6. FEniCS Demo Session

Anders Logg
logg@tti-c.org

Toyota Technological Institute at Chicago

Sixth Winter School in Computational Mathematics
Geilo, March 5-10 2006

Downloading and Installing FEniCS

FEniCS Demos

Static linear elasticity

The Stokes equations

Convection–diffusion

Live Demo

- Chapter 10 in lecture notes
- FFC manual, DOLFIN manual

Downloading the source code

Source code available at

```
http://www.fenics.org/
```

Unpacking the source code:

```
# tar zxf FIAT-0.2.3.tar.gz
# tar zxf ffc-0.3.0.tar.gz
# tar zxf dolfin-0.6.0.tar.gz
```

Requirements:

- PETSc 2.3.1
- Libxml2
- Python
- Python Numeric
- Python LinearAlgebra

Accessing (hg) repositories

Mercurial Source Control Management (SCM):

- Fast
- Light-weight
- Implemented in Python
- Distributed
- `http://www.selenic.com/mercurial/`

FEniCS repositories at

```
http://www.fenics.org/hg/
```

Clone repositories:

```
# hg clone http://www.fenics.org/hg/ffc
# hg clone http://www.fenics.org/hg/dolfin
```

Working with (hg) repositories

CVS-like commands:

```
# hg add file
# hg remove file
# hg status file
# hg commit
```

Every directory (clone) is a complete repository:

```
# hg clone dolfin dolfin-dev
# cd dolfin-dev
# <edit files>
# hg commit
# hg push ../dolfin
```

Installing FIAT

Follows standard for installation of Python packages:

```
# python setup.py install
```

Installs FIAT in

```
/usr/lib/python2.4/site-packages/FIAT/
```

Non-root:

```
# mkdir ~/local
# python setup.py install --home ~/local
```

Installing FFC

Follows standard for installation of Python packages:

```
# python setup.py install
```

Installs FFC in

```
/usr/lib/python2.4/site-packages/ffc/
```

Non-root:

```
# mkdir ~/local
# python setup.py install --home ~/local
```

Installing DOLFIN

Follows GNU standard for installation of C/C++ libraries:

```
# ./configure
# make
# make install
```

Compiling demos:

```
# make demo
```

Non-root:

```
# mkdir ~/local
# ./configure --prefix=~/local
# make
# make install
```

Static linear elasticity

Differential equation:

$$\begin{aligned} -\nabla \cdot \sigma(u) &= f && \text{in } \Omega \\ u &= u_0 && \text{on } \Gamma_0 \subset \partial\Omega \\ \sigma(u)\hat{n} &= 0 && \text{on } \partial\Omega \setminus \Gamma_0 \end{aligned}$$

Stress tensor:

$$\sigma(v) = 2\mu\, \epsilon(v) + \lambda\, \mathrm{trace}(\epsilon(v)) I$$

Strain tensor:

$$\epsilon(v) = \frac{1}{2}\left(\nabla v + (\nabla v)^\top\right)$$

Variational (finite element) formulation

Find $U \in V_h$ such that

$$\int_\Omega \nabla v : \sigma(U) \, \mathrm{d}x = \int_\Omega v \cdot f \, \mathrm{d}x$$

for all $v \in \hat{V}_h$

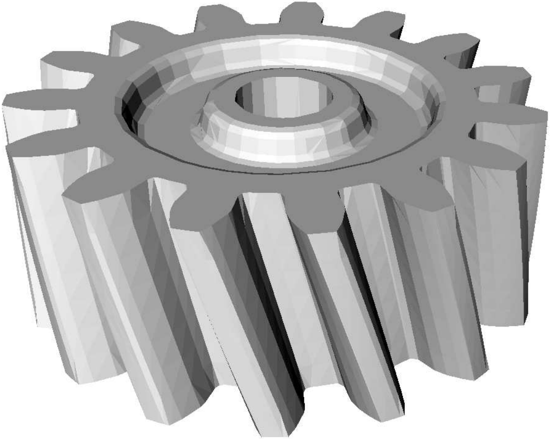
Implementation

```
element = FiniteElement("Vector Lagrange", "tetrahedron", 1)

v = BasisFunction(element)
U = BasisFunction(element)
f = Function(element)

E  = 10.0
nu = 0.3

mu    = E / (2*(1 + nu))
lmbda = E*nu / ((1 + nu)*(1 - 2*nu))
```

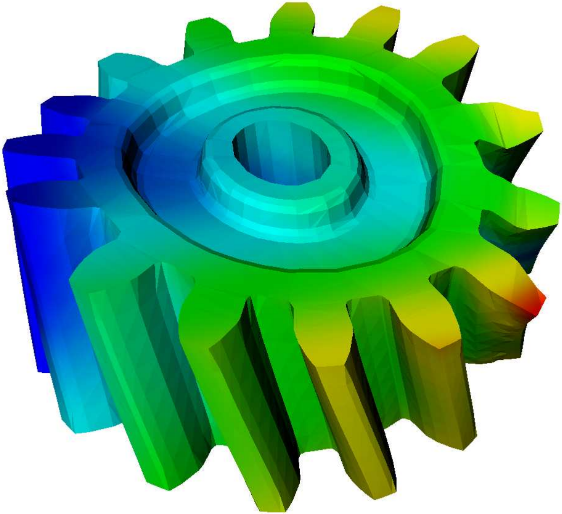
Implementation (cont'd)

```
def epsilon(v):
    return 0.5*(grad(v) + transp(grad(v)))

def sigma(v):
    return 2*mu*epsilon(v) + \
           lmbda*mult(trace(epsilon(v)), Identity(len(v)))

a = dot(grad(v), sigma(U))*dx
L = dot(v, f)*dx
```

Original domain



Solution



The Stokes equations

Differential equation:

$$
\begin{aligned}
-\Delta u + \nabla p &= f && \text{in } \Omega \\
\nabla \cdot u &= 0 && \text{in } \Omega \\
u &= u_0 && \text{on } \partial\Omega
\end{aligned}
$$

- Velocity $u = u(x)$
- Pressure $p = p(x)$

Variational (finite element) formulation

Find $(U, P) \in V_h = V_h^u \times V_h^p$ such that

$$\int_\Omega \nabla v : \nabla U - (\nabla \cdot v) P + q \nabla \cdot U \, \mathrm{d}x = \int_\Omega v \cdot f \, \mathrm{d}x$$

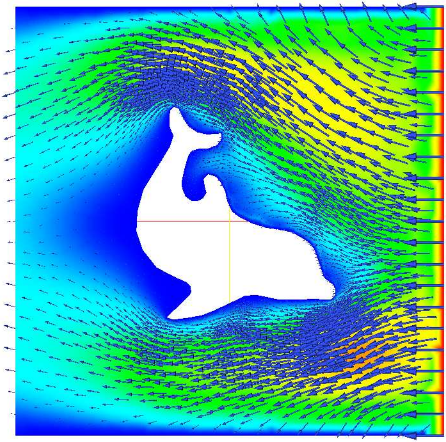
for all $(v, q) \in \hat{V}_h = \hat{V}_h^u \times \hat{V}_h^q$

- Approximating spaces $\hat{V}_h$ and V_h must satisfy the Babuška–Brezzi inf–sup condition
- Use Taylor–Hood elements:
  - P_q for velocity
  - P_{q-1} for pressure

Implementation

```
P2 = FiniteElement("Vector Lagrange", "triangle", 2)
P1 = FiniteElement("Lagrange", "triangle", 1)
TH = P2 + P1

(v, q) = BasisFunctions(TH)
(U, P) = BasisFunctions(TH)

f = Function(P2)

a = (dot(grad(v), grad(U)) - div(v)*P + q*div(U))*dx
L = dot(v, f)*dx
```

Solution (velocity field)



Stabilization

- Circumvent the Babuška–Brezzi condition by adding a stabilization term
- Modify the test function according to

$$(v, q) \rightarrow (v, q) + (\delta \nabla q, 0)$$

with $\delta = \beta h^2$

Find $(U, P) \in V_h = V_h^u \times V_h^p$ such that

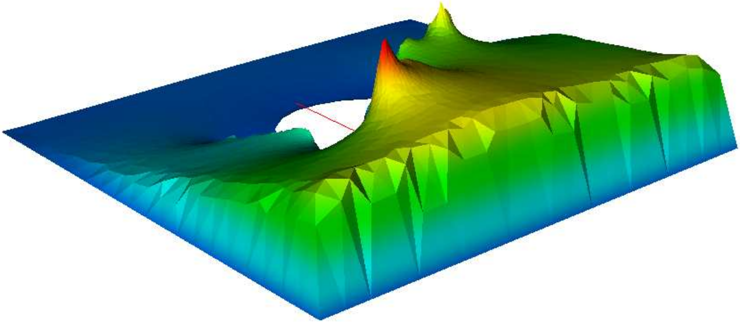
$$\int_\Omega \nabla v : \nabla U - (\nabla \cdot v) P + q \nabla \cdot U + \delta \nabla q \cdot \nabla P \, \mathrm{d}x = \int_\Omega v \cdot f \, \mathrm{d}x$$

for all $(v, q) \in \hat{V}_h = \hat{V}_h^u \times \hat{V}_h^q$

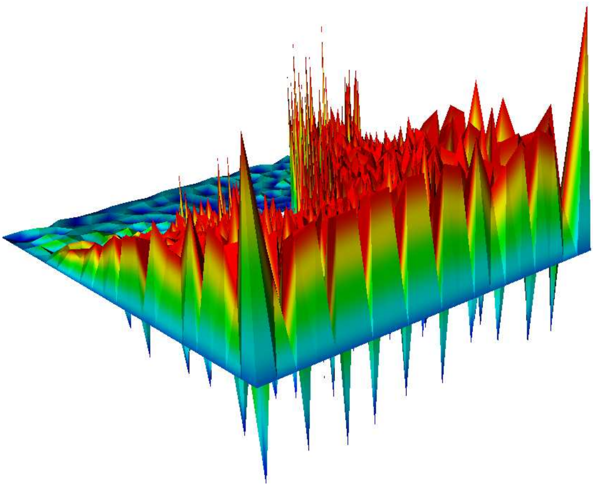
Implementation

```
vector = FiniteElement("Vector Lagrange", "triangle", 1)
scalar = FiniteElement("Lagrange", "triangle", 1)
system = vector + scalar

(v, q) = BasisFunctions(system)
(U, P) = BasisFunctions(system)

f = Function(vector)
h = Function(scalar)

d = 0.2*h*h

a = (dot(grad(v), grad(U)) - div(v)*P + q*div(U) + \
     d*dot(grad(q), grad(P)))*dx
L = dot(v + mult(d, grad(q)), f)*dx
```

Solution (pressure, stabilized)



Solution (pressure, unstabilized)



Convection–diffusion

Differential equation:

$$\begin{aligned} \dot{u} + b \cdot \nabla u - \nabla \cdot (c \nabla u) &= f && \text{in } \Omega \times (0, T] \\ u &= u_\partial && \text{on } \partial\Omega \times (0, T] \\ u &= u_0 && \text{at } \Omega \times \{0\} \end{aligned}$$

- Velocity field $b = b(x)$ from Stokes solution
- Omit stabilization for simplicity

Variational (finite element) formulation

A variational problem on each time step:

$$\int_{t_{n-1}}^{t_n} \int_\Omega (v, \dot{U}) + v\, b \cdot \nabla U + c \nabla v \cdot \nabla U \,\mathrm{d}x\,\mathrm{d}t = \int_{t_{n-1}}^{t_n} \int_\Omega v\, f \,\mathrm{d}x\,\mathrm{d}t$$

Find $U^n \in V_h$ such that

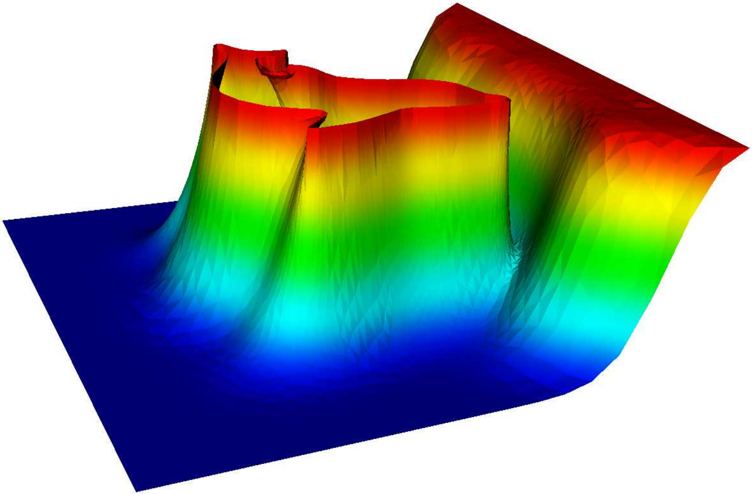
$$\frac{1}{k_n} \int_\Omega v\,(U^n - U^{n-1}) + v\, b \cdot \nabla (U^n + U^{n-1})/2 + c\, \nabla v \cdot \nabla (U^n + U^{n-1})/2 \,\mathrm{d}x = \int_{t_{n-1}}^{t_n} \int_\Omega v\, f \,\mathrm{d}x\,\mathrm{d}t$$

for all $v \in \hat{V}_h$, where $k_n = t_n - t_{n-1}$

Implementation

```
scalar = FiniteElement("Lagrange", "triangle", 1)
vector = FiniteElement("Vector Lagrange", "triangle", 2)

v  = BasisFunction(scalar)
U1 = BasisFunction(scalar)
U0 = Function(scalar)
b  = Function(vector)
f  = Function(scalar)

c = 0.005
k = 0.05

a = v*U1*dx + 0.5*k*(v*dot(b, grad(U1)) + \
    c*dot(grad(v), grad(U1)))*dx
L = v*U0*dx - 0.5*k*(v*dot(b, grad(U0)) + \
    c*dot(grad(v), grad(U0)))*dx + k*v*f*dx
```

Solution



Live demo

- Installing FIAT + FFC + DOLFIN
- FEniCS/C++
  - Implementing the variational problem
  - Compiling the variational problem
  - Writing the C++ program
  - Compiling and running the program
- FEniCS/Python
  - Writing the Python script
  - Running the script
- Visualizing the solution
  - ParaView
  - MayaVi
  - MATLAB