Rakenteiden Mekaniikka (Journal of Structural Mechanics)
Vol. 50, No 3, 2017, pp. 300 – 303
https://rakenteidenmekaniikka.journal.fi/index
https://doi.org/10.23998/rm.65040
©The author(s) 2017.
Open access under CC BY-SA 4.0 license.

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

Natural frequency calculations with JuliaFEM

Marja Rapo, Jukka Aho and Tero Frondelius[1]

Summary. This article presents a natural frequency analysis performed with JuliaFEM - an open-source finite element method program. The results are compared with the analysis results pruduced with a commercial software. The comparison shows that the calculation results between the two programs do not differ significantly.

Key words: natural frequency analysis, JuliaFEM, finite element method

Received 21 June 2017. Accepted 14 August 2017. Published online 21 August 2017

Introduction

It is reasonable to survey free, fast and easy-to-use alternatives for FEM calculations [9]. JuliaFEM [3] is an open-source [2] finite element solver written in the Julia programming language [1] that offers a free alternative for commercial FEM programs. The JuliaFEM software library is a framework that allows for the distributed processing of large finite element models across clusters of computers using simple programming models.

This example demonstrates a JuliaFEM natural frequency [4, 6, 7] calculation for an example model.

The example model and results

The example model is a bracket that is attached from its bolt holes to two adapter plates via tie contacts [8]. The plates are constrained from one side as fixed. Figure 1 shows the model and the locations of the contacts and constraints. The material is linear and elastic in this calculation. The material parameters are $E_1 = 208$ GPa, $\nu_1 = 0.3$ and $\rho_1 = 7800$ kg/m^3 for the bracket and $E_2 = 165$ GPa, $\nu_2 = 0.275$ and $\rho_2 = 7100$ kg/m^3 for the adapter plates.

The calculated natural frequencies [5] in the first six eigenmodes are presented in Table 1 in two decimal places. Also the error when comparing the JuliaFEM results to the commercial software results is presented in Table 1. Figure 2 shows a picture of the first six eigenmodes of the bracket. The source code for the simulation is shown in listing 1.

[1]Corresponding author. tero.frondelius@wartsila.com

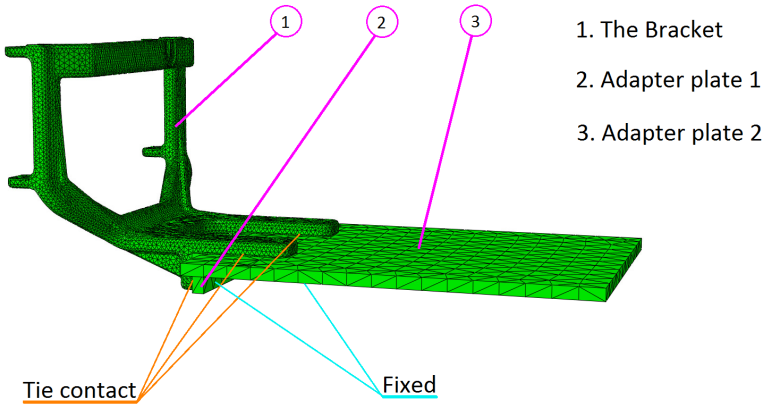

Figure 1: The model and the locations of the contacts and constraints.

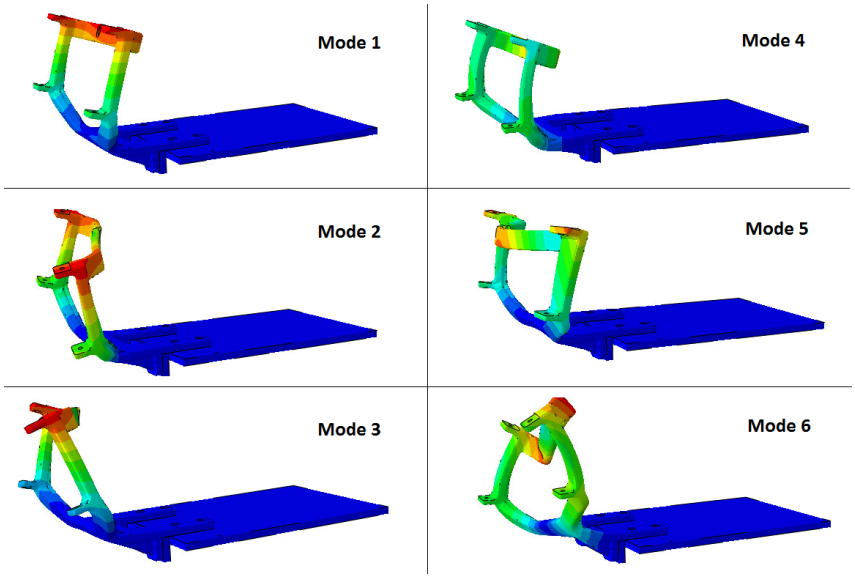

Figure 2: The first six Eigen modes of the Bracket.

Table 1: Natural frequencies of the modes 1-6 calculated with JuliaFEM and compared to the results of commercial software. The average of the absolute error is $\Delta\bar{f} = 0.082$, and the average relative error is $\Delta\bar{f}/\bar{f} = 0.026$ %.

Mode	f [Hz] JuliaFEM	f [Hz] Commercial software	Δf	Δf / f
1	111.38	111.36	0.023	0.021 %
2	155.03	154.98	0.050	0.032 %
3	215.40	215.34	0.059	0.027 %
4	358.76	358.67	0.091	0.025 %
5	409.65	409.61	0.044	0.011 %
6	603.51	603.29	0.225	0.037 %

Code Listing 1 JuliaFEM code

```
using JuliaFEM
using JuliaFEM.Preprocess
using JuliaFEM.Postprocess
using JuliaFEM.Abaqus: create_surface_elements

# read mesh
mesh = abaqus_read_mesh("LDU_ld_r2.inp")
info("element sets = ", collect(keys(mesh.element_sets)))
info("surface sets = ", collect(keys(mesh.surface_sets)))

# create field problem with two different materials
bracket = Problem(Elasticity, "LDU_Bracket", 3)
els1 = create_elements(mesh, "LDUBracket")
els2 = create_elements(mesh, "Adapterplate1", "Adapterplate2")
update!(els1, "youngs modulus", 208.0E3)
update!(els1, "poissons ratio", 0.30)
update!(els1, "density", 7.80E-9)
update!(els2, "youngs modulus", 165.0E3)
update!(els2, "poissons ratio", 0.275)
update!(els2, "density", 7.10E-9)
bracket.elements = [els1; els2]

# create boundary condition from node set
fixed = Problem(Dirichlet, "fixed", 3, "displacement")
fixed_nodes = mesh.node_sets[:Face_Constraint_1]
fixed.elements = [Element(Poi1, [nid]) for nid in fixed_nodes]
update!(fixed.elements, "displacement 1", 0.0)
update!(fixed.elements, "displacement 2", 0.0)
update!(fixed.elements, "displacement 3", 0.0)

""" A helper function to create tie contact. """
function create_interface(mesh::Mesh, slave::String, master::String)
    interface = Problem(Mortar, "tie contact", 3, "displacement")
    interface.properties.dual_basis = false
    slave_elements = create_surface_elements(mesh, slave)
    master_elements = create_surface_elements(mesh, master)
    nslaves = length(slave_elements)
    nmasters = length(master_elements)
    update!(slave_elements, "master elements", master_elements)
    interface.elements = [slave_elements; master_elements]
    return interface
  end

# call helper function to create tie contacts
tie1 = create_interface(mesh,
    "LDUBracketToAdapterplate1",
    "Adapterplate1ToLDUBracket")
tie2 = create_interface(mesh,
    "LDUBracketToAdapterplate2",
    "Adapterplate2ToLDUBracket")

# add field and boundary problems to solver
solver = Solver(Modal, bracket, fixed, tie1, tie2)
# save results to Xdmf data format ready for ParaView visualization
solver.xdmf = Xdmf("results")
# solve 6 smallest eigenvalues
solver.properties.nev = 6
solver.properties.which = :SM
solver()
```

Conclusion

The calculation results between the two programs do not differ significantly. On the basis of this example, it can be concluded that FEM calculations can be calculated with an open-source code instead of a commercial FEM software.

References

[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. URL https://doi.org/10.1137/141000671.

[2] Kevin Carillo and Chitu Okoli. The open source movement: a revolution in software development. *Journal of Computer Information Systems*, 49(2):1–9, 2008.

[3] Tero Frondelius and Jukka Aho. JuliaFEM —open source solver for both industrial and academia usage. *Rakenteiden Mekaniikka*, 50(3):229–233, 2017. URL https://doi.org/10.23998/rm.64224.

[4] Johannes Heilala, Teemu Kuivaniemi, Juho Könnö, and Tero Frondelius. Concept calculation tool for dynamics of generator set common baseframe. *Rakenteiden Mekaniikka*, 50(3):353–356, 2017. URL https://doi.org/10.23998/rm.64925.

[5] Walter C Hurty. Dynamic analysis of structural systems using component modes. *AIAA journal*, 3(4):678–685, 1965.

[6] Evgeniya Kiseleva, Juho Könnö, Niclas Liljenfeldt, Teemu Kuivaniemi, and Tero Frondelius. Topology optimisation of the in-line engine turbocharger bracket. *Rakenteiden Mekaniikka*, 50(3):323–325, 2017. URL https://doi.org/10.23998/rm.65071.

[7] Antti Korpela, Marko Jokinen, Teemu Kuivaniemi, and Tero Frondelius. W4L20 VEBIC genset dynamics —baseframe design. *Rakenteiden Mekaniikka*, 50(3):292–295, 2017. URL https://doi.org/10.23998/rm.64943.

[8] Michael A. Puso. A 3d mortar method for solid mechanics. *International Journal for Numerical Methods in Engineering*, 59(3):315–336, 2004. URL https://doi.org//10.1002/nme.865.

[9] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, and Robert Lee Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.

Marja Rapo and Tero Frondelius
Wärtsilä
Järvikatu 2-4
65100 Vaasa
marja.rapo@wartsila.com, tero.frondelius@wartsila.com

Jukka Aho
Global Boiler Works Oy
Lumijoentie 8
90400 Oulu
jukka.aho@gbw.fi