11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
July 20–25, 2014, Barcelona, Spain

FIREDRAKE, A TOOLCHAIN FOR PERFORMANCE-PORTABLE AUTOMATED FINITE ELEMENT SIMULATION.

David A. Ham[1,2], Gheorghe-Teodor Bercea[2], Colin J. Cotter[1], Paul H. J. Kelly[2], Michael Lange[3], Nicolas Loriant[2], Fabio Luporini[2], Andrew T. T. McRae[1], Lawrence Mitchell[2], and Florian Rathgeber[2]

[1] Department of Mathematics, Imperial College London, David.Ham@imperial.ac.uk, `http://firedrake.org`
[2] Department of Computing, Imperial College London
[3] Department of Earth Science and Engineering, Imperial College London

Key words: *Finite element method, Firedrake, UFL, FFC, code generation.*

Here we present Firedrake, a Python toolkit for the automatic generation of simulation software using the finite element method. Firedrake brings together UFL[1], a user-friendly high-level mathematical language for the finite element method, PyOP2[5], an abstraction for mesh-based parallel computing, FFC[4], an automated compiler for finite element operators, and PETSc[2], the state-of-the-art parallel solver library. Together, these deliver a multilayered abstraction of the process of numerical simulation.

This approach enables the composition of expertise across diverse domains: application scientists and engineers, numerical analysts, compiler experts, and parallel performance experts can apply their knowledge to the layers of the simulation task in which they specialise, without having to become experts in all of the other aspects. The result is a system which can solve a vast range of partial differential equations using diverse finite element spaces, exploiting hybrid MPI/OpenMP/vectorised parallelisation when executing on CPUs, while switching to CUDA or OpenCL when execution on a GPU is requested.

The finite element method is a particularly powerful technique for the calculation of approximate numerical solutions to partial differential equations which describe key processes in science and engineering. A key advantage of the finite element method over other approaches is that the numerical discretisation can be fully specified in succinct and expressive mathematical notation, completely independently of the low-level implementation of the method. This feature of the method was exploited by the FEniCS project[4] to create a finite element simulation environment in which succinct, mathematical expressions are automatically transformed into executing finite element simulations.

Firedrake adopts the same high-level UFL abstraction as FEniCS, but has a substantially different implementation pathway underneath. This demonstrates that the UFL abstraction is indeed implementation independent, and has the potential to become a widely adopted language for the representation of finite element problems. Indeed, Firedrake is sufficiently faithful to the FEniCS interface that Dolfin-adjoint, which enables the automatic execution of highly efficient adjoints to FEniCS simulations[3], works essentially unmodified with Firedrake.

However, Firedrake is not a FEniCS clone: by changing the underlying implementation, Firedrake can employ new performance otimisations in vectorisation and common subexpression elimination, it can support new element topologies such as wedges, which are critical in applications such as weather and climate, and it supports parallelisation features, such as GPU support and support for DG methods in MPI parallel, which are unavailable in FEniCS.

In this presentation, we will lay out the core structure and abstractions of the Firedrake system, and demonstrate its capacity to produce performant implementations of challenging finite element problems. We will demonstrate the performance portability of the system and, presentation time permitting, its capacity to solve adjoint as well as forward problems.

REFERENCES

[1] Martin Sandve Alnæs, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. Unified Form Language: A domain-specific language for weak formulations of partial differential equations. *to appear in ACM Transactions on Mathematical Software*, abs/1211.4047, 2013.

[2] S Balay, J Brown, K Buschelman, V Eijkhout, W Gropp, D Kaushik, M Knepley, L Curfman McInnes, B Smith, and H Zhang. Petsc users manual revision 3.2. 2011.

[3] P. E. Farrell, D. A. Ham, S. W. Funke, and M. E. Rognes. Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35:C359–393, 2013.

[4] Anders Logg, Kent-Andre Mardal, and Garth N. Wells, editors. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[5] Florian Rathgeber, Graham R. Markall, Lawrence Mitchell, Nicolas Loriant, David A. Ham, Carlo Bertolli, and Paul H.J. Kelly. Pyop2: A high-level framework for performance-portable simulations on unstructured meshes. In *High Performance Computing, Networking Storage and Analysis, SC Companion:*, pages 1116–1123, Los Alamitos, CA, USA, 2012. IEEE Computer Society.