MoFEM: An open source, parallel finite element library

3 May 2019

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` belongs to the class of open source finite element libraries, such as `Deal.II`, `MFEM`, `libMesh`, `FEniCS`, or `FreeFEM++` to name a few. These provide users with generic tools for solving PDEs and developers with methods to implement bespoke finite elements. `MoFEM` is developed to provide free finite element framework, incorporating modern approximation approaches and data structures, for engineers, students and academics. It is designed to solve bespoke engineering problems, it was primarily designed to solve crack propagation for structural integrity assessment of safety-critical structures (see Figure 1).

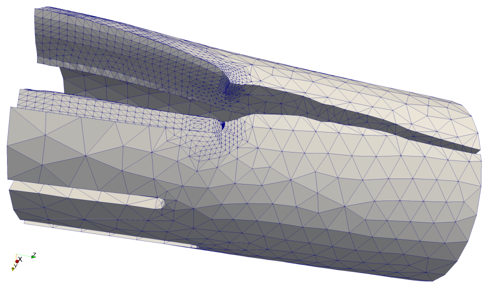

Figure 1: Brittle crack propagation.

The need for solutions to increasingly complex problems demands control over numerical errors; otherwise, we will be unable 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 [@guo1986hp], 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, the properties of hierarchical and heterogenous approximation basis, constructed Legendre [@ainsworth2003hierarchic] or Jacobi [@fuentes2015orientation] polynomials, can be exploited. Such bases enable to rise approximation locally and produce sparse and well-conditioned systems of equations. Moreover algebraic system constructed with hierarchical basis can be naturally restricted to lower dimensions and used as preconditioner, e.g. multigrid 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 this latter case, different approximation bases, such as the Bernstein-Bézier basis [@ainsworth2011bernstein], 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, $\mathbf{H}(\mathbf{curl})$, $\mathbf{H}(\mathbf{div})$ and L^2) are used.

`MoFEM` incorporates all the 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 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, integration and dense tensor algebra at the integration point level. `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) [@petsc-user-ref], components for handling mesh and topology from MOAB (Mesh-Oriented Database) [@tautges_moab:2004] and

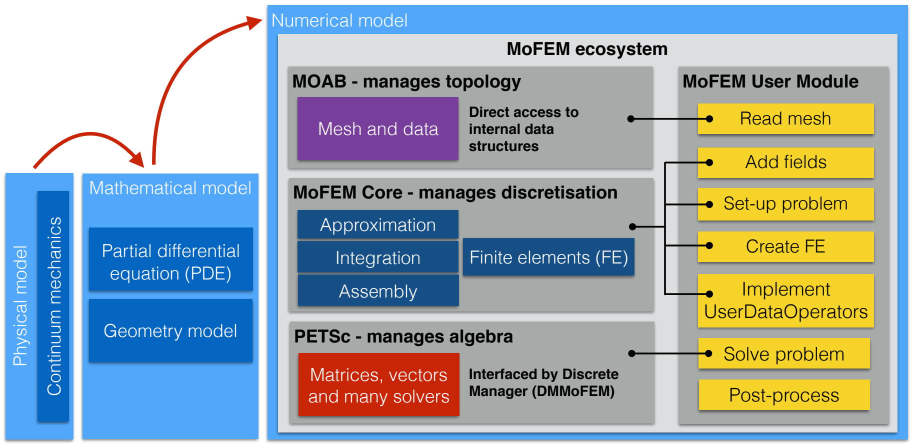

Figure 2: Basic design of `MoFEM`. Adopted from [@MoFEMWebPage].

data structures from Boost libraries [@boost-web-page]. An illustration of how these packages are utilised in `MoFEM` is shown in Figure 2. Finally, `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 significant number of users. Figure 3 shows different components that are employed in the ecosystem including popular pre- and post-processing software.

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, e.g. Lagrangian, Nédélec, 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, $\mathbf{H}(\mathbf{curl})$, $\mathbf{H}(\mathbf{div})$ and L^2) can be arbitrarily mixed in a finite element to create new capabilities for solving complex problems efficiently.

`MoFEM` data structures enable easy enrichment of approximation fields and modification of basis functions, for example, for resolving singularity at the crack front. Applying this 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 the benchmark in terms of how finite element codes are implemented,

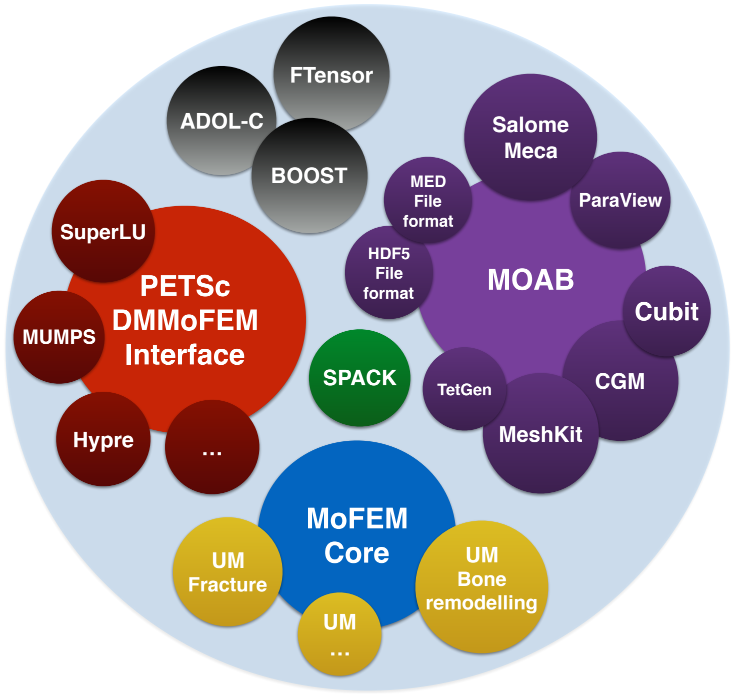

Figure 3: Ecosystem of `MoFEM`. Adopted from @MOABWebPage.

introducing a concept of *user-defined data operators* acting on fields that are associated with entities (vertices, edges, faces and volumes) rather on the finite element directly. 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 MoFEM installation instructions for more details.

Examples and Capabilities

`MoFEM` was initially created with the financial support of the Royal Academy of Engineering and EDF Energy to solve the problem of crack propagation [@kaczmarczyk2017energy]. Over time, the domain of applications expanded to include computational homogenisation [@ullah2019unified], bone remodelling and fracture [@lew2020numerical], modelling of the gel rheology [@richardson2018multiphysics] 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.

`MoFEM` is designed to provide efficient tools for solving a wide variety of user-defined problems. Figure 5 shows an example of error-driven *p*-adaptivity on hierarchical approximation basis with a multi-grid solver applied to the Scordelis-Lo perforated roof problem [@kaczmarczyk2016prism].

`MoFEM` provides a convenient application programming interface allowing a user to freely choose the approximation basis (e.g. Legrende or Jacobi polynomials) independently from 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 @gopalakrishnan2012second for an example of such a space. A `MoFEM` user can freely set the approximation order on each entity of an element separately, e.g. edge, face, volume, or define a field on the skeleton. In Figure 6, we present a convergence study for the mixed formulation of a transport/heat conduction problem. In the code snippet below, we outline defining approximation space, basis and order for each field in this example.

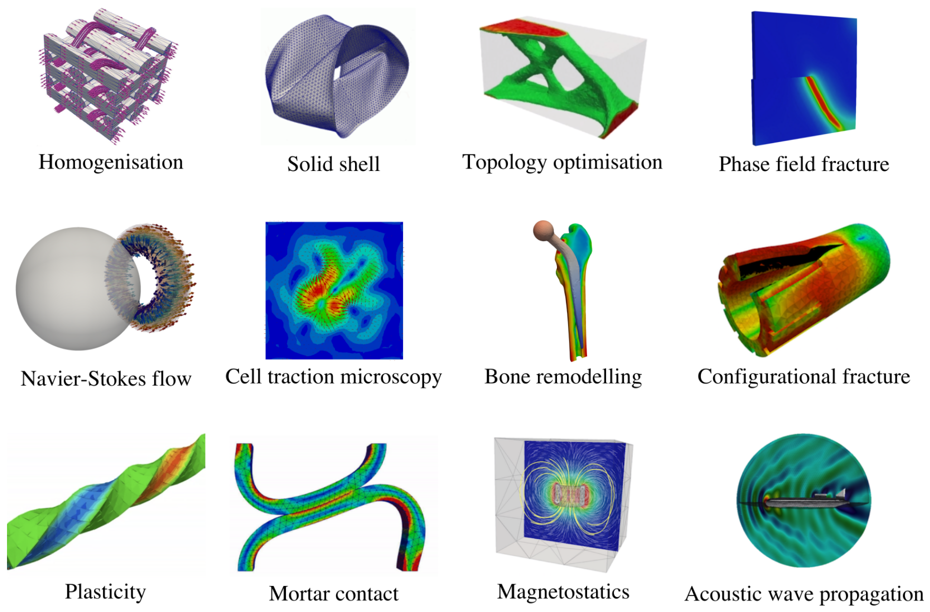

Figure 4: Examples of user modules implemented using `MoFEM`.

```
// 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);
```

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.

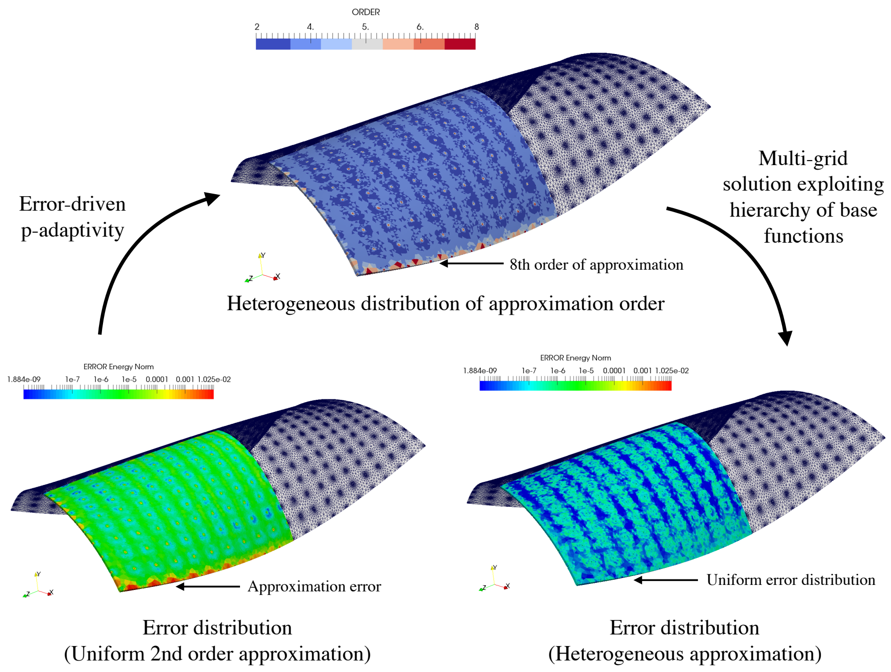
ORDER
2
4.
5.
6.
8
Error-driven p-adaptivity
Multi-grid solution exploiting hierarchy of base functions
8th order of approximation
Heterogeneous distribution of approximation order
ERROR Energy Norm
1.884e-09
1e-7
1e-6
1e-5
0.0001
0.001
1.025e-02
Approximation error
Uniform error distribution
Error distribution (Uniform 2nd order approximation)
Error distribution (Heterogeneous approximation)

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

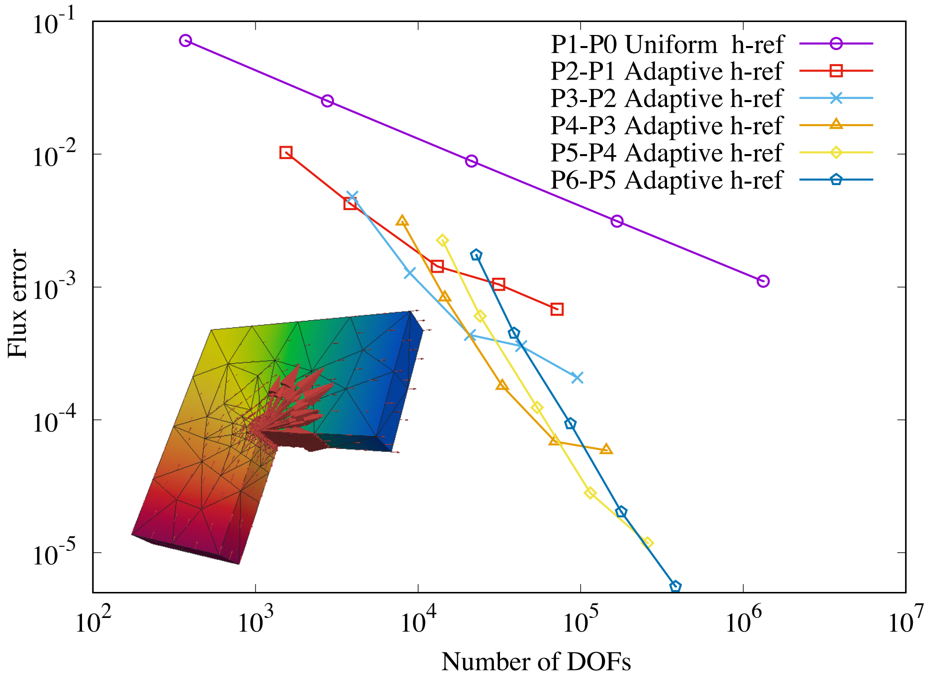

Figure 6: A convergence study of h-adaptivity for the mixed formulation of the stationary transport/heat conduction problem (see inset of the figure for the geometry), with the comparison of different polynomial orders, denoted as 'Pn-Pm', where n is order of approximation for the flux and m is the order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

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. RCSRF1516\2\18) and Lord Kelvin Adam Smith programme at University of Glasgow.

References