PYTHON-SOVELLUS ABAQUS-OHJELMAN LASKENTAMALLIN JA TULOSTEN KÄSITTELYYN

Inga Mattila

Opinnäytetyö
Kesäkuu 2014
Kone- ja tuotantotekniikka
Tuotekehitys

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Kone- ja tuotantotekniikan koulutusohjelma
Tuotekehitys

INGA MATTILA:
Python-sovellus Abaqus-ohjelman laskentamallin ja tulosten käsittelyyn

Opinnäytetyö 56 sivua
Kesäkuu 2014

Tässä työssä oli tavoitteena tutustua Python-ohjelmointikielen sekä Abaqus Scripting -käyttöliittymän käyttöön FEM-mallin ja laskentatulosten käsittelyn automatisoimiseksi ja nopeuttamiseksi. Tarkoituksena oli toteuttaa Python-sovellus, joka käsittelee FE-analyysiohjelmalla Abaqus tehtyä mallia sekä saatuja tuloksia. Python-sovelluksen haluttiin parametrisoivan mallia ja poimivan haluttuja tuloksia Abaqus-ohjelman binäärisestä tulostiedostosta. Työ toteutettiin Teknologian tutkimuskeskus VTT:llä.

Toteutettu Python-sovellus poimii Abaqus-ohjelman tulostiedostosta maksimisiirtymät ja -jännitykset sekä tarvittaessa kaikki solmusiirtymät koko mallista tai halutusta mallin osa-alueesta eri materiaaliarvoilla (kimmomoduuli ja Poissonin vakio). Sovelluksen toimivuutta tarkasteltiin useiden erityyppisten FEM-mallien avulla. Testiesimerkkien tulosten perusteella suureiden maksimiarvojen poiminta tulostiedostosta soveltuu ristikko-, kehä-, levy- ja solidimalleille sekä materiaalin parametrisointi levy- ja solidimalleille. Toteutetulla sovelluksella materiaalien parametrisointi ja tulosten poiminta yksinkertaistuvat ja nopeutuvat erillisillä laskentamalleilla ja FEM-ohjelman peruskäskyillä toteutettuihin analyyseihin verrattuna.

Sovellusta voidaan laajentaa materiaaliarvojen osalta myös muiden ominaisuuksien luomiseen sekä parametrisointiin. Lisäksi tulisi perehtyä kuormitusten ja reunaehtojen määrittämiseen Abaqus Scripting -käyttöliittymällä. Tällöin pystytään toistamaan nopeasti esimerkiksi usean pisteen kuormitus, jos mallin elementtiverkkoa tai analyysityyppiä halutaan vaihtaa.

Koska tutkimuksen kohteena olevat mallit ovat usein kookkaita, mallien laskenta suoritetaan tehokkailla palvelinkoneilla. Abaqus Scripting -käyttöliittymällä tehdyn sovelluksen skaalautumista moniytimisessä palvelinkoneessa tulisi tarkastella, jotta voidaan varmistua laskentatehon maksimaalisesta käytöstä.

Tämä versio työstä on julkinen, eikä sisällä liitteitä.

Asiasanat: mallinnus, FEM, Python, Abaqus, Abaqus Scripting Interface

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Mechanical and Production Engineering
Product Development

INGA MATTILA:
Python Script for the FEA Software Abaqus to Process the Model and the Results

Bachelor's thesis 56 pages
June 2014

The objective of this study was to investigate the use of Python programming language and Abaqus Scripting Interface to automate and speed up the FE modelling process. The purpose was to create a Python script that modifies the model and reads the results from the binary output database generated by the FEA software Abaqus. The study was carried out in the VTT Technical Research Centre of Finland.

The script extracts the maximum displacements and maximum stresses of the FE model from the output database and the model can be parameterized with different material values (Young's modulus and Poisson's ratio). One can examine the values of the whole model or of parts only. In addition, it is possible to extract all node displacements of the model. The script was tested with different models and the results demonstrated the applicability of the code for truss, beam, shell and continuum models.

In the future, it is possible to make the script to support a wider range of material properties to be parameterized. Modelling of loads and boundary conditions can be performed by means of Abaqus Scripting Interface, for example, to fasten the handling of many loading conditions.

This is the public version of the study and does not include appendices.

Key words: modelling, FEM, Python, Abaqus, Abaqus Scripting Interface

SISÄLLYS

LYHENTEET JA TERMIT

Abaqus/CAE	Complete Abaqus Environment
CFD	Computational Fluid Dynamics
CLI	Command Line Interface
FEA	Finite Element Analysis
FEM	Finite Element Method, elementtimenetelmä
GUI	Graphical User Interface
kerneli	käyttöjärjestelmän ydin
mdb	model database
odb	output database
olio	ohjelman perusyksikkö, sisältää joukon loogisesti yhteenkuuluvaa tietoa ja toiminnallisuutta
solidi	kontinuumi, jatkuvan kiinteän aineen kappale
E	kimmomoduuli
G	liukumoduuli
ν	Poissonin vakio
τ	leikkausjännitys
σ	normaalijännitys
γ	liukuma
$\{\varepsilon\}$	venymävektori
[B]	kinemaattinen matriisi
[C]	vaimennusmatriisi
[E]	kimmomatriisi
[k]	elementin jäykkyysmatriisi
[K]	rakenteen jäykkyysmatriisi
[M]	massamatriisi
[N]	muotofunktiomatriisi
{d}	elementin siirtymävektori
{R}	kuormitusvektori
{$\dot{U}$}	nopeusvektori
{u}	elementin solmusiirtymävektori
{U}	rakenteen solmusiirtymävektori
{$\ddot{U}$}	kiihtyvyysvektori

1 JOHDANTO

Lujuuslaskennallisia suureita voidaan ratkaista numeerisilla menetelmillä, joista parhaaksi on osoittautunut elementtimenetelmä. Elementtimenetelmään perustuvien FE-analyysiohjelmien avulla voidaan mallintaa haluttuja rakenteita, joiden tuntemattomat suureet pystytään ratkaisemaan kohtuullisessa ajassa suurista laskennallisista operaatioista huolimatta. FEM-mallin toteutuksen, laskennan ja tulosten tarkastelun apuna voidaan käyttää erillisiä, ohjelmoitavia sovelluksia esimerkiksi toistuvien työvaiheiden tekemiseen.

Tässä työssä tutustuttiin FE-analyysiohjelmassa Abaqus olevan Abaqus Scripting -käyttöliittymän sekä Python-ohjelmointikielen käyttöön FEM-mallin toteutuksen ja tulosten käsittelyn automatisoimiseksi ja nopeuttamiseksi. Tarkoituksena oli toteuttaa Python-sovellus, joka käsittelee FE-analyysiohjelmalla Abaqus tehtyä mallia sekä saatuja tuloksia. Työssä toteutetun Python-sovelluksen haluttiin parametrisoivan mallia ja poimivan saatuja tuloksia ohjelman Abaqus tuottamasta odb-tiedostosta.

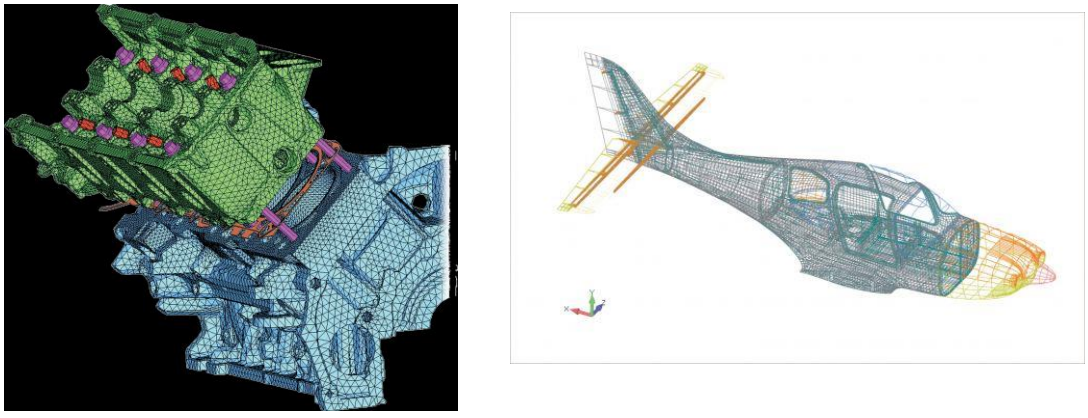
Työ toteutettiin Teknologian tutkimuskeskus VTT:n Tehokkaat koneet ja kulkuvälineet -tutkimusalueella, jossa rakenteiden mekaanisen käyttäytymisen numeerinen analysointi yleisesti sekä ohjelman Abaqus soveltaminen ovat keskeinen osa toimintaa. Työssä toteutetun sovelluksen lähdekoodi on sijoitettu liitteeseen, joka ei ole julkinen. Sovelluksen toteutusta on analysoitu tekstissä yleisellä tasolla.

2 ELEMENTTIMENETELMÄ

Lujuusoppi eli solidin mekaniikka on fysiikan ala, jossa tutkitaan erilaisten kuormitusten vaikutusta kiinteiden, tuettujen rakenteiden käyttäytymiseen. Tavoitteena on ratkaista muun muassa rakenteeseen syntyviä jännityksiä, muodonmuutoksia ja siirtymiä tai kiihtyvyyksiä ja nopeuksia. Näitä suureita voidaan ratkaista muun muassa elementtimenetelmän, *finite element method* (FEM) tai *finite element analysis* (FEA), avulla muodostamalla rakenteelle fysikaalisiin lakeihin perustuva matemaattinen malli.

Elementtimenetelmä otettiin käyttöön 1950-luvulla ensin lentokoneteollisuudessa, josta se levisi nopeasti myös muiden teollisuudenalojen käyttöön. Nimen ”finite element” keksi amerikkalainen Ray Clough vuonna 1960. Elementtimenetelmän luotettavuus kasvoi tutkimusten myötä ja 1960-luvun lopusta alkaen markkinoilla on ollut yleiskäyttöisiä FE-analyysiohjelmia. (Cook, Malkus, Plesha & Witt 2001, 10)

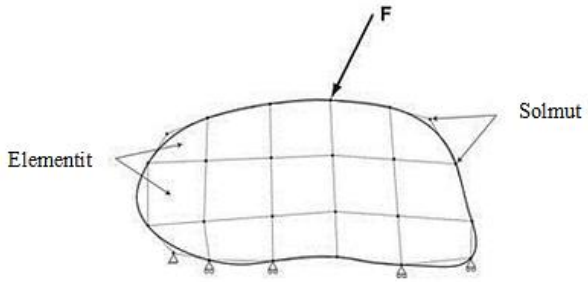
FE-analyysien avulla rakenteen toimivuutta voidaan tarkastella jo suunnitteluvaiheessa, jolloin vältytään ylimääräisten prototyyppien tekemiseltä ja päästään parhaaseen lopputulokseen nopeammin. Myös erilaisten vaihtoehtojen, esimerkiksi materiaalien, kokeilu on nopeaa. (kuva 1)



KUVA 1. FEM-malleja (Abaqus Unified FEA Brochure 2013, Ultralight Aircrafts 2009)

2.1 Periaate

Elementtimenetelmässä tutkittava rakenne jaetaan pieniksi osa-alueiksi eli elementeiksi. Elementit yhdistyvät toisiinsa pisteissä, joita kutsutaan solmuiksi, ja muodostavat kappaleeseen yhtenäisen elementtiverkon (kuva 2). Elementtimenetelmässä tuntemattomien suureiden yhtälöt määritellään ensin paikallisesti yhden elementin alueella. Saadut yhtälöt kootaan koko rakenteen kattavaan yhtälösysteemiin, josta saadaan halutut ratkaisut. Menetelmää voidaan soveltaa viivarakenteisiin (sauvat ja palkit), pintarakenteisiin (levyt/laatat ja kuoret) ja solidirakenteisiin (Lähteenmäki 2009, 4.7).

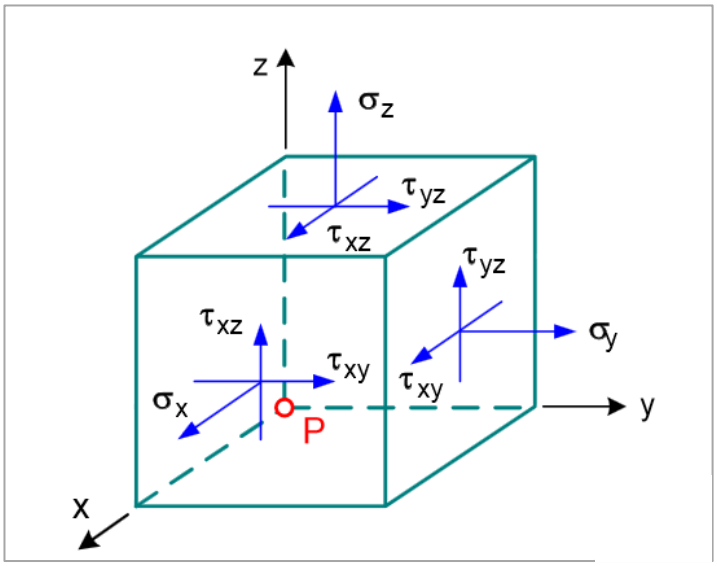


KUVA 2. Piirrosesimerkki kohteen diskretisoinnista elementtimalliksi reunaehtoineen **(**Morgan & Bouxsein 2005, 9)

Elementtimenetelmän ratkaisut voivat perustua siirtymä-, voima- tai sekamenetelmiin. Siirtymämenetelmään perustuvassa elementtimenetelmässä elementtiverkon perusyhtälöstä ratkaistaan ensin solmusiirtymät, joiden avulla saadaan ratkaistua muut tuntemattomat suureet (Lähteenmäki 2009, 4.8). Tässä työssä on tarkasteltu siirtymämenetelmään perustuvaa elementtimenetelmää.

2.2 Lineaarisen lujuusopin perusyhtälöt

Tarkasteltaessa lujuusopillista tilannetta elementtimenetelmän yhtälösysteemi muodostuu lujuusopin perusyhtälöistä. Tällöin perussuureita ovat jännityskomponentit (normaalijännitys σ ja leikkausjännitys τ kuvan 3 mukaisesti), muodonmuutoskomponentit (venymä ε ja liukuma γ) sekä siirtymäkomponentit (u, v, w) (Lähteenmäki 2008 – 2009, I.31). Perussuureet kytkeytyvät toisiinsa jännityskomponenttien tasapainoyhtälöiden, materiaaliyhtälöiden ja kinemaattisten yhtälöiden kautta. Yhtälöissä huomioidaan perussuureiden lisäksi rakenteeseen kohdistuvat voimat f ja materiaaliominaisuudet (esimerkiksi kimmomoduuli E, liukumoduuli G ja Poissonin vakio ν).



KUVA 3. Kolmiulotteisen elementin jännityskomponentit **(**Lähteenmäki 2012 - 2013, 3)

Lineaariset lujuusopin perussuureita koskevat yhtälöt muodostuvat jännityskomponenttien tasapainoyhtälöistä

$$\sigma_{x,x} + \tau_{xy,y} + \tau_{xz,z} + f_x = 0 \quad (1)$$

$$\tau_{xy,x} + \sigma_{y,y} + \tau_{yz,z} + f_y = 0 \quad (2)$$

$$\tau_{xz,x} + \tau_{yz,y} + \sigma_{z,z} + f_z = 0 \quad (3)$$

kinemaattisista yhtälöistä

$$\varepsilon_x = u_{,x} \quad (4)$$

$$\varepsilon_y = v_{,y} \quad (5)$$

$$\varepsilon_z = w_{,z} \quad (6)$$

$$\gamma_{xy} = u_{,y} + v_{,x} \quad (7)$$

$$\gamma_{xz} = u_{,z} + w_{,x} \quad (8)$$

$$\gamma_{yz} = v_{,z} + w_{,y} \quad (9)$$

ja materiaaliyhtälöistä

$$\varepsilon_x = \frac{1}{E}[\sigma_x - \nu(\sigma_y + \sigma_z)] \quad (10)$$

$$\varepsilon_y = \frac{1}{E}[\sigma_y - \nu(\sigma_x + \sigma_z)] \quad (11)$$

$$\varepsilon_z = \frac{1}{E}[\sigma_z - \nu(\sigma_x + \sigma_y)] \quad (12)$$

$$\gamma_{xy} = \tau_{xy} / G \quad (13)$$

$$\gamma_{xz} = \tau_{xz} / G \quad (14)$$

$$\gamma_{yz} = \tau_{yz} / G \quad (15)$$

(Lähteenmäki 2008 – 2009, I.32).

2.3 Elementin alueen yhtälöt

Siirtymämenetelmään perustuvan elementtimenetelmän perusyhtälöt voidaan johtaa potentiaalienergian ääriarvoperiaatteesta, virtuaalisten siirtymien periaatteesta ja Galerkinin menetelmällä (Cook ym. 2001, 179). Potentiaalienergian ääriarvoperiaatteessa rakenne on tasapainossa, kun potentiaalienergia on pienimmillään. Tällöin perussuureet saadaan ratkaistua niitä hallitsevista osittaisdifferentiaaliyhtälöistä. Tuntemattomien suureiden yhtälöt määritellään ensin paikallisesti yhden elementin alueella. Elementin siirtymille {d} saadaan yhtälö

$$\{d\} = [N]\{u\}, \quad (16)$$

missä {u} on solmusiirtymävektori ja [N] matriisi, joka sisältää interpoloinnissa käytettävät muotofunktiot. Venymä {ε} saadaan muotofunktioiden osittaisista derivaatoista muodostetun kinemaattisen matriisin [B] ja solmusiirtymien {u} avulla seuraavasti:

$$\{\varepsilon\} = [B]\{u\}. \quad (17)$$

Yhden elementin jäykkyysmatriisiksi muodostuu yhtälö

$$[k] = \int [B]^T[E][B]\, dV, \quad (18)$$

missä [E] on kimmomatriisi ja $[B]^T$ kinemaattisen matriisin transpoosi. (Cook ym. 2001, 88 - 89; Lähteenmäki 2009, 4.8).

2.4 Globaali yhtälösysteemi

Elementin sisällä muodostetut elementtikohtaiset yhtälöt kootaan globaaliin yhtälösysteemiin, jolloin saadaan ratkaistua koko tutkittavan kappaleen tuntemattomat suureet (Kouhia & Tuomala 2009, 41). Tuntemattomille solmuarvoille muodostetaan lineaarinen yhtälöryhmä sijoittelusummaamalla elementtien jäykkyysmatriisit, solmukuormitukset ja ekvivalenttiset solmukuormitukset. Muodostunutta yhtälöryhmää sanotaan elementtiverkon perusyhtälöksi eli tasapainoyhtälöksi. Tämä statiikan perusyhtälö kirjoitetaan muotoon

$$\mathbf{[K]\{U\} = \{R\}}, \tag{19}$$

missä [K] on globaali jäykkyysmatriisi, {U} siirtymävektori ja {R} kuormitusvektori.

Dynamiikan tarkasteluissa tulee ottaa huomioon myös massa, kiihtyvyys, vaimennus ja nopeus. Tällöin perusyhtälö saa muodon

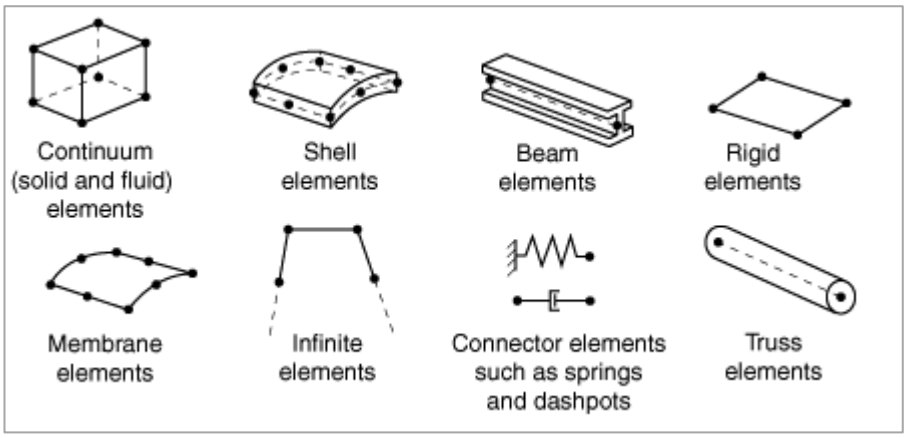
$$\mathbf{[M]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} = \{R\}}, \tag{20}$$

missä [M] on massamatriisi, {Ü} kiihtyvyysvektori, [C] vaimennusmatriisi ja {U̇} nopeusvektori (Cook ym. 2001, 376).

Elementtimenetelmällä saatu ratkaisu on likimääräinen. Tarkkuuteen vaikuttavat muun muassa interpoloinnista ja integroinnista syntyvät virheet, elementtikuormitusten korvaaminen ekvivalenttisilla solmukuormituksilla ja geometrian mallinnusvirheet. (Lähteenmäki 2009, 5.1, 6.1)

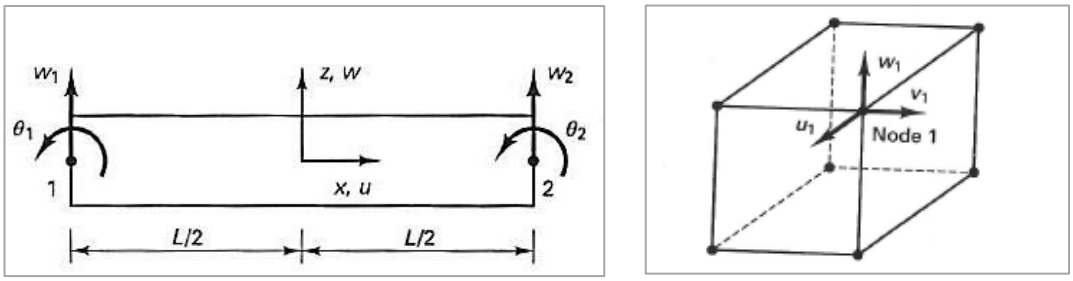
2.5 Elementtityypit

Erityyppisiä elementtejä on olemassa runsaasti, ja sopiva elementtityyppi valitaan tarkasteltavan lujuusopillisen tapauksen mukaan. Esimerkiksi FE-analyysiohjelmassa Abaqus on saatavilla yli kuusisataa elementtityyppiä (kuva 4).



KUVA 4. Yleisesti käytetyt elementtityypit (Abaqus 6.12 Documentation 2014)

Elementtityypit ryhmitellään solidi- eli kontinuumielementteihin ja rakenne-elementteihin. Solidielementtien solmupisteet sijaitsevat rakenteen luonnollisissa pisteissä, kun taas rakenne-elementtien solmupisteet sijaitsevat elementin keskipinnassa (kuva 5). Solidielementit ovat kaksiulotteisia (mm. tasojännitys- ja tasovenymäelementit eli levyelementit ja rotaatiosymmetriset elementit) tai kolmiulotteisia. Rakenne-elementit voivat olla kaksi- tai kolmiulotteisia, kuten esimerkiksi sauva-, palkki-, laatta- ja kuorielementit. (Bathe 1996, 341, 397)

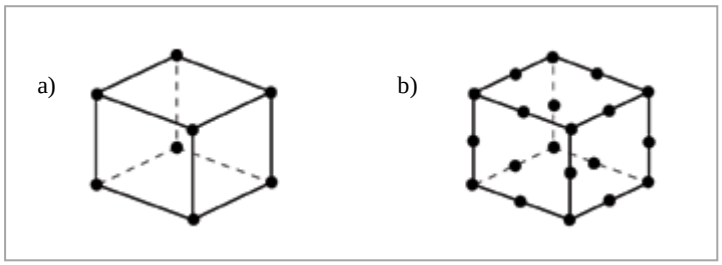


KUVA 5. Solmupisteiden sijainti a) rakenne-elementin (palkki) keskipinnassa ja b) solidielementin (3D) rakenteen luonnollisissa pisteissä (Bathe 1996, 274, 292)

Rakenne-elementeissä malli on viiva- tai keskipintarakenne, jolle määritellään tarvittavat suureet, kuten esimerkiksi materiaaliparametrit, poikkipinnan pinta-ala, neliömomentti ja kuoren paksuus (Elementtimenetelmät I 2012 – 2013, 229). Vapausasteita (siirtymät ja rotaatiot) on yleensä kolme, mikäli elementti on kaksiulotteinen ja kuusi, mikäli elementti on kolmiulotteinen. Rakenne-elementtimallien laskenta vie usein vähemmän laskentakapasiteettia, koska elementtien määrä on yleensä huomattavasti pienempi kuin solidielementtejä käytettäessä. (Bathe 1996, 199).

Solidielementti myötäilee kappaleen muotoa ollen jatkuva eli kontinuumi ja sille määritellään tarvittavat materiaaliparametrit, esimerkiksi kimmomoduuli ja Poissonin vakio. Solmuilla on kaksi (2D) tai kolme (3D) vapausastetta (siirtymät). Solidielementtejä käytetään, kun mallin geometria on monimuotoinen ja yksinkertaisempien elementtityyppien käyttö ei ole mahdollista (Lähteenmäki 2006, 1.11).

Yleisimmin käytetään lineaarisia ja kvadraattisia elementtejä (kuva 6). Kvadraattiset elementit ovat laskennallisesti tarkempia ja ne seuraavat hyvin kappaleen kaarevia linjoja.



KUVA 6. a) Lineaarinen ja b) kvadraattinen solidielementti (Abaqus 6.12 Documentation 2014)

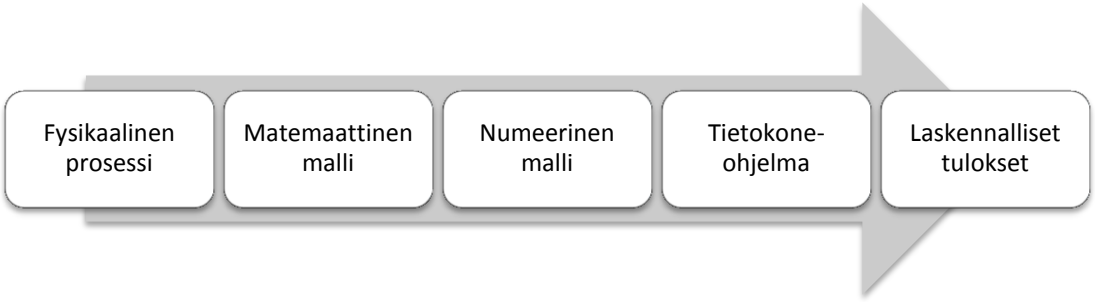
2.6 Lineaarinen ja epälineaarinen FE-analyysi

Rakenteen käyttäytyminen voi olla lineaarista tai epälineaarista riippuen geometristen siirtymien suuruudesta, materiaaliominaisuuksista ja pintojen välisistä kontakteista analyysin aikana. Lineaarisessa analyysissä siirtymien oletetaan olevan hyvin pieniä, materiaali on lineaarisesti kimmoista ja reunaehdot pysyvät samoina kuormituksen aikana. Laskenta suoritetaan alusta loppuun samaa jäykkyysmatriisia käyttäen (Bathe 1996, 485)

Epälineaarisessa analyysissä materiaaliominaisuudet ja pintojen väliset kontaktit voivat muuttua analyysin aikana. Materiaalin epälineaarisuus voi johtua esimerkiksi viskoelastisuudesta tai plastisuudesta. Muodonmuutokset tai siirtymät voivat olla suuria. Tapahtuvat muutokset otetaan huomioon laskennan aikana iteratiivisesti päivittämällä pääasiassa jäykkyysmatriisia. Tämä monimutkaistaa ratkaisua huomattavasti. (Bathe 1996, 485; Cook ym. 2001, 595)

3 OHJELMOINTI LASKENNALLISESSA TIETEESSÄ

Matemaattisten mallien sisältämät laskennalliset operaatiot ovat niin suuria, että niiden toteuttamiseen tarvitaan tietotekniikkaa. Ohjelmoinnin avulla saadaan toteutettua sovelluksia, jotka suorittavat halutut matemaattiset operaatiot suhteellisen nopeasti, vaikka niiden analyyttinen ratkaiseminen käsin olisi mahdotonta. Matemaattisesti määritetyt numeeriset algoritmit muutetaan ohjelmointikielten avulla tietokoneohjelmaksi, jolloin ne ovat tietokoneen ymmärrettävissä. Laskennallisen prosessin eteneminen kohteen fysikaalisesta tarkastelusta ohjelmointiin ja laskennallisiin tuloksiin voidaan esittää ideaalisena kaaviona (kuva 7). (Gustafsson 2011, 11)

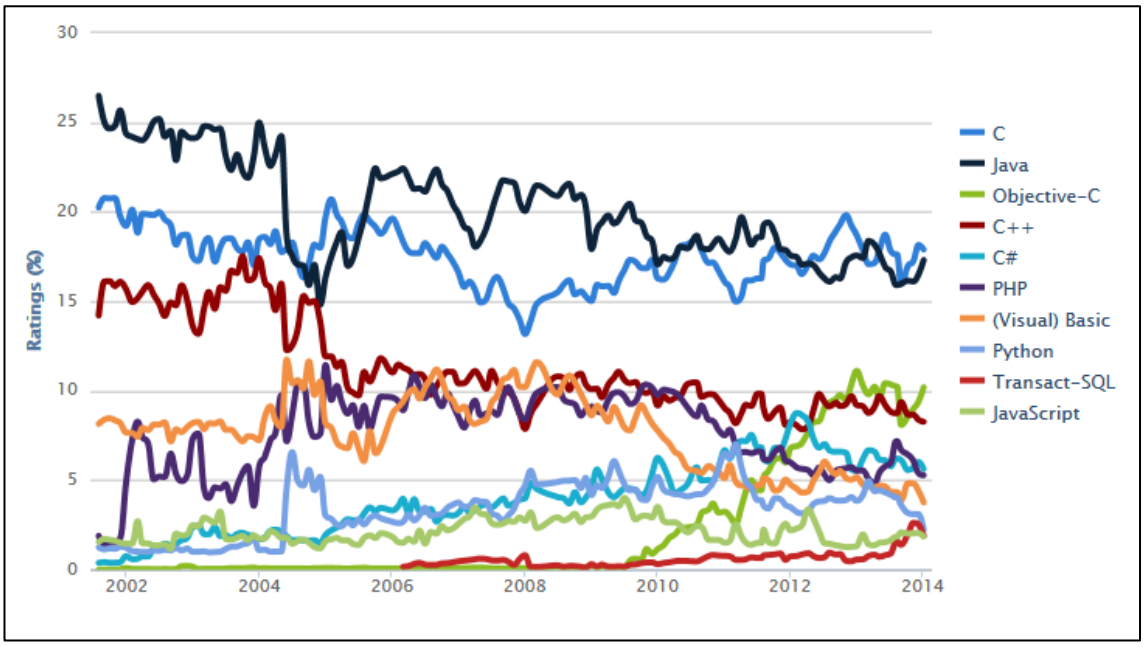


KUVA 7. Laskennallinen prosessi (Gustafsson 2011, 11)

FE-analyysiohjelmien avulla käyttäjä voi suorittaa laskennallisia operaatioita ilman ohjelmointikielten käyttöä. Analyysiohjelmia voidaan kuitenkin haluttaessa käyttää myös erilaisten ohjelmasovellusten kautta ohjelmointia hyväksi käyttäen. Ohjelmointia voidaan myös hyödyntää täydentämällä analyysiohjelmia sovelluksilla, jotka esimerkiksi automatisoivat toistettavia työvaiheita, laajentavat toiminnallisuutta uusilla ominaisuuksilla tai käsittelevät saatuja tuloksia halutusti.

3.1 Ohjelmoinnin perusajatus

Ohjelmoinnilla tarkoitetaan toimintaohjeita, jotka välitetään tietokoneelle konekielellä. Ohjelmoimalla toteutettava ohjelma muodostuu lähdekoodista. Se toteutetaan ohjelmointikielellä, joka käännetään tai tulkataan konekielelle. Käännös tapahtuu ennen ohjelman suorittamista ja tulkkaus ohjelman suorittamisen aikana. Erilaisia ohjelmointikieliä on runsaasti ja niistä käytetyimpiä (kuva 8) ovat mm. C, Java, Objective-C, C++, PHP, Visual Basic ja Python (TIOBE Programming Community Index 2013; Vihavainen, Paksula, Luukkainen, Laaksonen, Mikkola, Laurinharju & Pärtel 2013, 80).



KUVA 8. Kymmenen maailmanlaajuisesti suosituinta ohjelmointikieltä (TIOBE Programming Community Index 2013)

Tietokoneohjelmat voidaan toteuttaa joko proseduraalisella ohjelmoinnilla tai olio-ohjelmoinnilla. Proseduraalinen ohjelma koostuu erillisistä suoritettavista osista eli funktioista. Olio-ohjelmoinnissa tietoja käsitellään olioiden avulla. Olio on ohjelman perusyksikkö, joka sisältää joukon loogisesti yhteenkuuluvaa tietoa ja toiminnallisuutta. Olion sisältämää tietoa voidaan käsitellä olion sisältämien metodien avulla. (Vihavainen ym. 2013, 80, Hetland 2005, 139)

3.2 Laskennallisessa tieteessä yleisimmin käytetyt ohjelmointikielet

Laskennallisessa tieteessä yleisimmin käytetyt ohjelmointikielet ovat olleet Fortran, C, C++ ja Java. Nykyään tieteellisissä ohjelmistoissa käytetään laaja-alaisesti myös Python-kieltä. Pythonin käytön laskennallisessa ohjelmoinnissa odotetaan kasvavan lähitulevaisuudessa sen helppokäyttöisyyden ja luettavuuden tuomien etujen vuoksi. Myös monia muita ohjelmointikieliä käytetään laskennallisessa tieteessä, kuten Visual Basic, Perl, Ruby, Tcl, Smalltalk, Cobol, Ada, Algol, Pascal, Haskell, Common Lisp ja Scheme (Rashed & Ahsan 2012, 27). Akateemisessa tutkimuksessa käytetään lisäksi Matlabia, joka on interaktiivinen vektori- ja matriisilaskentaohjelmisto sekä matriisikieli. (Gustafsson 2011, 292; Rashed & Ahsan 2012, 26 – 28; History of the Software 2013)

3.2.1 Fortran, C ja C++

Fortran (FORmula TRANslation) kehitettiin 1950-luvulla erityisesti tieteellisen laskennan tarpeisiin ja se oli ensimmäinen yleisesti käytetty korkeatasoinen ohjelmointikieli. Iästään huolimatta kieltä käytetään edelleen erityisesti monimutkaisessa tieteellisessä laskennassa. C-kieli kehitettiin 1970-luvulla käyttöjärjestelmien luomiseen ja se on todella tehokas ja joustava kieli. C-kieltä ei erityisesti kehitetty tieteelliseen käyttöön, joten siitä puuttuu joitakin ominaisuuksia, kuten kompleksilaskenta. C++ kehitettiin 1980-luvulla C-kieltä laajentamalla tarkoituksena helpottaa mm. olioperustaista ohjelmointia. (Rashed & Ahsan 2012, 26 - 27)

3.2.2 Matlab

Matlabin (MATrix LABoratory) ensimmäinen versio kehitettiin 1970-luvulla ja se kirjoitettiin Fortran-kielellä. 1980-luvulla Matlab kirjoitettiin uudelleen C-kielellä, johon nykyiset versiot perustuvat. Matlab on matriisikieli, joka soveltuu laajasti tieteelliseen ja tekniseen laskentaan. Lisäksi Matlabiin saatavilla olevalla Simulink-ohjelmalla voidaan simuloida monialaisia dynaamisia järjestelmiä. Kalliiden lisenssien takia Matlabin käyttöä on rajatumpaa esimerkiksi Pythonin käyttöön verrattuna. Lisäksi Matlabin uudet versiot eivät välttämättä tue aiempien versioiden koodia, mikä hankaloittaa ohjelmien jakamista ja uudelleenkäyttämistä. (Apiola & Laine 2010; Python vs Matlab 2013)

3.2.3 Python

Python on korkean tason ohjelmointikieli, joka on tulkattava ja oliosuuntautunut. Kielen loi Guido van Rossum 1990-luvun alussa. Pythonin etuna on erittäin hyvä luettavuus, minkä johdosta kieltä on helppo käyttää ja sovellukset ovat näin ollen helposti muokattavissa ja jaettavissa myös muiden käyttäjien kanssa. Python on vapaasti kaikkien käytettävissä ja hyödynnettävissä, ja tästä johtuen se myös keskustelee hyvin muiden ohjelmointikielten kanssa. (About Python 2013)

Python on ohjelmointikielenä tehokaskäyttöinen ja se on maailmanlaajuisesti suosittu. Sitä verrataan monesti Tcl-, Perl-, Ruby-, Scheme- tai Java-kieliin. Usein käyttökohtee-

na ovat yksinkertaiset sovellukset, mutta Python kelpaa myös monimutkaisten sovellusten toteuttamiseen. Python on sulautettu osaksi useita ohjelmistotuotteita, kuten esimerkiksi FEA-ohjelmaan Abaqus. (Rashed & Ahsan 2012, 28)

Pythonin haittapuolena pidetään sen hitautta esimerkiksi Java-, C- tai C++ -kieliin verrattuna. Hitaus johtuu siitä, että Python on tulkattava eikä käännetty kieli. Vaikka Python saattaa olla kielenä hitaampi, aikaa voidaan säästää ohjelmoinnin tekemisessä kielen helppokäyttöisyyden johdosta. Useimmissa ohjelmissa nopeusero ei ole huomattavissa. (Hetland 2005)

4 FE-ANALYYSIOHJELMA ABAQUS

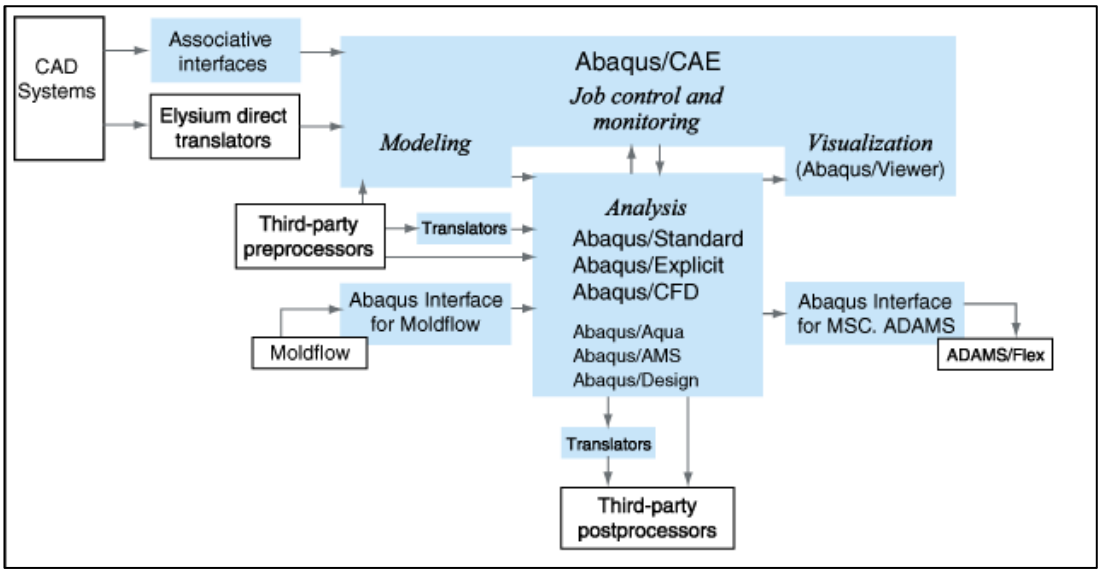
Abaqus Unified FEA -ohjelma on elementtimenetelmään perustuva ohjelma, jonka toi markkinoille vuonna 1978 perustettu Abaqus-yhtiö. Yhtiön osti vuonna 2005 Dassault Systèmes, jonka tuoteperheeseen kuuluvat muun muassa Catia, SolidWorks, Delmia, Enovia, Geovia ja Simulia. FE-analyysiohjelma Abaqus on osa Simulia-tuoteperhettä. (Abaqus Company Profile)

Abaqus-ohjelmalla voidaan ratkaista muun muassa rakenteellisia ongelmia lineaarisilla sekä epälineaarisilla analyyseilla. Tutkittavana kohteena voi olla esimerkiksi koneen rakenteen ominaisvärähtely, tukirakenteiden kuormituksenkestävyys, lentokoneen aerodynaamiset ominaisuudet, paineastian (mm. kaasupullot, kuumavesikattilat) rakenteellinen kestävyys, laitteen teräskuoren optimaalinen ainevahvuus tai vaihtoehtoisten materiaalien käytön arviointi tuotteen ominaisuuksien optimoimiseksi. Abaqus-ohjelmalla voidaan tehdä myös mm. lämmönsiirtoon, elektroniikkakomponenttien lämmönhallintaan, akustiikkaan, sähkömagnetiikkaan ja virtausdynamiikkaan liittyvää laskentaa.

Abaqus-ohjelmalla voidaan luoda malleja joko graafisesti Abaqus/CAE GUI:lla eli graafisella käyttöliittymällä tai ohjelmoimalla Abaqus Scripting -käyttöliittymällä, joka on Python-ohjelmointikielen laajennus.

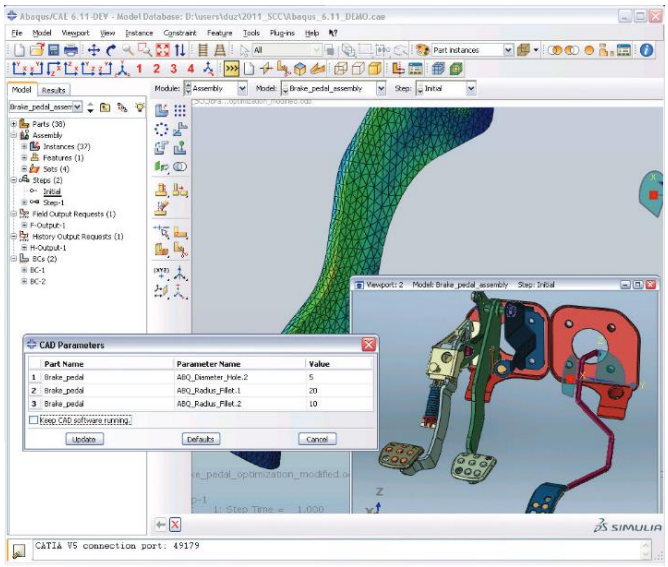
4.1 Rakenne

Abaqus muodostuu Abaqus/CAE-ympäristöstä (Complete Abaqus Environment) sekä analyysiympäristöistä: Abaqus/Standard, Abaqus/Explicit ja Abaqus/CFD (kuva 9).



KUVA 9. Ohjelman Abaqus rakenne (Abaqus 6.12 Documentation 2014)

Abaqus/CAE:ssa luodaan malli sekä suoritetaan ja seurataan laskentaa. Tuloksia tarkastellaan ja jälkikäsitellään sen sisältämässä Abaqus/Viewer -ympäristössä. Abaqus/CAE:n mallinnusprosessi jakautuu moduuleihin. Moduuleita on yksitoista (*part*, *property*, *assembly*, *step*, *interaction*, *load*, *mesh*, *optimization*, *job*, *visualization* ja *sketch*). Moduuleihin sisältyvät mallinnusprosessin kaikki työvaiheet: mallin geometrian luonti ja kokoonpano, materiaaliominaisuuksien ja pintojen määrittely, pintojen sekä osien vuorovaikutusten määrittely, analyysityypin määrittely, reunaehtojen ja kuormitusten anto, verkon luonti, laskenta sekä tulosten tarkastelu. Malli voidaan luoda graafisesti Abaqus/CAE GUI:ssa (kuva 10) tai ohjelmoimalla Abaqus Scripting -käyttöliittymällä.

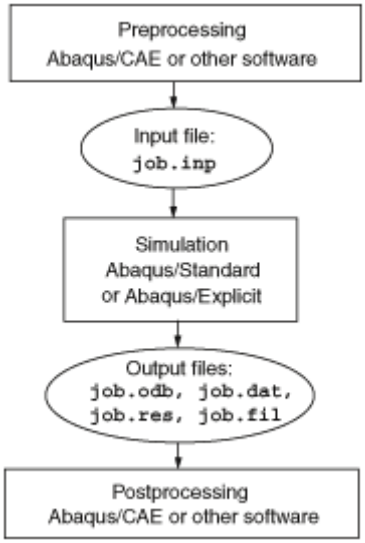


KUVA 10. Abaqus/CAE GUI -käyttöliittymä (Abaqus Unified FEA Brochure 2013)

Abaqus Unified FEA -ohjelma sisältää kolme analyysiympäristöä: Abaqus/Standard, Abaqus/Explicit ja Abaqus/CFD. Abaqus/Standard-analyysiympäristössä ratkaistaan yleisiä staattisia ja dynaamisia rakenneanalyysejä, kuten kulkuneuvojen, koneiden ja laitteiden kestävyyttä jännityksiä ja kuormituksia vastaan sekä ominaisvärähtelytaajuuksia resonanssin välttämiseksi. Abaqus/Explicit on tarkoitettu nopeiden dynaamisten ilmiöiden mallintamiseen, joita ovat esimerkiksi kappaleeseen vaikuttavat törmäykset, putoamiset ja räjähdykset. Abaqus/CFD on virtauslaskennan työkalu, jolla voidaan tutkia fluidien fysikaalista käyttäytymistä ja vaikutuksia erilaisissa tapauksissa, kuten putkistoissa, tuuliolosuhteissa ja ihmisen elimistössä.

4.2 Analyysiprosessi

FE-analyysiin sisältyy (kuva 11) esikäsittely eli laskentamallin luonti (*preprocessing*), laskenta (*simulation*) sekä jälkikäsittely (*postprocessing*) eli tulosten tarkastelu ja arviointi. Ohjelmassa Abaqus mallin esikäsittelyn voi toteuttaa Abaqus/CAE GUI- tai Abaqus Scripting -käyttöliittymällä. Syntyvän inp-tiedoston avulla suoritetaan varsinainen laskenta halutussa analyysiympäristössä, jonka tuloksena muodostuu useampi tiedosto, mm. odb-, fil-, dat- ja res -tiedostot. Tulosten käsittely voidaan toteuttaa Abaqus/CAE GUI:lla, Abaqus Scripting -käyttöliittymällä tai jollain muulla ohjelmalla, esim. Altair Hyperview.

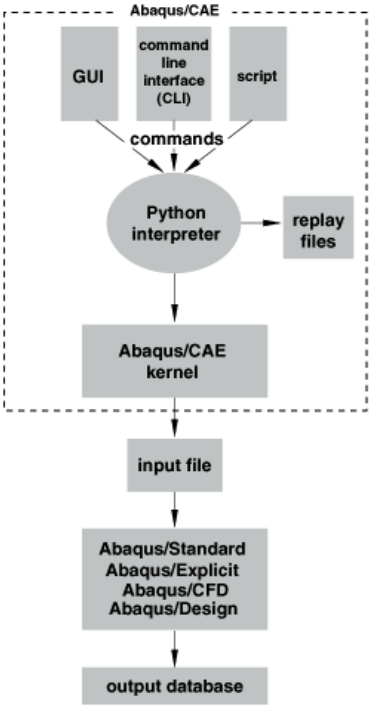


KUVA 11. Analyysiprosessin kuvaus (Abaqus 6.12 Documentation 2014)

Odb-tulostiedosto sekä fil- ja res-tiedostot ovat tietokoneen luettavaksi tarkoitettuja binääritiedostoja. Odb-tiedosto sisältää kaikki laskennassa saadut tulokset, joita voidaan lukea GUI- tai Abaqus Scripting -käyttöliittymällä. Fil-tiedostoa käytetään siirrettäessä tietoa muihin ohjelmiin tulosten jälkikäsittelyä varten ja res-tiedosto liittyy laskennan uudelleenkäynnistämiseen. Dat-tiedosto on luettavissa ja siihen tulostuvat halutut tulokset taulukkomuodossa.

4.3 Python osana ohjelmaa Abaqus

Käytettäessä graafista käyttöliittymää Abaqus/CAE GUI mallin tekemiseen, laskentaan ja tulosten käsittelyyn, muodostaa GUI komennot Python-ohjelmointikielellä (kuva 12). GUI on siis vain graafinen käyttöliittymä käyttäjän ja Python-koodia lukevan Abaqus/CAE-kernelin eli ytimen välissä. Abaqus/CAE-kerneli eli ydin on Abaqus/CAE-ympäristön keskus, jossa kaikki rakenteet, luokitukset ja ominaisuudet ovat määriteltynä. Tulkatut Python-komennot lähetetään Abaqus/CAE-kerneliin, joka lukee komennot ja käyttää sisältämiään asetuksia ja parametreja luoden mallin. Graafisen käyttöliittymän GUI sijasta voidaan käyttää Abaqus Scripting -käyttöliittymää, jonka sovellukset ovat Python-sovelluksia.



KUVA 12. Python-ohjelmointikieli osana ohjelmaa Abaqus (Abaqus 6.12 Documentation 2014)

4.4 Abaqus Scripting -käyttöliittymä

Abaqus Scripting -käyttöliittymä on oliosuuntautuneen Python-ohjelmointikielen räätälöity laajennus Abaqus-ohjelmaan. Abaqus Scripting -sovellukset ovat siis laajennettuja Python-sovelluksia, joissa käytetään samaa lausekieltä ja komentoja kuin Python-kielessä ja toiminnot ovat listatyyppisten olioiden sisällä. Abaqus Scripting -käyttöliittymää käyttämällä päästään käsiksi Abaqus/CAE:n sisältämään toiminnallisuuteen ilman graafisen käyttöliittymän GUI käyttöä. Käyttöliittymän avulla voi mm. luoda ja muuttaa mallia; luoda, muuttaa ja suorittaa analyysiajoja; lukea ja kirjoittaa tulostiedostoa (odb) sekä tarkastella tuloksia.

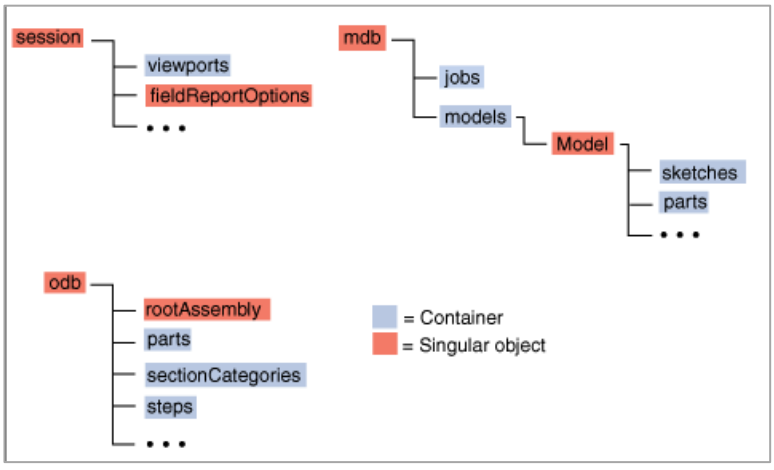
5 OHJELMOINTI ABAQUS SCRIPTING -LIITTYMÄLLÄ

Abaqus Scripting -käyttöliittymällä tehtävät sovellukset muodostuvat Python-kielen komennoista. Ennen Abaqus Scripting -sovellusten kirjoittamisen aloittamista onkin tutustuttava Python-kielen perusteisiin. Python on kielenä helposti ymmärrettävä ja omaksuttava, vaikka aiempaa ohjelmointikokemusta ei olisi.

Abaqus Scripting -käyttöliittymä laajentaa Python-kieltä noin viidelläsadalla tietotyypillä, joiden välillä on erilaisia yhteyksiä (Abaqus 6.12 Documentation 2014). Tietotyypeistä muodostetaan hierarkkinen oliorakenne. Oliot voivat esimerkiksi viitata toisiinsa tai ne voivat olla toisen olion sisäisiä olioita. Abaqus Scripting -sovelluksen oliot ja niiden sisältämät metodit ja tietokentät vaativat perehtymistä, jotta halutut tiedot ja toiminnot löytyvät ja toteutuvat luotettavasti. Ohjelmoinnissa hyvänä apuna ovat välitulostukset olioiden sisällön tarkistamiseksi.

5.1 Pääoliot

FEM-malli muodostuu Abaqus Scripting -käyttöliittymän olioista. Malli on jaettu kolmen pääolion alle, joita ovat Session, Mdb ja Odb (kuva 13). Olioiden muodostama rakenne myötäilee graafisen käyttöliittymän GUI mallin rakennepuuta, joten sen avulla voidaan pyrkiä hahmottamaan olioiden sisältämää tietoa ja funktioita. Koko mallirakennetta ei pystytä esittämään yhdellä kuvalla, koska olioita on paljon. (Abaqus 6.12 Documentation 2014)



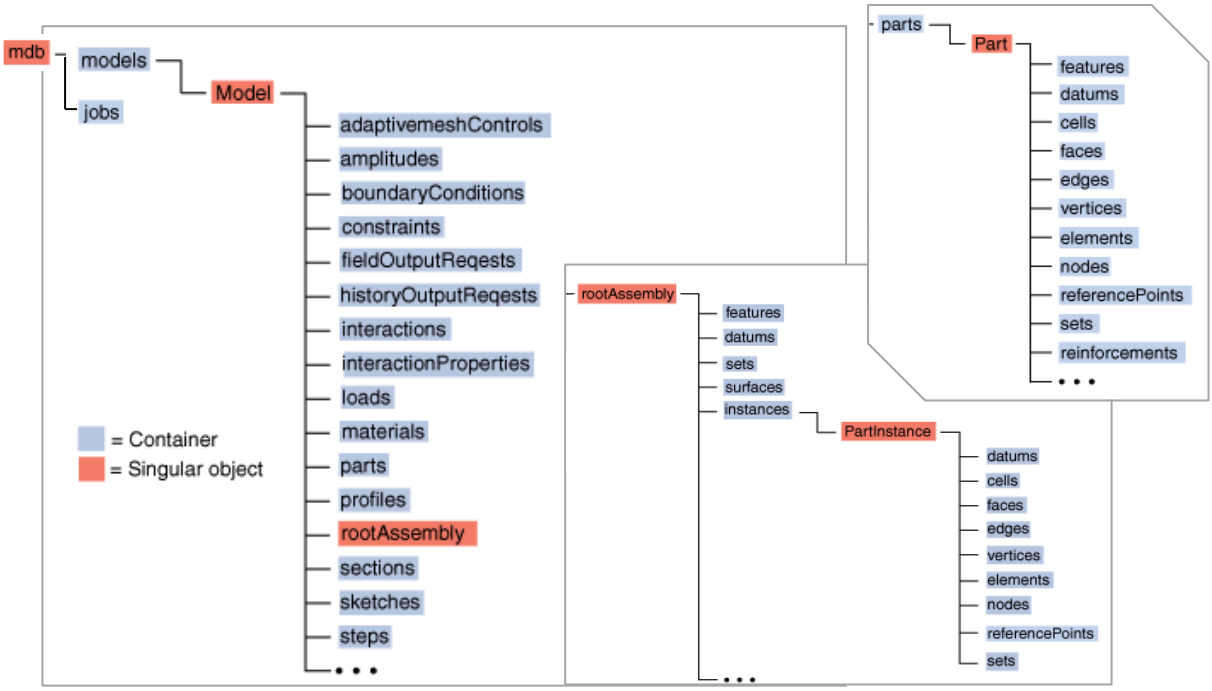
KUVA 13. Session-, Mdb- ja Odb-oliot sekä niiden sisältämät yleisimmin käytetyt oliot (Abaqus 6.12 Documentation 2014)

5.1.1 Session

Session-olion alla sijaitsevat toiminnot, jotka määrittelevät tapauskohtaisesti istunnon asetukset. Nämä toiminnot eivät välity muihin istuntoihin. Ne määrittelevät mm. raportointiin ja ikkunaan liittyviä asetuksia. Session-oliota voidaan käyttää esimerkiksi istunnon nimeämiseen: *session.viewports[name='Viewport-1']*.
(Abaqus 6.12 Documentation 2014)

5.1.2 Mdb

Mdb-olio sisältää oliot, joita käytetään mallinnusprosessin vaiheisiin, esimerkiksi mallin geometrian luontiin, suoritettavan analyysin, rasituksien ja reunaehtojen määrittämiseen ja mallin ajoon (kuva 14). Vain tulosten tarkastelussa ei käytetä Mdb-oliota. Käytettäessä Mdb-oliota tulee sovelluksen alkuun lisätä lause *from abaqus import* *. Mdb-olion yleisimmin käytetyt oliot ovat Part ja RootAssembly. Part-oliolla voidaan luoda mallin geometria esimerkiksi seuraavasti:
mdb.model['Example'].Part(name='Beam', dimensionality=THREE_D,...).
(Abaqus 6.12 Documentation 2014)



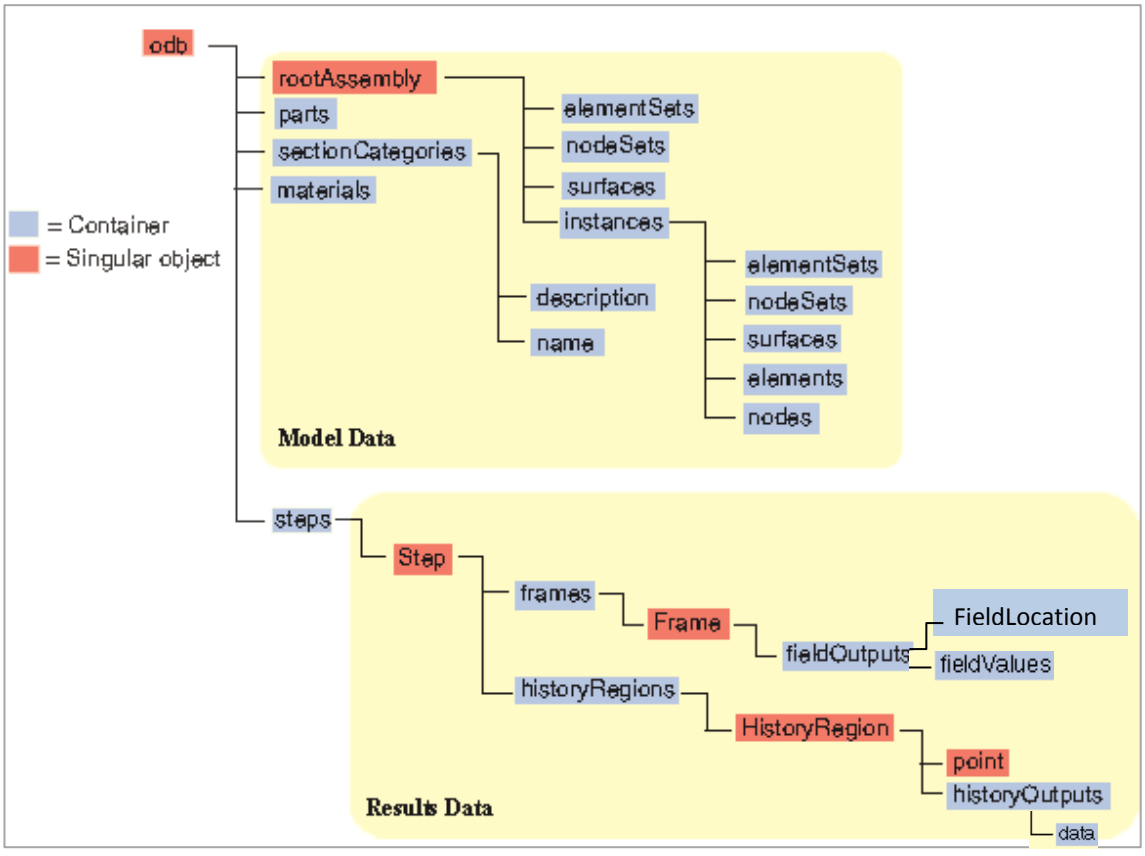
KUVA 14. Mdb-olion rakenne (Abaqus 6.12 Documentation 2014)

Mdb-oliossa voi olla useita Job-olioita. Job-olio viittaa Model-olioon, mutta ei ole sen omistama. Mallin laskenta käynnistetään Job-oliolla seuraavasti:
mdb.Job(name='JobName', model='Example', type=ANALYSIS, ...).

5.1.3 Odb

Odb-olio (*output database*) tallentuu odb-tiedostoon, joka sisältää sekä mallin tiedot että saadut tulokset (kuva 15). Tallentuvia mallin tietoja ovat muun muassa solmukoordinaatit, elementtityypit ja aluemäärittelyt (*sets*). Tulokset sisältävät suoritetussa analyysissa ratkaistujen suureiden arvot, esimerkiksi siirtymät, jännitykset ja venymät. (Abaqus 6.12 Documentation 2014)

Käytettäessä Odb-oliota, tulee sovelluksen alkuun lisätä lauseet *from odbAccess import ** ja *from abaqusConstants import **. Jotta tuloksia voidaan kutsua odb-tiedostosta, tulee tiedosto ensin avata:
odb = openOdb(path='c:/data/abaqusModel/example.odb').



KUVA 15. Odb-olion rakenne (Abaqus 6.12 Documentation 2014)

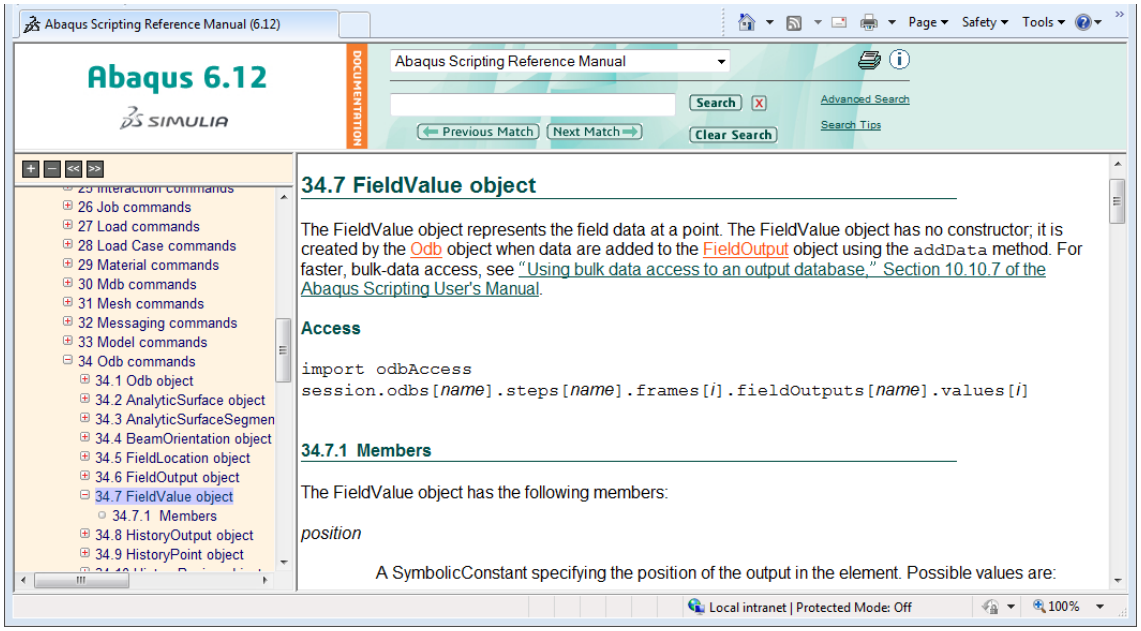
Odb-olio sisältää useita tietorakenteita. Esimerkiksi OdbStep-olio on Odb-olion alainen ja Frame-olio on OdbStep-olion alainen. FieldOutput-oliolla on kaksi jäsentä, FieldValue ja FieldLocation. FieldValue-jäsen sisältää muun muassa solmu- (NodeLabel) ja elementtinumerot (ElementLabel), joiden avulla tulokset saadaan haettua siten, että tiedetään sekä tulos että sen sijainti. Esimerkiksi siirtymät ja siirtymiä vastaavat solmunumerot löytyvät odb-tiedostosta seuraavasti:

data = odb.steps['Step-1'].frames['Frame-1'].fieldOutputs['U'].values

node = odb.steps['Step-1'].frames['Frame-1'].fieldOutputs['U']. values[x].nodeLabel.

5.2 Abaqus Scripting Reference -manuaali

Kaikkien Python-kieltä täydentävien Abaqus Scripting -käyttöliittymän olioiden määrittelyt löytyvät Abaqus Scripting Reference -manuaalista (kuva 16). Manuaalissa kuvataan oliot (esim. Odb, OdbStep, OdbFrame, FieldOutput, FieldValue) sekä niiden sisältämät metodit eli jäsenfunktiot (esim. addData(), getSubset()) ja tietokentät (esim. position, nodeLabel, mises, tresca).



KUVA 16. Abaqus Scripting Reference -manuaali, FieldValue-olion määrittely

5.3 Sovellusten ajo

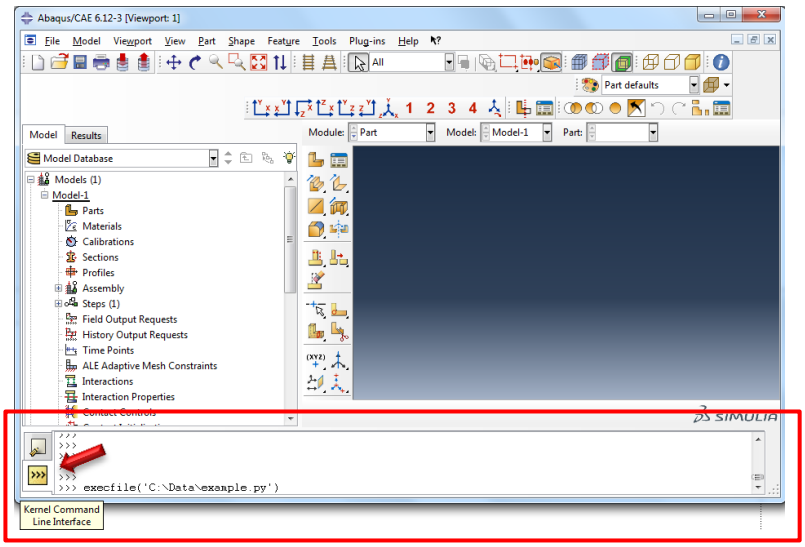
Abaqus Scripting -sovelluksia voidaan ajaa Abaqus/CAE GUI ja CLI -käyttöliittymien kautta ohjelman sisällä tai komentoikkunan (Command Prompt) kautta ilman, että Abaqus-ohjelman graafinen käyttöliittymä GUI on aktiivinen.

5.3.1 Ajo GUI:ssa

Sovellus voidaan ajaa Abaqus/CAE GUI -käyttöliittymällä valitsemalla *File* > *Run Script*. Tätä kautta voidaan ajaa sovelluksia, jotka suorittavat koko simulointiprosessin sekä osittaisia sovelluksia. Osittaiset sovellukset voivat esimerkiksi luoda mallille vain materiaalin. Tällaisia sovelluksia ei voida ajaa itsenäisesti ilman, että muokattava malli on käsiteltävänä GUI:ssa, koska ne eivät luo kaikkia simuloinnin edellyttämiä osa-alueita, kuten mallin geometriaa, kuormituksia ja ajoa. (Puri 2011, 35)

5.3.2 Ajo CLI-käyttöliittymällä

Sovellus voidaan ajaa Abaqus/CAE CLI -käytöliittymällä (*command line interface*), joka sijaitsee graafisen käyttöliittymän alalaidassa (kuva 17). CLI-liittymässä sovellus voidaan ajaa Python-käskyillä. Koodin tulee sisältää lauseet *from abaqus import** ja *from abaqusConstants import**. Jos tiedosto sijaitsee työhakemistossa, sovellus voidaan ajaa käskyllä execfile(*”tiedostonimi.py”)*, tai antamalla tiedostonimen lisäksi myös Python-sovelluksen tiedostopolku. (Puri 2011, 39)



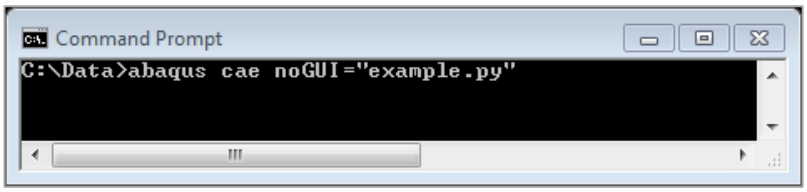
KUVA 17. CLI -käyttöliittymä (Command Line Interface)

5.3.3 Ajo Windows-käyttöjärjestelmän komentoikkunassa

Jotta sovelluksen voi ajaa komentoikkunan kautta, tulee tietokoneen olla Abaqus-ajokelpoinen. Systeemipolkuun tulee lisätä ”C:\SIMULIA\Abaqus\Commands” -hakemisto, jos sitä ei ole lisätty sinne Abaqus-asennuksen yhteydessä. Hakemistosta löytyvät komennot, joilla Abaqus käynnistyy komentoriviltä. (Puri 2011, 36)

Mikäli sovellus käyttää Abaqus/CAE-moduulia, se täytyy tulkata Abaqus/CAE kernelillä. Tällöin sovellus voidaan ajaa Command Prompt:in kautta (kuva 18) käskyllä *abaqus cae noGUI=”tiedostonimi.py”* (Abaqus 6.12 Documentation 2014). Käsky *noGUI* estää graafisen käyttöliittymän aukeamisen prosessin aikana. Abaqus/CAE -moduulia käytetään, kun muokataan mallia tai mitä tahansa analyysiin liittyvää osuutta. Tällöin käytetään Mdb- ja Session-olioita.

Mikäli Abaqus/CAE-moduulia ei käytetä sovelluksessa, se tulkataan Python-tulkin kautta ja se voidaan ajaa komentoriviltä käskyllä *abaqus python ”tiedostonimi.py”*. Koodin tulee lisäksi sisältää lause *from odbAccess import* * (Abaqus 6.12 Documentation 2014). Abaqus/CAE-moduulia ei tarvita luettaessa tuloksia odb-tiedostosta, jolloin käytetään Odb-olioita.



KUVA 18. Sovelluksen ajo Command Prompt:ssa Windows-käyttöjärjestelmässä

5.4 Tulosten käsittely

Tuloksia voidaan tarkastella piirtämällä kuvaajia Abaqus Scripting -käyttöliittymän XYPlot-moduulien avulla suoraan odb-tiedostosta luettavista tuloksista. Kuvaajia voidaan piirtää myös saatujen tekstitiedostojen sisältämistä tuloksista ulkopuolisella tulostenkäsittelyohjelmalla tai Pythonilla, esimerkiksi erillisen kuvaajien piirtämiseen tarkoitetun Matplotlib-kirjaston Pylab-moduulin avulla. Tällöin tulosten käsittely kannattaa eriyttää Abaqus Scripting -käyttöliittymästä, koska käytettävän Abaqus-version sisältämä Python-versio ei välttämättä tue haluttujen ulkopuolisten kirjastojen käyttöä (kuten Matplotlib), jotka eivät sisälly Pythonin valmiiseen luokkakirjastoon.

6 PYTHON-SOVELLUS

Tässä työssä toteutettiin Python-sovellus, joka käsittelee Abaqus-ohjelmalla tehtyä mallia sekä poimii odb-tiedostossa olevia tuloksia. Sovellus toteutettiin teksti- ja lähdekoodieditorilla Notepad++. Työssä käytettiin Abaqus-ohjelman versiota 6.12-3, jonka sisältämä Python on versio 2.6.2. Toteutetun sovelluksen lähdekoodi ja alkuarvotiedosto on esitetty liitteissä, jotka eivät ole julkisia.

Toteutetulla Python-sovelluksella voidaan poimia maksimisiirtymät ja -jännitykset automatisoidusti odb-tulostiedostosta sekä parametrisoida mallin materiaalia. Sovelluksella voidaan tarkastella halutun alueen tuloksia muodostamalla Abaqus/CAE GUI:ssa tarkasteltavasta kohteesta alue (*set*) ja tarkastelemalla tämän alueen maksimiarvoja tai parametrisoimalla vain tätä aluetta. Sovellus toimii erilaisille malleille, kuten sauva-, palkki, kuori- ja solidimalleille. Sovellus toimii myös solidirakenteisille malleille, jotka perustuvat orpoon verkkoon. Orpo verkko muodostuu solmuista, elementeistä ja pinnoista ilman alkuperäistä geometriaa, jolloin esimerkiksi materiaaliominaisuudet tulee määritellä elementeille geometrisen mallin sijasta.

6.1 Maksimiarvojen poiminta odb-tiedostosta

Sovellus poimii siirtymien ja jännitysten maksimiarvot. Sovellus toteutettiin siten, että maksimiarvoja voidaan poimia koko mallista tai rajatummalta alueelta. Rajattu alue (*set*) tulee määritellä malliin sitä luotaessa.

Siirtymien maksimiarvo voidaan poimia *x*-, *y*- ja *z*-suuntien arvoista, koko tutkitun alueen arvoista maksimisiirtymänä tai itseisarvojen maksimina (*magnitude*).

Maksimijännitys voidaan poimia Mises- tai Tresca -vertailujännityksenä. Lisäksi voidaan määritellä, mistä kohtaa elementtiä jännitysarvot on laskettu, kuten esimerkiksi integrointipisteissä, elementin keskipisteessä tai solmupisteissä.

6.2 Materiaalin parametrisointi

Sovelluksella voidaan parametrisoida mallin materiaalia. Sovellus muodostaa mallille materiaalin ja suorittaa laskennan annetuilla parametriarvoilla (kimmomoduuli ja Poissonin vakio). Saadut tulokset poimitaan odb-tiedostosta automatisoidusti parametrisoinnin aikana ja tallennetaan tekstitiedostoon.

Parametrisointi voidaan tehdä koko mallille tai vain tiettyyn osaan mallia. Mikäli osat on tehty erillisinä kappaleina (*part*), voidaan parametrisointi osoittaa vain tietylle kappaleelle. Muutoin voidaan malliin tehdä setti, jolle materiaalin parametrisointi suoritetaan.

Parametrisointi voidaan toteuttaa ilman Abaqus/CAE:n graafisen käyttöliittymän avaamista.

7 PYTHON-SOVELLUKSEN AJO KOEMALLEILLA

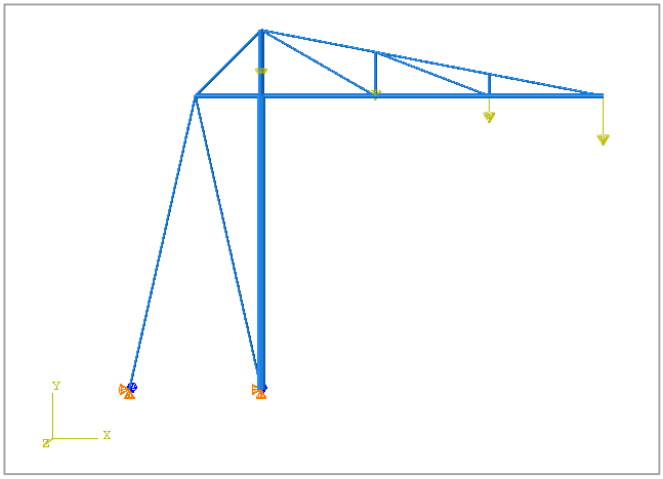
Toteutetun Python-sovelluksen toimivuutta tarkasteltiin Abaqus-ohjelman versiolla 6.12-3, ja yksikköjärjestelmänä käytettiin metristä järjestelmää [N, m]. Tarkastelulla haluttiin todentaa Python-sovelluksen toimivuutta erityyppisille FEM-malleille.

7.1 Maksimiarvojen poiminta

Maksimisiirtymät ja -jännitykset odb-tiedostosta poimivan Python-sovelluksen toimivuutta tarkasteltiin ristikko-, kehä-, levy- ja solidirakenteiden avulla sekä lineaarisella että epälineaarisella analyysillä. Tuloksia tarkasteltiin sekä Abaqus/CAE GUI -käyttöliittymällä että toteutetulla Python-sovelluksella.

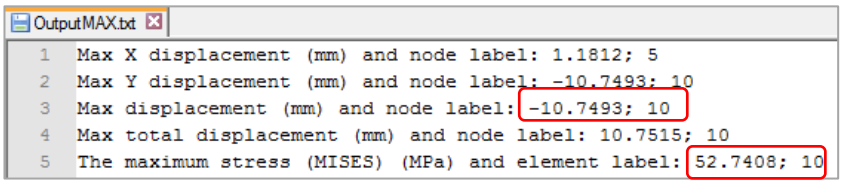
7.1.1 Ristikkorakenne

Tarkasteltavaksi ristikkorakenteeksi valittiin katsomoristikko (kuva 19), jonka materiaali on terästä ja sauvat poikkileikkaukseltaan pyöreät. Materiaalin kimmomoduuli on 210 GPa ja Poissonin vakio 0,3. Elementtityyppinä käytetään 2-solmuista sauvaelementtiä T2D2. Katsomoristikon katokseen kohdistuu kuvan mukaisesti neljä pystysuuntaista pistevoimaa, ja ristikko on jäykästi tuettu kahdesta alimmasta kohdasta. Maksimijännitys laskettiin elementtien integrointipisteiden Mises-vertailujännityksen arvoista.

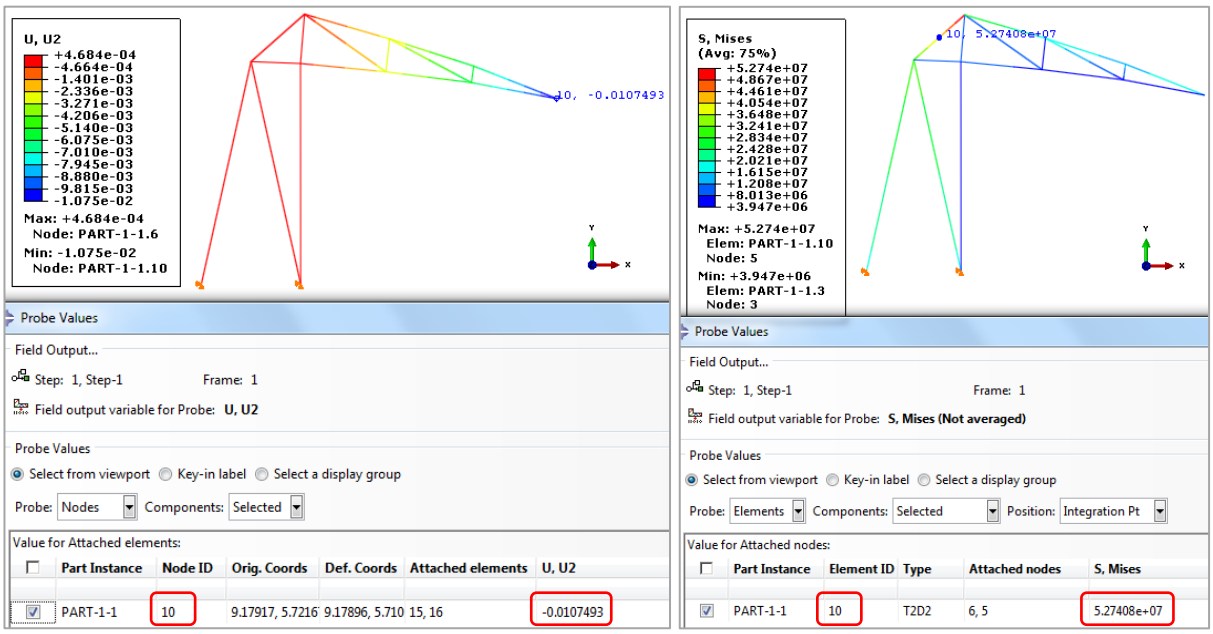


KUVA 19. Tarkastellun ristikkorakenteen tuenta ja kuormitus

Python-sovelluksella lineaarisesti tarkastellun katsomoristikon maksimisiirtymä (*U*) on 10,7493 mm ja se sijaitsee solmussa 10. Maksimijännitys (*S*) löytyy elementistä 10 ja on 52,7408 MPa (kuva 20). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaavat arvot (kuva 21).



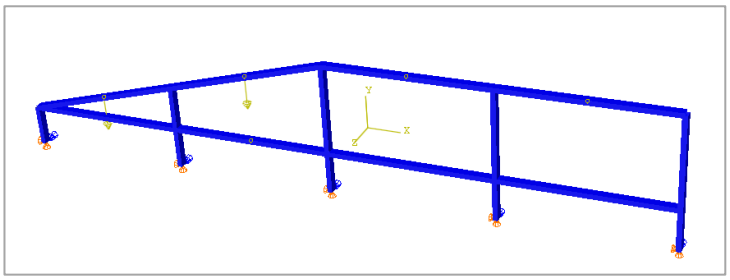
KUVA 20. Katsomoristikon maksimisiirtymät (*U*) (maksimisiirtymä solmussa 10) ja maksimijännitys (*S*, Mises) elementissä 10 Python-sovelluksella, lineaarinen analyysi



KUVA 21. Katsomoristikon maksimisiirtymä (*U*) solmussa 10 ja maksimijännitys (*S*, Mises) elementissä 10 Abaqus/CAE GUI -käyttöliittymällä, lineaarinen analyysi

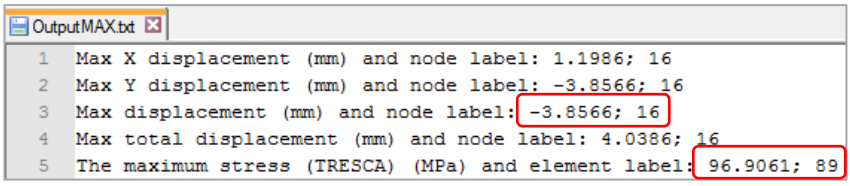
7.1.2 Kehärakenne

Kehärakenteena tarkasteltiin rakenneteräksestä tehtyä siirtorampin tukikehää (kuva 22). Materiaalin kimmomoduuli on 210 GPa ja Poissonin vakio 0,3. Elementtityyppinä käytetään 2-solmuista palkkielementtiä B21. Rakenne on jäykästi tuettu tukikehän alaosasta ja sitä kuormitetaan kahdella pystysuuntaisella pistekuormituksella.

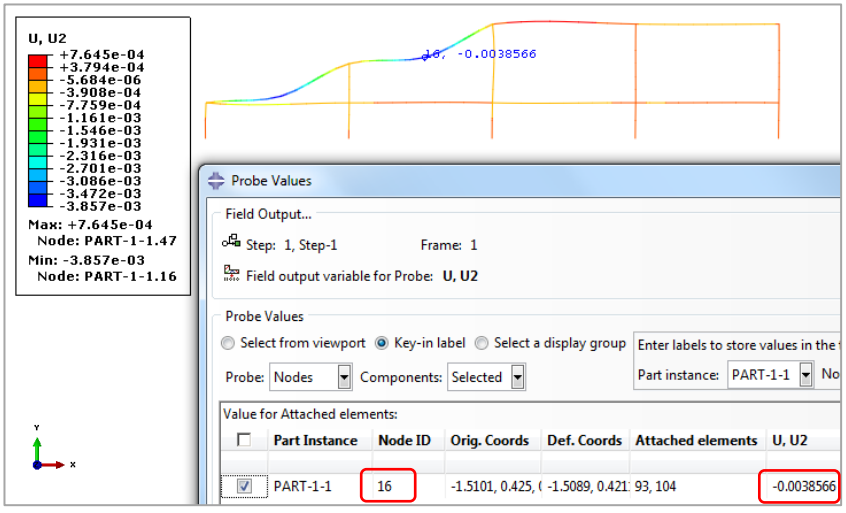


KUVA 22. Tarkastellun kehärakenteen tuenta ja kuormitus

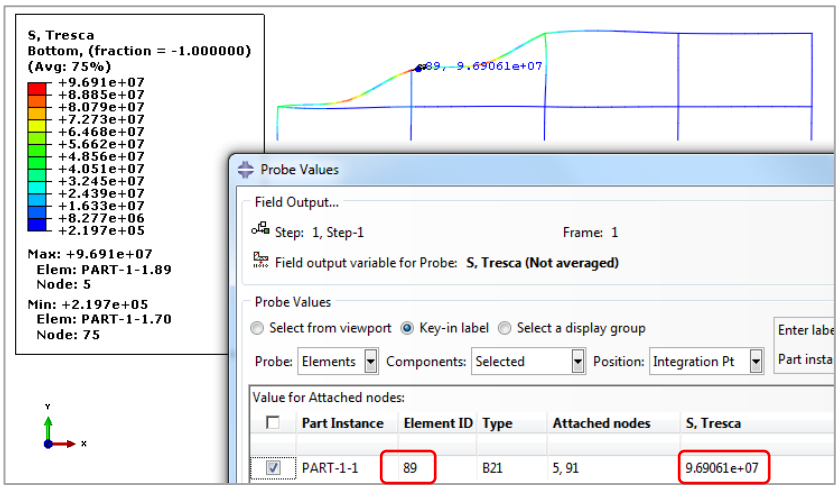
Python-sovelluksella lineaarisesti tarkastellun kehärakenteen maksimisiirtymä (*U*) on -3,8566 mm ja se sijaitsee solmussa 16. Maksimijännitys (*S*) löytyy elementistä 89 ja on 96,9061 MPa (kuva 23). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaavat arvot (kuva 24; kuva 25). Maksimijännitys laskettiin elementtien integrointipisteiden Tresca-vertailujännityksen arvoista.



KUVA 23. Kehärakenteen maksimisiirtymä (*U*) solmussa 16 ja maksimijännitys (*S, Tresca*) elementissä 89 Python-sovelluksella, lineaarinen analyysi

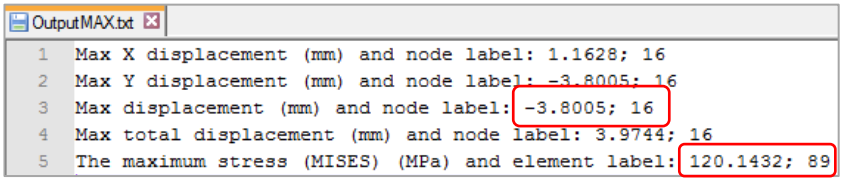


KUVA 24. Kehärakenteen maksimisiirtymä (*U*) solmussa 16 Abaqus/CAE GUI -käyttöliittymällä, lineaarinen analyysi

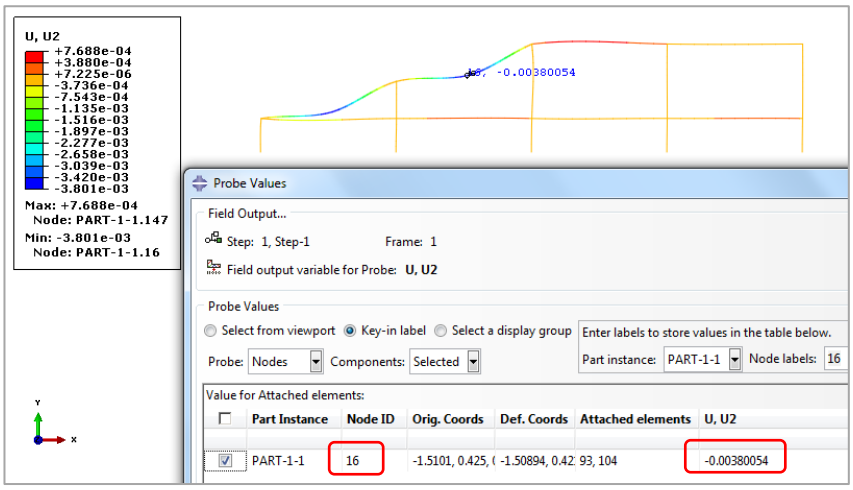


KUVA 25. Kehärakenteen maksimijännitys (*S, Tresca*) elementissä 89 Abaqus/CAE GUI -käyttöliittymällä, lineaarinen analyysi

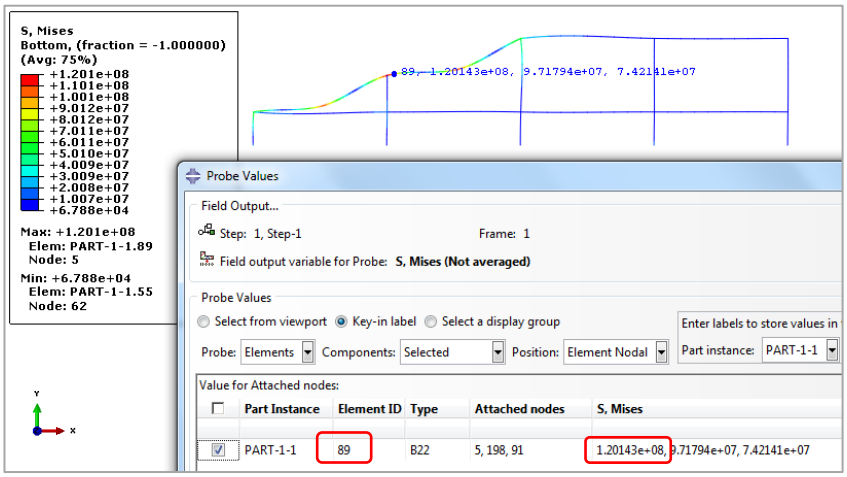
Kehärakennetta tarkasteltiin myös epälineaarisella analyysillä. Tällöin Python-sovelluksella tarkastellun kehärakenteen maksimisiirtymä (*U)* on -3,8005 mm ja se sijaitsee solmussa 16. Maksimijännitys (*S)* löytyy elementistä 89 ja on 120,1432 MPa (kuva 26). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaavat arvot (kuva 27; kuva 28). Maksimijännitys laskettiin Mises-vertailujännityksen arvoista. Abaqus ilmoitti koko rakenteelle elementin solmupisteiden (*element nodal*) arvoista löytyneen maksimijännityksen (kuva 28), joten vertailtavuuden vuoksi myös Python-sovelluksella haettiin koko rakenteen maksimijännitys elementin solmupisteistä.



KUVA 26. Kehärakenteen maksimisiirtymä (*U*) solmussa 16 ja maksimijännitys (*S*, Mises) elementissä 89 Python-sovelluksella, epälineaarinen analyysi



KUVA 27. Kehärakenteen maksimisiirtymä (*U*) solmussa 16 Abaqus/CAE GUI -käyttöliittymällä, epälineaarinen analyysi



KUVA 28. Kehärakenteen maksimijännitys (*S*, Mises) elementissä 89 Abaqus/CAE GUI -käyttöliittymällä, epälineaarinen analyysi

7.1.3 Levyrakenne

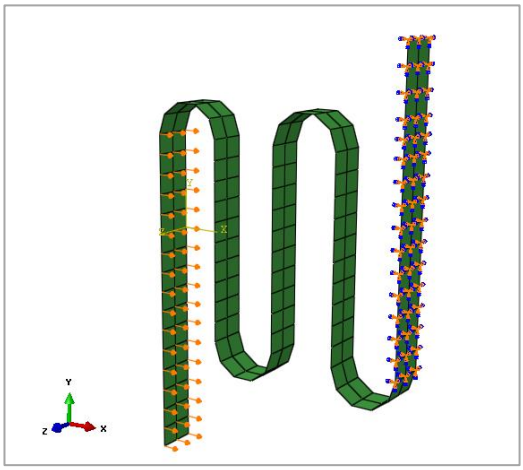
Levyrakennetta tarkasteltiin kahdella esimerkillä, joista toinen on jousimainen teräslevyrakenne ja toinen kumilevy, jossa on iso ja pieni reikä.

Lineaarisessa analyysissa tarkasteltu jousimainen levyrakenne on austeniittista ruostumatonta terästä (SS 302), jonka kimmomoduuli on 193 GPa ja Poissonin vakio 0,25. Plastisuusarvot on esitetty taulukossa 1. Elementtityyppinä käytetään 4-solmuista kuorielementtiä S4R (redusoitu integrointi). Rakenne perustuu orpoon verkkoon. Levyn

vasemman pään pystyosuudelle annetaan pakotettu *x*-suuntainen siirtymä ja oikea reuna on jäykästi tuettu (kuva 29).

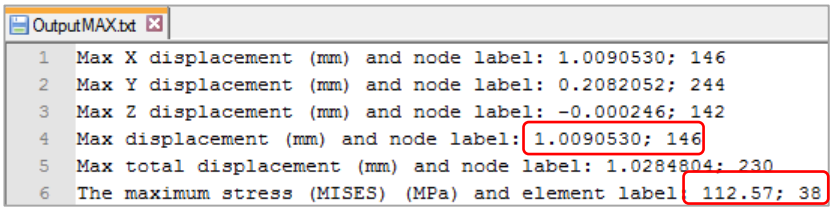
TAULUKKO 1. Teräksen (SS 302) plastisuusarvot

Myötöjännitys [GPa]	Pysyvä venymä [%]
241	0
280	2
320	3
380	5
420	7,5
500	20

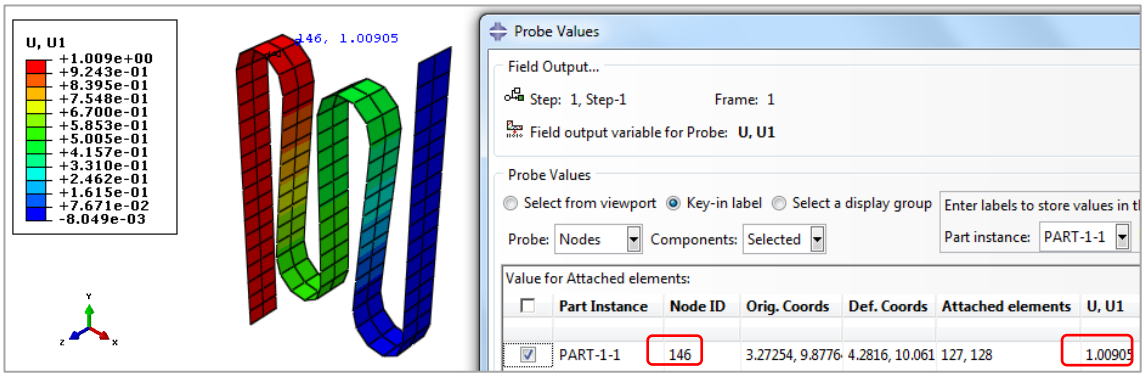


KUVA 29. Tarkastellun levyrakenteen tuenta ja kuormitus

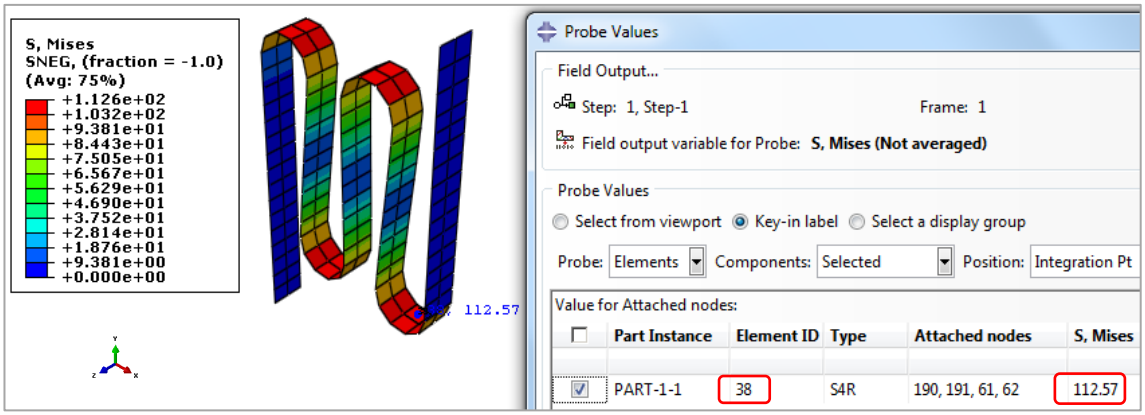
Python-sovelluksella lineaarisesti tarkastellun jousimaisen levyrakenteen maksimisiirtymä (*U*) on 1,009053 mm ja se sijaitsee solmussa 146. Maksimijännitys (*S*) löytyy elementistä 38 ja on 112,57 MPa (kuva 30). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaavat arvot (kuva 31; kuva 32). Tässä tapauksessa ohjelmassa Abaqus on yksikköjärjestelmänä [N, mm]. Maksimijännitys laskettiin elementtien integrointipisteiden Mises-vertailujännityksen arvoista.



KUVA 30. Levyrakenteen maksimisiirtymä (*U*) solmussa 146 ja maksimijännitys (*S*, Mises) elementissä 38 Python-sovelluksella, lineaarinen analyysi

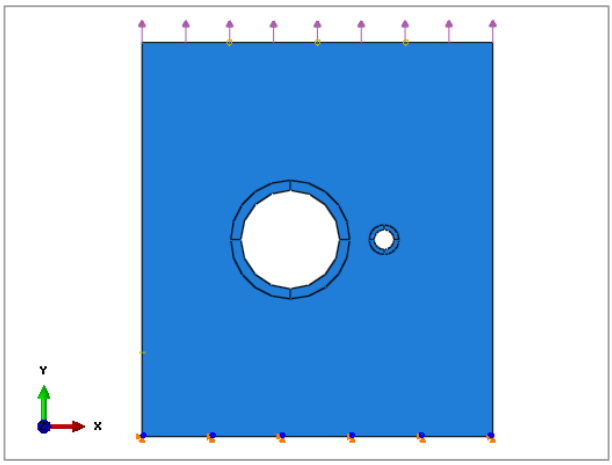


KUVA 31. Levyrakenteen maksimisiirtymä (*U*) (yksikkö mm) solmussa 146 Abaqus/CAE GUI -käyttöliittymällä, lineaarinen analyysi



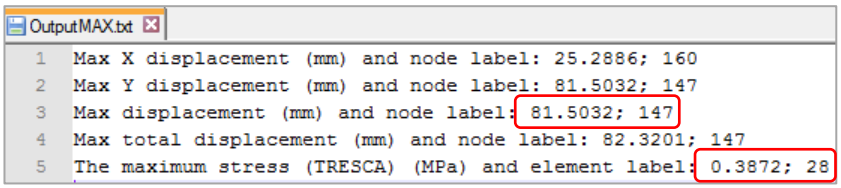
KUVA 32. Levyrakenteen maksimijännitys (*S*, Mises) (yksikkö MPa) elementissä 38 Abaqus CAE /GUI -käyttöliittymällä, lineaarinen analyysi

Materiaaliepälineaarisessa analyysissa levyrakenteena tarkasteltiin suorakulmiolevyä, jossa on iso ja pieni reikä. Levy on hyperelastista kumia, jonka materiaalimalli on 2-parametrinen Mooney-Rivlinin materiaali, jossa parametri C10 on 0,1724 MPa, C01 on 0,0483 MPa ja D1 on 0 MPa^{-1}. Elementtityyppinä käytetään 8-solmuista tasojännityselementtiä CPS8R (redusoitu integrointi). Levyn alareuna on jäykästi kiinnitetty ja yläreunassa on tasainen vetojännitys (kuva 33).

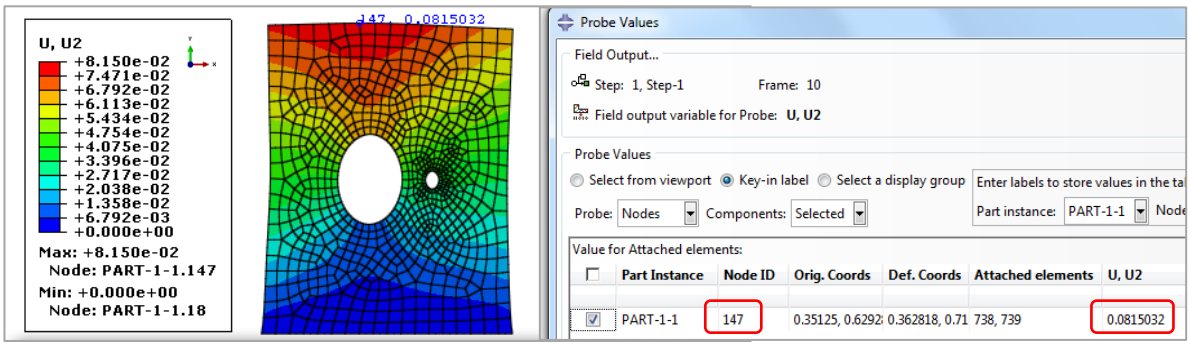


KUVA 33. Tarkastellun suorakulmiolevyn tuenta ja kuormitus

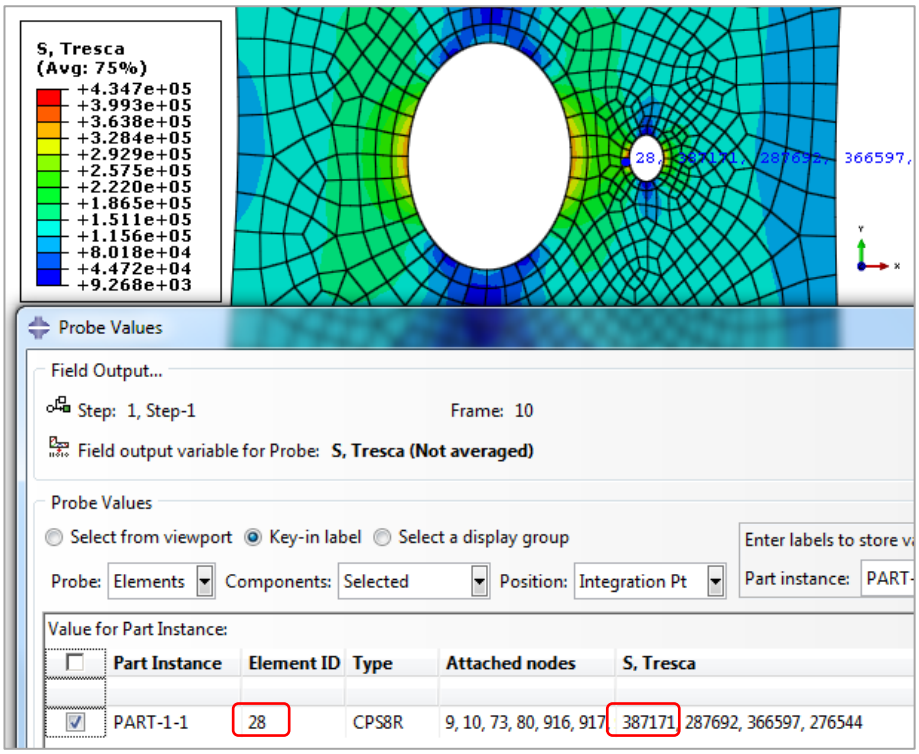
Python-sovelluksella epälineaarisesti tarkastellun levyrakenteen maksimisiirtymä (*U*) on 81,5032 mm ja se sijaitsee solmussa 147. Maksimijännitys (*S*) löytyy elementistä 28 ja on 0,3872 MPa (kuva 34). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaavat arvot (kuva 35; kuva 36). Maksimijännitys laskettiin elementtien integrointipisteiden Tresca-vertailujännityksen arvoista.



KUVA 34. Levyrakenteen maksimisiirtymät (*U*) (maksimisiirtymä solmussa 147) ja maksimijännitys (*S, Tresca*) elementissä 28 Python-sovelluksella, epälineaarinen analyysi



KUVA 35. Levyrakenteen maksimisiirtymä (*U*) solmussa 147 Abaqus/CAE GUI -käyttöliittymällä, epälineaarinen analyysi

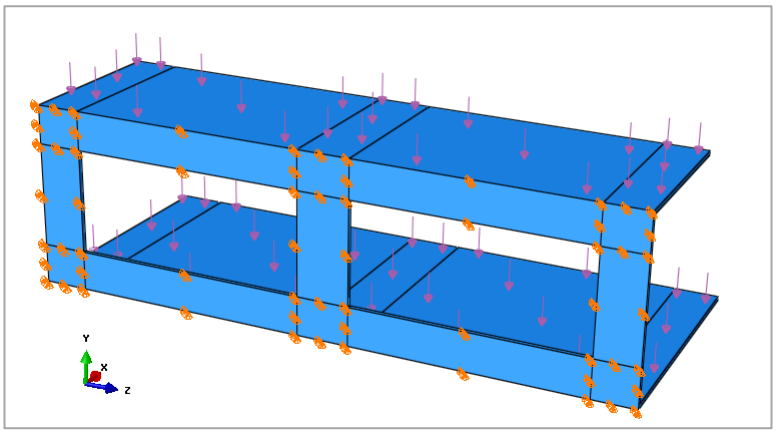


KUVA 36. Levyrakenteen maksimijännitys (*S, Tresca*) elementissä 28 Abaqus/CAE GUI -käyttöliittymällä, epälineaarinen analyysi

7.1.4 Solidirakenne

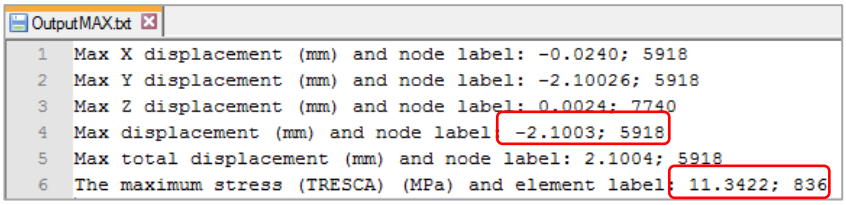
Solidirakennetta tarkasteltiin hyllyrakenteen sekä laippaliitoksen avulla.

Lineaarisessa analyysissä tarkastelun kohteena oli alumiininen hyllyrakenne, jonka kimmomoduuli on 70 GPa ja Poissonin vakio 0,33. Elementtityyppinä käytetään 20-solmuista tiiliskivielementtiä C3D20R (redusoitu integrointi). Rakenteen y-suuntainen takaseinä on jäykästi tuettu ja hyllyihin kohdistetaan tasaisen paineen kuormitus (kuva 37).

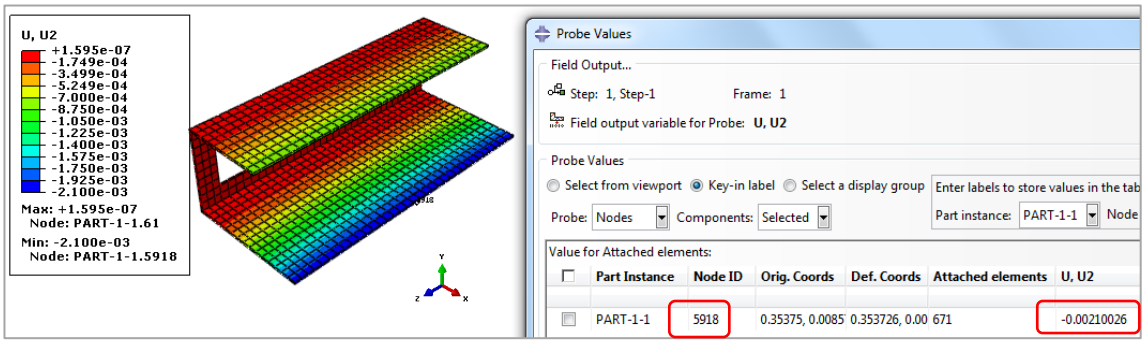


KUVA 37. Tarkastellun hyllyrakenteen tuenta ja kuormitus

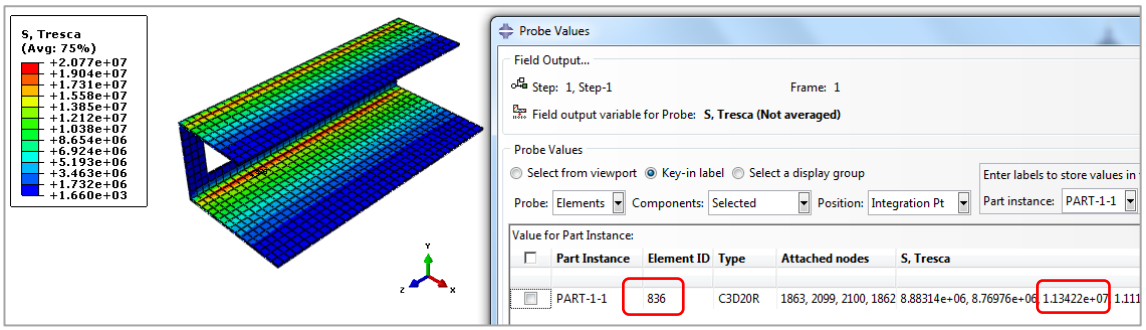
Python-sovelluksella lineaarisesti tarkastellun hyllyrakenteen maksimisiirtymä (*U*) on -2,1003 mm ja se sijaitsee solmussa 5918. Maksimijännitys (*S*) löytyy elementistä 836 ja on 11,3422 MPa (kuva 38). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaavat arvot (kuva 39; kuva 40). Maksimijännitys laskettiin elementtien integrointipisteiden Tresca-vertailujännityksen arvoista.



KUVA 38. Solidirakenteen maksimisiirtymä (*U*) solmussa 5918 ja maksimijännitys (*S, Tresca*) elementissä 836 Python-sovelluksella, lineaarinen analyysi

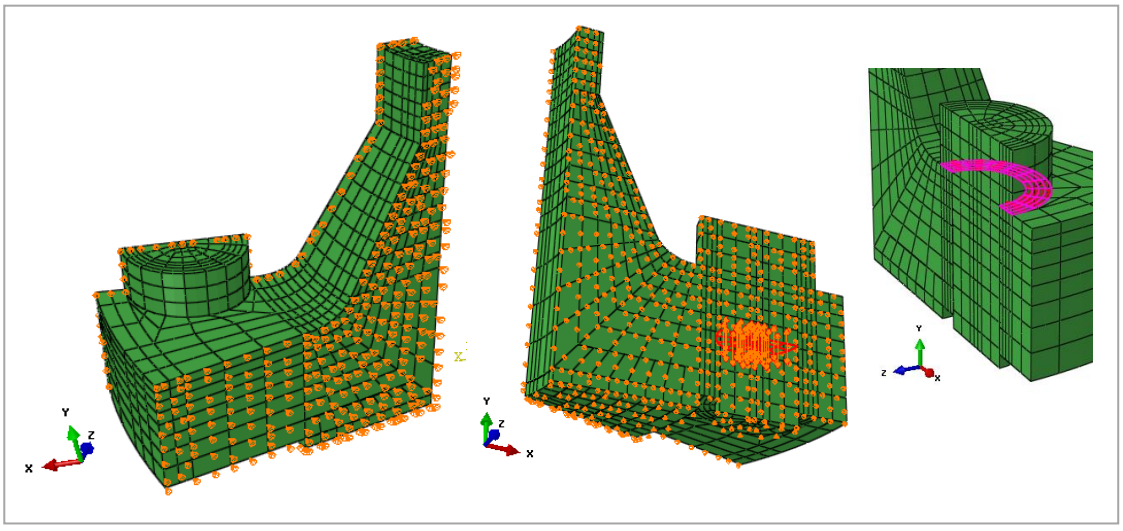


KUVA 39. Solidirakenteen maksimisiirtymä (*U*) solmussa 5918 Abaqus/CAE GUI -käyttöliittymällä, lineaarinen analyysi



KUVA 40. Solidirakenteen maksimijännitys (*S, Tresca*) elementissä 836 Abaqus/CAE GUI -käyttöliittymällä, lineaarinen analyysi

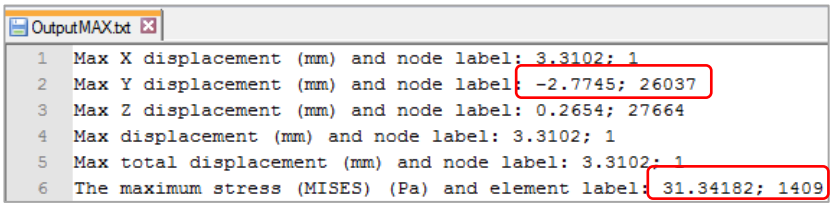
Epälineaarisessa analyysissä tarkasteltiin teräksisen laippaliitoksen jännityksiä ja siirtymiä. Laipan ja pultin materiaalin kimmomoduuli on 206 GPa ja Poissonin vakio 0,3. Laipan elementtityyppinä käytetään 20-solmuista tiiliskivielementtiä C3D20R ja pultissa 27-solmuista tiiliskivielementtiä C3D27R (redusoitu integrointi). Rakenne perustuu orpoon verkkoon. Tarkastelussa otetaan huomioon symmetrisyys (kuva 41). Rakenteen sivujen *z*-suuntaiset siirtymät sekä pultin ja holkin alapinnan *y*-suuntaiset siirtymät on estetty. Lisäksi pultin ja laipan välille on määritelty kontaktipinta (kuva 41). Kuormitus kohdistetaan pulttiin.



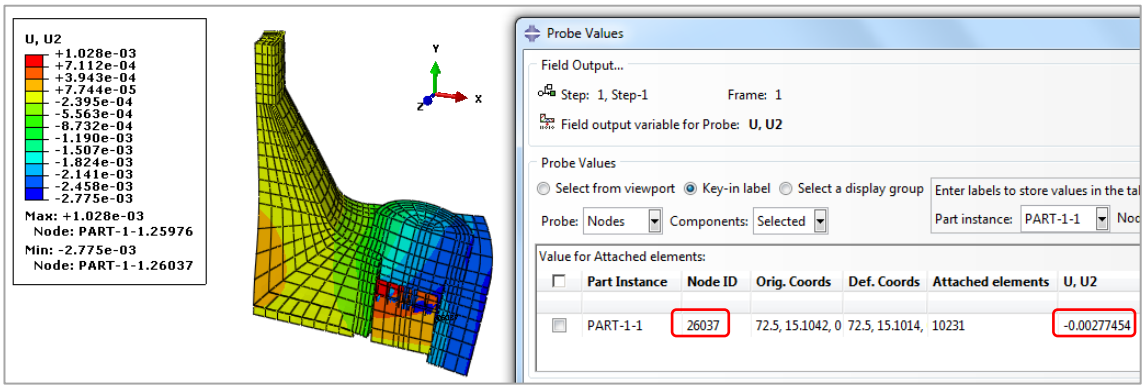
KUVA 41. Tarkastellun laippaliitoksen tuenta ja kuormitus

Solidirakenteen maksimijännitystä tarkasteltiin koko rakenteesta sekä pelkästä pultista Mises-vertailujännityksenä. Koko rakenteen maksimijännitys laskettiin elementtien solmupisteiden arvoista (kuva 42; kuva 44). Lisäksi haluttiin tarkastella vain pultin maksimijännitystä. Malliin tehtiin elementtisetti pultin alueelle ja tämän alueen tuloksista määritettiin Python-sovelluksella maksimijännitys elementin integrointipisteissä.

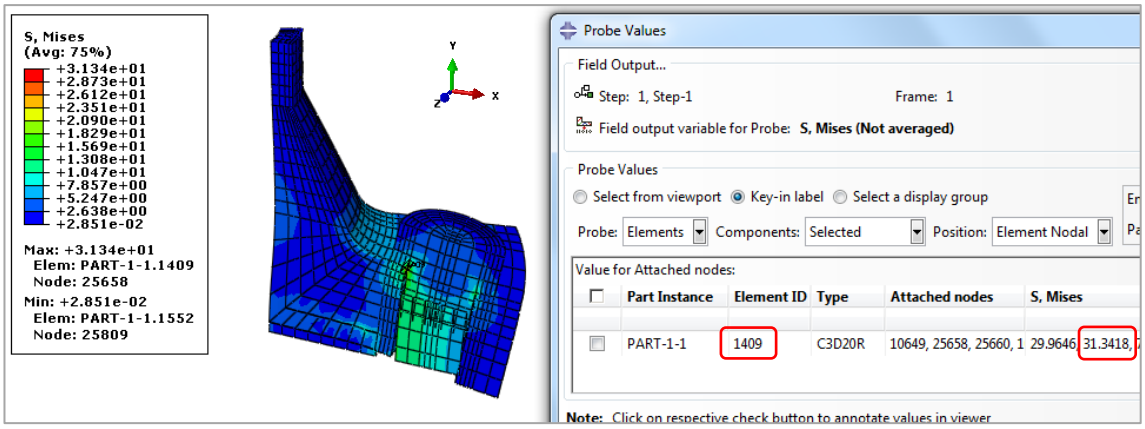
Python-sovelluksella epälineaarisesti tarkastellun koko laipparakenteen y-suuntainen maksimisiirtymä (*U*) on -2,7745 mm ja se sijaitsee solmussa 26037. Maksimijännitys (*S*) löytyy elementistä 1409 ja on 31,34182 MPa (kuva 45). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaavat arvot (kuva 43; kuva 44).



KUVA 42. Koko laippaliitoksen y-suuntainen maksimisiirtymä (*U*) solmussa 26037 ja elementtien solmupisteiden maksimijännitys (*S*, Mises) elementissä 1409 Python-sovelluksella, epälineaarinen analyysi

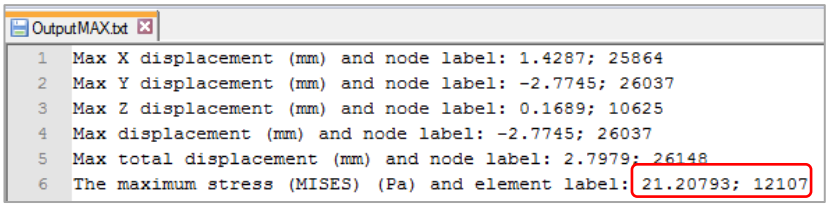


KUVA 43. Koko laippaliitoksen y-suuntainen maksimisiirtymä (*U*) solmussa 26037 Abaqus/CAE GUI -käyttöliittymällä, epälineaarinen analyysi

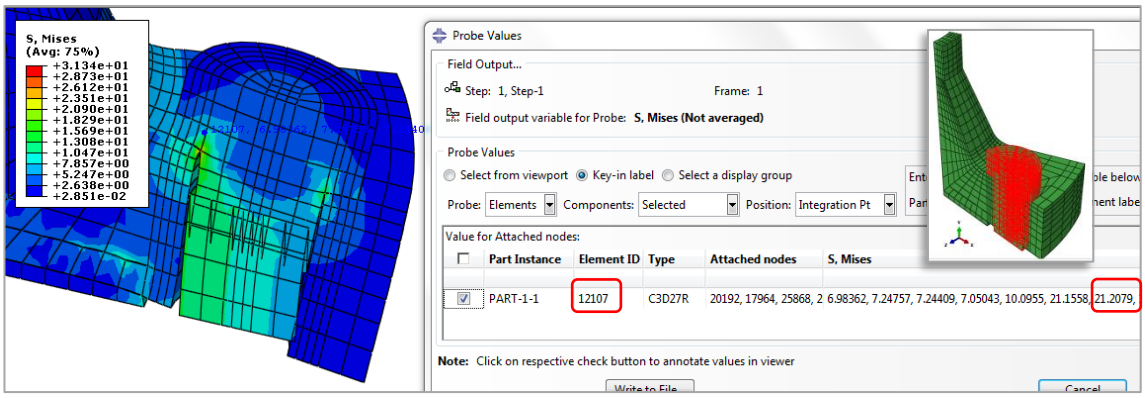


KUVA 44. Laippaliitoksen maksimijännitys (*S*, Mises) elementissä 1409 Abaqus/CAE GUI -käyttöliittymällä, epälineaarinen analyysi

Python-sovelluksella epälineaarisesti tarkastellun pultin y-suuntainen maksimijännitys (*S*) löytyy elementistä 12107 ja on 21,20793 MPa (kuva 45). Abaqus/CAE GUI -käyttöliittymällä saadaan vastaava arvo (kuva 46).



KUVA 45. Pultin maksimisiirtymät sekä maksimijännitys (*S*, Mises) elementissä 12107 Python-sovelluksella, epälineaarinen analyysi



KUVA 46. Pultin maksimijännitys (*S*, Mises) elementissä 12107 Abaqus/CAE GUI -käyttöliittymällä, epälineaarinen analyysi

7.2 Materiaalin parametrisointi

Materiaalin parametrisointi suoritettiin levyrakenteelle sekä laipparakenteen pultille. Epälineaariset analyysit suoritettiin siten, että malleja parametrisoitiin erikseen sekä kimmomoduulin että Poissonin vakion suhteen. Laipparakenteen pultille suoritettiin lisäksi parametrisointi molempien muuttujien suhteen. Laipparakenteen malli perustuu orpoon verkkoon.

7.2.1 Parametrisointi kimmomoduulin suhteen

Materiaalin parametrisointia kimmomoduulin suhteen tarkasteltiin levyrakenteisella mallilla (kuva 33). Tarkastellussa suorakulmiolevyssä on iso ja pieni reikä. Elementtityyppinä käytetään 8-solmuista tasojännityselementtiä (CPS8R) (redusoitu integrointi). Levyn alareuna on jäykästi kiinnitetty ja yläreunassa on tasainen vetojännitys.

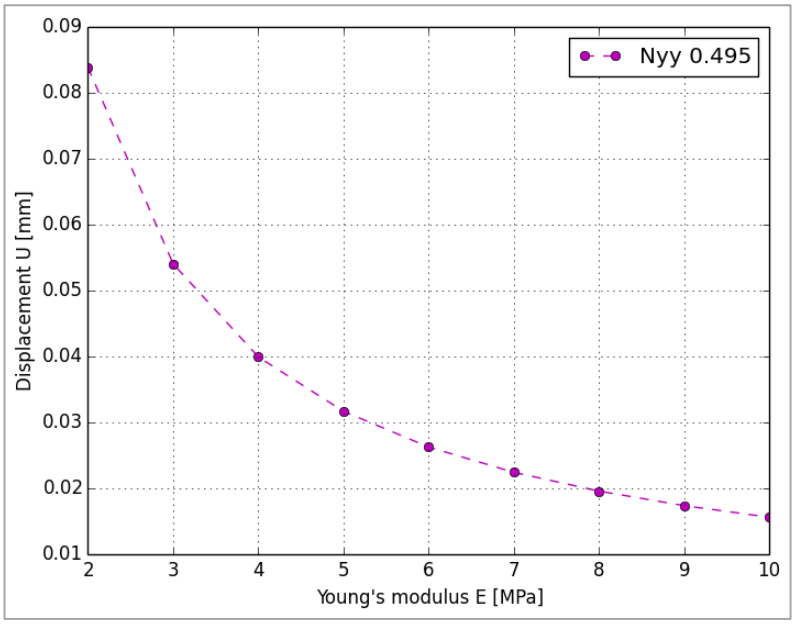
Analyysi on epälineaarinen ja tarkasteltuna materiaalina on kumi. Kimmomoduuli muuttuu välillä 2 - 10 MPa. Poissonin vakio on 0,495. Maksimijännitys laskettiin elementtien integrointipisteiden arvoista Mises-vertailujännityksenä.

Saadut tulokset tallentuivat tekstitiedostoon (kuva 47). Parametrisoinnin tulokset on esitetty kuvaajassa (kuva 48), joka toteutettiin erillisellä Python-sovelluksella.

Job-ShellKumi_Results.txt

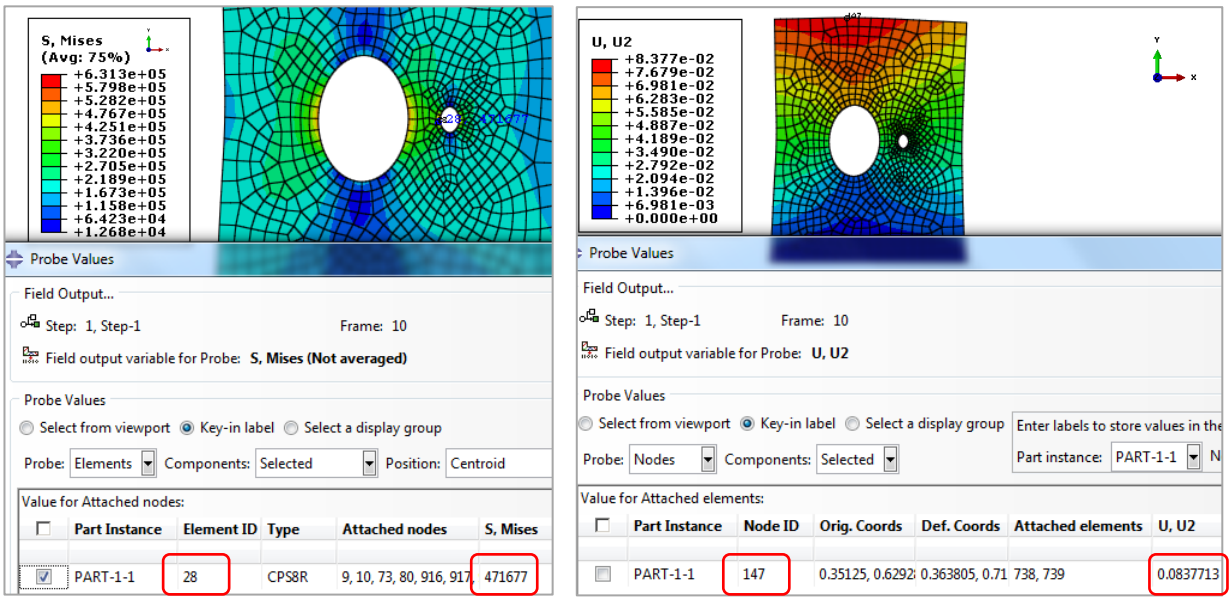
```
MAXIMUM DISPLACEMENT (Umax) AND STRESS (Smax)

E           Nyy         Node(Umax) Umax          Element(Smax) Smax
-------------------------------------------------------------------
2.00E+06    0.495       147        0.083771      28            4.72E+05
3.00E+06    0.495       2414       0.053971      28            4.44E+05
4.00E+06    0.495       2414       0.039852      28            4.32E+05
5.00E+06    0.495       2414       0.031597      10            4.26E+05
6.00E+06    0.495       2414       0.026177      10            4.25E+05
7.00E+06    0.495       2414       0.022346      10            4.24E+05
8.00E+06    0.495       2414       0.019494      10            4.23E+05
9.00E+06    0.495       2414       0.017287      10            4.22E+05
1.00E+07    0.495       2414       0.015530      10            4.22E+05


ALL MAXIMUM DISPLACEMENTS

E           Nyy         NodeX  Ux max         NodeY  Uy max         Node max  U max          NodeTot UTot max
-------------------------------------------------------------------------------------------------------------
2.00E+06    0.495       160    0.026284       147    0.083771       147       0.083771       147     0.084707
3.00E+06    0.495       2284   0.016731       2414   0.053971       2414      0.053971       2414    0.054389
4.00E+06    0.495       2284   0.012330       2414   0.039852       2414      0.039852       2414    0.040113
5.00E+06    0.495       2284   0.009772       2414   0.031597       2414      0.031597       2414    0.031784
6.00E+06    0.495       2284   0.008097       2414   0.026177       2414      0.026177       2414    0.026323
7.00E+06    0.495       2284   0.006914       2414   0.022346       2414      0.022346       2414    0.022465
8.00E+06    0.495       2284   0.006033       2414   0.019494       2414      0.019494       2414    0.019594
9.00E+06    0.495       2284   0.005351       2414   0.017287       2414      0.017287       2414    0.017374
1.00E+07    0.495       2284   0.004809       2414   0.015530       2414      0.015530       2414    0.015606
```

KUVA 47. Python-sovelluksella saatu tulostiedosto, kumilevyn parametrisointi kimmomoduulin suhteen. Maksimisiirtymä (*U*) solmussa 147 ja maksimijännitys (*S*) elementissä 28, kun kimmomoduuli on 2 MPa. Maksimisiirtymä (*U*) solmussa 2414 ja maksimijännitys (*S*) elementissä 10, kun kimmomoduuli on 10 MPa.

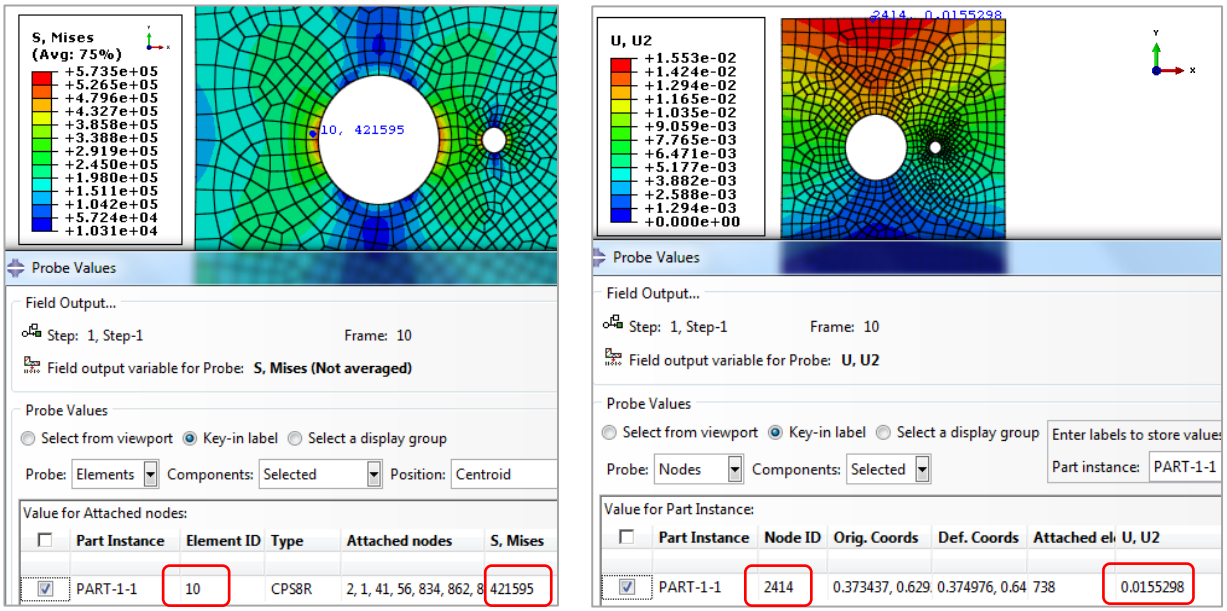


KUVA 48. Kumilevyn parametrisointi kimmomoduulin suhteen välillä 2 - 10 MPa

Tulokset tarkastettiin Abaqus/CAE GUI -käyttöliittymällä kahden eri kimmomoduulin suhteen. Sekä GUI -käyttöliittymällä että Python-sovelluksella kumilevyn maksimisiirtymä (*U*) kimmomoduulilla 2 MPa on 0,083771 mm ja se sijaitsee solmussa 147. Maksimijännitys (*S*) löytyy elementistä 28 ja on 0,471677 MPa (kuva 47; kuva 49). Kumilevyn maksimisiirtymä (*U*) kimmomoduulilla 10 MPa on molemmilla tavoilla tarkasteltuna 0,015530 mm ja se sijaitsee solmussa 2414. Maksimijännitys (*S*) löytyy elementistä 10 ja on 0,421595 MPa (kuva 47; kuva 50).



KUVA 49. Kumilevyn (kimmomoduuli 2 MPa) maksimisiirtymä (*U*) solmussa 147 ja maksimijännitys (*S*) elementissä 28 Abaqus/CAE GUI -käyttöliittymällä



KUVA 50. Kumilevyn (kimmomoduuli 10 MPa) maksimisiirtymä (*U*) solmussa 2414 ja maksimijännitys (*S*) elementissä 10 Abaqus/CAE GUI -käyttöliittymällä

7.2.2 Parametrisointi Poissonin vakion suhteen

Poissonin vakion suhteen materiaalia parametrisoitiin laippaliitoksessa (kuva 41), joka on solidirakenne ja perustuu orpoon verkkoon. Tarkastelun kohteena on laippaliitoksen pultti. Pultin alueelle muodostettiin solmu- ja elementtisetit, jotta saatiin parametrisoitua materiaalia vain halutulta kohdalta, koska malli on tässä tapauksessa yhtenäinen kappale. Analyysi on epälineaarinen ja tarkasteltuna materiaalina on polyvinyylikloridimuovi (PVC). Pultin alueella Poissonin vakio muuttuu välillä 0,26 - 0,34 ja kimmomoduuli on 2 GPa. Muualla laippaliitoksessa Poissonin vakio on 0,32 ja kimmomoduuli 2 GPa.

Laipan (kuva 41) elementtityyppinä käytetään 20-solmuista tiiliskivielementtiä C3D20R ja pultissa 27-solmuista tiiliskivielementtiä C3D27R (redusoitu integrointi). Tarkastelussa otetaan huomioon symmetrisyys. Rakenteen sivujen *z*-suuntaiset siirtymät ja pultin ja holkin alapinnan *y*-suuntaiset siirtymät on estetty. Lisäksi pultin ja laipan välille on määritelty kontaktipinta. Kuormitus kohdistetaan pulttiin.

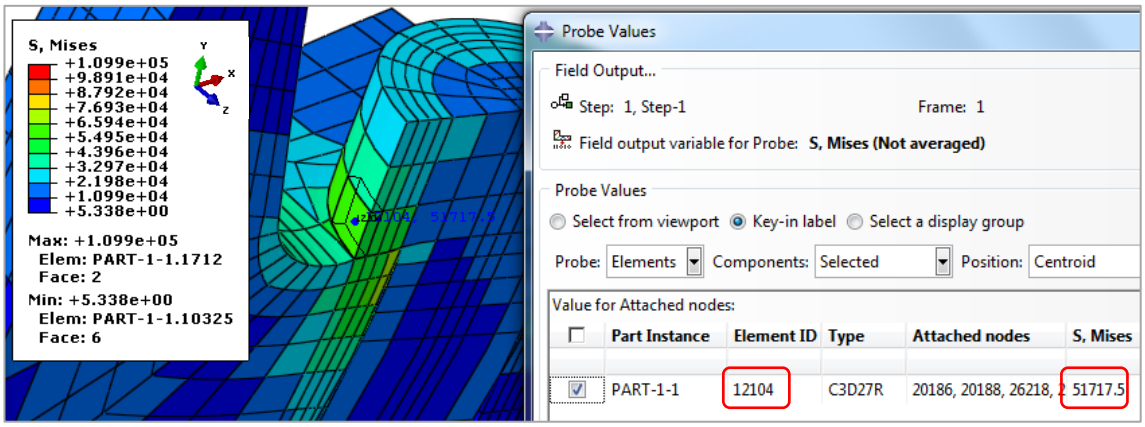
Maksimijännitys laskettiin elementtien integrointipisteiden arvoista Mises-vertailujännityksenä. Saadut tulokset tallentuivat tekstitiedostoon (kuva 51), josta tuloksia voidaan tarkastella halutusti.

Job-LaippaPVC_Results.txt

```
MAXIMUM DISPLACEMENT (Umax) AND STRESS (Smax)

E [Pa]      Nyy         Node(Umax) Umax [m]     Element(Smax) Smax [Pa]
-----------------------------------------------------------------------
2.00E+09    0.26        26148      0.001715900      12104      5.28218E+04
2.00E+09    0.28        26148      0.001697688      12104      5.24527E+04
2.00E+09    0.30        26148      0.001678715      12104      5.20849E+04
2.00E+09    0.32        26148      0.001658963      12104      5.17175E+04
2.00E+09    0.34        26148      0.001638407      12104      5.13498E+04


ALL MAXIMUM DISPLACEMENTS

E [Pa]      Nyy         NodeX  Ux max [m]    NodeY  Uy max [m]    NodeZ  Uz max [m]
-----------------------------------------------------------------------------------
2.00E+09    0.26        26148  -0.000906     26148  0.001716      10312  -0.000169
2.00E+09    0.28        26148  -0.000902     26148  0.001698      10312  -0.000166
2.00E+09    0.30        26148  -0.000898     26148  0.001679      10312  -0.000164
2.00E+09    0.32        26148  -0.000893     26148  0.001659      10312  -0.000162
2.00E+09    0.34        26148  -0.000888     26148  0.001638      10312  -0.000159


E [Pa]      Nyy       Node max   U max [m]   NodeTot UTot max [m]
-----------------------------------------------------------------
2.00E+09    0.26        26148    0.001716       26148   0.001940
2.00E+09    0.28        26148    0.001698       26148   0.001923
2.00E+09    0.30        26148    0.001679       26148   0.001904
2.00E+09    0.32        26148    0.001659       26148   0.001884
2.00E+09    0.34        26148    0.001638       26148   0.001863
```

KUVA 51. Tulostiedosto Python-sovelluksella, jännitys (*S*) elementissä 12104, laippaliitoksen pultin parametrisointi Poissonin vakion (*Nyy*) suhteen

Maksimijännityksen tulosta tarkasteltiin lisäksi Abaqus/CAE GUI -käyttöliittymällä, kun kimmomoduuli on 200 GPa ja Poissonin vakio 0,32. Sekä GUI -käyttöliittymällä että Python-sovelluksella laippaliitoksen maksimijännitys (*S*) löytyy elementistä 12104 ja on 0,517175 MPa (kuva 51; kuva 52).



KUVA 52. Laippaliitoksen pultin jännitys (*S*) Abaqus/CAE GUI -käyttöliittymällä elementissä 12104 kimmokertoimen ollessa 200 GPa ja Poissonin vakion 0,32

7.2.3 Parametrisointi kimmomoduulin ja Poissonin vakion suhteen

Laippaliitoksen pulttia parametrisoitiin pelkän kimmomoduulin lisäksi vielä sekä kimmomoduulin että Poissonin vakion suhteen. Analyysi on epälineaarinen ja tarkasteltuna materiaalina on polyvinyylikloridimuovi (PVC). Kimmomoduuli muuttuu välillä 1 – 3,5 GPa ja Poissonin vakio välillä 0,26 - 0,34. Maksimijännitys laskettiin elementtien integrointipisteiden arvoista Mises-vertailujännityksenä.

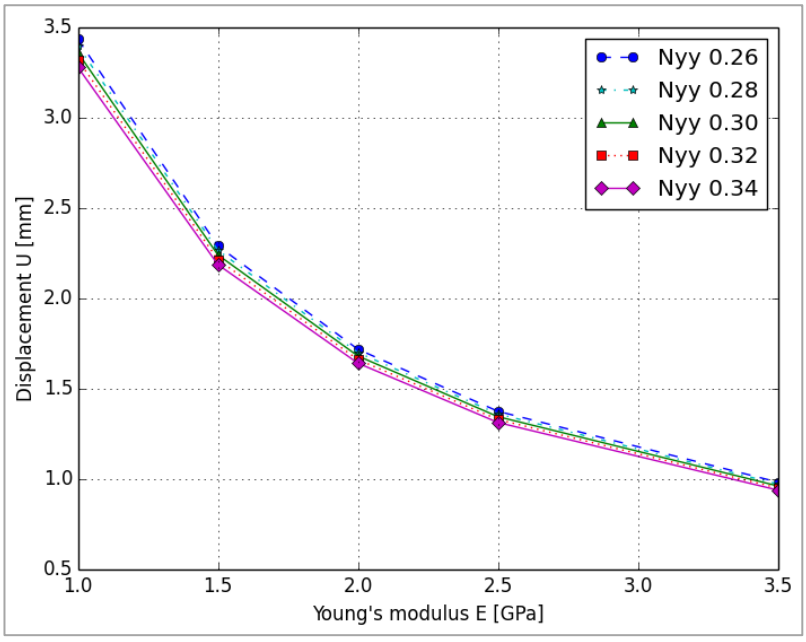
Tulokset tallentuivat tekstitiedostoon (kuva 53). Tuloksista erillisellä Python-sovelluksella piirrettyjen kuvaajien perusteella havaitaan, että tutkitussa tapauksessa kimmomoduulin muutos vaikutti maksimisiirtymän suuruuteen, mutta Poissonin vakion muutoksella ei ollut yhtä suurta vaikutusta (kuva 54). Poissonin vakion muutos vaikutti lineaarisesti jännityksen muuttumiseen, mutta kimmomoduulin muutoksella ei ollut merkitystä (kuva 55).

Job-LaippaPVC_Results_EjaNyy.txt

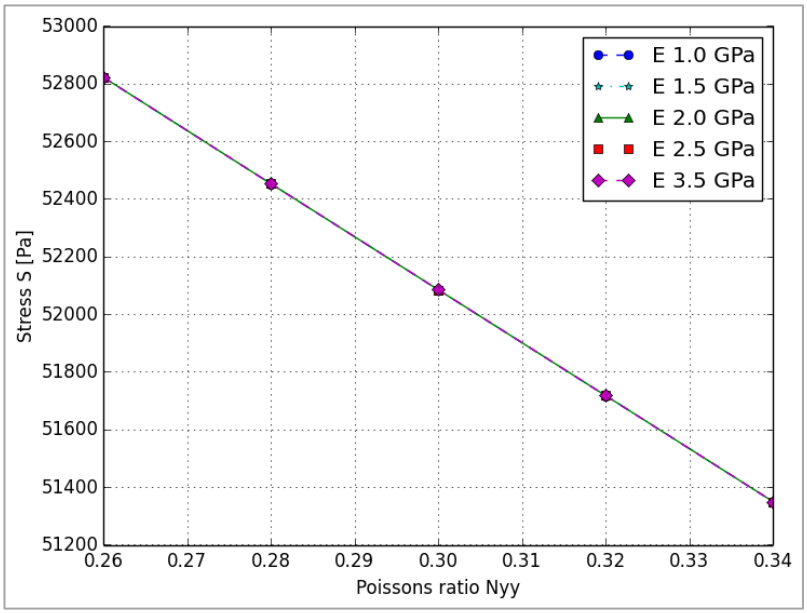
MAXIMUM DISPLACEMENT (Umax) AND STRESS (Smax)

E	Nyy	Node(Umax)	Umax	Element(Smax)	Smax
1.00E+09	0.26	26148	0.003431531	12104	5.28216E+04
1.00E+09	0.28	26148	0.003395115	12104	5.24525E+04
1.00E+09	0.30	26148	0.003357179	12104	5.20846E+04
1.00E+09	0.32	26148	0.003317684	12104	5.17173E+04
1.00E+09	0.34	26148	0.003276581	12104	5.13495E+04
1.50E+09	0.26	26148	0.002287797	12104	5.28217E+04
1.50E+09	0.28	26148	0.002263517	12104	5.24526E+04
1.50E+09	0.30	26148	0.002238221	12104	5.20848E+04
1.50E+09	0.32	26148	0.002211888	12104	5.17174E+04
1.50E+09	0.34	26148	0.002184483	12104	5.13497E+04
2.00E+09	0.26	26148	0.001715900	12104	5.28218E+04
2.00E+09	0.28	26148	0.001697688	12104	5.24527E+04
2.00E+09	0.30	26148	0.001678715	12104	5.20849E+04
2.00E+09	0.32	26148	0.001658963	12104	5.17175E+04
2.00E+09	0.34	26148	0.001638407	12104	5.13498E+04
2.50E+09	0.26	26148	0.001372749	12104	5.28219E+04
2.50E+09	0.28	26148	0.001358178	12104	5.24528E+04
2.50E+09	0.30	26148	0.001342998	12104	5.20849E+04
2.50E+09	0.32	26148	0.001327195	12104	5.17176E+04
2.50E+09	0.34	26148	0.001310749	12104	5.13498E+04
3.50E+09	0.26	26148	0.000980562	12104	5.28220E+04
3.50E+09	0.28	26148	0.000970153	12104	5.24529E+04
3.50E+09	0.30	26148	0.000959309	12104	5.20850E+04
3.50E+09	0.32	26148	0.000948021	12104	5.17177E+04
3.50E+09	0.34	26148	0.000936272	12104	5.13499E+04

KUVA 53. Ote tulostiedostosta, laippaliitoksen pultin parametrisointi kimmomoduulin (*E*) ja Poissonin vakion (*Nyy*) suhteen



KUVA 54. Laippaliitoksen pultin parametrisointi kimmomoduulin *E* suhteen Poissonin vakion *Nyy* eri arvoilla, siirtymä *U* [mm]



KUVA 55. Laippaliitoksen pultin parametrisointi Poissonin vakion *Nyy* suhteen kimmomoduulin *E* eri arvoilla, jännitys *S* [Pa]

Sovelluksella voi parametrisoitaessa poimia myös kaikkien tarkastelun kohteena olevien solmujen siirtymät (kuva 56). Tästä on hyötyä, jos halutaan tarkastella pienempää aluetta kerrallaan.

Job_NodeResults.txt

```
DISPLACEMENTS AT ALL NODES

E           Nyy         Node            Ux              Uy              Uz
--------------------------------------------------------------------------------
1.00E+09    0.26        9820            -0.000760       0.002174        0.000100
1.00E+09    0.26        26187           -0.000927       0.002948        0.000000
1.00E+09    0.26        26188           -0.000926       0.002923        0.000000
1.00E+09    0.26        26189           -0.000924       0.002898        0.000000
1.00E+09    0.26        26190           -0.000921       0.002873        -0.000000
1.00E+09    0.26        26191           -0.000917       0.002853        0.000000
1.00E+09    0.26        26192           -0.001731       0.001394        -0.000000
1.00E+09    0.26        25971           -0.000247       0.002064        0.000000
1.00E+09    0.26        26193           -0.001736       0.001439        0.000000
1.00E+09    0.26        26194           -0.001740       0.001484        0.000000
1.00E+09    0.26        26195           -0.001745       0.001530        0.000000
1.00E+09    0.26        26196           -0.001750       0.001576        0.000000
1.00E+09    0.26        25991           0.000121        -0.000058       -0.000000
1.00E+09    0.26        26197           -0.001755       0.001622        0.000000
1.00E+09    0.26        26198           -0.001759       0.001668        0.000000
1.00E+09    0.26        25997           0.000199        -0.000131       0.000000
1.00E+09    0.26        26199           -0.001763       0.001714        0.000000
1.00E+09    0.26        26004           0.000252        -0.000020       0.000000
```

KUVA 56. Ote tulostiedostosta, kaikkien solmusiirtymien arvot

7.3 Tulosten tarkastelu

Python-sovelluksen toimivuus todennettiin erityyppisten FEM-mallien avulla. Maksimisiirtymät ja -jännitykset poimittiin ristikko-, kehä-, levy- ja solidirakenteiden lineaarisella ja epälineaarisella analyysillä saaduista tuloksista. Ristikko- ja kehärakenteissa materiaalina oli teräs sekä levyrakenne-esimerkeissä teräs ja kumi. Solidimallien materiaalina oli alumiini sekä teräs.

Materiaalin parametrisointia tarkasteltiin levy- ja solidimallien avulla, jolloin levymallissa materiaalina oli kumi ja solidimallissa polyvinyylikloridimuovi (PVC). Esimerkkinä oli myös orpoon verkkoon perustuva malli. Materiaali parametrisoitiin erikseen kimmomoduulin tai Poissonin vakion sekä yhtäaikaisesti molempien suhteen.

Tarkastellut ristikko- ja kehämallit olivat kaksiulotteisia ja levy- ja solidimallit kolmeulotteisia. Elementtityyppinä käytettiin ristikkomallissa 2-solmuisia sauvaelementtejä, kehämallissa 2-solmuisia palkkielementtejä ja levymalleissa 4-solmuisia kuorielementtejä ja 8-solmuisia tasojännityselementtejä. Solidimalleissa käytettiin 20- ja 27-solmuisia tiiliskivielementtejä.

Esimerkkitapausten perusteella Python-sovelluksen maksimiarvojen poiminta binäärisestä tulostiedostosta soveltuu ristikko-, kehä-, levy- ja solidimalleille. Materiaalin parametrisointi soveltui tarkastelluille levy- ja solidimalleille. Lineaaristen ja epälineaaristen analyysien tulokset olivat kaikissa tapauksissa samat sekä Abaqus/CAE GUI- että Abaqus Scripting -käyttöliittymällä tarkasteltuna.

8 JOHTOPÄÄTÖKSET JA POHDINTA

FE-analyysiohjelman Abaqus sisältämä Abaqus Scripting -käyttöliittymän on Python-ohjelmointikielen laajennus. Sen avulla Abaqus-ohjelmaa voidaan käyttää Python-kielellä ohjelmoimalla erilaisia sovelluksia. Abaqus Scripting -käyttöliittymä on erittäin laaja ja sen sisältämiin olioihin ja niiden metodeihin tutustumalla voidaan käyttöliittymää hyödyntää useaan tarkoitukseen. Sovelluksilla voi luoda mallin ja suorittaa laskennan kokonaan tai vain muokata mallia osittain. Osittaiset, mallia muuttavat sovellukset toimivat Abaqus/CAE GUI:ssa yhdessä muokattavan mallin kanssa. Muutoin voidaan toimia ilman graafista käyttöliittymää.

Työssä toteutettiin Python-sovellus, joka poimii Abaqus-ohjelman tuottamasta binäärisestä odb-tiedostosta maksimisiirtymät ja -jännitykset sekä muuttaa mallin materiaaliarvoja. Haluttaessa sovelluksella voidaan tarkastella tai parametrisoida koko mallia tai vain pienempää osa-aluetta mallista. Maksimiarvot voidaan poimia erilaisten FEM-mallien tulostiedostoista ja parametrisoida levy- ja solidimallien materiaalia. Analyysi voi olla lineaarinen tai epälineaarinen. Sovelluksen toimintaa todennettiin vertaamalla saatuja tuloksia Abaqus/CAE GUI:n antamiin tuloksiin. Todettiin, että tulokset vastaavat toisiaan, joten sovelluksen toimintaa voidaan pitää luotettavana. Toteutetulla sovelluksella saadaan nopeutettua tulosten poimintaa sekä suoritettua nopeasti materiaalien parametrisointi. Sovellusta voidaan jatkossa laajentaa materiaaliarvojen osalta myös muiden ominaisuuksien luomiseen sekä parametrisointiin.

Seuraavana sovelluskohteena tulisi perehtyä kuormitusten ja reunaehtojen määrittämiseen. Tällöin pystytään toistamaan nopeasti esimerkiksi usean pisteen kuormitus, jos mallin elementtiverkkoa tai analyysityyppiä halutaan vaihtaa. Voitaisiin myös toteuttaa mobiliteettimatriisi, jossa voima syötetään yhdelle solmulle kerrallaan.

Koska tutkimuksen kohteena olevat mallit ovat usein kookkaita, ne vaativat paljon laskentakapasiteettia. Tästä syystä mallien laskenta suoritetaan tehokkailla palvelinkoneilla. Jatkossa kannattaisi selvittää, miten Abaqus Scripting -käyttöliittymällä tehty sovellus skaalautuu tehokkaassa, moniytimisessä palvelinkoneessa siten, että kaikki koneen laskentateho saadaan käyttöön. Tulisi ottaa selvää, tekeekö Abaqus skaalauksen itsestään vai tuleeko sovelluksessa ottaa jotain huomioon laskentatehon maksimaalisen käytön takaamiseksi.

LÄHTEET

Abaqus 6.12 Documentation. 2014.

Abaqus Company Profile. Luettu 6.5.2014
http://imechanica.org/files/2%20Reading%20-%20About%20ABAQUS.pdf

Abaqus Unified FEA Brochure. 2013. Luettu 26.4.2014.
http://www2.3ds.com/fileadmin/PRODUCTS/SIMULIA/PDF/brochures/simulia-abaqus-unified-fea-brochure.pdf

Apiola, H. & Laine, M. 2010. Matlab-opas. Espoo: Aalto-yliopisto. Luettu 27.12.2013.
http://math.aalto.fi/~apiola/matlab/opas/ei-niin-lyhyt/

About Python. 2013. Luettu 27.12.2013. http://www.python.org/about/

Bathe, K-J. 1996. Finite Element Procedures. Yhdysvallat: Prentice Hall.

Cook, R. D. & Malkus, D. S. & Plesha M. E. & Will, R. J. 2001. Concepts and Applications of Finite Element Analysis. Yhdysvallat: Wiley.

Elementtimenetelmät I (461033A). 2012 – 2013. Oulu: Oulun yliopisto. Tulostettu 27.4.2014. http://me.oulu.fi/files/tx_opetus/113/1346655249_fem_luennot.pdf

Gustafsson, B. 2011. Fundamentals of Scientific Computing. Saksa: Springer.

Hetland, M. L. 2005. Beginning Python: From Novice to Professional. Yhdysvallat: Apress.

History of the Software. 2013. Python Software Foundation. Luettu 27.12.2013.
http://docs.python.org/2/license.html#history-of-the-software

Kouhia, R. & Tuomala, M. 2009. Rakennetekniikan Numeeriset Menetelmät. Luentomoniste. Tampere: Tampereen teknillinen yliopisto. Luettu 4.4.2014.

Lähteenmäki, M. 2008 - 2009. Lujuusopin jatkokurssi, luentokalvot I.4. Tampere: Tampereen ammattikorkeakoulu. Luettu 4.4.2014.
http://personal.inet.fi/koti/mlahteen/arkistot/lujjk_pdf/luper_4_k.pdf

Lähteenmäki, M. 2012 - 2013. Lujuusoppi 2, luentokalvot. Tampere: Tampereen ammattikorkeakoulu. Luettu 2.5.2014.
http://personal.inet.fi/koti/mlahteen/arkistot/luj2_pdf/jtila_k.pdf

Lähteenmäki, M. 2009. Elementtimenetelmän jatkokurssi. Luentomoniste. Tampere: Tampereen ammattikorkeakoulu.

Morgan, E.F. & Bouxsein M. L. 2005. Use of Finite Element Analysis to Assess Bone Strength. BoneKEy-Osteovision 2005 December 2(12), 9.
http://www.nature.com/bonekey/knowledgeenvironment/2005/0512/bonekey20050187/full/bonekey20050187.html

News, TL 4000. 2009. Ultralight Aircrafts. Luettu 24.4.2014.
http://tl-ultralight.cz/en/news-page-10/tl-4000/

Puri, G. M. 2011. Python Scripts for Abaqus, Learn by Example. Yhdysvallat.

Python vs Matlab. 2013. Tulostettu 28.12.2013.
http://www.pyzo.org/python_vs_matlab.html

Rashed, Md. G. & Ahsan, R. 2012. Python in Computational Science: Applications and Possibilities. International Journal of Computer Applications (0975 – 8887) Volume 46 – No. 20.

TIOBE Programming Community Index. 2013. TIOBE Software. Luettu 27.12.2013.
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Vihavainen, A. & Paksula, M. & Luukkainen, M. & Laaksonen, A. & Mikkola, P. & Laurinharju, J. & Pärtel, M. 2013. Ohjelmoinnin perusteet Python-kielellä. Helsinki: Helsingin yliopisto. Tulostettu 27.12.2013.
http://www.cs.helsinki.fi/group/linkki/materiaali/python-perusteet/materiaali.html