

# Findings for Automatic Parallel Finite Elements

September 23, 2013

This project, which finished in 2009, focused on the development and application of automated finite element software and led to eight archival publications (many highly cited) and the birth of the FEniCS project (<http://fenicsproject.org>). Today, FEniCS is widely available, with binary installation available for Windows, Mac, and several Linux varieties with supported source installation on many other platforms

## 1 FIAT

The initial impetus for the grant was the FIAT project [1], developed by the PI to provide a declarative framework for describing and generating arbitrary-order simplicial basis functions. This tool was the first of its kind and enabled general implementation of many powerful but complicated finite element spaces. Direct FIAT-related work on this project includes [3], which has 20 citations on Google Scholar, in which we redevelop FIAT internals to utilize level 3 BLAS to make basis construction efficient and [4], where we develop alternate recurrence relations for prime bases that admit general automatic differentiation. This sets up FIAT to support elements like Hermite and Argyris with derivative degrees of freedom at vertices that are singular in the collapsed-coordinate mapping. Perhaps future work will enable wider use of these nonstandard elements.

## 2 FFC and FErari

Given the ability to generate basis functions, we studied the task of generating code for finite element bilinear forms. The first such work in this was the `ffc` project with Anders Logg [6], which now shows 78 Google Scholar citations. `ffc` provides a domain-specific language for variational forms and compiles C++ code that links against the DOLFIN library. This code is highly efficient for constant coefficient forms, far surpassing standard quadrature-based implementations. More efficient compilation strategies were developed in [2], and we generalized `ffc` to support  $H(\text{div})$  and  $H(\text{curl})$  discretizations with Marie Rognes in [11], giving the first general-purpose implementation of arbitrary-order Raviart-Thomas-Nédélec elements. For example, for solving the mixed form of Poisson's equation, we now can use the code:

```
# Define function spaces and mixed (product) space
BDM = FunctionSpace(mesh, "BDM", 1)
DG = FunctionSpace(mesh, "DG", 0)
W = BDM * DG

# Define trial and test functions
(sigma, u) = TrialFunctions(W)
(tau, v) = TestFunctions(W)
```

```
# Define source function
f = Expression('10*exp(-(pow(x[0] - 0.5, 2) + pow(x[1] - 0.5, 2)) / 0.02)')

# Define variational form
a = (dot(sigma, tau) + div(tau)*u + div(sigma)*v)*dx
L = - f*v*dx
```

to create the Brezzi-Douglas-Marini[] finite element coupled with piecewise constants (first three commands) using FIAT, create test and trial functions and forcing function, and specify the bilinear form and right-hand side functional (last two lines). A simple call to `solve` then completes the simulation. Similarly, Nédélec elements are supported so that curl-conforming discretizations may also be used. Ølgaard and Wells [10] also extended `ffc` to provide a general quadrature-based mode that greatly improved the power of `ffc` to handle more complex variational forms.

While `ffc` generates efficient code, we also studied the problem of further optimizing by reducing instruction counts. In the FErari project, which grew out of [5], we produced a series of papers [8, 9, 7] studying discrete structures that lead to reduced-arithmetic algorithms. In many cases, we could reduce the number of operations required to form an element matrix by a factor of 3 or more. Many of the FErari publications exceed 20-30 Google Scholar citations.

## References

- [1] R. C. Kirby. FIAT: A new paradigm for computing finite element basis functions. *ACM Trans. Math. Software*, 30:502–516, 2004.
- [2] R. C. Kirby and A. Logg. Efficient compilation of a class of variational forms. *ACM Transactions on Mathematical Software*, 33(3), 2007.
- [3] Robert C. Kirby. Optimizing FIAT with Level 3 BLAS. *to appear in ACM Transactions on Mathematical Software*, 2006.
- [4] Robert C Kirby. Singularity-free evaluation of collapsed-coordinate orthogonal polynomials. *ACM Transactions on Mathematical Software (TOMS)*, 37(1):5, 2010.
- [5] Robert C. Kirby, Matthew G. Knepley, Anders Logg, and L. Ridgway Scott. Optimizing the evaluation of finite element matrices. *SIAM J. Sci. Comput.*, 27(3):741–758 (electronic), 2005.
- [6] Robert C. Kirby and Anders Logg. A compiler for variational forms. *ACM Transactions on Mathematical Software*, 32(3):417–444, 2006.
- [7] Robert C. Kirby and Anders Logg. Benchmarking domain-specific compiler optimizations for variational forms. *ACM Transactions on Mathematical Software*, 35(2):10, July 2008. Article 10, 18 pages.
- [8] Robert C. Kirby, Anders Logg, L. Ridgway Scott, and Andy R. Terrel. Topological optimization of the evaluation of finite element matrices. *SIAM J. Sci. Comput.*, 28(1):224–240 (electronic), 2006.
- [9] Robert C. Kirby and L. Ridgway Scott. Geometric optimization of the evaluation of finite element matrices. *SIAM J. Sci. Comput.*, 29(2):827–841 (electronic), 2007.

- [10] K. B. Ølgaard and G. N. Wells. Optimisations for quadrature representations of finite element tensors through automated code generation. *ACM Transactions on Mathematical Software*, 37(1), 2010.
- [11] Marie E. Rognes, Robert C. Kirby, and Anders Logg. Efficient assembly of  $H(\text{div})$  and  $H(\text{curl})$  conforming finite elements. *SIAM Journal on Scientific Computing*, 31(6):4130–4151, 2009.