



TEKNILLINEN TIEDEKUNTA

**IMPLISIITTINEN AIKAINTEGROINTI  
YLEISTETYLLÄ  $\alpha$ -MENETELMÄLLÄ  
ELEMENTTIMENETELMÄLASKENNASSA**

Sami Kreivi

KONETEKNIIKAN TUTKINTO-OHJELMA

Kandidaatintyö 2019

# TIIVISTELMÄ

Implisiittinen aikaintegrointi yleistetyllä  $\alpha$ -menetelmällä elementtimenetelmälaskennassa

Sami Kreivi

Oulun yliopisto, Konetekniikan tutkinto-ohjelma

Kandidaatintyö 2019, 36 s.

Työn ohjaaja yliopistolla: Hannu Koivurova

Tämän kandidaatintyön tavoitteena oli luoda Python-ohjelmointikielinen yleistetty  $\alpha$ -menetelmää käyttävä laskenta-algoritmi elementtimenetelmällä ratkaistaville dynaamisille lineaarisille rakenteellisen mekaniikan ongelmille. Kyseisellä algoritmilla haluttiin saada vertailutulosten avulla selville COMSOL Multiphysics -laskentaohjelmistossa ilmennyt ongelma sen yleistettyä  $\alpha$ -menetelmää käyttävässä ratkaisijassa. Virheen paikantamiseksi ratkaistiin samoja esimerkkitapauksia samoilla parametreilla niin tätä työtä varten luodulla laskenta-algoritmilla kuin COMSOL Multiphysics -laskentaohjelmistolla, jotta voitiin rajata ongelmien lähde. Tärkeimpänä tuloksena mainittakoon se, että kun yleistetyn  $\alpha$ -menetelmän algoritmin numeerinen vaimennus minimoidaan, antaa COMSOL virheellisiä tuloksia tai ei pysty suorittamaan laskentaa edes loppuun asti. Työn tuloksia hyödynnetään jatkossa siten, että työtä varten luotu algoritmi annetaan käytettäväksi Oulun yliopiston konetekniikan osaston Elementtimenetelmät II -kurssin opetusmateriaaliin. Myös COMSOL Multiphysics -ohjelman kehittäjille lähetetään luettavaksi tämä kandidaatintyö ongelman korjaamista varten.

*Asiasanat: elementtimenetelmä, dynaaminen, aikaintegrointi, ohjelmointi*

## **ABSTRACT**

Implicit Time Integration with Generalized- $\alpha$  Method in Finite Element Method Calculation

Sami Kreivi

University of Oulu, Degree Programme of Mechanical Engineering

Bachelor's thesis 2019, 36 pp.

Supervisor at the university: Hannu Koivurova

The aim of this bachelor's thesis was to create a calculation algorithm which uses the generalized  $\alpha$ -method and is used to solve dynamic linear problems of structural mechanics with finite element method. The algorithm was programmed with Python programming language. The algorithm was used to get results of comparison to pinpoint the source of problems occurred in calculation program COMSOL Multiphysics and especially in its solver which uses the generalized  $\alpha$ -method. To pinpoint the problem the same example problems were solved with the same parameters with both the algorithm created and COMSOL Multiphysics. The most important result of this thesis is that COMSOL gives inaccurate results or can't even finish computing when the numerical damping is minimized. The results of this thesis will be utilized so that the algorithm created will be given to University of Oulu's Department of Mechanical Engineering's course Elementtimenetelmät II as teaching material. As well this bachelor's thesis will be sent to the developers of COMSOL Multiphysics for solving the problem in the program.

*Keywords: finite element method, dynamic, time integration, programming*

## ALKUSANAT

Tämän kandidaatintyön päätarkoituksena oli luoda Python-ohjelmointikielinen yleistä  $\alpha$ -menetelmää käyttävä laskenta-algoritmi elementtimenetelmäskentää varten. Kyseisen algoritmin avulla oli tarkoitus saada vertailukelpoisia tuloksia samanlaisiin elementtimenetelmäohjelmistoilla laskettuihin tapauksiin, jotta etenkin COMSOL Multiphysics -elementtimenetelmäohjelmistossa ilmenneet yleistettyyn  $\alpha$ -menetelmään liittyvät virheet saataisiin paikannettua. Työhön liittynyt algoritmin luominen sekä tulosten vertaaminen laskentatapojen välillä tehtiin kesän ja syksyn 2019 aikana ja raportointi syksyn 2019 aikana. Erityisesti haluaisin kiittää Oulun yliopiston konetekniikan tutkinto-ohjelman yliopistonlehtoreita työn ohjaajaa Hannu Koivurovaa sekä aihetta ehdottanutta Jouko Lumijärveä.

Oulu, 15.11.2019

*Sami Kreivi*

Työn tekijä

# SISÄLLYSLUETTELO

TIIVISTELMÄ

ABSTRACT

ALKUSANAT

SISÄLLYSLUETTELO

MERKINNÄT JA LYHENTEET

|                                              |    |
|----------------------------------------------|----|
| 1 JOHDANTO .....                             | 7  |
| 1.1 Työn tavoite .....                       | 7  |
| 1.2 Työn rajaus .....                        | 7  |
| 2 LUJUUSLASKENTA ELEMENTTIMENETELMÄLLÄ ..... | 8  |
| 2.1 Teoria .....                             | 8  |
| 2.2 Laskentaohjelmistot .....                | 14 |
| 3 DYNAAMISET ONGELMAT .....                  | 15 |
| 3.1 Implisiittinen aikaintegrointi .....     | 15 |
| 3.2 Yleistetty $\alpha$ -menetelmä .....     | 16 |
| 4 YLEISTETYN ALFA-MENETELMÄN ALGORITMI ..... | 18 |
| 5 ESIMERKKITAPAUKSET .....                   | 21 |
| 5.1 Ulokepalkki .....                        | 21 |
| 5.2 Kone .....                               | 22 |
| 6 TULOSTEN TARKASTELU JA VERTAILU .....      | 24 |
| 6.1 Ulokepalkki .....                        | 24 |
| 6.2 Kone .....                               | 29 |
| 7 YHTEENVETO .....                           | 35 |
| LÄHDELUETTELO                                |    |

## MERKINNÄT JA LYHENTEET

|                     |                                        |
|---------------------|----------------------------------------|
| $A$                 | poikkipinta-ala                        |
| $\mathbf{a}$        | kiihtyvyyssvektori                     |
| $\mathbf{B}$        | elementin kiertomatriisi               |
| $\mathbf{C}$        | rakenteen vaimennusmatriisi            |
| $E$                 | kimmomoduuli                           |
| $F$                 | voima                                  |
| $\mathbf{F}$        | rakenteen kuormitusvektori             |
| $f$                 | taajuus                                |
| $H$                 | korkeus                                |
| $I$                 | neliömomentti                          |
| $J$                 | hitausmomentti                         |
| $\mathbf{K}$        | rakenteen jäykkyysmatriisi             |
| $\mathbf{k}$        | elementin jäykkyysmatriisi             |
| $L$                 | pituus                                 |
| $\mathbf{M}$        | rakenteen massamatriisi                |
| $n$                 | pyörimisnopeus                         |
| $t$                 | aika                                   |
| $\mathbf{U}$        | rakenteen siirtymävektori              |
| $\dot{\mathbf{U}}$  | rakenteen nopeusvektori                |
| $\ddot{\mathbf{U}}$ | rakenteen kiihtyvyyssvektori           |
| $\mathbf{u}$        | siirtymävektori                        |
| $\mathbf{V}$        | elementin vastaavuusmatriisi           |
| $\mathbf{v}$        | nopeusvektori                          |
| $\alpha_a$          | numeerisen vaimennuksen säätöparametri |
| $\alpha_f$          | algoritminen parametri                 |
| $\alpha_m$          | algoritminen parametri                 |
| $\alpha_r$          | Rayleigh-vaimennusvakio                |
| $\beta$             | algoritminen parametri                 |
| $\beta_r$           | Rayleigh-vaimennusvakio                |
| $\gamma$            | algoritminen parametri                 |

|          |                 |
|----------|-----------------|
| $\Delta$ | muutos          |
| $\theta$ | pyörimiskulma   |
| $\nu$    | Poissonin vakio |
| $\rho$   | tiheys          |
| $\Omega$ | kulmanopeus     |

# 1 JOHDANTO

## 1.1 Työn tavoite

Tavoitteena oli tutkia COMSOL Multiphysics -laskentaohjelmistossa ilmenneitä dynaamisiin laskentatapauksiin ja etenkin näiden yleistettyä  $\alpha$ -menetelmää käyttävään ratkaisijaan liittyviä ongelmia. Olennaisena osana tätä tutkimusta oli laatia Python-ohjelmointikielinen samaa aikaintegrointimenetelmää käyttävä laskenta-algoritmi, jonka avulla laskettiin vertailutuloksia COMSOL Multiphysics -ohjelmistolla saaduille tuloksille. Luodusta algoritmista pystyttiin myös paikantaa COMSOL Multiphysics -ohjelmiston lähdekoodin virhe suhteellisen tarkasti. Työssä käsiteltävä ongelma ilmeni eräällä Oulun yliopiston konetekniikan tutkinto-ohjelman kurssin Elementtimenetelmät II -luennolla, jonka aiheena oli dynaamisten ongelmien ratkaisu elementtimenetelmällä.

## 1.2 Työn rajaus

Työ päätettiin rajata tarkastelemaan vain lineaarisesti käyttäytyviä rakenteiden mekaniikan malleja. Ottamalla epälineaarisuus huomioon olisi myös työn tuloksena saatua laskenta-algoritmia täytynyt muuttaa epälineaarisille tapauksille sopivaksi. Jos taas olisi tarkasteltu teknisiä ongelmia rakenteiden mekaniikkaa laajemmin, olisi työ voinut paisua huomattavasti. Tutkimus rajattiin käsittelemään vain kahta kappaleessa 5 esiteltyä dynaamista esimerkkitapausta, joiden ratkaisuun käytettiin käsinlaskennassa Pythonia tarkastelun alla olevan COMSOL Multiphysics -ohjelmiston lisäksi.

## 2 LUJUUSLASKENTA ELEMENTTIMENETELMÄLLÄ

### 2.1 Teoria

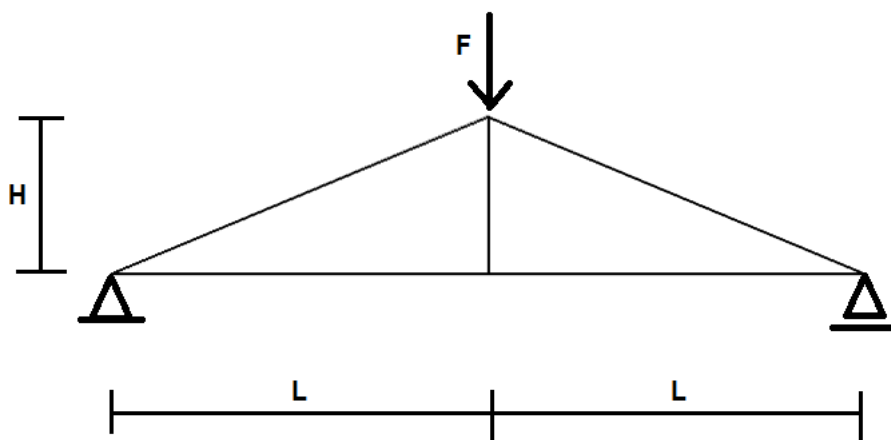
Elementtimenetelmä on alun perin ilmailualaa varten kehitetty numeerinen laskentamenetelmä rakenteellisen mekaniikan ongelmien ratkaisuun ja analysointiin. Tämän nykyään hyvin laajalle alalle vakiintuneen menetelmän parhaana puolena pidetään sen tehokkuutta, tarkkuutta ja soveltuvuutta erittäin monimutkaisiin insinööriyön ongelmiin. Myöhemmin tätä menetelmää on sovellettu rakenteiden mekaniikan ongelmien lisäksi myös muihin teknisiin ongelmiin, kuten lämmön johtumisen ja siirtymisen, virtauksien sekä sähkö- ja magneettikenttien laskemiseen. Elementtimenetelmän suosion uskotaan piilevän myös sen helppokäyttöisyydessä. Kun menetelmän tietokoneohjelma on kerran luotu jollekin laskentatapaukselle, voidaan saman ohjelman avulla ratkaista mikä tahansa samantyylinen ongelma vain pienillä ohjelman muutoksilla. (Rao 2005, s. 3)

Perustavana ideana elementtimenetelmässä on jakaa monimutkainen ongelma useisiin pienempiin ja yksinkertaisempiin osiin tarkasteltavan rakenteen mukaan. Näissä toisiinsa solmupisteillä kiinnittyneissä pienemmissä osissa eli elementeissä ratkaistaan yksinkertaisempi ongelma, jonka jälkeen nämä tulokset yhdistetään taas alkuperäisen rakenteen mukaiseen muotoonsa. Tapauksen muuttujien, kuten esimerkiksi siirtymien, jännityksien, paineen tai nopeuden, oletetaan muuttuvan elementin sisällä approksimaatiofunktioiden mukaisesti, koska todellista muuttujien arvojen vaihtelua elementtien sisällä ei tiedetä. Kun kirjoitetaan kenttäyhtälöt (esimerkiksi tasapainoyhtälöt) koko rakenteelle, tuntemattomina muuttujina toimivat elementtien välisten solmupisteiden arvot. Kun kenttäyhtälöt ratkaistaan, saadaan tulokseksi haetun muuttujan arvot solmupisteissä, jonka jälkeen voidaan taas jakaa muuttujien arvot elementtienkin sisälle approksimaatiofunktioiden mukaan. (Rao 2005, s. 53)

Erilaisia elementtejä ja näiden ominaisuuksia on olemassa erittäin paljon, ja oikean elementin valinta riippuukin pääasiassa tarkasteltavan rakenteen muodosta (Rao 2005, s. 53). Yksinkertaisimmillaan elementti voikin olla 1D-viivaelementti, joita käytetään esimerkiksi langan venymisen laskennassa. Yleisimpiä ovat kuitenkin 2D- ja 3D-elementit, joita käytetään esimerkiksi palkki- ja soliditapausten laskennassa. Elementeille

määritellään vapausasteet, joihin laskennan muuttujat jaotellaan niiden vaikutussuunnan ja -pisteen mukaan. Elementtien koko ja määrä vaikuttavat myös suuresti laskennan lopputulokseen. Voidaan olettaa, että suuremmalla määrällä pienikokoisia elementtejä saadaan tarkempi lopputulos kuin vähäisellä määrällä suurempia elementtejä, mutta suurempi määrä elementtejä vie laskennalta aina enemmän aikaa (Rao 2005, s. 53).

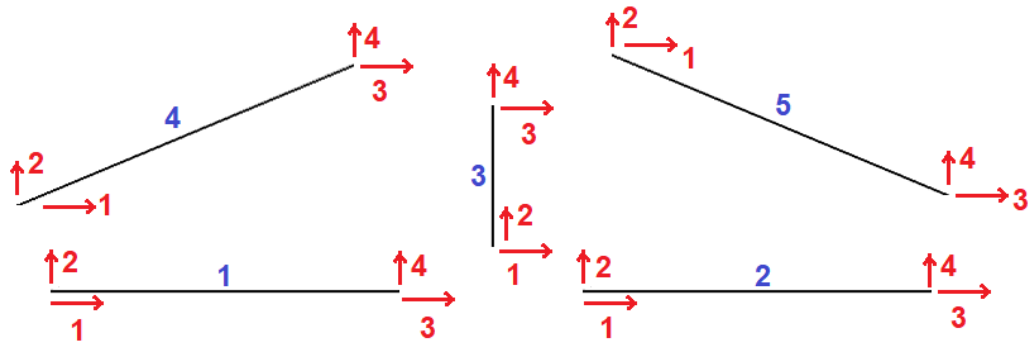
Seuraavana esitellään elementtimenetelmän ratkaisuvaiheiden havainnollistamiseksi erään yksinkertaisen staattisen sauvaristikkotapauksen (Kuva 1) ratkaisu:



Kuva 1. Esimerkkitehtävän tarkasteltava rakenne.

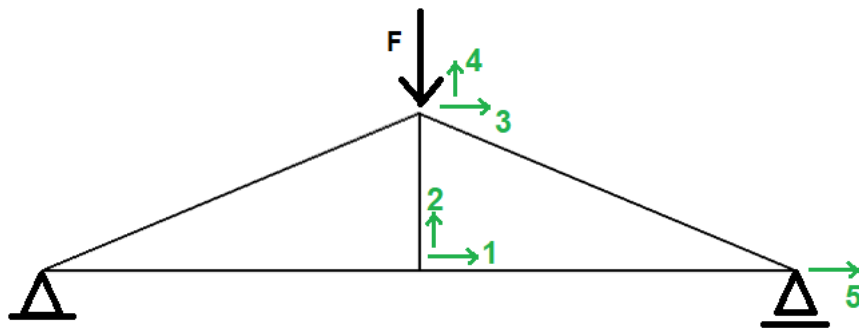
Kyseessä on siis 2D-sauvaristikkorakenne, jonka vasen pääty on tuettu niveltuennalla ja oikea pääty  $x$ -akselin suuntaisesti liikkuvalla niveltuennalla. Ristikon oikean päädyn  $y$ -akselin suuntaiset siirtymät on siis estetty, mutta  $x$ -akselin suuntaiset siirtymät ovat mahdollisia. Kaikkien sauvojen kimmomoduuli  $E = 210$  GPa, Poissonin vakio  $\nu = 0.3$  ja poikkipinta-ala  $A = 30$  mm<sup>2</sup>. Rakenteen mitat ovat:  $H = 0.5$  m ja  $L = 1.2$  m, ja rakennetta kuormittava pistevoima  $F = 2.7$  kN. Tehtävänä on laskea elementtimenetelmällä voiman  $F$  vaikutuspisteen  $x$ - ja  $y$ -akselin suuntaiset siirtymät.

Ensimmäisenä jaetaan rakenne elementteihin. Tässä tapauksessa muodostetaan jokaisesta sauvasta oma elementtinsä ja numeroidaan ne sinisillä numeroilla kuvan 2 mukaan. Tapauksessa käytetään kaksisolmuista neljän vapausasteen sauvaelementtiä. Jokaisen elementin vapausaste numeroidaan punaisilla numeroilla. Vapausasteet ovat solmupisteissä mahdollisesti tapahtuvien siirtymien vaikutussuuntia.



Kuva 2. Esimerkkitehtävän elementtijako ja elementtien vapausasteet.

Rakenteellekin annetaan vapausasteet vihreillä numeroilla kuvan 3 mukaisesti. Koska vasemman päädyn niveltuennan kohdalla ei synny siirtymiä x- eikä y-akselin suunnassa ja oikean päädyn tuennan kohdalla ei synny y-akselin suuntaisia siirtymiä, voidaan näihin kohtiin jättää vapausasteet määrittämättä.



Kuva 3. Esimerkkitehtävän rakenteen vapausasteet.

Yksiaksaalisen sauvaelementin jäykkyydsmatriisi on

$$\mathbf{k}_e = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (1)$$

ja se saadaan laajennettua 2D-sauvaelementin jäykkyydsmatriisiksi lisäämällä siihen y-akselin suuntaisten vapausasteiden komponentit seuraavasti

$$\mathbf{k}_e = \frac{EA}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad (2)$$

missä  $\mathbf{k}_e$  on elementin  $e$  jäykkyysmatriisi,  
 $E$  on elementin kimmomoduuli,  
 $A$  on elementin poikkipinta-ala ja  
 $L$  on elementin pituus. (Liu & Quek 2003, s. 77 & 81)

Koska osa elementeistä ei ole x-akselin suuntaisia, täytyy ne kääntää kiertomatriisin avulla seuraavasti

$$\mathbf{k}_e = \mathbf{B}^T \bar{\mathbf{k}}_e \mathbf{B}, \quad (3)$$

missä  $\bar{\mathbf{k}}_e$  on kääntämätön elementin  $e$  jäykkyysmatriisi ja  
 $\mathbf{B}$  on seuraavan yhtälön muotoinen elementin kiertomatriisi

$$\mathbf{B} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & \cos(\theta) & \sin(\theta) \\ 0 & 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}, \quad (4)$$

missä taas  $\theta$  on elementin kulma x-akselin suhteen. (Liu & Quek 2003, s. 72)

Elementtien jäykkyysmatriiseiksi saadaan nyt seuraavien yhtälöiden mukaiset matriisit

$$\mathbf{k}_1 = \mathbf{k}_2 = \begin{bmatrix} 5250000 & 0 & -5250000 & 0 \\ 0 & 0 & 0 & 0 \\ -5250000 & 0 & 5250000 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{Pa/m}, \quad (5)$$

$$\mathbf{k}_3 \approx \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 12600000 & 0 & -12600000 \\ 0 & 0 & 0 & 0 \\ 0 & -12600000 & 0 & 12600000 \end{bmatrix} \text{Pa/m}, \quad (6)$$

$$\mathbf{k}_4 \approx \begin{bmatrix} 4129267 & 1720528 & -4129267 & -1720528 \\ 1720528 & 716887 & -1720528 & -716887 \\ -4129267 & -1720528 & 4129267 & 1720528 \\ -1720528 & -716887 & 1720528 & 716887 \end{bmatrix} \text{Pa/m ja} \quad (7)$$

$$\mathbf{k}_5 \approx \begin{bmatrix} 4129267 & -1720528 & -4129267 & 1720528 \\ -1720527 & 716887 & 1720528 & -716887 \\ -4129267 & 1720528 & 4129267 & -1720528 \\ 1720528 & -716887 & -1720528 & 716887 \end{bmatrix} \text{Pa/m.} \quad (8)$$

Seuraavaksi määritetään elementtien vastaavuusmatriisit, jotka sisältävät tiedon siitä, miten elementtien vapausasteet vastaavat koko tarkasteltavan rakenteen vapausasteista. Vastaavuusmatriiseissa sarakkeet vastaavat rakennevapausasteita ja rivit elementtivapausasteita. Jos nämä vapausasteet vastaavat toisiaan asetetaan kyseiseksi alkiksi 1, muuten alkiksi jätetään 0. Seuraavissa yhtälöissä on näkyvissä elementtien 1-5 vastaavuusmatriisit

$$\mathbf{V}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad (9)$$

$$\mathbf{V}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (10)$$

$$\mathbf{V}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad (11)$$

$$\mathbf{V}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \text{ ja} \quad (12)$$

$$\mathbf{V}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \text{ (Lumijärvi 2018)} \quad (13)$$

Elementtien jäykkyys- ja vastaavuusmatriisien avulla saadaan muodostettua koko rakenteen jäykkyysmatriisi sijoittelusummaamalla seuraavasti

$$\mathbf{K} = \sum_{i=1}^N \mathbf{V}_i^T \mathbf{k}_i \mathbf{V}_i, \quad (14)$$

missä  $\mathbf{K}$  on koko rakenteen jäykkyysmatriisi,  
 $i$  on elementin numero ja  
 $N$  on elementtien lukumäärä. (Lumijärvi 2018)

Näin saadaan kyseessä olevan esimerkkitehtävän koko tarkasteltavan rakenteen jäykkyysmatriisiksi

$$\mathbf{K} \approx \begin{bmatrix} 10.5 & 0 & 0 & 0 & -5.25 \\ 0 & 12.6 & 0 & -12.6 & 0 \\ 0 & 0 & 8.26 & 0 & -4.13 \\ 0 & -12.6 & 0 & 14.03 & 1.72 \\ -5.25 & 0 & -4.13 & 1.72 & 9.38 \end{bmatrix} \text{MPa/m.} \quad (15)$$

Tapauksen kuormitusvektori saadaan sijoittamalla pistevoima  $F$  rakennevapausastetta 4 vastaavaan alkioon eli

$$\mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -F \\ 0 \end{bmatrix}, \quad (16)$$

missä  $\mathbf{F}$  on rakenteen kuormitusvektori. (Liu & Quek 2003, s. 73)

Tässä vaiheessa tiedetään jo koko rakenteen kuormitusvektori  $\mathbf{F}$  sekä jäykkyysmatriisi  $\mathbf{K}$ , joten rakenteen siirtymät  $\mathbf{U}$  saadaan selville ratkaisemalla tasapainoyhtälö

$$\mathbf{KU} = \mathbf{F}, \quad (17)$$

missä  $\mathbf{U}$  on rakenteen siirtymävektori. (Liu & Quek 2003, s. 58)

Tarkoilla arvoilla laskettuna esimerkkitapauksen tulokseksi saadaan

$$\mathbf{U} \approx \begin{bmatrix} 0.617 \\ -3.364 \\ 0.617 \\ -3.364 \\ 1.234 \end{bmatrix} \text{mm.} \quad (18)$$

Rakenteen siirtymävektorista (18) voidaan poimia voiman  $F$  vaikutuspisteen siirtymiä vastaavien rakennevapausasteiden 3 ja 4 siirtymien arvot, joiden perusteella kyseisen pisteen siirtymiksi saadaan 0.617 mm x-akselin ja -3.364 mm y-akselin suunnassa.

## 2.2 Laskentaohjelmistot

Elementtimenetelmää ei yleisesti käytetä edellisen kappaleen esimerkkitehtävän tavalla vaihe kerrallaan edeten, vaan apuna on tavallisesti jokin automatisoitu tietokoneohjelma nopeuttamassa ja helpottamassa laskennan vaiheita. Näissä ohjelmistoissa mallinnetaan yleensä vain tarkastelussa olevan rakenteen geometria, syötetään kaikki tarvittavat alkuarvot, parametrit ja materiaaliarvot, luodaan rakenteelle elementtiverkko ja määritetään rakenteeseen vaikuttavat kuormitukset sekä reunaehdot (Liu & Quek 2003, s. 4). Ohjelma muodostaa tarvittavat elementtimenetelmän yhtälöt ja ratkaisee tapauksen automaattisesti, jonka jälkeen laskennan tulosten tarkastelu ja niiden analysointi esimerkiksi simuloinnin tai visualisoinnin keinoin jää taas ohjelman käyttäjän vastuulle.

Muutamia esimerkkejä tällaisista laskentaohjelmistoista ovat:

- Altair HyperMesh,
- ANSYS,
- Autodesk Simulation,
- COMSOL Multiphysics,
- NASTRAN ja
- SIMULIA Abaqus.

## 3 DYNAAMISET ONGELMAT

### 3.1 Implisiittinen aikaintegrointi

Dynaamisissa rakenteellisen mekaniikan tapauksissa ratkaistaan liikeyhtälö, joka on vaimennetussa tapauksessa

$$\mathbf{M}\ddot{\mathbf{U}}+\mathbf{C}\dot{\mathbf{U}}+\mathbf{K}\mathbf{U}=\mathbf{F}, \quad (19)$$

missä  $\mathbf{M}$  on rakenteen massamatriisi,  
 $\ddot{\mathbf{U}}$  on rakenteen kiihtyvyyksvektori ajan funktiona,  
 $\mathbf{C}$  on rakenteen vaimennusmatriisi,  
 $\dot{\mathbf{U}}$  on rakenteen nopeusvektori ajan funktiona,  
 $\mathbf{K}$  on rakenteen jäykkyyssmatriisi  
 $\mathbf{U}$  on rakenteen siirtymävektori ajan funktiona ja  
 $\mathbf{F}$  on rakenteen kuormitusvektori ajan funktiona.  
 (Chung & Hulbert 1993)

Yksi dynaamisten ongelmien ratkaisumenetelmistä on aikaintegrointi, jossa liikeyhtälö ratkaistaan tietyn aika-askeleen välein. Aikaintegrointimenetelmät voidaan jakaa vielä kahteen luokkaan: eksplisiittisiin ja implisiittisiin. Tässä työssä tarkastellaan implisiittisiä menetelmiä, joissa ratkaisua haetaan jokaisen aika-askeleen välein käyttämällä lähtöarvoina aina edellisen aika-askeleen ratkaisusta saatuja tuloksia. Implisiittinen aikaintegrointi on laskennallisesti melko työläs menetelmä, mutta sen hyvänä puolena on sen soveltuvuus monenlaisiin tapauksiin.

Implisiittisen aikaintegroinnin ratkaisut jokaisen aika-askeleen välein saadaan yhtälöstä

$$\mathbf{y}(t_{i+1})=\mathbf{y}(t_i)+\Delta t\mathbf{f}(t_{i+1},\mathbf{y}(t_{i+1})), \quad (20)$$

missä  $y$  on yhtälön ratkaisu ajan funktiona,  
 $t$  on aika,  
 $i$  on kunkin ratkaisun järjestysnumero,  
 $\Delta t = t_{i+1} - t_i$  on aika-askel ja  
 $f$  on differentiaalifunktio, jolle ratkaisua haetaan. (Butcher 2003, s. 57)

### 3.2 Yleistetty $\alpha$ -menetelmä

Tässä työssä tarkastellaan implisiittisistä aikaintegrointimenetelmistä yleistettyä  $\alpha$ -menetelmää, joka on vuonna 1993 ensimmäisen kerran esitelty Hilber-Hughes-Taylor (HHT- $\alpha$ ) sekä Wood-Bossak-Zienkiewicz (WBZ- $\alpha$ ) -menetelmistä modifioitu algoritmi. Yleistetty  $\alpha$ -menetelmä on optimoitu vaimentamaan numeerisesti käyttäjän valitsemia korkeita taajuuksia, mutta jättämään matalat taajuudet vaimentamatta. Kyseisen menetelmän muuttujien määrittelyt ja niiden yhdistäminen rakenteellisen mekaniikan dynaamisen tapauksen liikeyhtälöön on esitelty yhtälöissä

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \Delta t \mathbf{v}_i + \Delta t^2 \left( \frac{1}{2} - \beta \right) \mathbf{a}_i + \beta \mathbf{a}_{i+1}, \quad (21)$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \Delta t \left( (1 - \gamma) \mathbf{a}_i + \gamma \mathbf{a}_{i+1} \right), \quad (22)$$

$$\mathbf{M} \mathbf{a}_{i+1 - \alpha_m} + \mathbf{C} \mathbf{v}_{i+1 - \alpha_f} + \mathbf{K} \mathbf{u}_{i+1 - \alpha_f} = \mathbf{F}(t_{i+1 - \alpha_f}), \quad (23)$$

$$\mathbf{a}_0 = \mathbf{M}^{-1} (\mathbf{F}(0) - \mathbf{C} \mathbf{v} - \mathbf{K} \mathbf{u}), \quad (24)$$

$$\mathbf{u}_{i+1 - \alpha_f} = (1 - \alpha_f) \mathbf{u}_{i+1} + \alpha_f \mathbf{u}_i, \quad (25)$$

$$\mathbf{v}_{i+1 - \alpha_f} = (1 - \alpha_f) \mathbf{v}_{i+1} + \alpha_f \mathbf{v}_i, \quad (26)$$

$$\mathbf{a}_{i+1 - \alpha_m} = (1 - \alpha_m) \mathbf{a}_{i+1} + \alpha_m \mathbf{a}_i \text{ ja} \quad (27)$$

$$t_{i+1 - \alpha_f} = (1 - \alpha_f) t_{i+1} + \alpha_f t_i, \quad (28)$$

missä  $\mathbf{v}$  on rakenteen nopeusvektori,  
 $\mathbf{a}$  on rakenteen kiihtyvyyssvektori,  
 $\beta$  on algoritminen parametri muotoa (29),  
 $\gamma$  on algoritminen parametri muotoa (30),  
 $\mathbf{a}_0$  on alkukiihtyvyys,  
 $\alpha_f$  on algoritminen parametri muotoa (31) ja  
 $\alpha_m$  on algoritminen parametri muotoa (32).

Menetelmän algoritmiset parametrit määritellään seuraavilla yhtälöillä

$$\beta = \frac{1}{4}(1 - \alpha_m + \alpha_f)^2, \quad (29)$$

$$\gamma = \frac{1}{2} - \alpha_m + \alpha_f, \quad (30)$$

$$\alpha_f = \frac{\alpha_a}{\alpha_a + 1} \text{ ja} \quad (31)$$

$$\alpha_m = \frac{2\alpha_a - 1}{\alpha_a + 1}, \quad (32)$$

missä  $\alpha_a \in [0, 1]$  on käyttäjän määrittämä numeerisen vaimennuksen arvo.

Numeerisen vaimennuksen vahvuuden määrittävä arvo  $\alpha_a$  voidaan valita siten, että menetelmä vaimentaa korkeat taajuudet kokonaan yhden aika-askeleen jälkeen ( $\alpha_a=0$ ), menetelmä ei vaimenna korkeita taajuuksia lainkaan ( $\alpha_a=1$ ) tai jotain näiden tapausten väliltä.

(Chung & Hulbert 1993)

## 4 YLEISTETYN ALFA-MENETELMÄN ALGORITMI

COMSOL Multiphysics -laskentaohjelmiston yleistetyn  $\alpha$ -menetelmän algoritmin ongelman löytämiseksi tarvittiin vertailutuloksia jollain toisella ohjelmalla. Toiseksi laskentamenetelmäksi valittiin Pythonilla suoritettu käsinlaskenta elementtimenetelmää käyttäen. Python-laskentaa varten täytyi kirjoittaa yleistettyä  $\alpha$ -menetelmää käyttävä algoritmifunktio, joka on esillä kuvissa 4-5. Tämän funktion kirjoittaminen mahdollisti vertailutulosten saamisen lisäksi myös sen, että COMSOL Multiphysics -laskentaohjelmistossa ilmenneiden ongelmien alkuperää voitiin rajata tarkemmin.

Työtä varten kirjoitettu odeIMPga-funktio tarvitsee toimiakseen Pythonin numpy-moduulin sekä inspect-moduulin isfunction-funktion. Argumentteina odeIMPga-funktiolle annetaan rakenteen massamatriisi  $M$ , rakenteen vaimennusmatriisi  $C$ , rakenteen jäykkyysmatriisi  $K$ , rakenteen kuormitusvektori  $f$ , laskennan lopetusaika  $t1$ , laskennassa käytettävä aika-askel  $dt$ , rakenteen alkusiirtymävektori  $u0$ , rakenteen alkunopeusvektori  $ud0$  ja laskennassa käytettävä numeerisen vaimennuksen vahvuuden säätävä arvo  $aa$ . Funktio palauttaa laskennan eri vaiheissa käytetyt ajat  $t$  sekä näitä aikoja vastaavat siirtymät  $y$  vektoreina. Näiden palautettujen arvojen avulla voi piirtää myöhemmin aika-siirtymäkuvaajan.

Funktion odeIMPga koodissa riveillä 3-4 tuodaan ensin funktiolle tarvittavat moduulit käyttöön, jonka jälkeen yleistettyä  $\alpha$ -menetelmää käyttävän algoritmin määrittely aloitetaan rivillä 6. Rivit 7-19 on varattu koodia selventäville kommenteille. Rivillä 20 tarkistetaan, onko funktiolle argumenttina annettu voimavektori  $f$  ajasta riippuvainen funktio inspect-moduulin isfunction-funktiolla. Jos  $f$  on funktio, asetetaan aluksi muuttujaan  $ft$  funktion  $f$  arvo ajan arvolla nolla rivillä 21. Jos taas  $f$  ei ole funktio, asetetaan riveillä 22-23 muuttujaan  $ft$  voimavektori  $f$  sellaisenaan. Rivillä 24 määritellään algoritmin parametrit  $am$  sekä  $af$  aiemmin esiteltyjen yhtälöiden (31) ja (32) mukaisesti ja rivillä 25 parametrit  $b$  ja  $g$  yhtälöiden (29) sekä (30) mukaisesti. Rivillä 26 asetetaan alkuehtoina funktiolle annetut alkusiirtymävektori  $u0$  ja alkunopeusvektori  $ud0$  muuttujiin  $u$  ja  $v$ . Rivillä 27 ratkaistaan tapauksen alkukiihtyvyyksivektori  $a$  numpy-moduulin linalg.solve-funktiolla yhtälön (24) mukaan. Rivillä 28 lasketaan aika-askelten lukumäärä  $n$  ja rivillä 29 luodaan vektorit  $t$  sekä  $y$  laskennan tulosten tallennusta varten.

```

3 import numpy as np
4 from inspect import isfunction
5
6 def odeIMPga(M,C,K,f,t1,dt,u0,ud0,aa=0.5):
7     # Ratkaisee liikeyhtälön  $Mu'' + Cu' + Ku = f(t)$ 
8     # kun  $t=[\theta,t1]$  implisiittisellä aikaintegroinnilla
9     # yleistetyllä alfa-menetelmällä: Chung et al, JoAM, 60:2, s. 371
10    # INPUT: M = massamatriisi
11    #         C = vaimennusmatriisi
12    #         K = jäykkymatriisi
13    #         f = voimavektori
14    #         t1 = lopetusaika, dt = aika-askel,
15    #         u0 = u( $\theta$ ), ud0=u'( $\theta$ ) (alkuehdot)
16    #         aa = menetelmän alpha-parametri (numeerinen vaimennus),
17    #         kun aa = 1 -> minimivaimennus ja kun aa = 0 -> maksimivaimennus
18    # OUTPUT: t (1xn) ajan hetket
19    #         y(i,:)=u^T @ t(i) (u:n transpoosi)
20    if isfunction(f):
21        ft=f(0.)
22    else:
23        ft=f
24    am=(2*aa-1)/(aa+1); af=aa/(aa+1)
25    b=((1.-am+af)**2)/4.; g=0.5-am+af
26    u=u0; v=ud0 # v=u'
27    a=np.linalg.solve(M,ft-np.dot(C,v)-np.dot(K,u)) # a=u''
28    n=int(round(t1/dt))+1
29    t=np.zeros(n); y=np.zeros((n,K.shape[0]))

```

Kuva 4. Python-koodi odeIMPga-funktiosta 1/2.

Rivillä 30 asetetaan aika-askeleen järjestysnumero i alkamaan nolasta ja lisätään vektorin y ensimmäiselle riville tapauksen alkusiirtymä u. Koko tarkasteluajan läpi käyvä while-silmukka aloitetaan rivillä 31. Seuraavaksi riveillä 32-33 asetetaan muuttujaan ft voimavektorin f arvo ajan alkuarvolla, jos f on funktio ja i on nolla. Riveillä 34-35 taas muutetaan muuttujan ft arvo vastaamaan yhtälöä (28) sekä yhtälön (23) yhtäsuuruusmerkin oikeaa puolta, jos voimavektori f on funktio. Jos f ei ole funktio, pidetään muuttujan ft arvona edelleen rivien 36-37 mukaan voimavektoria f. Rivillä 38 tallennetaan muuttujaan am1 alkukiihtyvyyksvektorin a arvo. Tämän jälkeen riveillä 39-42 ratkaistaan ensimmäisen aika-askeleen jälkeen voimassa oleva kiihtyvyyksvektori a yhtälön (23) mukaisesta liikeyhtälöstä numpy-moduulin linalg.solve-funktiolla. Liikeyhtälön ratkaisun jälkeen ratkaistaan riveillä 43-44 myös samalle askeleelle siirtymävektori u sekä nopeusvektori v yhtälöiden (21) ja (22) mukaisesti. Kun käsiteltävälle aika-askeleelle on saatu ratkaistua siirtymät ja nopeudet, lisätään vektorin t seuraavaksi alkioksi seuraavan aika-askeleen mukainen aika rivillä 45. Rivillä 46 tallennetaan ratkaistu siirtymävektori u vektorin y seuraavaksi riviksi. Lopuksi vielä rivillä 47 lisätään aika-askeleen järjestysnumeroa yhdellä, jonka jälkeen while-silmukka

aloittaa kierroksensa riviltä 31 uudestaan niin kauan, että päästään lopetusaikaan  $t_1$  asti. Silmukan päättyessä funktio palauttaa vektorit  $t$  ja  $y$  rivillä 48.

```

30 i=0; y[0,:]=u
31 while t[-1]+dt < t1:
32     if isfunction(f) and i == 0:
33         ft=f(t[i])
34     elif isfunction(f):
35         ft=f((1-af)*t[i]+af*t[i-1])
36     else:
37         ft=f
38     am1=a
39     a=np.linalg.solve(M*(1-am)+dt*g*(1-af)*C+dt**2*b*(1-af)*K,
40         ft-af*np.dot(C,v)-af*np.dot(K,u)-am*np.dot(M,a)\
41         -(1-af)*np.dot(C,v+dt*(1-g)*a)\
42         -(1-af)*np.dot(K,u+dt*v+dt**2*(0.5-b)*a))
43     u=u+dt*v+dt**2*((0.5-b)*am1+b*a)
44     v=v+dt*((1-g)*am1+g*a)
45     t[i+1]=t[i]+dt
46     y[i+1,:]=u.T
47     i+=1
48 return t,y

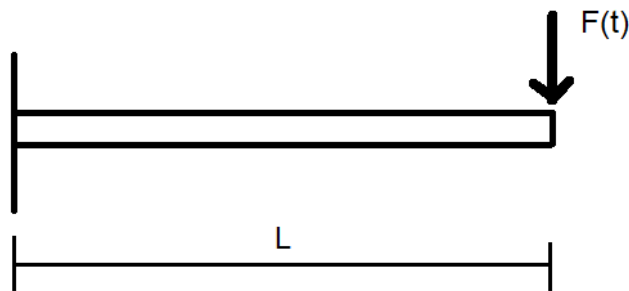
```

Kuva 5. Python-koodi odeIMPga-funktiosta 2/2.

## 5 ESIMERKKITAPAUKSET

### 5.1 Ulokepalkki

Ensimmäinen laskentatapaus, jonka tuloksia eri laskentamenetelmillä verrataan, on kuvassa 6 esillä oleva yksinkertainen ulokepalkki. Palkki on kiinnitetty kiinteästi vasemmasta päästään ja sen oikeassa päässä vaikuttaa kuvan mukaisesti voima  $F$ , joka riippuu ajasta  $t$ . Tarvittavat lähtöarvot laskentaan ovat pituus  $L=3$  m, kimmomoduuli  $E=210$  GPa, poikkipinta-ala  $A=18$  cm<sup>2</sup>, neliömomentti  $I_z=270$  cm<sup>4</sup>, tiheys  $\rho=7800$  kg/m<sup>3</sup>, maksimivoima  $F_0=5$  kN, voimaimpulssin kestoaika  $t_{imp}=0.5$  s, laskennan lopetusaika  $t_1=10$  s ja Rayleigh-vaimennusvakiot  $\alpha_r=0.5$  1/s ja  $\beta_r=10^{-4}$  s.



Kuva 6. Esimerkkitapaus 1: Ulokepalkki.

Palkin päässä vaikuttavaa voimaa kuvataan joko harmonisena (33) tai transienttina voimana (34) alla olevien yhtälöiden mukaan:

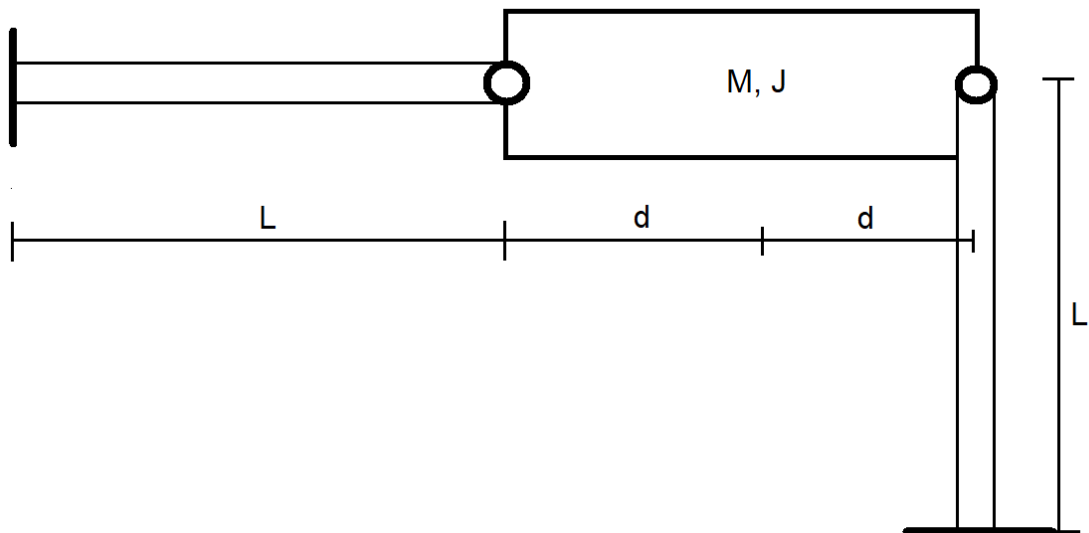
$$F(t)=F_0\cos(\Omega t), \quad \Omega=2\pi \quad (33)$$

$$F(t)=\frac{F_0 t}{t_0}, \quad 0 \leq t \leq 0.5 \text{ s.} \quad (34)$$

Kummassakin tapauksessa on selvitetty voiman vaikutuspisteen pystysuuntainen siirtymä ajan suhteen ja tulokset ovat nähtävillä kappaleessa 6.

## 5.2 Kone

Toinen käsiteltävistä esimerkkitalouksista on kuvassa 7 esillä oleva kahdella palkilla tuettu kone, jota mallinnetaan jäykkänä kappaleena. Koneen massa on  $M=50$  kg ja hitausmomentti  $J=1$  kgm<sup>2</sup>. Koneen sisällä liikkuvien osien epäkeskoisuus aiheuttaa koneen keskipisteeseen vaaka- ja pystysuuntaiset voimat  $F_x$  ja  $F_y$  sekä momentin  $M_z$ . Vaaka- ja pystysuuntaiset voimat sekä momentti ovat pyörimiskulmasta  $\theta$  riippuvia funktioita, joka taas on ajasta  $t$  riippuva funktio. Koneita tukevat palkit (neliöputkipoikkileikkauksen mitat 50x50x5 mm) on tuettu toisesta päästään kiinteästi ja toisesta päästään niveltuennalla koneeseen. Laskentaan tarvittavat lähtöarvot ovat teräspalkkien pituus  $L=1$  m, palkkien kimmomoduuli  $E=210$  GPa, palkkien tiheys  $\rho=7850$  kg/m<sup>3</sup>, koneen puolikas pituus  $d=50$  cm, koneen osien etäisyys pyörimiskeskistä  $e=0.5$  m, osien massa  $m=1$  kg ja Rayleigh-vaimennusvakiot  $\alpha_r=0.3$  1/s sekä  $\beta_r=10^{-4}$  s. Aikatarkastelu kyseiselle tapaukselle tehdään välillä  $0 \leq t \leq t_4$ , jossa  $t_1=0.1$  s,  $t_2=0.6$  s,  $t_3=0.7$  s ja  $t_4=1.0$  s.



Kuva 7. Esimerkkitalous 2: Kone.

Koneen keskipisteeseen vaikuttavat voimat sekä momentti ovat mallinnettu seuraavien yhtälöiden mukaisesti:

$$F_x(\theta) = em\Omega^2 \sin(\theta), \quad (35)$$

$$F_y(\theta) = em\Omega^2 \cos(\theta) \text{ ja} \quad (36)$$

$$M_z(\ddot{\theta}) = (em)^2 \ddot{\theta}, \quad (37)$$

missä  $\Omega$  on yhtälön (38) mukainen kulmanopeus,  
 $\theta$  on yhtälön (39) mukainen pyörimiskulma ja  
 $\ddot{\theta}$  on yhtälön (40) mukainen pyörimiskulman toinen derivaatta:

$$\Omega(t) = \frac{t}{t_1} \Omega_{max} \quad (38)$$

$$\theta(t) = \frac{1}{2} \Omega t \quad (39)$$

$$\ddot{\theta}(t) = (me)^2 \frac{\Omega_{max}}{t_1}, \quad (40)$$

missä taas  $\Omega_{max}$  on maksimikulmanopeus:

$$\Omega_{max} = 2\pi n_{max}, \quad (41)$$

missä  $n_{max}$  on maksimipyörimisnopeus.

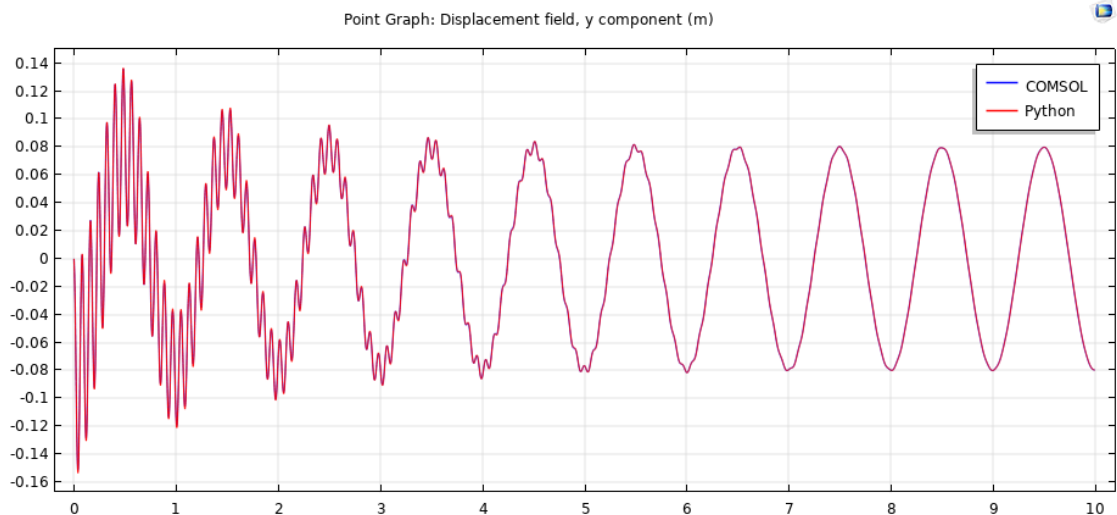
Tapausta tarkastellaan kolmella eri pyörimisnopeudella, jotka ovat  $0.5f_{min}$ ,  $f_{min}$  sekä  $1.5f_{min}$ . Näissä arvoissa  $f_{min}$  on rakenteen alin vaimennetun värähtelyn ominaistajuuus. Koneen sisäisten osien epäkeskoisuuden aiheuttamat kuormitukset muuttuvat siten, että koneen kiihdyttäessä aikavälillä  $0 \leq t < t_1$  koneen pyörimisnopeus kasvaa ja koneen jarruttaessa aikavälillä  $t_2 \leq t < t_3$  koneen pyörimisnopeus laskee lineaarisesti nolnaan. Epäkeskoisuudesta johtuva momentti vaikuttaa ainoastaan kiihdytyksen ja jarrutuksen aikana, mutta pysty- ja vaakasuuntaiset voimat pysyvät vakioina myös koneen tasaisen käynnin aikana aikavälillä  $t_1 \leq t < t_2$ . Laskennassa on ratkaistu koneen keskipisteen pystysuuntainen siirtymä ajan suhteen jokaisella pyörimisnopeudella ja tulokset ovat nähtävillä kappaleessa 6.

## 6 TULOSTEN TARKASTELO JA VERTAILU

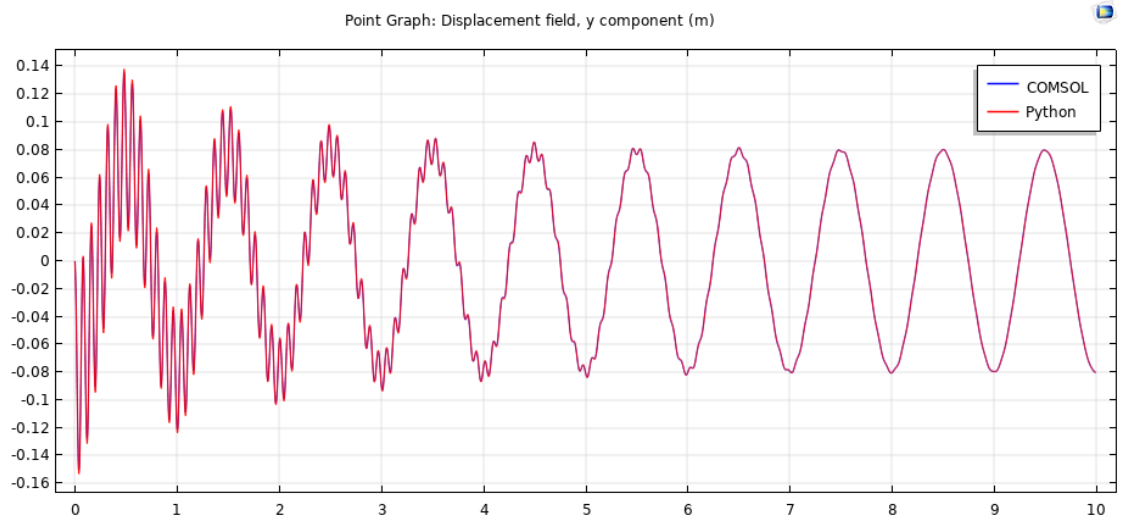
Kappaleessa 5 esiteltyjen esimerkkitapausten tulokset ratkaistiin sekä COMSOL Multiphysics -laskentaohjelmiston yleistä  $\alpha$ -menetelmää käyttävällä ratkaisijalla sekä tätä työtä varten luodulla yleisen  $\alpha$ -menetelmän laskenta-algoritmeilla. Kummassakin laskentatavassa käytettiin samoja parametreja joka tilanteessa, jotta tulosten vertailukelpoisuus pysyisi mahdollisimman hyvänä. Kummankin esimerkkitapausten jokainen kuormitustapaus ratkaistiin numeerisen vaimennuksen algoritmisen kertoimen  $\alpha_\alpha$  eri arvoilla. COMSOLista tulostetuissa kuvaajissa Python-algoritmeilla ratkaistu käyrä on merkitty punaisella ja COMSOLilla ratkaistu käyrä sinisellä.

### 6.1 Ulokepalkki

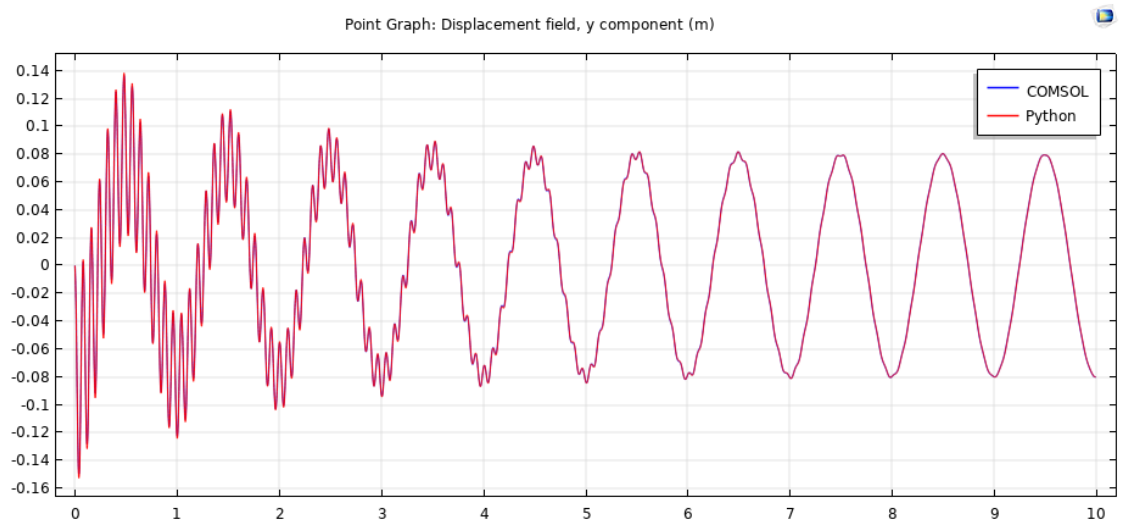
Kuvissa 8-12 on esillä esimerkkitapausten 1 voiman vaikutuspisteen pystysiirtymä ajan funktiona harmonisen voiman kuormitustapauksessa.



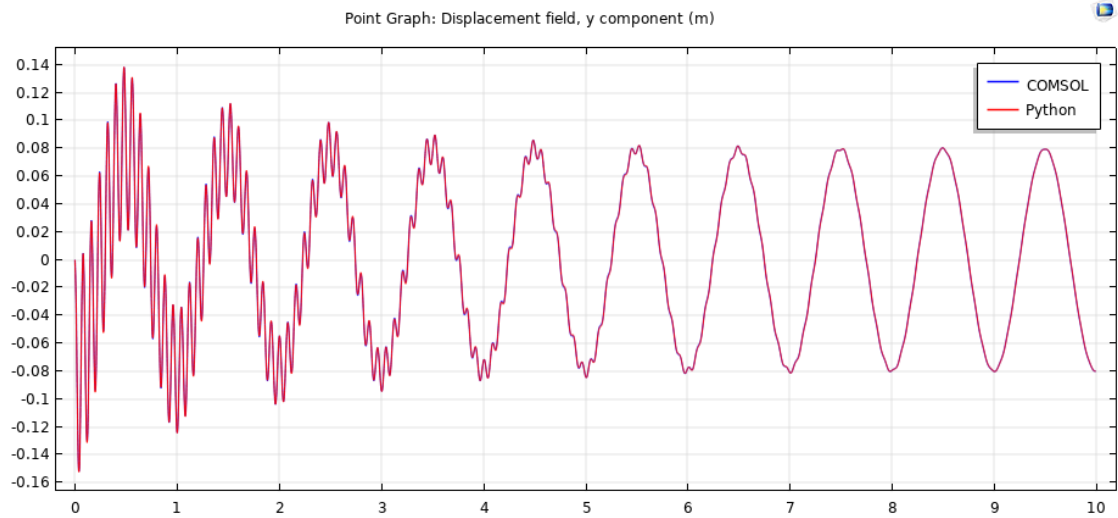
Kuva 8. Esimerkkitapaus 1: Harmoninen,  $\alpha_\alpha=0$



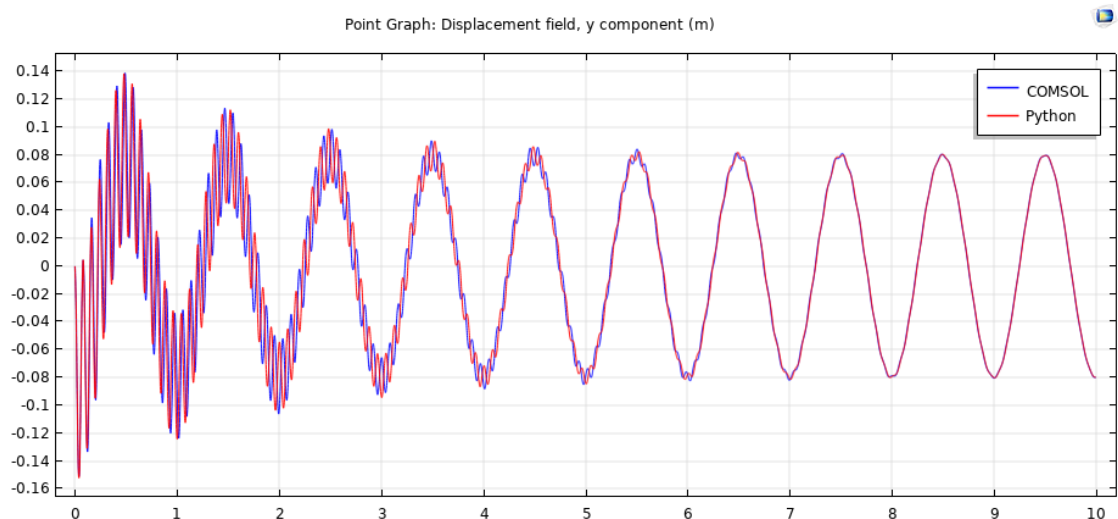
Kuva 9. Esimerkkitapaus 1: Harmoninen,  $\alpha_\alpha=0.25$



Kuva 10. Esimerkkitapaus 1: Harmoninen,  $\alpha_\alpha=0.5$



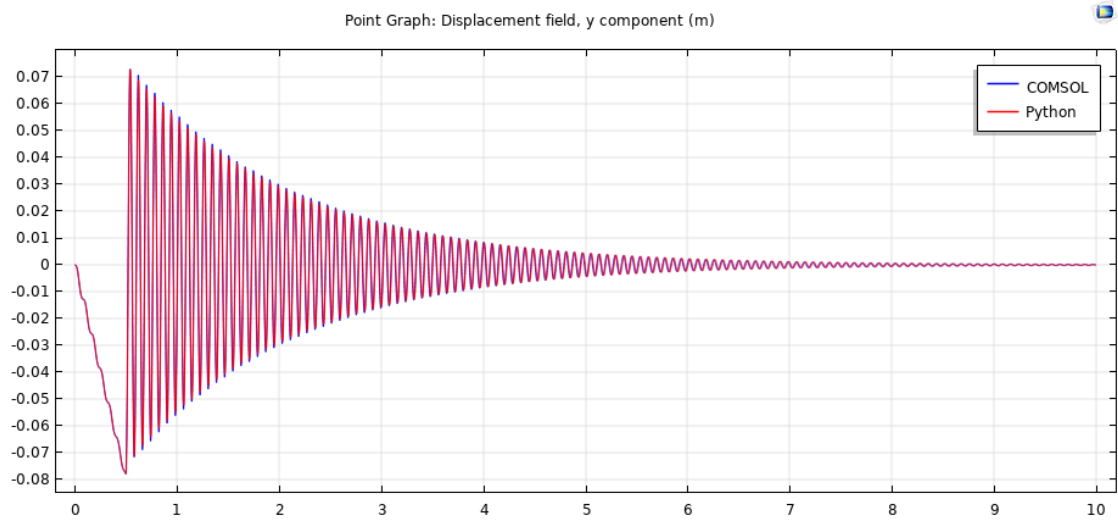
Kuva 11. Esimerkkitapaus 1: Harmoninen,  $\alpha_\alpha=0.75$



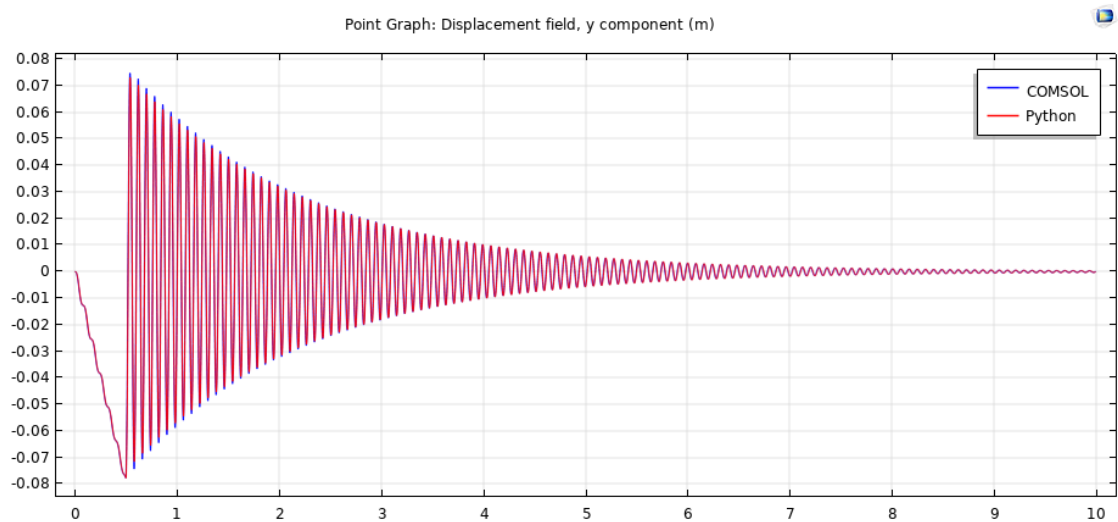
Kuva 12. Esimerkkitapaus 1: Harmoninen,  $\alpha_\alpha=1$

Kuvista voidaan nähdä, että COMSOLilla (sininen) ja Pythonilla (punainen) lasketut käyrät yhtyvät lähes täysin lukuun ottamatta kuvaa 12, jossa numeerisen vaimennuksen algoritminen kerroin on 1. Tällöin numeerisen vaimennuksen tulisi olla minimissään.

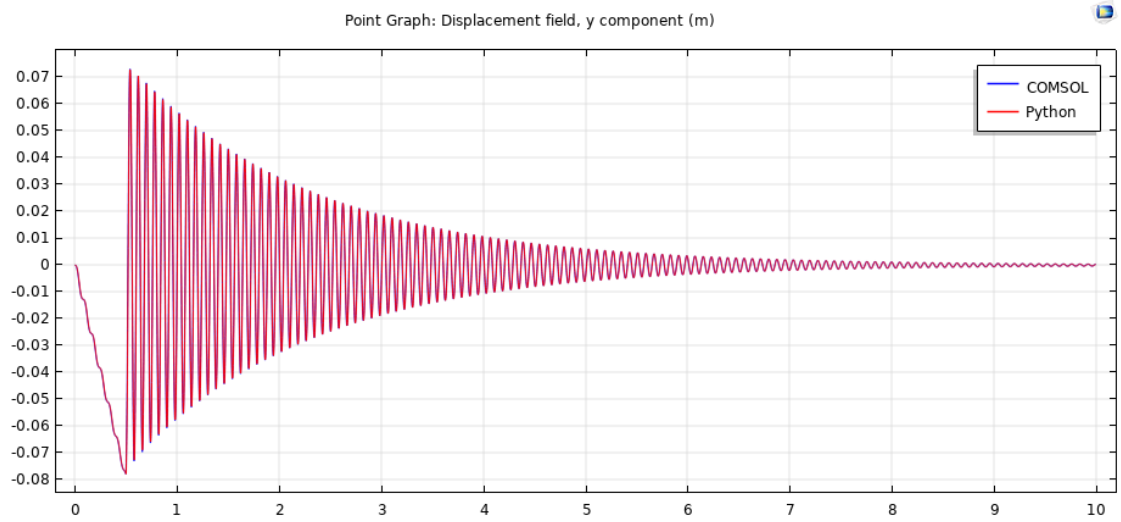
Seuraavissa kuvissa 13-17 on esitelty samasta tapauksesta voiman vaikutuspisteen pystysiirtymä ajan suhteen transientin voiman kuormitustapauksessa.



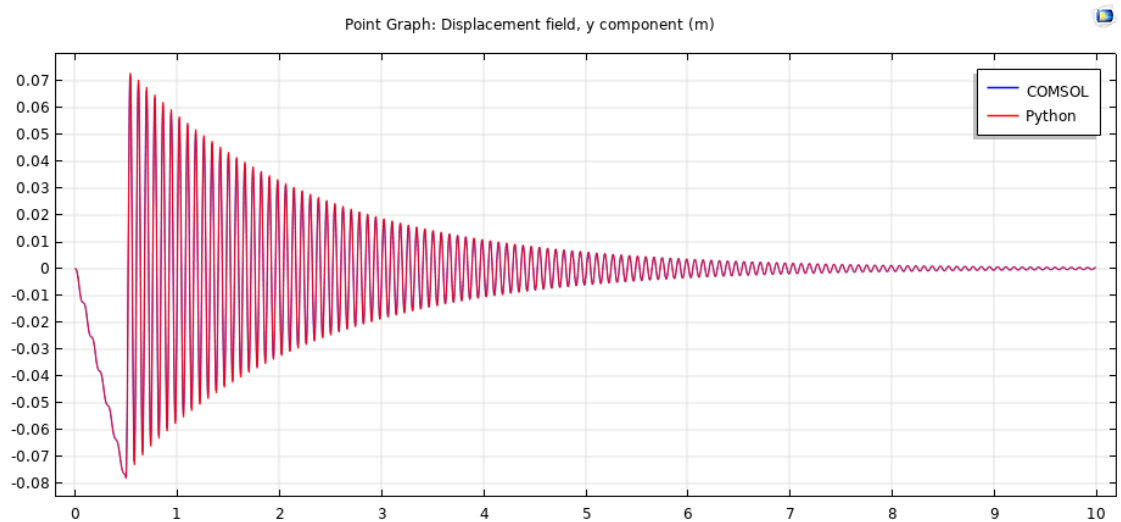
Kuva 13. Esimerkkitapaus 1: Transientti,  $\alpha_\alpha=0$



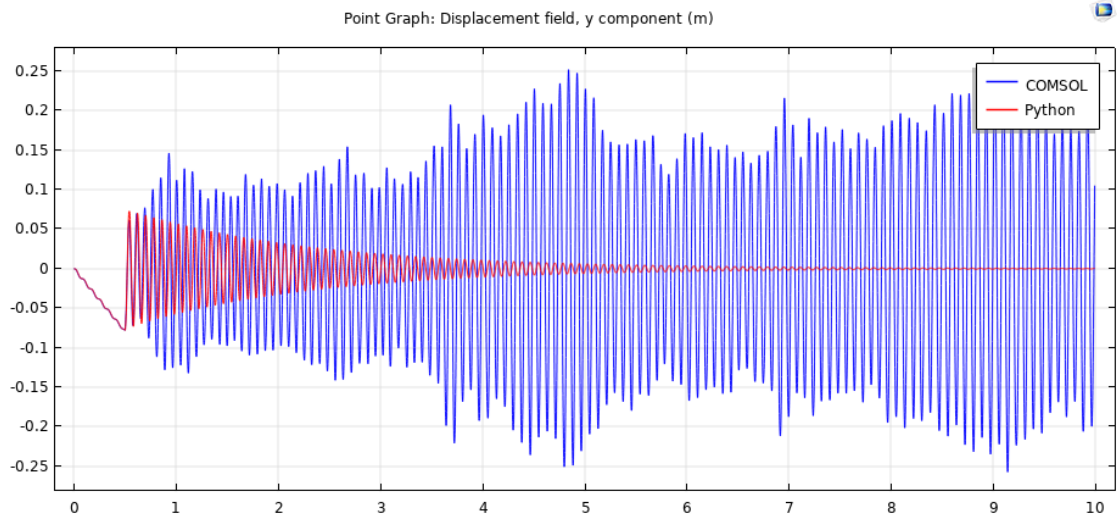
Kuva 14. Esimerkkitapaus 1: Transientti,  $\alpha_\alpha=0.25$



Kuva 15. Esimerkkitapaus 1: Transientti,  $\alpha_\alpha=0.5$



Kuva 16. Esimerkkitapaus 1: Transientti,  $\alpha_\alpha=0.75$

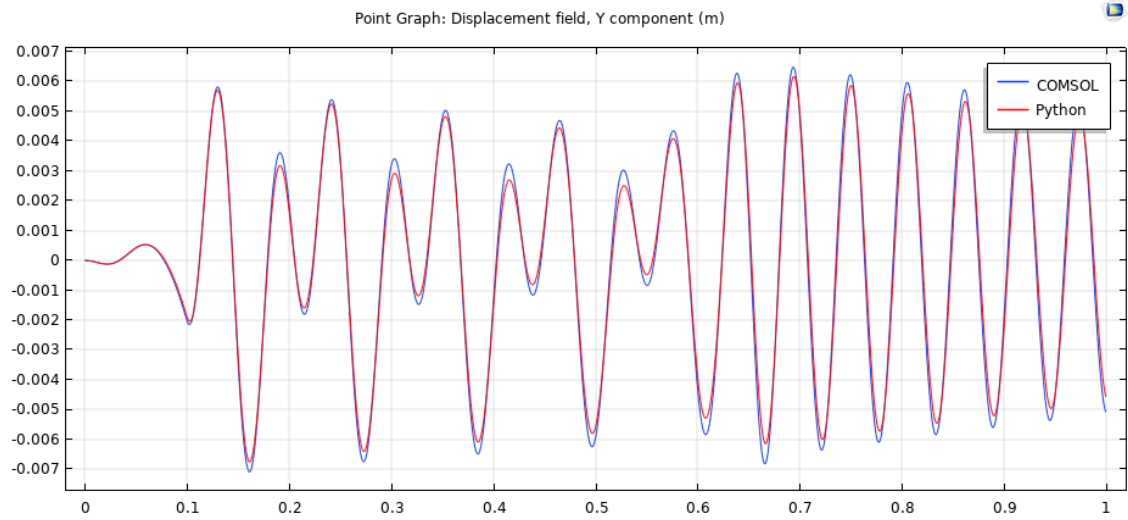


Kuva 17. Esimerkkitapaus 1: Transientti,  $\alpha_\alpha=1$

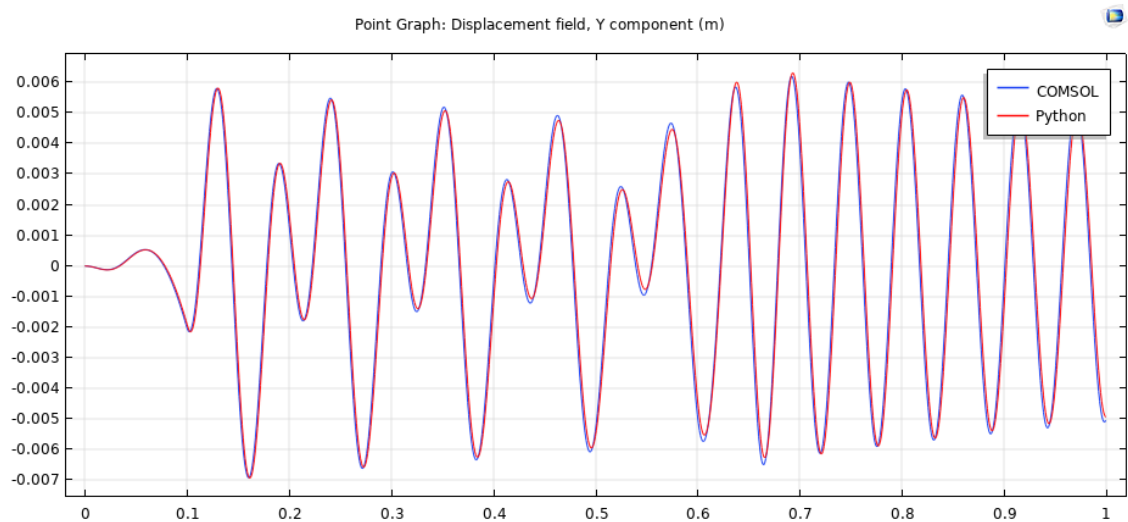
Kuvissa 13-16 aika-siirtymäkäyrät sekä COMSOLilla että Pythonilla laskettuna vastaavat taas erittäin hyvin toisiaan jokaisella  $\alpha_\alpha$ :n arvolla, mutta kuvassa 17 ero Pythonilla ja COMSOLilla lasketuissa tuloksissa on jo erittäin suuri. Tässä tapauksessa numeerisen vaimennuksen algoritmisen kertoimen arvo on 1, eli numeerisen vaimennuksen pitäisi olla minimissään. Jo tästä ensimmäisestä esimerkkitapauksesta voidaan siis päätellä, että COMSOL Multiphysics -ohjelman ongelmat yleistä  $\alpha$ -menetelmää käyttävässä ratkaisijassa liittyvät jollain tavalla numeerisen vaimennuksen algoritmiseen kertoimeen  $\alpha_\alpha$ .

## 6.2 Kone

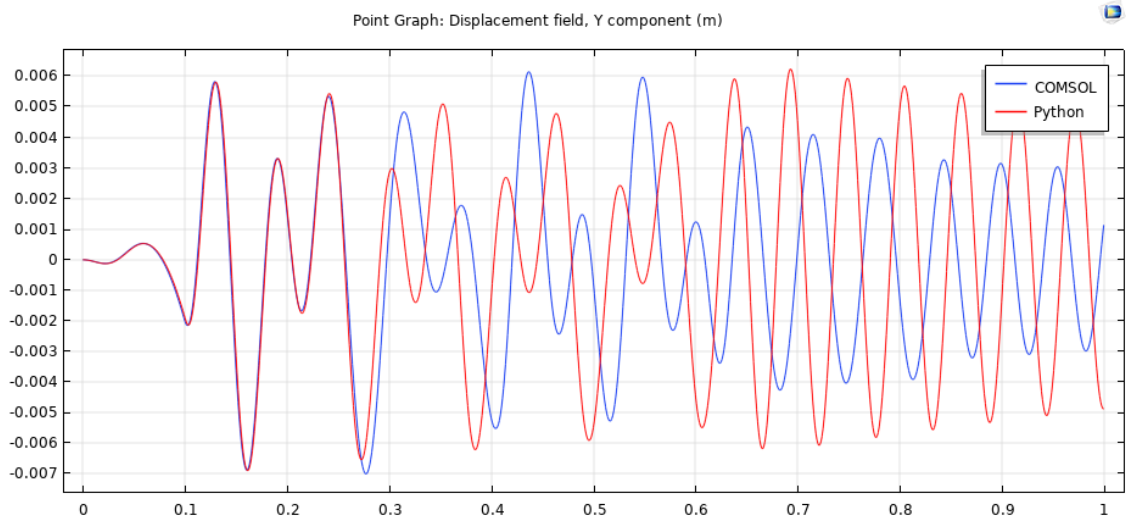
Kuvissa 18-26 on esitelty esimerkkitapauksen 2 koneen keskipisteen pystysuuntainen siirtymä ajan suhteen sekä COMSOLilla että Pythonilla laskettuna yleistä  $\alpha$ -menetelmää käyttäen. Kuvat 18-20 on laskettu käyttäen pyörimisnopeuden arvoa  $n_{max}=0.5f_{min}$ , kuvat 21-23 arvolla  $n_{max}=f_{min}$  ja kuvat 24-26 arvolla  $n_{max}=1.5f_{min}$ . Jokaisen pyörimisnopeuden tapauksesta on esitelty vain kuvaajat numeerisen vaimennuksen algoritmisella kertoimella 0, 0.5 ja 0.999. Toisena  $\alpha_\alpha$ :n arvona käytettiin arvoa 0.999 siitä syystä, ettei COMSOL suorittanut laskemista loppuun kahden tunninkaan aikana tässä esimerkkitapauksessa  $\alpha_\alpha$ :n arvolla 1, joka edelleen rajaa COMSOLin lähdekoodin virhettä kyseisen kertoimen toimintaan.



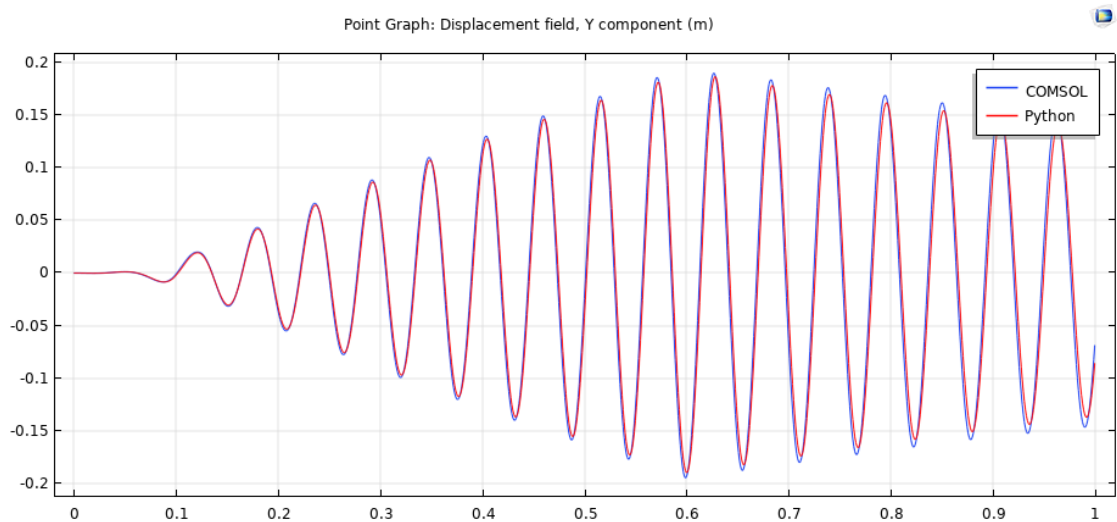
Kuva 18. Esimerkkitapaus 2:  $n_{max}=0.5f_{min}$ ,  $\alpha_\alpha=0$



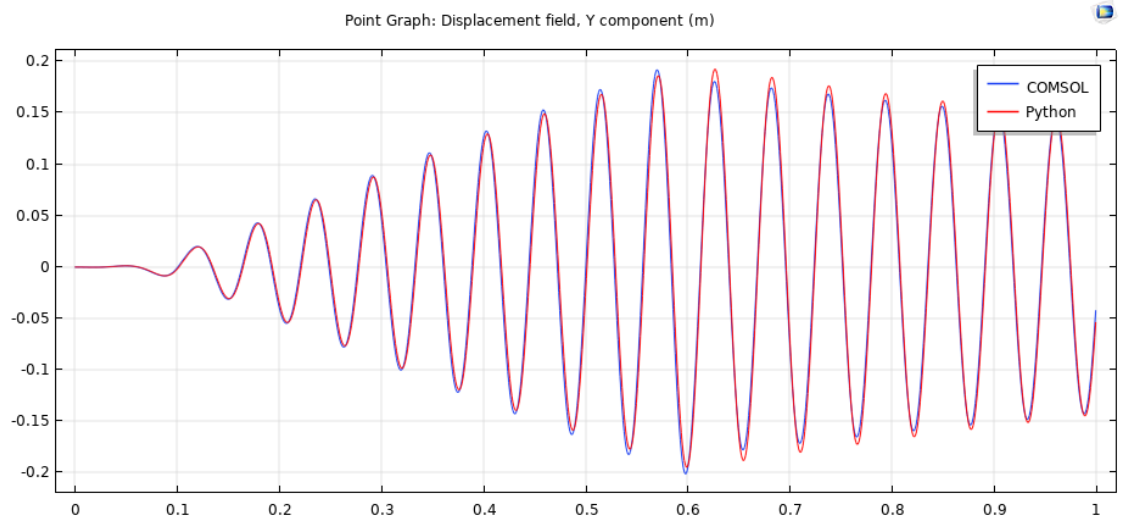
Kuva 19. Esimerkkitapaus 2:  $n_{max}=0.5f_{min}$ ,  $\alpha_\alpha=0.5$



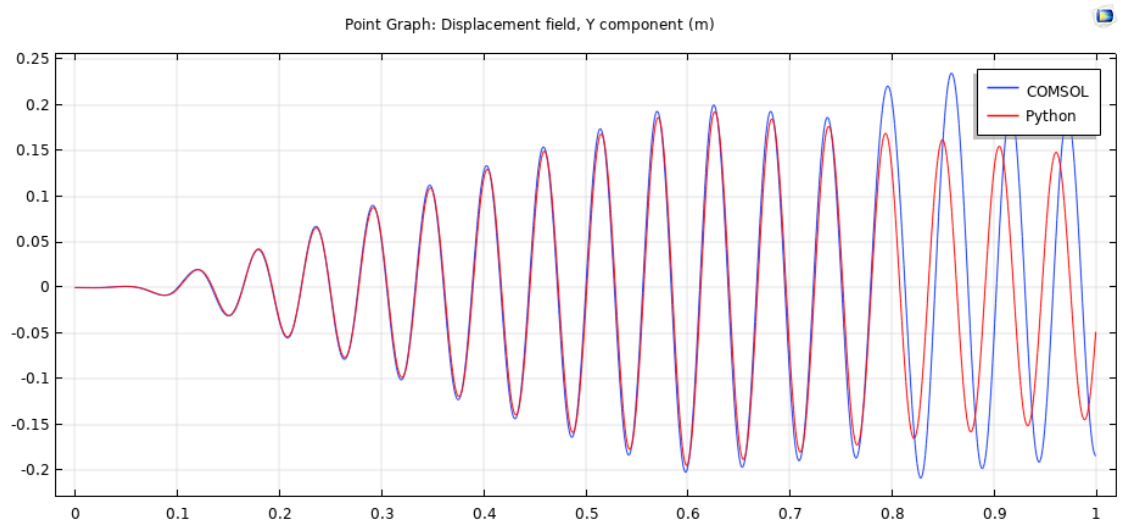
Kuva 20. Esimerkkitapaus 2:  $n_{max}=0.5f_{min}$ ,  $\alpha_a=0.999$



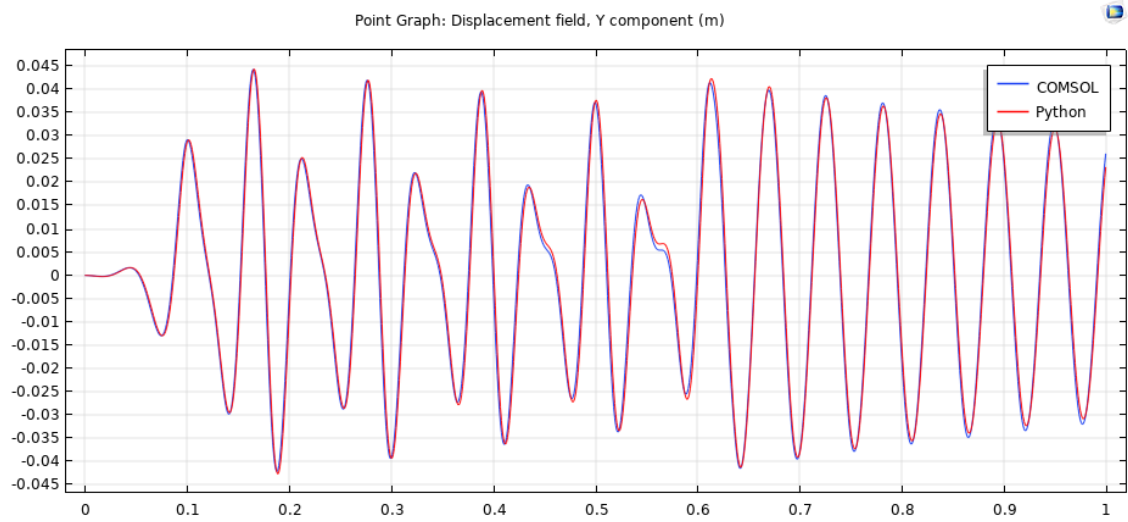
Kuva 21. Esimerkkitapaus 2:  $n_{max}=f_{min}$ ,  $\alpha_a=0$



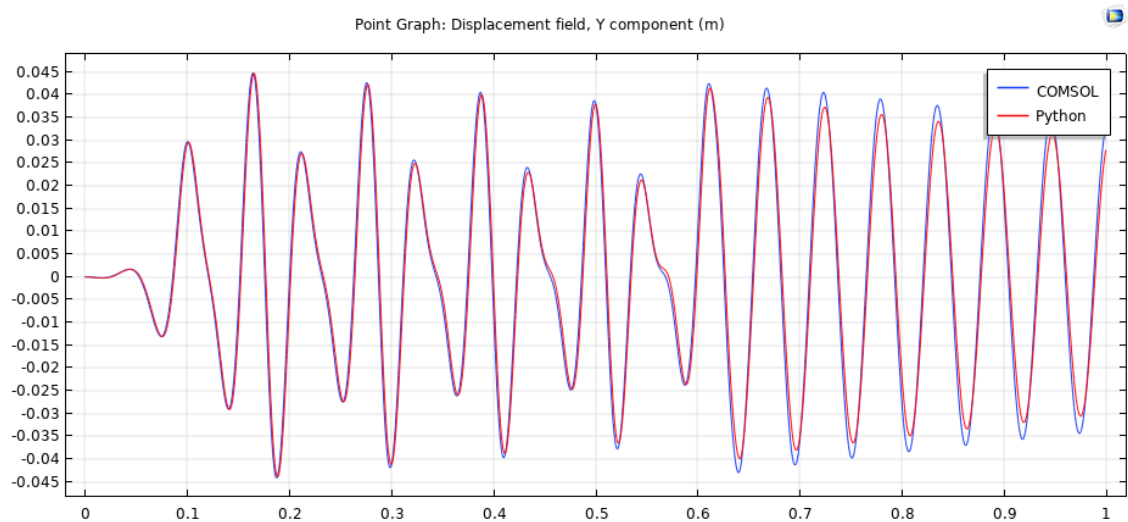
Kuva 22. Esimerkkitapaus 2:  $n_{max}=f_{min}$ ,  $\alpha_a=0.5$



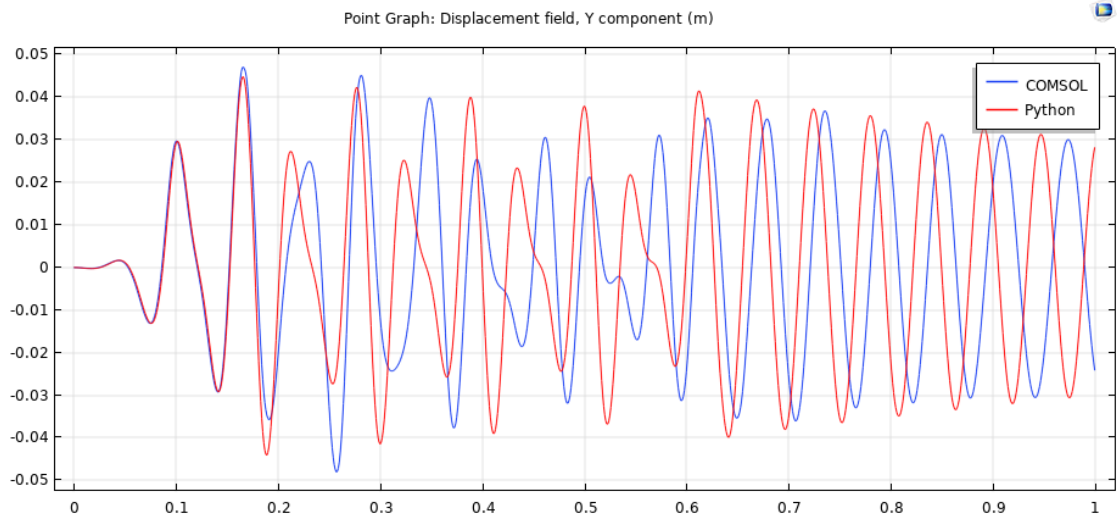
Kuva 23. Esimerkkitapaus 2:  $n_{max}=f_{min}$ ,  $\alpha_a=0.999$



Kuva 24. Esimerkkitapaus 2:  $n_{max}=1.5f_{min}$ ,  $\alpha_\alpha=0$

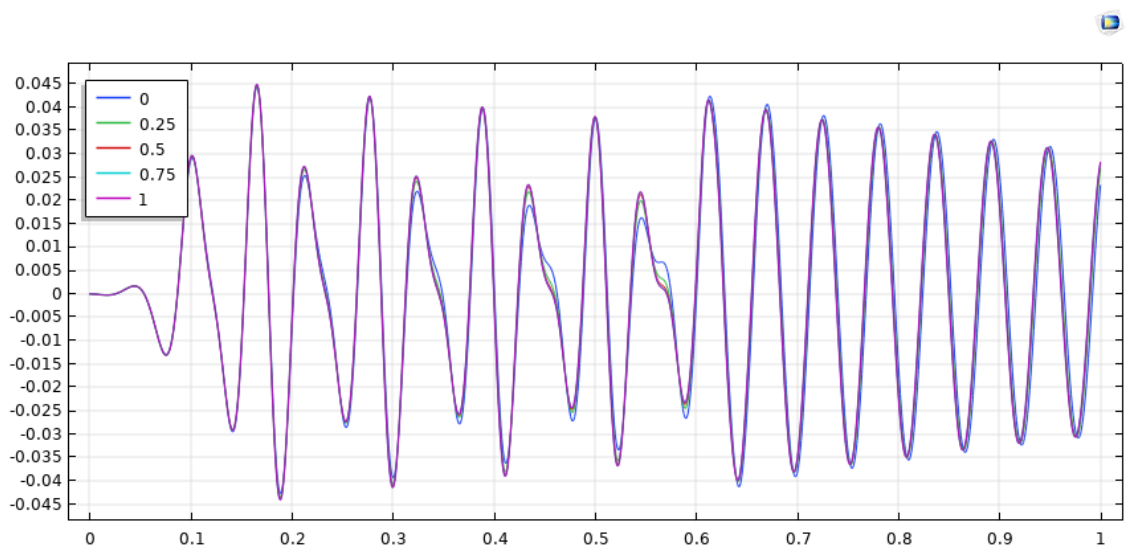


Kuva 25. Esimerkkitapaus 2:  $n_{max}=1.5f_{min}$ ,  $\alpha_\alpha=0.5$



Kuva 26. Esimerkkitapaus 2:  $n_{max}=1.5f_{min}$ ,  $\alpha_a=0.999$

Esimerkkitapauksen 2 tuloksista voidaan nähdä saman ilmiön toistuvan kuin esimerkkitapauksessa 1. Kun numeerisen vaimennuksen algoritmisen kertoimen  $\alpha_a$  arvo alkaa lähestyä lukua yksi, COMSOL- ja Python-laskennan tulokset alkavat erota selvästi toisistaan. Toisaalta, jos verrataan pelkästään Pythonilla laskettuja tuloksia esimerkiksi pyörimisnopeuden arvolla  $n_{max}=1.5f_{min}$  numeerisen vaimennuksen algoritmisilla kertoimilla 0, 0.25, 0.5, 0.75 ja 1 sekä piirretään ne samaan kuvaajaan (Kuva 27), huomataan, etteivät tulokset eroa läheskään yhtä paljon verrattuna COMSOLilla laskettuihin.



Kuva 27. Esimerkkitapaus 2: Python-tulokset,  $n_{max}=1.5f_{min}$

## 7 YHTEENVETO

Tämän työn tarkoituksena oli löytää virheitä COMSOL Multiphysics -laskentaohjelmiston yleistettyä  $\alpha$ -menetelmää käyttävästä implisiittisestä aikaintegrointialgoritmista. Virheen paikantamista varten luotiin Python-kielinen yleistä  $\alpha$ -menetelmää käyttävä laskenta-algoritmi, jonka avulla saatiin vertailukelpoisia tuloksia.

Esimerkkitapaukset, joista vertailutuloksia ratkaistiin, olivat vielä suhteellisen yksinkertaisia malleja verrattuna siihen, mitä elementtimenetelmällä yleensäkin ja yleistetyllä  $\alpha$ -menetelmällä voitaisiin ratkaista. Näistäkin esimerkkitapauksista voitiin selvästi nähdä, millä parametrien arvoilla COMSOL Multiphysicsin ratkaisija toimii, ja milloin tulokset ovat joko selvästi virheellisiä tai laskentaa ei pystytä suorittamaan loppuun asti.

COMSOLilla ja Python-algoritmilla ratkaistujen tulosten avulla voidaan paikantaa COMSOLin virheen löytyvän sen ratkaisijan numeerisen vaimennuksen kertoimesta  $\alpha_\alpha$ , tai pikemminkin muista tämän kertoimen avulla määritellyistä parametreista. Virhe löytyy mahdollisesti COMSOLin ratkaisijan muuttujien  $\alpha_f$ ,  $\alpha_m$ ,  $\beta$  tai  $\gamma$  määrittelyistä. Seuraava askel COMSOL Multiphysics -laskentaohjelmiston vian korjaamisessa voisi olla yhteydenotto kyseisen ohjelman kehittäjiin, ja tämän raportin lähettäminen heille.

## LÄHDELUETTELO

Butcher, J. C., 2003. Numerical Methods for Ordinary Differential Equations. 1. painos. New York: John Wiley & Sons, 440 s. ISBN 978-0-471-96758-3

Chung, J. & Hulbert, G. M., 1993. A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized- $\alpha$  Method. Journal of Applied Mechanics, Vol. 60, s. 371-375.

Liu, G. R. & Quek, S. S., 2003. Finite Element Method: A Practical Course. Oxford: Butterworth-Heinemann, 348 s. ISBN 0-7506-5866-5

Lumijärvi, J., 2018. Sauvaelementit. Elementtimenetelmät I -kurssin luentomateriaali. Oulun yliopisto: Konetekniikan koulutusohjelma. Syyslukukausi, 2018.

Rao, S. S., 2005. The Finite Element Method in Engineering. 4. painos. Amsterdam: Butterwoth-Heinemann, 664 s. ISBN 0-7506-7828-3