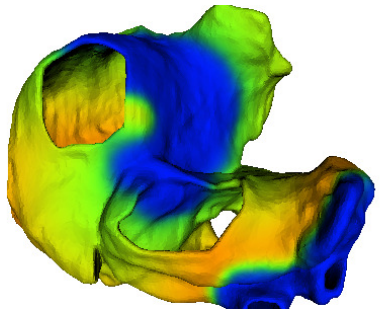
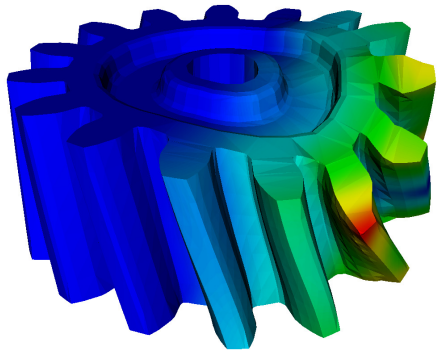
The FEniCS Project

Martin Alnæs, Johan Hake, Anders Logg,
Kent-Andre Mardal, Marie E. Rognes,
Garth N. Wells, Kristian B. Ølgaard,
and many others

Center for Biomedical Computing,
Simula Research Laboratory

2011–06–08





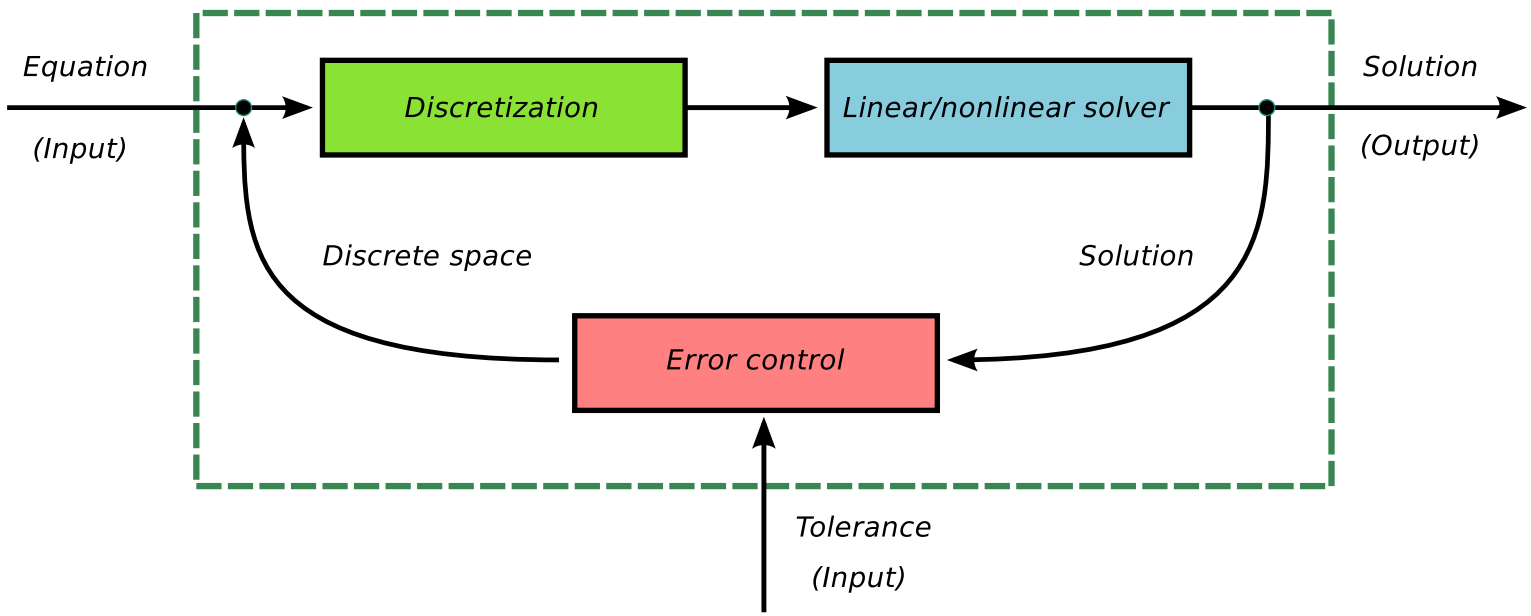
Automated solution of differential equations

Input

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

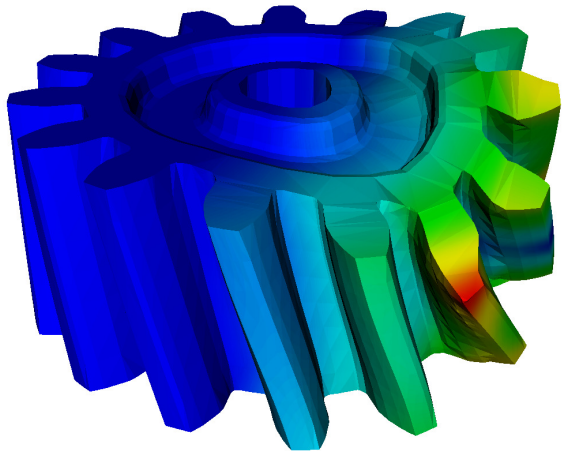
Output

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



FEniCS is an automated programming environment for differential equations

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

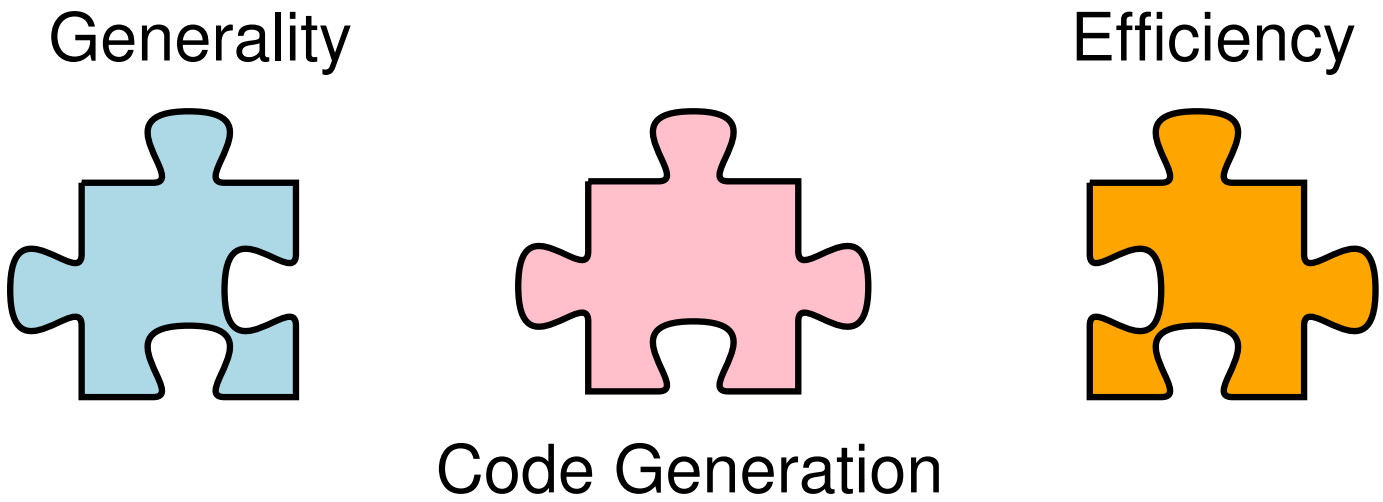


`http://www.fenicsproject.org/`

Collaborators

University of Chicago, Argonne National Laboratory, Delft University of Technology, Royal Institute of Technology KTH, Simula Research Laboratory, Texas Tech University, University of Cambridge, ...

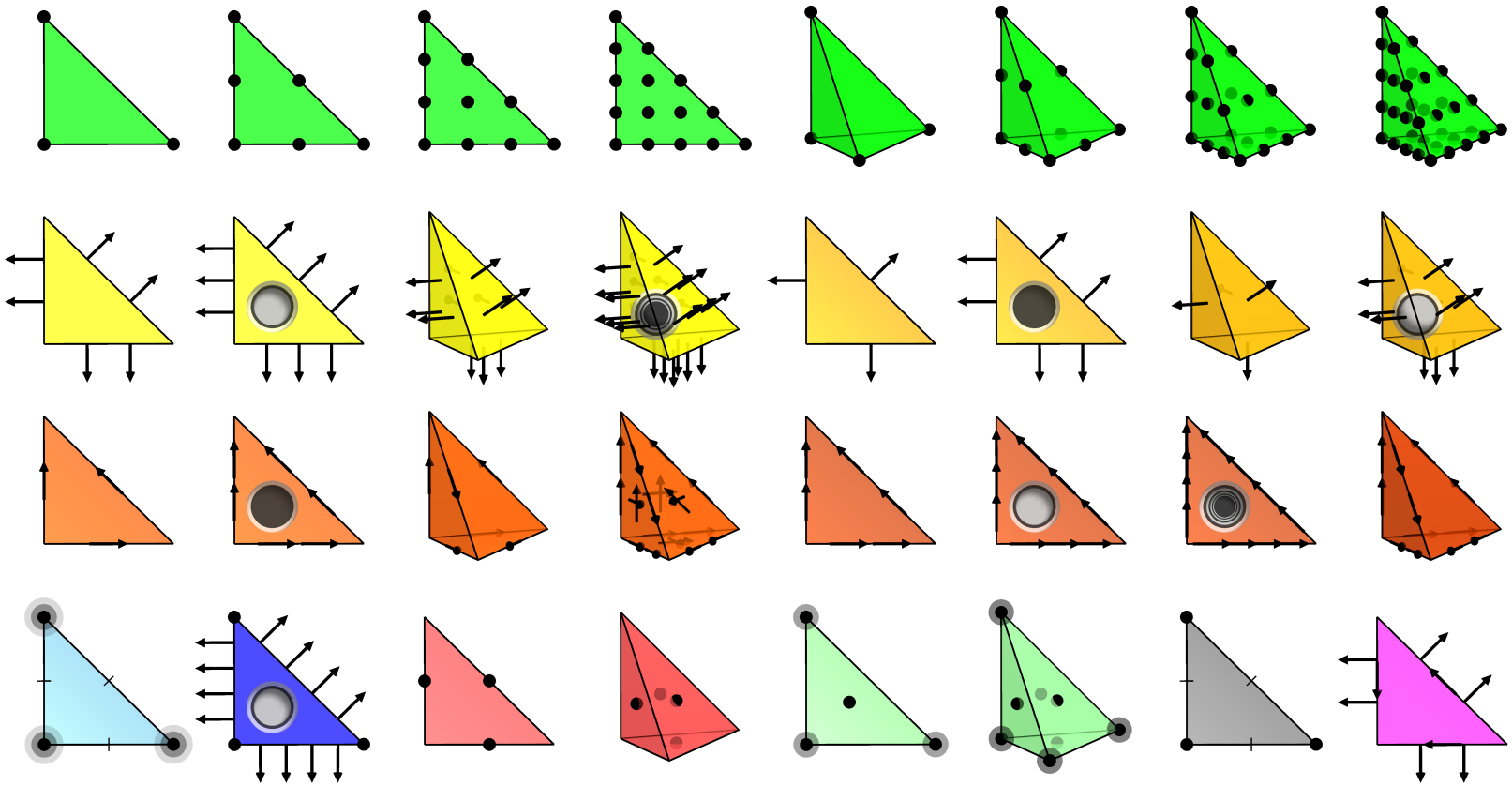
FEniCS is new technology combining generality, efficiency, simplicity and reliability

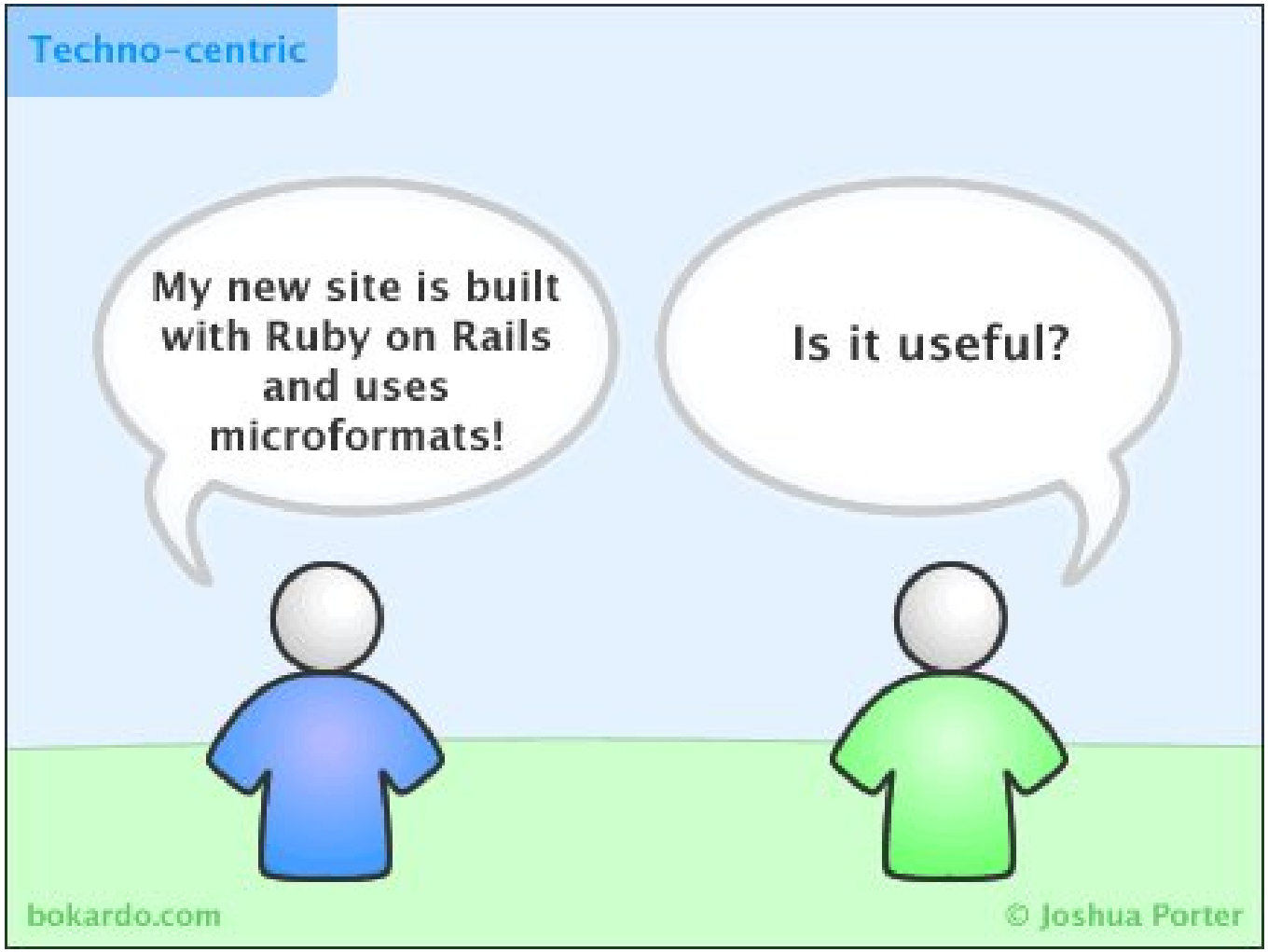


- Generality through *abstraction*
- Efficiency through *code generation*, *adaptivity*, *parallelism*
- Simplicity through *high level scripting*, *automation*
- Reliability through *adaptive error control*

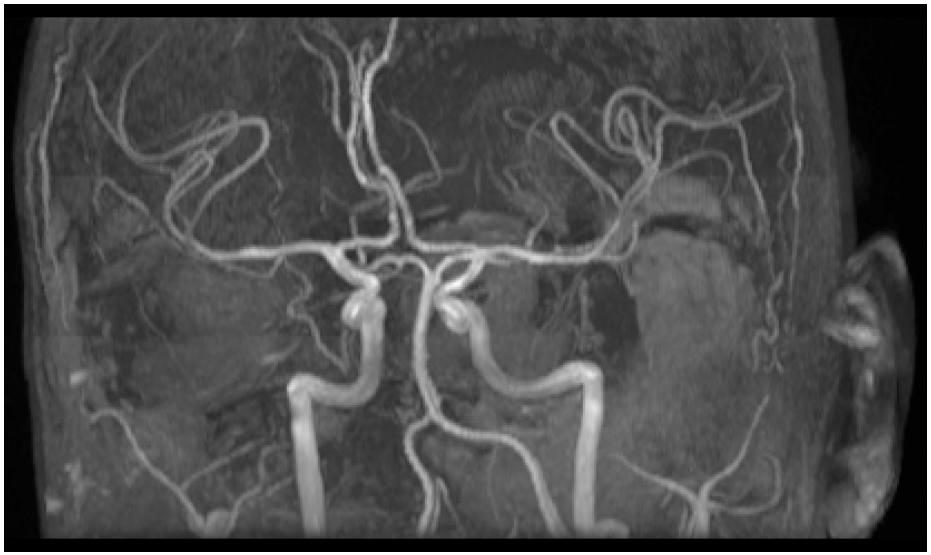
FEniCS is automated FEM

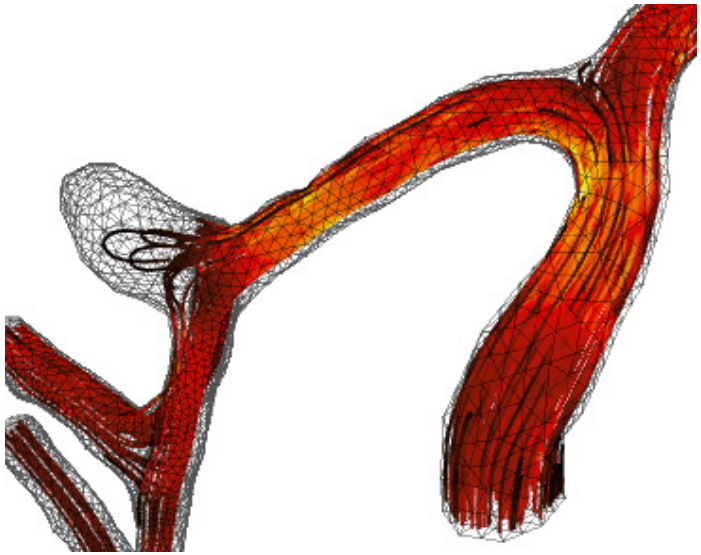
- Automated generation of basis functions
- Automated evaluation of variational forms
- Automated finite element assembly
- Automated adaptive error control




Techno-centric
My new site is built with Ruby on Rails and uses microformats!
Is it useful?
bokardo.com
© Joshua Porter

Computational hemodynamics





- Low wall shear stress may trigger aneurysm growth
- Solve the incompressible Navier–Stokes equations on patient-specific geometries

$$\dot{u} + \nabla u \cdot u - \nabla \cdot \sigma(u, p) = f$$
$$\nabla \cdot u = 0$$

Valen-Sendstad, Mardal, Logg, *Computational hemodynamics* (2011)

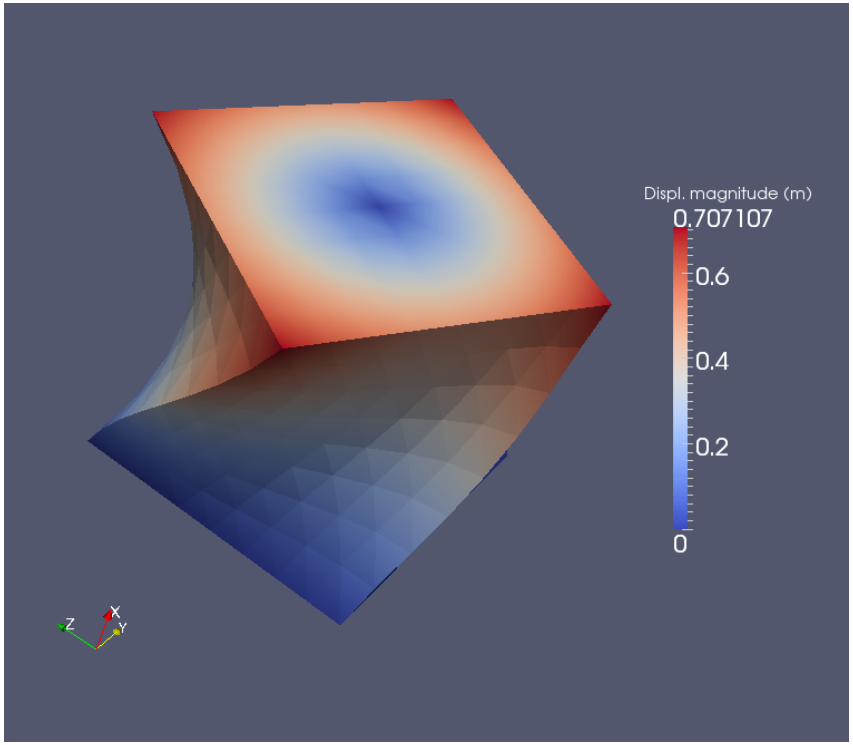
Computational hemodynamics (contd.)

```
# Define Cauchy stress tensor
def sigma(v,w):
    return 2.0*mu*0.5*(grad(v) + grad(v).T)  -
w*Identity(v.cell().d)

# Define symmetric gradient
def epsilon(v):
    return  0.5*(grad(v) + grad(v).T)

# Tentative velocity step (sigma formulation)
U = 0.5*(u0 + u)
F1 = rho*(1/k)*inner(v, u - u0)*dx +
rho*inner(v, grad(u0)*(u0 - w))*dx \
   + inner(epsilon(v), sigma(U, p0))*dx \
   + inner(v, p0*n)*ds - mu*inner(grad(U).T*n, v)*ds \
   - inner(v, f)*dx
a1 = lhs(F1)
L1 = rhs(F1)

# Pressure correction
a2 = inner(grad(q), k*grad(p))*dx
L2 = inner(grad(q), k*grad(p0))*dx - q*div(u1)*dx

# Velocity correction
a3 = inner(v, u)*dx
L3 = inner(v, u1)*dx + inner(v, k*grad(p0 - p1))*dx
```

- The Navier–Stokes solver is implemented in Python/FEniCS
- FEniCS allows the solver to be implemented in a minimal amount of code

Valen-Sendstad, Mardal, Logg, *Computational hemodynamics* (2011)

Tissue mechanics and hyperelasticity

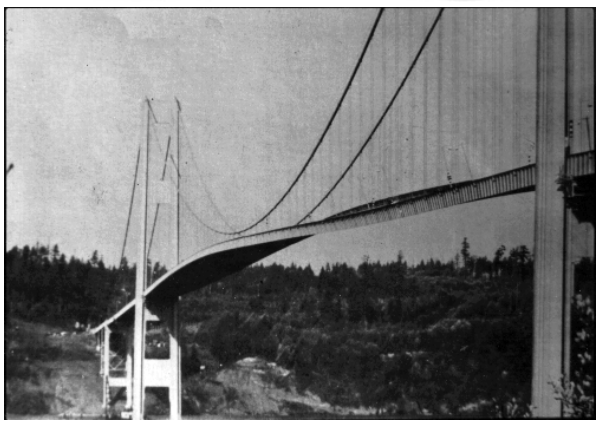



```
class Twist(StaticHyperelasticity):

    def mesh(self):
        n = 8
        return UnitCube(n, n, n)

    def dirichlet_conditions(self):
        clamp = Expression(("0.0", "0.0", "0.0"))
        twist = Expression(("0.0",
          "y0 + (x[1]-y0)*cos(theta)
              - (x[2]-z0)*sin(theta) - x[1]",
          "z0 + (x[1]-y0)*sin(theta)
              + (x[2]-z0)*cos(theta) - x[2]"))
        twist.y0 = 0.5
        twist.z0 = 0.5
        twist.theta = pi/3
        return [clamp, twist]

    def dirichlet_boundaries(self):
        return ["x[0] == 0.0", "x[0] == 1.0"]

    def material_model(self):
        mu    = 3.8461
        lmbda = Expression("x[0]*5.8+(1-x[0])*5.7")

        material = StVenantKirchhoff([mu, lmbda])
        return material

    def __str__(self):
        return "A cube twisted by 60 degrees"
```

- CBC.Solve is a collection of FEniCS-based solvers developed at the CBC
- CBC.Twist, CBC.Flow, CBC.Swing, CBC.Beat, ...

H. Narayanan, *A computational framework for nonlinear elasticity* (2011)

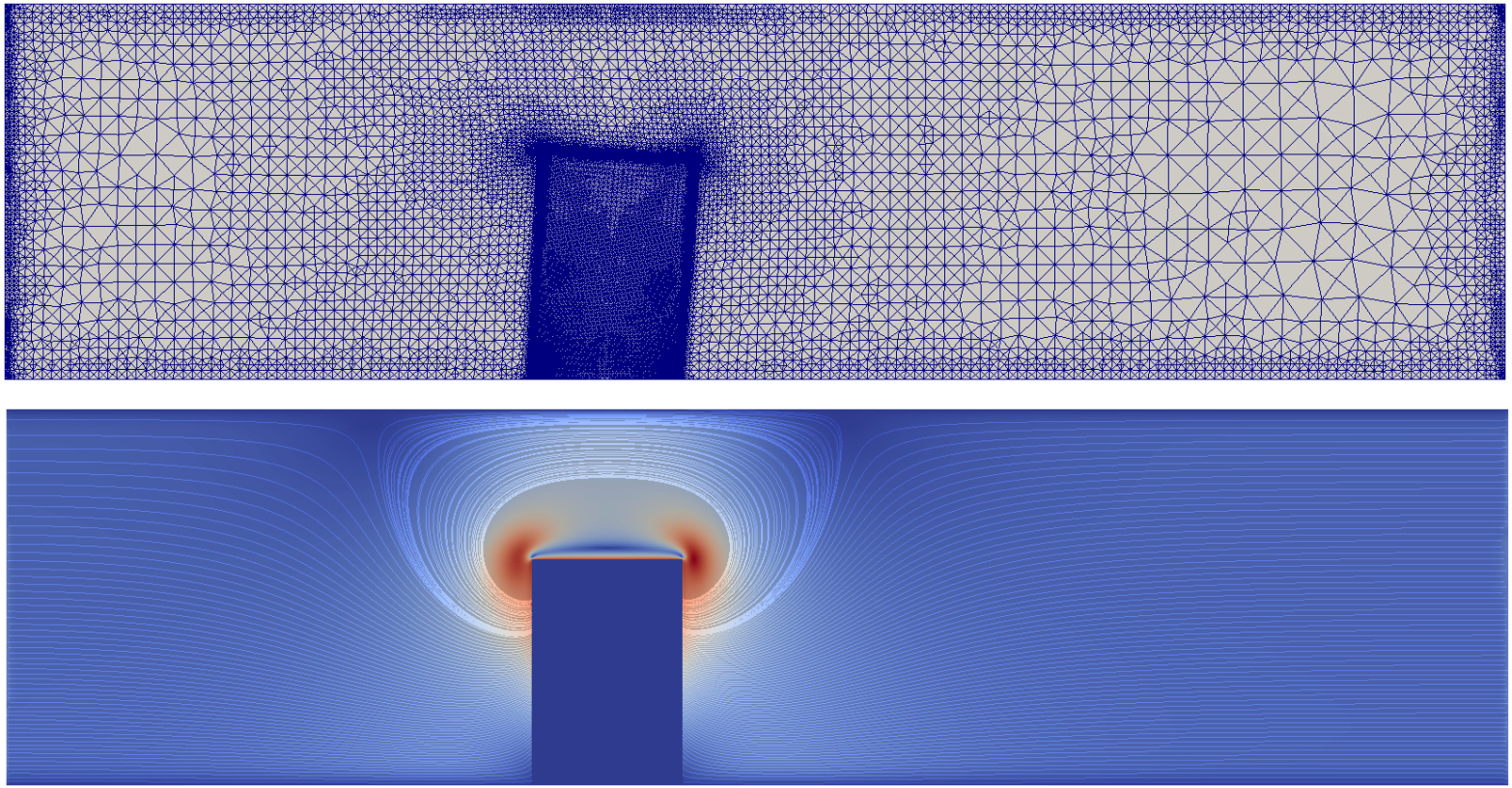
Fluid–structure interaction



- The FSI problem is a computationally very expensive coupled multiphysics problem
- The FSI problem has many important applications in engineering and biomedicine

Images courtesy of the Internet

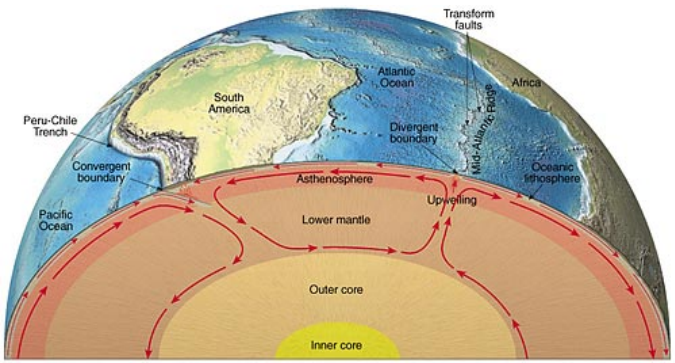
Fluid–structure interaction (contd.)



- Fluid governed by the incompressible Navier–Stokes equations
- Structure modeled by the St. Venant–Kirchhoff model
- Adaptive refinement in space and time

Selim, Logg, Narayanan, Larson, *An Adaptive Finite Element Method for FSI* (2011)

Computational geodynamics



$$-\operatorname{div}\sigma' - \nabla p = (Rb\,\phi - Ra\,T)\,e$$

$$\operatorname{div} u = 0$$

$$\frac{\partial T}{\partial t} + u \cdot \nabla T = \Delta T$$

$$\frac{\partial \phi}{\partial t} + u \cdot \nabla \phi = k_c \Delta \phi$$

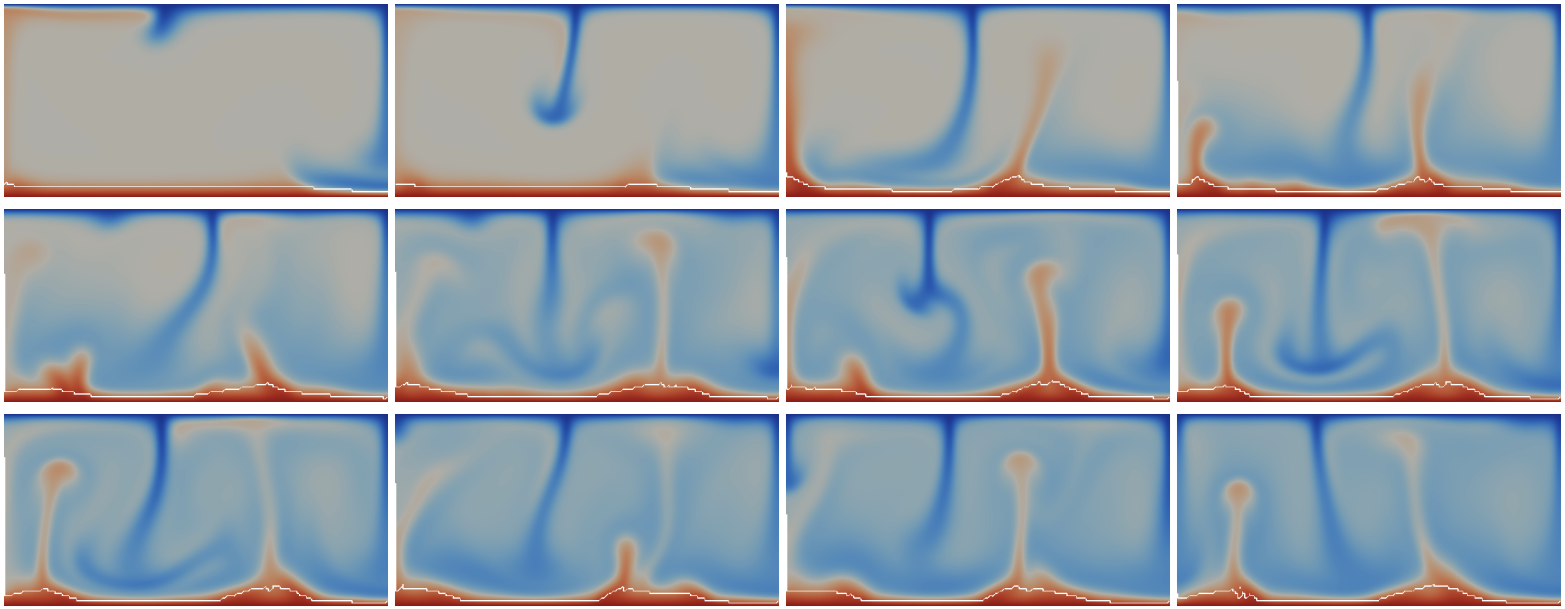
$$\sigma' = 2\eta\dot{\varepsilon}(u)$$

$$\dot{\varepsilon}(u) = \frac{1}{2}\left(\nabla u + \nabla u^T\right)$$

$$\eta = \eta_0 \exp\left(-bT/\Delta T + c(h - x_2)/h\right)$$

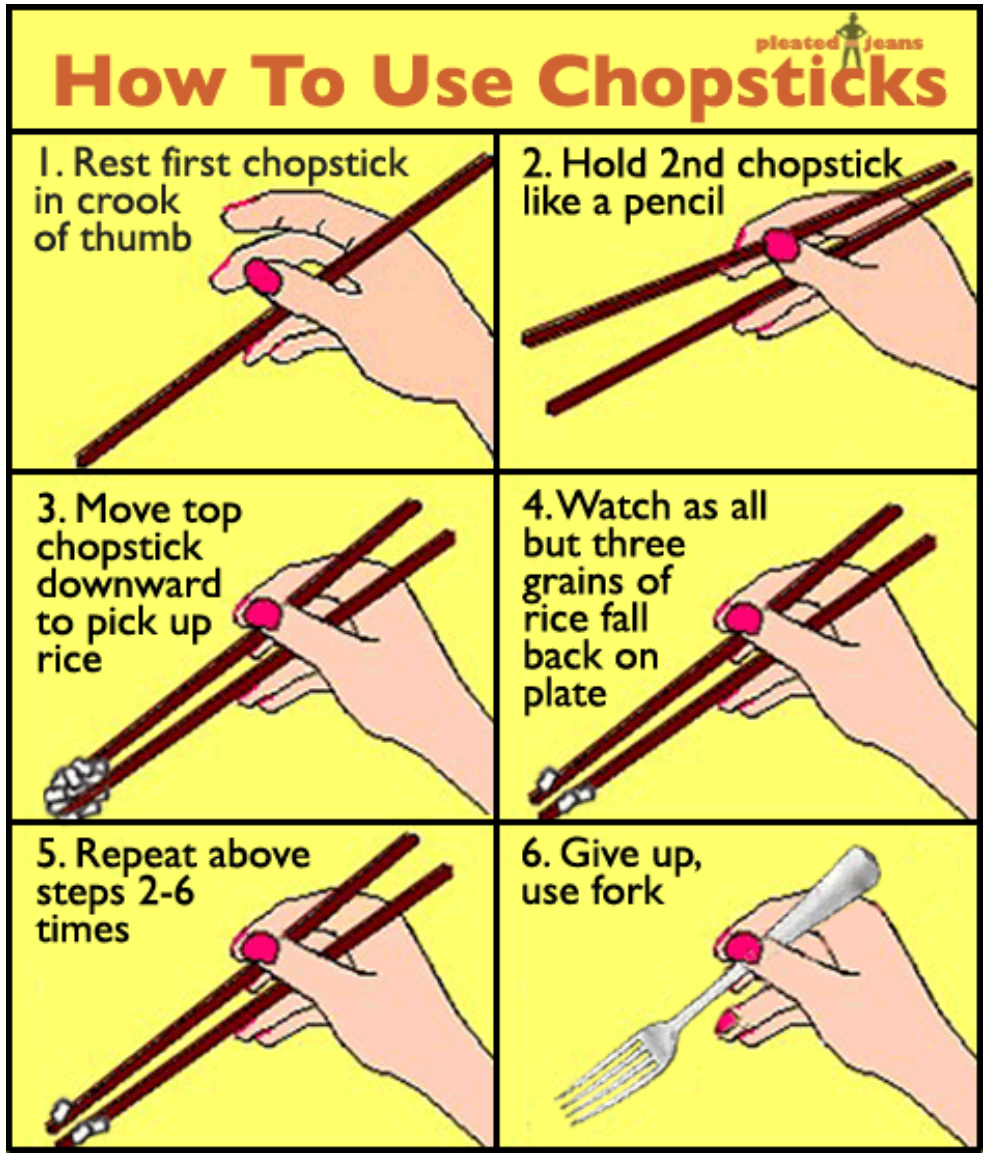
Image courtesy of the Internet

Computational geodynamics (contd.)



- The mantle convection simulator is implemented in Python/FEniCS
- Images show a sequence of snapshots of the temperature distribution

Vynnytska, Clark, Rognes, *Dynamic simulations of convection in the Earth's mantle* (2011)


How To Use Chopsticks
pleated jeans
1. Rest first chopstick in crook of thumb
2. Hold 2nd chopstick like a pencil
3. Move top chopstick downward to pick up rice
4. Watch as all but three grains of rice fall back on plate
5. Repeat above steps 2-6 times
6. Give up, use fork

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

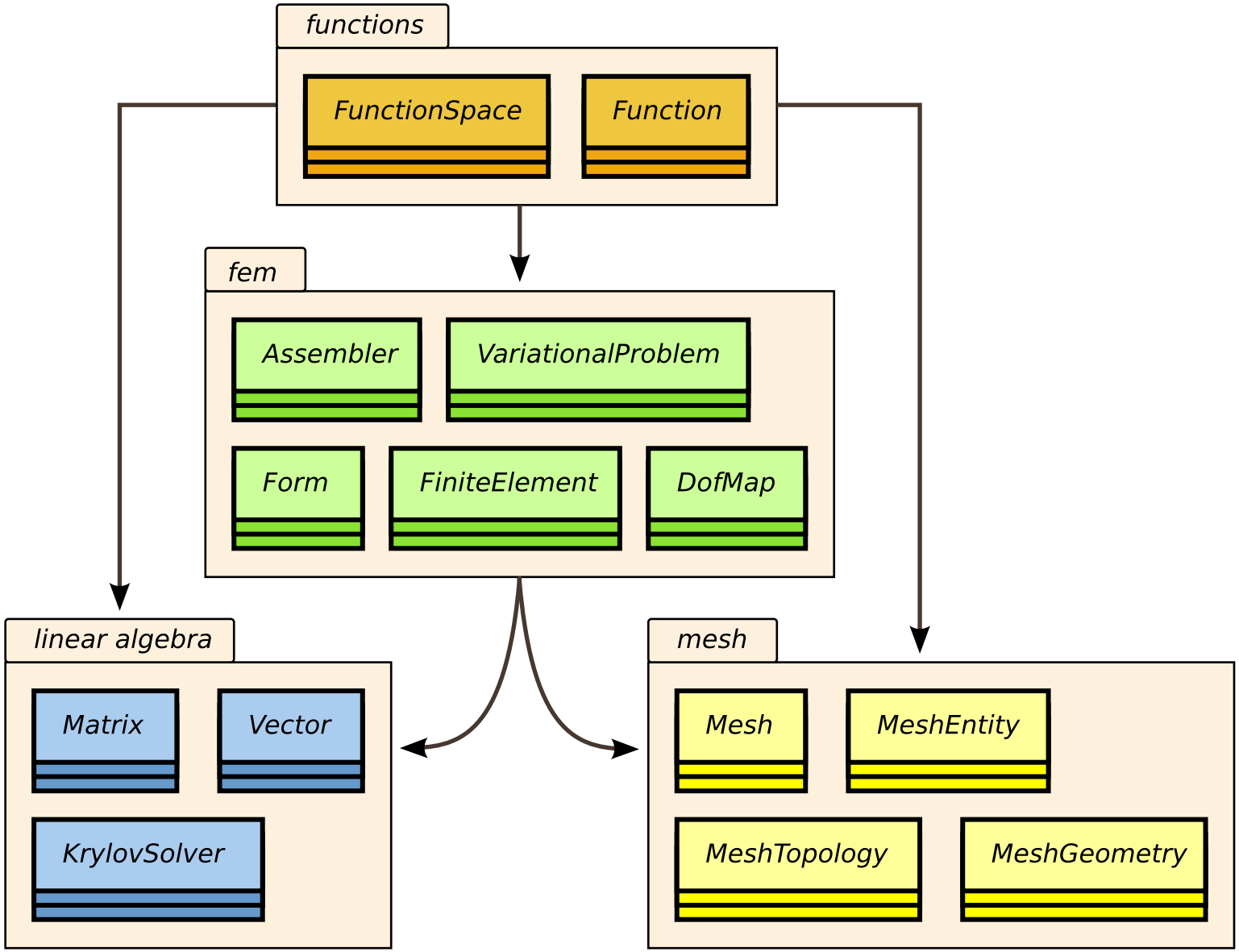
Find $u \in V$ such that

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

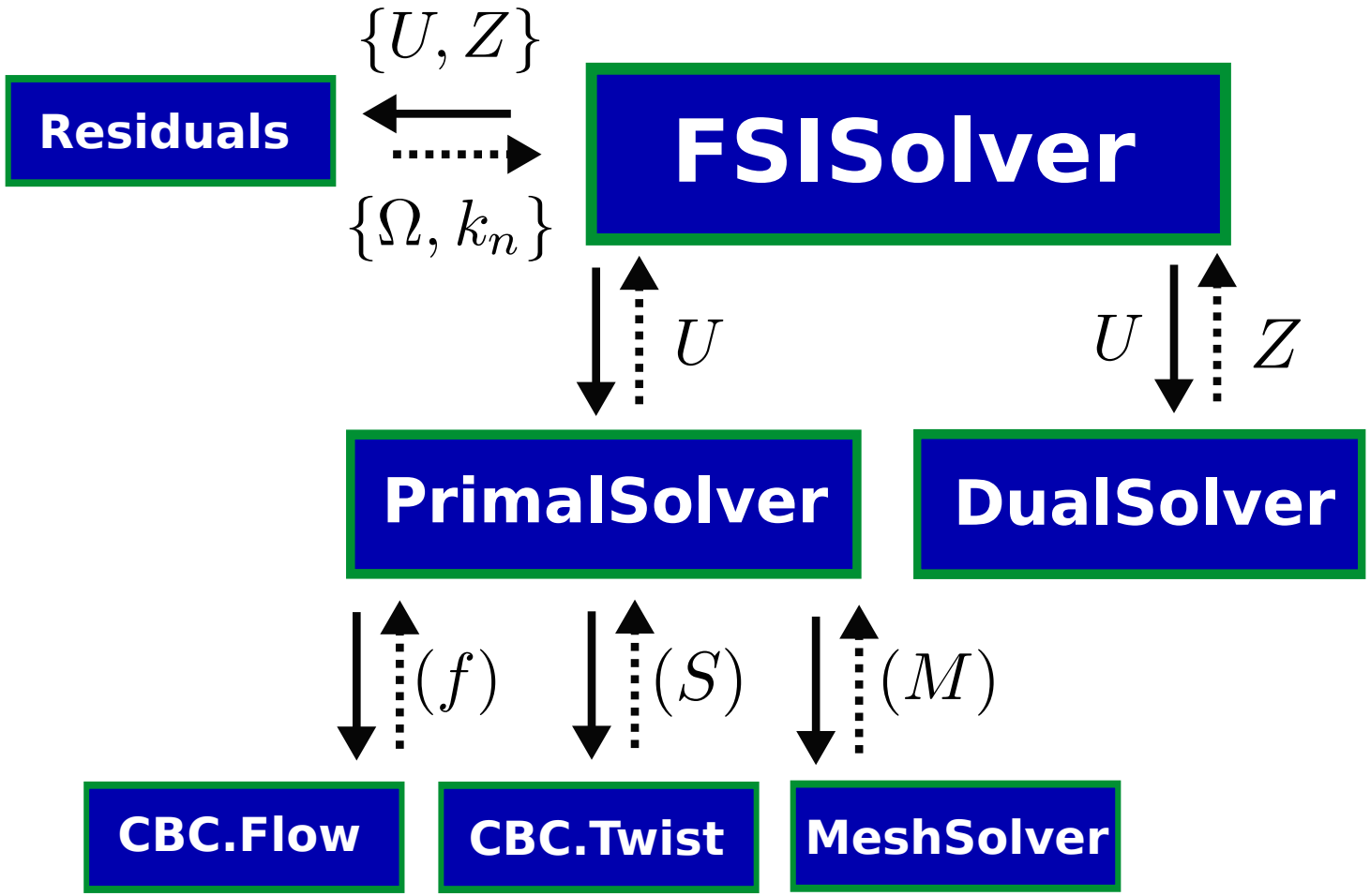
Hello World in FEniCS: implementation

```
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())

problem = VariationalProblem(a, L, bc)
u = problem.solve()
plot(u)
```

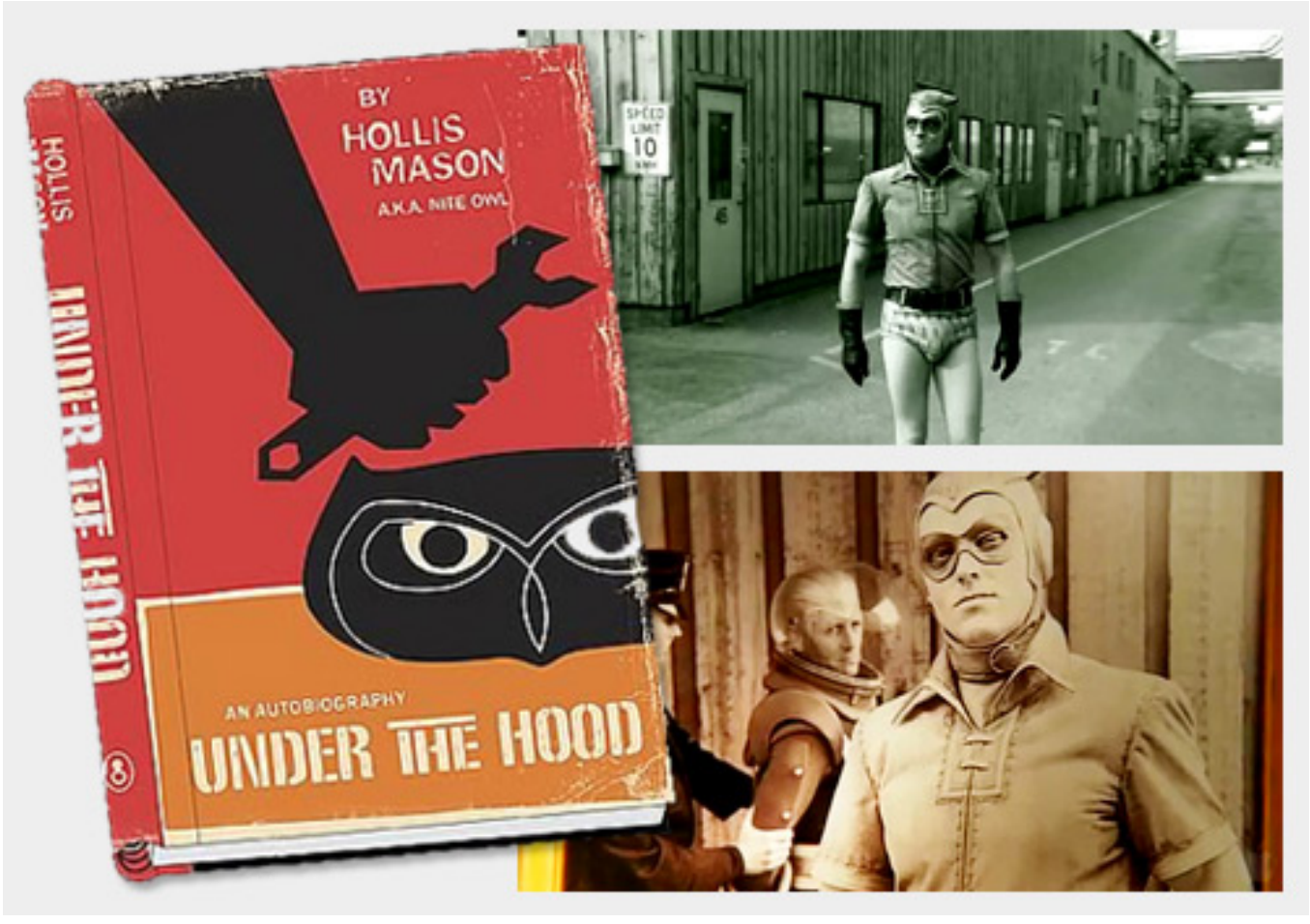
FEniCS (DOLFIN) class diagram



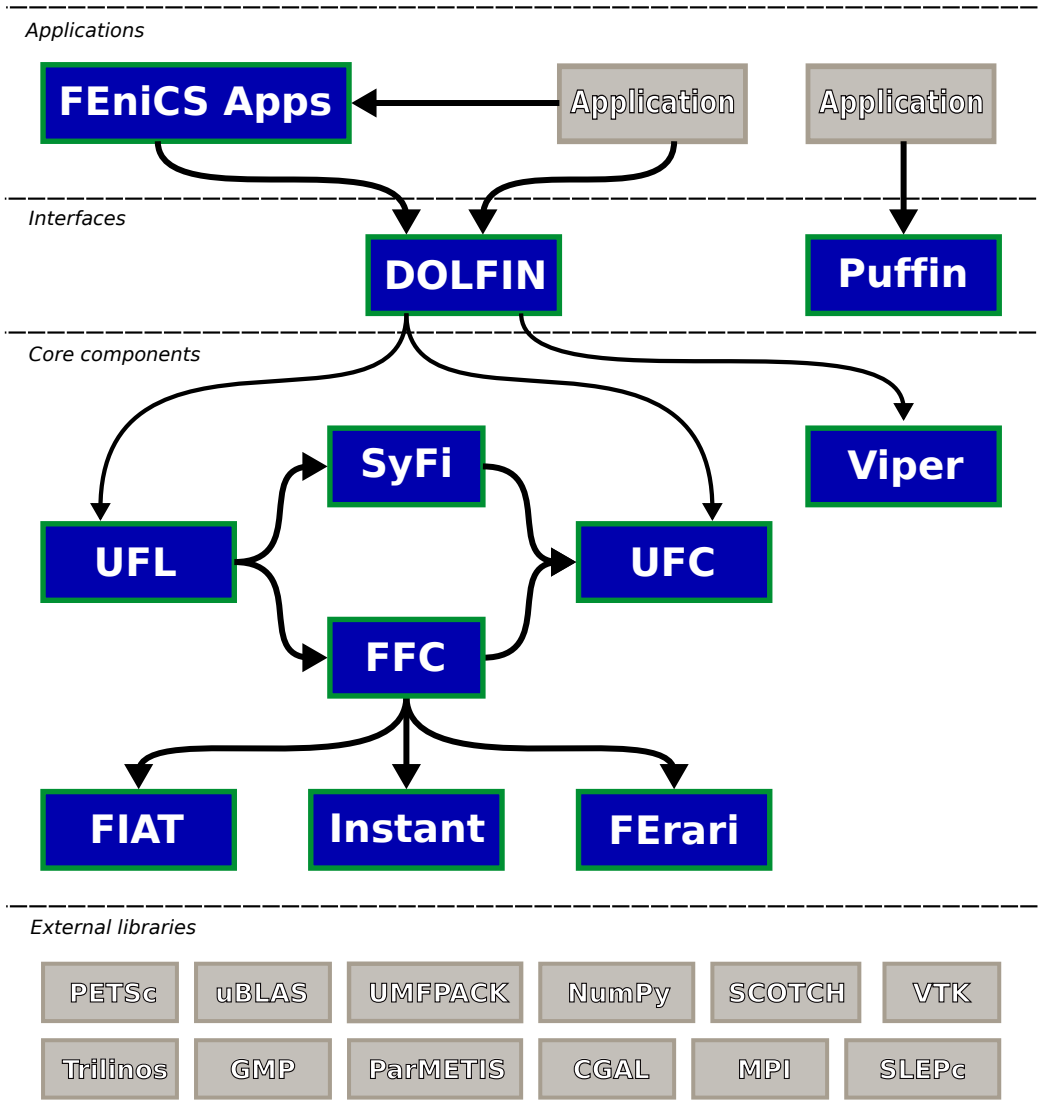
Implementation of advanced solvers in FEniCS



K. Selim, *An adaptive finite element solver for fluid–structure interaction problems* (2011)


BY
HOLLIS
MASON
A.K.A. NITE OWL
HOLLIS MASON
UNDER THE HOOD
AN AUTOBIOGRAPHY
UNDER THE HOOD
SPEED LIMIT 10

FEniCS software components



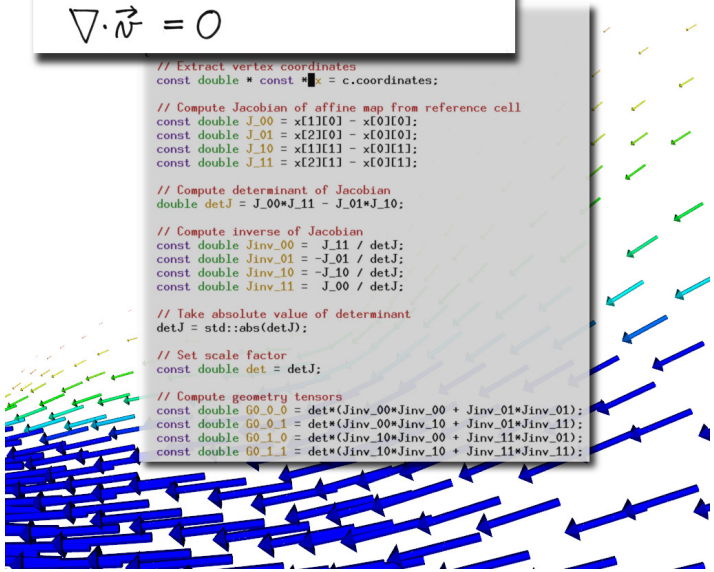
Automatic code generation

Equation (variational form)

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + \vec{u}\cdot\nabla\vec{u}\right) = -\nabla p + \mu\nabla^2\vec{u} + \rho\vec{b}$$
$$\nabla\cdot\vec{u} = 0$$

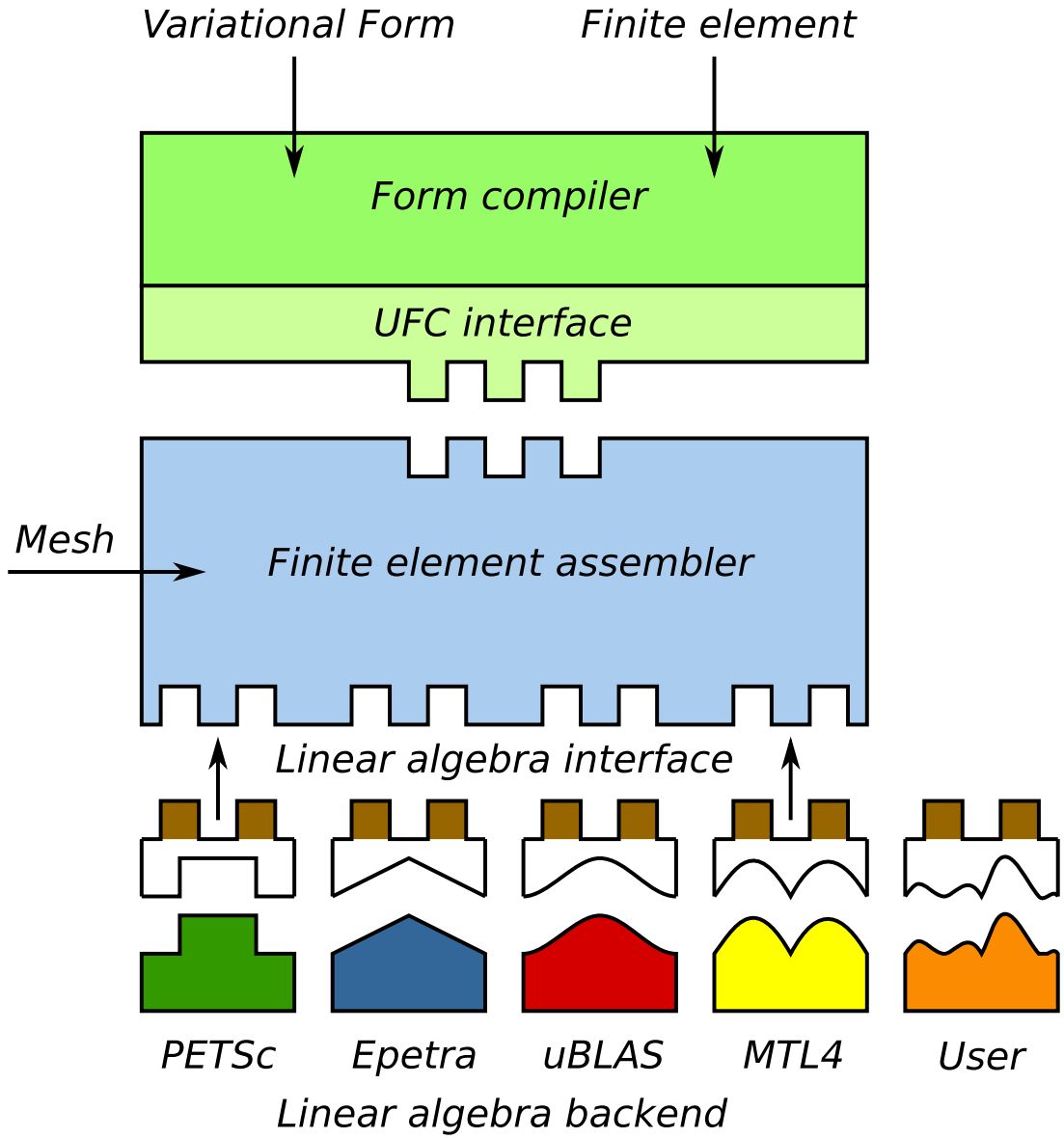
Form compiler

Application-specific code



```
// Extract vertex coordinates
const double * const * x = c.coordinates;

// Compute Jacobian of affine map from reference cell
const double J_00 = x[1][0] - x[0][0];
const double J_01 = x[2][0] - x[0][0];
const double J_10 = x[1][1] - x[0][1];
const double J_11 = x[2][1] - x[0][1];

// Compute determinant of Jacobian
double detJ = J_00*J_11 - J_01*J_10;

// Compute inverse of Jacobian
const double Jinv_00 =  J_11 / detJ;
const double Jinv_01 = -J_01 / detJ;
const double Jinv_10 = -J_10 / detJ;
const double Jinv_11 =  J_00 / detJ;

// Take absolute value of determinant
detJ = std::abs(detJ);

// Set scale factor
const double det = detJ;

// Compute geometry tensors
const double G0_0_0 = det*(Jinv_00*Jinv_00 + Jinv_01*Jinv_01);
const double G0_0_1 = det*(Jinv_00*Jinv_10 + Jinv_01*Jinv_11);
const double G0_1_0 = det*(Jinv_10*Jinv_00 + Jinv_11*Jinv_01);
const double G0_1_1 = det*(Jinv_10*Jinv_10 + Jinv_11*Jinv_11);
```

Combining it all with external libraries

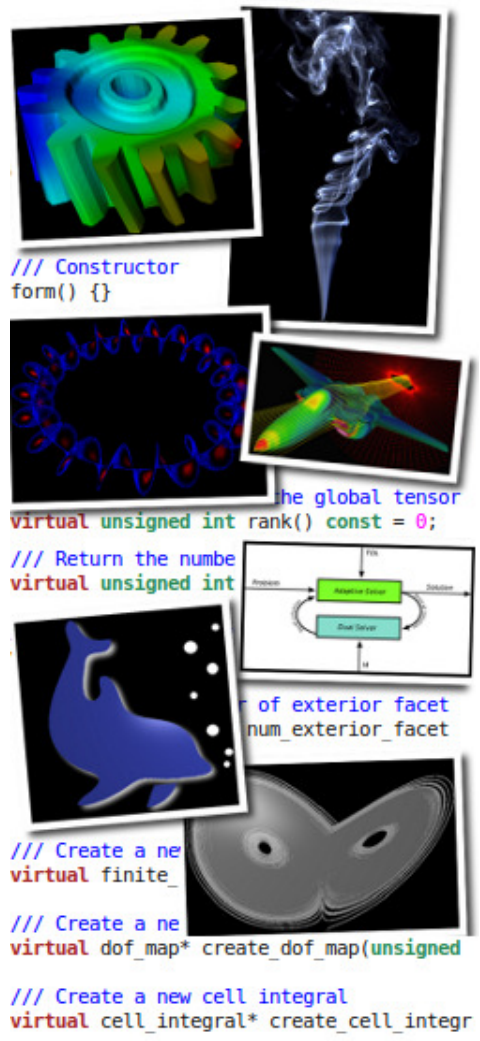


Summary

- Automated solution of differential equations
- Simple installation
- Simple scripting in Python
- Efficiency by automated code generation
- Free/open-source (LGPL)

Upcoming events

- Release of 1.0 (2011)
- Book (2011)
- New web page (2011)
- Mini courses / seminars (2011)



`http://www.fenicsproject.org/`

`http://www.simula.no/research/acdc/`