Geosci. Model Dev. Discuss., 6, C1642–C1646, 2013
www.geosci-model-dev-discuss.net/6/C1642/2013/

Geoscientific Model Development Discussions
Open Access

Interactive comment on "Automating the solution of PDEs on the sphere and other manifolds in FEniCS 1.2" *by* M. E. Rognes et al.

M. E. Rognes et al.

david.ham@imperial.ac.uk

Received and published: 10 October 2013

We would like to thank the reviewer for a particularly thorough analysis of our manuscript. We have addressed the individual points raised below, indicating the revisions we have made to the manuscript as appropriate.

1. In writing this, we were referring to our experience in developing, for example, mantle convection codes in DOLFIN in comparison with the similar development process in the Fluidity framework. The subject of this paper is the numerical solution of PDEs, we do not consider parametrisations. As the reviewer correctly points out, it is much harder to abstract these to a higher level, and we do not address this subject in this paper. In more carefully reviewing the cases we have

Full Screen / Esc
Printer-friendly Version
Interactive Discussion
Discussion Paper

studied, we have definitely replaced tens of thousands of lines of Fortran with hundreds of lines of UFL and Python. Since we have not, at this stage, completely replaced a multi-hundred thousand line code with FEniCS, we have removed this claim from the revised manuscript.

2. We appreciate that, for a reader very familiar with the finite element method, the mathematical formulation is quite expansive. If we were to publish this work in a purely finite element forum, section 2 would be presented in a much more abbreviated form. However by publishing in GMD we are seeking to reach out to readers whose primary expertise is in the physical systems being studied or in the properties of the equations concerned. We therefore feel that in the context of GMD, a more detailed exposition of the mathematical formulation is appropriate. We note in this regard that the first reviewer wrote approvingly of our description of the mathematics.

   We then turn to the possibility, raised by the reviewer, of merging sections 2 and 3. We are not keen to do this as we feel that it is important to maintain the distinction between the mathematical formulation, which is a straight theoretical exposition, and its implementation in code. We feel that it is a frequent failing in computational science publications that the mixing of theoretical derivations and implementation details makes it unclear to the reader which is which. By maintaining the clear distinction between sections 2 and 3 we seek to avoid this.

3. We neglected to mention parallelisation in the original manuscript since the work we did was orthogonal to the parallel execution of DOLFIN. We will add another section explaining that the parallel execution of DOLFIN is unaffected by the changes we made and give a reference for the latest parallel performance work which has been undertaken on DOLFIN.

   The essential feature of this work was the modifications required to the local assembly operations generated by FFC in order to support immersed manifolds.

The parallel performance of the code is governed by the distribution of the global assembly operation, and by the parallelisation of the matrix solves. Were we to embark on an extensive parallel scaling exercise, we would essentially be testing code which was not modified in the course of this work. In the case of the matrix solvers, which often dominate performance, these are not even directly a part of FEniCS: instead this task is outsourced to PETSc or (at the users choice) another external library.

4. In this paper we used two different approaches to obtaining finite element approximations to vector fields on manifolds, one of which being the constraint approach of (Côté, 1988), and the other being *via* the Piola transformation. Neither of these is the same as (Bernard et al, 2009) which we cited in the introduction. We will add some text into the end of Section 2.3 that describes the latter approach, and clarifies this distinction.

5. The L_2 norm in Equation (56) is defined in Equation (47) and is the usual definition of the L_2 norm. We will modify the captions of figures 10 and 12 to include the errors in energy and mass conservation. These are 1.4×10^{-15} and 1.7×10^{-14} respectively.

6. We will add some further text explaining why we did not modify the curl/rot operator (they are the same thing in UFL), namely because they require the choice of a normal vector, which would require modifications to the element kernel interface in UFC. Furthermore, as shown in the examples, sometimes one wishes to use the actual normal to the mesh, and sometimes one wishes to use a continuous approximation to the normal on the manifold that is being approximated. In section 5.5, it is crucial to use the actual mesh normal to obtain the required mimetic properties. We are not aware of any other operators that the user may require to build equations, or at least there are not any other existing UFL operators that we did not extend to manifolds. It is in any case important to realise that all dif-

ferential operators in UFL such as “div”, “curl” etc., immediately execute as soon as the expression is instantiated, and they are replaced by UFL expressions in terms of components of the gradient of the field (or its components), so if the user wishes they can create their own operators that output UFL expressions.

7. We will add a definition of A, the local element tensor, to the caption of Figure 6. The full code generation process in FFC is already documented in Logg et al. (2012d), an exposition which is too substantive to reproduce in this paper, so we will add a citation referring the reader to that source. In response to the reviewer’s particular question, G is the geometry tensor, which is an intermediate value in the computation of the element integral. Its presence is an implementation detail.

8. The labels will be made larger in size and we will add labels.

9. We will replace Figure 15 with a figure showing the velocity, height and vorticity fields at day 15.

10. Just in time compilation refers to the process of generating C++ code implementing form assembly at runtime, and compiling it “just in time” to be used. We will include a definition of the term and a reference to the expanded discussion at Logg et al. (2012d) in the revised manuscript.

11. Were it supported, the ability to specify functions over cells of different dimension could, for example, be used to impose a Lagrange multiplier over the surface of a mesh while solving an equation over the mesh as a whole. We will add an explanation to this effect to the revised manuscript.

12. This sentence will be rephrased.

13. The duplicate “in” will be removed.

14. This sentence will be split into two to aid clarity, however it is otherwise correct.

References

Logg, A., Ølgaard, K. B., Rognes, M. E., and Wells, G. N.: FFC: the FEniCS Form Compiler, in: Automated Solution of Differential Equations by the Finite Element Method, vol. 84 of Lecture Notes in Computational Science and Engineering, edited by: Logg, A., Mardal, K.-A., and Wells, G. N., chap. 11, Springer, 2012d.

Interactive comment on Geosci. Model Dev. Discuss., 6, 3557, 2013.