




# MoFEM: An open source, parallel finite element library

Łukasz Kaczmarczyk<sup>1</sup>, Zahur Ullah<sup>2</sup>, Karol Lewandowski<sup>1</sup>, Xuan Meng<sup>1</sup>, Xiao-Yi Zhou<sup>3</sup>, Ignatios Athanasiadis<sup>1</sup>, Hoang Nguyen<sup>1</sup>, Christophe-Alexandre Chalons-Mouriesse<sup>1</sup>, Euan J. Richardson<sup>1</sup>, Euan Miur<sup>1</sup>, Andrei G. Shvarts<sup>1</sup>, Mebratu Wakeni<sup>1</sup>, and Chris J. Pearce<sup>1</sup>

<sup>1</sup> Glasgow Computational Engineering Centre, James Watt School of Engineering, University of Glasgow, Glasgow, G12 8QQ, UK <sup>2</sup> School of Mechanical & Aerospace Engineering, Queen's University, Belfast, BT7 1NN, UK <sup>3</sup> Department of Bridge Engineering, Tongji University, 1239 Siping Road, Shanghai, 200092, China

DOI: [10.21105/joss.01441](https://doi.org/10.21105/joss.01441)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Jed Brown](#) 

## Reviewers:

- [@tjhei](#)
- [@chrisrichardson](#)
- [@Kevin-Mattheus-Moerman](#)
- [@vijaysm](#)
- [@chennachaos](#)

Submitted: 04 May 2019

Published: 25 January 2020

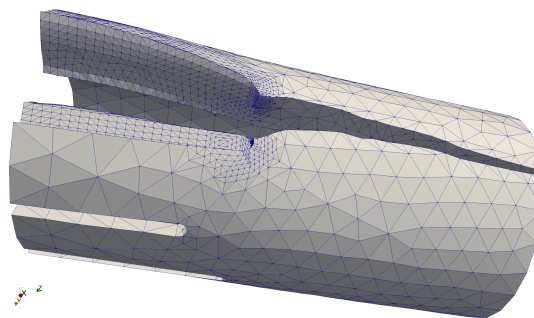
## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

## Introduction and Motivation

MoFEM (Mesh-oriented Finite Element Method) is a C++ library for managing complexities related to the finite element method (FEM). FEM is a widely used numerical approach for solving partial differential equations (PDEs) arising in various physical problems. MoFEM is developed to provide a finite element library incorporating modern approximation approaches and data structures for engineers, students and academics.

MoFEM belongs to a class of open source finite element libraries, such as Deal.II (Arndt et al., 2019), MFEM (Kolev & Dobrev, 2010), libMesh (Kirk, Peterson, Stogner, & Carey, 2006), FEniCS (Alnæs et al., 2015) and FreeFEM++ (Hecht, 2012), which provide users with generic tools for solving PDEs and developers with frameworks for implementing bespoke finite elements. MoFEM is specifically designed to solve complex engineering problems, enabling seamless integration of meshes that comprise multiple element types and element shapes, which are typically encountered in industrial applications. The development of MoFEM has been primarily targeting the problem of crack propagation for structural integrity assessment of safety critical structures (see [Figure 1](#)).



**Figure 1:** Brittle crack propagation.

The need for solutions to increasingly complex problems demands control over numerical errors; otherwise, it would be difficult to distinguish discretisation artefacts from the real physical phenomena. A brute force approach based on mesh refinement (so-called *h-adaptivity*) leads to a low polynomial convergence rate and, therefore, is severely limited by the current computing capabilities. A more elegant approach was paved by Guo & Babuška (1986), who showed that if one could simultaneously increase the mesh density and the interpolation order, i.e. employ

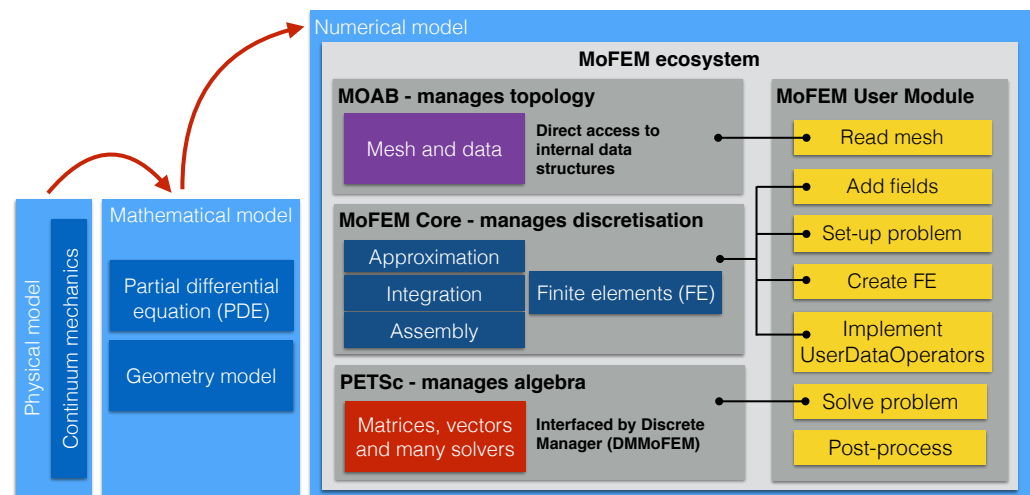
*hp*-adaptivity, exponential convergence is achievable. This has been seen as the ‘Holy Grail’ of numerical methods.

However, raising the order of approximation comes with a computational cost: the algebraic solver time and the matrix assembly time are increased. Unfortunately, there is no universal solution to tackle these two difficulties simultaneously. To reduce the solver time, properties of hierarchical and heterogeneous approximation bases, constructed using Legendre (Ainsworth & Coyle, 2003) or Jacobi (Fuentes, Keith, Demkowicz, & Nagaraj, 2015) polynomials, can be exploited. Such bases permit to increase approximation order locally and produce sparse and well-conditioned systems of equations. Moreover, algebraic systems constructed with hierarchical bases can be naturally restricted to lower dimensions for use as a preconditioner, e.g. with multi-grid solvers. This approach is ideal for elliptic problems such as solid elasticity; however, for hyperbolic problems the efficiency bottleneck could be in the assembly time, e.g. for acoustic wave propagation. In the latter case, different approximation bases, such as the Bernstein-Bézier basis (Ainsworth, Andriamaro, & Davydov, 2011), allowing for fast numerical integration, could be an optimal solution. Finally, the adaptive choice of the mesh density and the approximation order is driven by numerical errors, which can be effectively estimated if error evaluators are embedded into the FE formulation. This leads to a family of mixed or mixed-hybrid finite elements that are stable if combinations of different approximation spaces ( $H^1$ ,  $\mathbf{H}(\text{curl})$ ,  $\mathbf{H}(\text{div})$  and  $L^2$ ) are used.

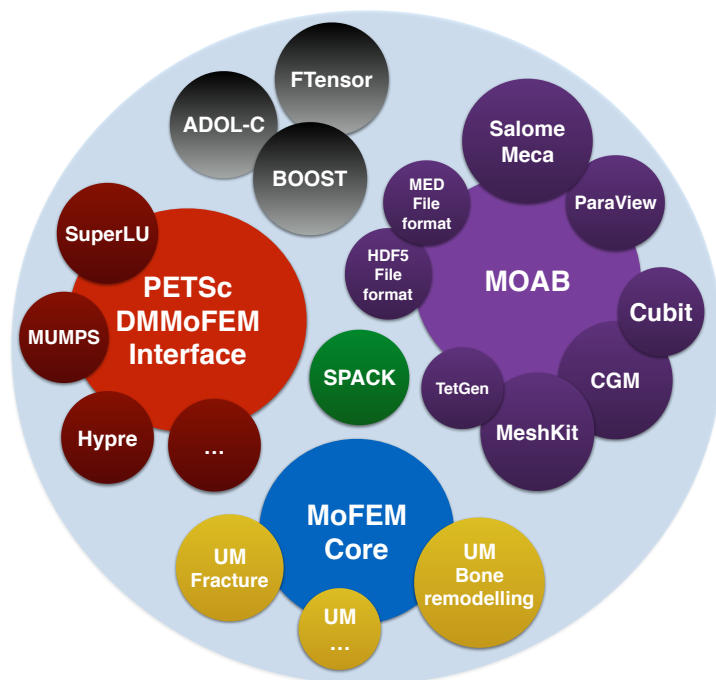
MoFEM incorporates all solutions discussed above for *hp*-adaptivity, enabling rapid implementation of the finite element method, i.e. relieving the user from programming complexities related to bookkeeping of degrees of freedom (DOFs), finite elements, matrix assembly, etc. Therefore, MoFEM provides efficient tools for solving a wide range of complex engineering-related problems: multi-dimensional (involving solid, shell and beam elements), multi-domain (e.g. interaction between solid and fluid), multi-scale (e.g. homogenisation with  $\text{FE}^2$ ) and multi-physics (e.g. thermo-elasticity). Moreover, MoFEM supports mixed meshes, consisting of different element types, for example, tetrahedra and prisms.

## Design

Modern finite element software is an ecosystem that manages various complexities related to mesh and topology, sparse algebra and approximation, numerical integration and dense tensor algebra at the integration point level. However, MoFEM has not developed all these capabilities from scratch. Instead, MoFEM integrates advanced scientific computing tools for sparse algebra from [PETSc](#) (Portable, Extensible Toolkit for Scientific Computation) (Balay et al., 2019), components for handling mesh and topology from [MOAB](#) (Mesh-Oriented Database) (Tautges, Meyers, Merkley, Stimpson, & Ernst, 2004) and data structures from [Boost libraries](#) (“Boost Web page,” 2020). An illustration of how these packages are utilised in MoFEM is shown in [Figure 2](#). Moreover, MoFEM’s core library is developed to manage complexities directly related to the finite element method. Therefore, each part of this ecosystem has its own design objectives, and appropriate programming tools can be selected from a spectrum of solutions. Resilience of the MoFEM ecosystem is ensured since the underpinning components have dynamic and established groups of developers and a significant number of users. Different components employed in the ecosystem are illustrated in [Figure 3](#), including popular pre- and post-processing software.



**Figure 2:** Basic design of MoFEM. Adopted from (“MoFEM Web page,” 2020).



**Figure 3:** Ecosystem of MoFEM. Adopted from (“MOAB Web page,” 2020).

Traditional finite element codes are element-centric, i.e. the type of an element defines the approximation space and basis. Therefore, they are not able to fully exploit the potential of emerging approximation methods. On the contrary, the design of data structures for approximation of field variables in MoFEM is independent of the specific finite element formulation, e.g. Lagrangian, Nédélec or Raviart-Thomas, since each finite element is constructed by a set of lower dimension entities on which the approximation fields are defined. Consequently, different approximation spaces ( $H^1$ ,  $\mathbf{H}(\text{curl})$ ,  $\mathbf{H}(\text{div})$  and  $L^2$ ) can be suitably combined in a finite element to create new stable mixed formulations for solving complex problems efficiently.

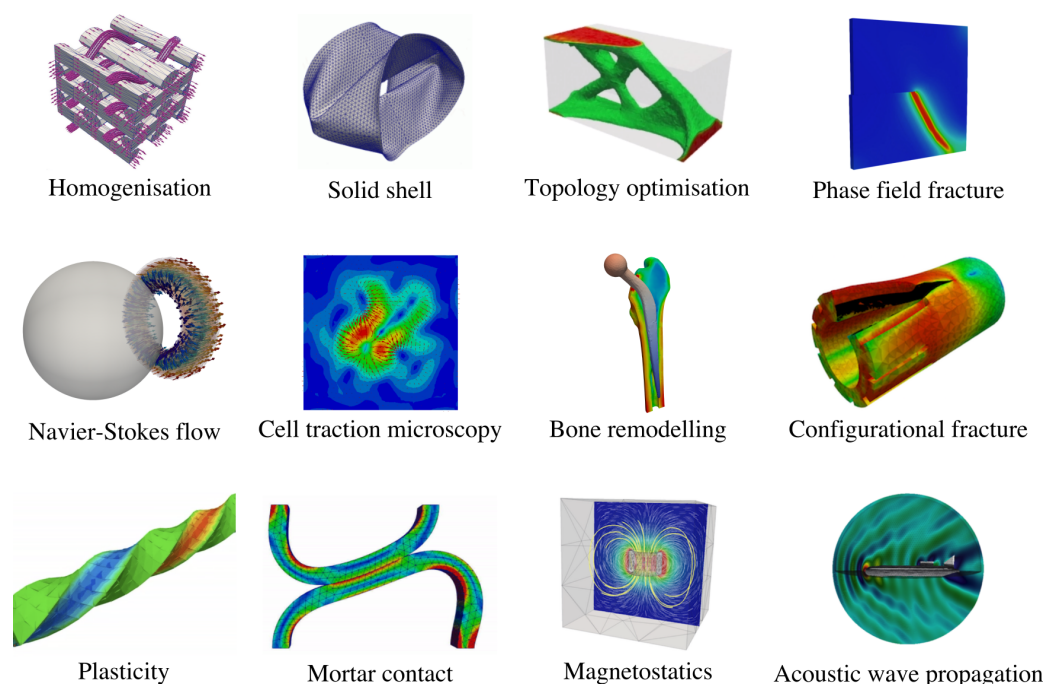
MoFEM data structures enable easy enrichment of approximation fields and modification of basis functions, for example, for resolution of singularity at a crack front. Applying such technique,

it is almost effortless to construct transition elements between domains with different problem formulation and physics, e.g. from two-field mixed formulation to a single-field. One can easily implement elements with an anisotropic approximation order, which depends on direction in curvilinear basis, e.g. solid shells with arbitrary higher approximation order on the surface and arbitrary lower order through the thickness of the shell. This approach also sets a benchmark on how finite element codes could be implemented, introducing a concept of pipelines of *user-defined data operators* acting on fields that are associated with entities (vertices, edges, faces and volumes). Such an approach simplifies code writing, testing and validation, making the code more resilient to bugs.

Furthermore, MoFEM's core library provides functionality for developing *user modules* (see [Figure 2](#)) where applications for particular problems can be implemented. This toolkit-like structure allows for independent development of modules with different repositories, owners and licences, being suitable for both open-access academic research and private industrial sensitive projects. At the same time, the MoFEM core library is licensed under the [GNU Lesser General Public License](#) and it can be deployed and developed using the package manager [Spack](#); see the [MoFEM installation instructions](#) for more details.

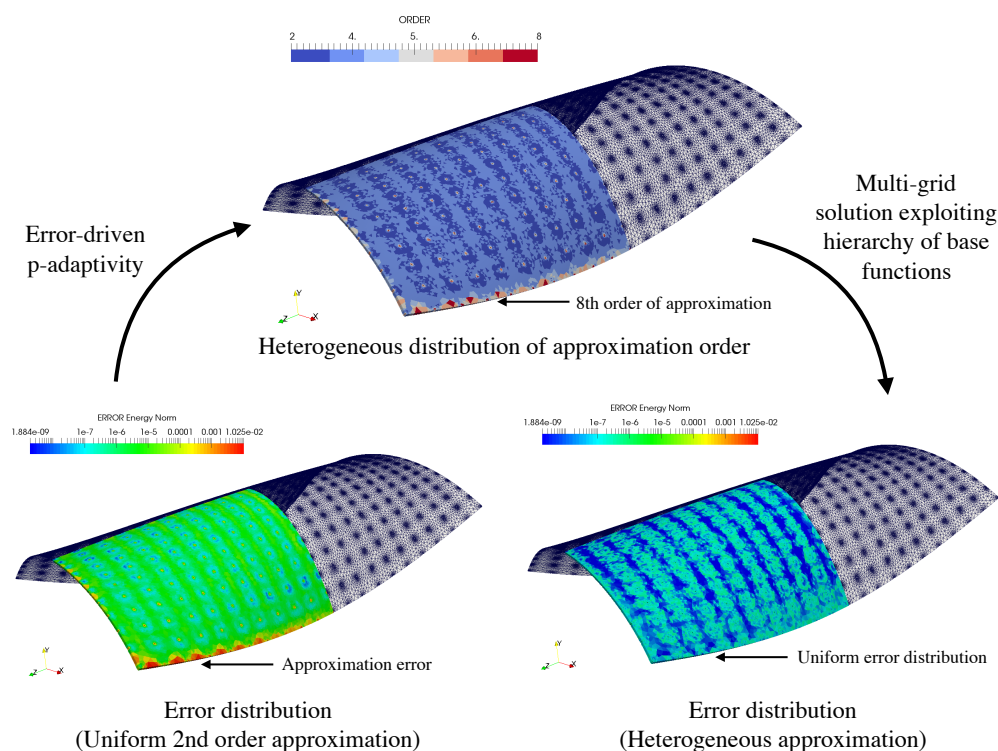
## Examples and Capabilities

MoFEM was initially created to solve the problem of brittle crack propagation using thermodynamically consistent framework (Kaczmarczyk, Ullah, & Pearce, 2017). Over time, the domain of applications expanded to include computational homogenisation (Ullah et al., 2019), bone remodelling and fracture (Lewandowski, Kaczmarczyk, Athanasiadis, Marshall, & Pearce, 2020), modelling of gel rheology (Richardson, 2018) and acoustics problems. Moreover, MoFEM includes an extensive library of example applications such as soap film, solid shell, topology optimisation, phase field fracture, Navier-Stokes flow, cell traction microscopy, bone remodelling, configurational fracture, plasticity, mortar contact, magnetostatics and acoustic wave propagation as shown in [Figure 4](#).



**Figure 4:** Examples of user modules implemented using MoFEM.

MoFEM is designed to provide efficient tools for solving a wide variety of user-defined problems. In Figure 5 an example of error-driven  $p$ -adaptivity is presented, where a hierarchical approximation basis with a multi-grid solver is applied to the perforated Scordelis-Lo Roof problem (Kaczmarczyk, Ullah, & Pearce, 2016).



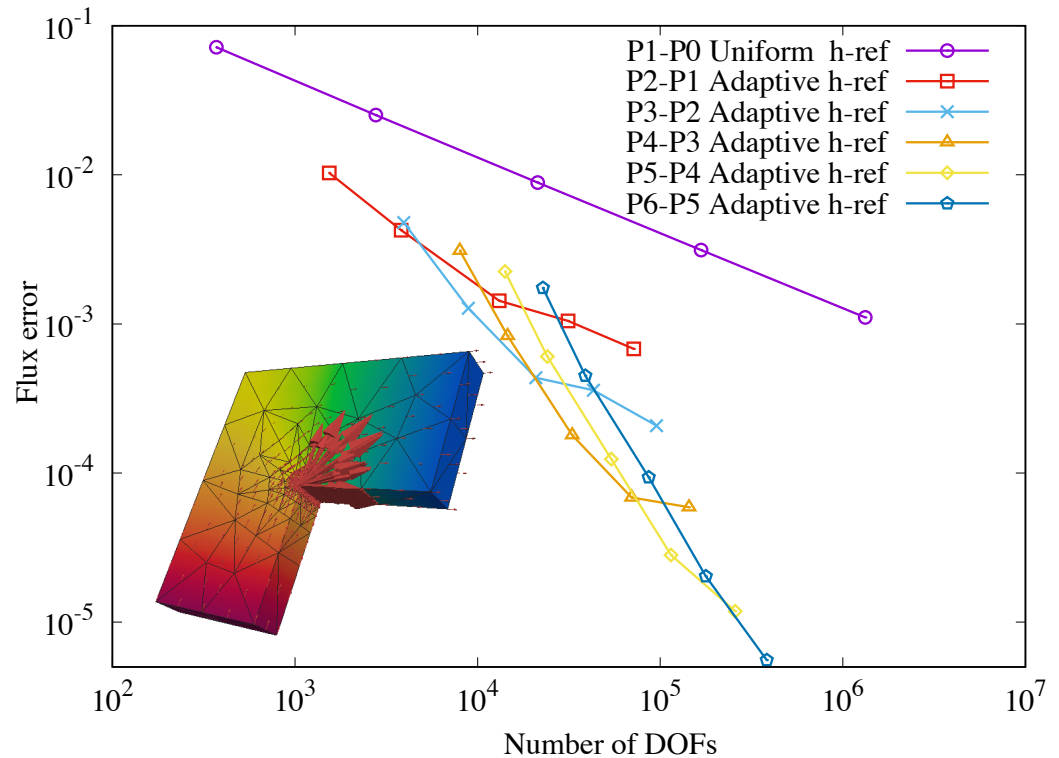
**Figure 5:** Example of  $p$ -adaptivity for hierarchical and heterogeneous approximation with multi-grid solver applied to the perforated Scordelis-Lo roof problem using a solid shell element.

MoFEM provides a convenient application programming interface allowing user to freely choose the approximation basis (e.g. Legendre or Jacobi polynomials) independently of the approximation space, and type and dimension of the field. A user can approximate scalar and vectorial fields on scalar basis functions, or vectorial and tensorial fields on vectorial bases. Moreover, MoFEM permits the construction of tensorial fields on tensorial bases, e.g. bubble basis of zero normal and divergence-free basis functions; see Gopalakrishnan & Guzmán (2012) for an example of such a space. A MoFEM user can also freely set the approximation order on each entity of an element separately, e.g. edge, face, volume, or define a field on the skeleton. Figure 6 presents a convergence study for the mixed formulation of a transport/heat conduction problem; the code snippet below outlines the definition of approximation space, basis and order for each field in this example.

```
// add fields of fluxes and values to the mesh
// define approximation space, basis and number of coefficients
mField.add_field(fluxes, HDIV, DEMKOWICZ_JACOBI_BASE, 1);
mField.add_field(values, L2, AINSWORTH_LEGENDRE_BASE, 1);
// get meshset consisting of all entities in the mesh
EntityHandle mesh_set = mField.get_moab().get_root_set();
// add mesh entities of different type to each field
// adding tetrahedra implies adding lower dimension entities
mField.add_ents_to_field_by_type(mesh_set, MBTET, fluxes);
```



```
mField.add_ents_to_field_by_type(mesh_set, MBTET, values);
// define approximation order for each field
// separately for each entity
mField.set_field_order(mesh_set, MBTET, fluxes, order+1);
mField.set_field_order(mesh_set, MBTRI, fluxes, order+1);
mField.set_field_order(mesh_set, MBTET, values, order);
```



**Figure 6:** A convergence study of  $h$ -adaptivity for the mixed formulation of the stationary transport/heat conduction problem (see inset of the figure for the geometry) with comparison of different polynomial orders, denoted as  $P_n$ - $P_m$ , where  $n$  is the order of approximation for the flux and  $m$  is the order for the field values (temperature or density). Note that the flux is approximated in a subspace of  $\mathbf{H}(\text{div})$  while the field values are in a subspace of  $L^2$ . For more details, see the “Mixed formulation and integration on skeleton” tutorial on (“MoFEM Web page,” 2020).

## Conclusions

MoFEM introduces a novel architecture of FEM software, designed to exploit advantages of emerging finite element technologies and to enable rapid implementation of numerical models for complex engineering problems involving multi-physics and multi-scale processes.

## Acknowledgements

MoFEM development has been supported by EDF Energy Nuclear Generation Ltd., EPSRC (grants EP/R008531/1 and EP/K026925/1), The Royal Academy of Engineering (grant no. RCSR1516\2\18) and Lord Kelvin Adam Smith programme at University of Glasgow.

## References

- Ainsworth, M., Andriamaro, G., & Davydov, O. (2011). Bernstein-Bézier finite elements of arbitrary order and optimal assembly procedures. *SIAM Journal on Scientific Computing*, 33(6), 3087–3109. doi:[10.1137/11082539x](https://doi.org/10.1137/11082539x)
- Ainsworth, M., & Coyle, J. (2003). Hierarchic finite element bases on unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 58(14), 2103–2130. doi:[10.1002/nme.847](https://doi.org/10.1002/nme.847)
- Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., et al. (2015). The fenics project version 1.5. *Archive of Numerical Software*, 3(100). doi:[10.11588/ans.2015.100.20553](https://doi.org/10.11588/ans.2015.100.20553)
- Arndt, D., Bangerth, W., Clevenger, T. C., Davydov, D., Fehling, M., Garcia-Sanchez, D., Harper, G., et al. (2019). The deal.II library, version 9.1. *Journal of Numerical Mathematics*. doi:[10.1515/jnma-2019-0064](https://doi.org/10.1515/jnma-2019-0064)
- Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., et al. (2019). *PETSc users manual* (No. ANL-95/11 - Revision 3.12). Argonne National Laboratory.
- Boost Web page. (2020). Retrieved from <https://www.boost.org>
- Fuentes, F., Keith, B., Demkowicz, L., & Nagaraj, S. (2015). Orientation embedded high order shape functions for the exact sequence elements of all shapes. *Computers & Mathematics with applications*, 70(4), 353–458. doi:[10.1016/j.camwa.2015.04.027](https://doi.org/10.1016/j.camwa.2015.04.027)
- Gopalakrishnan, J., & Guzmán, J. (2012). A second elasticity element using the matrix bubble. *IMA Journal of Numerical Analysis*, 32(1), 352–372. doi:[10.1093/imanum/drq047](https://doi.org/10.1093/imanum/drq047)
- Guo, B., & Babuška, I. (1986). The  $h$ - $p$  version of the finite element method. *Computational Mechanics*, 1(1), 21–41. doi:[10.1007/BF00298636](https://doi.org/10.1007/BF00298636)
- Hecht, F. (2012). New development in freefem++. *J. Numer. Math.*, 20(3-4), 251–265. doi:[10.1515/jnum-2012-0013](https://doi.org/10.1515/jnum-2012-0013)
- Kaczmarczyk, L., Ullah, Z., & Pearce, C. (2016). Prism solid-shell with heterogonous and hierarchical approximation basis. *UKACM Cardiff, UK*. doi:[10.5281/zenodo.789521](https://doi.org/10.5281/zenodo.789521)
- Kaczmarczyk, L., Ullah, Z., & Pearce, C. J. (2017). Energy consistent framework for continuously evolving 3D crack propagation. *Computer Methods in Applied Mechanics and Engineering*, 324, 54–73. doi:[10.1016/j.cma.2017.06.001](https://doi.org/10.1016/j.cma.2017.06.001)
- Kirk, B. S., Peterson, J. W., Stogner, R. H., & Carey, G. F. (2006). libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3–4), 237–254. doi:[10.1007/s00366-006-0049-3](https://doi.org/10.1007/s00366-006-0049-3)
- Kolev, T., & Dobrev, V. (2010). MFEM: Modular Finite Element Methods Library. doi:[10.11578/dc.20171025.1248](https://doi.org/10.11578/dc.20171025.1248)
- Lewandowski, K., Kaczmarczyk, Ł., Athanasiadis, I., Marshall, J. F., & Pearce, C. J. (2020). Numerical investigation into fracture resistance of bone following adaptation. Retrieved from <http://arxiv.org/abs/2001.00647>
- MOAB Web page. (2020). Retrieved from <https://press3.mcs.anl.gov/sigma/>
- MoFEM Web page. (2020). Retrieved from <http://mofem.eng.gla.ac.uk>
- Richardson, E. J. (2018). *A multiphysics finite element model of the wood cell wall* (PhD thesis). University of Glasgow. Retrieved from <http://theses.gla.ac.uk/9150/>
- Tautges, T. J., Meyers, R., Merkley, K., Stimpson, C., & Ernst, C. (2004). *MOAB: a mesh-oriented database* (SAND2004-1592). Sandia National Laboratories. doi:[10.2172/970174](https://doi.org/10.2172/970174)

Ullah, Z., Zhou, X.-Y., Kaczmarczyk, L., Archer, E., McIlhagger, A., & Harkin-Jones, E. (2019). A unified framework for the multi-scale computational homogenisation 3D-textile composites. *Composites Part B: Engineering*. doi:[10.1016/j.compositesb.2019.03.027](https://doi.org/10.1016/j.compositesb.2019.03.027)