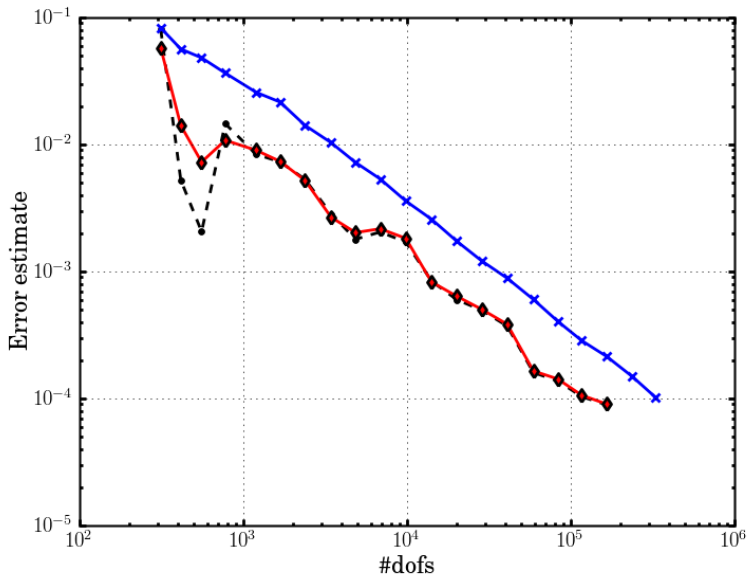
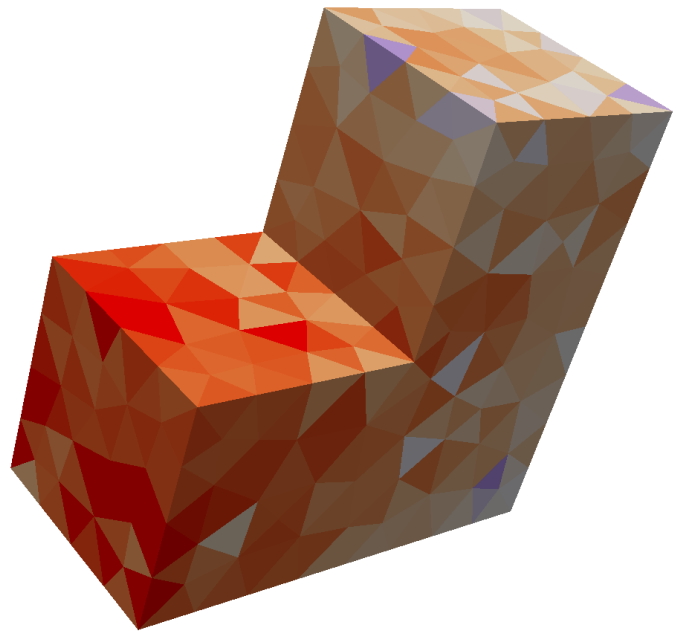
# Automated goal-oriented error control for stationary variational problems (in 12.5 min)

Marie E. Rognes

Simula Research Laboratory



'Automated goal-oriented error control I: stationary variational problems'.

Marie E. Rognes and Anders Logg. In preparation. 2010.

# What is automated goal-oriented error control?

## Input

- PDE: find $u \in V$ such that $a(v, u) = L(v) \quad \forall\, v \in V$
- Quantity of interest/Goal: $\mathcal{M} : V \to \mathbb{R}$
- Tolerance: $\epsilon > 0$

## Challenge

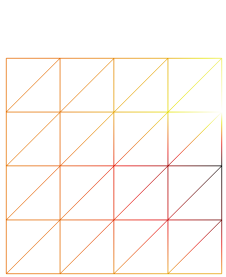
Find $V_h \subset V$ such that $|\mathcal{M}(u) - \mathcal{M}(u_h)| < \epsilon$ where $u_h \in V_h$ is determined by

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

## DOLFIN

```
pde = AdaptiveVariationalProblem(a - L, M)
u_h = pde.solve(1.0e-3)
```

# Adaptivity = solve – estimate – indicate – refine

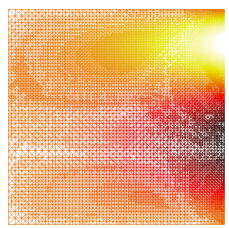


Solve

$u_h$

Estimate

$\eta_h < \epsilon$



Indicate

$\{\eta_T\}_{T \in \mathcal{T}_h}$

Refine

Dual

# The error measured in the goal is the residual of the dual solution

1. Define residual

$$r(v) := L(v) - a(v, u_h)$$

2. Introduce dual problem

$$\text{Find } z \in V\text{:} \quad a^*(v, z) = \mathcal{M}(v) \quad \forall\, v \in V$$

3. Dual solution + residual $\implies$ error

$$\mathcal{M}(u) - \mathcal{M}(u_h) = L(z) - a(z, u_h) = r(z) = r(z - z_h)$$

4. A good dual approximation $\tilde{z}_h$ gives computable error estimate

$$\eta_h = r(\tilde{z}_h)$$

5. Error indicators ... ?

# Let us take Poisson's equation as an example for manual derivation of error indicators

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

Recall error representation:

$$\mathcal{M}(u) - \mathcal{M}(u_h) = r(z) = \int_\Omega z f - \nabla z \cdot \nabla u_h \,\mathrm{d}x$$

Residual decomposition

$$r(v) = \sum_{T \in \mathcal{T}_h} \int_T v \underbrace{(f + \operatorname{div} \nabla u_h)}_{R_T} + \int_{\partial T} v \underbrace{(-\nabla u_h \cdot n)}_{R_{\partial T}} \,\mathrm{d}s$$

Error indicators:

$$\eta_T = |\langle \tilde{z}_h - z_h, R_T \rangle_T + \langle \tilde{z}_h - z_h, [\![ R_{\partial T} ]\!] \rangle_{\partial T}|$$

# The residual decomposition can be automatically computed for a class of residuals

**Have:** $a - L$ and $u_h \implies r$

**Want:** $\eta_T = |\langle \tilde{z}_h - z_h, R_T \rangle_T + \langle \tilde{z}_h - z_h, [\![ R_{\partial T} ]\!] \rangle_{\partial T}|$

**Need:** Residual decomposition $R_T$, $R_{\partial T}$ for each cell $T$

## Assumptions

1. $r(v) = \sum_T r_T(v)$
2. $r_T(v) = \int_T v \cdot R_T + \int_{\partial T} v \cdot R_{\partial T}$
3. $R_T \in P_k(T)$, $R_{\partial T}|_e \in P_q(e)$ for some integer $k, q$

# We can compute $R_T$ and $R_{dT}$ by solving small local variational problems

Recall assumption:

$$r_T(v) = \int_T v \cdot R_T \,\mathrm{d}x + \int_{\partial T} v \cdot R_{\partial T} \quad \text{with} \quad R_T \in P_k(T)$$

Let

- $b_T : T \to \mathbb{R}$ such that $b_T|_{\partial T} = 0$ (Bubble)
- $\{\phi_i\}_{i=1}^n$ be a basis for $P_k(T)$

**Lemma**

*$R_T$ is uniquely determined by the equations*

$$\int_T b_T \phi_i \cdot R_T \,\mathrm{d}x = r_T(b_T \phi_i)$$

$i = 1, \ldots, n$

```
b_T = Bubble(...)
R_T = TrialFunction(P_k)
phi = TestFunction(P_k)

a = inner(b_T*phi, R_T)*dx
L = r(b_T*phi)

pde = VariationalProblem(a, L)
R_T = pde.solve()
```

# An improved dual approximation can be computed by higher-order extrapolation

Dual approximation using primal discretization:

$$a^*(v, z_h) = \mathcal{M}(v) \quad \forall\, v \in V_h$$

gives $\eta_h = r(z_h) = 0$ (BAD).

## Suggested solution

Let $W_h \supset V_h$.

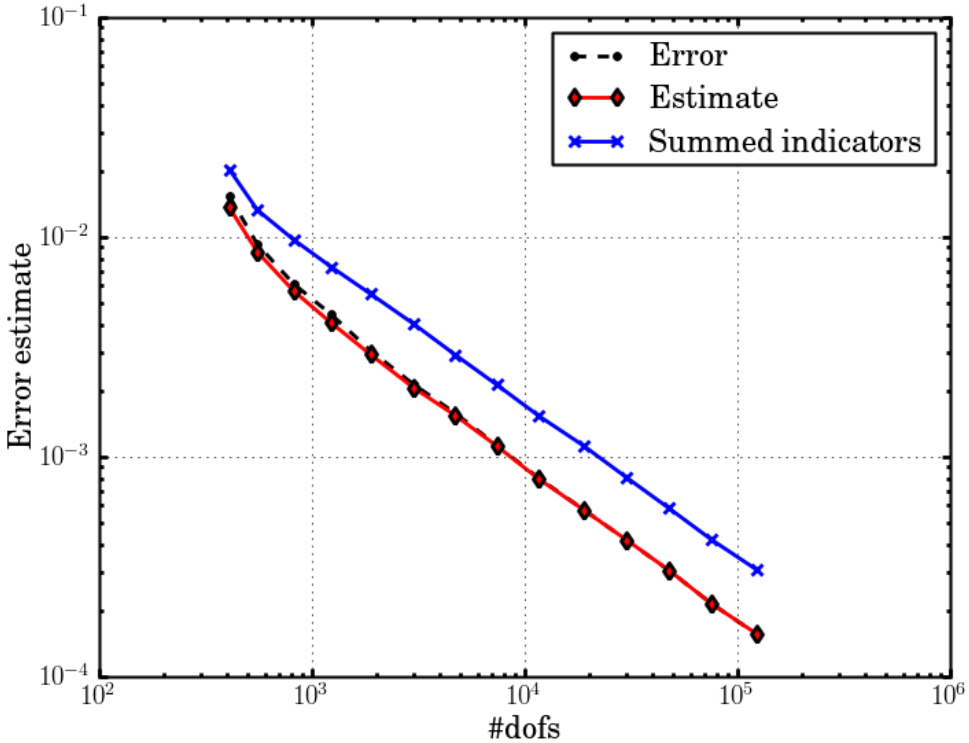
```
V_h = FunctionSpace(mesh, "CG", 1)
W_h = FunctionSpace(mesh, "CG", 2)
```

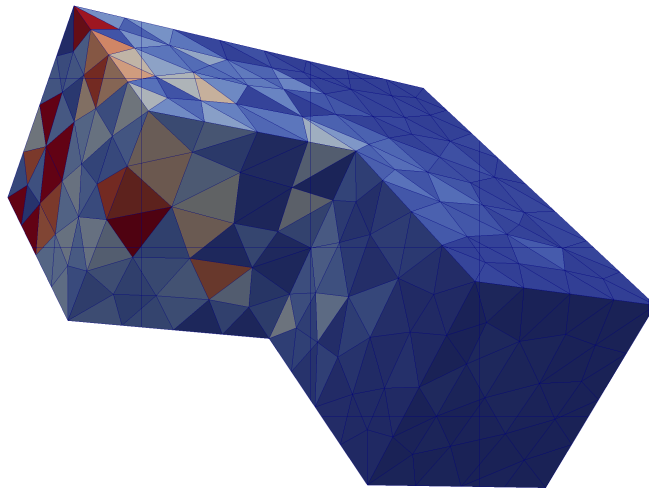
Improve approximation by a higher-order least-squares curve fitting procedure (extrapolation):

$$z_h \mapsto \tilde{z}_h = E_h z_h, \quad E_h : V_h \to W_h$$

```
z_h_tilde = extrapolate(z_h, W_h)
```

# The error estimates are virtually perfect for Poisson on a 3D L-shape

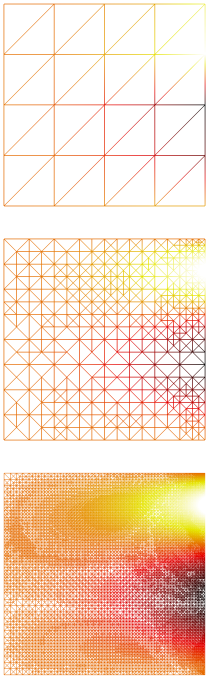


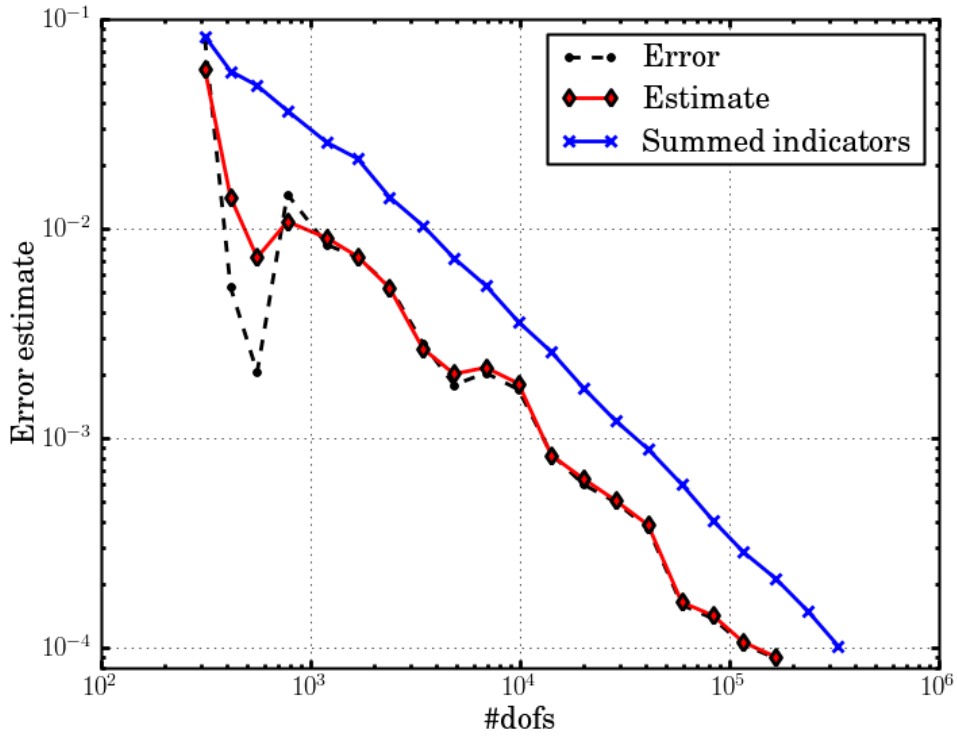



$$a(v,u) = \langle \nabla v, \nabla u \rangle,$$

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

# The error estimates are highly satisfactory for a three-field mixed elasticity formulation also

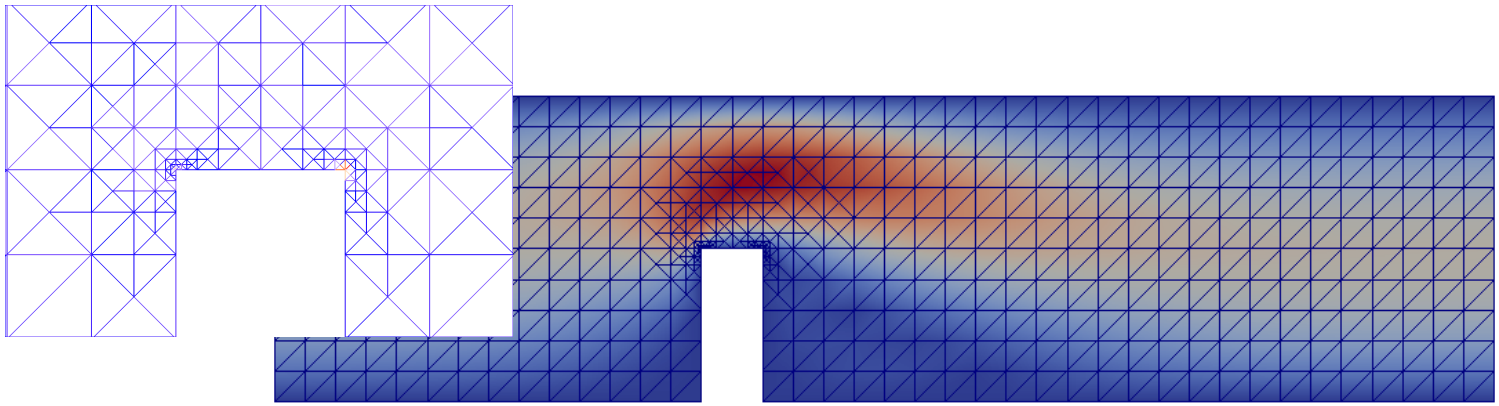





$$a((\tau, v, \eta), (\sigma, u, \gamma)) = \langle \tau, A\sigma \rangle + \langle \operatorname{div} \tau, u \rangle + \langle v, \operatorname{div} \sigma \rangle + \langle \tau, \gamma \rangle + \langle \eta, \sigma \rangle$$

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

# Goal-oriented adaptivity is worth it



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)

# Define test functions and unknown(s)
(v, q) = TestFunctions(V * Q)
w = Function(V * Q)
(u, p) = (as_vector((w[0], w[1])), w[2])

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

# Define goal and pde
M = u[0]*ds(0)
pde = AdaptiveVariationalProblem(F, bcs=[...], M, u=w, ...)

# Compute solution
(u, p) = pde.solve(1.e-4).split()
```