Rakenteiden Mekaniikka (Journal of Structural Mechanics)
vol. 54, nro 2, 2021, s. 80–94
http://rakenteidenmekaniikka.journal.fi/index
https://doi.org/10.23998/rm.103377

VERTAISARVIOITU
KOLLEGIALT GRANSKAD
PEER-REVIEWED
www.tsv.fi/tunnus

Avoimen lähdekoodin hyödyntäminen väsymisanalyysissä

Ilkka Valkonen

Tiivistelmä Artikkelissa esitetään menetelmät väsymisdatan ja yksikkökuormien yhdistämisestä ja sen käytöstä avoimen lähdekoodin FEM-ohjelmistojen yhteydessä väsymisvaurion selvittämiseksi. Kirjallisuudesta löytyvän esimerkin ja ohjelmalla lasketut tulokset olivat lähellä toisiaan. Avoimen lähdekoodin tarjoamat ratkaisut ovat siis varteenotettavia moniaksiaalisen väsymisen analysoinnissa.

Avainsanat: väsyminen, moniaksiaalinen, aikasarja, yksikkökuorma, avoin lähdekoodi

Vastaanotettu: 18.3.2021. Hyväksytty: 8.4.2021. Julkaistu verkossa: 14.6.2021.

Johdanto

Todellisen rakenteen väsymisen arviointi spektrikuormituksen alaisena saattaa olla kustannuksiltaan raskas, jos tavoitteena on jokaisen solmun vaurion määrittely ja kuormituksen aikasarjassa on suuri määrä datapisteitä. Lisäksi tulosten havainnollistaminen on haastavaa. Asian voi yrittää ratkaista selvittämällä kriittiset pisteet, joille tämän jälkeen suoritetaan tarkempi analysointi. Tässä ratkaisussa on riskinä, että jokin kohta jää huomioimatta, aiheuttaen mahdollisesti käytön aikana vaurioita ja huomattavia kustannuksia.

Ongelmaa voi lähestyä siten, että rakenteelle lasketaan yksikkökuormilla jännitystaso, jota sitten kerrotaan aikasarjan tiedoilla. Tällöin elementtimentelmästä tarvitsee vain yhden ratkaisun yksikkökuormaa kohti, lopun työn ollessa vaurion laskentaa. Jos mallissa on useita solmuja, on edellä mainittu vaurion laskenta aikaa vievää ilman automaatiota. Menetelmää on käsitelty Gaierin esitelmissä.[5, 6]

Ongelmaa on käsitelty myös Norbergin ja Ohlssonin artikkelissa[7], jossa on esitelty vauriokriteerejä ja kevyesti sivuttu edellä mainittuja esitelmiä.

Tähän tarkoitukseen löytyy joitain kaupallisia ohjelmistoja[5], mutta niiden korkeat lisenssikustannukset estävät käytännössä niiden käyttöä laajalti. Korkeat lisenssikustannukset voi välttää avoimen lähdekoodin ratkaisuilla [2, 4], jotka vastaavasti vaativat käyttäjiltään korkeaa ammattitaitoa.

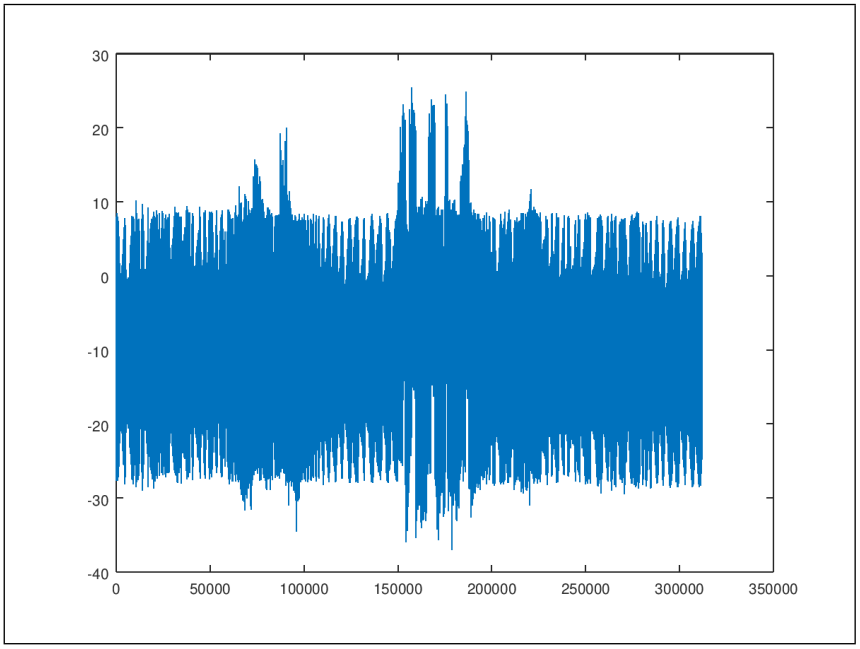
Tässä esityksessä keskitytään yhteen moniaksiaaliseen malliin [3], josta löytyy kirjallisuudesta dokumentoitu analyyttinen ratkaisu [9], sekä sen laskenta avoimen lähdekoodin ohjelmistolla selvittäen sen käyttökelpoisuutta väsymisanalyysin tekoon.

Aikasarjan ja yksikkökuorman yhdistäminen

Tässä oletetaan, että rakenne toimii lineaarisesti, jolloin yksinkertaisen kertolaskun käyttö on mahdollista. Kuormitus voidaan kuvata yksinkertaisella funktiolla (1).

$$\mathbf{F} = f(t) \tag{1}$$

Esimerkki kuormituksesta on kuvassa 1. Käytännössä kuormituksen esittämien jatkuvana funktiona on haastavaa, joten se käytännössä täytyy esittää aika–kuormitusarvopareina.



Kuva 1. Kuormitusdata ajan funktiona.

Yksikkökuorman, eli ulkoisen kuorman arvolla 1 ja kuormitustiedon, eli aikasarjan avulla voidaan määrittää koko rakenteen jännityksen vaihtelu kaavan (2) avulla.

$$\boldsymbol{\sigma}_i(t) = \mathbf{F} \cdot \boldsymbol{\sigma}_{i,yk} \tag{2}$$

Kaavassa (2) $\boldsymbol{\sigma}_{yk}$ yksikkökuorman aihettama jännitys rakenteessa. Olennaista on siis, että $\boldsymbol{\sigma}_{yk}$ lasketaan (1) yksikköjä käyttäen arvolla 1.

Edellä kuvatulla menetelmällä voidaan laskea erillisesti kuormitustapaukset, esimerkiksi voima kahdesta eri suunnasta, tai vielä useampia vaihtoehtoja. Tällöin voidaan jännitykset yhdistää yksinkertaisella lineaarikombinaatiolla (3).

$$\boldsymbol{\sigma}_{tot}(t) = \sum \boldsymbol{\sigma}_i(t) \tag{3}$$

Yhtälössä (3) alaindeksi σ_i kuvaa eri kuormitustapausten jännitystensoria. Nyt lasketulla yhdistetyllä jännitysaikasarjalla voidaan laskea vaurio, jonka aikasarjan kuormanvaihtelu aiheutaa.

Vaurion laskenta

Vaurion laskemiseksi ruotsalainen Palmgren [8] (ns. Palmgren–Miner-menetelmä) esitteli menetelmän, missä kokonaisvaurio saadaan osavaurioiden summana (4) siten, että vau-

rioiden summa on 1 vaurion tapahtuessa.

$$D = \sum \frac{N_i}{N_{f,i}} \tag{4}$$

Kaavassa (4) N_i syklit tietyllä jännitysheilahdustasolla ja $N_{f,i}$ on syklien määrä vaurioon kyseisellä jännitysheilahdustasolla.

Yksiaksiaalisessa tapauksessa vaurion laskenta on hyvin suoraviivaista. Moniaksiaaliseen tapaukseen on kehitetty muutamia menetelmiä, joista mainittakoon van Dang ja Fatemi–Socie (5). Käytännön ongelma näissä on materiaaliparametrien saanti. Parametrien kehitys vaatii aksiaaliset väsytyskokeet ja lisäksi vääntöväsytyskokeet.

$$\frac{\Delta\gamma_{max}}{2} \cdot \left(1 + k \cdot \frac{\sigma_{n,max}}{S_y}\right) = C \tag{5}$$

Kaavassa (5) k ja C ovat vakioita, jotka määritellään kokeiden perusteella.

Väsytystapaus kirjallisuudesta

Esimerkkitapauksena on käytetty *Metal Fatigue in Engineering* -kirjan esimerkkiä sivuilta 333–337. Tehtävä on ohutseinäisen putken väsytys veto–puristusväsytyksessä, sekä saman aikaisessa väännössä. Jännityssuhde on $R = -1$. Esimerkkitapaus on kirjassa laskettu maksimileikkauksen menetelmällä sekä Fatemin–Socien kriittisen tason menetelmällä. Esimerkin materiaalio minaisuudet ovat taulukossa 1. Materiaalin myötöraja oli 380 MPa, sekä kuormitusamplitudit $\epsilon = 0,0026$ ja $\gamma = 0,0057$. Venymähistoria oli kaavojen (6) ja (7) mukainen.

$$\epsilon_x = \epsilon \cdot \sin(\omega \cdot t) \tag{6}$$

$$\epsilon_{xy} = \gamma \cdot \sin(\omega \cdot t + 90°) \tag{7}$$

Kuvan 2 jännityksistä voi päätellä leikkaustason suunnan vaihtelun, sekä yksinkertaisen väsytyssyklin päättelemisen vaikeuden, kuten esimerkiksi vetojännityksen laskiessa Treskan vertailujännitys kasvaa.

Taulukko 1. Materiaalitiedot

Väsytystapaus	G, E	$\tau_f^{'}, \sigma_f^{'}$	b_0, b	$\gamma_f^{'}, \epsilon_f^{'}$	c_0, c
Vääntö	80e3 MPa	505 MPa	−0,097	0,413	−0,445
Veto–puristus	205e3 MPa	948 MPa	−0,092	0,26	−0,445

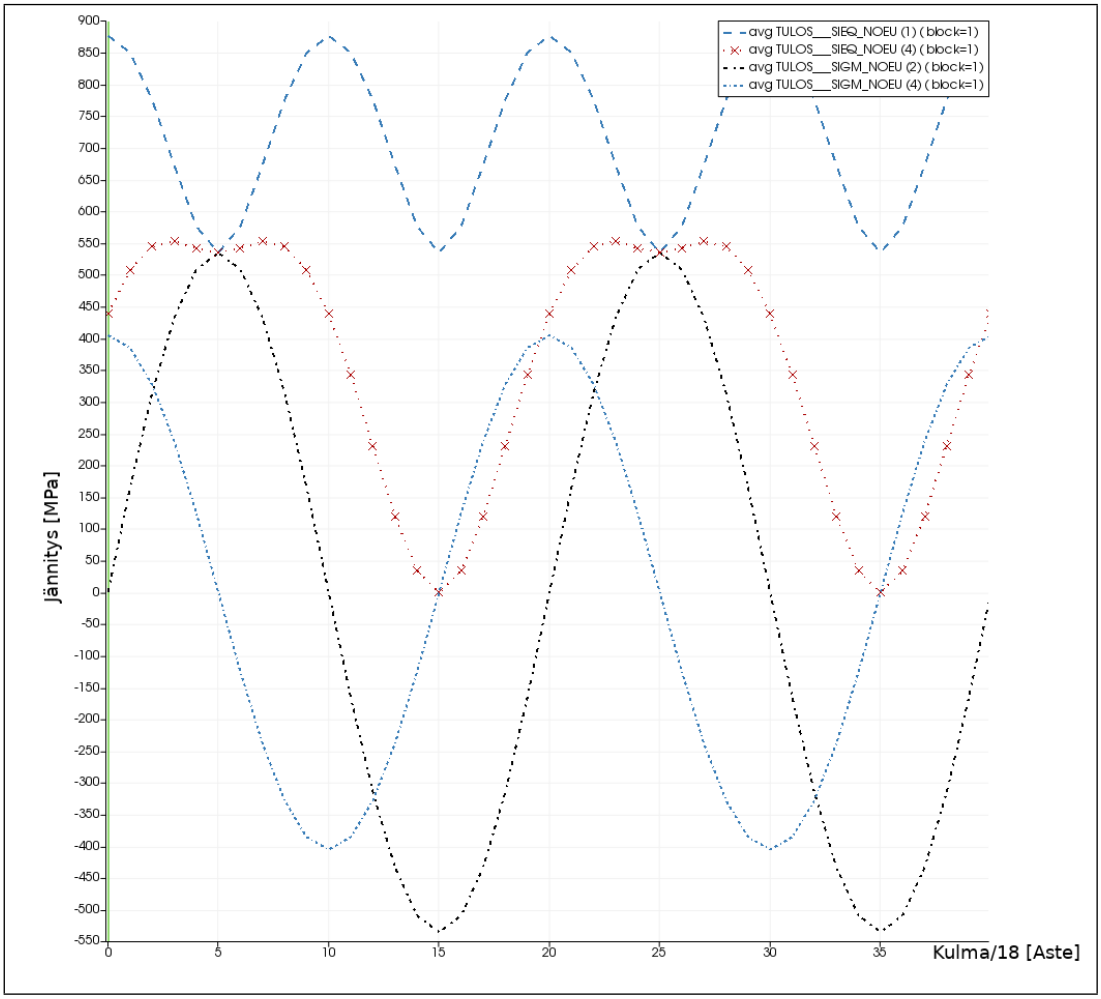
Tarkasteltaessa elinikälaskelmia, maksimileikkauksen menetelmä antaa eliniäksi noin 23 000 sykliä (8), kun $\Delta\gamma_{max} = 0,0113$.

$$\frac{\Delta\gamma_{max}}{2} = \frac{\tau_f^{'}}{G} \cdot (2 \cdot N_f)^{b_0} + \gamma_f^{'} \cdot (2 \cdot N_f)^{c_0} = \frac{505}{80\,000} \cdot (2 \cdot N_f)^{-0,097} + 0,413 \cdot (2 \cdot N_f)^{-0,445} \tag{8}$$

Vastaavasti Fatemi–Socie-menetelmällä (5) saadaan tulokseksi noin 6000 sykliä (9), kun $\Delta\gamma_{max} = 0,0113$ ja $\sigma_{n,max} = 350$ MPa.

$$\frac{\Delta\gamma_{max}}{2}\left(1 + 0,6 \cdot \frac{\sigma_{n,max}}{380}\right) = \frac{505}{80\,000} \cdot (2 \cdot N_f)^{-0,097} + 0,413 \cdot (2 \cdot N_f)^{-0,445} \tag{9}$$

Koetulos esimerkin (9) väsytyskokeesta oli noin 5300 sykliä.[9]



Kuva 2. Esimerkin Trescan vertailujännitys (SIEQ_NOEU(1)), maksimi pääjännitys (SIEQ_NOEU(4)), sekä normaali vetojännitys (SIGM_NOEU(2)) ja leikkausjännitys (SIGM_NOEU(4)). Kuvassa on kaksi sykliä, eli 720°.

Esimerkkitapauksen analysointi avoimen lähdekoodin ratkaisulla

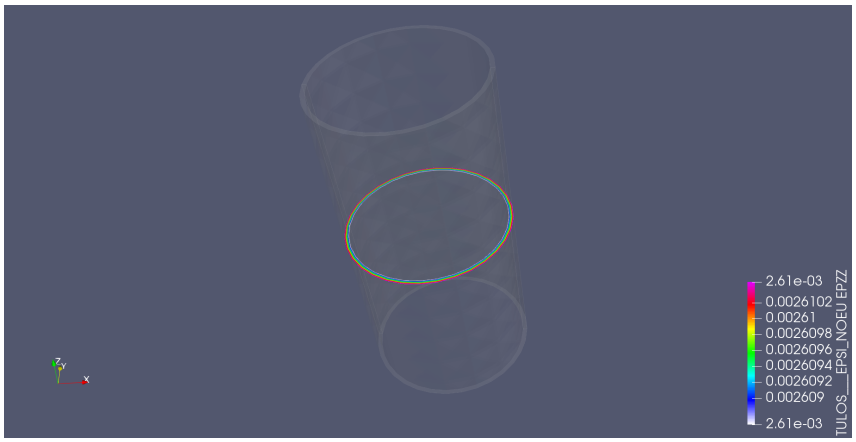
Esimerkkitapauksessa ohjelmistona on käytetty Code-Aster avoimen lähdekoodin ohjelmistoa. Etuina siinä on ylläpitäjänä toimiva vahva organisaatio, joka käyttää sitä omissa toiminnoissaan ja siten käytännössä kantaa aitoa vastuuta ohjelmistosta. Toinen etu on ohjelmiston hallintaan käytetty Python-kieli, joka yleisenä ja suhteellisen helppona mahdollistaa ongelmien laskennan automatisoinnin.

Laskentamallin dimensiot ja kuormitusten määrittely

Laskentamallina käytettiin ohutseinäistä sylinteriä, mitat taulukossa 2. Lähteessä on kuormitustiedot annettu venyminä, jotka Hooken lain avulla arvot on muutettu painekuormaksi ja vääntömometiksi. Muunnos varmistettiin tekemällä staattinen analyysi, jonka venymäarvoja verrattiin lähtötietoihin (kuvat 3. . . 5). Tuloksia verrattessa on muistettava yhteys $\gamma_{xz} = 2 \cdot \epsilon_{xz}$ ja vastaavasti $\gamma_{yz} = 2 \cdot \epsilon_{yz}$.

Taulukko 2. Laskentamallin mitat

Pituus [mm]	Ulkohalkaisija [mm]	Seinämänpaksuus [mm]
400	200	5



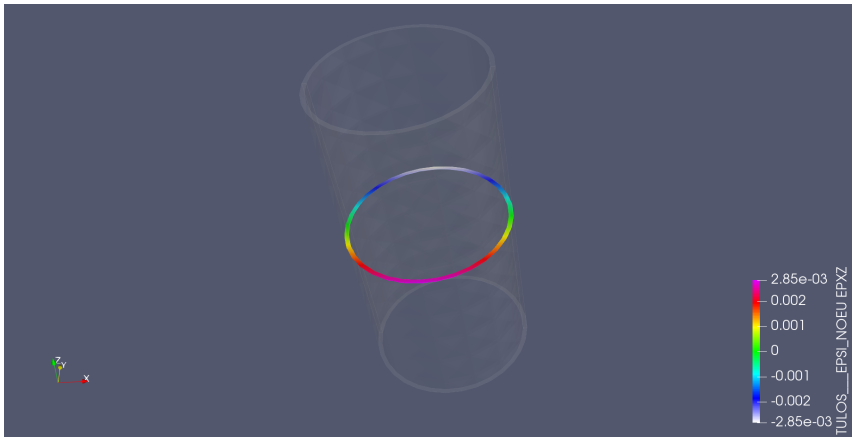
Kuva 3. Aksiaalivenymä,ϵ_z

Vaurion laskenta

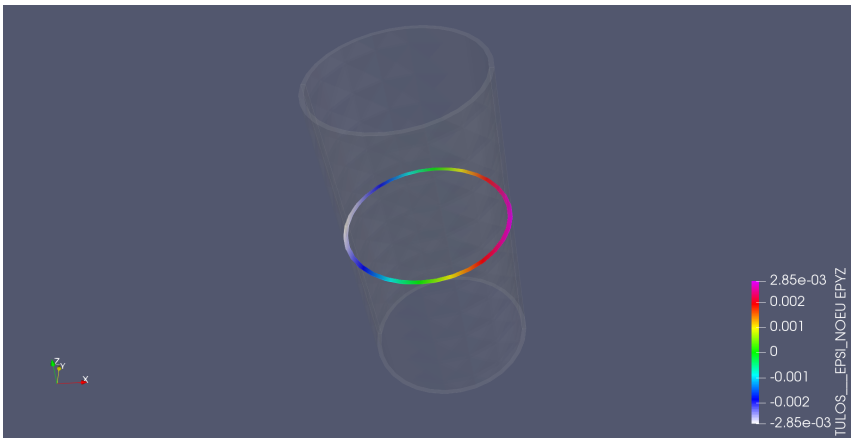
Code-Aster-ohjelmisto tarjoaa väsymisvaurion laskemiseksi operaattorit *CALC_FATIGUE* ja *POST_FATIGUE* [1], joista ensiksi mainittu laskee koko mallin vaurion, jolloin koko mallia pitää kuormittaa halutulla väsytyskuormalla. Jos on tarve käyttää pitkää spektrikuormitusta, saattaa laskennasta tulla hyvin raskas tätä operaattoria käytettäessä.

Operaattori *POST_FATIGUE* laskee halutuille solmuille vaurion. Kyseinen operaattori ei vaadi tuloksia Code-Asterista, vaan siihen voi sytöttää jännitys- ja venymätiedot Python-operaatioilla muista lähteistä tarvittaessa. Tässä esimerkkitapauksessamme hyödynnämme Code-Aster-analyysiä kuorma-amplitudin jännitysten ja venymien ratkaisemiseksi.

Käytännössä laskennan voi suorittaa yhtenä ajona, joka sisältää kaikki vaurion laskentaan tarvittavat toimet. Olennaiset Code-Asterin komentotiedoston tiedot ilman ulkoisia



Kuva 4. Leikkaus,ϵ_{xz}



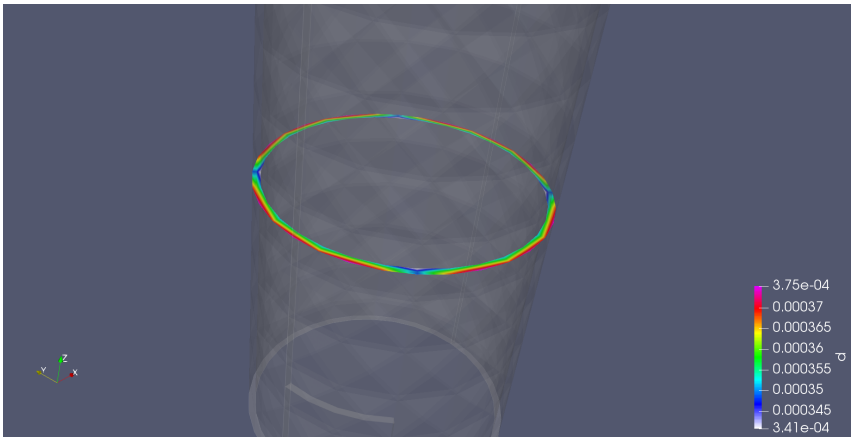
Kuva 5. Leikkaus,ϵ_{yz}

Python operaatioita on esitetty liitteessä 1.

Vaurion laskenta Code-Asterilla voidaan lyhyesti ryhmitellä seuraavasti:

- kuormitusaikasarjojen luku
- lineaarinen rakenneanalyysi yksikkökuormilla ($\boldsymbol{\sigma}_{yk}$, kaava (2))
- kuormitusaikasarjan kertominen yksikkökuorman jännityksillä ja venymillä
- laskettujen eri yksikkökuormien jännitysten ja kuormitusaikasarjojen tulojen yhdistäminen
- vaurion laskenta halutuissa kohdissa.
- tulosten muodostaminen.

Edellä mainituilla periaatteella laskettiin vaurio kuvan 2 jännitykset ja vastaavat venymät yhdistämällä. Vaurioksi kahden syklin kuormituksella saatiin 3,75e-4 (kuva 6). Kokonaiselinikän on siis $2 \cdot \frac{1}{3{,}75e{-}4} = 5333$ sykliä. Tulos oli siis hyvin lähellä koetulosta ja kirjassa laskettua arvoa.



Kuva 6. Kahden syklin aiheuttama vaurio Fatemi–Socie-kriteerillä.

Johtopäätökset ja yhteenveto

Kuormituksen ollessa multiaksiaalista ja erityisesti vaihesiirtoa omaavaa, on väsymismitoitus haastavaa ja perinteisillä lähestymistavoilla, kuten esimerkiksi maksimi leikkausjännityskriteerillä, tuloksena mahdollisesti epärealistisen pitkät eliniät, kuten esimerkki osoittaa [9].

Tähän ongelmaan on kehitetty kriittiseen tasoon perustuvia ratkaisuja, kuten Mataki-, van Dang- ja Fatemi–Socie-menetelmät. Kuitenkin kuormituksen ollessa vaihtuva-amplidinen ja suresta määrästä datapisteistä muodostuva, on laskenta helposti aikaa vievää, jos tarkoituksena on laskea ratkaisu elementtimentelmällä jokaiselle ajan hetkelle.

Laskennan keventämiseksi on tässä artikkelissa esitetty menetelmä, jossa elementtimetelemästä tarvittavien ratkaisujen määrää saadaan huomattavasti vähennettyä ja suurin osa laskenta-ajasta käytetään vaurion laskemiseen. Menetelmän rajoituksena on, että se soveltuu vain tapauksille, jotka voidaan yhdistää vektori- ja matriisilaskennan säännöillä. Menetelmän taloudellinen hyödyntäminen on mahdollista käyttäen avoimen lähdekoodin työkaluja, joka tässä artikkelissa on ollut Ranskalainen Code-Aster, jossa on mahdollista yhdistää elementtimetelmä Pythonilla tehtyihin laskutoimituksiin ja operaatioihin.

Esimerkkinä on laskettu kirjallisuudesta esimerkkitapaus, jonka tulos vastasi hyvin kirjassa esitettyjä tuloksia. Tässä esitetty menetelmä tarjoaa siis taloudellisen ratkaisun moniaksiaalisten väsymisongelmien selvittämiseksi, kun kuormitus on esitetty aikasarjoina.

Viitteet

[1] Electricié de France, *Critères multi-axiaux d,amorçage en fatigue, R7.04.04,* 2013, URL: www.code-aster.org/V2/doc/v14/fr/man_r/r7/r7.04.04.pdf

[2] Electricié de France, *Finite element code_aster, Analysis of Structures and Thermomechanics for Studies and Research,* 1989–2020.

[3] A. Fatemi ja D.F. Socie, A Critical Plane Approach to Multiaxial Fatigue Damage Including Out-of-Phase Loading. *Fatigue and Fracture Engineering Materials and Structures*, Vol. 11, No. 3,1988, p. 149.

[4] Frondelius, T., & Aho, J. (2017). JuliaFEM - open source solver for both industrial and academia usage. Rakenteiden Mekaniikka, 50(3), 229–233. https://doi.org/10.23998/rm.64224

[5] Gaier, C., *Theory and applications of* FEMFAT - *A* FE - *Postprosessing tool for fatigue analysis* 1999, Fatigue Conference 1999 Beijing

[6] Gaier C. Steinwender G. Dannbauer H. FEMFAT-MAX: a FE- posprocessor for fatigue analysis of multiaxially loaded components, 2000, www.femfat.com, presented at NAFEMS seminar, Fatigue analysis, Wiesbaden 2000.

[7] S. Norberg, M. Olsson, A fast, versatile fatigue post-processor and criteria evaluation, International Journal of Fatigue, Volume 27, Issues 10–12, 2005, Pages 1335–1341, ISSN 0142-1123, https://doi.org/10.1016/j.ijfatigue.2005.07.011.

[8] A. Palmgren, Die Lebansdauer von Kugellagern. *Zeitschrift des Vereines Deutscher Ingenieure, 68, 14, 339–341*, 1924.

[9] R.I Stephens, A. Fatemi, R.R. Stephens, H.O. Fuchs *Metal Fatigue in Engineering*, John Wiley & Sons, Inc. 2. painos, 2001.

Ilkka Valkonen
Stressfield Oy
Innopoli 2, Tekniikantie 14, 02150 Espoo
ilkka.valkonen@stressfield.fi

Liite 1. Code-Aster komentotiedoston olennaiset käskyt moniaksiaaliseen väsymismitoitukseen

```
DEBUT()


VERKKO = LIRE_MAILLAGE(
UNITE=20
)




MALLI = AFFE_MODELE(AFFE=(_F(MODELISATION=('3D', ),
PHENOMENE='MECANIQUE',
TOUT='OUI'),
_F(
GROUP_MA=('LAPIJ1', ),
MODELISATION=('POU_D_E', ),
PHENOMENE='MECANIQUE',)),
MAILLAGE=VERKKO)

LASEL = AFFE_CARA_ELEM(
POUTRE=_F(
CARA=('R', 'EP'),
GROUP_MA=('LAPIJ1', ),
SECTION='CERCLE',
VALE=(200.0, 5.0),),
MODELE=MALLI,
)

TERAS = DEFI_MATERIAU(ELAS=_F(E=205000.0,
NU=0.3))

LASMAT = AFFE_MATERIAU(AFFE=_F(MATER=(TERAS, ),
TOUT='OUI'),
MODELE=MALLI)

LOAD = AFFE_CHAR_MECA(
DDL_IMPO=_F(
DX=0.0,
DY=0.0,
DZ=0.0,
GROUP_MA=('FIXING', )
),
MODELE=MALLI,
PRES_REP=_F(
GROUP_MA=('VOPI', ),
PRES=-534.9794,
```

```
)
)

LOAD2 = AFFE_CHAR_MECA(
FORCE_NODALE=_F(
GROUP_NO=('VAANTON', ),
MZ=129545.454e3,
),
MODELE=MALLI
)

RAMPPI_1 = DEFI_FONCTION(NOM_PARA='INST',
VALE=(0.0, 0.0, 1.0, 1.0, 2.0, 0.0, ))

RAMPPI_2 = DEFI_FONCTION(NOM_PARA='INST',
VALE=(0.0, 0.0, 1.0, 0.0, 2.0, 1.0, ))

AIKA = DEFI_LIST_REEL(DEBUT=0.0,
INTERVALLE=(_F(JUSQU_A=1.0,
NOMBRE=1),
_F(JUSQU_A=2.0,
NOMBRE=1),
))

STLA = MECA_STATIQUE(CHAM_MATER=LASMAT,
CARA_ELEM=LASEL,
EXCIT=(_F(CHARGE=LOAD,
FONC_MULT=RAMPPI_1),
_F(CHARGE=LOAD2,
FONC_MULT=RAMPPI_2),
),
LIST_INST=AIKA,
MODELE=MALLI)

TULOS = CALC_CHAMP(CONTRAINTE=('SIGM_NOEU',  ),
GROUP_MA='ALLVOL',
CRITERES=('SIEQ_NOEU',),
DEFORMATION=('EPSI_NOEU', ),
RESULTAT=STLA)

ULOS1 = CREA_CHAMP(NOM_CHAM=('SIGM_NOEU'),
INST=1.0,
OPERATION='EXTR',
RESULTAT=TULOS,
TYPE_CHAM='NOEU_SIEF_R')

EULOS1 = CREA_CHAMP(NOM_CHAM=('EPSI_NOEU'),
INST=1.0,
```

```
OPERATION='EXTR',
RESULTAT=TULOS,
TYPE_CHAM='NOEU_EPSI_R')

ULOS2 = CREA_CHAMP(NOM_CHAM=('SIGM_NOEU'),
INST=2.0,
OPERATION='EXTR',
RESULTAT=TULOS,
TYPE_CHAM='NOEU_SIEF_R')

EULOS2 = CREA_CHAMP(NOM_CHAM=('EPSI_NOEU'),
INST=2.0,
OPERATION='EXTR',
RESULTAT=TULOS,
TYPE_CHAM='NOEU_EPSI_R')

jasix1 = ULOS1.EXTR_COMP('SIXX',['ALLVOLS'],0).valeurs
jasiy1 = ULOS1.EXTR_COMP('SIYY',['ALLVOLS'],0).valeurs
jasiz1 = ULOS1.EXTR_COMP('SIZZ',['ALLVOLS'],0).valeurs
jasixy1 = ULOS1.EXTR_COMP('SIXY',['ALLVOLS'],0).valeurs
jasixz1 = ULOS1.EXTR_COMP('SIXZ',['ALLVOLS'],0).valeurs
jasiyz1 = ULOS1.EXTR_COMP('SIYZ',['ALLVOLS'],0).valeurs

easix1 = EULOS1.EXTR_COMP('EPXX',['ALLVOLS'],0).valeurs
easiy1 = EULOS1.EXTR_COMP('EPYY',['ALLVOLS'],0).valeurs
easiz1 = EULOS1.EXTR_COMP('EPZZ',['ALLVOLS'],0).valeurs
easixy1 = EULOS1.EXTR_COMP('EPXY',['ALLVOLS'],0).valeurs
easixz1 = EULOS1.EXTR_COMP('EPXZ',['ALLVOLS'],0).valeurs
easiyz1 = EULOS1.EXTR_COMP('EPYZ',['ALLVOLS'],0).valeurs

jasix2 = ULOS2.EXTR_COMP('SIXX',['ALLVOLS'],0).valeurs
jasiy2 = ULOS2.EXTR_COMP('SIYY',['ALLVOLS'],0).valeurs
jasiz2 = ULOS2.EXTR_COMP('SIZZ',['ALLVOLS'],0).valeurs
jasixy2 = ULOS2.EXTR_COMP('SIXY',['ALLVOLS'],0).valeurs
jasixz2 = ULOS2.EXTR_COMP('SIXZ',['ALLVOLS'],0).valeurs
jasiyz2 = ULOS2.EXTR_COMP('SIYZ',['ALLVOLS'],0).valeurs

easix2 = EULOS2.EXTR_COMP('EPXX',['ALLVOLS'],0).valeurs
easiy2 = EULOS2.EXTR_COMP('EPYY',['ALLVOLS'],0).valeurs
easiz2 = EULOS2.EXTR_COMP('EPZZ',['ALLVOLS'],0).valeurs
easixy2 = EULOS2.EXTR_COMP('EPXY',['ALLVOLS'],0).valeurs
easixz2 = EULOS2.EXTR_COMP('EPXZ',['ALLVOLS'],0).valeurs
easiyz2 = EULOS2.EXTR_COMP('EPYZ',['ALLVOLS'],0).valeurs

# print(jasi)

# syklit1 = syklit * jasi[0]
```

```
WOKUE2 = DEFI_FONCTION(INTERPOL=('LOG', 'LOG'),
NOM_PARA='EPSI',
PROL_DROITE='CONSTANT',
PROL_GAUCHE='CONSTANT',
VALE=(3.2102e-4, 2.0e12, 4.9438e-4, 2.0e10, 7.8623e-4, 2.0e8,
0.0014424, 2.0e6, 0.0022472, 2.0e5, 0.0040729, 2.0e4,
0.0172866, 2.0e3))

MATE = DEFI_MATERIAU(FATIGUE=_F(MANSON_COFFIN=WOKUE2),
CISA_PLAN_CRIT=_F(
COEF_CISA_TRAC=1.2,
CRITERE='FATESOCI_MODI_AV',
FATSOC_A=0.00157895
),
)

sx = syklit * jasix1[j] + tsyklit * jasix2[j]
sy = syklit * jasiy1[j] + tsyklit * jasiy2[j]
sz = syklit * jasiz1[j] + tsyklit * jasiz2[j]
sxy = syklit * jasixy1[j] + tsyklit * jasixy2[j]
sxz = syklit * jasixz1[j] + tsyklit * jasixz2[j]
syz = syklit * jasiyz1[j] + tsyklit * jasiyz2[j]

ex = syklit * easix1[j] + tsyklit * easix2[j]
ey = syklit * easiy1[j] + tsyklit * easiy2[j]
ez = syklit * easiz1[j] + tsyklit * easiz2[j]
exy = syklit * ( easixy1[j] + tsyklit * easixy2[j] ) * 1.0
exz = syklit * ( easixz1[j] + tsyklit * easixz2[j] ) * 1.0
eyz = syklit * ( easiyz1[j] + tsyklit * easiyz2[j] ) * 1.0

if i in listv:

kx =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(sx)
)

ky =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(sy)
)

kz =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(sz)
```

```
)

kxy =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(sxy)
)

kxz =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(sxz)
)

kyz =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(syz)
)

kex =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(ex)
)

key =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(ey)
)

kez =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(ez)
)

kexy =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(exy)
)

kexz =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(exz)
```

```
)

keyz =  DEFI_FONCTION(
NOM_PARA='INST',
ABSCISSE=(luvut),
ORDONNEE=(eyz)
)

ax =  POST_FATIGUE(CHARGEMENT='MULTIAXIAL',
#    COMPTAGE='RAINFLOW',
CRITERE='FATESOCI_MODI_AV',
DOMMAGE='MANSON_COFFIN',
HISTOIRE=_F(
SIGM_XX=kx,
SIGM_XY=kxy,
SIGM_XZ=kxz,
SIGM_YY=ky,
SIGM_YZ=kyz,
SIGM_ZZ=kz,
EPS_XX=kex,
EPS_XY=kexy,
EPS_XZ=kexz,
EPS_YY=key,
EPS_YZ=keyz,
EPS_ZZ=kez,
),
MATER=MATE,
PROJECTION='DEUX_AXES',
TYPE_CHARGE='NON_PERIODIQUE'
)

bx = CALC_TABLE(
ACTION=_F(
NOM_PARA=('DOMMAGE', ),
OPERATION='CALCUL',
TYPE_CALCUL=('SOMM', )
),
TABLE=ax
)

cx = EXTR_TABLE(
NOM_PARA='DOMMAGE',
TABLE=bx,
TYPE_RESU='REEL'
)

d = CREA_CHAMP(
AFFE=(afelix),
```

```
MODELE=MALLI,
OPERATION='AFFE',
TYPE_CHAM='NOEU_SIEF_R'
)

IMPR_RESU(
RESU=(_F(
RESULTAT=TULOS
), _F(
CHAM_GD=d
),
),
UNITE=80
)

# print(tuta)
# print(solnu)
# print(s)

FIN()
```