

FEniCSX: A sustainable future for the FEniCS Project

Michal Habera, Jack S. Hale, Chris Richardson, Johannes Ring,
Marie E. Rognes, Nathan Sime, Garth N. Wells.

MS42 Improving Productivity and Sustainability for Parallel Computing Software
SIAM Parallel Processing 2020, Seattle WA, USA.

Outline

1. Brief history and impact of FEniCS project
2. Sustainability through *formalised governance*
3. Sustainability through *automated workflows*
4. Sustainability through innovation: *FEniCSX*

What is the FEniCS Project?

- From the FEniCS project plan, October 2003:

2 The vision of FEniCS

The vision of **FEniCS** is to set a new standard in CMM, which can be described as the *Automation of CMM*, towards the goals of *generality*, *efficiency*, and *simplicity*, concerning *mathematical methodology*, *implementation*, and *application*.

The FEniCS Project: A brief history



- An Open Source software for the automated solution of partial differential equations using the finite element method.
- Term *FEniCS Project* first used around 2002.
- First *FEniCS Project conference* in 2005 at Toyota Technological Institute.
 - Ridgway Scott, Matt Knepley, Rob Kirby, Hans Petter Langtangen, Andy Terrel, Garth Wells, Johan Hoffmann, Anders Logg, Johan Jansson...
- Key part of community building: (nearly) annual conferences.
 - 2006 Delft, 2008 Baton Rouge, 2009 Oslo, 2010 Stockholm, 2011 Lubbock, 2012 Oslo, 2013 Cambridge, 2014 Paris, 2015 London, 2016 Oslo, **2017 Luxembourg**, 2018 Oxford, 2019 Washington DC, **2020 Cambridge**.



FEniCS 2020

FEniCS 2020

**FEniCS 2020 at University of Cambridge, 24-26
June 2020.**

Key tech breakthroughs: Domain specific languages and code generation

- Finite Element Automatic Tabulator (FIAT) 2002 -
- Unified Form Language (UFL), 2008 -
- FEniCS Form Compiler (FFC), 2004 -

UFL, FE symbolic language

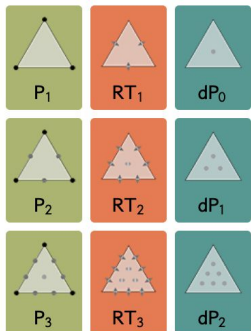
$\text{inner}(\text{grad}(\mathbf{u}), \text{grad}(\mathbf{v})) * dx$

FFC, UFL→C compiler

```
void tabulate_tensor_poisson_cell_integral_43165b9e21c850b7e46d14f843b
...
const ufc_scalar_t* c,
const double* restrict coordinate,
const int* unused_local_index,
const int* cell_orientation)
{
    // Precomputed values of basis functions and precomputations
    // FE* dimensions: [entities][points][dofs]
    // PI* dimensions: [entities][dofs][dofs] or [entities][dofs]
    // PM* dimensions: [entities][dofs][dofs]
    alignas(32) static const ufc_scalar_t FE3_C0_D01_Q1[1][1][2] = {
    // Unstructured piecewise computations
    const double J_c0 = coordinate_dofs[0] * FE3_C0_D01_Q1[0][0][0] +
    const double J_c3 = coordinate_dofs[1] * FE3_C0_D01_Q1[0][0][0] +
    const double J_c1 = coordinate_dofs[0] * FE3_C0_D01_Q1[0][0][0] +
    const double J_c2 = coordinate_dofs[1] * FE3_C0_D01_Q1[0][0][0] +
```

FIAT, FE oracle

\mathbf{u}, \mathbf{v} in (Lagrange, 1st order)



Poisson equation in UFL

```
geometry = VectorElement("Lagrange", triangle, 2)
mesh = Mesh(geometry)
element = FiniteElement("Lagrange", triangle, 2)
V = FunctionSpace(mesh, element)

u, v = TrialFunction(V), TestFunction(V)
f = Coefficient(V)
a = inner(grad(u), grad(v)) * dx
L = inner(f, v) * dx
```


Applications 1



dolfin-adjoint

FEniCS-Shells

A FEniCS Project-based library for simulating thin structures.

hiPPYlib - Inverse Problem PYthon library

build passing

docs passing

JOSS 10.21105/joss.00940

DOI 10.5281/zenodo.596931

FENaPack - FEniCS Navier-Stokes preconditioning package

[1] Farrell, P.E. et al. 2013. Automated Derivation of the Adjoint of High-Level Transient Finite Element Programs. SIAM Journal on Scientific Computing. 35, 4 (Jan. 2013), C369–C393. DOI:<https://doi.org/10.1137/120873558>.

[2] Villa, U. et al. 2018. hiPPYlib: An Extensible Software Framework for Large-Scale Inverse Problems. Journal of Open Source Software. 3, 30 (Oct. 2018), 940. DOI:<https://doi.org/10.21105/joss.00940>.

[3] Hale, J.S. et al. 2018. Simple and extensible plate and shell finite element models through automatic code generation tools. Computers & Structures. 209, (Oct. 2018), 163–181. DOI:<https://doi.org/10.1016/j.compstruc.2018.08.001>.

Applications 2



multiphenics - easy prototyping of multiphysics problems in FEniCS



RBniCS - reduced order modelling in FEniCS

tIGAr

A Python library for isogeometric analysis



[1] Ballarin, F. et al. 2017, <https://mathlab.sissa.it/multiphenics>

[2] Hesthaven, J.S. et al. 2015. Reduced Basis Methods. SpringerBriefs in Mathematics. Springer International Publishing. 27–43.

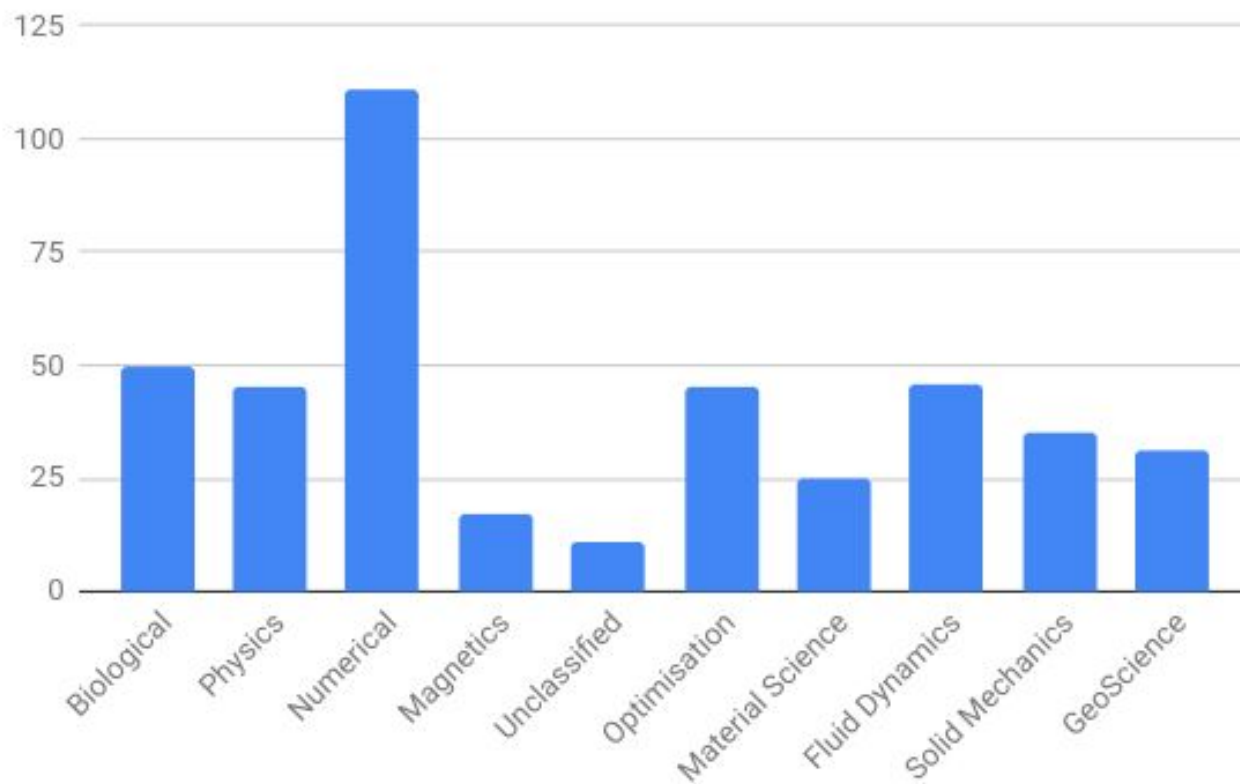
[3] Kamensky, D. and Bazilevs, Y. 2019. tIGAr: Automating isogeometric analysis with FEniCS. Computer Methods in Applied Mechanics and Engineering. 344, (Feb. 2019), 477–498. DOI:<https://doi.org/10.1016/j.cma.2018.10.002>.

Popular and broadly used

| | | |
|---|------|------|
| Automated solution of differential equations by the finite element method: The FEniCS book A Logg, KA Mardal, GN Wells Lecture Notes in Computational Science and Engineering 84 | 1893 | 2012 |
| The FEniCS project version 1.5 M Alnæs, J Blechta, J Hake, A Johansson, B Kehlet, A Logg, ... Archive of Numerical Software 3 (100) | 732 | 2015 |
| DOLFIN: Automated finite element computing A Logg, GN Wells ACM Transactions on Mathematical Software (TOMS) 37 (2), 1-28 | 536 | 2010 |
| Unified form language: A domain-specific language for weak formulations of partial differential equations MS Alnæs, A Logg, KB Ølgaard, ME Rognes, GN Wells ACM Transactions on Mathematical Software (TOMS) 40 (2), 1-37 | 231 | 2014 |

$\Sigma \sim 3400$

Publications citing FEniCS Book by Subject Area



Sustainability
through
*formalised
governance*

Joining NumFOCUS (2016)



- NumFocus (<https://numfocus.org>)
 - “Better tools to build a better world”
 - “From Netflix to NASA, researchers use NumFOCUS’ open source tools to solve the most challenging problems”
- Advantages:
 - Prestige/stamp-of-quality (NumPy, Julia, Stan, pandas, AstroPy, matplotlib...).
 - Access to commercial sponsors (Microsoft, Facebook, IBM...).
 - Access to Google Summer of Code programme under NumFOCUS umbrella.
 - NumFOCUS annual conferences.
 - All legal and financial aspects of running foundation offloaded.

Who has the right to make key technical decisions?

- Possible models:
 - Benevolent dictator for life (BDFL) e.g. Larry Wall (Perl), Linus Torvalds (Linux).
 - Foundation model, e.g. Apache Software Foundation.
 - **Steering Council within larger foundation, NumFOCUS .**
- Community led:
 - All users, developers and contributors of the FEniCS Project
- Steering Council:
 - Members of the community who have made significant contributions over a sustained period of time.
- Advisory board (Lois Curfman McInnes et al.):
 - Ensure the long-term well-being of the project. Oversight.

Money

- NumFOCUS gives us the legal infrastructure to:
 - accept donations.
 - spend money.
 - organise payments for events, e.g. conferences.
- in a transparent way, independent of any of the institutional partners (largely Universities).
- NumFOCUS has:
 - small development grants.
 - support travel grants to FEniCS Conference.

Google Summer of Code (GSoC)

- We participate in GSoC through NumFOCUS.
 - GSoC is highly competitive for students and **organisations**.
 - Our project proposals have **always been accepted** by Google.
 - Possible without NumFOCUS? Perhaps not.



Google Summer of Code

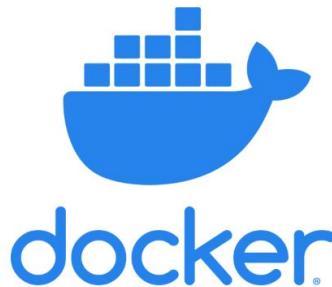
| 2017 | 2018 | 2019 |
|--|--|---|
| Michal Habera higher order XDMF and visualization | Fabian Loschner SIMD and vectorisation | Igor Baratta KaHIP mesh partitioning |
| Ivan Yashchuk quadrilateral and hexahedral meshes | Igor Baratta complex numbers support | Abhinav Gupta mesh pipeline from gmsh |

stays involved in
FEniCS
development

Sustainability
through
automated
workflows

FEniCS Project has been through a lot of workflow tools...

- Version control (launchpad, Bitbucket, **github**)
- Unit testing (Python unittest, **pytest**)
- Automated testing (launchpad, Jenkins, Atlassian Bamboo, **CircleCI**)
- Platforms for executing tests (physical machines, virtual machines, **Docker containers**)



Criteria for choosing workflow tools

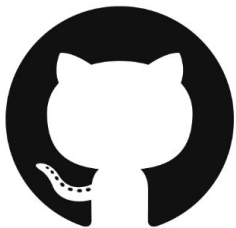
1. Follow the herd (Bitbucket to github).
2. Minimise money spent (physical machines to Docker containers).
3. Be as lazy as possible (fragile custom infrastructure to infrastructure as code).
4. When it breaks, make it someone else's problem (Atlassian Bamboo to CircleCI).



Contributor opens pull request on github.
Code is reviewed by project maintainer.



CircleCI runs full test suite inside Docker
container.



If

- tests pass AND
 - branch up to date with master branch
- pull request *can* be merged by project
maintainer.



**Sustainability
through
innovation:
FEniCSX**

“Every computing infrastructure project that initially meets one need well will eventually expand in scope to only meet several needs poorly.”

William Benton

active

bug maintenance

2003

2008

2018

2020

Dolfin



Dolfinx



FFC



FFCx



UFL



FIAT



Criticism of FEniCS libraries

- works well if remaining within the supported abstractions
- difficult to extend, especially from Python interface
- difficult to experiment with new methods at low level
- slow development progress
- many ways of killing the performance in Python

FEniCSX as an incremental rewrite

- keep most of high-level abstractions
- allow manual implementation of all operations
- less OOP design
- more explicit behaviour

<https://fenicsproject.org/fenics-project-roadmap-2019/>

- Compact, modular core designed to be extensible and to support custom additions
- Smaller line count, faster build, **faster CI tests**
- Follow established standards and conventions wherever possible (design, packaging, testing, etc)
- **Properly separated C++ and Python interfaces**
- Simplified software engineering
- **Distributed memory parallel design throughout**
- **Improved documentation**
- Faster just-in-time compilation
- Support for modern Python JIT tools, e.g. Numba
- Simple implementation of fast user ‘kernels’ from Python
- Provide just one way to perform an operation wherever possible

New tools became available...

Numba

Python and NumPy LLVM based JIT compiler



Eigen

C++ template library for linear algebra



C++11, C++14, C++17



| Package (without deps) | C++/C lines | Python lines |
|------------------------|---------------|--------------|
| Dolfin | 90 000 | 22 000 |
| Dolfinx | 46 000 | 9 000 |
| Deal.ii | 830 000 | |
| PETSc | 580 000 | |
| Firedrake | | 37 000 |
| Eigen | 125 000 | |

generated using David A. Wheeler's 'SLOCCount'.

Example: Static condensation (1)

```
@numba.cfunc(c_signature, nopython=True)
```

```
def knl(A_, w_, c_, coords_, e, c):
```

```
    A = numba.carray(A_, (Usize, Usize))
```

```
    A00 = numpy.zeros((Ssize, Ssize))
```

```
    kernel00(ffi.from_buffer(A00), ...)
```

```
    A01 = numpy.zeros((Ssize, Usize))
```

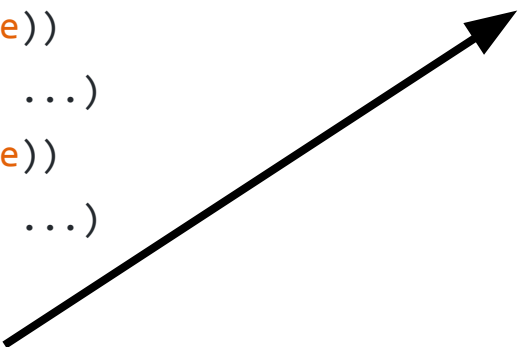
```
    kernel01(ffi.from_buffer(A01), ...)
```

```
    A10 = numpy.zeros((Usize, Ssize))
```

```
    kernel10(ffi.from_buffer(A10), ...)
```

```
    # A = - A10 * A00-1 * A01
```

```
    A[:, :] = - A10 @ numpy.linalg.solve(A00, A01)
```



Any numpy supported
operations, many
implemented with BLAS,
LAPACK

Example: Static condensation (2)

```
a_cond = Form([U, U])  
a_cond.set_tabulate_tensor(..., kn1.address)
```

```
A_cond = assemble_matrix(a_cond)  
A_cond.assemble()
```

Conclusions

- significantly reduced codebase
- faster and more scalable
- faster and simpler CI/CD
- highly customizable at all levels
 - ... and also performant thanks to Numba (cffi) in Python
- new features: complex numbers, block assembly, ...

<https://github.com/FEniCS/dolfinx>

<https://github.com/FEniCS/ffc>