.7.



Slika 4.7 – Primer modela z uporabo osmiških dreves [4.6]

Model je lahko sestavljen tudi iz elementov, ki so različnih velikosti in oblik. Primer takšnih modelov so vsi prostorski modeli, zgrajeni za simulacije po metodi končnih elementov. Programi za izdelavo mrež končnih elementov lahko samodejno zgradijo mrežo poljubnega prostorskega modela, tako da ves material zapolnijo s končnimi elementi. Zelo pogosto so takšni elementi tetraedrične oblike, ker jih je enostavneje uporabiti pri samodejnem mreženju. V metodi končnih elementov je na splošno zaželena heksaedrična oblika končnih elementov, ki zahteva precej bolj kompleksen postopek za pripravo mreže. Na sliki 4.8 je prikazana prostorska mreža tetraedričnih končnih elementov.



Slika 4.8 – Mreža tetraedričnih končnih elementov

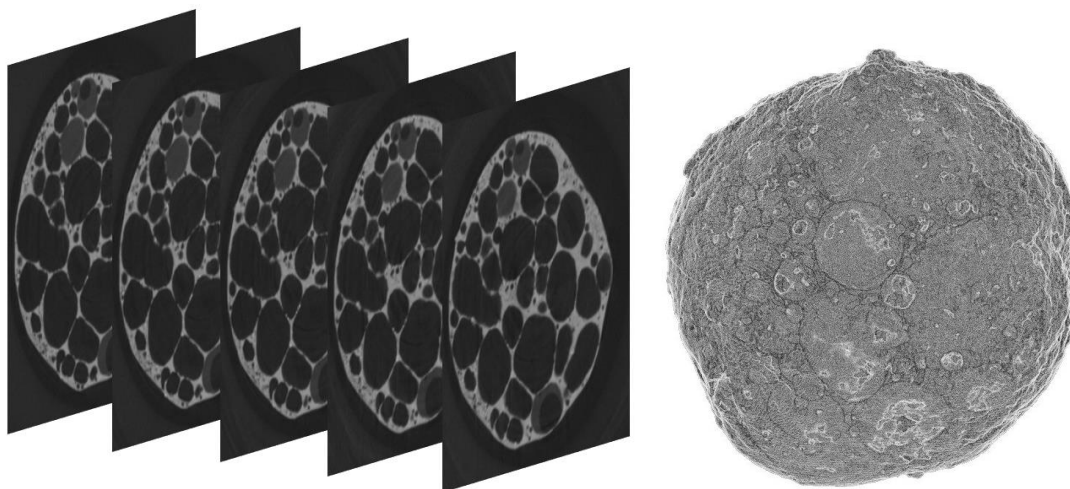
Ob težavah s pomnilnikom je volumetrične modele tudi razmeroma težko prikazovati na zaslonu. Volumetričen model je seveda mogoče najenostavneje prikazati na 3D-volumetričnem prikazovalniku. Ker takšne naprave praktično še niso dosegljive, je volumetričen model treba pretvoriti v slikovne točke na ekranu. Ker je vokslov veliko, je tudi veliko poligonov, ki jih je treba spremeniti v slikovne točke na 2D-zaslonu. To lahko naredimo z različnimi oblikami metode sledenja žarku (angl. *raytracing*) [4.7] ali s pretvorbo volumetričnega modela v poligonski model in prikazovanje poligonov na ekranu. Obe tehniki zahtevata precej časa, zato si različni volumetrični modelirniki zelo počasi utirajo pot do uporabnikov.

Uporaba volumetričnih modelirnikov je v veliki meri v domeni računalniških iger [4.4], od katerih je danes ena priljubljenejših igra Minecraft [4.8], v kateri lahko igralci modelirajo poljubne modele, kot je prikazano na sliki 4.9.



Slika 4.9 – Izdelava preprostega avtomobila v igri Minecraft [4.9]

Volumetrično modeliranje se uporablja tudi v številnih aplikacijah za obdelavo podatkov slik računalniške tomografije – CT (angl. *Computed Tomography*) [4.10], in sicer najprej v medicini in zadnje čase tudi v inženirstvu za skeniranje strojev, konstrukcij in materialov [4.11]. V teh primerih se pri skeniranju uporabljajo X-žarki, ki so v ozkem snopu usmerjeni skozi opazovani objekt in na detektorju žarkov tvorijo sliko ene tanke rezine objekta. Nato se skener premakne za določen korak v navpični smeri ( $z$ ) in posname novo rezino. Rezultat skeniranja je veliko število slik, na katerih so prikazane posamezne rezine. Na levi strani slike 4.10 je prikazanih samo nekaj rezin materiala, ki je bil skeniran z mikro CT-skenerjem. Z ustrežno programsko opremo (npr. [4.12]) lahko te slike analiziramo. Na desni strani slike 4.10 je prikazan volumetrični model por, rekonstruiran na osnovi skeniranih slik materiala. Tako izdelan volumetrični model je možno opazovati v različnih pogledih in prerezih, lahko pa ga izvozimo v obliki površinskega modela v poljuben sodoben prostorski modelirnik.

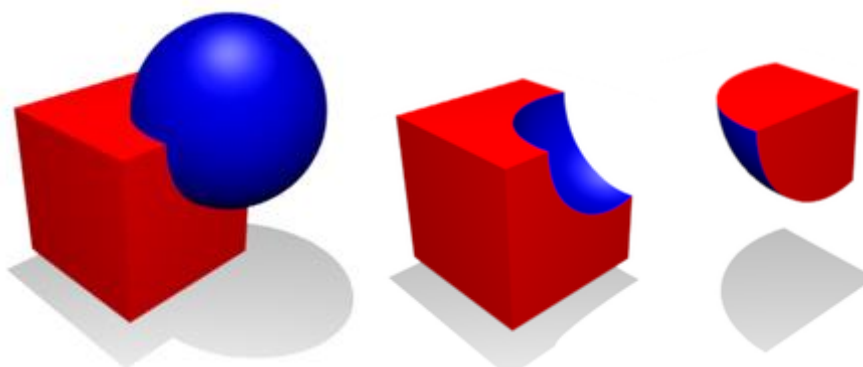


Slika 4.10 – Rezine, pridobljene z računalniško tomografijo, in volumetrični model materiala

### 4.3 Prostorski modeli

Delo z volumetričnim modelirnikom si lahko predstavljamo kot zlaganje kock Lego. Poleg vokslov v obliki kocke bi potrebovali še drugačne oblike, da bi bolj učinkovito zgradili kompleksen model. Zato so pri prvih modelirnikih uporabljali parametrizirane osnovne gradnike, kot so: kvader, valj, krogla, torus, konus itd. Osnovne gradnike je uporabnik nato zgolj postavljaj na, ob ali pod druge gradnike in tako gradil model. Ta tehnika se je imenovala sestavljanje primitivnih gradnikov (angl. *pure primitive instancing*) in ni bila najbolj uporabna, saj ni omogočala kombinacije gradnikov.

Zato med prve prave modelirnike štejemo modelirnike CSG (Constructive Solid Geometry) [4.13]. Z modelirnikom CSG lahko preprosta geometrijska telesa kombiniramo na osnovi Boolove algebre, prikazane na sliki 4.11.



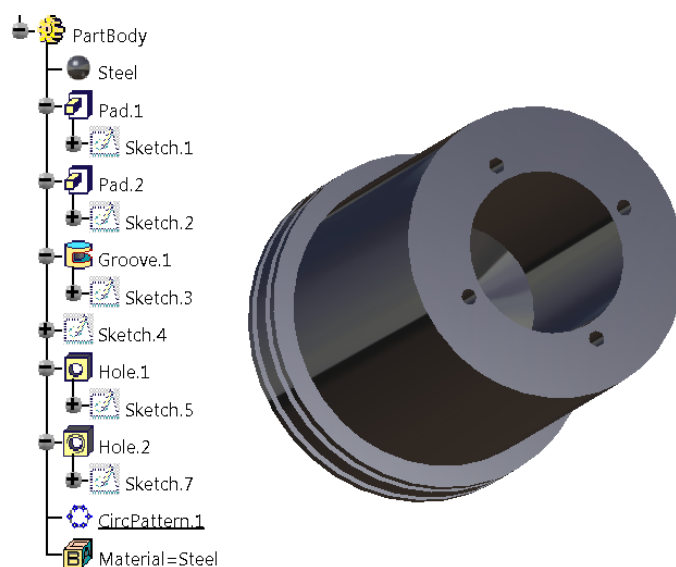
Slika 4.11 – Unija, razlika in presek kocke in krogle v modelirniku CSG [4.13]

Tak način modeliranja je zelo kreativen, saj lahko s kombiniranjem preprostih geometrijskih gradnikov izdelamo kompleksne modele, do katerih je mogoče priti na različne načine. Poleg

osnovnih gradnikov, ki jim lahko poljubno spreminjamo mere, imamo danes na voljo še gradnike poljubnih oblik, ki jih uporabnik izdelava sam. Zato danes večina prostorskih modelirnikov vključuje tudi modelirnik CSG, kjer so Boolove operacije skrite za bolj prijaznimi imeni ukazov, kot sta spajanje, rezanje itd.

Rezultat Boolove operacije v modelirniku CSG mora biti pravilno prostorsko telo. Ker je včasih mogoče, da operacija ni uspešna, sodobni programi onemogočijo nadaljevanje dela, dokler napaka ni odpravljena. To zagotavlja, da je izdelan model vedno konsistenten in ga je mogoče v nadaljevanju uporabiti za izdelavo mreže končnih elementov.

Operacije modeliranja z Boolovo algebro je mogoče simbolno opisati tudi z drevesno strukturo, zato modelirniki danes omogočajo prikazovanje drevesne strukture v vsakem trenutku, kot je prikazano na sliki 4.12 v programu Catia.



Slika 4.12 – Drevesna struktura CSG

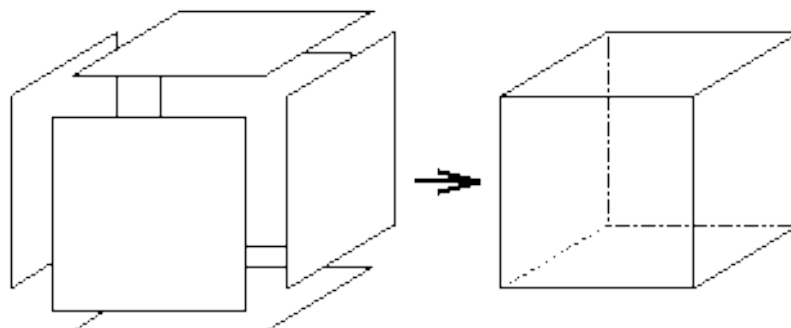
V današnjih modelirnikih so na voljo uporabniško izdelani gradniki in veliko število operacij z njimi. Operacije in gradniki v drevesni strukturi se danes imenujejo značilke (angl. *features*) [4.14]. Zato se tudi modeliranje CSG v današnjih modelirnikih imenuje modeliranje z značilkami (angl. *feature based design*).

Glavna težava čistega modelirnika CSG je prikazovanje modela po izvedenih operacijah. Podobno kot pri volumetričnem modeliranju imamo tudi tukaj na voljo dva načina. Model lahko prikažemo z metodo sledenja žarku ali pretvorbo v površinski model. Do nedavnega je bilo upodabljanje (angl. *rendering*) z metodo sledenja žarku dokaj zamuden proces, danes pa sodobne naprave za računalniško grafiko omogočajo sledenje žarku brez časovne zakasnitve. Zato je teoretično mogoče izdelati čisti modelirnik CSG.

V času, ko se je začel razvoj 3D-modelirnikov, pa je bilo nemogoče v realnem času upodobiti model z metodo sledenja žarku, zato danes vsi modelirniki temeljijo na zapisu modela s površinami (angl. *Boundary Representation*), ki ga označujem s kratico B-rep [4.15]. Na sliki

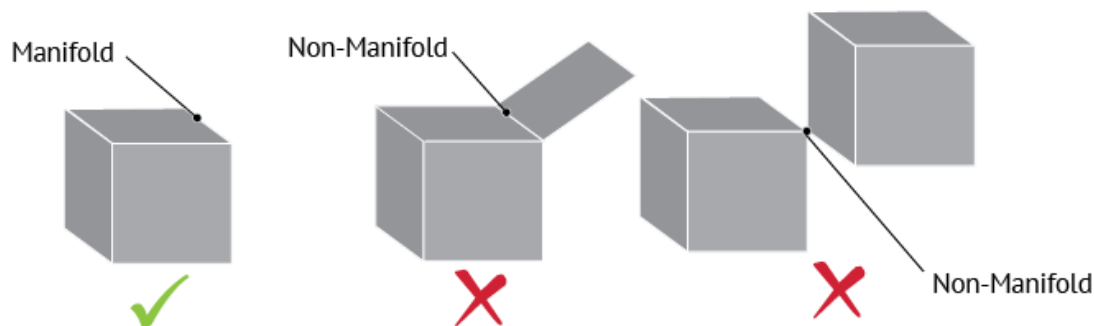


4.13 je prikazano, kako so prostorski objekti predstavljeni s površinami. Površine so lahko poljubnih oblik in ni nujno, da so samo ravninske ploskve, kot v primeru kocke na sliki 4.13. Pri sodobnih modelirnikih so površine določene s parametričnimi površinami NURBS [4.16].



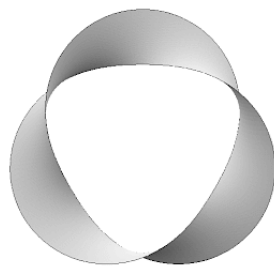
Slika 4.13 – B-rep predstavitev 4.13

Pri modelih B-rep ločimo med površinami, ki tvorijo zaprto pravilno telo (angl. *manifold*), in površinami, ki so odprte vrste in predstavljajo zgolj površinski model (angl. *non-manifold*). Dejanska telesa morajo biti modelirana tako, da so pravilna in zaprta, zato se lahko v operacijah prostorskega modelirnika pojavijo napake, ki so prikazane na sliki 4.14 .



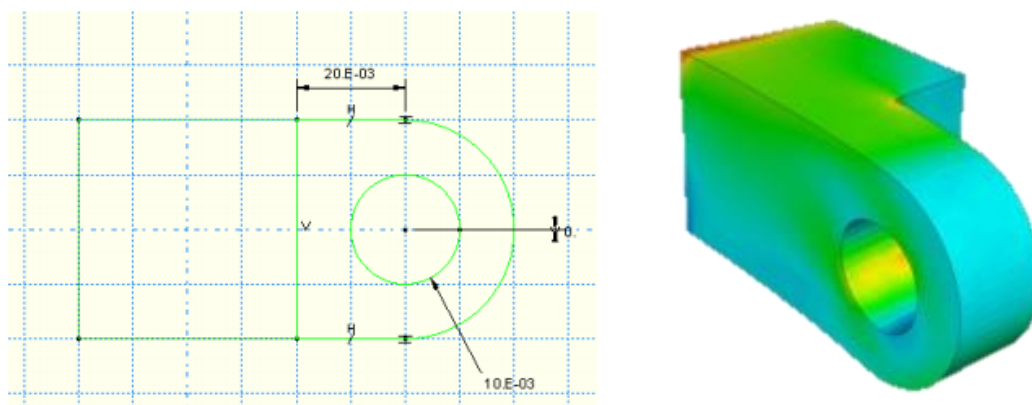
Slika 4.14 – Predstavitev B-rep [4.17]

V določenih primerih želimo modelirati zgolj površine v t. i. površinskih modelirnikih, kjer pa lahko izdelamo tako površine, ki tvorijo zaprto pravilno telo, kot tudi površine, ki so odprte vrste in predstavljajo zgolj površinski model (npr. površina, prikazana na sliki 4.15).

Slika 4.15 – Površina *non-manifold* [4.18]

Od načina in objekta modeliranja je odvisno, kaj bo mogoče z izdelanim modelom narediti. Če je izdelano telo sestavljeno iz zaprtih površin in je shranjeno kot prostorski model, je mogoče v nadaljevanju izdelati prostorske ali površinske mreže končnih elementov. Če imamo model *non-manifold*, pa je mogoče izdelati le površinske mreže končnih elementov.

Današnji zmogljivi modelirniki omogočajo modeliranje tako zaprtih prostorskih modelov kot tudi odprtih površinskih modelov. Modeliranje poteka tako, da v programu običajno skiciramo približno obliko, ki jo nato natančno določimo z geometrijskimi in dimenzijskimi omejitvami. Izdelano skico uporabimo v 3D-ukazih, s katerimi lahko skico izvlečemo ali zavrtimo v prostor. S skiciranjem na različne ravnine objekta in dodatnimi 3D- ter CSG-ukazi kombiniranja teles zgradimo želeni geometrijski model za simulacijo. Na sliki 4.16 sta prikazana skica in primer izdelanega modela v programu ABAQUS.



Slika 4.16 – Modelirnik v programu ABAQUS

Geometrijski model je lahko tudi uvožen iz poljubnega modelirnika, če so podatki zapisani v standardni obliki. Geometrijski model je lahko tudi v obliki zapisa modelirniških knjižnic Parasolid [4.19] ali ACIS [4.20]. V zadnjem času skoraj vsa programska oprema podpira standard STEP za prenos geometrijskih podatkov [4.21].

Pri uvozu geometrijskih podatkov je zelo pomembno, da so v datoteki shranjeni vsi podatki o modelu. Programi za simulacije lahko izdelajo mrežo končnih elementov le na osnovi natančnih

geometrijskih podatkov, ki so običajno zapisani v obliki parametričnih površin NURBS. V nekaterih primerih je mogoče model ponovno izdelati tudi brez teh podatkov. V tem primeru mora program za simulacije rekonstruirati parametrične površine, kar pa ni vedno mogoče na osnovi podatkov v datoteki, iz katere uvažamo geometrijo. Prav tako je pri uvozu geometrije potrebna pozornost glede skladnosti merskih enot.

#### 4.4 Literatura

- [4.1] Wire-frame model [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Wire-frame\\_model](https://en.wikipedia.org/wiki/Wire-frame_model). [Datum dostopa 5. 12. 2017].
- [4.2] SCIA Engineer [splet], Dosegljivo: <https://www.scia.net/en/software/product-selection/scia-engineer>. [Datum dostopa 5. 12. 2017].
- [4.3] SCIA Engineer - Technical specifications [splet], Dosegljivo: <https://www.scia.net/en/software/product-selection/scia-engineer/technical-specifications>. [Datum dostopa 5. 12. 2017].
- [4.4] Voxel [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Voxel>. [Datum dostopa 5. 12. 2017].
- [4.5] Octree [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Octree>. [Datum dostopa 5. 12. 2017].
- [4.6] Octree Textures on the GPU [splet], Dosegljivo: [https://developer.nvidia.com/gpugems/GPUGems2/gpugems2\\_chapter37.html](https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_chapter37.html). [Datum dostopa 5. 12. 2017].
- [4.7] Next-Gen 3D Rendering Technology: Voxel Ray Casting [splet], Dosegljivo: <http://www.tomshardware.com/reviews/voxel-ray-casting.2423.html>. [Datum dostopa 5. 12. 2017].
- [4.8] Minecraft [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Minecraft>. [Datum dostopa 5. 12. 2017].
- [4.9] How To Build Car In Minecraft [splet], Dosegljivo: <https://www.youtube.com/watch?v=ajUvvjv7yPY>. [Datum dostopa 5. 12. 2017].
- [4.10] CT scan [splet], Dosegljivo: [https://en.wikipedia.org/wiki/CT\\_scan](https://en.wikipedia.org/wiki/CT_scan). [Datum dostopa 5. 12. 2017].
- [4.11] Industrial computed tomography [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Industrial\\_computed\\_tomography](https://en.wikipedia.org/wiki/Industrial_computed_tomography). [Datum dostopa 5. 12. 2017].
- [4.12] Fiji [splet], Dosegljivo: <http://fiji.sc/Fiji>. [Datum dostopa 5. 12. 2017].
- [4.13] Constructive solid geometry [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Constructive\\_solid\\_geometry](https://en.wikipedia.org/wiki/Constructive_solid_geometry). [Datum dostopa 5. 12. 2017].
- [4.14] J. Duhovnik, I. Demšar, P. Drešar, *Modeliranje z značilkami na osnovi SolidWorks*. Ljubljana: Univerza v Ljubljani, Fakulteta za strojništvo, 2014.

- [4.15] Boundary representation [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Boundary\\_representation](https://en.wikipedia.org/wiki/Boundary_representation). [Datum dostopa 5. 12. 2017].
- [4.16] Non-uniform rational B-spline [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Non-uniform\\_rational\\_B-spline](https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline). [Datum dostopa 5. 12. 2017].
- [4.17] Repair Your File for 3D Printing [splet], Dosegljivo: <http://www.sculpteo.com/en/tools/repair-your-file-3d-printing/>. [Datum dostopa 5. 12. 2017].
- [4.18] Explore Logo, Deer and more [splet], Dosegljivo: <https://www.pinterest.com/pin/454371049878644627/>. [Datum dostopa 5. 12. 2017].
- [4.19] Parasolid [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Parasolid>. [Datum dostopa 5. 12. 2017].
- [4.20] ACIS [splet], Dosegljivo: <https://en.wikipedia.org/wiki/ACIS>. [Datum dostopa 5. 12. 2017].
- [4.21] ISO 10303 [splet], Dosegljivo: [https://en.wikipedia.org/wiki/ISO\\_10303](https://en.wikipedia.org/wiki/ISO_10303). [Datum dostopa 5. 12. 2017].

## 5. Materialne lastnosti v MKE

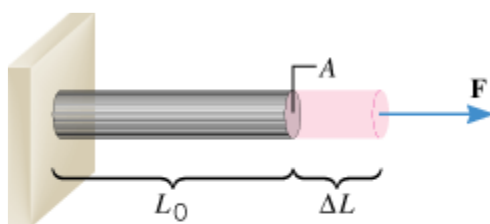
Materialni podatki so že v CAD-modelu običajno povezani z geometrijskim modelom in kadar je program za numerične analize tesno povezan s CAD-programom, se ti podatki prenesejo samodejno. Včasih pa je treba materialne podatke v programih za numerične analize ponovno določiti.

Zahtevnost in specifika določene numerične analize zahtevata materialne podatke, ki to analizo omogočajo. V primeru linearne statične analize trdin je dovolj, če za izotropne materiale podamo zgolj dva materialna parametra: modul elastičnosti  $E$  in Poissonovo razmerje  $\nu$ . V primerih različnih nelinearnih ali dinamičnih analiz ali v primeru anizotropnih materialov pa je določanje materialnih podatkov precej bolj zahtevno.

Omeniti je treba tudi, da so programi za numerične analize zelo pogosto popolnoma neodvisni od sistema merskih enot in šele kombinacija dolžinskih mer, obremenitev in materialnih podatkov določa sistem enot v dani simulaciji.

### 5.1 Modul elastičnosti in Poissonovo razmerje

Modul elastičnosti ali Youngov modul [5.1] je določen iz linearnega razmerja med napetostmi in specifičnimi deformacijami v območju elastične proporcionalnosti materiala [5.2]. Najlažje lahko modul elastičnosti izmerimo, če izvedemo enosni natezni preizkus materiala in merimo pomike tako, kot je prikazano na sliki 5.1. Seveda meritev v praksi ni tako preprosta, saj se zaradi obremenitve spreminja tudi površina preseka  $A$ .



Slika 5.1 – Enosna natezna obremenitev

V primeru poenostavitve lahko na osnovi tega preizkusa enostavno izračunamo modul elastičnosti z uporabo enačbe (5.1).

$$E = \frac{F}{\frac{\Delta L}{L_0}} = \frac{\sigma}{\epsilon} \quad (5.1)$$

V preglednici 5.1 je navedenih nekaj vrednosti modula elastičnosti. SI-enota modula elastičnosti je Pa oz.  $\text{N/m}^2$ . Če v simulacijah uporabljamo druge dolžinske enote ali enote obremenitev, je enoto modula elastičnosti treba ustrezno spremeniti. V strojništvu najpogosteje uporabljamo

mm za dolžinske enote in N za obremenitve. V takšnih primerih podajamo modul elastičnosti v MPa oz. N/mm<sup>2</sup>.

Preglednica 5.1 – Modul elastičnosti različnih materialov

Material	Modul elastičnosti [N/m <sup>2</sup> ]	Modul elastičnosti [N/mm <sup>2</sup> ]
volfram	$3,6 \cdot 10^{11}$	$3,6 \cdot 10^5$
jeklo	$2,1 \cdot 10^{11}$	$2,1 \cdot 10^5$
baker	$1,1 \cdot 10^{11}$	$1,1 \cdot 10^5$
aluminij	$6,9 \cdot 10^{10}$	$6,9 \cdot 10^4$
steklo Pyrex	$6,2 \cdot 10^{10}$	$6,2 \cdot 10^4$
najlon	$3,7 \cdot 10^9$	$3,7 \cdot 10^3$
teflon	$3,7 \cdot 10^8$	$3,7 \cdot 10^2$

Strižni modul  $G$  določimo z enačbo (5.2).

$$G = \frac{E}{2(1 + \nu)} \quad (5.2)$$

Če na telo deluje enakomerna sila z vseh strani (hidrostatično napetostno stanje), telo spremeni svoj volumen, vendar ohrani obliko. Razmerje med volumskimi napetostmi in volumskimi deformacijami je definirano kot modul stisljivosti, ki je določen kot:

$$K = \frac{\sigma}{\frac{\Delta V}{V}} = \frac{E}{3(1 - 2\nu)} \quad (5.3)$$

V skladu z drugim zakonom termodinamike morajo biti vsi trije moduli (modul elastičnosti, strižni modul in modul stisljivosti) večji od 0. Iz tega sledi, da ima lahko Poissonovo razmerje [5.3][5.4] le vrednosti, večje od  $-1$  in manjše od  $0,5$ . Poissonovo razmerje je določeno kot:

$$\nu = -\frac{\varepsilon_y}{\varepsilon_x} \quad (5.4)$$

kjer je  $\varepsilon_x$  vzdolžna specifična deformacija in  $\varepsilon_y$  prečna specifična deformacija. Obe deformaciji prav tako določimo na osnovi meritev sprememb dolžine in prečnega premera pri enoosnem nateznem preizkusu materiala kot:

$$\varepsilon_x = \frac{\Delta L}{L_0} \quad \text{in} \quad \varepsilon_y = \frac{\Delta d}{d_0} \quad (5.5)$$

Zgornja mejna vrednost  $0,5$  pomeni nestisljiv material, ki v naravi ne obstaja. Najbližje nestisljivosti so plini in tekočine, od trdnin pa gume. Pri tem je Poissonovo razmerje zelo blizu, vendar manjše od  $0,5$ . V programih za numerične analize se lahko pojavi napaka, če vnesemo Poissonovo razmerje v vrednosti  $0,5$ , saj bi v tem primeru postal modul stisljivosti neskončen (v enačbi (5.3) postane imenovalec enak 0).

Večina trdnin ima Poissonovo razmerje med 0 in 0,5, kar pomeni, da tak material pod vplivom zunanje obremenitve v večji ali manjši meri spremeni svoj volumen. Material, ki ima Poissonovo razmerje enako 0, ima prečno specifično deformacijo  $\varepsilon_y$  enako nič, kar pomeni, da ne spremeni prečne oblike pod vplivom obremenitve. Tak material je pluta, ki ga uporabljamo za zamaške, saj kljub veliki obremenitvi pri vstavljanju zamaška v steklenico še vedno ohrani prvotno obliko.

V preglednici 5.2 so navedene vrednosti Poissonovega razmerja za nekatere materiale. Ponekod je naveden razpon, vendar je za natančno simulacijo vedno treba točno določiti Poissonovo razmerje za izbran material. V večini primerov ima odstopanje od dejanske vrednosti Poissonovega razmerja majhen vpliv na natančnost rezultatov. Tudi možnosti napake uporabnika so veliko večje pri izboru modula elastičnosti, kjer lahko uporabnik hitro naredi veliko napako zaradi napačnih enot ali napačne izbire vrednosti modula elastičnosti.

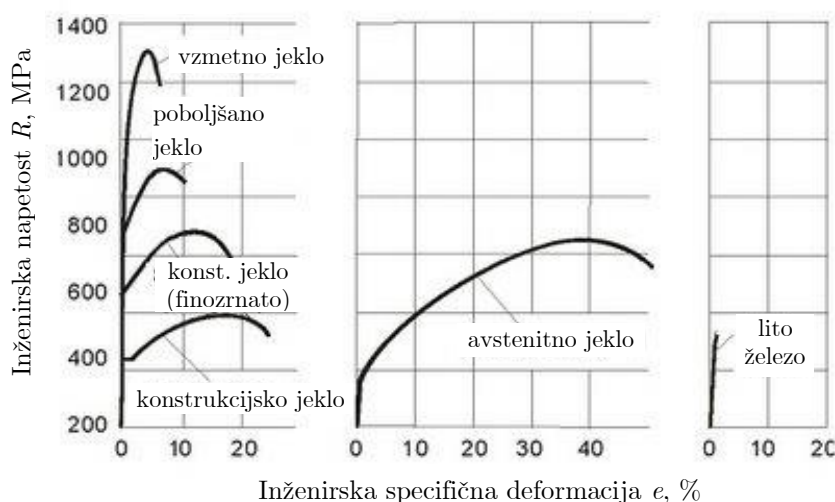
Preglednica 5.2 – Poissonovo razmerje različnih materialov

Material	Poissonovo razmerje
guma	~ 0,5
zlato	0,42
glina	0,3–0,45
magnezij	0,35
titan	0,34
baker	0,33
aluminij	0,33
nerjavno jeklo	0,3–0,31
jeklo	0,27–0,3
siva litina	0,21–0,26
pesek	0,2–0,45
beton	0,2
steklo	0,18–0,3
pena	0,1–0,4
pluta	~0
avksetični materiali	$-1 < \nu < 0$

Določeni materiali pa imajo lastnost, da se pod vplivom natezne obremenitve odebelijo in pod vplivom tlačne obremenitve stanjšajo, zato imajo Poissonovo razmerje manjše od 0. Takšni materiali so v naravi zelo redki in jih imenujemo avksetični materiali (angl. *auxetic*) [5.5]. Danes so takšni materiali največkrat umetno narejeni v obliki celičnih struktur ali v obliki posebnih geometrijskih povezav. Čeprav so avksetični materiali danes še vedno manj poznani, so že kar nekaj časa prisotni v sodobnih izdelkih. Najbolj znan primer takšnega materiala je Gore-Tex [5.6].

## 5.2 Materialni podatki v zahtevnejših simulacijah

Za veliko večino linearnih inženirskih simulacij zgoraj opisani materialni podatki zadoščajo. Včasih pa želimo simulirati naravne pojave nekoliko bolj natančno in realistično. V takšnih primerih potrebujemo za uspešno simulacijo dodatne materialne podatke. Najpogosteje določitev materialnih podatkov v elastičnem področju ni zadostna. Ko obremenitve presežejo določene vrednosti, se material ne obnaša več elastično (po razbremenitvi se ne vrne v začetno stanje), ampak se trajno (plastično) deformira. Dokler je material v elastičnem področju proporcionalnosti, so napetosti v linearni odvisnosti od deformacij v skladu s Hookovim zakonom  $\sigma = E \cdot \varepsilon$ . Pri višjih obremenitvah, ko napetosti presežejo mejo plastičnosti, pa ta odvisnost postane nelinearna  $\sigma = f(\varepsilon)$ . To nelinearno odvisnost navadno določimo prav tako z enoosnim nateznim preizkusom materiala vse do njegove končne porušitve, na osnovi katerega dobimo različna specifična razmerja med inženirskimi napetostmi ( $R = F/A_0$ ) in inženirskimi specifičnimi deformacijami ( $e = \Delta L/L_0$ ) za različne materiale, kot je prikazano na sliki 5.2.



Slika 5.2 – Odvisnost inženirskih napetosti od inženirskih specifičnih deformacij

Ker pride pri nateznem preizkusu do kontrakcije prečnega prereza pri višjih obremenitvah, ko napetosti presežejo mejo plastičnosti, se prečni preizkušanca na tem mestu zmanjša ( $A < A_0$ ), zaradi česar se pojavijo višje, t. i. dejanske napetosti ( $\sigma = F/A$ ). Prav tako se v območju kontrakcije prereza pojavijo povečane dejanske specifične deformacije ( $\varepsilon = \ln[(L_0 + \Delta L)/L_0]$ ). Dejanske napetosti in dejanske specifične deformacije lahko določimo iz inženirskih vrednosti z naslednjima enačbama:

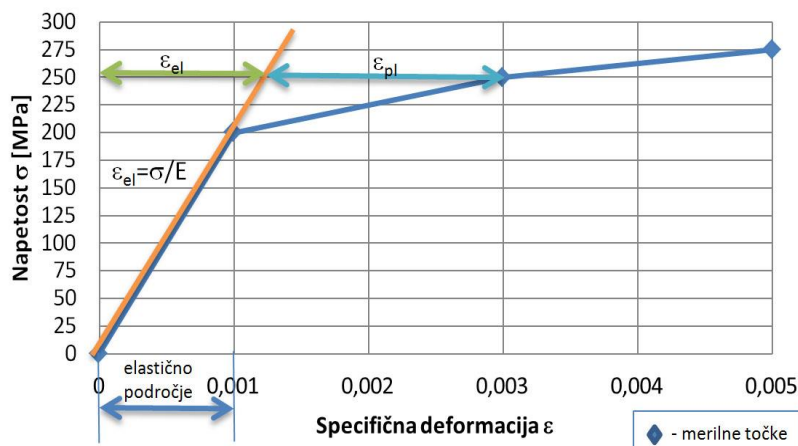
$$\sigma = R \cdot (1 + e) \quad \text{in} \quad \varepsilon = \ln(1 + e) \quad (5.6)$$

V območju plastičnega deformiranja materiala so tako dejanske napetosti in specifične deformacije vedno večje kot njihove inženirske vrednosti, določene z upoštevanjem zgolj začetnega preseka  $A_0$  in dolžine  $L_0$ .

V numeričnih analizah moramo pri upoštevanju nelinearnega (plastičnega) obnašanja materiala vedno podajati razmerja dejanskih napetosti  $\sigma$  in dejanskih specifičnih deformacij  $\varepsilon$  in ne



razmerja njihovih zgolj približnih inženirskih vrednosti  $R$  in  $e$ . Običajno razmerja  $\sigma - \epsilon$  podajamo s poenostavljenimi, odsekovno lineariziranimi grafi, kot to prikazuje slika 5.3.



Slika 5.3 – Odsekovno linearizirana odvisnost dejanskih napetosti od specifičnih deformacij

Pri tem vidimo, da se materialni podatki razdelijo na dva dela: linearnega (elastično območje), kjer je pomemben zgolj modul elastičnosti, in nelinearnega (plastično območje), kjer se odvisnost napetosti in specifičnih deformacij spreminja po drugačnem zakonu v primerjavi z elastičnim območjem.

Veliko bolj zahtevne so simulacije z anizotropnimi materiali, kjer so lastnosti materiala v različnih smereh različne. Glavno težavo predstavlja določanje materialnih podatkov, saj za določitev konstitutivne matrike  $\mathbf{D}$  (enačba 5.7) potrebujemo kar 36 načeloma neodvisnih podatkov, namesto zgolj dveh podatkov ( $E$  in  $\nu$ ) pri izotropnem materialu.

$$\boldsymbol{\sigma} = \mathbf{D} \cdot \boldsymbol{\epsilon}$$

$$\mathbf{D} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \quad (5.7)$$

Včasih so ti podatki lahko določeni s preizkusi ali pa so povezani z definicijo samega končnega elementa, ki je razdeljen na plasti tako, da ima vsaka plast drugačne lastnosti in lahko simuliramo različne laminatne oz. kompozitne materiale.

Poleg tega lahko v numeričnih analizah upoštevamo tudi spremembe temperature, zato je v takšnih primerih pomembno določiti različne termične materialne podatke, npr. temperaturni raztezek, toplotna prevodnost in sevanje. V določenih primerih je seveda treba upoštevati spremembo osnovnih materialnih podatkov (npr. modul elastičnosti) v odvisnosti od

temperature. Spreminjanje materialnih podatkov je v določenih primerih treba predvideti tudi med analizo. V takšnih primerih podatki niso več konstantni, ampak se funkcijsko spreminjajo med simulacijo.

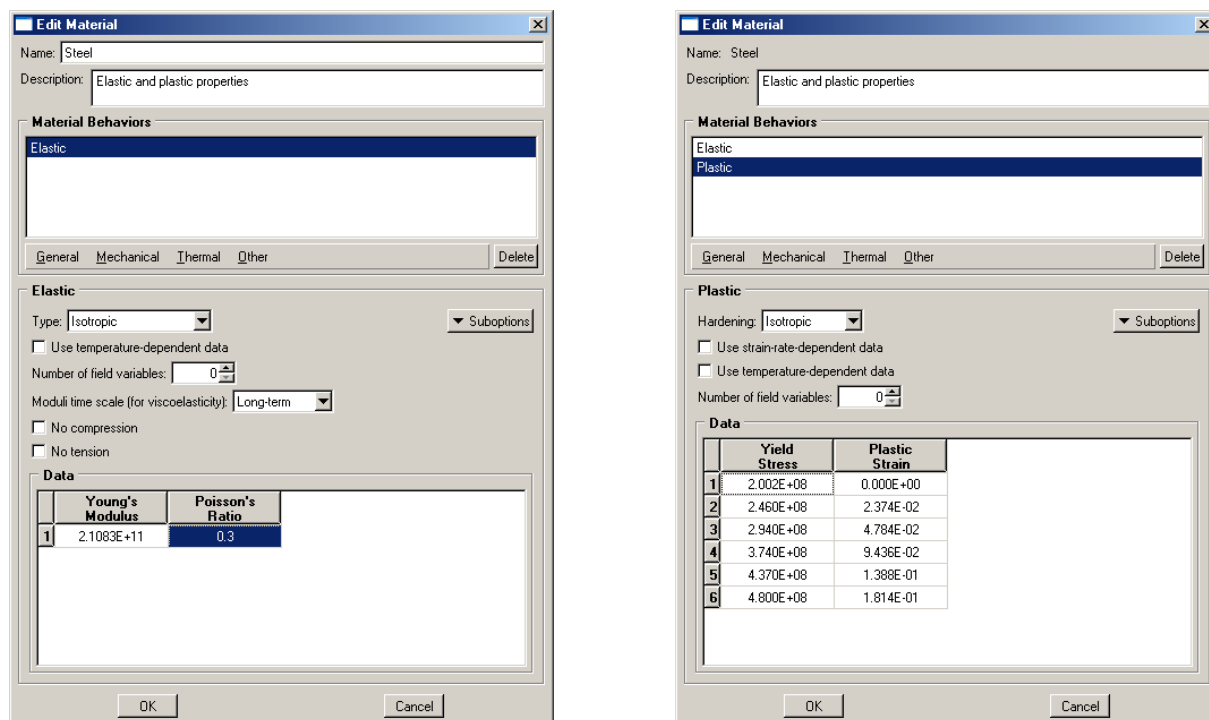
V nekaterih simulacijah želimo upoštevati tudi mikrostrukturo, kemijske lastnosti in poškodbe materiala, zato so pogosto potrebni dodatni podatki, ki jih takšne analize zahtevajo. Materialne parametre za namene tovrstnih numeričnih analiz lahko določimo z različnimi eksperimentalnimi metodami. Natančnost simulacij je nato odvisna od natančnosti uporabljenih materialnih parametrov.

### 5.3 Določanje materialnih podatkov v programih MKE

Različni programi, ki temeljijo na metodi končnih elementov, imajo različne oblike vnosa materialnih podatkov. Kot že omenjeno na začetku poglavja, morajo biti merske enote usklajene, tako da vnesemo zgolj ustrezno število za določen materialni podatek. Materiali imajo v programih tudi ime in oznako, na osnovi česar so povezani z določeno fizikalno lastnostjo oz. s končnim elementom. Za posamezni material lahko navadno določimo naslednje podatke:

- splošne:
  - gostota,
  - dušenje,
  - temperaturni raztezek,
  - lastni materialni parametri/zakoni,
- mehanske:
  - elastičnost,
  - plastičnost,
  - poškodbe,
  - dušenje,
  - razpoke,
  - viskoznost,
- termične:
  - toplotna prevodnost,
  - latentna toplotna,
  - specifična toplota,
- drugo:
  - hidrostatičnost,
  - akustika,
  - elektrika.

Vnos materialnih podatkov se uporablja v predpisanih oblikah; posamezne vrednosti lahko vnašamo v program ali v datoteko, kjer se vrednosti vnašajo na določeno mesto v predpisani številčni obliki. Možnosti za vnos podatkov v programu ABAQUS so prikazane na sliki 5.4.



Slika 5.4 – Določanje materialnih podatkov v ABAQUS-u

## 5.4 Literatura

- [5.1] Young's modulus [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Young%27s\\_modulus](https://en.wikipedia.org/wiki/Young%27s_modulus). [Datum dostopa 5. 12. 2017].
- [5.2] Hooke's law [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Hooke%27s\\_law](https://en.wikipedia.org/wiki/Hooke%27s_law). [Datum dostopa 5. 12. 2017].
- [5.3] Poisson's ratio [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Poisson%27s\\_ratio](https://en.wikipedia.org/wiki/Poisson%27s_ratio). [Datum dostopa 5. 12. 2017].
- [5.4] Strength of materials [splet], Dosegljivo: [https://en.wikiversity.org/wiki/Strength\\_of\\_materials/Lesson\\_2](https://en.wikiversity.org/wiki/Strength_of_materials/Lesson_2). [Datum dostopa 5. 12. 2017].
- [5.5] Auxetics [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Auxetics>. [Datum dostopa 5. 12. 2017].
- [5.6] Gore-Tex [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Gore-Tex>. [Datum dostopa 5. 12. 2017].
- [5.7] A. Ghaedizadeh, J. Shen, X. Ren, Y. M. Xie, "Tuning the Performance of Metallic Auxetic Metamaterials by Using Buckling and Plasticity," *Materials*, let. 9, št. 54, str. 1-17, 2016.
- [5.8] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 6. izdaja. Burlington: Elsevier Butterworth-Heinemann, 2005.



## 6. Fizikalne lastnosti numeričnih modelov

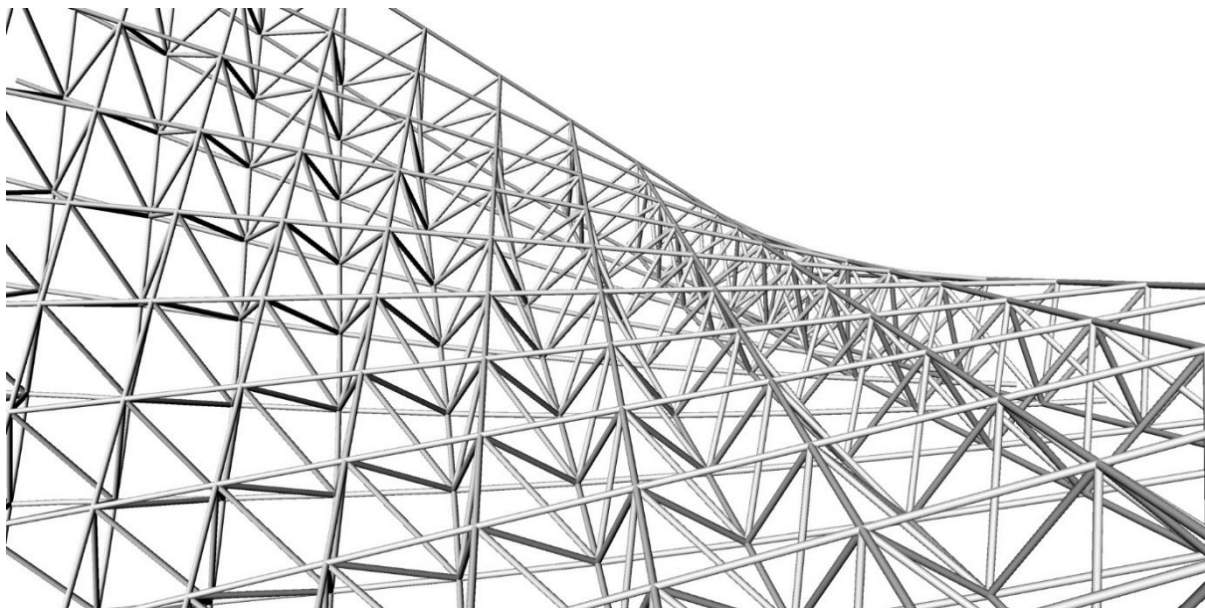
Fizikalne lastnosti modelov v programih, ki temeljijo na metodi končnih elementov, predstavljajo povezavo med geometrijo izdelka, materialnim modelom in mrežo končnih elementov. Fizikalne lastnosti (angl. *physical property*) so v programskih orodjih lahko poimenovane različno, npr. sekcije (angl. *section*). Vsaka fizikalna lastnost je lahko povezana z določenimi materialnimi lastnostmi [6.1]. Geometrija izdelka mora biti povezana z vsaj eno fizikalno lastnostjo. Izdelek je lahko sestavljen iz več različnih fizikalnih lastnosti, vendar morajo le-te biti med seboj kompatibilne. Večina fizikalnih lastnosti je določenih v treh dimenzijah, nekatere fizikalne lastnosti pa so določene samo v ravnini.

Fizikalna lastnost modela, ki se najpogosteje uporablja v konstrukterstvu, je homogeno polno telo (angl. *homogeneous solid body*), saj je najbolj uporabna in realistična. Z drugimi fizikalnimi lastnostmi skušamo model poenostaviti tako, da se fizikalne lastnosti uporabljajo za posebne primere točno določenih simulacij izdelka. Takšne poenostavitve so namenjene doseganju krajših časov računanja ali pa zmanjšujejo obseg modela, ki potem ne potrebuje tako velike količine pomnilnika. V določenih primerih uporaba namenske fizikalne veličine omogoča tudi natančnejše rezultate simulacij.

### 6.1 Linijske fizikalne lastnosti

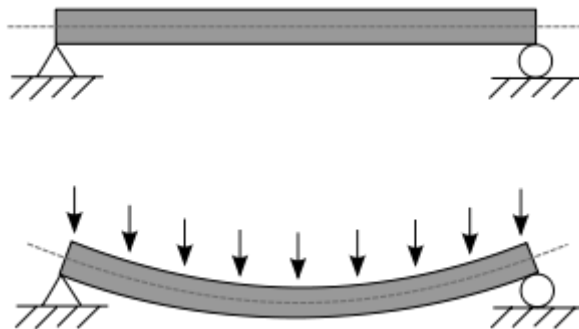
Linijske fizikalne lastnosti so najbolj pogosto povezane s prečnim prerezom (angl. *section*), ki ga nekateri programi uporabljajo kot ime za vse fizikalne lastnosti. Seveda prerez ni popolnoma ustrezna beseda za fizikalne lastnosti linije, ki poleg prereza vključujejo še nekatere druge podatke. Čeprav elemente, ki temeljijo na tej fizikalni veličini, lahko predstavimo z linijo, je poleg tega treba upoštevati, da ta linija lahko leži na poljubnem mestu in v poljubni smeri v prostoru. Zato ne gre za enodimenzionalne fizikalne veličine, ampak za tridimenzionalne. Iz tega razloga je poleg prereza pomembna tudi orientacija le-tega v prostoru. V metodi končnih elementov ločimo dve linijski fizikalni lastnosti: palico in nosilec.

Fizikalna veličina palica (angl. *truss* ali *bar*) [6.2] določa podatke paličnega končnega elementa, ki lahko prenaša le tlačno in natezno obremenitev. Prerez palice je vedno okrogel in edina zahtevana podatka sta površina prereza in material palice. Kljub omejenim sposobnostim lahko s povezavo tako preprostega elementa zgradimo in simuliramo zelo kompleksne palične konstrukcije, kot je prikazano na sliki 6.1.



Slika 6.1 – Paličje [6.3]



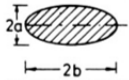



Po drugi strani je nosilec (angl. *beam*) [6.4] veliko bolj kompleksen element in zato je tudi definicija fizikalne veličine nosilca zahtevnejša. Nosilec, prikazan na sliki 6.2, prenaša ob tlačnih in nateznih obremenitvah tudi upogibno in vzvojno obremenitev.



Slika 6.2 – Nosilec [6.4]

Prerez nosilca je lahko poljubne oblike, ki mora biti definiran tako, da je mogoče določiti površino ter odpornostni in vztrajnostni moment. V preglednici 6.1 so prikazane enačbe za izračun fizikalnih lastnosti nosilcev za nekaj oblik prerezov. Programi za analize po metodi končnih elementov omogočajo modeliranje nosilcev s standardnimi ali poljubnimi prerezi nosilcev.

Preglednica 6.1 – Fizikalne lastnosti prerezov nosilcev

Prerez	$A$ ( $m^2$ )	$I$ ( $m^4$ )	$W$ ( $m^3$ )
	$\pi r^2$	$\frac{\pi}{4} r^4$	$\frac{\pi}{2} r^3$
	$b^2$	$\frac{b^4}{12}$	$\frac{b^3}{6}$
	$\pi ab$	$\frac{\pi}{4} a^3 b$	$\frac{\pi a^2 b}{4}$
	$bh$	$\frac{bh^3}{12}$	$\frac{bh^2}{6}$
	$\frac{\sqrt{3}}{4} a^2$	$\frac{a^4}{32\sqrt{3}}$	$\frac{a^3\sqrt{3}}{80}$
	$\pi(r_o^2 - r_i^2)$ $\approx 2\pi r t$	$\frac{\pi}{4}(r_o^4 - r_i^4)$ $\approx \pi r^3 t$	$\frac{\pi}{2}(r_o^3 - r_i^3)$ $\approx 2\pi r^2 t$

Iz enačb v preglednici 6.1 vidimo, da za nesimetrične prereze ni vseeno, kako je nosilec orientiran, saj so lahko vrednosti momentov precej različne. Zato je treba določiti orientacijo prereza v prostoru, ki bo upoštevana pri izračunu pomikov v metodi končnih elementov. Orientacija prereza je običajno določena z vektorjem v smeri koordinatne osi prereza. Fizikalna veličina prereza z izbranim materialom je lahko skupna drugim nosilcem v konstrukciji, orientacija prereza pa je lahko za vsak nosilec različna, zato je treba vsakemu nosilcu v konstrukciji določiti orientacijo prereza.

## 6.2 Ravninske fizikalne lastnosti

Ravninske fizikalne lastnosti modelov so po eni strani preproste, saj iz modela izločimo eno dimenzijo, po drugi strani pa so bolj kompleksne, saj zajemajo prirejene teoretične enačbe. Ločimo tri ravninske fizikalne lastnosti modelov, in sicer: ravninsko napetostno (angl. *plane stress*), ravninsko deformacijsko (angl. *plane strain*) in osno-simetrično (angl. *axisymmetric*).

### 6.2.1 Ravninsko napetostno stanje (RNS)

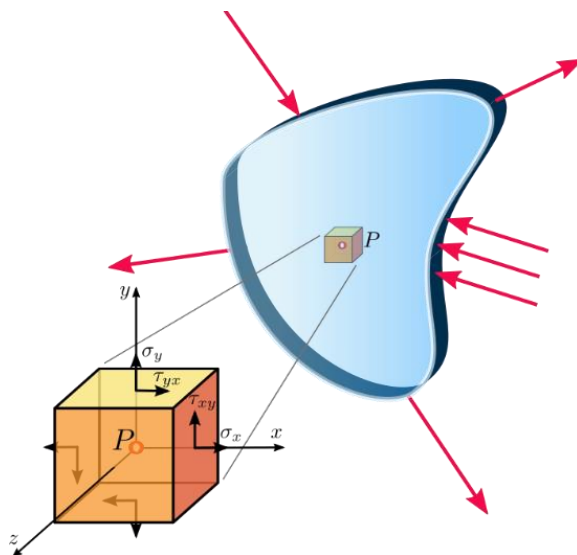
Ravninsko napetostno stanje [6.5] je značilno za stene, katerih konstantna debelina je bistveno manjša od drugih dveh razsežnosti in kjer vse obremenitve delujejo v ravnini stene, slika 6.3. Zaradi tega lahko predpostavimo, da so prečne napetosti v  $z$ -smeri zanemarljivo majhne ( $\sigma_z = \tau_{xz} = \tau_{yz} = 0$ ) in izračunavamo zgolj napetosti v ravnini plošče  $\sigma_x$ ,  $\sigma_y$ ,  $\tau_{xy}$ . Vendar se zaradi vpliva Poissonovega razmerja v steni pojavijo prečne specifične deformacije v  $z$ -smeri (stena spremeni prečno debelino pri vzdolžnem raztegu ali tlaku!), ki pa niso zanemarljive in jih moramo določiti. Zato velja, da so specifične deformacije  $\epsilon_x$ ,  $\epsilon_y$ ,  $\epsilon_z$ ,  $\gamma_{xy}$

načeloma različne od nič, strižne komponente v  $xz$ - in  $yz$ -ravninah pa so enake nič ( $\gamma_{xz} = \gamma_{yz} = 0$ ). Posledično lahko tako modeliramo zgolj srednjo površino stene s površinskimi končnimi elementi in tako 3D-problem pretvorimo v poenostavljeni 2D-problem, kjer razmerje med specifičnimi deformacijami in napetostmi v srednji  $xy$ -ravnini določimo z enačbama (6.1) in (6.2), specifično deformacijo v  $z$ -smeri pa z enačbo (6.3).

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & 0 & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \quad (6.1)$$

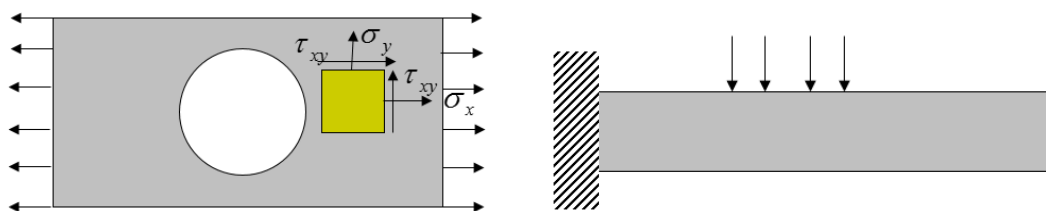
$$\mathbf{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & 0 & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \quad (6.2)$$

$$\varepsilon_z = -\frac{\nu}{1-\nu} (\varepsilon_x + \varepsilon_y) \quad (6.3)$$



Slika 6.3 – Ravninsko napetostno stanje stene [6.5]

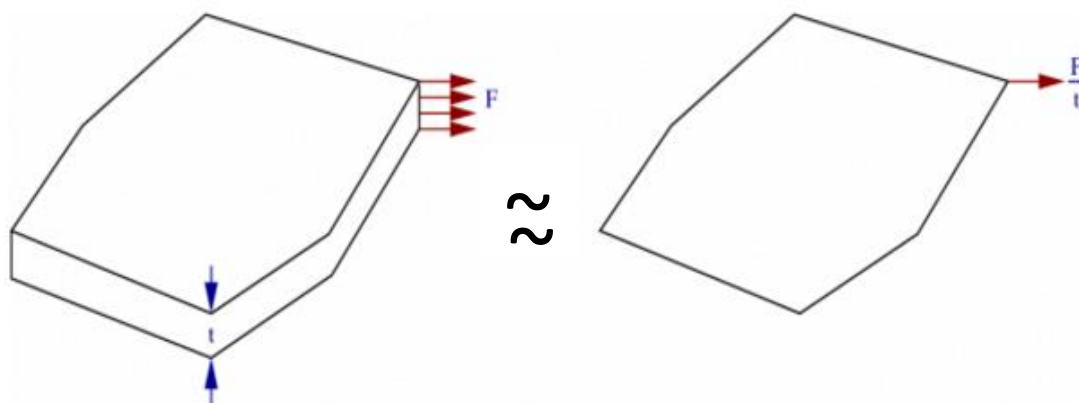
V današnjem času predstavlja simulacija v ravnini na neki način dodatno delo, saj je treba volumsko geometrijo modela poenostaviti v površinski model na srednji ravnini. Na sliki 6.4 sta prikazana tipična primera simulacij, ki jih lahko opravimo z uporabo fizikalne lastnosti ravninskega napetostnega stanja.



Slika 6.4 – Tanka stena z luknjo in ploska konzola



Obremenitve so pri ravninskih problemih običajno podane na enoto debeline, zato je v opisu parametrov fizikalnega modela edini uporabniško zahtevan parameter debelina, seveda poleg parametrov izbranega materiala izdelka. Obremenitve so predpostavljene kot enakomerno razporejene po debelini, zato so v posledičnem površinskem modelu deljene z debelino tako, kot je prikazano na sliki 6.5. Odvisno od posameznega programa je lahko obremenitev samodejno deljena z debelino ali pa mora uporabnik podati obremenitev, preračunano na enoto debeline.



Slika 6.5 – Obremenitev na enoto debeline [6.6]

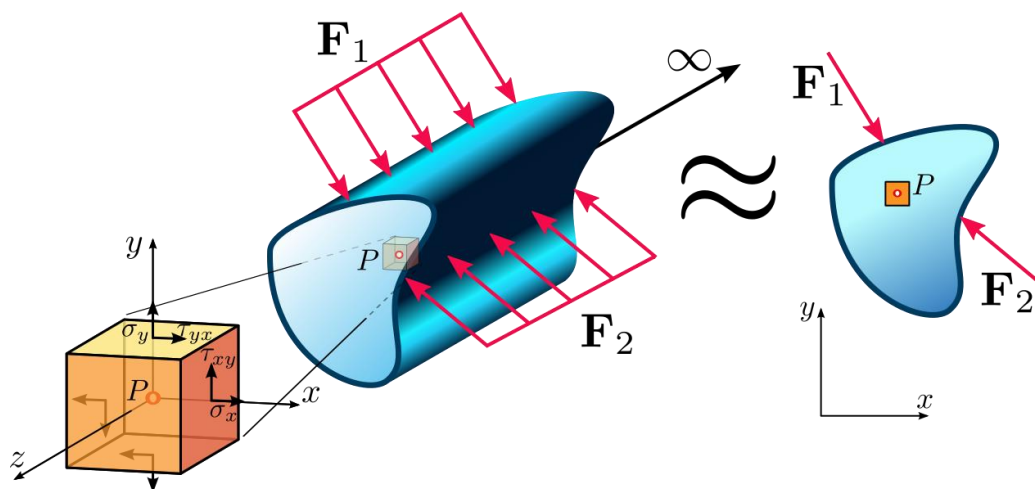
### 6.2.2 Ravninsko deformacijsko stanje (RDS)

Pri ravninskem deformacijskem stanju [6.5] je debelina obravnavanega problema bistveno večja glede na dimenzije konstantnega prečnega preseka, kjer vse obremenitve delujejo normalno na dolžinsko os in so vzdolž nje enakomerno razporejene, kot je prikazano na sliki 6.6. Zaradi tega lahko predpostavimo, da so specifične deformacije v vzdolžni  $z$ -smeri zanemarljivo majhne ( $\varepsilon_z = \gamma_{xz} = \gamma_{yz} = 0$ ) in izračunavamo zgolj specifične deformacije v prečnem prerezu  $\varepsilon_x$ ,  $\varepsilon_y$ ,  $\gamma_{xy}$ . Vendar se zaradi vpliva Poissonovega razmerja v plošči pojavijo vzdolžne napetosti v  $z$ -smeri, ki pa niso zanemarljive in jih moramo določiti. Zato velja, da so napetosti  $\sigma_x$ ,  $\sigma_y$ ,  $\sigma_z$ ,  $\tau_{xy}$  načeloma različne od nič, strižne napetosti v  $xz$ - ter  $yz$ -ravninah pa so enake nič ( $\tau_{xz} = \tau_{yz} = 0$ ). Posledično lahko tako modeliramo zgolj prečni prerez problema s površinskimi končnimi elementi in tako 3D-problem pretvorimo v poenostavljeni 2D-problem, kjer razmerje med specifičnimi deformacijami in napetostmi v prečnem  $xy$ -prerezu določimo z enačbama (6.4) in (6.5), napetost v vzdolžni  $z$ -smeri pa z enačbo (6.6).

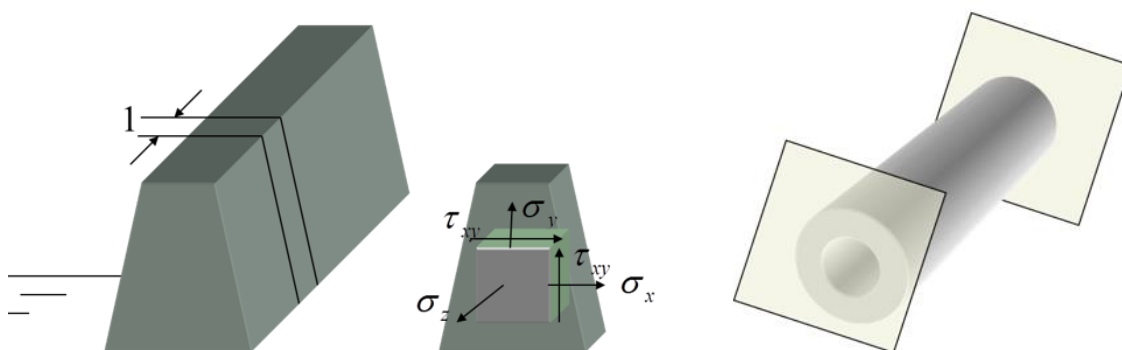
$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \quad (6.4)$$

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \quad (6.5)$$

$$\sigma_z = \nu(\sigma_x + \sigma_y) \tag{6.6}$$



Slika 6.6 – Ravninsko deformacijsko stanje Napaka! Vira sklicevanja ni bilo mogoče najti.



Slika 6.7 – Jez in cev s tlakom v notranjosti

V praksi je veliko primerov, ki jih lahko simuliramo z ravninskim deformacijskim stanjem s poenostavljenim ravninskim modelom v obliki prereza izvornega 3D-modela. Primera takšnih simulacij sta prikazana na sliki 6.7.

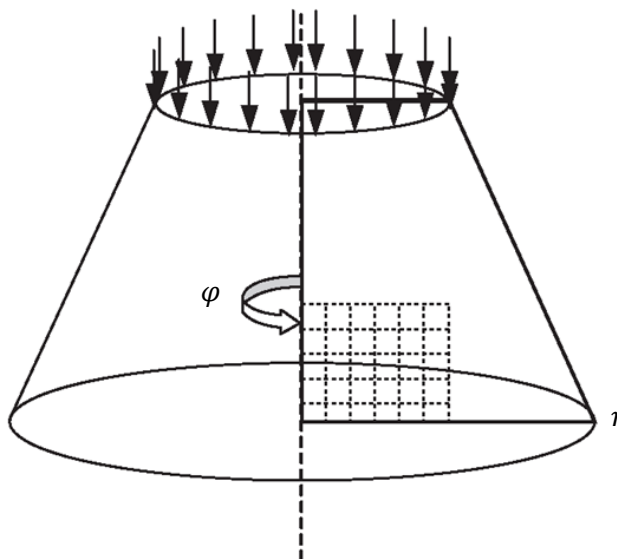
Tako kot pri ravninsko napetostnem stanju je tudi tukaj edini uporabniški parameter, ob materialu, debelina modela. Tudi tukaj mora biti obremenitev podana na enoto debeline tako, kot je prikazano na sliki 6.5, vendar je odvisno od programa, ali so obremenitve samodejno deljene z debelino ali mora uporabnik sam podati obremenitve, preračunane na enoto debeline.

### 6.2.3 Osno-simetrično stanje

Osno-simetrično stanje [6.7] je značilno za probleme z rotacijskimi geometrijami (cev, rezervoar ipd.) s konstantnim radialnim prerezom po obodu, kjer so robni pogoji (obremenitve in vpetja) enakomerno razporejeni po obodu oziroma simetrično na vzdolžno os. Običajno takšne probleme modeliramo v polarnem koordinatnem sistemu  $(r, z, \varphi)$ , kjer je vsaka točka modela določena z osjo rotacije, oddaljenostjo od osi  $r$ , višino  $z$  in rotacijo okoli osi  $\varphi$ . V tem primeru se vsi radialni prerezi po obodu problema deformirajo zgolj v radialni smeri in izračunavamo lahko le specifične deformacije  $\varepsilon_r$ ,  $\varepsilon_z$ ,  $\gamma_{rz}$  in napetosti  $\sigma_r$ ,  $\sigma_z$ ,  $\tau_{rz}$  v radialnem prerezu, pri čemer so napetosti in specifične deformacije v ravninah  $r\varphi$  in  $z\varphi$  enake nič ( $\tau_{r\varphi} = \tau_{z\varphi} = 0$ ;  $\gamma_{r\varphi} = \gamma_{z\varphi} = 0$ ). Zaradi osne simetrije in Poissonovega razmerja se po obodu pojavi enakomerno razporejena obodna specifična deformacija  $\varepsilon_\varphi$  in napetost  $\sigma_\varphi$ . Posledično lahko tako modeliramo zgolj radialni prerez osno-simetričnega problema v ravnini  $rz$  s površinskimi končnimi elementi (slika 6.8) in tako 3D-problem pretvorimo v poenostavljeni 2D-problem, kjer razmerje med specifičnimi deformacijami in napetostmi v radialnem prerezu ter obodni smeri določimo z enačbama (6.7) in (6.8).

$$\begin{Bmatrix} \sigma_r \\ \sigma_z \\ \sigma_\varphi \\ \tau_{rz} \end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & \frac{1-2\nu}{2} \\ 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_r \\ \varepsilon_z \\ \varepsilon_\varphi \\ \gamma_{rz} \end{Bmatrix} \quad (6.7)$$

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \quad (6.8)$$



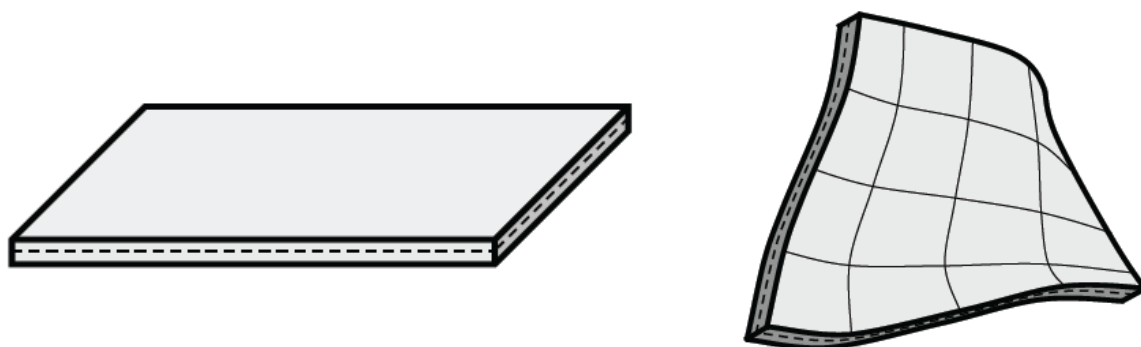
Slika 6.8 – Osno-simetrično stanje in površinski model

Uporabnik mora navadno še pred pripravo geometrije določiti, da gre za osno-simetričen model. Potem mora določiti os rotacije in geometrijo prečnega prereza v ravnini  $rz$ . Pogosto je v programu že vnaprej določena  $z$ -os, ki jo lahko izberemo za os rotacije. Velikokrat pa je namesto koordinatne  $z$ -osi v ta namen izbrana koordinatna  $y$ -os, ki je v računalniških programih običajno prikazana kot vertikalna os.

V osno-simetričnih analizah je treba upoštevati, da so tudi podpore in obremenitve osno-simetrične. Točkovna sila na modelu pomeni enakomerno razporeditev obremenitve okoli osi rotacije. Zato so takšni modeli primerni le za probleme, ki so v vseh ozirih osno-simetrični. Numerični model je potem veliko manjši kot ustrezen prostorski model, zato so tudi analize časovno in prostorsko veliko manj zahtevne.

### 6.3 Plošče, lupine in membrane

Plošče, lupine in membrane (slika 6.9) imajo prav tako kot stene debelino bistveno manjšo od drugih dveh razsežnosti, vendar pri njih obremenitve delujejo tako v srednji ravnini kot tudi prečno nanjo (prečne sile, upogibni momenti). Ob upoštevanju primernih teoretičnih predpostavk (superpozicija membranskega in upogibnega napetostnega stanja) jih lahko modeliramo in izračunavamo zgolj na srednji površini s površinskimi končnimi elementi. Plošče (angl. *plate*) so določene v ravnini in so precej podobne elementom, ki temeljijo na RNS, vendar plošče omogočajo le obremenitve in pomike, ki so zunaj ravnine elementa (npr. pravokotno na ploskev elementa). Plošče lahko postanejo tudi membrane (angl. *membrane*), če je njihova upogibna togost zanemarljiva. Plošče tako simulirajo trde ploskve (npr. kovinske, vezane lesene plošče), membrane pa simulirajo elastične ploskve (npr. gumijaste ponjave).



Slika 6.9 – Plošča in lupina

Če imamo kombinacijo obremenitev oz. pomikov, ki so v ravnini elementa in zunaj ravnine elementa, je treba uporabiti lupine (angl. *shell*), slika 6.9. Lupine so običajno prostorsko ukrivljene.

Ker lahko veliko praktičnih problemov rešujemo z lupinami, so lupinski numerični modeli v praksi precej pogosti. Debelina plošče oz. lupine je dejansko edina fizikalna veličina, ki jo mora uporabnik vnesti in s stališča modeliranja so ti končni elementi razmeroma preprosti. Za te končne elemente so v uporabi precej kompleksne teorije [6.8], [6.9] in za dejanski problem je treba izbrati najprimernejšo. Poleg tega lahko z njimi modeliramo tudi kompozitne materiale, pri katerih je treba določiti materialne lastnosti za posamezne sloje v lupinskem elementu. Zaradi tega ti elementi niso najbolj primerna izbira za začetnike.

## 6.4 Prostorski fizikalni modeli

Pri prostorskih fizikalnih modelih uporabnik ne vnaša dodatnih podatkov, saj je fizikalni model določen z geometrijo, ki v celoti opisuje prostorski model. Fizikalne veličine je kljub temu treba določiti ter z njimi vzpostaviti povezavo med geometrijskim modelom in materialnimi podatki. S stališča uporabnika je fizikalne veličine najpreprosteje določiti v primeru prostorskega modela.

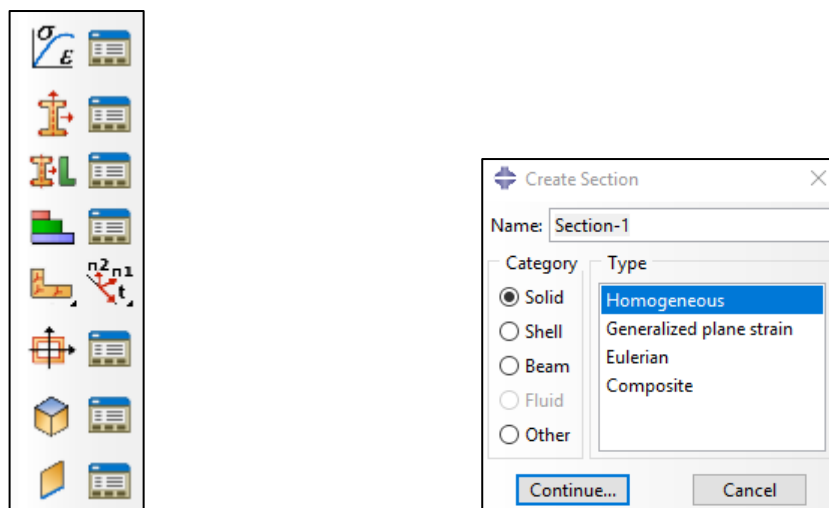
## 6.5 Določanje fizikalnih veličin v programih MKE

Fizikalne veličine so nujno potrebne v programih MKE predvsem zaradi določanja presekov in debelin. Včasih je material neposredno povezan s končnimi elementi, ki so določeni z geometrijo modela, zelo pogosto pa je material povezan posredno z vključitvijo v fizikalno veličino. Imena, ki jih programi uporabljajo za fizikalne veličine, so lahko različna (npr. *physical property*, *section* itd.).

V programu ABAQUS se fizikalne lastnosti določajo s skupino ukazov, imenovano lastnosti (angl. *property*). Orodna vrstica z ukazi je prikazana na sliki 6.10, kjer s prvo ikono določimo materialne podatke, z drugo ikono pa se prikaže okno za izbor vrste fizikalne lastnosti, ki je na sliki prikazan na desni strani.

Vsaka ikona na levi strani slike 6.10 je povezana z določenim ukazom, ki je povezan s fizikalnimi veličinami. S tretjo ikono priredimo fizikalno veličino določenemu delu geometrije. V modelu je lahko več fizikalnih veličin in vsaka lahko vsebuje enak ali drugačen material. Z ikono v četrti vrstici definiramo lastnosti kompozitnih materialov. V peti vrstici določamo orientacije materiala po modelu ali po posameznih palicah. V šesti vrstici izbiramo in določamo prerez paličnih elementov. V sedmi vrstici lahko površino prostorskega modela spremenimo v »kožo« oz. v površine, ki jim lahko dodamo lastnosti plošč, membran ali lupin. V osmi vrstici pa lahko konturo površine spremenimo v robove oz. palice, ki jim določimo palične fizikalne lastnosti.

Veliko ukazov je posvečenih paličnim elementom, kjer je določanje fizikalnih lastnosti najbolj kompleksno. Prav tako je potrebna posebna pozornost pri določanju lastnosti površinskega modela, še posebej če uporabljamo kompozitne materiale. Najmanj zahtevno je določevanje fizikalnih lastnosti za prostorski model.



Slika 6.10 – Fizikalne lastnosti v programu ABAQUS

## 6.6 Literatura

- [6.1] S. N. Pataik, D. A. Hopkins, *Strength of materials*, Boston. Elsevier Butterworth-Heinemann, 2004.
- [6.2] Truss [splet], Dosegljivo: <https://en.wikipedia.org/wiki/Truss>. [Datum dostopa 5. 12. 2017].
- [6.3] Exercise 1: Space Truss Structure [splet], Dosegljivo: <http://wiki.theprovingground.org/usc-arch517-exercise1>. [Datum dostopa 5. 12. 2017].
- [6.4] Beam [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Beam\\_\(structure\)](https://en.wikipedia.org/wiki/Beam_(structure)). [Datum dostopa 5. 12. 2017].
- [6.5] Plane stress [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Plane\\_stress](https://en.wikipedia.org/wiki/Plane_stress). [Datum dostopa 5. 12. 2017].
- [6.6] Plane stress/strain thickness in ABAQUS [splet], Dosegljivo: <http://imechanica.org/node/3290>. [Datum dostopa 5. 12. 2017].
- [6.7] Axisymmetric [splet], Dosegljivo: <http://www.thefreedictionary.com/axisymmetric>. [Datum dostopa 5. 12. 2017].
- [6.8] Kirchhoff–Love plate theory [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Kirchhoff%E2%80%93Love\\_plate\\_theory](https://en.wikipedia.org/wiki/Kirchhoff%E2%80%93Love_plate_theory). [Datum dostopa 5. 12. 2017].
- [6.9] Mindlin–Reissner plate theory [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Mindlin%E2%80%93Reissner\\_plate\\_theory](https://en.wikipedia.org/wiki/Mindlin%E2%80%93Reissner_plate_theory). [Datum dostopa 5. 12. 2017].

## 7. Sestava numeričnih modelov

Pogosto so izdelki sestavljeni iz različnih delov, ki so med seboj povezani. Pri modeliranju pripravimo vsak del posebej, nato pa jih s transformacijami namestimo na predvidena mesta. Simulacije so sicer preprostejše, če je celoten numerični model iz enega kosa, vendar v resničnem svetu pogosto najdemo primere, kjer so deli med seboj le v stiku ali povezava med njimi ni popolnoma toga. S simulacijo moramo te povezave tudi analizirati in jih predhodno ustrezno modelirati, kar bo obravnavano v naslednjih poglavjih. Tukaj bodo predstavljene metode za geometrijsko sestavljanje ločenih delov, kar je pogosto pogoj za modeliranje robnih pogojev in obremenitev. Programi za predprocesiranje pogosto zahtevajo pripravo sestave, tudi če je objekt numerične analize samo iz enega dela. V takem primeru je delo uporabnika zelo preprosto, saj mora del le dodati v sestavo. Sestava več delov pa pogosto predvideva uporabo transformacij v prostoru. Ker transformacije uporabimo tudi pri preslikavah, bo tukaj podan tudi pregled različnih 3D-pogledov in predstavitev 3D-teles na računalniškem zaslonu [7.1].

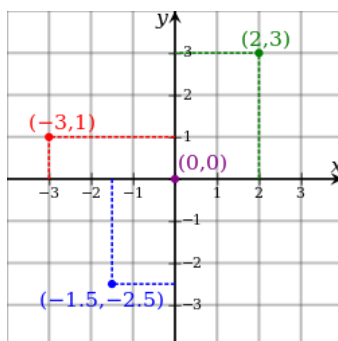
### 7.1 Koordinatni sistemi

René Descartes z latinskim imenom Cartesius si je zamislil povezavo med algebro in geometrijo tako, da je številke zapisal na daljico, slika 7.1. Vsako število lahko tako modeliramo na tem neskončnem številskem traku ali koordinatni osi.



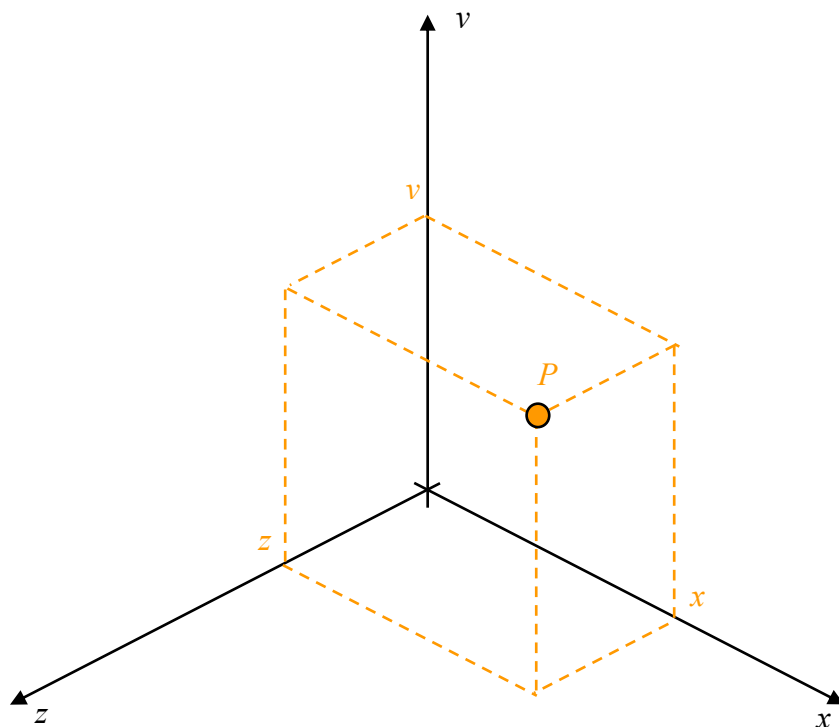
Slika 7.1 – Kartezična koordinatna os [7.2]

Zatem je bila dodana še druga os pravokotno na izvorno in nastal je 2D-koordinatni sistem na sliki 7.2, kjer lahko vsako točko v ravnini predstavimo s parom števil, koordinat vzdolž posameznih koordinatnih osi. Prvo število pove, za koliko je točka oddaljena od izhodišča v  $x$ -smeri, drugo pa, koliko je točka oddaljena od izhodišča v  $y$ -smeri.



Slika 7.2 – Dvodimenzionalni kartezični koordinatni sistem [7.2]

Za predstavitev točk v prostoru dodamo še tretjo  $z$ -os, ki je pravokotna na obe drugi osi. Tako dobimo pravokotni tridimenzionalni kartezični koordinatni sistem, slika 7.3. Ker tretja os gleda proti nam, je ne moremo videti, dokler koordinatnega sistema ne zavrtimo v prostoru in prikažemo v projekciji. V vsakodnevnem življenju si običajno predstavljamo  $xy$ -ravnino kot tla in  $z$ -os kot višino od tal. Pri risanju na računalniku pa pogosto rišemo narise modelov na  $xy$ -ravnino, kot je to prikazano na sliki 7.3, kjer kaže  $y$ -os navzgor. Na ta način je prehod med  $xy$ -ravnino in prostorsko predstavitvijo precej bolj sprejemljiv.

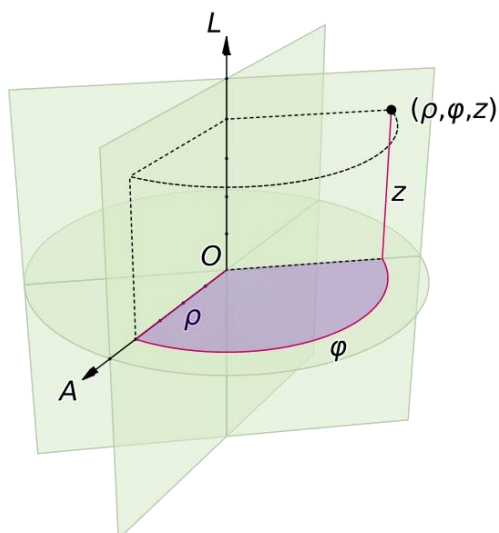


Slika 7.3 – Tridimenzionalni kartezični koordinatni sistem

Vsaka točka je v prostoru predstavljena s tremi koordinatami  $(x, y, z)$ , ki so dejansko komponente krajevnega vektorja od izhodišča do točke v prostoru. Tak kartezični koordinatni sistem se danes najpogosteje uporablja v praksi, seveda pa je mogoče točke v prostoru opisati tudi drugače.

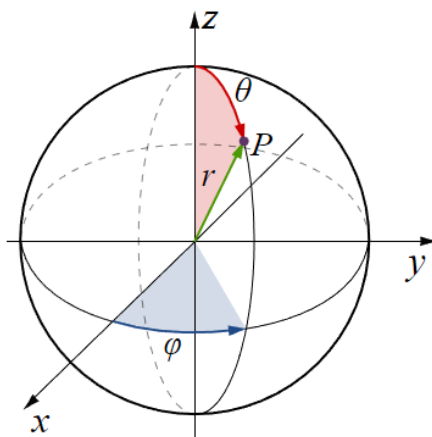
V primeru valjastih (cilindričnih) teles je bolje uporabiti polarni (cilindrični) koordinatni sistem [7.3], prikazan na sliki 7.4. V polarnem koordinatnem sistemu namesto  $xy$ -ravnine uporabimo  $\rho\varphi$ -ravnino, kjer je  $\rho$  polmer (oddaljenost v poljubni smeri od koordinatnega izhodišča) in  $\varphi$  kot zasuka okoli izhodišča v ravnini, medtem ko  $z$ -os označuje razdaljo od izhodiščne ravnine. Pozicije točk so tudi tukaj zapisane s tremi koordinatami  $(\rho, \varphi, z)$ , ki enolično določajo položaj točke v prostoru.





Slika 7.4 – Polarni koordinatni sistem [7.3]

Podobno lahko točke opišemo v krogelnem (sferičnem) koordinatnem sistemu [7.4], prikazanem na sliki 7.5, ki je precej podoben polarnemu koordinatnemu sistemu, le da namesto linearne  $z$ -koordinata uporabimo kot zasuka  $\Theta$  pravokotno na izhodiščno ravnino. Pozicije točk so potem enolično določene s tremi komponentami  $(r, \Phi, \Theta)$ .



Slika 7.5 – Sferični koordinatni sistem

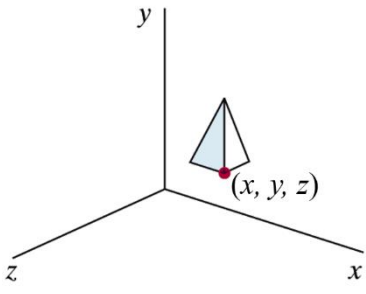
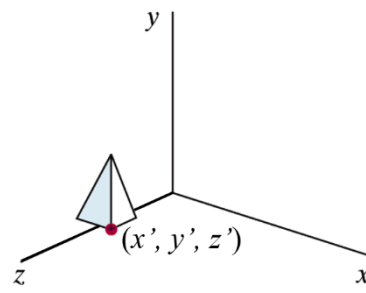
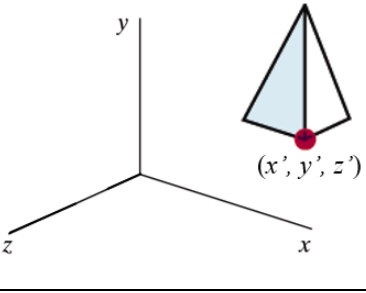
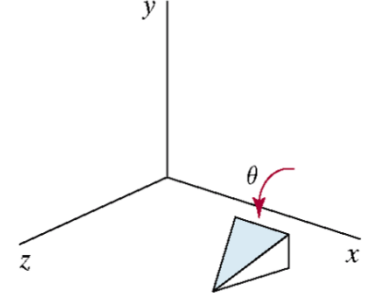
Seveda lahko dva koordinatna sistema prekrijemo tako, kot je prikazano na sliki 7.5, kjer imamo hkrati prikazana krogelni koordinatni sistem in kartezični koordinatni sistem. Prav tako je vedno mogoča pretvorba iz zapisa v enem koordinatnem sistemu v zapis v drugem koordinatnem sistemu.

Ti trije koordinatni sistemi se najpogosteje uporabljajo, niso pa edini koordinatni sistemi. Razviti so bili še številni drugi koordinatni sistemi [7.2], ki se uporabljajo v različne specifične namene.

## 7.2 Transformacije

Sestavni deli so običajno modelirani z relativno postavitvijo. V določenih primerih je modeliranje lahko absolutno, kar pomeni, da objekt modeliramo na njegovem mestu v sestavi in v primerni velikosti ter orientaciji. Pri absolutnem modeliranju pri sestavi ne potrebujemo več nobenih transformacij, vendar je takšno modeliranje precej zahtevno in zahteva natančno načrtovanje modeliranja.

Preglednica 7.1 – Osnovne transformacije objektov

Operacija	Rezultat
Osnovna postavitvev	
Translacija	
Skaliranje	
Rotacija	

Zato je modeliranje z relativno postavitvijo, kjer v skicirki objekte modeliramo okoli koordinatnega izhodišča, precej bolj razširjeno. Seveda je pri takšnem načinu modeliranja treba objekte pri sestavljanju transformirati, da so primerni za sestavo.

Osnovne tri transformacije so translacija (sprememba položaja), skaliranje (sprememba velikosti) in rotacija (sprememba usmerjenosti). V preglednici 7.1 so prikazane osnovne transformacije.

Transformacije v nadaljevanju besedila so izvedene v kartezičnem koordinatnem sistemu. Pri translaciji je treba definirati pomike v smereh koordinatnih osi. Nove koordinate  $(x', y', z')$  po translaciji dobimo z enačbami (7.1) tako, da stari koordinati  $(x, y, z)$  prištejemo vrednost pomika  $(dx, dy, dz)$ .

$$\begin{aligned}x' &= x + dx \\y' &= y + dy \\z' &= z + dz\end{aligned}\tag{7.1}$$

Skaliranje izvedemo tako, da izhodiščne komponente  $(x, y, z)$  pomnožimo s skalirnimi faktorji  $(S_x, S_y, S_z)$ , kot je prikazano v enačbah (7.2). Če so skalirni faktorji večji od 1, se objekt poveča, če pa so manjši od 1, se objekt zmanjša. Če so skalirni faktorji enaki 1, objekt ostane enak. Stranski učinek skaliranja je, da se objekt pri zmanjševanju ali povečevanju hkrati tudi premakne. Zato je pogosto po skaliranju treba uporabiti translacijo, da objekt premaknemo na želeno mesto.

$$\begin{aligned}x' &= S_x \cdot x \\y' &= S_y \cdot y \\z' &= S_z \cdot z\end{aligned}\tag{7.2}$$

Rotacija je precej bolj kompleksna, saj lahko objekt zavrtimo okoli katerekoli osi. Zato za vsako rotacijo rabimo drug niz enačb, ki transformirajo koordinate telesa. Za ravninske like uporabimo rotacijo okoli  $z$ -osi, pri kateri se spremenita koordinati  $x$  in  $y$ ,  $z$ -koordinata pa ostane enaka. Rotacije okoli  $z$ -osi,  $x$ -osi in  $y$ -osi so prikazane v enačbah (7.3), (7.4) in (7.5).

$$\begin{aligned}x' &= x \cdot \cos \theta - y \cdot \sin \theta \\y' &= x \cdot \sin \theta + y \cdot \cos \theta \\z' &= z\end{aligned}\tag{7.3}$$

$$\begin{aligned}x' &= x \\y' &= y \cdot \cos \theta - z \cdot \sin \theta \\z' &= y \cdot \sin \theta + z \cdot \cos \theta\end{aligned}\tag{7.4}$$

$$\begin{aligned}
 x' &= z \cdot \sin \theta + x \cdot \cos \theta \\
 y' &= y \\
 z' &= z \cdot \cos \theta - x \cdot \sin \theta
 \end{aligned}
 \tag{7.5}$$

Ker računalniški programi veliko lažje računajo z vektorji in matrikami, je pogosto treba sestaviti in spreminjati transformacijsko matriko. Transformacijska matrika za ravninske transformacije je reda  $3 \times 3$ . Na voljo imamo samo dve komponenti koordinate. V prostoru je matrika velika  $4 \times 4$ , vektorji koordinat pa imajo samo tri komponente  $(x, y, z)$ . Zato so vse koordinate homogenizirane (projicirane z višje dimenzije) z zadnjo koordinato, ki ima vrednost 1 [7.5].

Poljubna točka je torej s homogenimi koordinatami zapisana z vektorjem, ki je določen s tremi koordinatami in vrednostjo 1 v četrti vrstici tako, kot je prikazano v enačbi (7.6).

$$\mathbf{p} = \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}
 \tag{7.6}$$

Translacijo lahko v transformacijski matriki [7.6] predpišemo s komponentami v četrtem stolpcu matrike tako, kot je prikazano v enačbi (7.7).

$$\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \tag{7.7}$$

Skaliranje je definirano s skalirnimi faktorji, ki so zapisani na diagonali. Če ne želimo spremembe velikosti, so skalirni faktorji enaki 1 tako, kot je prikazano v enačbi (7.7).

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \tag{7.8}$$

Rotacije okoli  $x$ -,  $y$ - in  $z$ -osi so prikazane v enačbi (7.9).

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \tag{7.9}$$

Transformacijo izvedemo tako, da točko, zapisano v obliki enačbe (7.6), pomnožimo s transformacijsko matriko, ki je lahko zapisana v eni od oblik, zapisanih v enačbah (7.7), (7.8) in (7.9). Rezultat je točka, zapisana s homogenimi koordinatami, ki določajo točko po transformaciji.

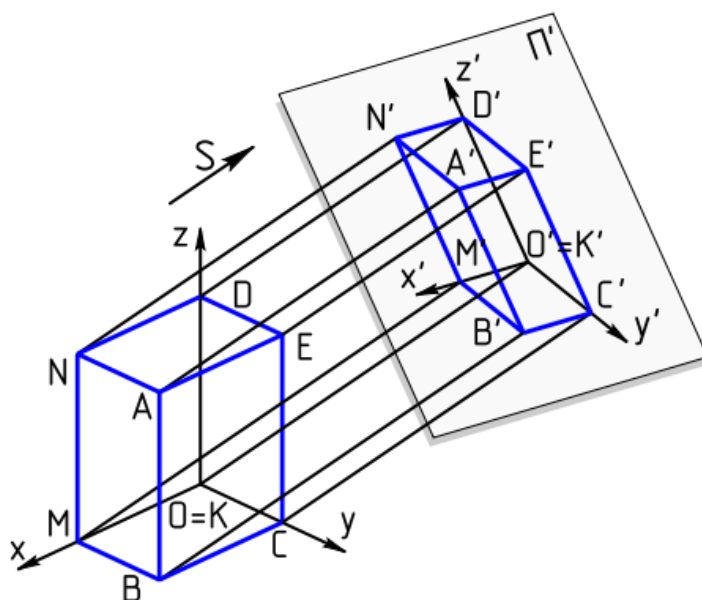
Operacije transformacije niso komutativne, kar pomeni, da je treba operacije izvesti v točno določenem zaporedju. Če zaporedje operacij zamenjamo, bo tudi rezultat drugačen. Če želimo rotacijo poljubnega objekta okoli njegovega težišča, je treba težišče objekta najprej premakniti v koordinatno izhodišče, nato izvršimo rotacijo okoli izhodišča in potem objekt premaknemo na izhodiščno mesto.

Skaliranje in rotacije lahko izvedemo tako, da so komponente transformacije povezane, zato se ohrani osnovna oblika izdelka. Če skaliranje izvedemo z različnimi faktorji povečave v različnih oseh, dobimo spremembo oblike, saj se poruši razmerje med velikostmi v koordinatnih oseh. Podobno popačenje oblike dobimo, če izvedemo rotacijo samo v eni smeri, npr. da določimo vrednost matrike, različno od 0, le v drugem stolpcu prve vrstice transformacijske matrike. Rezultat je običajno oblika striga, saj je rotacija le delno izvršena. Takšne transformacije niso najbolj zaželeni v inženirski praksi, kjer navadno težimo k ohranitvi prej določene oblike telesa, zato so takšne transformacije oblik navadno posledica napak pri transformaciji.

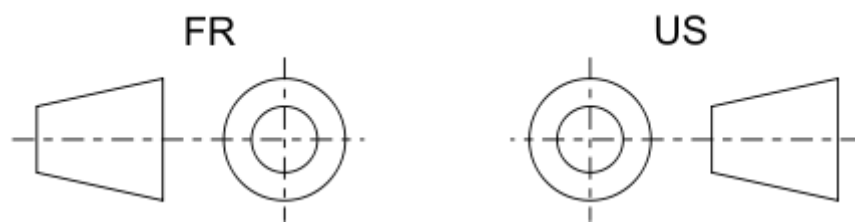
### 7.3 3D-pogledi in projekcije

Ker je večina naprav za prikazovanje računalniških slik še vedno ravninskih, je za prikazovanje prostorskih objektov treba predpisati projekcijo na ravnino ekrana. Projekcija je tudi neke vrste transformacija, pri kateri koordinate točk zmanjšamo za eno dimenzijo. Rezultat je odvisen od tega, kakšno projekcijo uporabimo [7.7].

V inženirstvu so zelo pomembne vzporedne projekcije, kot je na primer izometrična projekcija [7.8]. Izometrično v grščini pomeni, da imamo enako mero. Objekti v takšni projekciji so prikazani tako, da so njihove mere neodvisne od gledišča. Gledišče je pri vzporednih projekcijah neskončno daleč od projekcijske ravnine in navidezni »žarki« iz katerekoli točke objekta so vzporedni. Primer vzporedne projekcije je prikazan na sliki 7.6.



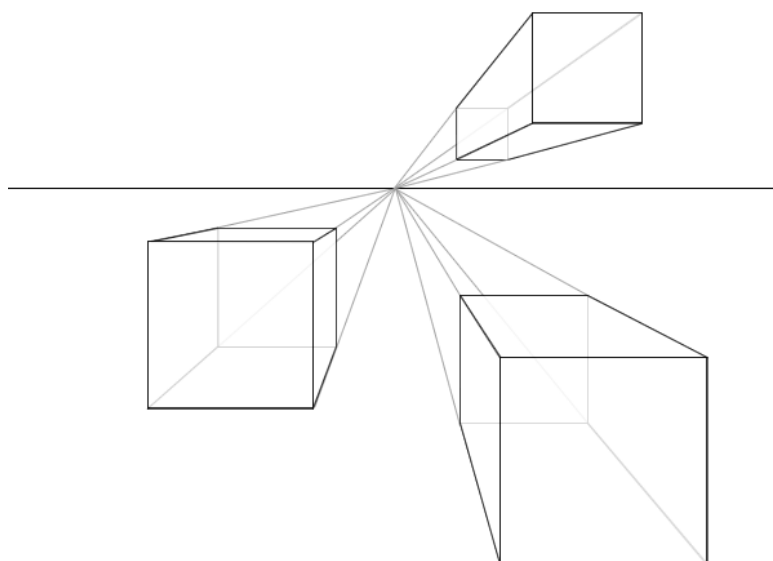
Slika 7.6 – Vzporedna projekcija [7.8]



Slika 7.7 – Ortogonalna (pravokotna) projekcija [7.9]

Posebna vrsta vzporednih projekcij so tiste, kjer je gledišče postavljeno pravokotno na eno od koordinatnih osi. Takšne projekcije se imenujejo ortogonalne (pravokotne) projekcije [7.9] in jih v inženirstvu uporabljamo za risanje tehniške dokumentacije izdelka. Najbolj pogoste so tri ortogonalne projekcije; to so naris, tloris in stranski ris. V svetu se ortogonalne projekcije razlikujejo v postavitvi gledišča in postavitvi pogleda. Osnovni pogled je vedno naris in v ISO-standardu je pod njim tloris, ki prikazuje pogled na izdelek od zgoraj, stranski ris pa je na desni in prikazuje pogled z leve strani. Drugi standardi pa lahko prikazujejo projekcijo tako, da prikazuje pogled na isti strani, kot je gledišče. Nad narisom je tloris (pogled od zgoraj), pod narisom pa je lahko prikazan pogled od spodaj. Prav tako lahko prikažemo dva stranska risa, pri katerih je prikazan pogled na objekt z iste smeri. Na sliki 7.7 je za isti naris pod oznako FR prikazan evropski način prikazovanja stranskega risa (stranski ris na desni strani), z oznako US pa ameriški način (stranski ris na levi strani).

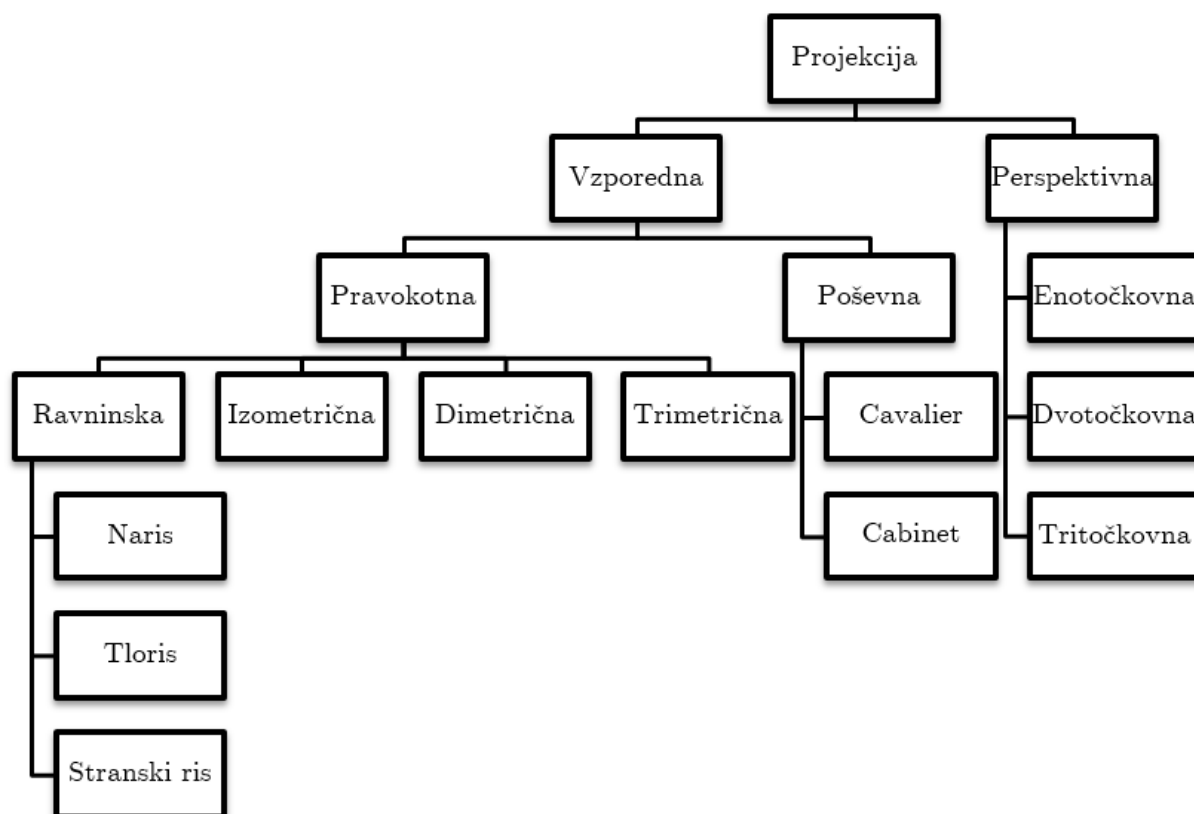
Objekti so v vzporedni projekciji prikazani nerealistično, vendar v projekciji ohranjajo obliko. Zato je lažje dojemati razmerje med stranicami in meriti dimenzije objektov. Za nekoliko bolj realistično prikazovanje vzporednih projekcij se uporabljajo poševne in vzporedne projekcije z različnimi merili v različnih smereh, ki omogočajo bolj atraktivno upodobitev objektov. Takšne projekcije imajo posebna imena, kot so Cabinet, Cavalier, Military, Oblique, Dimetrične, Trimetrične itd.



Slika 7.8 – Perspektivna projekcija [7.10]

Za realistično upodobitev predmetov lahko uporabimo perspektivno projekcijo [7.10], pri kateri so dimenzije večje za del objekta, ki je bližje gledišču, in manjše za del objekta, ki je oddaljen od gledišča. Gledišče ima določeno pozicijo v prostoru in razdalja med glediščem in objektom določa projekcijo. Žarki pri tej projekciji niso vzporedni, ampak konvergirajo v točko, kot je prikazano na sliki 7.8.

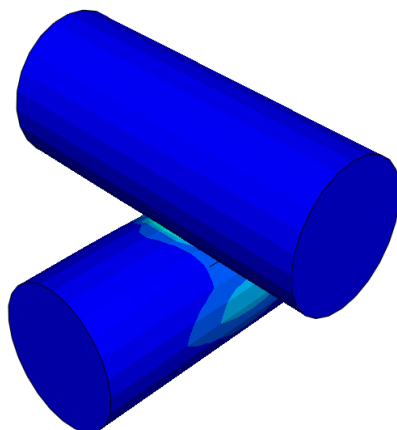
Žarki lahko konvergirajo v eno ali več točk in tako ločujemo različne perspektivne projekcije. Popačenost objektov ni najbolj zaželena v inženirskem delu, zato so takšne projekcije zelo redke v programih za simulacijo. Običajno je možnost takšne projekcije na voljo v programu, vendar nima prave uporabnosti. Na sliki 7.9 je podan pregled različnih projekcij v obliki drevesne strukture.



Slika 7.9 – Pregled projekcij

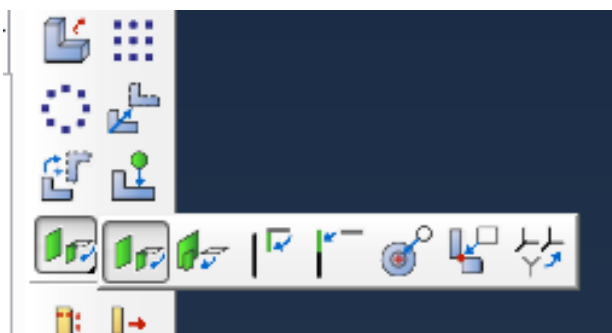
## 7.4 Transformacije v programu ABAQUS

Numerični model je treba v programu ABAQUS vedno sestaviti v sestavo, tudi če model predstavlja samo en del. Če je celoten model sestavljen iz več delov, je treba te dele sestaviti z uporabo geometrijskih transformacij. Smiselno je uporabiti kopije enakih delov, modeliranje več enakih delov ni racionalno. Kopije istih delov nato ustrezno namestimo v sestavo kot v primeru kontakta dveh valjev na sliki 7.10, kjer isti model valja skopiramo, premaknemo in zavrtimo.

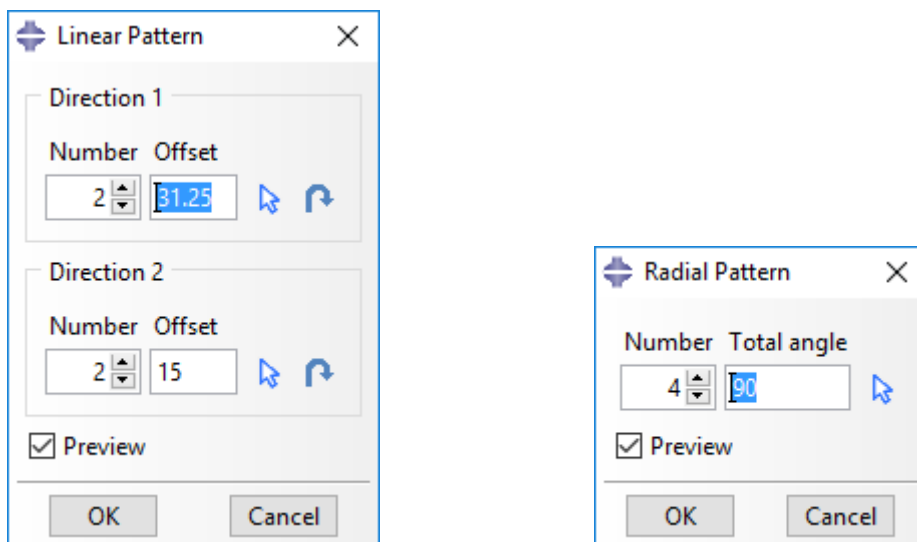


Slika 7.10 – Kontakt dveh valjev

Transformacije lahko naredimo ročno v obliki translacij, skaliranj in rotacij, lahko pa uporabimo tudi ukaze, ki so običajni za modeliranje sestave (npr. stiki, soležnost, vzporednost, koncentričnost). Orodna vrstica s temi ukazi je prikazana na sliki 7.11.



Slika 7.11 – Ukaz za sestavo

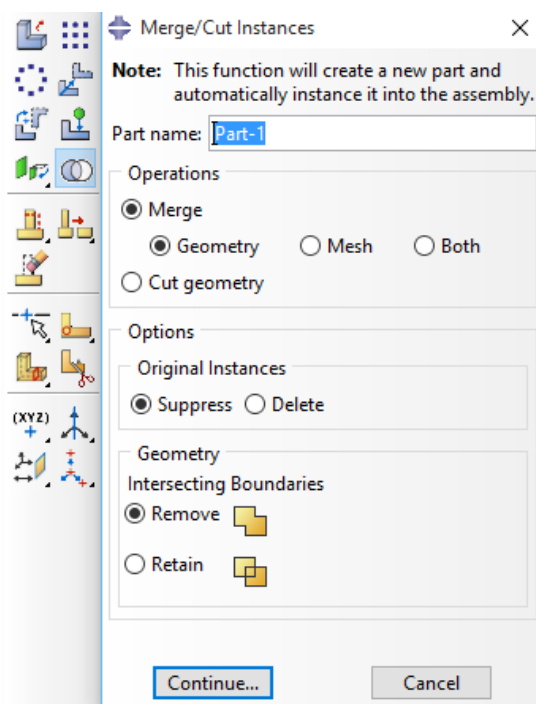


Slika 7.12 – Vzorci za sestavo



Posebnost so transformacije, pri katerih z enim ukazom isti del namnožimo večkrat v obliki vzorca. Vzorci so lahko pravokotni ali krožni. Na sliki 7.12 je primer ukaznega okna za določitev pravokotnega in krožnega vzorca, pri katerem lahko določamo število kopij dela in razdaljo oz. kot med namnoženimi deli.

Posebnost sestave v programu ABAQUS je ukaz, s katerim lahko ločene dele povežemo v en del in jih tako združene analiziramo. To je primerno takrat, ko smo uvozili celotno že sestavljeno sestavo iz drugega CAD-programa, zdaj pa želimo vse dele, ki so že na pravem mestu, med seboj spojiti. V tem primeru lahko uporabimo okno za združevanje delov, prikazano na sliki 7.13.



Slika 7.13 – Združevanje delov

Če želimo obravnavati ločene dele v sestavi, moramo uvoženi sestavi določiti še vse interakcije v obliki povezav ali kontaktov. Pred analizo morajo biti določene vse interakcije in vsi robni pogoji. Noben del ne sme ostati nepovezan ali nepodprt, sicer se bo statična analiza predčasno končala z napako.

## 7.5 Literatura

- [7.1] D. Marsh, *Applied Geometry for Computer Graphics and CAD*. London: Springer-Verlag, 2004.
- [7.2] Coordinate system [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Coordinate\\_system](https://en.wikipedia.org/wiki/Coordinate_system). [Datum dostopa 5. 12. 2017].

- 
- [7.3] Cylindrical coordinate system [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Cylindrical\\_coordinate\\_system](https://en.wikipedia.org/wiki/Cylindrical_coordinate_system). [Datum dostopa 5. 12. 2017].
- [7.4] Spherical coordinate system [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Spherical\\_coordinate\\_system](https://en.wikipedia.org/wiki/Spherical_coordinate_system). [Datum dostopa 5. 12. 2017].
- [7.5] Homogene koordinate [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Homogene\\_koordinate](https://sl.wikipedia.org/wiki/Homogene_koordinate). [Datum dostopa 5. 12. 2017].
- [7.6] Transformation matrix [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Transformation\\_matrix](https://en.wikipedia.org/wiki/Transformation_matrix). [Datum dostopa 5. 12. 2017].
- [7.7] 3D projection [splet], Dosegljivo: [https://en.wikipedia.org/wiki/3D\\_projection](https://en.wikipedia.org/wiki/3D_projection). [Datum dostopa 5. 12. 2017].
- [7.8] Isometric projection [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Isometric\\_projection](https://en.wikipedia.org/wiki/Isometric_projection). [Datum dostopa 5. 12. 2017].
- [7.9] Orthographic projection [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Orthographic\\_projection](https://en.wikipedia.org/wiki/Orthographic_projection). [Datum dostopa 5. 12. 2017].
- [7.10] Perspective (graphical) [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Perspective\\_\(graphical\)](https://en.wikipedia.org/wiki/Perspective_(graphical)). [Datum dostopa 5. 12. 2017].

## 8. Robni pogoji

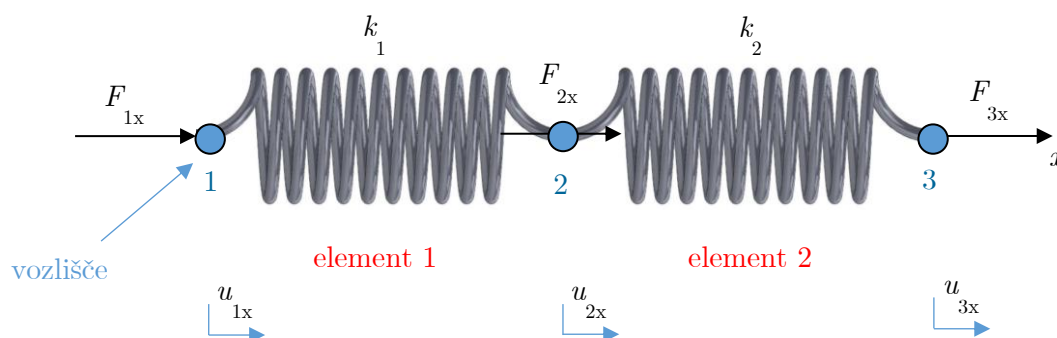
Robni pogoji ali podpore so med najpomembnejšimi uporabniškimi odločitvami pri metodi končnih elementov in se navezujejo na tri Newtonove zakone [8.1]. Po prvem zakonu telo miruje ali se giblje enakomerno, če nanj ne deluje nobena sila ali pa je vsota vseh sil, ki delujejo nanj, enaka nič. V statičnih sestavih ali simulacijah mirujočih sestavov je telo običajno v ravnotežju tako, kot predpisuje prvi Newtonov zakon in ob tem je navadno podprto oz. na določenih mestih v stiku z drugim telesom.

Drugi zakon govori o tem, da je pospešek telesa premo sorazmeren z rezultanto sil na telo in obratno sorazmeren z maso telesa. Ta zakon uporabljamo predvsem pri dinamičnih simulacijah, kjer telesa med simulacijo spreminjajo hitrost.

Reakcijske in kontaktne sile pa določamo po tretjem Newtonovem zakonu, ki pravi: Če prvo telo deluje na drugo telo s silo, deluje to na prvo z enako veliko, a nasprotno usmerjeno silo. V simulacijah zato na mestih, kjer so podpore, vedno dobimo izračunane reakcijske sile, ki so enake in nasprotno usmerjene od sil, ki delujejo na telo. Na tak način lahko tudi preverimo, ali smo pravilno podali obremenitve, saj mora biti vsota vseh obremenitev enaka vsoti vseh reakcij v podporah. Podpore so nujno potrebne v kvazistatičnih simulacijah, da zagotovimo ustrezno ravnotežno stanje.

### 8.1 Podpore v eni dimenziji

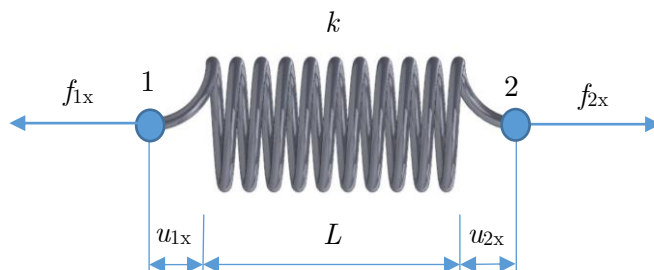
Kadar imamo sistem, ki ga sestavljata dve vzmeti tako, kot je prikazano na sliki 8.1, nas zanimajo pomiki vzmeti v odvisnosti od zunanjih sil  $F$  in togosti vzmeti  $k$ .



Slika 8.1 – Zaporedni vzmetni sistem [8.2]

V metodi končnih elementov tak sistem razdelimo na posamezne končne elemente in v zgornjem primeru je logično, da vsako vzmet zamenjamo z enim končnim elementom. Elementi so določeni z vozlišči, zato na začetku in koncu vzmeti dodamo vozlišče tako, kot je prikazano na sliki 8.1. Vozlišče 2 je skupno elementu 1 in elementu 2.

Pod vplivom zunanjih obremenitev se vzmeti deformirajo v odvisnosti od togosti vzmeti  $k$ . Znanе so zunanje obremenitve in togosti, neznanke pa so trije pomiki  $u$  v vozliščih. Problem je razmeroma preprost, saj nas zanimajo razmere le v eni dimenziji in zato privzamemo, da imajo vozlišča omogočene le pomike v  $x$ -smeri. Tako ima vsako vozlišče v elementih na sliki 8.1 le eno prostostno stopnjo.



Slika 8.2 – Ena vzmet

Za vzmet na sliki 8.2 velja Hookov zakon [8.3], ki ga lahko zapišemo z enačbo (8.1).

$$f_{2x} = k(u_{2x} - u_{1x}) \quad (8.1)$$

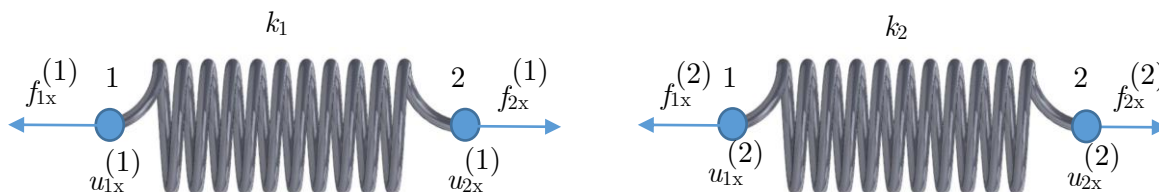
Ker je vsota vseh sil enaka 0 (enačba (8.2)), lahko enačbo (8.1) preoblikujemo v enačbo (8.3), ki jo lahko zapišemo tudi v matrični obliki, kot je prikazano z enačbo (8.4).

$$f_{1x} + f_{2x} = 0 \quad (8.2)$$

$$f_{1x} = -f_{2x} = -k(u_{2x} - u_{1x}) \quad (8.3)$$

$$\begin{Bmatrix} f_{1x} \\ f_{2x} \end{Bmatrix} = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{Bmatrix} u_{1x} \\ u_{2x} \end{Bmatrix} \quad (8.4)$$

Sistem vzmeti s slike 8.1 lahko razdelimo na posamezna elementa in določimo enačbo (8.4) za vsak element posebej, kot je prikazano na sliki 8.3 ter v enačbah (8.5) in (8.6).



Slika 8.3 – Posamezna elementa

$$\begin{Bmatrix} f_{1x}^{(1)} \\ f_{2x}^{(1)} \end{Bmatrix} = \begin{bmatrix} k_1 & -k_1 \\ -k_1 & k_1 \end{bmatrix} \begin{Bmatrix} u_{1x}^{(1)} \\ u_{2x}^{(1)} \end{Bmatrix} \quad (8.5)$$

$$\begin{Bmatrix} f_{1x}^{(2)} \\ f_{2x}^{(2)} \end{Bmatrix} = \begin{bmatrix} k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_{1x}^{(2)} \\ u_{2x}^{(2)} \end{Bmatrix} \quad (8.6)$$

Ko smo razdelili sistem na posamezne elemente, smo podvojili prostostne stopnje. Skupne prostostne stopnje, prikazane v enačbi (8.7), omogočajo sestavo enačb v sistem enačb, ki ga nato lahko rešimo.

$$\begin{aligned} u_{1x} &= u_{1x}^{(1)} \\ u_{2x} &= u_{2x}^{(1)} = u_{1x}^{(2)} \\ u_{3x} &= u_{2x}^{(2)} \end{aligned} \quad (8.7)$$

Podobno velja za sile, kjer mora biti v vsaki točki vsota vseh sil enaka nič. Zato lahko zapišemo enačbe (8.8), kjer dobimo povezave med lokalnimi in globalnimi obremenitvami.

$$\begin{aligned} F_{1x} - f_{1x}^{(1)} &= 0 \Rightarrow F_{1x} = f_{1x}^{(1)} \\ F_{2x} - f_{2x}^{(1)} - f_{1x}^{(2)} &= 0 \Rightarrow F_{2x} = f_{2x}^{(1)} + f_{1x}^{(2)} \\ F_{3x} - f_{2x}^{(2)} &= 0 \Rightarrow F_{3x} = f_{2x}^{(2)} \end{aligned} \quad (8.8)$$

Ker imamo v sistemu tri globalne prostostne stopnje, bo globalna togostna matrika ranga 3, zato je najbolje, da razširimo tudi enačbi (8.5) in (8.6) v enačbi (8.9) in (8.10).

$$\begin{Bmatrix} f_{1x}^{(1)} \\ f_{2x}^{(1)} \\ 0 \end{Bmatrix} = \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} u_{1x}^{(1)} \\ u_{2x}^{(1)} \\ 0 \end{Bmatrix} \quad (8.9)$$

$$\begin{Bmatrix} 0 \\ f_{1x}^{(2)} \\ f_{2x}^{(2)} \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} 0 \\ u_{1x}^{(2)} \\ u_{2x}^{(2)} \end{Bmatrix} \quad (8.10)$$

Če seštejemo enačbi (8.9) in (8.10), dobimo globalen sistem enačb (8.11), ki omogoča rešitev sistema vzmeti s slike 8.1.

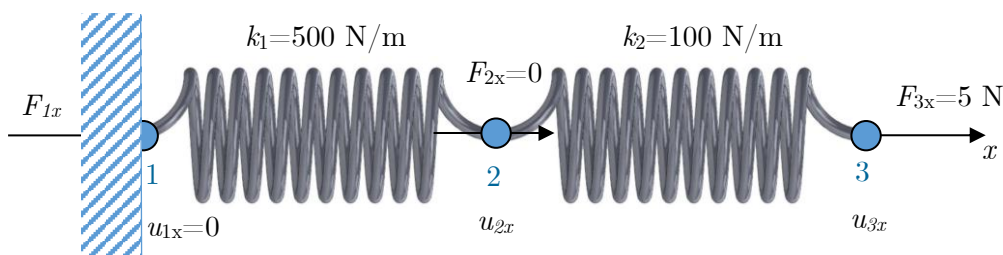
$$\begin{Bmatrix} F_{1x} \\ F_{2x} \\ F_{3x} \end{Bmatrix} = \begin{bmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{bmatrix} \begin{Bmatrix} u_{1x} \\ u_{2x} \\ u_{3x} \end{Bmatrix} \quad (8.11)$$

V enačbi (8.11) je zapisan sistem enačb, ki ga lahko zapišemo v razviti obliki z enačbami (8.12).

$$\begin{aligned} F_{1x} &= k_1 \cdot u_{1x} - k_1 \cdot u_{2x} + 0 \cdot u_{3x} \\ F_{2x} &= -k_1 \cdot u_{1x} + (k_1 + k_2) \cdot u_{2x} - k_2 \cdot u_{3x} \\ F_{3x} &= 0 \cdot u_{1x} - k_2 \cdot u_{2x} + k_2 \cdot u_{3x} \end{aligned} \quad (8.12)$$

Če sta togosti vzmeti  $k_1$  in  $k_2$  znani, imamo še vedno šest neznank in tri enačbe. Ker v vozlišču 2 ni zunanjih obremenitev, lahko privzamemo, da je  $F_{2x}$  enako nič.

Iz sistema enačb (8.12), ki ga lahko enolično rešimo le, če je neznank toliko, kot je enačb, izračunamo poljubne vrednosti pomikov in sil. Zato je treba problem s slike 8.1 spremeniti tako, da bo sistem nekje pritrjen, poleg tega pa mora biti poleg togosti znana zunanja obremenitev sistema. Primer tako spremenjenega sistema je prikazan na sliki 8.4.



Slika 8.4 – Rešljiv sistem vzmeti

Če vstavimo številke v sistem enačb (8.12), dobimo enačbe (8.13), v katerih imamo zdaj le še tri neznanke. Pomik v prostostni stopnji, kjer je podpora, je znan in je enak 0. Namesto nepremične podpore bi lahko vozlišču predpisali tudi poljubno vrednost pomika, saj dokler je vrednost enega pomika znana, je sistem še vedno rešljiv. Neznanka pa je vrednost reakcije v podprtem vozlišču.

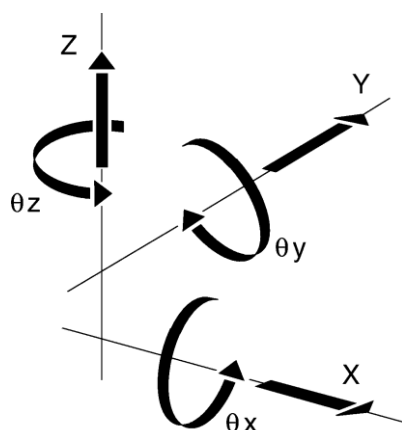
$$\begin{aligned} F_{1x} &= -500 \cdot u_{2x} \\ 0 &= 600 \cdot u_{2x} - 100 \cdot u_{3x} \\ 5 &= -100 \cdot u_{2x} + 100 \cdot u_{3x} \end{aligned} \quad (8.13)$$

Iz sistema enačb lahko izračunamo neznane veličine, kot je prikazano v enačbi (8.14).

$$\begin{aligned} u_{1x} &= 0 \\ u_{2x} &= 0,01 \text{ m} \Rightarrow F_{1x} = -5 \text{ N} \\ u_{3x} &= 0,06 \text{ m} \end{aligned} \quad (8.14)$$

## 8.2 Prostostne stopnje

V primeru v poglavju 8.1 so bile vse prostostne stopnje v eni dimenziji – v  $x$ -smeri. V mehanskem sistemu v prostoru pa poznamo 6 prostostnih stopenj, kot je prikazano na sliki 8.5 [8.4]. Vsako telo oz. točka v prostoru se lahko premakne v smereh koordinatnih osi – v  $x$ -smeri,  $y$ -smeri in  $z$ -smeri. Poleg tega so možni še trije zasuki okoli vseh treh osi. Najlažje si je prostostne stopnje predstavljati z gibanjem letala, kjer so določeni premiki in zasuki tudi poimenovani tako, kot je prikazano v preglednici 8.1. Poleg šestih osnovnih mehanskih prostostnih stopenj so včasih definirane tudi dodatne prostostne stopnje, specifične za nekatere elemente, in druge nemehanske prostostne stopnje, ki jih želimo simulirati. Te prostostne stopnje so lahko tudi specifične za različne programske pakete. Dodatne prostostne stopnje, uporabljene v programu ABAQUS, so navedene v preglednici 8.1 s številčno oznako, večjo od šest.

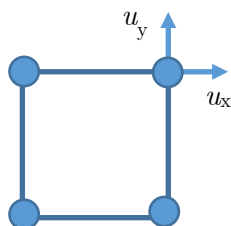


Slika 8.5 – Prostostne stopnje v prostoru

Število prostostnih stopenj pri metodi končnih elementov je odvisno od tipa končnega elementa. Nekateri končni elementi so že po definiciji omejeni z dimenzijami, zato ravninski končni elementi ne morejo imeti prostorskih prostostnih stopenj. Ravninski končni elementi imajo le dve prostostni stopnji v vsakem vozlišču ( $x$ -pomik in  $y$ -pomik), kot je prikazano na sliki 8.6. Pri ravninskih elementih so definirani samo pomiki vozlišč. Zasuk okoli  $z$ -osi ni definiran za te elemente, saj z njimi simuliramo ravninsko toge konstrukcije. Zasuka okoli  $x$ - in  $y$ -osi bi pomenila pomike zunaj ravnine, kar pa za te elemente ni sprejemljivo.

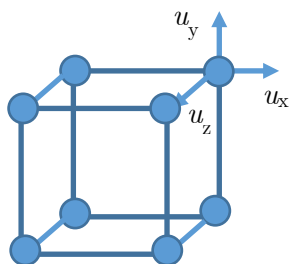
Preglednica 8.1 – Prostostne stopnje (ABAQUS)

Številska oznaka	Koordinatna oznaka	Slovensko ime	Angleško ime
1	$u_x$	levo-desno	left-right
2	$u_y$	naprej-nazaj	forward-backward
3	$u_z$	gor-dol	up-down
4	$r_x$	valjanje	roll
5	$r_y$	naklanjanje	pitch
6	$r_z$	sukanje	yaw
7	/	zvijanje v odprtih presekih nosilcev	warping in open-section beam elements
8	/	akustični tlak, tlak v porah, hidrostatični tlak v tekočinah	acoustic pressure, pore pressure, or hydrostatic fluid pressure
9	/	električni potencial	electric potential
11	/	temperatura	temperature
12+	/	temperatura po globini nosilcev in površinskih elementov	temperature through the thickness of beams and shells



Slika 8.6 – Ravninski končni elementi

Podobno velja za prostostne stopnje pri prostorskih (angl. *solid*) končnih elementih (slika 8.7), ki imajo v vsakem vozlišču definirane le tri prostostne stopnje ( $x$ -pomik,  $y$ -pomik in  $z$ -pomik) in tudi tukaj ni definiranih zasukov, ker prav tako simuliramo toge konstrukcije, ki ne omogočajo zasukov v vozliščih.

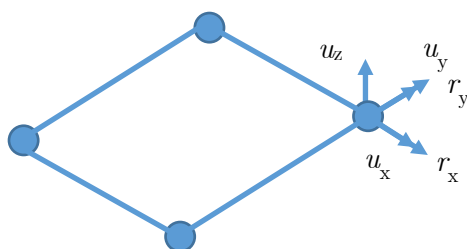


Slika 8.7 – Prostorski končni elementi

Čeprav ravninski in prostorski končni elementi ne omogočajo rotacijskih prostostnih stopenj v vozliščih, ti elementi kljub temu prenašajo momente. Nemogoče je vozlišča teh elementov obremeniti z momentom. V ta namen moramo uporabiti vozlišče ali točko z rotacijsko prostostno stopnjo in jo povezati z elementom. Kombinacija prostostnih stopenj v ravninskem ali prostorskem elementu omogoča simuliranje zasukov. Element na sliki 8.7 ima skupaj 24 prostostnih stopenj (8 vozlišč po 3 prostostne stopnje) in s kombinacijo pomikov v prostorskih stopnjah lahko definiramo poljuben zasuk.

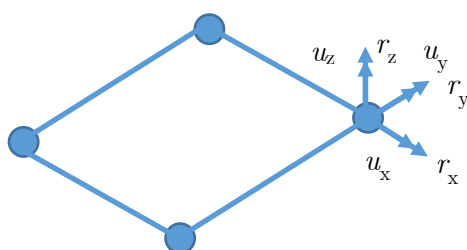
Plošče in membrane imajo v vsakem vozlišču pet prostostnih stopenj, kot je prikazano na sliki 8.8. Omogočene imajo vse pomike ( $x$ -pomik,  $y$ -pomik in  $z$ -pomik) in zasuke zunaj ravnine (okoli  $x$ - in  $y$ -osi), medtem ko zasuki v ravnini (okoli  $z$ -osi) niso definirani.





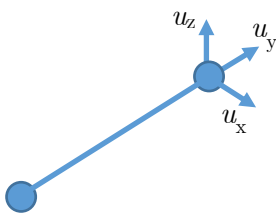
Slika 8.8 – Plošče in membrane

Lupine imajo v vseh vozliščih definiranih vseh šest mehanskih prostostnih stopenj, kot je prikazano na sliki 8.9. V vsakem vozlišču so definirani trije pomiki in trije zasuki.



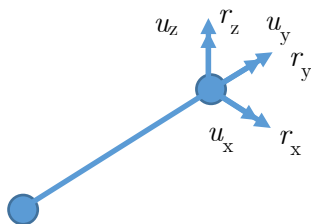
Slika 8.9 – Lupine

Linjski končni elementi tipa palica (angl. *truss*) na sliki 8.10 imajo definirane le tri pomike ( $x$ -pomik,  $y$ -pomik in  $z$ -pomik) v vsakem vozlišču. Zasuki pa niso definirani za ta element, ki prenaša le natezne in tlačne obremenitve, upogiba in torzije pa ne.



Slika 8.10 – Linjski element palica

Linjski končni elementi tipa nosilec (angl. *beam*) na sliki 8.11 ima definiranih vseh 6 mehanskih prostostnih stopenj ( $x$ -pomik,  $y$ -pomik in  $z$ -pomik ter zasuk okoli  $x$ -osi, zasuk okoli  $y$ -osi in zasuk okoli  $z$ -osi) v vsakem vozlišču. Za nosilce z odprtim profilom je predvidena še dodatna prostostna stopnja 7, s katero lahko kontroliramo torzijo v odprtih profilih nosilcev.

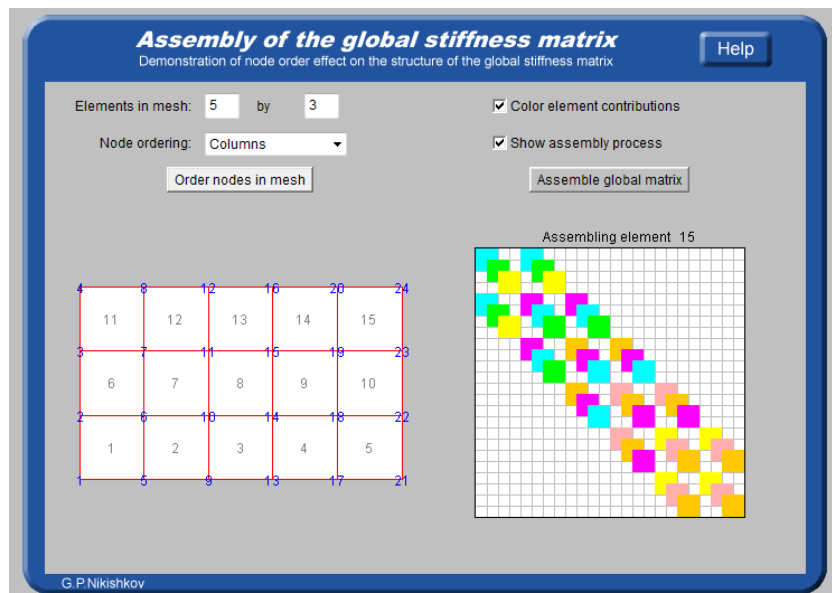


Slika 8.11 – Linijski element nosilec

Običajno uporabljamo le mehanske prostostne stopnje, včasih pa je treba dodati še dodatne prostostne stopnje, kot so navedene v preglednici 8.1. Tako kot z določanjem vrednosti mehanskih prostostnih stopenj lahko na določenih mestih določimo tudi mejne vrednosti drugih veličin, kot je npr. temperatura.

Tako kot v primeru v poglavju 8.1 je treba vse prostostne stopnje pred analizo sestaviti v obliki sistema enačb. Prostostne stopnje, ki jih uporabnik določi, so znane, vse druge prostostne stopnje pa so neznanke v sistemu enačb.

Program, ki temelji na metodi končnih elementov, za vsako prostostno stopnjo določi prispevek togosti, nato te prispevke sešteje v globalno togostno matriko podobno, kot je to prikazano za preprost primer v enačbi 8.11. Za metodo končnih elementov je značilno, da je globalna togostna matrika pasovna in simetrična, kot je razvidno s slike 8.12 [8.5].



Slika 8.12 – Sestavljanje prostostnih stopenj v globalno togostno matriko

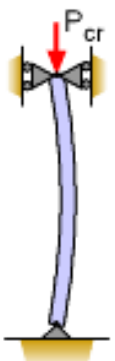
### 8.3 Določanje robnih pogojev

Vsaki prostostni stopnji lahko določimo vrednost pred numerično analizo. S tem določimo znane pomike konstrukcije. Najpogosteje je vrednost pomika prostostne stopnje enaka nič, kar pomeni, da vozlišče v smeri prostostne stopnje miruje. Če vrednost prostostne stopnje ni določena, je prostostna stopnja prosta, kar pomeni, da bo vrednost pomika določena med analizo. V primeru statične analize mora telo mirovati, zato je zahtevano, da morajo biti nekje na modelu določeni pomiki in rotacije v globalnih koordinatnih oseh. Če pomik ali rotacija v določeni smeri ni omejena, to pomeni, da se lahko model v tisti smeri prosto giblje in zaradi tega ni mogoče enoznačno določiti pomikov in reakcij, kot je razvidno iz primera v poglavju 8.1. Na sliki 8.13 je palica na zgornji strani prosta, spodaj pa so vse prostostne stopnje fiksirane ( $u_x = 0$ ,  $u_y = 0$ ,  $u_z = 0$ ,  $r_x = 0$ ,  $r_y = 0$ ,  $r_z = 0$ ), zato rečemo, da je vozlišče tam fiksno (nepomično) vpeto.



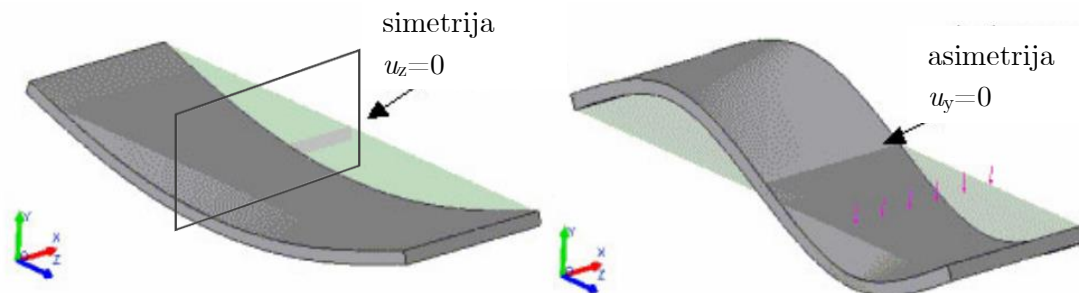
Slika 8.13 – Prosto (zgoraj) in popolnoma vpeto (spodaj) vozlišče [8.6]

Če v vozlišču vpnemo pomike in dovolimo zasuke, je vozlišče členkasto vpeto, kot je prikazano na sliki 8.14, kjer je spodnje vozlišče členkasto vpeto in so vsi pomiki enaki 0, zasuki pa so prosti ( $u_x = 0$ ,  $u_y = 0$ ,  $u_z = 0$ ). Zgornje vozlišče na sliki 8.14 ima omejena le dva pomika ( $u_x = 0$ ,  $u_y = 0$ ), pomik v  $z$ -smeri in vsi zasuki pa so prosti.



Slika 8.14 – Členkasto vpetje [8.6]

Kadar je model simetričen, lahko simetrijo upoštevamo v analizi in tako precej zmanjšamo velikost problema. Na simetrijski ravnini je treba prostorske stopnje primerno omejiti. Na simetrijsko ravnino lahko namestimo simetrične ali asimetrične robne pogoje in v odvisnosti od tega lahko pričakujemo za primer simetrije plošče rezultat, kot je prikazan na sliki 8.15.



Slika 8.15 – Simetrija in asimetrija [8.7]

Za simetrijo in asimetrijo pri prostorskih elementih je dovolj omejiti le en pomik, v splošnem pa je treba omejiti tudi zasuke, če so podprti v izbranem končnem elementu (npr. nosilci, lupine). V preglednici 8.2 so prikazani simetrični in asimetrični robni pogoji v vseh treh koordinatnih smereh z ustreznimi fiksiranimi pomiki in zasuki.

Preglednica 8.2 – Simetrične in asimetrične prostostne stopnje

Vrsta simetrije in asimetrije	Prostostne stopnje
simetrija v $x$ -smeri	$u_x = r_y = r_z = 0$
simetrija v $y$ -smeri	$u_y = r_x = r_z = 0$
simetrija v $z$ -smeri	$u_z = r_x = r_y = 0$
asimetrija v $x$ -smeri	$u_y = u_z = r_x = 0$
asimetrija v $y$ -smeri	$u_x = u_z = r_y = 0$
asimetrija v $z$ -smeri	$u_x = u_y = r_z = 0$

Prostostne stopnje so običajno določene v kartezijskem koordinatnem sistemu tako kot v primerih zgoraj. Seveda pa lahko vedno določimo globalni ali lokalni položaj tudi v cilindričnem ali sferičnem koordinatnem sistemu. Programska oprema omogoča ustvarjanje in uporabo več koordinatnih sistemov, ki jih lahko nato uporabimo v numeričnem modelu. V vrteninah tako pogosto uporabljamo cilindrični koordinatni sistem, saj lahko prostostne stopnje omejimo tudi v aksialni in radialni smeri.

Poleg omejevanja prostostnih stopenj s specifičnimi vrednostmi je v modelu tudi zelo zanimivo povezovanje prostostnih stopenj. Model je lahko sestavljen iz večjega števila posameznih delov, analizirati pa ga želimo kot celoto. V takšnih primerih moramo posamezne

dele povezati med seboj. Najlažje je dele kar »zlit« skupaj, kar pomeni, da namesto ločenih točk na skupnih površinah ustvarimo množico točk, ki so skupne telesoma. Vendar na tak način izgubimo posamezne dele in imamo samo en del, zato pogosto raje ohranimo posamezna telesa in ustvarimo povezave med točkami enega in drugega telesa. Povezave med točkami predstavljajo v numeričnem modelu povezave med prostostnimi stopnjami teles.

Povezave med prostostnimi stopnjami so lahko toge, kar pomeni, da je pomik prostostne stopnje enak kot pomik povezane prostostne stopnje. Večkrat pa so lahko te povezave bolj kompleksne – od primerov, ko so povezane le nekatere prostostne stopnje, druge pa so proste (npr. lahko so povezani pomiki, zasuki pa so prosti), do primerov, ko je razmerje med prostostnimi stopnjami opisano z enačbami. Uporaba povezav je odvisna od mehanizma, ki ga želimo simulirati, saj mora dovolj kvalitetno opisati dejanski fizikalni sistem.

Posebno pozornost je treba nameniti povezovanju vozlišč, ki imajo različno število prostostnih stopenj. Kadar želimo prostorskemu modelu dodati ojačitve tako, da na določenih mestih namestimo linijske elemente tipa nosilec, imamo v vozliščih različno število prostostnih stopenj. V vozliščih prostorskega elementa imamo samo tri prostostne stopnje in če je isto vozlišče tudi del linijskega elementa nosilec, ima poleg treh pomikov še dodatne tri zasuke. V takšnih spojih je treba te dodatne prostorske stopnje omejiti, saj se lahko sicer pojavijo napake pri analizi v obliki izračunanih neomejenih zasukov. Zaradi tega se pojavi navidezno premikanje statičnega modela, ki bi moral biti v statičnem ravnotežju. Takšna napaka, ki povzroči pojav mehanizma v statičnem modelu, se lahko pojavi tudi zaradi drugih napak uporabnika, ki je napačno določil robne pogoje.

Napake zaradi mehanizma odkrije sistem za numerične analize, zato sodijo med lažje napake uporabnika. Veliko težje so napake, kjer statični model še vedno ostane v ravnotežju, vendar so rezultati zaradi napačnih robnih pogojev napačni. Včasih so napetosti na mestih podpor prevelike ali pa so pomiki drugje preveliki. Pogosto te napake odkrijemo šele z meritvami obstoječega fizikalnega problema, zato je vedno dobro opraviti validacijo modela.

Včasih lahko določene napake zaradi robnih pogojev blizu zaokrožitev ali na prehodih med različnimi gostotami mreže tudi zanemarimo, če niso na kritičnih področjih. Takšno ignoriranje oz. zanemarjanje podatkov pa zahteva veliko izkušenj in precej kvalitetno obrazložitev, zato je vedno boljše poiskati robne pogoje, ki kvalitetnejše opišejo razmere na modelu. Pri tem so poleg spreminjanja robnih pogojev večkrat potrebne spremembe tudi drugih delov numeričnega modela (npr. spreminjanje gostote mreže, spreminjanje geometrije itd.).

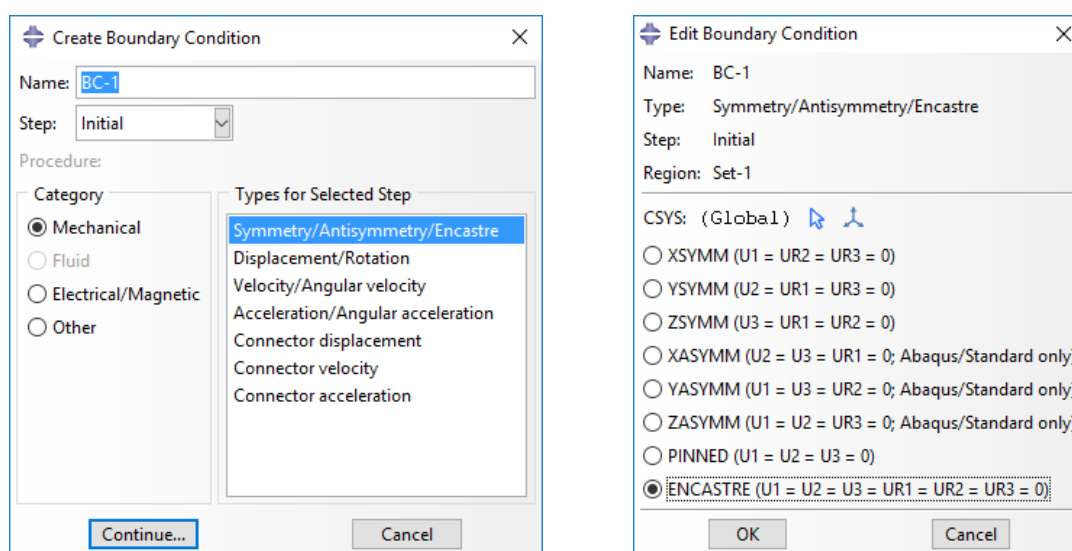
## 8.4 Določanje robnih pogojev v programu ABAQUS

Pri metodi končnih elementov so prostostne stopnje določene v vozliščih, ki so odvisne od mreže končnih elementov. Če robne pogoje določimo v vozliščih, bodo ti izbrisani v primeru spremembe mreže. Zato je bolj primerno določati robne pogoje na geometrijskih elementih, kot so ploskve, robovi ali točke. Poseben primer so t. i. referenčne točke, ki so kreirane neodvisno od modela in jih nato povežemo z geometrijo modela. Če se spremeni geometrija

modela (npr. delitev površine, dodajanje nove površine ali roba itd.), bodo tudi v tem primeru robni pogoji izgubljeni in jih bo treba znova določiti.

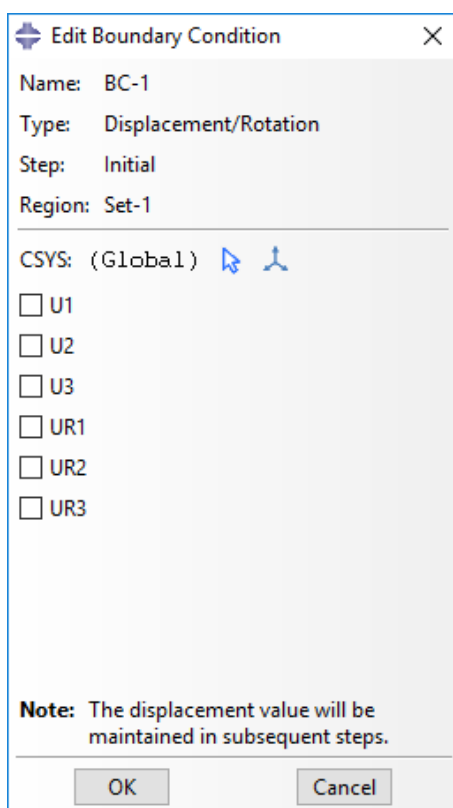
V prvem koraku določanja robnih pogojev je zato treba določiti področje, ki mu želimo predpisati določene robne pogoje. Področje je določeno z uporabniškim izborom in lahko vsebuje povezana geometrijska področja oz. posamezno izbrane točke. Program na osnovi izbora ustvari množico, ki jo nato uporablja za določanje robnih pogojev. Čeprav uporabnik določi robne pogoje za geometrijsko področje, program to vedno pretvori v ustrezne robne pogoje za pripadajoča vozlišča končnih elementov.

Na sliki 8.16 je prikazano okno za izbiro vgrajenih prostostnih stopenj v programu ABAQUS, kjer lahko vozlišča fiksiramo, členkasto vpnemo ter določimo simetrične ali asimetrične robne pogoje. V oknu lahko tudi določimo ali izberemo koordinatni sistem v vrstici, označeni s CSYS, nato pa izberemo ustrezn način vpetja.



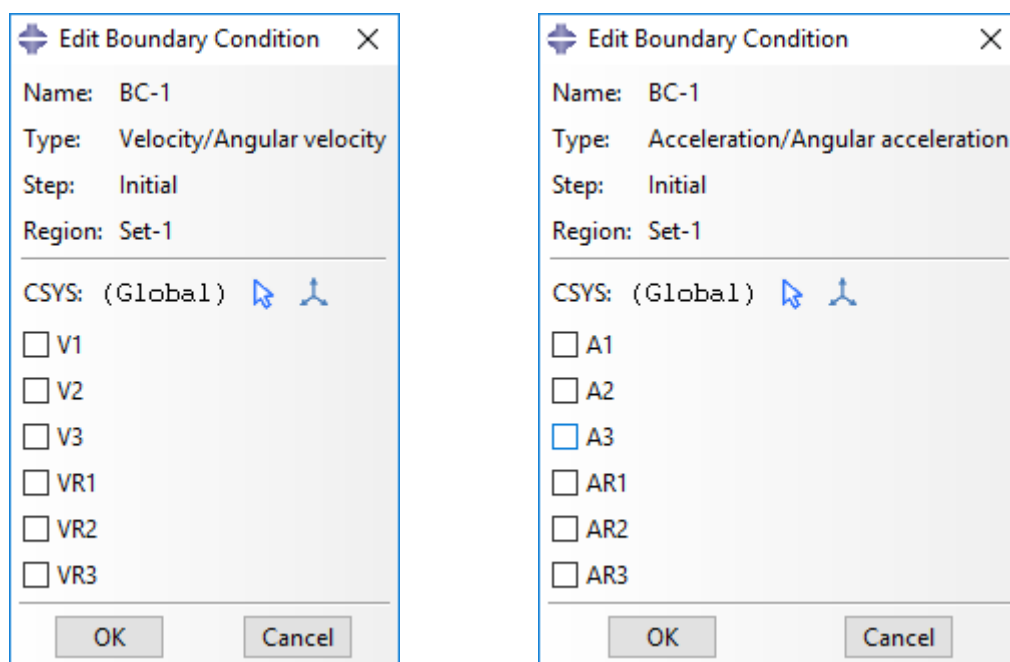
Slika 8.16 – Določitev robnih pogojev v programu ABAQUS

Namesto vgrajenih kombinacij prostostnih stopenj lahko izberemo tudi posamezne prostostne stopnje za izbrana vozlišča, kot je prikazano na sliki 8.17. Uporabnik lahko izbere poljubno kombinacijo pomikov in zasukov v izbranem koordinatnem sistemu za množico vozlišč na izbranem področju.



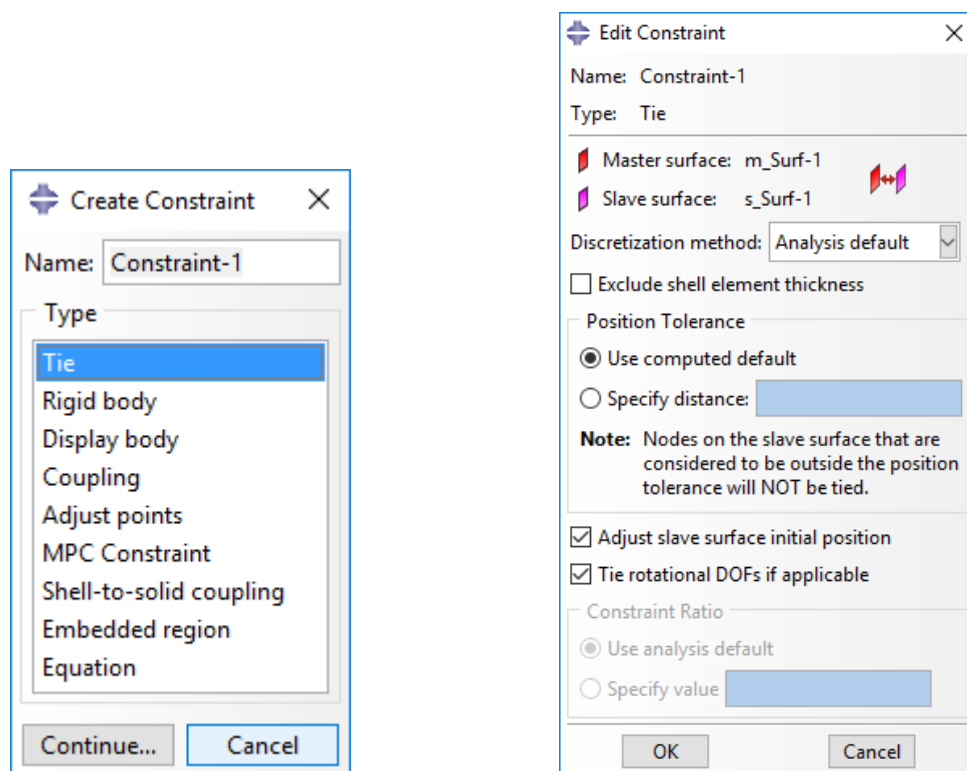
Slika 8.17 – Določanje pomikov in zasukov

V primeru dinamičnih analiz lahko določimo tudi robne pogoje v obliki hitrosti in pospeškov, kot je prikazano na sliki 8.18.



Slika 8.18 – Določanje hitrosti in pospeškov

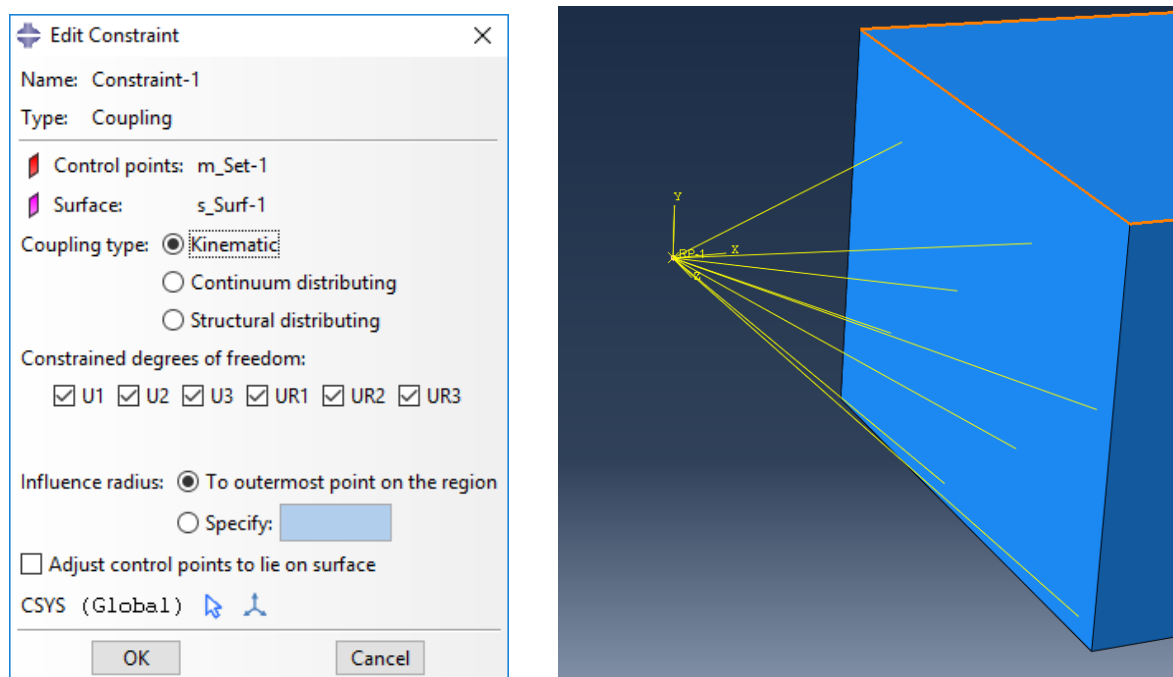
Za povezovanje prostostnih stopenj je v programu ABAQUS na voljo kar nekaj načinov povezovanja, prikazanih na levi strani slike 8.19, od katerih je najbolj pogosto uporabljen ukaz toga povezava (angl. *tie*), s katerim togo povežemo izbrana področja, kot je prikazano na desni strani slike 8.19.



Slika 8.19 – Povezovanje prostostnih stopenj

Zelo zanimiv in priljubljen je način predpisovanja robnih pogojev z uporabo ukaza povezave (angl. *coupling*), prikazanega na sliki 8.20. Ukaz uporabimo tako, da zunaj geometrijskega modela ustvarimo referenčno točko, ki ji predpišemo želene robne pogoje. Nato pa referenčno točko povežemo z geometrijo modela. Tip povezave je lahko tog (angl. *kinematic*), kjer so izbrane prostostne stopnje neposredno povezane.





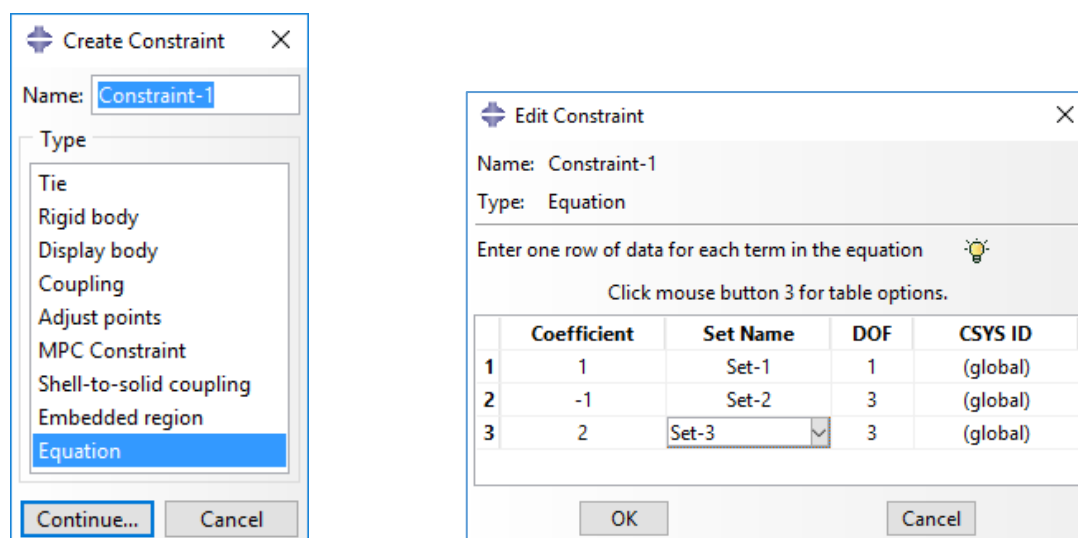
Slika 8.20 – Povezovanje prostostnih stopenj

Tip ukaza povezave (angl. *coupling*) je lahko tudi porazdeljen (angl. *continuum distributing*, *structural distributing*), kjer pa prostostne stopnje geometrije niso toga povezane s prostostnimi stopnjami referenčne točke. Porazdeljene povezave so v teh primerih določene glede na razmerje med rezultantami sil v povezanih geometrijskih vozliščih in sil ter momentom v referenčni točki, pri čemer mora biti ohranjeno ravnotežje sil in momentov okoli referenčne točke. Na tak način je lahko prostostna stopnja referenčne točke v določeni smeri omejena, medtem ko je v povezani geometrijski točki v isti prostostni stopnji izračunan pomik oziroma zasuk.

Uporabnik lahko predpiše tudi enačbo, s katero določa odnos med posameznimi prostostnimi stopnjami poljubnih področij modela. Tako lahko poveže prostostne stopnje treh regij z enačbo 8.15, kjer Skupina1, Skupina2 in Skupina3 označujejo regije z množicami vozlišč.

$$u_x^{(Skupina1)} - u_z^{(Skupina2)} + 2 \cdot u_z^{(Skupina2)} = 0 \quad (8.15)$$

V programu ABAQUS lahko uporabimo za povezovanje prostostnih stopenj tudi ukaz enačba (angl. *equation*). Enačbo vstavimo v obliki preglednice na sliki 8.21. Pomiki numeričnega modela bodo nato izračunani tako, kot je predpisano z enačbo 8.15.



Slika 8.21 – Povezovanje prostostnih stopenj

Določanje prostostnih stopenj v programu za pripravo modela za numerične analize je običajno uporabniku precej prijazno in je zato razmeroma preprosto opravilo. Izbor posameznih geometrijskih elementov (ploskev, robov, točk) pa je odvisen od modeliranja fizikalnega problema, kar pa je pri kompleksnih analizah lahko zelo zahtevno.

## 8.5 Literatura

- [8.1] Newtonovi zakoni gibanja [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Newtonovi\\_zakoni\\_gibanja](https://sl.wikipedia.org/wiki/Newtonovi_zakoni_gibanja). [Datum dostopa 5. 12. 2017].
- [8.2] Introduction to Finite Elements [splet], Dosegljivo: <http://homepages.rpi.edu/~des/IFEA2011Fall.html>. [Datum dostopa 5. 12. 2017].
- [8.3] Hookov zakon [splet], Dosegljivo: [https://sl.wikipedia.org/wiki/Hookov\\_zakon](https://sl.wikipedia.org/wiki/Hookov_zakon). [Datum dostopa 5. 12. 2017].
- [8.4] Degrees of freedom (mechanics) [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Degrees\\_of\\_freedom\\_\(mechanics\)](https://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics)). [Datum dostopa 5. 12. 2017].
- [8.5] Introduction to the Finite Element Method [splet], Dosegljivo: <http://www.ewp.rpi.edu/hartford/~ernesto/S2011/IFEM/Simple/Nikishkov2004.pdf>. [Datum dostopa 5. 12. 2017].
- [8.6] Mechanics – Theory [splet], Dosegljivo: [https://www.ecourses.ou.edu/cgi-bin/ebook.cgi?doc=&topic=me&chap\\_sec=09.2&page=theory](https://www.ecourses.ou.edu/cgi-bin/ebook.cgi?doc=&topic=me&chap_sec=09.2&page=theory). [Datum dostopa 5. 12. 2017].
- [8.7] How to use symmetry and anti-symmetry boundary conditions [splet], Dosegljivo: [https://www.clear.rice.edu/mech403/HelpFiles/CW\\_sym\\_anti-symmetry\\_BC.pdf](https://www.clear.rice.edu/mech403/HelpFiles/CW_sym_anti-symmetry_BC.pdf). [Datum dostopa 5. 12. 2017].

## 9. Obremenitve numeričnih modelov

V nasprotju s podporami obremenitve niso nujno potrebne za vsako vrsto numeričnih analiz. V poglavju 8.1 je prikazan problem sistema vzmeti, kjer imamo definirane pomike in obremenitve. Če so v tem primeru obremenitve enake 0, bodo tudi rezultati v obliki pomikov, deformacij in napetosti enaki nič. V primeru modalne analize, ko nas zanimajo le lastne frekvence sistema, pa dobimo rezultate tudi brez predpisanih obremenitev.

Zato je treba še pred določanjem obremenitev vedeti, kakšna analiza bo izvedena za rešitev danega problema. Včasih je numerična analiza sestavljena iz več korakov, pri čemer lahko za vsak korak določimo pripadajočo obremenitev.

Obremenitve se lahko razlikujejo po lokaciji in po obliki. Lokacija obremenitve je sicer v metodi končnih elementov vedno prostostna stopnja vozlišča, kot je prikazano v poglavju 8.1. Ob pomoči uporabniškega vmesnika pa lahko obremenitev podajamo kot točkovno (v točki oziroma vozlišču) ali kot porazdeljeno po robu, površini ali volumnu. V sodobnih programih je lokacija obremenitev vedno vezana na geometrijske elemente in šele programi za predprocesiranje predpisane obremenitve pretvorijo v ekvivalentne obremenitve vozlišč v skladu z njihovimi prostostnimi stopnjami.

Vrsta obremenitve je običajno sila, vendar so možne tudi druge obremenitve, kot so momenti, tlaki, pomiki, hitrosti, pospeški, temperature itd. Obremenitve so lahko statične (neodvisne od časa) ali dinamične (se spreminjajo s časom).

Enote obremenitve morajo biti usklajene z enotami geometrije in materiala, da dobimo rezultate v želenih enotah. Nekateri programi imajo enote organizirane sistemsko, pri večini programov za numerične analize pa mora uporabnik skrbeti za usklajenost enot.

### 9.1 Vrste obremenitev

Osnovne mehanske obremenitve so sile, ki jih lahko namestimo na geometrijske elemente. Če silo porazdelimo na rob, dobimo linijsko obremenitev, če jo porazdelimo na površino, dobimo površinsko obremenitev in če jo porazdelimo na volumen, dobimo volumsko obremenitev.

Porazdeljena obremenitev, ki je pravokotna na površino, je tlak, ki je lahko določen z rezultanto oz. ekvivalentno silo (npr. v N) ali pa z obremenitvijo na enoto površine (npr. N/mm<sup>2</sup>). Pozitivne vrednosti obremenitve, porazdeljene po površini, določajo tlačne obremenitve, negativne vrednosti pa natezne (ali tlak na notranji strani površine).

Obremenitve, porazdeljene v drugih smereh, so definirane kot vlek (angl. *traction*); pri teh obremenitvah je poleg vrednosti obremenitve treba podati še smer obremenjevanja. Medtem ko je rezultat tlačnih obremenitev enostavno predvidljiv, je predvidevanje vlečnih obremenitev precej bolj zapleteno. Zato je po analizi s takšnimi obremenitvami treba analizirati reakcije, ki pokažejo, ali je bila uporabljena pravilna obremenitev za dani problem.

Določenih obremenitev ne moremo predpisati le na posamezen rob ali površino. Takšna je recimo obremenitev zaradi lastne teže. Takšne obremenitve predpišemo na volumen oz. na

telo in jih pogosto imenujemo telesne ali volumnske obremenitve (angl. *body force*). Silo teže definiramo s telesno silo tako, da določimo vrednost težnostnega pospeška v smeri delovanja gravitacije. Podobno določimo tudi obremenitve zaradi krožnega gibanja. S pospeškom v radialni smeri predpišemo centrifugalno silo v statičnih simulacijah.

V primeru dinamičnih simulacij so obremenitve s hitrostmi in pospeški običajne, vendar jih ne predpišemo vedno kot volumnske obremenitve. V določenih primerih hitrost ali pospešek predpišemo posameznim geometrijskim elementom ali vozliščem končnih elementov.

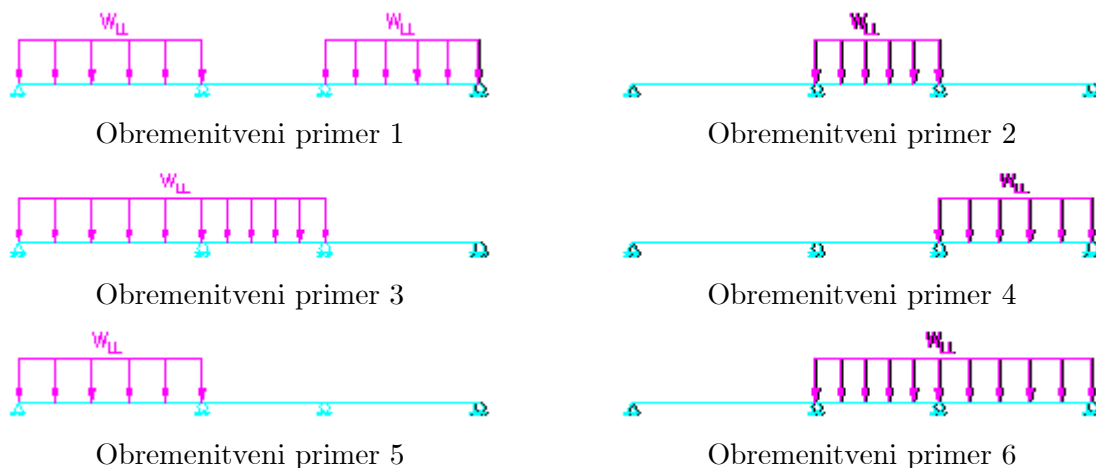
Poleg mehanskih analiz se pogosto izvajajo tudi toplotne analize, kjer so tako kot pomiki v mehanskih analizah tudi temperature lahko hkrati robni pogoji in obremenitve modela. V toplotnih analizah so najpogostejše obremenitve različni izvori toplote ali toplotni tokovi. Za različne numerične analize so seveda na voljo specifične obremenitve. To so na primer akustične, električne ali kakšne druge obremenitve, ki so odvisne od vrste numerične analize.

Obremenitev je za statične analize podana le v obliki konstantne vrednosti, medtem ko se pri dinamičnih analizah obremenitev lahko spreminja s časom. Pri tem običajno predpišemo preproste harmonične ali periodične obremenitve. Dinamična obremenitev je lahko tudi stohastična, kar pomeni, da obremenitev predpišemo v obliki preglednice oz. grafa, kjer je v vsakem trenutku vrednost obremenitve drugačna.

V statičnih analizah lahko opazujemo tudi kombinacije različnih obremenitev, pri čemer želimo rezultate analize za vsako kombinacijo obremenitev posebej. Seveda je možno izvajati zaporedne analize z različnimi obremenitvami, vendar je veliko bolj preprosto, da za isti numerični model samo zamenjamo obremenitve.

Obremenitveni primeri so včasih le poljubne kombinacije obremenitev numeričnega modela, včasih pa so predpisane tako, kot to velja za preračune konstrukcij po EC-standardu [9.1], kjer so predpisane kombinacije obremenitev z lastno težo, vetrom, snegom itd.

Vsak obremenitveni primer je označen z nazivom, kot je prikazano na sliki 9.1. Uporabnik ima po analizi vpogled v toliko ločenih rezultatov, kolikor je bilo predpisanih obremenitvenih primerov (angl. *load case*).



Slika 9.1 – Različni obremenitveni primeri nosilca

Če imamo več obremenitvenih primerov, so ti med seboj neodvisni, kar pomeni, da obremenitveni primeri ne vplivajo drug na drugega. Včasih analizo razdelimo na več korakov, kjer v vsakem koraku model dodatno obremenimo. V takem primeru se analiza nadaljuje tam, kjer se je prejšnji korak končal. Rezultat zato ni neodvisni rezultat posamezne analize, ampak dobimo končni rezultat le na koncu zadnjega koraka analize. Zato je treba vnaprej vedeti, ali želimo opazovati skupen (kumulativen) rezultat različnih zaporednih obremenitev ali želimo rezultat za vsak obremenitveni primer posebej.

## 9.2 Točkovne obremenitve

Točkovna obremenitev se neposredno preslika v obremenitev v prostostnih stopnjah vozlišča. Vozlišča končnih elementov pa so brez dimenzij. To pomeni, da je površina, na katero deluje sila enaka nič, zato so po končani analizi napetosti v tem vozlišču praktično neskončne, kar je posledica singularne rešitve Boussinesq-Cerrutijevega problema v teoriji elastičnosti [9.2]. Vendar je metoda končnih elementov aproksimativna metoda, ki temelji na pohlevnih polinomskih interpolacijskih funkcijah, ki niso sposobne opisati singularne rešitve v okolici vozlišča s točkovno obremenitvijo. Zaradi principa Saint-Venanta [9.3] vpliv singularnosti točkovne obremenitve pada z oddaljenostjo od obremenjenega vozlišča, zato lahko v splošnem uporabimo točkovne obremenitve, vendar tako, da pri vrednotenju rezultatov zanemarimo rezultate v neposredni okolici vozlišča, kjer je predpisana točkovna obremenitev.

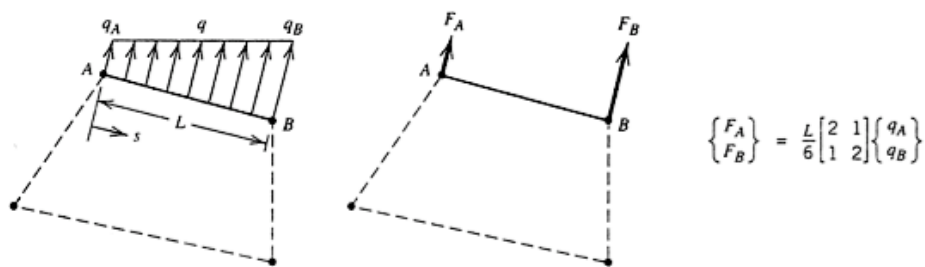
Bolj pogosto uporabimo točkovne obremenitve na geometrijskih točkah oz. referenčnih točkah, ki niso neposredno del geometrijskega modela. Te točke nato ustrezno povežemo z geometrijo numeričnega modela, da se obremenitev pravilno razporedi po vozliščih numeričnega modela.

Točkovne obremenitve uporabnik poda v smereh prostostnih stopenj. Tako je obremenitev vedno podana po komponentah. Silo podamo v smereh pomikov, morebitne momente pa v smereh zasukov, če vozlišče omogoča šest prostostnih stopenj. Pri tem je treba biti pozoren na pozitivne in negativne smeri delujočih obremenitev, ki so skladne s pozitivnimi in negativnimi smermi osi uporabljenega koordinatnega sistema.

## 9.3 Porazdeljene obremenitve

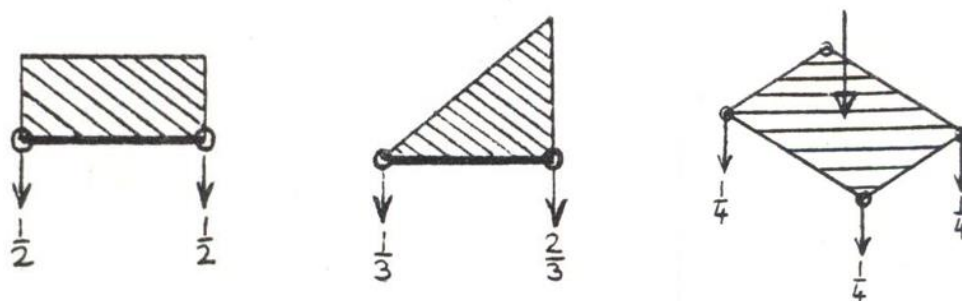
V realnem svetu točkovne obremenitve ne obstajajo, zato obremenitev vedno deluje na določenem območju. Zaradi tega tudi v simulacijah takšne obremenitve porazdelimo po določenem področju numeričnega modela, pri čemer jih pretvorimo v ustrezne vozliščne obremenitve s končnimi elementi diskretiziranega modela z uporabo enačb (3.26) oziroma (3.50) in (3.51). Pri tem je poleg števila prostostnih stopenj vozlišč končnega elementa pomembna tudi stopnja polinoma interpolacijskih funkcij končnega elementa.

Za linearne elemente je ta porazdelitev razmeroma preprosta. Primer razdelitve linijske obremenitve, porazdeljene po sosednjih vozliščih končnega elementa z linearnimi interpolacijskimi funkcijami, je prikazan na sliki 9.2.



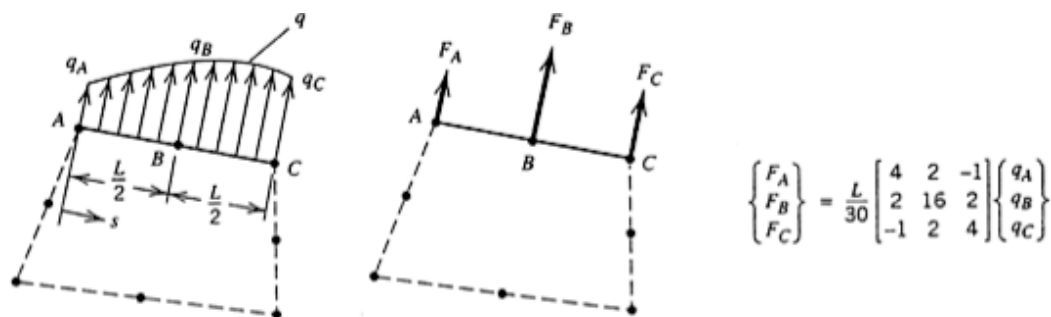
Slika 9.2 – Porazdelitev obremenitve pri linearnih linijskih končnih elementih

Na sliki 9.3 so prikazane porazdelitve ekvivalentne obremenitve v vozlišča pri različnih obremenitvah na robove in ploskve v primeru linearnih končnih elementov.



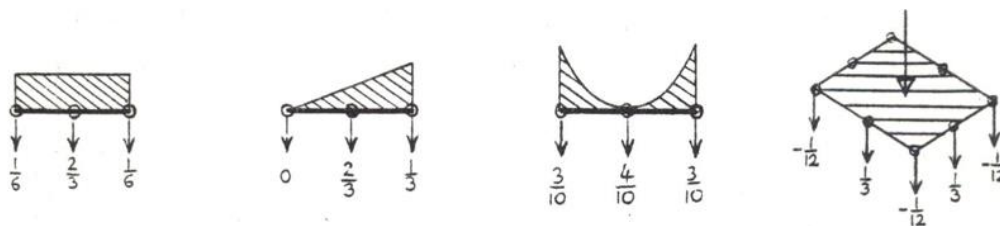
Slika 9.3 – Porazdelitev ekvivalentne obremenitve pri linearnih končnih elementih [9.4]

V končnih elementih s kvadratnimi interpolacijskimi funkcijami imamo na stranicah kvadratne oz. parabolične interpolacijske funkcije, zato je tudi porazdelitev obremenitve na vozlišča s tem sorazmerna, kot je prikazano na sliki 9.4.



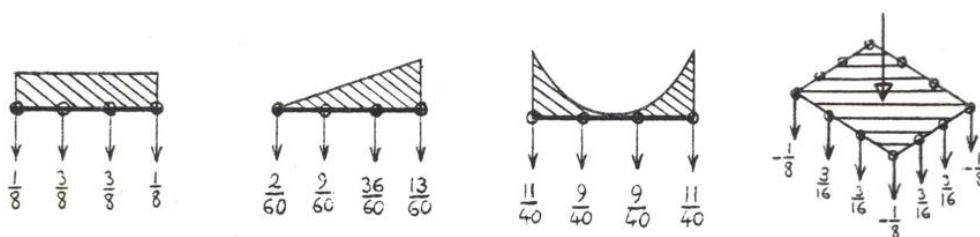
Slika 9.4 – Porazdelitev obremenitve pri kvadratnih ploskovnih končnih elementih

Izračunani deleži ekvivalente obremenitve v vozliščih kvadratnih končnih elementov so prikazani na sliki 9.5 glede na porazdeljeno obremenitev na robu ali ploskvi.



Slika 9.5 – Porazdelitev ekvivalentne obremenitve pri kvadratnih končnih elementih [9.4]

Na podoben način lahko izračunamo obremenitve v vozliščih tudi za kubične končne elemente, kot je prikazano na sliki 9.6.



Slika 9.6 – Porazdelitev ekvivalentne obremenitve pri kubičnih končnih elementih [9.4]

V sodobnih predprocesorjih za metodo končnih elementov uporabnik obremenitve običajno povezuje z geometrijskim modelom neodvisno od mreže končnih elementov. Pogosto v začetni fazi dodajanja obremenitev sploh še ni tvorjene mreže končnih elementov v numeričnem modelu. Kasneje, ko se generira mreža končnih elementov, program samodejno poveže obremenitve z vozlišči končnih elementov tako, kot je prikazano zgoraj. V posebnih primerih, ko uporabnik predpisuje obremenitve neposredno v vozlišča, mora upoštevati prispevke v vsakem končnem elementu in nato sešteti prispevke sosednjih elementov v skupnem vozlišču.

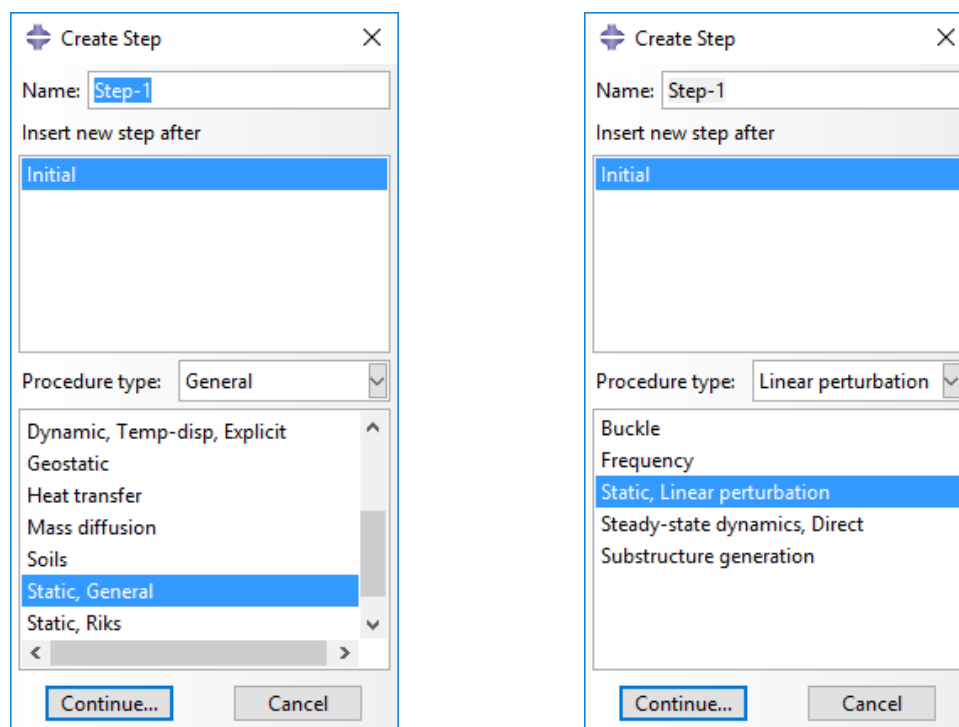
Obremenitev pa ni vedno skladna z geometrijskim modelom. Tako morda v resnici obremenitev deluje samo na del ploskve ali roba, ki je del geometrijskega modela. V takšnih primerih mora uporabnik geometrijski model ustrezno prilagoditi. Običajno so v ta namen na voljo ukazi za delitev volumna, površine ali roba. Zato numerični model virtualno razrežemo, da dobimo ločeno površino ali rob za določitev obremenitev.

## 9.4 Predpisovanje obremenitev v programu ABAQUS

V programu ABAQUS je treba najprej izdelati ali uvoziti geometrijski model, ki ga je treba dodati v sestavo. Ko je sestava določena, ustvarimo obremenitveni korak analize (angl. *step*), v katerega lahko vključimo obremenitve. Podpore so lahko predpisane v vgrajenem t. i. začetnem koraku (angl. *initial step*), za obremenitve pa je treba narediti nov korak tako, kot je prikazano na sliki 9.7.

Na levi strani slike 9.7 je splošen tip analize (angl. *general*), kjer lahko z izborom izberemo statično analizo (angl. *static – general*). Namesto tega lahko izberemo tudi drugo vrsto analize, kot so toplotne, dinamične ali kakšne druge nelinearne analize.

Na desni strani slike 9.7 so prikazane možnosti, če želimo analize z več obremenitvenimi primeri (angl. *linear perturbation*). Tudi tukaj lahko izbiramo med več možnostmi in se lahko odločimo za statične, dinamične, modalne ali uklonske analize.

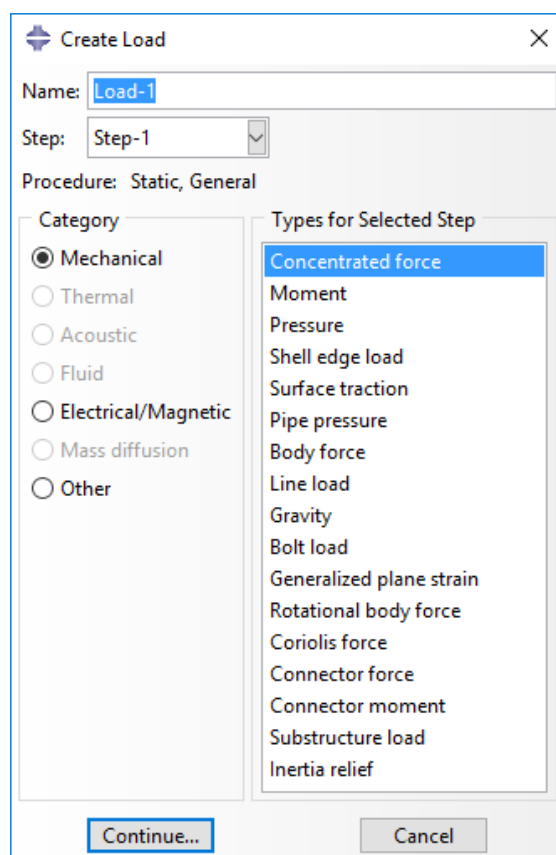


Slika 9.7 – Ustvarjanje obremenitvenega koraka

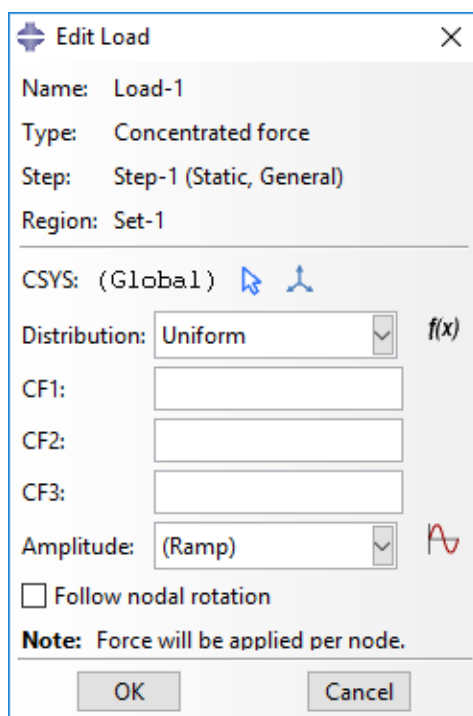
Ko je določen tip numerične analize, lahko v obremenitvenem koraku dodamo obremenitev (angl. *load*) in jo povežemo z geometrijo numeričnega modela. Na sliki 9.8 je prikazano okno za izbor vrste obremenitve. Vidimo, da lahko uporabnik izbira med mehanskimi, toplotnimi, akustičnimi, tekočinskimi, elektromagnetnimi, masno difuzijskimi in drugimi obremenitvami. V okviru mehanskih obremenitev je na voljo širok nabor različne vrste obremenitev. Ko izberemo eno od obremenitev (npr. točkovno silo), je treba najprej izbrati geometrijske elemente, na katere bo obremenitev delovala, nato pa je v dodatnem oknu, kot npr. na sliki 9.9 za točkovno obremenitev, treba vnesti parametre obremenitve.

V primeru točkovne obremenitve je treba izbrati koordinatni sistem. Nato lahko uporabnik izbere obliko porazdelitve, ki je lahko enakomerna (angl. *uniform*), kot na sliki 9.9, ali podana v obliki analitičnega polja. Sledijo komponente obremenitve. V primeru statične obremenitve je amplituda samo ena, v primeru drugačnih vrst obremenitve pa lahko obremenitve določimo tako, kot je za predvideno vrsto simulacije potrebno. Način podajanja obremenitev se seveda lahko precej razlikuje za različne vrste obremenitev. Kljub temu je določanje obremenitev v programu ABAQUS dovolj enostavno, hkrati pa omogoča prilagodljivo določitev obremenitev za vse podprte vrste analiz.





Slika 9.8 – Dodajanje obremenitve – izbor vrste obremenitve



Slika 9.9 – Dodajanje obremenitve – točkovna obremenitev

## 9.5 Literatura

- [9.1] The EN Eurocodes [splet], Dosegljivo: <http://eurocodes.jrc.ec.europa.eu/>. [Datum dostopa 5. 12. 2017].
- [9.2] Linear Elasticity: Singular Solutions for the Half-Space [splet], Dosegljivo: <https://www.brown.edu/Departments/Engineering/Courses/EN224/halfspace/halfspace.html>. [Datum dostopa 5. 12. 2017].
- [9.3] Saint-Venant's Principle [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Saint-Venant%27s\\_Principle](https://en.wikipedia.org/wiki/Saint-Venant%27s_Principle). [Datum dostopa 5. 12. 2017].
- [9.4] T. K. Hellen, S. J. Protheroe, "The BERSAFE finite element system," *Computer-Aided Design*, let. 6, št. 1, str. 15–24, 1974.
- [9.5] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 6. izdaja. Burlington: Elsevier Butterworth-Heinemann, 2005.

## 10. Mreženje s končnimi elementi

Postopek prostorske razdelitve geometrijskega modela na končne elemente imenujemo diskretizacija. Končna numerična rešitev je odvisna od kvalitete in velikosti končnega elementa. Težava je v tem, da ne vemo vnaprej, kakšen končni element je najprimernejši za specifičen problem. V idealnem primeru interpolacijska funkcija končnega elementa sovпада z globalno funkcijo deformacije, vendar je to izjemno redko.

Priprava mreže končnih elementov je danes lahko že popolnoma avtomatizirana, vendar je rezultat pogosto nezadovoljiv. Zato mora uporabnik skrbeti, da je mreža končnih elementov dovolj kvalitetna in zadostne gostote za želeno natančnost rezultatov. Ob geometrijskem modeliranju je mreženje običajno najbolj delovno intenziven del gradnje numeričnega modela. Velikokrat je mreženje tudi zelo podobno geometrijskemu modeliranju, saj je treba za kvalitetno mrežo končnih elementov kompleksen geometrijski model deliti na bolj enostavne geometrijske oblike. Pri mreženju se delo razlikuje v odvisnosti od dimenzije modela. Priprava mreže je nekoliko drugačna za linijske, površine in volumske geometrijske elemente.

### 10.1 Uvoz geometrije modela

V programu za generacijo mreže oz. predprocesorja lahko navadno modeliramo tudi geometrijo problema. Zelo pogosto je predmet numerične analize objekt, ki je bil predhodno geometrijsko modeliran v 3D-modelirniku za potrebe konstruiranja in izdelave tehniške dokumentacije. Slednjega lahko brez težav uvozimo v ustrezen predprocesorski program za pripravo numeričnega modela.

Najbolj primeren format za uvoz podatkov je standard STEP (ISO 10303) [10.1], ki omogoča najbolj celovit prenos geometrije. STEP-datoteka ima običajno končnico `.stp`, iz katere pa ne moremo sklepati, kaj datoteka vsebuje. Čeprav je STEP-format standardiziran, se lahko zgodi tudi, da je zapis v drugačni verziji standarda, ki ga program ne razume, ali pa se pri branju datoteke pojavijo napake.

Pogosto lahko geometrijski model uvozimo tudi v formatih modelirniških knjižnic, kot sta ACIS (s končnico `.sat`) in Parasolid (s končnico `.x_t`), ki je običajno primeren za izdelavo mreže končnih elementov.

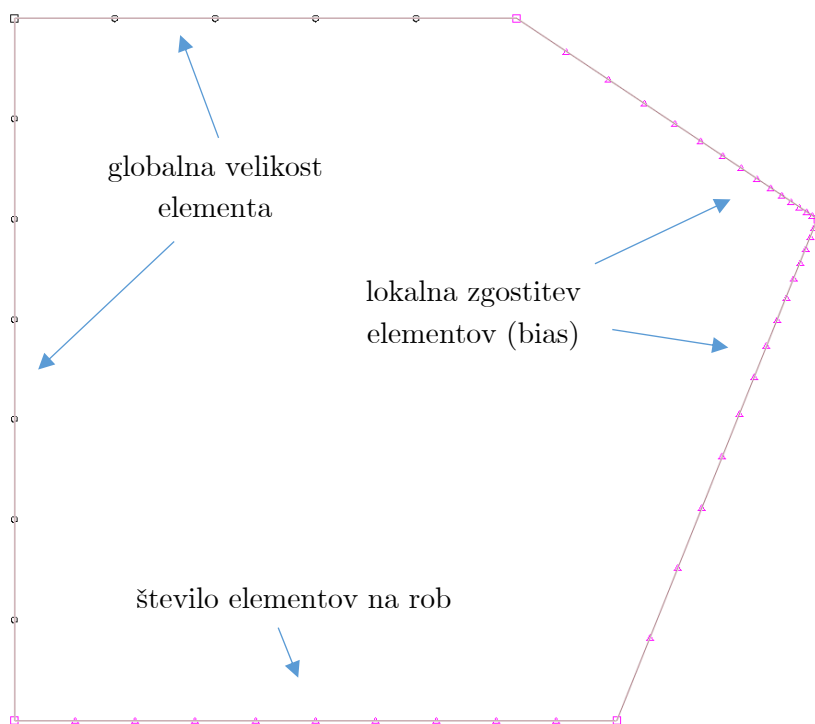
Geometrijski model lahko uvozimo tudi v obliki standarda IGES, s katerim lahko precej dobro opišemo linije in površine, veliko slabše pa prostorsko geometrijo. To je posledica dejstva, da je IGES že zelo star format in ne podpira prostorske geometrije.

Programi za generiranje mreže končnih elementov potrebujejo zapis natančne geometrije, kar pomeni, da mora biti z enačbo površine določena poljubna točka geometrije. To omogočajo le zapisi geometrije s parametričnimi površinami (npr. NURBS [10.2]). Program lahko skuša ustvariti natančen geometrijski model, če takšen zapis geometrije ni na voljo v uvoženem modelu. Ustvarjanje natančnega modela je lahko uspešno ali neuspešno, v zadnjem primeru je treba geometrijo ponovno izdelati.

Včasih se lahko tudi zgodi, da se napake pojavijo pri uvozu geometrije ali pa je uvožen model neuporaben za naše potrebe. Velikokrat se težave pojavijo, ko na videz ustrezen model ne zadošča za izvedbo numerične analize. Tako lahko prikažemo prostorski model izdelka v datoteki s končnico .STL, vendar poligonska oblika zapisa površine ne omogoča generacije mreže in pogosto tudi ni mogoče pretvoriti takšnega modela v natančno geometrijsko obliko.

## 10.2 Gostota mreže končnih elementov

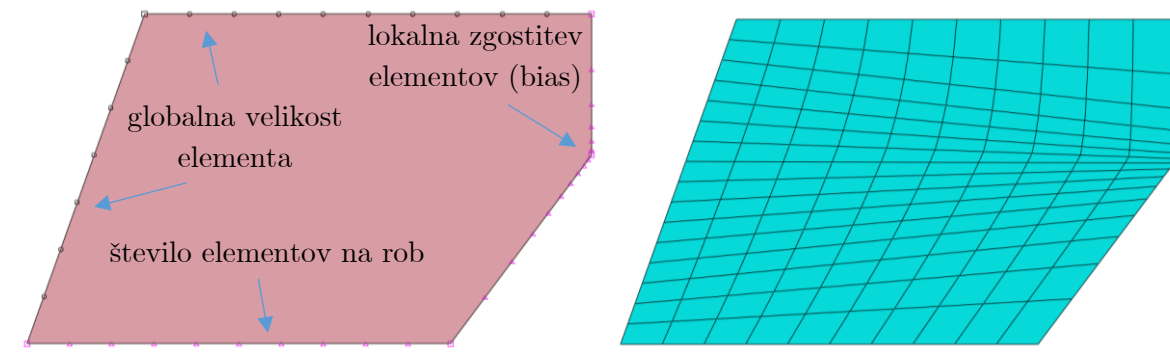
Verjetno najenostavnejše mreženje je mreženje po robovih, kjer določamo, koliko in kako bodo linijski elementi (angl. *truss*) razporejeni po robu. Pri mreženju sta na voljo dve osnovni možnosti: globalna velikost posameznega elementa ali število elementov, razporejenih na posamezen rob. Dodatna možnost je še lokalna zgostitev mreže (angl. *bias*), pri kateri se, na določenem mestu na robu, velikost elementov zmanjšuje s predpisanim faktorjem. Vse tri načine določanja velikosti končnega elementa lahko vidimo na sliki 10.1.



Slika 10.1 – Mreža linijskih končnih elementov

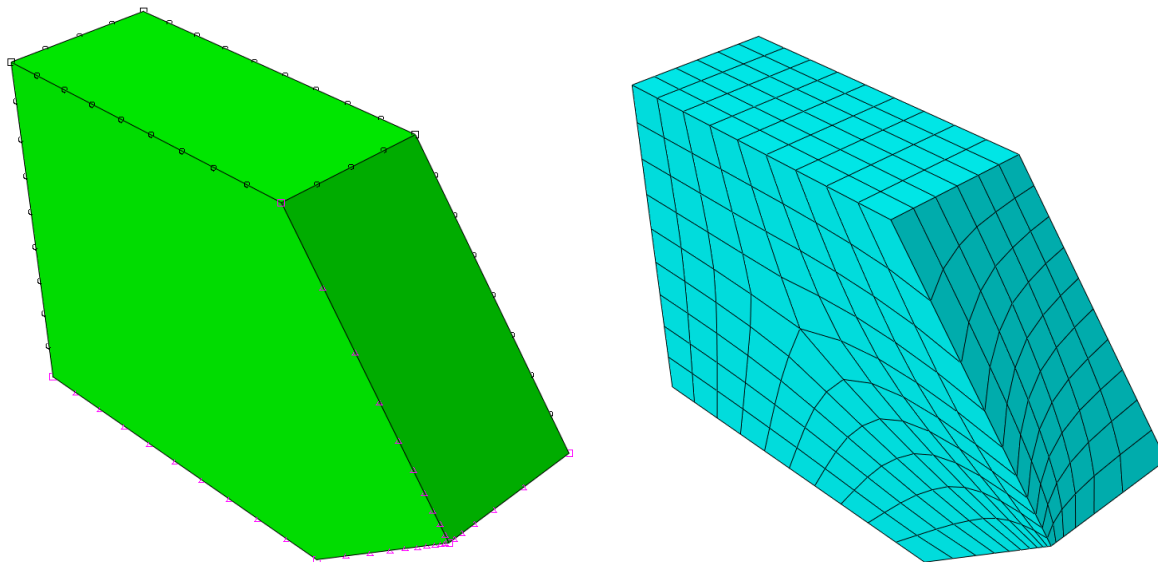
S predpisano globalno velikostjo elementov v dolžinskih enotah dosežemo diskretizacijo z elementi točno določene velikosti. Glede na geometrijo program generira elemente tako, da so vsi približno tako veliki, kot je predpisana velikost. Razporeditev določenega števila elementov na rob pa pomeni, da se celotna dolžina roba razdeli, velikost posameznega elementa pa je odvisna od dolžine roba in želenega števila elementov na rob. Zgoščevanje pomeni, da elementi spremenijo velikost v določeni smeri z želenim faktorjem. V primeru paličnih elementov so potem končni elementi natančno takšni, kot so predpisani z delitvijo.

Predpisana velikost elementov na robu za ravninske elemente pomeni velikost stranice končnega elementa na posameznem robu. Končna mreža pa je kombinacija predpisanih globalnih in lokalnih velikosti elementa. Na sliki 10.2 je na levi prikazan predpis velikosti elementa, na desni pa končna mreža ravninskih končnih elementov.



Slika 10.2 – Mreža ravninskih končnih elementov

Podobno lahko velikost končnih elementov določamo tudi za prostorske končne elemente. Na sliki 10.3 je določena globalna velikost elementov, število elementov na spodnji rob in zgostitev v treh robovih proti enemu oglišču telesa.

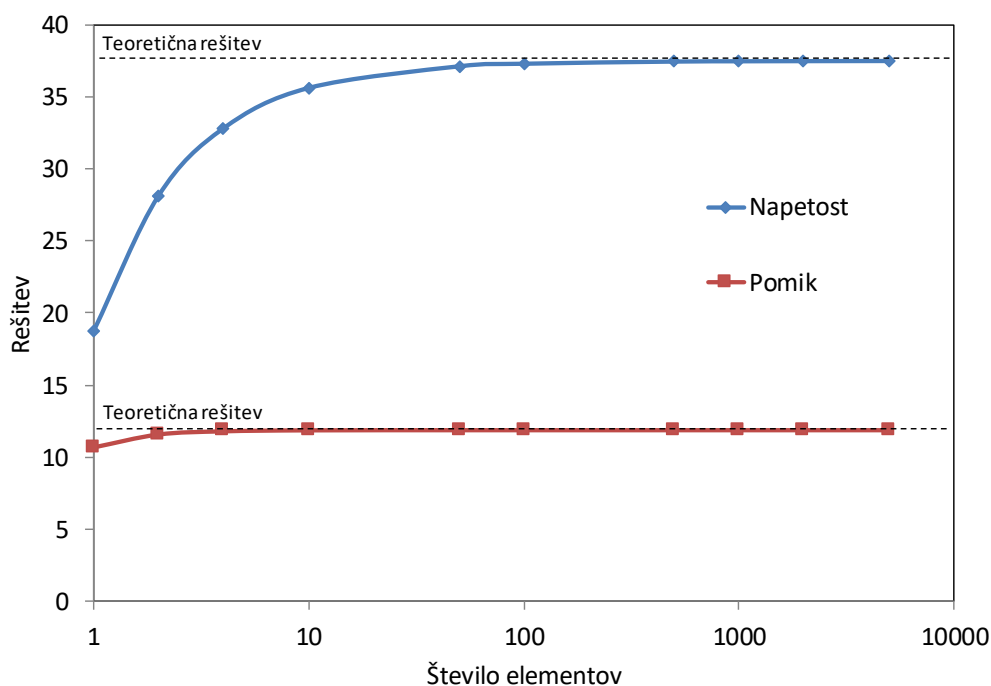


Slika 10.3 – Mreža prostorskih končnih elementov

Velikost končnih elementov je pri začetnih analizah običajno velika (število končnih elementov je majhno), nato pa velikost končnih elementov zmanjšujemo v območjih večjih gradientov rezultatov, s čimer povečujemo število končnih elementov, vse dokler ne dosežemo ustrezne konvergence rezultatov.

Zmanjševanje velikosti elementov je seveda omejeno z računalniško strojno opremo. Včasih je lahko velikost končnega elementa manjša od numerične natančnosti (npr. pri mikroskopskih modelih). Če je natančnost zapisa števil pri običajni natančnosti približno sedem decimalnih mest in je element tako majhen, da imata dve vozlišči isto koordinato, bo nemogoče izdelati takšen končen element. To se zgodi zelo redko, pogosteje število končnih elementov in z njimi povezani podatki presegajo količino vgrajenega pomnilnika. V določenih primerih pomaga virtualni pomnilnik, ki za dodaten pomnilnik uporablja kar prostor na trdem disku. Analiza potem deluje, vendar je zelo počasna. Če količina podatkov presega fizični in virtualni pomnilnik, zelo verjetno sploh ne bo mogoče izdelati mreže končnih elementov, analiza pa prav tako ne bo mogoča. Navadno so mreže s tako velikim številom končnih elementov povezane z ekstremno velikimi in kompleksnimi modeli, za katere potrebujemo superračunalnike. Večino drugih problemov lahko rešimo z zmanjševanjem kompleksnosti modela in seveda z gostitvijo mreže le do smiselne meje.

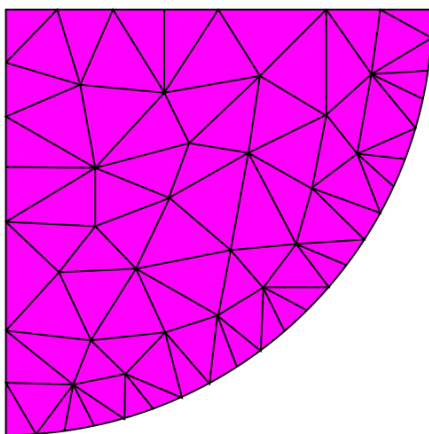
Na sliki 10.4 je prikazana konvergenčna analiza, kjer za isti objekt uporabimo več mrež končnih elementov z različno gostoto. Z večanjem števila končnih elementov rezultati konvergirajo k teoretičnim rešitvam. Pri redkih mrežah s premajhnim številom elementov je napaka relativno velika in zgostitev mreže zelo pozitivno vpliva na kvaliteto rezultatov. Razvidno je, da se pri manj gostih mrežah vrednosti pomikov, v primerjavi z rezultati napetosti, mnogo hitreje približajo pravilnim rezultatom. Med numeričnimi modeli z večjim številom elementov je zelo majhna razlika v natančnosti. Rezultat je skoraj enak, čeprav število elementov narašča. Nadaljnja gostitev mreže ima lahko celo negativne rezultate, saj se lahko poveča numerična napaka zaradi končne aritmetike računalnika.



Slika 10.4 – Konvergenčna analiza numeričnih rezultatov

### 10.3 Samodejno mreženje s končnimi elementi

Večina programov za predprocesiranje samodejno generira mrežo končnih elementov. Takšna mreža je lahko nestrukturirana, strukturirana ali hibridna [10.3]. Za poljuben geometrijski model je mogoče vedno generirati nestrukturirano mrežo končnih elementov [10.4]. Nestrukturirano mrežo običajno sestavljajo trikotniki ali tetraedri in temelji na algoritmu Jima Ruperta [10.5], ki je izdelal algoritem šele v začetku devetdesetih let prejšnjega stoletja. Pred tem je bila generacija mrež vedno ročna ali delno samodejna z modeliranjem mreže končnih elementov. Na sliki 10.5 je prikazana nestrukturirana mreža trikotnih končnih elementov.

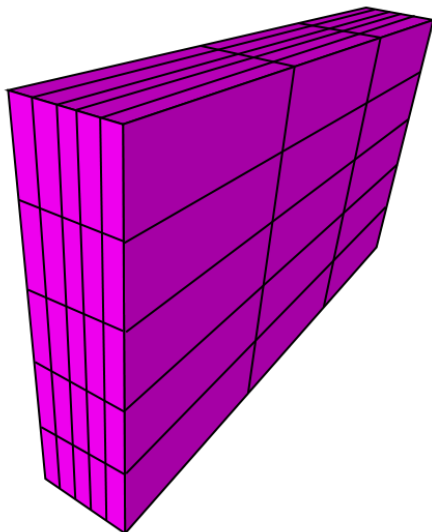


Slika 10.5 – Nestrukturirana 2D-mreža [10.4]

Z nestrukturirano mrežo lahko zamrežimo poljubno geometrijo, vendar je lahko rezultat mreža zelo slabe kvalitete. Že v osnovi končni elementi v obliki trikotnika oz. tetraedra dajejo manj natančne rezultate analiz kot štirikotni elementi. Pogosto je to posledica razvoja končnega elementa, kjer trikotni končni element ni posebej razvit, ampak je zgolj kolapsiran četverokotnik, pri katerem sta dve oglišči združeni v eno. Pri paraboličnih ali kubičnih končnih elementih je seveda v eno vozlišče združenih še več vozlišč. Poleg tega so tudi trikotniki različnih oblik, kot je razvidno na sliki 10.5. Zato je nestrukturirana mreža primerna za začetne analize ali za mreženje izredno kompleksnih modelov, ko nimamo časa izdelati kvalitetnejše mreže.

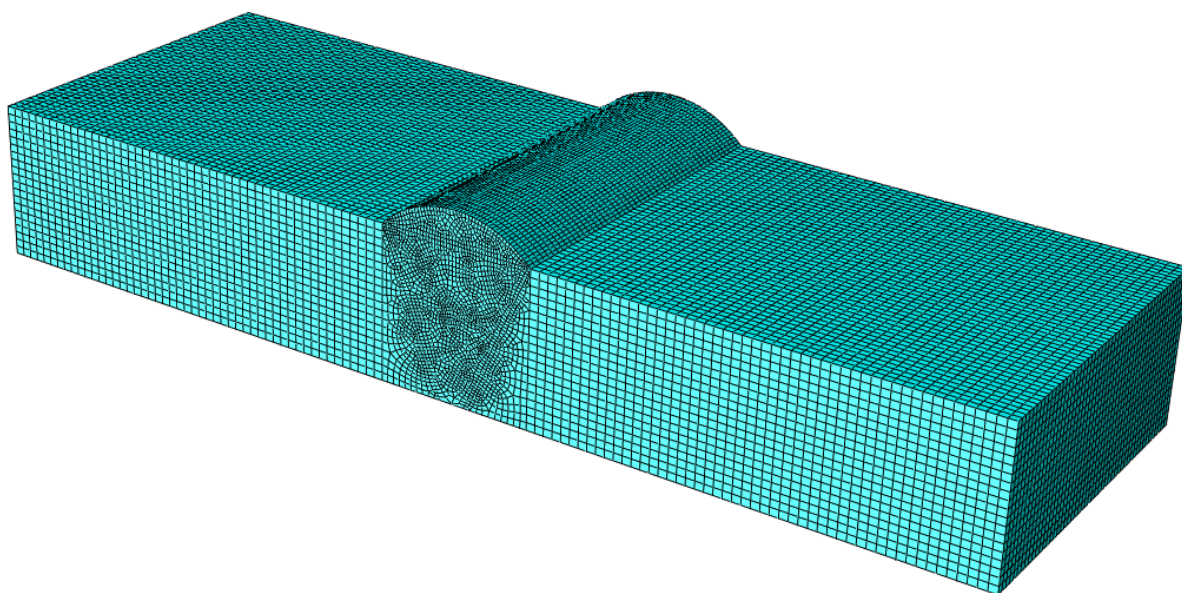
V treh dimenzijah dosežemo najnatančnejše rezultate s strukturirano mrežo, sestavljeno iz štirikotnikov oz. heksaedrov v [10.6]. Strukturirana mreža heksaedrskih končnih elementov je prikazana na sliki 10.6. Idealno mrežo bi dobili, če bi lahko izdelali kartezijevo mrežo, kjer so vsi elementi kocke enake velikosti, saj je osnovni končni element v naravnih koordinatah idealne oblike kocke. Vsako odstopanje od idealne oblike povečuje napako pri numeričnem rezultatu. V splošnem strukturirane mreže končnih elementov dajejo veliko boljše rezultate analiz kot nestrukturirane mreže, saj pri njih elementi manj odstopajo od idealnih oblik. Težava je v tem, da je mogoče strukturirane mreže samodejno ustvariti le za relativno

enostavne geometrijske oblike. Vsak bolj kompleksen geometrijski model je treba geometrijsko razdeliti na enostavne oblike, da lahko tvorimo strukturirano mrežo končnih elementov.



Slika 10.6 – Strukturirana 3D-mreža [10.6]

Deljenje modela je izredno zahtevno opravilo in zato je mreženje, kljub samodejni generaciji, še vedno časovno najbolj intenziven proces v fazi predprocesiranja. Zato so numerični modeli pogosto sestavljeni iz hibridnih mrež, kot je mreža prikazana na sliki 10.7.



Slika 10.7 – Hibridna mreža

Del geometrije, ki ga je možno razdeliti na osnovna geometrijska telesa je strukturirano zamrežen s končnimi elementi, geometrijsko bolj kompleksen del pa nestrukturirano. V

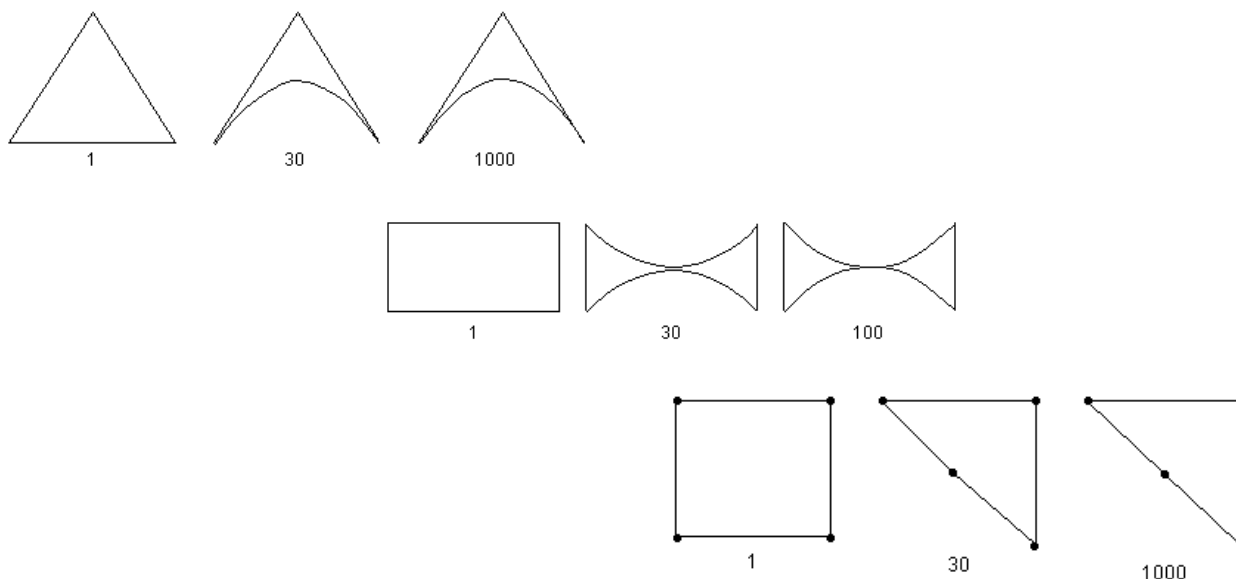


odvisnosti od sposobnosti predprocesorja in kompleksnosti geometrije je nestrukturiran del lahko sestavljen iz heksaedrov, kombinacije heksaedrov in tetraedrov ali samo iz tetraedrov.

#### 10.4 Kvaliteta končnih elementov

Končni element je definiran v lokalnem naravnem koordinatnem sistemu in celotna teoretična izpeljava je izvedena za končni element idealne oblike. Idealni končni elementi so enakostranični trikotniki, kvadrati in kocke. V realnosti pa ne moremo vseh modelov zamrežiti z elementi idealnih oblik. Zaradi prilagajanja dejanski geometrijski obliki lahko posamezen končni element precej odstopa od idealne oblike. Zaradi razlike v obliki končnega elementa se razlikujejo tudi rezultati. Natančne rezultate je mogoče pričakovati le v primeru idealne oblike končnega elementa. V skrajnih primerih je napaka tako velika, da rezultati niso več uporabni; v takšnih primerih običajno že sami programi za generacijo mreže preprečijo uporabo mreže, kjer so končni elementi preveč popačeni. Preslikava iz lokalnega koordinatnega sistema v realni koordinatni sistem modela z uporabo oblikovnih funkcij, ki so v primeru izoparametričnih končnih elementov priročno enake interpolacijskim funkcijam, je opisana v poglavju 3.8.

Če izračunamo determinanto Jakobijeve matrike končnega elementa [10.7][10.8], dobimo vrednost, ki pokaže, kako popačen je končni element. Večja kot je vrednost determinante, bolj popačena je oblika končnega elementa v dejanskem modelu glede na idealno obliko končnega elementa. Na sliki 10.8 so prikazane oblike končnih elementov in pripadajoče vrednosti determinant Jakobijevih matrik.



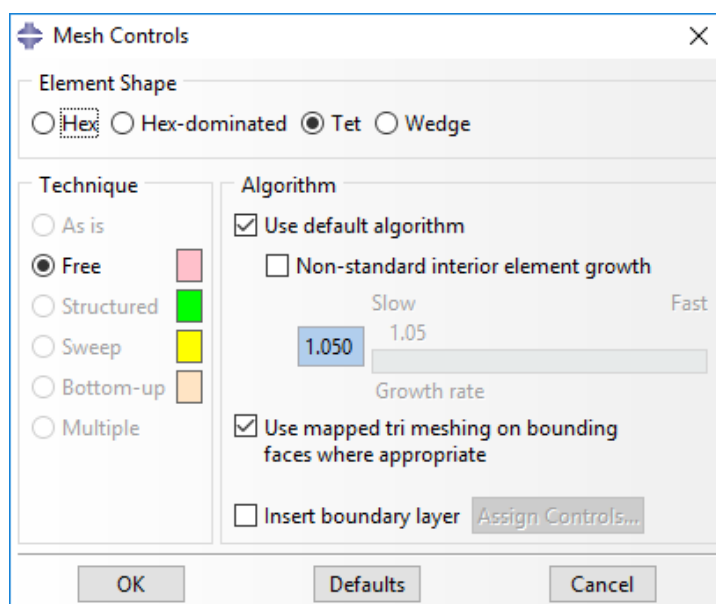
Slika 10.8 – Oblike končnih elementov in vrednosti determinant Jakobijeve matrike

Poleg tega programi za pripravo mrež končnih elementov kontrolirajo še posamezne geometrijske lastnosti končnega elementa, kot so zašiljenost, velikost vogalnih notranjih kotov, vzporednost stranic, zvitost, razmerje stranic itd., katerih dopustna razmerja lahko posebej nastavimo.

## 10.5 Mreženje v programu ABAQUS

Tako kot v večini programih za pred- in poprosesiranje je tudi v programu ABAQUS priprava mreže končnih elementov samodejna. Popolni avtomatizem navadno velja za mreže, kjer so končni elementi tetraedri ali trikotniki v ravnini. Pogosto želimo mreže s pravokotnimi elementi v ravnini in heksaedri v prostoru, saj ti končni elementi dajejo veliko boljše rezultate. Samodejna priprava takšne mreže deluje le na dovolj enostavnih geometrijah.

V programu ABAQUS sta kompleksnost modela in primernost generacije mreže priročno opredeljeni z obarvanostjo modela. Na sliki 10.9 je prikazano okno za izbor oblike elementa za kompleksen model, kjer je mogoče generirati mrežo končnih elementov le s tetraedri (angl. *free*). V tem primeru je celotna geometrija modela obarvana z roza barvo. Pri bolj enostavni geometriji bi bila barva modela rumena, kar pomeni, da je mogoče izdelati mrežo, ki bo sestavljena pretežno s heksaedri, vendar teh elementov ne bo mogoče enakomerno razporediti (angl. *sweep*). Enostavne geometrije (npr. kvader) je mogoče mrežiti strukturirano s heksaedri in takrat bodo modeli obarvani z zeleno barvo (angl. *structured*). Model, obarvan z oranžno barvo (angl. *bottom-up*), pa zahteva delno samodejno mreženje, kjer uporabnik omogoča, da mreža končnih elementov nekoliko odstopa od dejanske geometrije modela z namenom uspešne generacije mreže.



Slika 10.9 – Deljenje in popravljanje modela

Pogosto je enostavneje razdeliti kompleksen realni model v enostavne celice, ki jih je mogoče samodejno zamrežiti z delno strukturirano (angl. *sweep*) ali s strukturirano mrežo. Na sliki 10.10 so prikazane ikone programa ABAQUS, ki omogočajo različne načine deljenja in popravljanja geometrije numeričnega modela. V ta namen je treba uporabiti različne metode konstruiranja z računalnikom in deliti model po obstoječih robovih in ravninah ali narediti nove robove in ravnine, ki bodo model razdelile v enostavne geometrijske celice. To delo je lahko zelo dolgotrajno in zahteva precej izkušenj uporabnika.



Slika 10.10 – Deljenje in popravljanje modela

Na sliki 10.11 so prikazani ukazi za mreženje, ki omogočajo določanje globalne ali lokalne velikosti končnega elementa, mreženje celotnega modela ali mreženje posameznih delov modela, izbor oblike končnega elementa, delo z mreženjem približne geometrije (angl. *bottom-up*), izbor vrste končnega elementa v mreži vključno z določanjem reda polinoma oblikovne funkcije končnega elementa, kontrolo kvalitete končnih elementov in določanje posameznih plasti končnih elementov ter pretvorbo v površinski ali linijski model.



Slika 10.11 – Mreženje

## 10.6 Literatura

- [10.1] ISO 10303 [splet], Dosegljivo: [https://en.wikipedia.org/wiki/ISO\\_10303](https://en.wikipedia.org/wiki/ISO_10303). [Datum dostopa 5. 12. 2017].
- [10.2] Non-uniform rational B-spline [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Non-uniform\\_rational\\_B-spline](https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline). [Datum dostopa 5. 12. 2017].
- [10.3] Types of mesh [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Types\\_of\\_mesh](https://en.wikipedia.org/wiki/Types_of_mesh). [Datum dostopa 5. 12. 2017].
- [10.4] Unstructured grid [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Unstructured\\_grid](https://en.wikipedia.org/wiki/Unstructured_grid). [Datum dostopa 5. 12. 2017].
- [10.5] Ruppert's algorithm [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Ruppert%27s\\_algorithm](https://en.wikipedia.org/wiki/Ruppert%27s_algorithm). [Datum dostopa 5. 12. 2017].
- [10.6] Regular grid [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Regular\\_grid](https://en.wikipedia.org/wiki/Regular_grid). [Datum dostopa 5. 12. 2017].
- [10.7] Jacobian matrix and determinant [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Jacobian\\_matrix\\_and\\_determinant](https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant). [Datum dostopa 5. 12. 2017].
- [10.8] Jacobian Determinant [splet], Dosegljivo: <https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/jacobian/v/the-jacobian-determinant>. [Datum dostopa 5. 12. 2017].
- [10.9] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 6. izdaja. Burlington: Elsevier Butterworth-Heinemann, 2005.

## 11. Reševanje sistema enačb

V metodi končnih elementov je računalniško najbolj intenziven del računanje neznanih vrednosti numeričnega modela. Modul za računanje se imenuje reševalec (angl. *solver*), ki vsebuje različne algoritme za reševanje numeričnih modelov.

Pri preračunu statičnih problemov tako rešujemo zgolj sistem linearnih algebrskih enačb. Razvitih je bilo veliko metod za reševanje sistema linearnih enačb, ki jih lahko ločimo v dve glavni skupini: direktne in iterativne metode [11.1]. Pri tem vse metode upoštevajo specifično metode končnih elementov, kjer je sistem enačb zapisan s togostno matriko sistema in vektorjem zunanjih obremenitev. Značilnosti togostne matrike so pozitivna definitnost, simetričnost in pasovna organiziranost [11.2], kjer le pas ob diagonali matrike vsebuje vrednosti, različne od nič. Pri pasovni matriki, prikazani na sliki 11.1, je predvsem pomembna največja širina pasa vrednosti, različnih od nič. Največja širina (oz. zaradi simetričnosti vrednosti zgolj polširina) in število neznank (število vrst matrike) določata količino računalniškega pomnilnika, potrebnega za shranjevanje sistema enačb.

$$\begin{bmatrix} B_{11} & B_{12} & 0 & \cdots & \cdots & 0 \\ B_{21} & B_{22} & B_{23} & \ddots & \ddots & \vdots \\ 0 & B_{32} & B_{33} & B_{34} & \ddots & \vdots \\ \vdots & \ddots & B_{43} & B_{44} & B_{45} & 0 \\ \vdots & \ddots & \ddots & B_{54} & B_{55} & B_{56} \\ 0 & \cdots & \cdots & 0 & B_{65} & B_{66} \end{bmatrix}$$

Slika 11.1 – Pasovna matrika [11.2]

Reševanje sistema enačb je namreč časovno učinkovito le, kadar so vse vrednosti shranjene v računalniškem pomnilniku in ni treba brati in zapisovati iz počasnih zunanjih medijev (npr. trdih diskov). Metoda končnih elementov omogoča reševanje problemov z nekaj 10 milijoni neznank. V preglednici 11.1 je podan okvirni pregled potrebne količine pomnilnika in hitrosti računanja na različnih računalniških platformah za reševanje polnih matrik reda  $N$ , medtem ko preglednica 11.2 podaja enake podatke za reševanje pasovnih matrik s širino  $B = \sqrt{N}$ , ki so značilne za metodo končnih elementov. Podatki so povzeti po viru [11.3], kjer so navedeni časi računanja sicer precej večji, v preglednici pa smo upoštevali, da lahko zmogljivi računalniki danes računajo s hitrostjo GFLOPS, superračunalniki pa TFLOPS. Navedeni računski časi zajemajo samo dejansko procesiranje podatkov v centralni procesni enoti. Če pa upoštevamo še branje ter zapisovanje podatkov, so dejanski časi analiz lahko še precej daljši od časov, navedenih v preglednicah.

Navedeni računski časi upoštevajo, da so vsi podatki shranjeni v hitrem pomnilniku računalnika. Pomnilnik velikosti nekaj deset GB je na zmogljivih osebnih računalnikih dosegljiv, medtem ko je pomnilnik z 8 TB dandanes praktično nedosegljiv.

Preglednica 11.1 – Potreben pomnilnik in hitrost reševanja polnih matrik [11.3]

Red matrike $N$	Potreben pomnilnik (64-bitna vred.) $= 8N^2$ bajtov	Potrebno število računskih operacij [FLOPS] $= N^3/6$	Čas reševanja na zmogljivem osebнем računalniku	Čas reševanja na superračunalniku $u$
$10^4$	800 MB	$10^{12}/6$	3 min	2 s
$10^5$	80 GB	$10^{15}/6$	46 ur	30 min
$10^6$	8 TB	$10^{18}/6$	5 let	21 dni

V primeru pasovne matrike so računski časi bistveno krajši, saj je treba shranjevati manjše število členov matrike in posledično izvršiti manjše število računskih operacij kot pri polnih matrikah, preglednica 11.2. Izvorna preglednica, ki je bila izdelana pred dvajsetimi leti, je nekoliko razširjena in kaže stanje v današnjem času.

Preglednica 11.2 – Potreben pomnilnik in hitrost reševanja pasovnih matrik s širino  $B = \sqrt{N}$  [11.3]

Red matrike $N$	Potreben pomnilnik (64-bitna vred.) $= 8NB$ bajtov	Potrebno število računskih operacij [FLOPS] $= N^2/2$	Čas reševanja na zmogljivem osebнем računalniku	Čas reševanja na superračunalniku $u$
$10^4$	8 MB	$10^8/2$	0,05 s	0,0005 s
$10^5$	240 MB	$10^{10}/2$	5 s	0,05 s
$10^6$	8 GB	$10^{12}/2$	8 min	5 s
$10^7$	250 GB	$10^{14}/2$	13 ur	9 min
$10^8$	8TB	$10^{16}/2$	58 dni	16 ur

Zaradi splošne nedostopnosti računalnikov z nekaj TB pomnilnika so analize z matrikami velikosti nad 10 milijonov prostostnih stopenj v praksi zelo redke, čeprav se količine razpoložljivih hitrih pomnilnikov stalno povečujejo.

Mnogokrat pa ne rešujemo zgolj enostavnih statičnih problemov, ampak tudi časovno odvisne pojave, kjer je potrebna tudi integracija vodilnih enačb v času, ki je običajno izvedena s časovno koračno shemo. To v praksi pomeni, da v vsakem časovnem koraku vedno znova rešujemo sistem enačb, saj se lahko tako obremenitve in podpore kot tudi lastnosti modela s časom spreminjajo. Zato je zelo koristno, da pri metodi končnih elementov rešujemo sisteme s pasovnimi in simetričnimi matrikami, kar bistveno pospeši računske čase.

## 11.1 Direktno reševanje sistema enačb

Direktno reševanje sistema enačb označuje metode, pri katerih neposredno izračunamo rešitve iz danega sistema. Natančnost izračunanih rešitev je odvisna od natančnosti računanja in v idealnem primeru dobimo idealno natančne rešitve. V realnosti pa računamo na nekaj decimalnih mest natančno, zato se zgodi, da se pri računanju zaradi zaokroževanja pojavljajo napake, ki jih lahko tudi izračunamo. Zato pogosto rezultate sistema enačb spremlja podatek o napaki zaradi zaokroževanja. Na to napako ne moremo vplivati drugače, kot da računamo z večjim številom decimalnih mest, kar običajno pomeni tudi daljše računske čase.

Večina direktnih metod temelji na izpeljavi in optimizaciji Gaussove eliminacije [11.4]. Metoda se imenuje po nemškem matematiku Gaussu, čeprav so jo uporabljali na Kitajskem že vsaj leta 179 pred našim štetjem. Metoda temelji na postopku, v okviru katerega najprej v vrsticah matrike eliminiramo neznanke, nato pa izračunamo vrednosti neznank. Na ta način so vse neznanke zapisane le v prvi enačbi, v vsaki nadaljnji vrstici pa je ena neznanka manj. V zadnji vrstici sistema enačb na koncu ostane le ena neznanka, ki jo lahko preprosto izračunamo.

Celoten postopek je običajno razdeljen na dva dela. To sta:

1. eliminacija neznank in
2. določanje vrednosti neznank.

Po pripravi numeričnega modela v metodi končnih elementov je treba izračunati sistem enačb, ki ga lahko zapišemo v obliki enačbe 11.1, kjer so z  $x$  označene neznanke, z  $a$  in  $b$  pa so konstante v sistemu enačb.

$$\begin{array}{rcccccc} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n & & & & & b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n & = & & & & b_2 \\ \vdots & & \vdots & & \ddots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \dots + a_{nn}x_n & & & & & b_n \end{array} \quad (11.1)$$

V prvem koraku začnemo v prvi vrstici sistema enačb in iz vseh drugih vrstic eliminiramo prvo neznanko. Prvo enačbo pustimo enako in nato preoblikujemo vse druge enačbe, kot je prikazano v enačbi (11.2).

$$\begin{array}{rcccccc} \left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \dots + \left(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\right)x_n & & & & & b_2 - \frac{a_{21}}{a_{11}}b_1 \\ \left(a_{32} - \frac{a_{31}}{a_{11}}a_{12}\right)x_2 + \dots + \left(a_{3n} - \frac{a_{31}}{a_{11}}a_{1n}\right)x_n & = & & & & b_3 - \frac{a_{31}}{a_{11}}b_1 \\ \vdots & & \vdots & & \vdots & \vdots \\ \left(a_{n2} - \frac{a_{n1}}{a_{11}}a_{12}\right)x_2 + \dots + \left(a_{nn} - \frac{a_{n1}}{a_{11}}a_{1n}\right)x_n & & & & & b_n - \frac{a_{n1}}{a_{11}}b_1 \end{array} \quad (11.2)$$

Nato je treba postopek ponoviti tako, da ostane druga vrstica nespremenjena in odštejemo drugi člen pri vseh vrsticah navzdol. Cel postopek lahko zapišemo z algoritmom 11.1.

```

for j = 1, 2 ... n - 1 do
  for i = j + 1, j + 2 ... n do
    mij = aij / ajj
    for k = j + 1, j + 2 ... n do
      aik = aik - mijajk
    end
    bi = bi - mijbj
  end
end
end

```

Algoritem 11.1 – Eliminacija neznank [11.5]

Rezultat je trikotna matrika (11.3), kjer je v vsaki naslednji vrstici ena neznanka manj, vse do zadnje vrstice, kjer imamo le eno neznanko.

$$\begin{array}{rcl}
 a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n & b_1 & \\
 a'_{22}x_2 + a'_{23}x_3 + \dots + a'_{2n}x_n & b'_2 & \\
 \vdots & \vdots & \\
 a'_{nn}x_n & b'_n & 
 \end{array} = \begin{array}{l} b_1 \\ b'_2 \\ \vdots \\ b'_n \end{array} \quad (11.3)$$

Iz zadnje vrstice preprosto izračunamo neznanko  $x_n$  in potem to uporabimo v predzadnji vrstici in izračunamo  $x_{n-1}$ . Tako lahko z uporabo enačbe (11.4) izračunamo vse neznanke sistema. Ta korak se imenuje vzvratna substitucija (angl. *backward substitution*).

$$x_n = \frac{b'_n}{a'_{nn}} \quad (11.4)$$

$$x_i = \frac{b'_i - \sum_{k=i+1}^n a'_{ik}x_k}{a'_{ii}}$$

Pogosto pa želimo v metodi končnih elementov le spremeniti stolpec  $b$ , ki predstavlja robne pogoje (npr. zunanje obremenitve), in ne želimo pri eliminaciji spreminjati tega stolpca. Zato se uporablja tudi dekompozicija matrike  $A$  v dve trikotni obliki  $L$  in  $U$  [11.5]. Črka  $L$  predstavlja spodnjo (angl. *lower*) trikotno matriko,  $U$  pa zgornjo (angl. *upper*) trikotno matriko. Potem velja enačba 11.5, kjer so členi matrike  $A$  spremenjeni v člene matrike  $L$  in  $U$ , vektor  $b$  pa ostaja nespremenjen.

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b} \quad (11.5)$$

Če enačbo pretvorimo v dve enačbi in vektor vmesnih rešitev imenujemo  $y$ , lahko zapišemo enačbi (11.6).

$$\mathbf{Ly} = \mathbf{b} \quad \Rightarrow \quad \mathbf{Ux} = \mathbf{y} \quad (11.6)$$

Ko je osnovna matrika razdeljena na  $L$  in  $U$ , lahko preprosto, v dveh korakih z vzvratno substitucijo, dobimo rešitve  $y$  in  $x$ . Reševanje sistema enačb z dekompozicijo  $LU$  je precej



hitrejše od običajne Gaussove eliminacije, poleg tega je reševanje sistema enačb v sodobnih programih še dodatno optimirano in prilagojeno pasovnim matrikam. Zato danes analize potekajo zelo hitro, tako da so možne analize velikih numeričnih modelov.

Direktno reševanje sistema enačb pa ni najbolj primerno za matrike, kjer so nekateri diagonalni členi enaki nič (npr. metoda razširjenih Lagrangeevih multiplikatorjev [11.6]) in ki zahtevajo dodatno manipulacijo sistema enačb. Še večja težava je končna aritmetika v računalnikih, kjer je število decimalnih mest omejeno, kar privede do numeričnih napak zaradi velikega števila množenj in deljenj. Na to numerično napako uporabnik nima vpliva in lahko ugotovi le, kakšna vrsta napake se je pri analizi pojavila. Ta napaka je označena kot napaka zaradi zaokroževanja (angl. *roundoff error*) [11.7].

## 11.2 Iterativno reševanje sistema enačb

Gauss je tudi idejni oče iterativnih metod za reševanje sistema enačb. Osnovna metoda se imenuje Gauss-Siedel [11.8] in temelji na preprosti evalvaciji linearnih enačb (11.7).

$$x_i = \frac{b_i - \sum_{j=1}^n (j \neq i) a_{ij} x_j}{a_{ii}} \quad i = 1, 2 \dots n \quad (11.7)$$

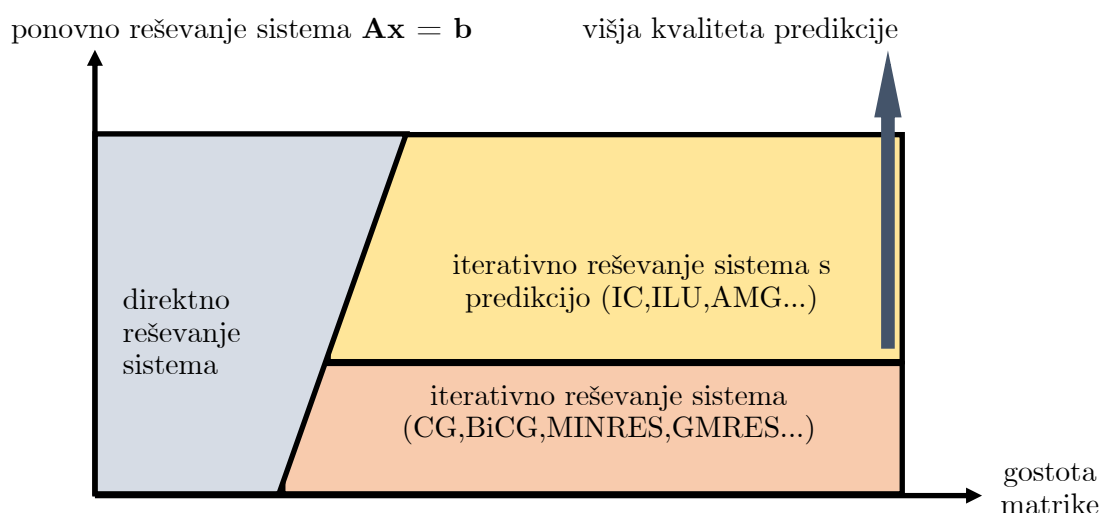
Ker so v enačbah neznanke  $x_j$ , katerih vrednosti niso znane, jih je treba na začetku ugibati. Napoved oz. začetne vrednosti neznank v prvem koraku so pri tem zelo pomembne. V ta namen so razviti precej kompleksni algoritmi. Ko iz posamezne enačbe (11.7) dobimo novo rešitev  $x_i$ , jo lahko uporabimo v naslednjih enačbah za izračun vrednosti drugih spremenljivk. Ko evalviramo vse enačbe v sistemu, preverimo odstopanje rešitev z enačbo (11.8), kjer računamo napako v odstotkih. Pred reševanjem lahko uporabnik določi želeno velikost napake. Ko je napaka dovolj majhna, se reševanje sistema konča.

$$|\epsilon_a|_i = \left| \frac{x_i^{nov} - x_i^{star}}{x_i^{nov}} \right| \cdot 100 \quad (11.8)$$

Tako dobimo rezultate sistema z določenim odstopanjem. Metoda konvergira, če je matrika pozitivno definitna in so diagonalni členi dominantni. Včasih metoda konvergira tudi, če nista izpolnjena omenjena pogoja, lahko pa se zgodi, da v takšnih primerih metoda divergira in se reševanje ne konča.

Razvitih je bilo veliko iterativnih reševalcev sistema enačb in njihov razvoj še vedno poteka. Iterativno reševanje sistema je tudi zelo zanimivo za vzporedno reševanje enačb na več procesorskih jedrih, saj je razdelitev problema na vzporedne procese dokaj preprosta. Kljub temu pa se iterativno reševanje sistema enačb uporablja za specifične probleme. Za običajne preračune v metodi končnih elementov, kjer imamo pasovne matrike, je še vedno bolj primerno direktno reševanje enačb.

Na sliki 11.2 je prikazana uporaba različnih metod za reševanje sistema enačb. Dokler rešujemo pasovne matrike, in večkrat rešujemo sistem enačb za različne vrednosti  $\mathbf{b}$ , so najbolj primerni sistemi za direktno reševanje sistema enačb. Kadar je matrika polna, pa je smiselna uporaba iterativnega reševanja enačb. Za večkratno reševanje polnih matrik so potrebne metode, ki imajo izpopolnjeno napoved začetnih vrednosti rešitev sistema.



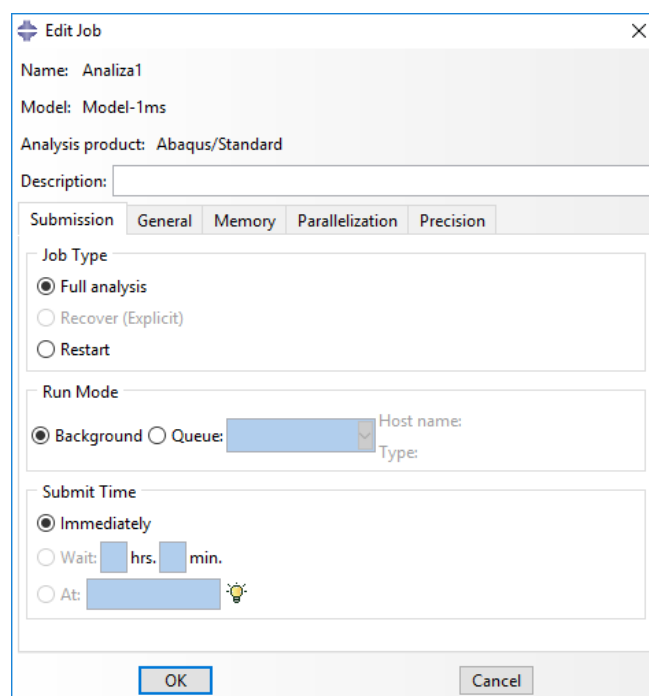
Slika 11.2 – Direktno ali iterativno reševanje sistema

### 11.3 Reševanje sistema enačb v ABAQUS-u

Po pripravi numeričnega modela je treba v ABAQUS-u ustvariti opravilo (angl. *job*), ki ga potem pošljemo v analizo interaktivno ali pa računamo v ozadju. Interaktivno pomeni, da mora ostati okno z grafičnim vmesnikom aktivno in računanje spremljamo z opcijo monitor. V ozadju pa računamo navadno na večjih sistemih, kjer so vsa opravila v vrsti in se izvajajo, ko so na voljo strojne kapacitete za analizo. Za zahtevne analize lahko določimo, da se izvajanje začne kasneje – z zakasnitvijo. Glavno okno za urejanje opravila je prikazano na sliki 11.3.

V zavihkih okna lahko določimo mesta datotek, kamor se bodo shranjevali vmesni in končni rezultati, določimo velikost potrebnega pomnilnika, način vzporednega reševanja in numerično natančnost.

Velikost pomnilnika je odvisna od velikosti problema, saj želi program celoten sistem enačb hraniti v sistemskem pomnilniku. V primeru 32-bitnih operacijskih sistemov je lahko zato največji problem obsegal velikost sistema enačb do 2 GB. Za 64-bitne sisteme pa so zaenkrat numerični modeli premajhni, vendar se lahko zgodi, da problem presega kapaciteto hitrega pomnilnika, ki mora biti zato razširjen s prostorom na trdem disku. To pa pomeni precej daljše čase reševanja sistema, saj je trdi disk precej počasnejši od pomnilnika.



Slika 11.3 – Urejanje opravila v programu ABAQUS

Večina procesorjev ima danes dve ali več jeder, zato je mogoče izbrati, s koliko jeder bo program izvajal analizo. Poleg tega lahko analizo izvajamo tudi na grafičnih karticah, ki imajo tudi več jeder za vzporedno računanje. Če računamo na enem računalniku, običajno izberemo vzporedno izvajanje z uporabo več niti, če pa gre za računalniško gručo z več povezanimi računalniki, pa izberemo način MPI (Message Passing Interface).

Za večino problemov zadostuje navadna natančnost, ko računamo na 7 decimalnih mest natančno. V primeru potrebe po večji natančnosti lahko izberemo dvojno natančnost (angl. *double precision*), kjer računamo na 16 decimalnih mest natančno. Ker imamo danes 64-bitne računalnike in ti računajo s 64-bitnimi števili v enem taktu, to ne pomeni več, da bo dvojna natančnost zahtevala veliko večje računske čase. Ker pa je računanje v enojni natančnosti optimirano, je še vedno hitrejše kot računanje z dvojno natančnostjo.

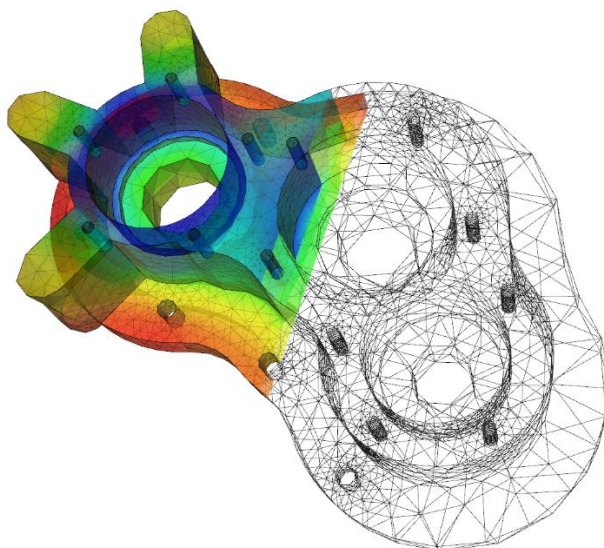
Reševanje sistema poteka z izbranim reševalcem, ki ga uporabnik lahko izbira pri določitvi koraka analize. Običajno gre za direktno reševanje sistema enačb, seveda pa je možno izbrati tudi različne iterativne načine reševanja sistema enačb.

## 11.4 Literatura

- [11.1] M. Schäfer, *Computational Engineering - Introduction to Numerical Methods*. Berlin: Springer-Verlag, 2006.
- [11.2] Band matrix [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Band\\_matrix](https://en.wikipedia.org/wiki/Band_matrix). [Datum dostopa 5. 12. 2017].
- [11.3] Solving FEM Equations [splet], Dosegljivo: <http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d/IFEM.Ch26.d/IFEM.Ch26.pdf>. [Datum dostopa 5. 12. 2017].
- [11.4] Gaussian elimination [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Gaussian\\_elimination](https://en.wikipedia.org/wiki/Gaussian_elimination). [Datum dostopa 5. 12. 2017].
- [11.5] Gaussian Elimination and Back Substitution [splet], Dosegljivo: <http://www.math.usm.edu/lambers/mat610/sum10/lecture4.pdf>. [Datum dostopa 5. 12. 2017].
- [11.6] Augmented Lagrangian method [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Augmented\\_Lagrangian\\_method](https://en.wikipedia.org/wiki/Augmented_Lagrangian_method). [Datum dostopa 5. 12. 2017].
- [11.7] Round-off error [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Round-off\\_error](https://en.wikipedia.org/wiki/Round-off_error). [Datum dostopa 5. 12. 2017].
- [11.8] Gauss–Seidel method [splet], Dosegljivo: [https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel\\_method](https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method). [Datum dostopa 5. 12. 2017].

## 12. Poprocesiranje numeričnih rezultatov

V fazi poprocesiranja analiziramo in interpretiramo rezultate računalniških simulacij ter jih prikažemo na različne možne načine, slika 12.1 [12.1]. Rezultate računalniških simulacij lahko prikažemo v obliki številskih preglednic, diagramov, slik ali dinamičnih animacij, odvisno od proučevanega fizikalnega problema.



Slika 12.1 – Poprocesiranje konstrukcijskega elementa [12.2]

Že pri zasnovi in določevanju parametrov računalniških simulacij je treba premišljeno izbrati izhodne veličine, saj lahko rezultati simulacij predstavljajo zelo obsežne podatke, ki včasih dosegajo velikostni razred terabajtov [12.3]. Zaradi tega je pomembna sposobnost upravljanja z obsežnimi podatkovnimi datotekami in bazami, še posebej v primeru omejenih informacijsko tehnoloških zmoglosti. Ne glede na sposobnosti grafične kartice je intuitiven poprocesor kot modul večjega programskega paketa ali kot samostojni program bistven za interpretacijo in prikaz izračunanih podatkov na zaslonu. Zavedati se moramo, da so rezultati numerične analize le številke v diskretnih točkah modela, v fazi poprocesiranja pa te podatke preoblikujemo in prikažemo v pregledni obliki. Pri tem morajo biti uporabniki poprocesorskih programov sposobni analizirati in vrednotiti pridobljene rezultate na različne načine, npr. v obliki primerjalnih napetosti, glavnih napetosti, deformacij, energij, momentov, reakcijskih sil itd.

### 12.1 Kontrola numeričnega modela in rešitev

Poprocesiranje se začne s temeljitim preverjanjem poteka reševanja numeričnega modela. Večina programskih paketov omogoča sledenje poteku rešitve, kjer je treba biti pozoren na sprotna opozorila ali napake, kljub morebitni zaključeni analizi. Naslednji korak je kontrola reakcijskih sil v podporah. Neujemanje obremenitev in reakcijskih sil v podporah ter posledično neravnovesje sil je osnovni pokazatelj glede ustreznosti drugih rezultatov izvedene analize.

Validacija in kontrola rešitev se pogosto izvedeta s pregledom poteka energij (npr. skupne energije v sistemu, gostote deformacijske energije) in z analizo odstopanja ter razlik v napetosti med sosednjimi elementi [12.4]. Po kontroli osnovnih veličin, ki bi lahko bile napačne zaradi numeričnih napak je smiselno natančno preučiti tiste veličine, ki so bile podane za izvedbo računalniške simulacije.

## 12.2 Številčni rezultati in preglednice

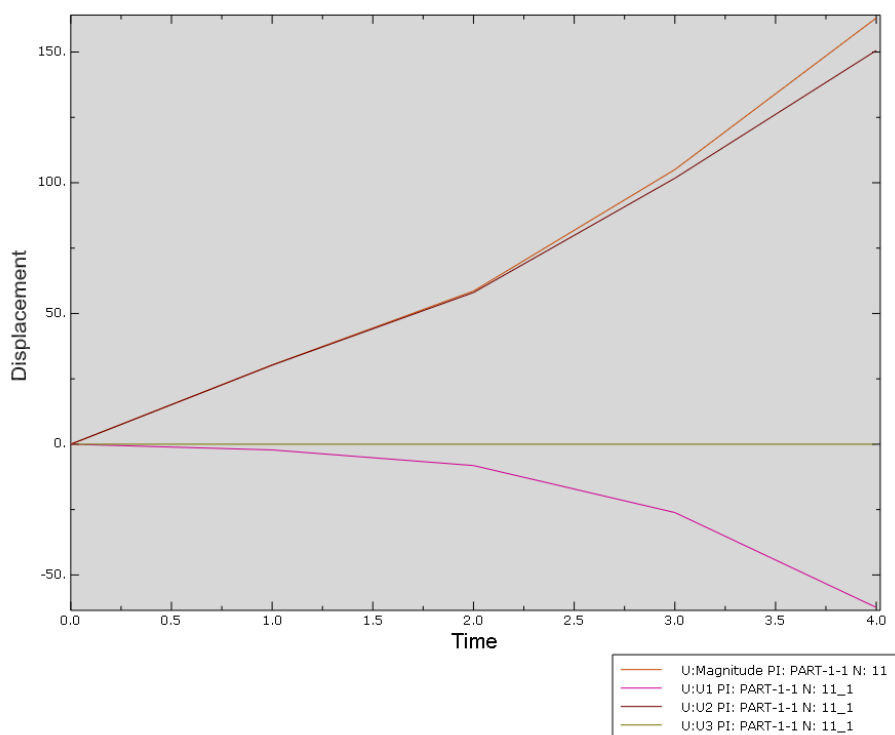
Po izvršeni numerični analizi so vsi želeni rezultati običajno že zbrani v datoteki za izpis rezultatov. Poprocesor omogoča analizo izbranih podatkov, njihovo obdelavo in zapis v datoteko, ki jo je možno odpreti oz. uvoziti v programih za urejanje besedila ali preglednic. Najpogosteje podatke prenesemo v obliki preglednic in jih tam podrobneje obdelamo, npr. v Microsoft Excelu. Pri tem je treba paziti na obliko zapisa decimalnega ločila, ki je v programih za numerične analize običajno pika, v Excelu pa je v primeru slovenske nastavitve programa decimalno ločilo vejica. Rezultati v obliki številčk predstavljajo tudi osnovo za pripravo diagramov, ki so opisani v naslednjem podpoglavju. Na sliki 12.2 je prikazan tipičen izpis rezultatov analize, kjer so izpisani pomiki s povzetkom najmanjših in največjih vrednosti.

<i>Field Output Report, written Mon Oct 30 10:38:03 2017</i>				
<i>Source 1</i>				
-----				
<i>ODB: D:/Ulbin/Fem/Talum2016/talum_T2190_02.odb</i>				
<i>Step: Step-1</i>				
<i>Frame: Increment 1: Step Time = 1.000</i>				
<i>Loc 1 : Nodal values from source 1</i>				
<i>Output sorted by column "Node Label".</i>				
<i>Field Output reported at nodes for part: PART-1-1</i>				
<i>Node</i>	<i>U.Magnitude</i>	<i>U.U1</i>	<i>U.U2</i>	<i>U.U3</i>
<i>Label</i>	<i>@Loc 1</i>	<i>@Loc 1</i>	<i>@Loc 1</i>	<i>@Loc 1</i>
-----				
<i>1</i>	<i>16.5502E-03</i>	<i>5.64699E-03</i>	<i>-65.5379E-06</i>	<i>15.5569E-03</i>
<i>2</i>	<i>17.4607E-03</i>	<i>7.68216E-03</i>	<i>6.57301E-06</i>	<i>15.6799E-03</i>
<i>3</i>	<i>17.4642E-03</i>	<i>8.55753E-03</i>	<i>238.067E-06</i>	<i>15.2221E-03</i>
<i>4</i>	<i>16.7914E-03</i>	<i>8.32375E-03</i>	<i>207.448E-06</i>	<i>14.5817E-03</i>
<i>5</i>	<i>18.8791E-03</i>	<i>8.50840E-03</i>	<i>175.443E-06</i>	<i>16.8522E-03</i>
<i>6</i>	<i>20.5846E-03</i>	<i>9.61349E-03</i>	<i>365.177E-06</i>	<i>18.1982E-03</i>
<i>7</i>	<i>27.0201E-03</i>	<i>9.93252E-03</i>	<i>65.2738E-06</i>	<i>25.1282E-03</i>
<i>8</i>	<i>47.3166E-03</i>	<i>9.23672E-03</i>	<i>-397.463E-06</i>	<i>46.4046E-03</i>
<i>9</i>	<i>53.0729E-03</i>	<i>8.80886E-03</i>	<i>-263.884E-06</i>	<i>52.3361E-03</i>
<i>10</i>	<i>45.8924E-03</i>	<i>6.46694E-03</i>	<i>-839.030E-06</i>	<i>45.4267E-03</i>
.....				
<i>169531</i>	<i>496.077E-06</i>	<i>387.311E-06</i>	<i>-7.32849E-06</i>	<i>-309.884E-06</i>
<i>169532</i>	<i>0.</i>	<i>315.714E-36</i>	<i>5.32388E-36</i>	<i>-74.6928E-36</i>
<i>Minimum</i>	<i>0.</i>	<i>-11.9537E-03</i>	<i>-5.19593E-03</i>	<i>-14.5388E-03</i>
<i>At Node</i>	<i>169532</i>	<i>46210</i>	<i>2409</i>	<i>125</i>
<i>Maximum</i>	<i>69.8134E-03</i>	<i>11.4309E-03</i>	<i>5.37437E-03</i>	<i>69.8124E-03</i>
<i>At Node</i>	<i>9501</i>	<i>48802</i>	<i>1595</i>	<i>9501</i>
<i>Total</i>	<i>4.22030E+03</i>	<i>12.2921</i>	<i>-2.38565</i>	<i>3.68205E+03</i>

Slika 12.2 – Številčni izpis rezultatov v obliki preglednice

## 12.3 Diagrami

Diagrami nazorno prikazujejo, kako se opazovana veličina spreminja v odvisnosti od položaja v prostoru ali času. Na vodoravno os (absciso) navadno nanašamo položaj ali čas, na navpično os (ordinato) pa izbrane izračunane fizikalne veličine, kot so napetosti, pomiki, energije, hitrosti, pospeški itd. Slika 12.3 prikazuje rezultate dinamične računalniške simulacije v smislu poteka pomikov izbrane točke v odvisnosti od časa.



Slika 12.3 – Primer diagrama

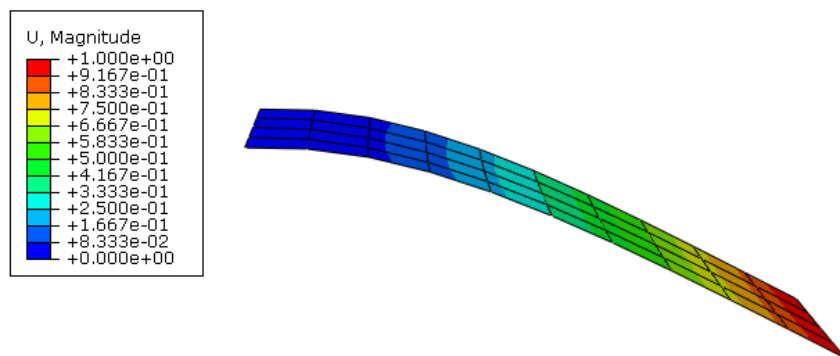
## 12.4 Slike

Najpogosteje se za prikaz rezultatov računalniških simulacij uporabljajo slike, npr. porazdelitev napetosti ali pomikov. Le-te je možno iz programov za poprocesiranje izvoziti in po potrebi naknadno obdelati v urejevalniku slik. Za slike sta najprimernejši vrsti datotek .jpg in .png.

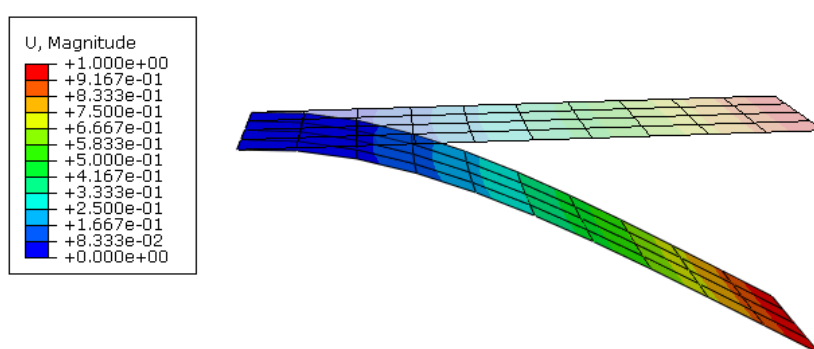
### 12.4.1 Pomiki

Pomiki so najpogosteje prva veličina, ki je po uspešno izvedeni računalniški simulaciji grafično prikazana v obliki deformirane oblike obravnavane strukture (slika 12.4). Pomike navadno prikažemo v barvah, kjer enake barve pomenijo področja enakih velikosti pomikov. Navadno se spekter barv začneja z najmanjšo prikazano vrednostjo pomikov (modro) in sega do največje vrednosti pomika (rdeče).

Običajno poprocesor samodejno bistveno poveča velikost pomikov z namenom očitnega prikaza deformirane oblike strukture v primerjavi z izhodiščno (nedeformirano) obliko, slika 12.5.

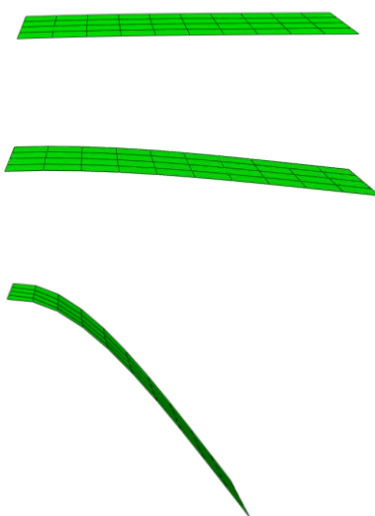


Slika 12.4 – Pomiki strukture



Slika 12.5 – Primerjava deformirane in nedeformirane strukture

Vendar pa poprocesorski programi omogočajo prilagajanje faktorja povečave/pomanjšave (slika 12.6), kar je želeno še posebej takrat, kadar nameravamo prikazati dejanske pomike v strukturi ali ko samodejno povečevanje preveč popači prikaz deformirane oblike [12.5].



Slika 12.6 – Različni faktorji povečave/pomanjšave pri enakih izračunanih pomikih



### 12.4.2 Specifične deformacije

Specifične deformacije podajajo relativno, od geometrije neodvisno mero pomikov. Pri tem ločimo normalne specifične deformacije  $\varepsilon_{ii}$ , ki povzročajo spremembo volumna, in strižne specifične deformacije  $\gamma_{ij}$ , ki povzročajo spremembo oblike obravnavanega problema.

Za prikaz deformacijskega stanja pogosto uporabimo tudi glavne specifične deformacije, saj lahko za poljubno deformacijsko stanje:

$$\underline{\underline{\varepsilon}} = \begin{bmatrix} \varepsilon_{xx} & \gamma_{xy} & \gamma_{xz} \\ \gamma_{yx} & \varepsilon_{yy} & \gamma_{yz} \\ \gamma_{zx} & \gamma_{zy} & \varepsilon_{zz} \end{bmatrix} \quad \gamma_{ij} = \gamma_{ji} \quad (12.1)$$

vedno določimo nov koordinatni sistem, katerega osi so pravokotne na ravnine, na katerih delujejo največje specifične deformacije in kjer strižnih specifičnih deformacij ni. Takšne specifične deformacije imenujemo glavne specifične deformacije ( $\varepsilon_1 > \varepsilon_2 > \varepsilon_3$ ):

$$\underline{\underline{\varepsilon}} = \begin{bmatrix} \varepsilon_1 & 0 & 0 \\ 0 & \varepsilon_2 & 0 \\ 0 & 0 & \varepsilon_3 \end{bmatrix} \quad (12.2)$$

in predstavljajo lastne vrednosti simetričnega deformacijskega tenzorja [12.5].

Pri nelinearnih elastoplastičnih analizah nas pogosto zanima, ali je notranja napetost v strukturi ob delovanju zunanje obremenitve presegla mejo tečenja, zaradi česar se struktura plastično (trajno) deformira. Velikost plastične specifične deformacije v poprocesorjih navadno interpretiramo s primerjalno plastično deformacijo; to je skalarna vrednost, ki jo izračunamo po enačbi (12.3).

$$\varepsilon_{eq}^p = \sqrt{\frac{2}{3} \varepsilon_{ij}^p \varepsilon_{ij}^p} \quad (12.3)$$

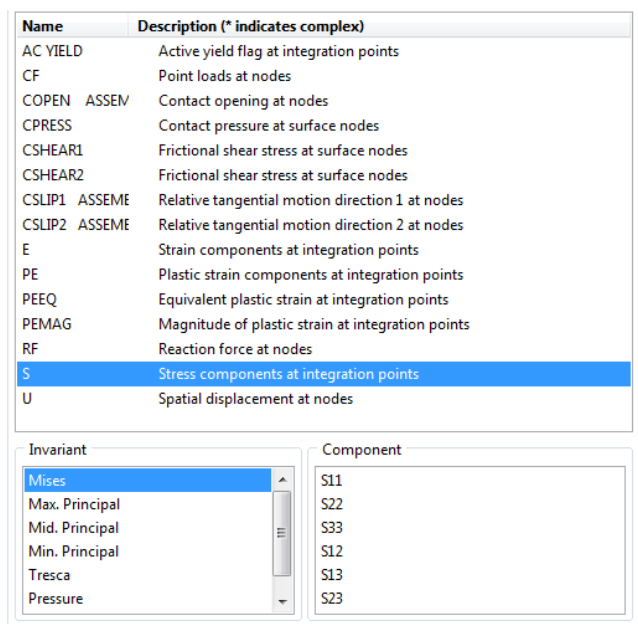
### 12.4.3 Napetosti

Napetosti podajajo relativno, od geometrije neodvisno, specifično obremenitev posamezne točke obravnavanega problema. Pri tem ločimo normalne napetosti  $\sigma_{ii}$ , ki so posledica normalnih specifičnih deformacij  $\varepsilon_{ii}$ , in strižne napetosti  $\tau_{ij}$ , ki so posledica strižnih specifičnih deformacij  $\gamma_{ij}$ . Napetosti so v posameznih materialih sorazmerne specifičnim deformacijam v skladu z njihovimi konstitutivnimi zakoni.

Cilj standardne napetostne analize je prikazati, da napetosti v vseh točkah obravnavane strukture ne presegajo določenih mej (dopustne napetosti, meje plastičnosti ipd.). Za namen vrednotenja lahko izberemo poljubno komponento napetostnega tenzorja, slika 12.7.

Prikaz posameznih komponent napetostnega tenzorja je manj običajen, saj nas navadno zanima celosten opis napetostnega stanja v točkah modela. Iz tega razloga pri vrednotenju napetosti pogosto obravnavamo primerjalno napetost, kot npr. napetost von Mises (enačba 12.4), ki jo lahko izračunamo na podlagi komponent napetostnega tenzorja.

$$\sigma_{vM} = \sqrt{\frac{1}{2} \left[ (\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 + 6(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2) \right]} \quad (12.4)$$



Slika 12.7 – Izbira napetostnih komponent za interpretacijo rezultatov

Pri interpretaciji rezultatov pogosto uporabimo tudi glavne napetosti, saj lahko za poljubno napetostno stanje:

$$\underline{\sigma} = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix} \quad \tau_{ij} = \tau_{ji} \quad (12.5)$$

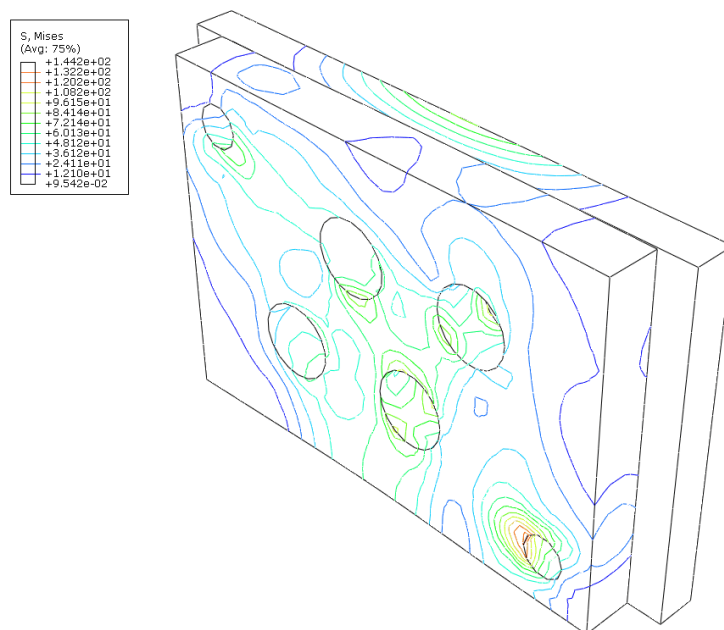
vedno določimo novi koordinatni sistem, katerega osi so pravokotne na ravnine, na katerih delujejo največje napetosti in kjer strižnih napetosti ni. Takšne napetosti imenujemo glavne napetosti ( $\sigma_1 > \sigma_2 > \sigma_3$ ):

$$\underline{\sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \quad (12.6)$$

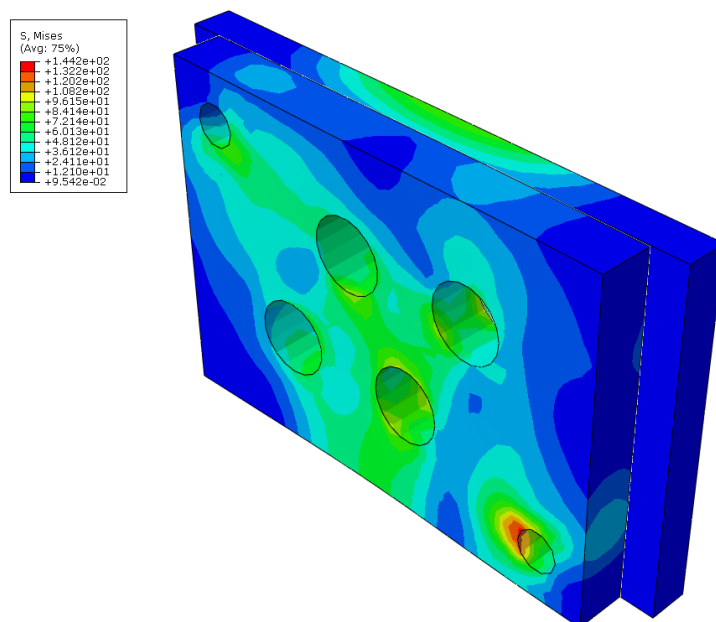
in predstavljajo lastne vrednosti simetričnega napetostnega tenzorja [12.5].

Velikost glavnih napetosti ali skalarne primerjalne napetosti, npr. napetost von Mises, se lahko prikaže na modelu s krivuljami, ki povezujejo iste vrednosti (angl. *isoline*), ali barvami in ustrezno barvno lestvico ali legendo. Porazdelitev napetosti po obarvani komponenti najpogosteje prikazujemo grafično z uporabo barv; to so t. i. konturne slike (angl. *contour map*). Te imajo lahko obliko diskretnih krivulj, ki povezujejo točke z enakimi vrednostmi napetosti (izolinije, angl. *isoline*), kot je prikazano na sliki 12.8, ali zvezno z uporabo senčene

porazdelitve napetosti, kot je prikazano na sliki 12.9. Slednji način je pogosteje uporabljen za poročila, saj omogoča pripravo slik v črno-beli barvi oz. v odtenkih sive [12.6].

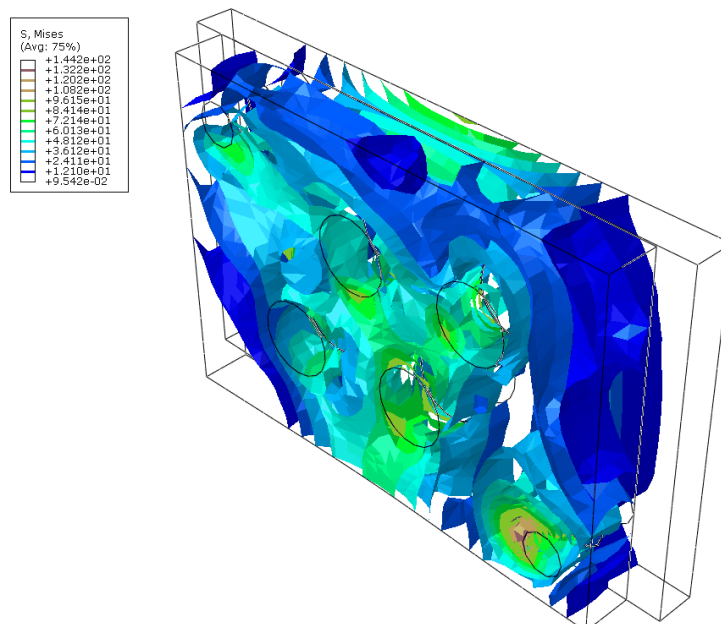


Slika 12.8 – Porazdelitev napetosti, prikazana z izolinijami



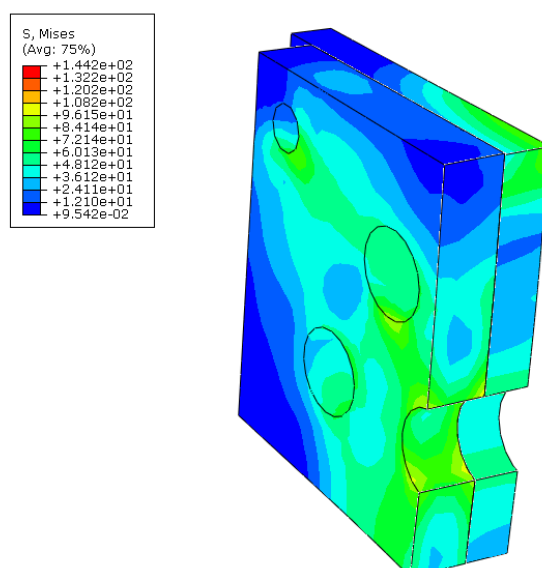
Slika 12.9 – Senčni prikaz porazdelitev napetosti

Eden izmed načinov prikaza porazdelitve napetosti je tudi uporaba t. i. izopovršin (angl. *isosurface*). Le-te predstavljajo površine, ki povezujejo točke z enakimi vrednostmi napetosti, slika 12.10.

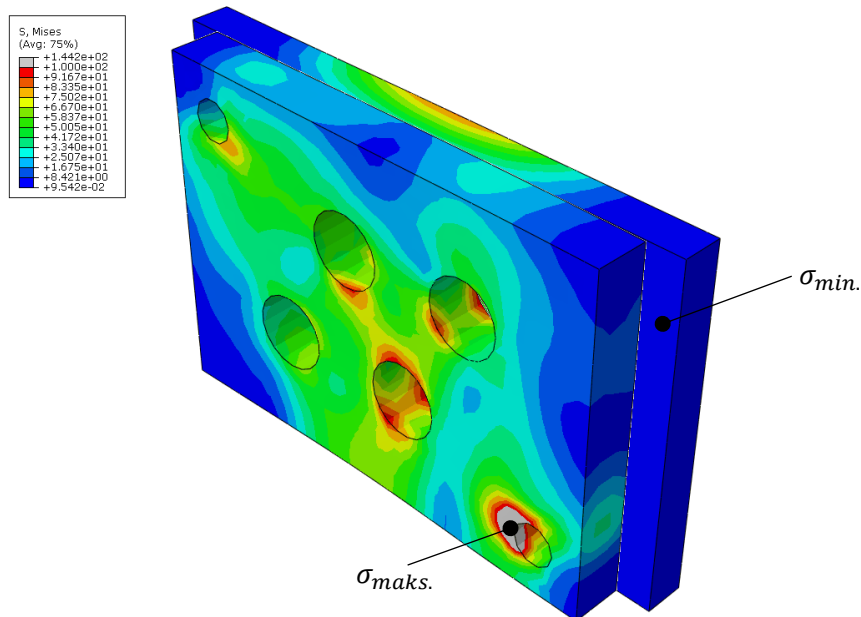


Slika 12.10 – Prikaz porazdelitve napetosti s površinami enakih napetosti

Z namenom preglednejšega prikaza notranjosti modela lahko le-tega tudi prerežemo in prikažemo porazdelitev napetosti na prerezani površini, kot to prikazuje slika 12.11. Če pa želimo ponazoriti preseganje določenih vrednosti napetosti, lahko prikazane napetosti omejimo z zgornjo in spodnjo mejo, kot prikazuje slika 12.12. Območja z napetostmi, ki niso znotraj predpisanih meja, so prikazane z drugo barvo in jih je možno hitro prepoznati.

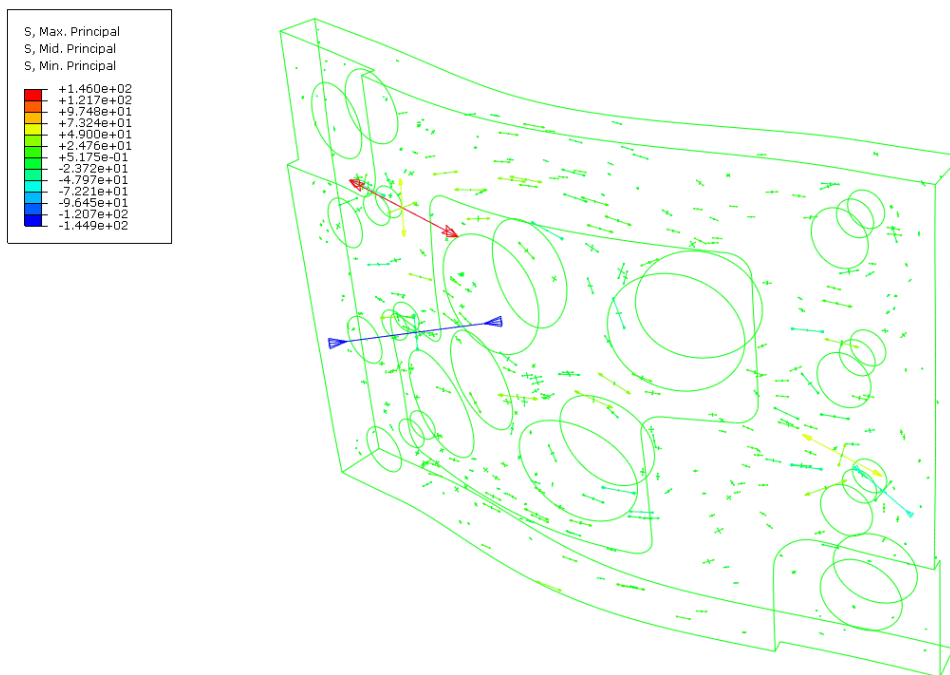


Slika 12.11 – Prikaz porazdelitev napetosti na prerezanem modelu



Slika 12.12 – Omejitve prikaza napetosti z zgornjo in spodnjo mejo

Vektorji glavnih napetosti so lahko prikazani kot z barvo označene puščice (vektorji), ki nakazujejo smer in velikost glavnih napetosti, slika 12.13. Takšne vektorje v smereh glavnih napetosti so včasih imenovali malteški križi, danes pa lahko prikazujemo različne rezultate analiz v obliki vektorjev, kjer smer označuje smer rezultata, dolžina in barva puščice pa nam pokažeta, kakšna je relativna velikost izbranega rezultata.



Slika 12.13 – Vektorji glavnih napetosti

## 12.5 Animacije

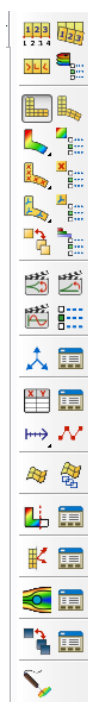
Animacije predstavljajo pregleden in priljubljen način prikaza poteka pomikov in napetosti, še posebej v primeru računalniških predstavitev. Pri animacijah se obravnavana veličina spreminja s časovno spreminjajočo se obremenitvijo (dinamične simulacije) ali s postopno obremenitvijo (vendar časovno neodvisno) – od neobremenjene do popolnoma obremenjene strukture (statične simulacije). V primeru analize pomikov se pogosto animira sprememba od nedeformiranega modela do prikazanih deformacij z uporabo faktorja povečave (posebej zanimive so animacije rezultatov velikih deformacij), v primeru opazovanja napetosti pa se animira sprememba napetosti od najmanjše do največje. Možen je tudi sočasen prikaz animiranih pomikov (sprememba geometrije) in napetosti (sprememba barve). Za izvoz animacij iz programa za poprocesiranje se najpogosteje uporabljajo animirana .gif in .avi ter zapis videoposnetka .mpeg.

## 12.6 Poprocesiranje v programu ABAQUS

Program ABAQUS deluje z bazo .cae v predprocesorju (angl. *pre-processor, model*) in z bazo .odb v poprocesorju (angl. *post-processor, results*), kot je prikazano na sliki 12.14. V datoteki .odb so zbrani vsi rezultati analize. Prikazujemo jih v zavihku rezultati (angl. *results*).



Slika 12.14 – Izbira rezultatov v programu ABAQUS



Slika 12.15 – Ikone v meniju za poprocesiranje v programu ABAQUS

Vsebino prikaza izbiramo z ikonami ali ukazi v meniju za poprocesiranje (slika 12.15). V zgornjem delu orodjarne so nastavitve prikaza, pod njimi pa ukazi za prikaz rezultatov.

## 12.7 Literatura

- [12.1] NAFEMS, *Guidelines to Finite Element Practice*. Glasgow: Bell and Bain Ltd., 1992.
- [12.2] Finite Elemente Methode [splet], Dosegljivo: <https://de.wikipedia.org/wiki/Finite-Elemente-Methode>. [Datum dostopa 5. 12. 2017].
- [12.3] FEA Buyer's Guide for Pre- and Postprocessing Software [splet], Dosegljivo: <http://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/6391/FEA-Buyers-Guide-for-Pre-and-Postprocessing-Software.aspx>. [Datum dostopa 5. 12. 2017].
- [12.4] Finite Element Analysis: Post-processing [splet], Dosegljivo: <http://www.finiteelement.com/feawhite4.html>. [Datum dostopa 5. 12. 2017].
- [12.5] D. Baguley, D. R. Hose, *How to – Choose a Finite Element Pre- & Post-processor*. East Kilbride: NAFEMS, 1994.
- [12.6] A. Umek, *Mehanika deformabilnih teles*. Ljubljana: Didakta, 1998.





## 13. Poročilo o izvedeni računalniški simulaciji

Poročilo predstavlja kratek, natančen in jedrnat dokument, ki opisuje izvedeno računalniško simulacijo in pogosto podaja priporočila za nadaljnje ukrepanje. Zato mora biti dokument jasen, pregleden in dobro strukturiran. Zahteve za natančno obliko in vsebino poročila se razlikujejo med ustanovami in podjetji, zato je pred začetkom pisanja poročila treba ugotoviti, ali obstajajo posebne smernice ali navodila.

Pisanje poročila navadno predstavlja zadnji korak pri analizi problema, ki sledi definiciji problema, zbiranju potrebnih informacij, reševanju problema in organizaciji rezultatov.

### 13.1 Vsebina poročila

Tehnično poročilo o izvedbi računalniški simulacij praviloma vsebuje naslednjih pet sklopov:

- opis in definicijo obravnavanega problema,
- opis numeričnega modela,
- opis izvedbe računalniških simulacij,
- prikaz rezultatov,
- zaključke s strokovnim mnenjem.

Seveda ima poročilo tudi naslovnico in seznam uporabljenih virov. Poročilo mora vsebovati vse podatke, ki so potrebni, da lahko analitik na njihovi podlagi v celoti ponovi in verificira izvedene računalniške simulacije. V nadaljevanju so navedeni in opisani posamezni sestavni deli poročila.

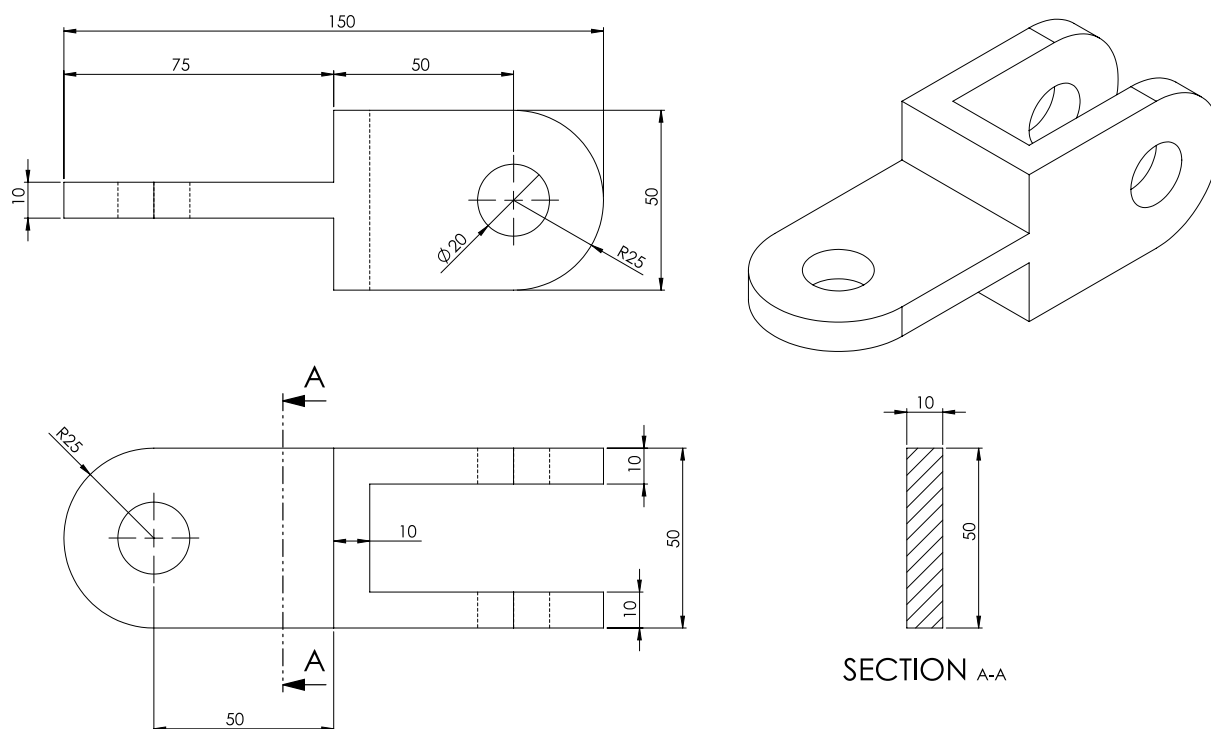
#### 13.1.1 Naslovna stran in uvodne strani

V splošnem mora naslovna stran poročila biti pregledna. Vsebovati mora logotip izvajalca, naslov poročila, morebitno oznako ter kodo projekta, naročnika projekta, izvajalca projekta, odgovornega vodjo projekta s sodelavci in kraj ter datum. Pri študentskih poročilih je treba vstaviti logotip in naziv univerze ter fakultete, navesti naslov poročila, ime in priimek, vpisno številko, naziv programa, smeri ter predmeta, v okviru katerega je bilo poročilo pripravljeno.

Naslovni strani sledijo uvodne strani, ki lahko zajemajo zahvalo, povzetek, kazalo, kazalo slik in preglednic ter seznam uporabljenih simbolov in kratic. Slednja izpustimo, če v besedilu, kjer simbol ali kratica nastopata, pojasnimo njun pomen.

#### 13.1.2 Opis in opredelitev problema

V uvodu so predstavljene komponente (sestave) s sliko dimenzij oziroma načrtom (slika 13.1), uporaba komponente ter namen in material, iz katerega je komponenta izdelana (preglednica 13.1). Podani so pramateri delovanja, oblika in velikost obremenitev ter opis podpor. Določena je vrsta predvidenega izračuna (kvazistatični, dinamični, linearni, nelinearni ipd.) in zahtevani rezultati (pomiki, deformacije, napetosti, temperature ipd.).



Slika 13.1 – Primer načrta obravnavane komponente

Preglednica 13.1 – Primer navedbe materialnih podatkov [13.1, 13.2]

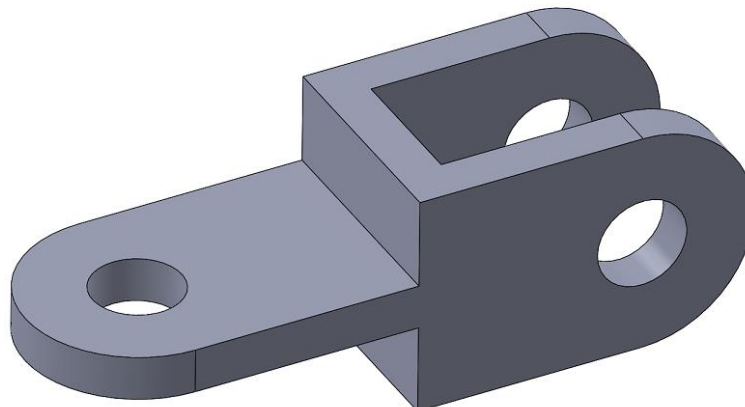
Vrsta jekla	Gostota [kg/m <sup>3</sup> ]	Modul elastičnosti [MPa]	Poissonovo število /	Meja plastičnosti [MPa]	Natezna trdnost [MPa]	Def. pri poružitvi [%]
S235JR	7800	210.000	0,30	235	360	25

Rezultate numeričnega preračuna, njihovo natančnost in ustreznost lahko prekontroliramo na osnovi analitičnih preračunov ali eksperimentalnih preizkusov. Kadar eksperimentalne meritve niso bile opravljene, izvedemo poenostavljen analitični preračun za določanje primerljivih rezultatov. Analitična določitev napetosti ali pomikov naj bo izvedena na vsaj enem izbranim mestu v komponenti (za kontrolo in primerjavo z numerično določeno napetostjo). Poročilo naj vsebuje postopek analitičnega preračuna z vsemi potrebnimi podatki in sliko, ki prikazuje mesto na komponenti, kjer so bile določene napetosti oziroma pomik.

### 13.1.3 Opis numeričnega modela

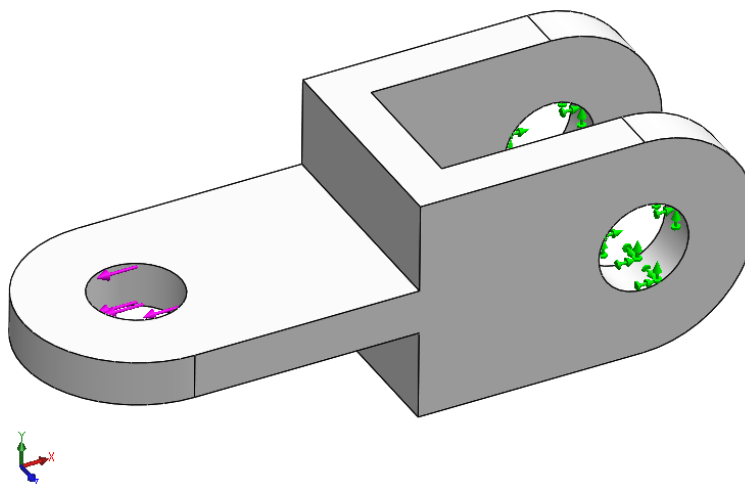
Del poročila, v katerem je obravnavan numerični model, vsebuje celovit opis geometrije, diskretizacije, materiala, obremenitev ter robnih in začetnih pogojev:

- geometrijski model: predstavljen je geometrijski model (slika 13.2) z opisom uporabljenih dimenzijskih (3D, 2D ali 1D) in geometrijskih (volumni, ploskve, linije, radiji, luknje in drugi detajli) poenostavitvev (npr. simetrija); opredeljene so merske enote za numerično analizo in izbran koordinatni sistem;



Slika 13.2 – Primer geometrijskega modela komponente

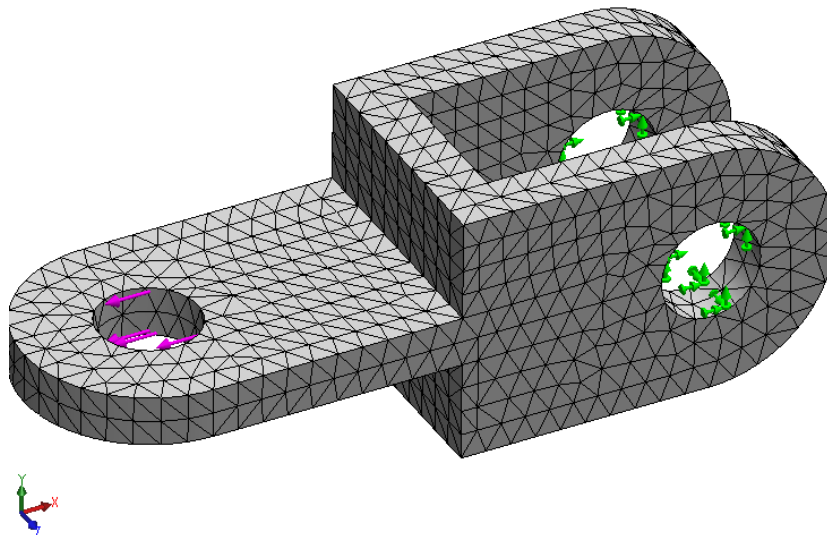
- materialni model: opisan je izbrani materialni (konstitutivni) model (npr. izotropen, linearno-elastičen, elastoplastičen ipd.), vključno z uporabljenimi materialnimi lastnostmi (npr. modul elastičnosti, Poissonovo razmerje, meja tečenja, krivulja utrjevanja ipd.);
- vpetje in obremenitve: podani morajo biti vrsta in smer delovanja posameznih vpetij in obremenitev, ki delujejo na numerični model, pri obremenitvah pa tudi njihova velikost; s sliko lahko ponazorimo mesto na komponenti, kjer delujejo določena vpetja in obremenitve (slika 13.3);



Slika 13.3 – Primer nepomične podpore (zelena barva) in obremenitve (vijolična barva)

- mreža končnih elementov: natančno je predstavljena prostorska diskretizacija numeričnega modela nedeformirane mreže končnih elementov komponente (slika 13.4); podani so izbrana

globalna in/ali lokalna velikost končnih elementov, tip in razporeditev končnih elementov; navedena so število končnih elementov in število vozlišč ter število prostostnih stopenj modela; po potrebi se lahko doda tudi slika predpriprave modela na mreženje (geometrijske delitve);



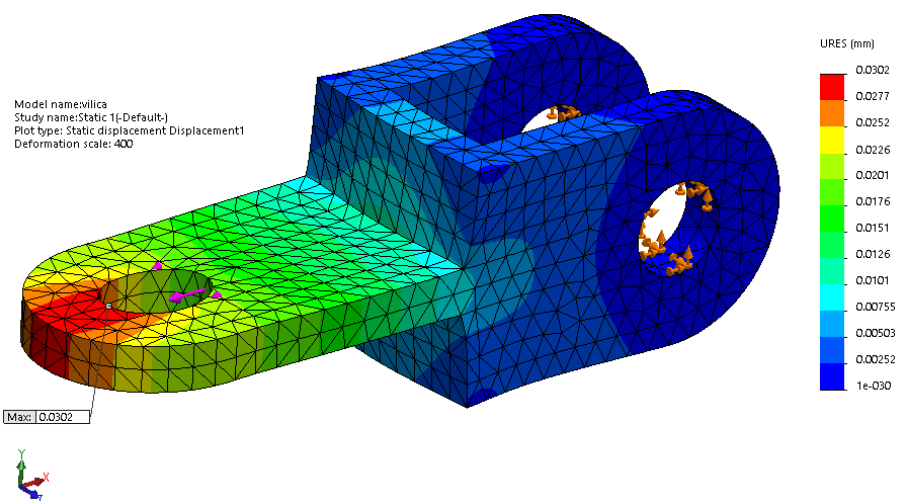
Slika 13.4 – Primer mreže končnih elementov

#### 13.1.4 Opis izvedbe računalniške simulacije

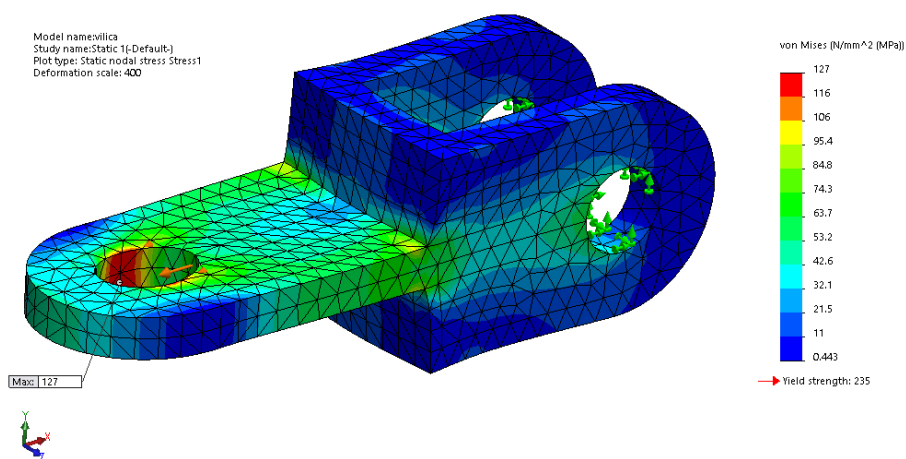
Pri opisu izvedbe simulacije navedemo imena in različice uporabljenega računalniškega programa ter dodatne podatke, potrebne za izvedbo simulacije (npr. simulacijski parametri, konvergenčni kriterij, število iteracij, paralelno procesiranje itd.). Pomembna podatka sta tudi računski čas (CPU, realni čas) posamezne simulacije in zmogljivost računalnika ali računalniške gruče, kjer je bila računalniška simulacija izvedena.

#### 13.1.5 Rezultati

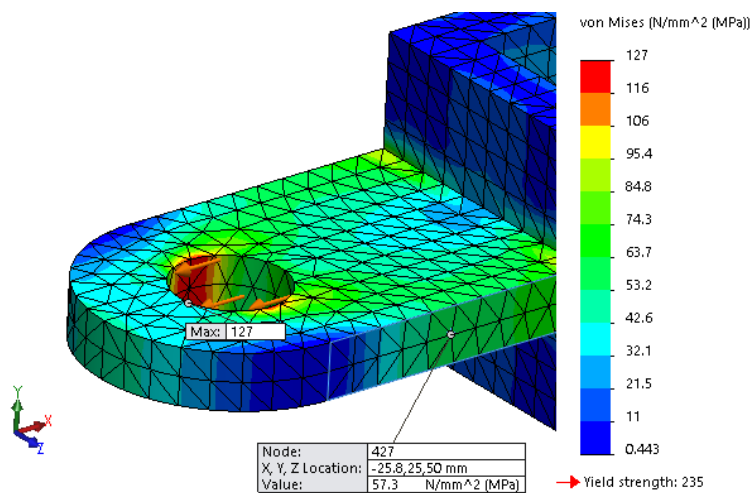
Rezultati numeričnih analiz se lahko podajo v številski (preglednice), grafični (diagrami, izolinije, animacije itd.) ali besedilni (opis proces deformacije, potek porušitve itd.) obliki. Poročilo navadno sestavlja predstavitev največjega pomika na komponenti z vrednostjo in oznako na sliki, sliko polja skupnih pomikov z legendo (slika 13.5), predstavitev največje napetosti na komponenti z vrednostjo in oznako na sliki ter sliko polja primerjalnih napetosti z legendo (slika 13.6). V tem delu poročila prav tako prekontroliramo ustreznost rezultata, tako da napetost ali pomik primerjamo z analitičnimi ali eksperimentalnimi rezultati (slika 13.7) in podamo odstopanja. Če želimo natančneje opisati posamezen pojav, v katerega smo dobili vpogled po izvedeni analizi, le-tega podrobno opišemo in po možnosti grafično predstavimo (slika 13.8). Pri grafičnem podajanju rezultatov moramo izbrati ustrezno velikost fonta, da so številke in črke dovolj velike, pri podajanju številčnih vrednosti pa rezultat po potrebi smiselno zaokrožimo.



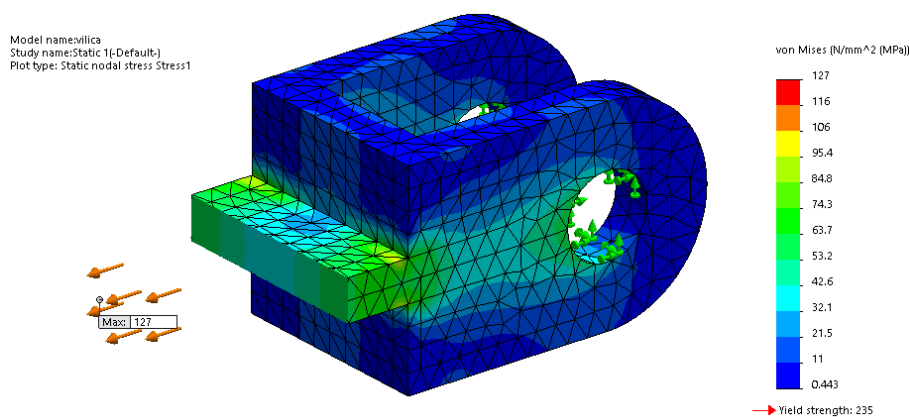
Slika 13.5 – Skupni pomiki (skalirni faktor pomikov 400x)



Slika 13.6 – Von Misesove primerjalne napetosti (skalirni faktor pomikov 400x)



Slika 13.7 – Napetost na zunanjem robu (za primerjavo s poenostavljeno analitično rešitvijo)



Slika 13.8 – Napetosti po celotnem prerezu

### 13.1.6 Zaključki in strokovno mnenje

V zaključkih strnemo izvedeni preračun, pridobljene rezultate primerjamo z dopustnimi vrednostmi. Prikažemo in obrazložimo morebitna mesta koncentracij napetosti ter podamo strokovno mnenje glede opravljenih simulacij in njihovih rezultatov. Po možnosti dodamo še predloge za izboljšave in morebitne spremembe konstrukcije.

### 13.1.7 Seznam uporabljenih virov

Seznam uporabljenih virov (literaturo) navedemo na koncu vsebinskega dela poročila, v zaporednem vrstnem redu, kot si sledijo sklici v besedilu. Pri navajanju literature uporabimo le oglat oklepaj z zaporedno številko literature, npr. [1].

### 13.1.8 Priloge

Kadar želimo poročilo obogatiti z dodatnimi obsežnejšimi podatki, tabelaričnimi vrednostmi in slikami, ki pa niso ključni za razumevanje poročila, jih dodamo na kocu poročila kot priloge.

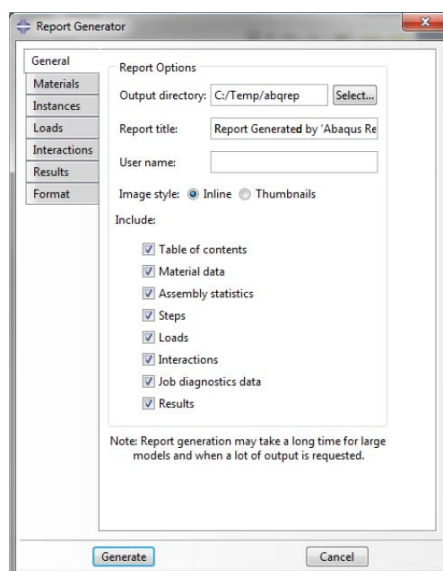
## 13.2 Oblika poročila

Oblika poročila mora biti v skladu z zahtevami in predlogami naročnika. Napisano naj bo z računalnikom, skrbno urejeno in vezano. Pri črno-belem tiskanju poročila je treba biti pozoren, da se ne izgubijo podatki, ki so v elektronski različici poročila prikazani barvno. Poglavja, podpoglavja in strani poročila naj bodo ustrezno oštevilčene. Slike, diagrami in preglednice naj bodo prav tako oštevilčeni, saj se je v besedilu nanje treba sklicevati, in podnaslovljeni.

### 13.3 Samodejno pripravljena poročila

Večina sodobnih inženirskih programskih paketov (npr. ABAQUS, CATIA, KISSsoft, SCIA Engineer) omogoča samodejno pripravo poročil (slika 13.9). Poročila so navadno ustvarjena v

obliki html ali pdf. Zaradi širokega nabora problemov, ki jih lahko obravnavamo, tovrstna poročila pogosto obravnavanega problema ne opišejo dovolj natančno in ne vključujejo vseh podatkov, ki bi utegnile zanimati naročnika poročila. Zato je samodejna poročila glede na zahteve naročnika poročila treba ustrezno dopolniti in preoblikovati.



Slika 13.9 – Priprava samodejnega poročila v programskem paketu ABAQUS

## 13.4 Literatura

- [13.1] MatWeb [splet], Dosegljivo: <http://www.matweb.com/>. [Datum dostopa 5. 12. 2017].
- [13.2] B. Kraut, *Strojniški priročnik*. Ljubljana: Tehniška založba Slovenije, 1998.
- [13.3] NAFEMS, *Guidelines to Finite Element Practice*. Glasgow: Bell and Bain Ltd., 1992.



Univerza v Mariboru

---

Fakulteta za strojništvo

