The FEniCS Project
Automated Solution of PDEs

Presented by Anders Logg*
Simula Research Laboratory, Oslo

2012–12–05

What is FEniCS?

FEniCS is an automated programming environment for differential equations

- C++/Python library
- Initiated 2003 in Chicago
- 1000–2000 monthly downloads
- Part of Debian and Ubuntu
- Licensed under the GNU LGPL

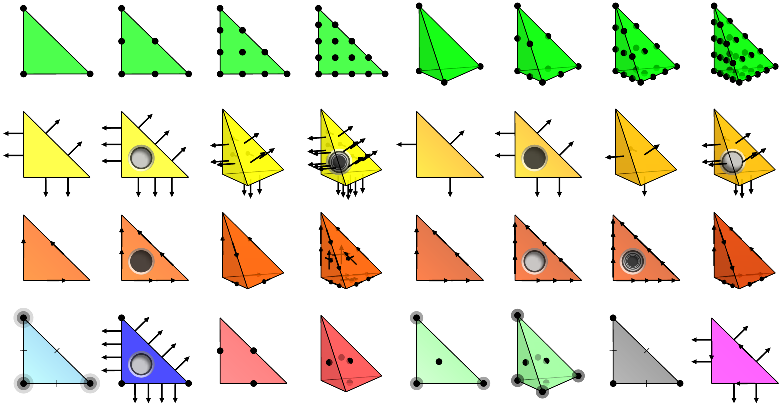
`http://fenicsproject.org/`

Collaborators

Simula Research Laboratory, University of Cambridge, University of Chicago, Texas Tech University, University of Texas at Austin, KTH Royal Institute of Technology, . . .

FEniCS is automated FEM

- Automated generation of basis functions
- Automated evaluation of variational forms
- Automated finite element assembly
- Automated adaptive error control



FEniCS is automated scientific computing

Input

- $A(u) = f$
- $\epsilon > 0$

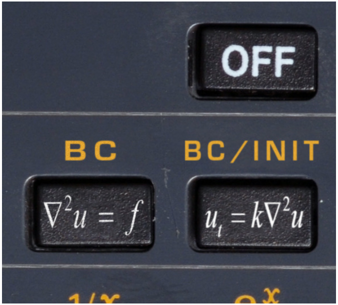
Output

- Approximate solution:

$$u_h \approx u$$

- Guaranteed accuracy:

$$\|u - u_h\| \leq \epsilon$$



How to use FEniCS?

Installation

- Official packages for Debian and Ubuntu
- Drag and drop installation on Mac OS X
- Binary installer for Windows
- Automated installation from source

Hello World in FEniCS: problem formulation

Poisson's equation

$$
\begin{aligned}
-\Delta u &= f \quad \text{in } \Omega \\
u &= 0 \quad \text{on } \partial\Omega
\end{aligned}
$$

Finite element formulation

Find $u \in V$ such that

$$
\underbrace{\int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x}_{a(u,v)} = \underbrace{\int_\Omega f\, v \,\mathrm{d}x}_{L(v)} \quad \forall\, v \in V
$$

Hello World in FEniCS: implementation

```
from dolfin import *

mesh = UnitSquare(32, 32)

V = FunctionSpace(mesh, "Lagrange", 1)
u = TrialFunction(V)
v = TestFunction(V)
f = Expression("x[0]*x[1]")

a = dot(grad(u), grad(v))*dx
L = f*v*dx

bc = DirichletBC(V, 0.0, DomainBoundary())

u = Function(V)
solve(a == L, u, bc)
plot(u)
```

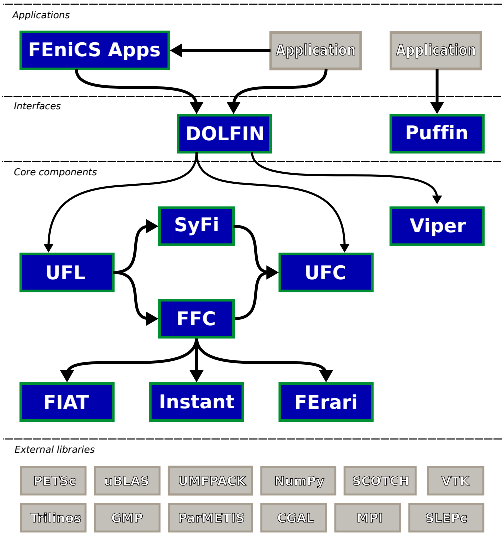
FEniCS under the hood

Basic API

- `Mesh`, `Vertex`, `Edge`, `Face`, `Facet`, `Cell`
- `FiniteElement`, `FunctionSpace`
- `TrialFunction`, `TestFunction`, `Function`
- `grad()`, `curl()`, `div()`, ...
- `Matrix`, `Vector`, `KrylovSolver`, `LUSolver`
- `assemble()`, `solve()`, `plot()`

- Python interface generated semi-automatically by SWIG
- C++ and Python interfaces almost identical

FEniCS software components



Quality assurance by continuous testing

fenics-buildbot	lucid-amd64	maverick-i386	mac-osx	linux64-exp
	9 (9) / 9	9 (9) / 9	9 (9) / 9	9 (9) / 9
ferari	Success	Success	Success	Success
fiat	Success	Success	Success	Success
ufc	Success	Success	Success	Success
instant	Success	Success	Success	Success
ufl	Success	Success	Success	Success
ffc	Success	Success	Success	building
viper	Success	Success	Success	Success
dolfin	Success	Success	Success	Success
syfi	Success	Success	Success	Success
	9 (9) / 9	9 (9) / 9	9 (9) / 9	9 (9) / 9

Automated error control

Automated goal-oriented error control

Input

- Variational problem: Find $u \in V$: $a(u, v) = L(v) \quad \forall\, v \in V$
- Quantity of interest: $\mathcal{M} : V \to \mathbb{R}$
- Tolerance: $\epsilon > 0$

Objective

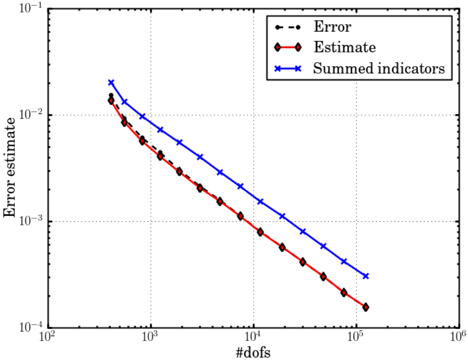
Find $V_h \subset V$ such that $|\mathcal{M}(u) - \mathcal{M}(u_h)| < \epsilon$ where

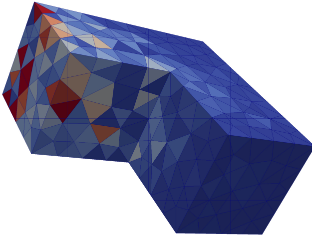
$$a(u_h, v) = L(v) \quad \forall\, v \in V_h$$

Automated in FEniCS (for linear and nonlinear PDE)

```
solve(a == L, u, M=M, tol=1e-3)
```

Poisson's equation

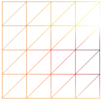


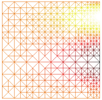


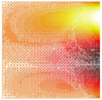
$$a(u, v) = \langle \nabla u, \nabla v \rangle$$

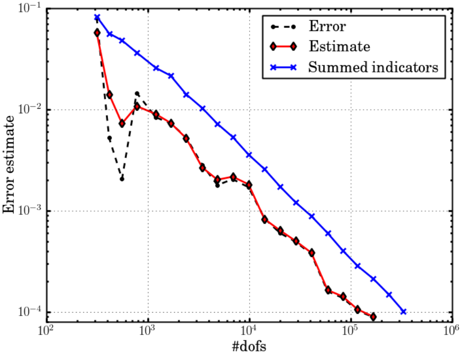
$$\mathcal{M}(u) = \int_{\Gamma} u \, \mathrm{d}s, \quad \Gamma \subset \partial\Omega$$

A three-field mixed elasticity formulation





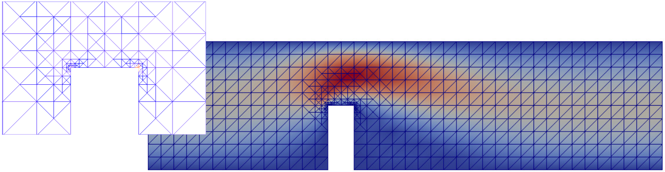




$$a((\sigma, u, \gamma), (\tau, v, \eta)) = \langle A\sigma, \tau\rangle + \langle u, \operatorname{div} \tau\rangle + \langle \operatorname{div} \sigma, v\rangle + \langle \gamma, \tau\rangle + \langle \sigma, \eta\rangle$$

$$\mathcal{M}((\sigma, u, \eta)) = \int_\Gamma g\, \sigma \cdot n \cdot t \,\mathrm{d}s$$

Incompressible Navier–Stokes



Outflux $\approx 0.4087 \pm 10^{-4}$

Uniform

1.000.000 dofs, N hours

Adaptive

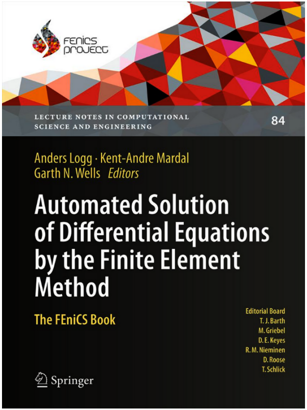
5.200 dofs, 127 seconds

```
from dolfin import *

class Noslip(SubDomain): ...

mesh = Mesh("channel-with-flap.xml.gz")
V = VectorFunctionSpace(mesh, "CG", 2)
Q = FunctionSpace(mesh, "CG", 1)
W = V*Q

# Define test functions and unknown(s)
(v, q) = TestFunctions(W)
w = Function(W)
(u, p) = split(w)

# Define (non-linear) form
n = FacetNormal(mesh)
p0 = Expression("(4.0 - x[0])/4.0")
F = (0.02*inner(grad(u), grad(v)) + inner(grad(u)*u), v)*dx
    - p*div(v) + div(u)*q + dot(v, n)*p0*ds

# Define goal functional
M = u[0]*ds(0)

# Compute solution
tol = 1e-4
solve(F == 0, w, bcs, M, tol)
```

Rognes, Logg, *Automated Goal-Oriented Error Control I* (2010)

Closing remarks

Ongoing activities



- Parallelization (2009)
- Automated error control (2010)
- Debian/Ubuntu (2010)
- Documentation (2011)
- FEniCS 1.0 (2011)
- The FEniCS Book (2012)

- **FEniCS'13**
  Cambridge March 2013
- Visualization, mesh generation
- Parallel AMR
- Hybrid MPI/OpenMP
- Overlapping/intersecting meshes