Rakenteiden Mekaniikka (Journal of Structural Mechanics)
Vol. 50, No 3, 2017, pp. 229 – 233
https://rakenteidenmekaniikka.journal.fi/index
https://doi.org/10.23998/rm.64224

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

JuliaFEM — open source solver for both industrial and academia usage

Tero Frondelius[1] and Jukka Aho

Summary. 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. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. The basic design principle is: Everything is non-linear. All physics models are non-linear from which the linearizations are made as special cases. This is a work in progress. Thus, if you share the vision, contribute and join the community.

Key words: JuliaFEM, FEM, open source, Julia, julialang, natural frequency, www.juliafem.org

Received 1 June 2017. Accepted 11 August 2017. Published online 21 August 2017.

Introduction

The JuliaFEM project was started in May 2015 and is MIT licensed [6]. However, the story actually starts from years back of studying and using other open-source FEM packages. The findings were mainly divided into two categories: namely first, academic projects with pure academic goals and parallelism missing and, second, fast parallel codes that were so hard to start using and typically written-in statically compiled languages. Customizing software written using such languages requires significant developer time, thus sacrificing human convenience and even more importantly, typically very limited developer resources for software execution speed.

JuliaFEM is an open-source finite element solver written in the Julia programming language, which is comprehensibly described in [3]. All the same reasoning are true for why Julia and why JuliaFEM, in other words why new programming language was needed. Reference [3] answers the first question throughly, thus the authors encourage readers to read it or visiting https://julialang.org/. JuliaFEM enables flexible simulation models, takes advantage of the scripting language interface which is easy to learn and embrace. In addition, it is a real programming environment where simulation can be combined with other analyses and work-flows. These features introduce a place for testing new ideas and simulation models to the academic world.

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

Julia introduces the following features: JIT/LLVM [10], multiple dispatch, metaprogramming, very powerful macros (full programming language with Julia syntax), static and dynamic typing. The speed of Julia is really close to the speed of C/FORTRAN code. Julia shows that one can have machine performance without sacrificing human convenience. Julia combines expertise from the diverse fields of computer science and computational science to create a new approach to numerical computing. Julia is designed to be easy and fast to use. One of the key features of Julia is multiple dispatch, a technique from computer science which automatically picks the right algorithm for the right circumstance [3].

Julia provides a wide range of libraries and a very skilled community in both skills and education, which enables a high level of abstraction in the programming code, such as using the automatic differentiation library ForwardDiff [16] for calculating gradients, jacobians and hessians for all types of field data. Using automatic differentiation in a FEM package is not a new idea; for example FEniCS [1], an open-source FEM solver, uses it comprehensively. Another important feature of the open-source software development is the method of building on top of existing open-source work. Thus JuliaFEM offers an io-interface to CODE ASTER [17], also an open source FEM solver. Developing JuliaFEM also encourages good practices, starting from unit testing both for smaller and larger functions and continuing to full integration testing of different platforms.

At the moment, users can perform the following analyses with JuliaFEM: elasticity, thermal, eigenvalue, contact mechanics, and quasi-static solutions. For visualization, JuliaFEM uses Paraview [2] which prefers XDMF [13] file format using XML to store light data and HDF [18] to store large data-sets, which is more or less the open-source standard.

Vision

On one hand, the vision of the JuliaFEM includes the opportunity for massive parallelization using multiple computers with MPI and threading as well as cloud computing resources in Amazon, Azure [7] and Google cloud services together with a company internal server [9]. And on the other hand, the real application complexity including the simulation model complexity as well as geometric complexity, see some examples in [4, 8, 19]. Not to forget that the reuse of the existing material models [11] as well as the whole simulation models are considered crucial features of the JuliaFEM package [5, 12].

Recreating the wheel again is definitely not anybody's goal, and thus we try to use and embrace good practices and formats as much as possible. We have implemented Abaqus/Calculix [21] input-file format support and maybe will in the future extend to Other FEM solver formats. Using modern development environments encourages the user towards fast development time and high productivity. For developing and creating new ideas and tutorials, we have used Jupyter notebooks [15] to make easy-to-use handouts.

The user interface for JuliaFEM is Jupyter Notebook [15], and Julia language itself is a real programming language. This makes it possible to use JuliaFEM as a part of a bigger solution cycle, including for example data mining, automatic geometry modifications, mesh generation, solution, and post-processing and enabling efficient optimization loops. In the next section a concrete example including statistical inference combined to FEM calculation is shown.

Big numerical example

Typical examples in industrial applications include non-linear solid mechanics [11], contact mechanics [20], finite strains [14], and fluid structure interaction problems. Here there is some simulation of machine parts having different amounts of elements and DOF's for comparison reasons. These simulations take a lot of computational resources, and here are specs of the used hardware: 24 x Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz with 512 GB total memory.

Table 1. This table demonstrates the solution times for some real industrial size FEM models

DOFS	Assembly time (s)	Solution time (s)	Total time (human readable)
3.0 M	458	312	16 min
10.8 M	3257	3255	2 h 4 min
12.6 M	3654	6318	3 h 4 min

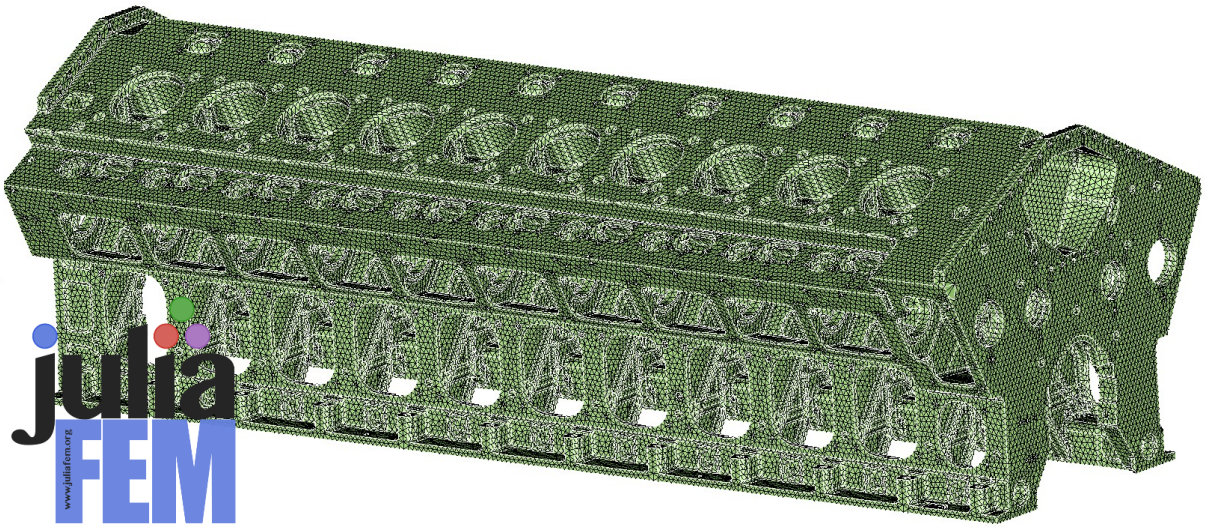

Figure 1. Real industrial size FEM model.

References

[1] Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100), 2015. URL https://doi.org/10.11588/ans.2015.100.20553.

[2] Utkarsh Ayachit et al. The ParaView guide: A parallel visualization application, kitware, 2015.

[3] 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.

[4] Tero Frondelius, Pasi Halla-aho, and Antti Mäntylä. Crankshaft development with virtual engine modelling. In *CIMAC Congress Helsinki*, 2016.

[5] Jussi Göös, Anton Leppänen, Antti Mäntylä, and Tero Frondelius. Large bore connecting rod simulations. *Rakenteiden Mekaniikka*, 50(3):275–278, 2017. URL https://doi.org/10.23998/rm.64658.

[6] Open Source Initiative et al. The MIT license, 2006.

[7] Roger Jennings. *Cloud Computing with the Windows Azure Platform*. Wrox Press Ltd., Birmingham, UK, UK, 2009. ISBN 0470506385, 9780470506387.

[8] Juho Könnö, Tero Frondelius, Thomas Resch, and Maria Jose Santos-Descalzo. Simulation based grid compliance. In *CIMAC Congress Helsinki*, 2016.

[9] Juho Könnö, Hannu Tienhaara, and Tero Frondelius. Wärtsilä digital design platform. *Rakenteiden Mekaniikka*, 50(3):234–238, 2017. URL https://doi.org/10.23998/rm.64621.

[10] Chris Lattner and Vikram Adve. LLVM: A compilation framework for lifelong program analysis and transformation. pages 75–88, San Jose, CA, USA, Mar 2004.

[11] Anton Leppänen, Asko Kumpula, Joona Vaara, Massimo Cattarinussi, Juho Könnö, and Tero Frondelius. Thermomechanical fatigue analysis of cylinder head. *Rakenteiden Mekaniikka*, 50(3):182–185, 2017. URL https://doi.org/10.23998/rm.64743.

[12] Antti Mäntylä, Jussi Göös, Anton Leppänen, and Tero Frondelius. Large bore engine connecting rod fretting analysis. *Rakenteiden Mekaniikka*, 50(3):239–243, 2017. URL https://doi.org/10.23998/rm.64914.

[13] E Mark et al. Enhancements to the extensible data model and format (xdmf). In *DoD High Performance Computing Modernization Program Users Group Conference, 2007*, pages 322–327. IEEE, 2007.

[14] Mikael Nyberg, Antti Mäntylä, and Tero Frondelius. Explosion simulation of pressurized components. *Rakenteiden Mekaniikka*, 50(3):198–200, 2017. URL https://doi.org/10.23998/rm.65076.

[15] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007. ISSN 1521-9615. URL https://doi.org/10.1109/MCSE.2007.53.

[16] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016. URL https://arxiv.org/abs/1607.07892.

[17] J. Russell and R. Cohn. *Code Aster*. Book on Demand, 2012. ISBN 9785511435350. URL https://books.google.fi/books?id=9iN8MAEACAAJ.

[18] The HDF Group. Hierarchical data format version 5, 2000-2010. URL http://www.hdfgroup.org/HDF5.

[19] Ilkka Väisänen, Antti Mäntylä, Antti Korpela, Teemu Kuivaniemi, and Tero Frondelius. Medium speed engine crankshaft analysis. *Rakenteiden Mekaniikka*, 50(3):341–344, 2017. URL https://doi.org/10.23998/rm.64916.

[20] Antti-Jussi Vuotikka, Mikael Nyberg, Heikki Karhinen, and Tero Frondelius. Contact sealing simulation of high pressured diesel injector. *Rakenteiden Mekaniikka*, 50(3): 313–317, 2017. URL https://doi.org/10.23998/rm.65060.

[21] C.O.M. Ximo. *Calculix*. Ject Press, 2011. ISBN 9786200185259. URL https://books.google.fi/books?id=8fSipwAACAAJ.

Tero Frondelius
Wärtsilä
Järvikatu 2-4
65100 Vaasa
tero.frondelius@wartsila.com

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