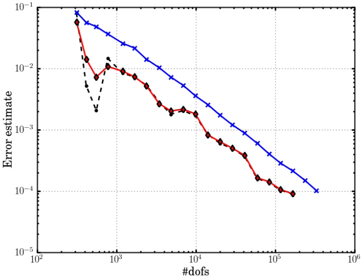
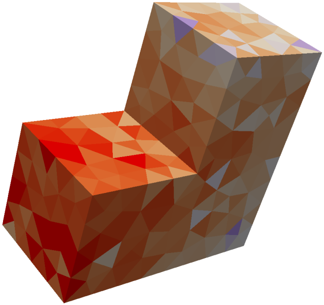
Automated goal-oriented error control with applications to nonlinear elasticity

Marie E. Rognes and Anders Logg

Simula Research Laboratory

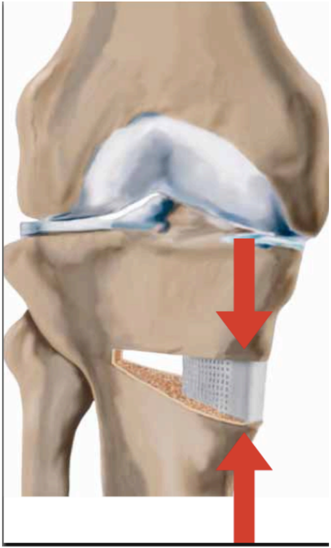




'Automated goal-oriented error control I: stationary variational problems'.

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

Motivation I: Artificial bone implants may be modelled using polymer-fluid mixtures (gels)



Find deformation x and volume fraction ϕ minimizing energy:

$$\mathcal{E}(x, \phi) = \int_\Omega W(x, \phi)\, dX$$

constrained by balance of mass:

$$\phi \det(\nabla x) = \phi_I$$

Quantity of interest

Shear stress at interface = ?

Motivation II: Linearization reduces the gel problem to a linear elasticity problem, but ...

$$\underbrace{W(x,\phi)}_{\text{Total potential}} = \underbrace{W_E(\nabla x,\phi)}_{\text{Elastic}} + \det(\nabla x)\big(\underbrace{W_{FH}(\phi)}_{\text{Flory-Huggins}} + c_{FH}\big)$$

Linearized boundary value problem

$$\sigma - C_{r(\phi_I)}[\nabla u] = r(\phi_I) I,$$
$$\operatorname{div} \sigma = 0.$$

[R., Micek and Calderer, SIAP, 2009]

Challenges (and solutions)

1. The small deformation regime too restrictive. (Automated differention!)
2. The full nonlinear problem is computationally intense. (Automated goal-oriented error control!)

The FEniCS project (www.fenics.org)

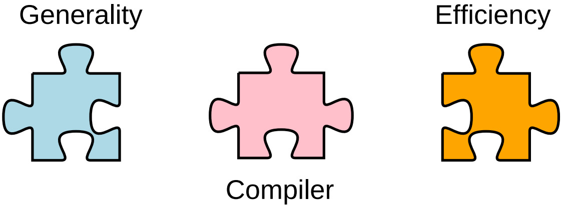
Free Software for Automated Scientific Computing

Agenda

1. Automation of discretization ✓
2. Automation of error control
3. ...

Key components

- High-level form language (UFL)
- Form compiler (FFC)
- Main interface (DOLFIN)



UFL closely resembles mathematical syntax (and supports automated differentiation of forms)

$$F = \nabla x$$
$$\phi = \phi_I \det(F)^{-1}$$
$$W_{FH}(\phi) = a\phi \ln \phi + \dots$$
$$W(x, \phi) = \dots$$
$$S = \frac{\partial W}{\partial F}$$

```
F = grad(x)
phi = phi_I*inv(det(F))

W_FH = a*phi*ln(phi) + ...
W = W_E + det(F)*(W_FH + c_FH)

S = diff(W, F)
```

Variational formulation

$$B(v; x) = \int_\Omega S \cdot \nabla v \, dX$$

```
v = TestFunction(V)
B = inner(S, grad(v))*dx
```

Variational problem: find x such that

$$B(v; x) = 0 \quad \forall\, v \in V$$

```
pde = VariationalProblem(B, ...,)
x = pde.solve()
```

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

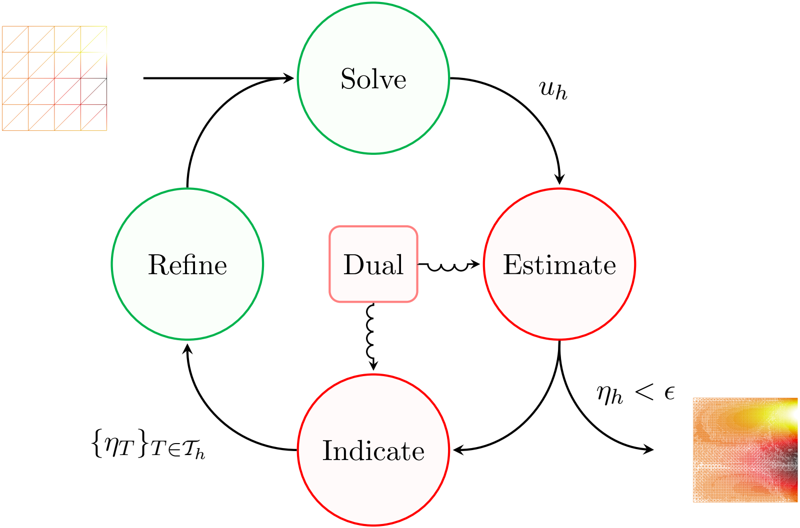
Find $V_h \subset V$ such that $|\mathcal{M}(u) - \mathcal{M}(u_h)| < \epsilon$ where $u_h \in V_h$ is determined by

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

FEniCS/DOLFIN

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

Adaptivity = solve – estimate – indicate – refine



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

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

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:

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

Babuška and Rheinboldt, '79, Verfürth '89, '94, '98, '00, Becker and Rannacher '01, ...

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 by solving a small variational problem on each cell

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)

lhs = inner(b_T*phi, R_T)*dx
rhs = r(b_T*phi)

pde = VariationalProblem(lhs, rhs)
R_T = pde.solve()
```

R. and Logg '10 (In preparation)

We can compute $R_{\partial T}$ by solving a small variational problem on each facet of each cell

By assumption

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

Aim

To compute R_e for each facet $e \subset \partial T$ for each cell $T \in \mathcal{T}_h$:

Let

- $b_e : T \to \mathbb{R}$ be such that $b_e|_{\partial T \setminus e} = 0$
- $\{\psi_i\}_{i=1}^m$ be a basis for $P_k(e)$ (NB: $\psi_i : T \to \mathbb{R}$)

Lemma

R_e is uniquely determined by the equations

$$\int_e b_e \psi_i \cdot R_e \, \mathrm{d}s = r_T(b_e \psi_i) - \int_T b_e \psi_i \cdot R_T \, \mathrm{d}x \quad i = 1, \ldots, m$$

R. and Logg '10 (In preparation)

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

Dual problem

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

can be generated and solved automatically.

Problem

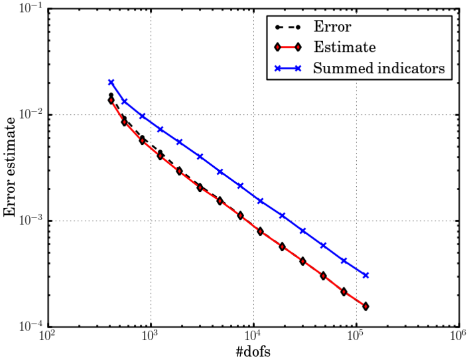
With same discretization as primal: $\eta_h = r(z_h) = 0$.

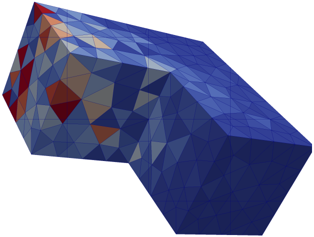
Suggested solution

Let $W_h \supset V_h$. Improve approximation by a patch-based least-squares curve fitting procedure:

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

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

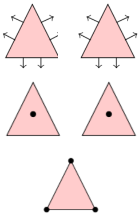


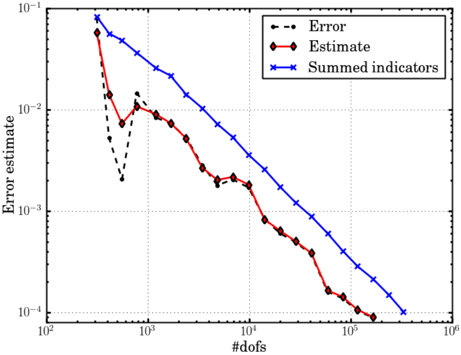


$$a(v, u) = \langle \nabla v, \nabla u \rangle,$$

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

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





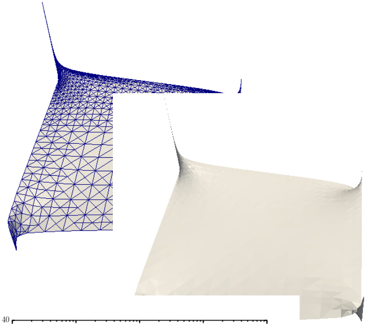
$$a((\tau, v, \eta), (\sigma, u, \gamma)) = \langle \tau, A\sigma \rangle + \langle \operatorname{div} \tau, u \rangle + \langle v, \operatorname{div} \sigma \rangle + \langle \tau, \gamma \rangle + \langle \eta, \sigma \rangle$$

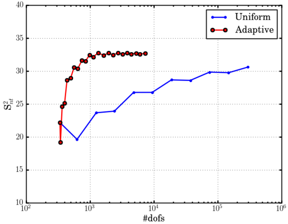
$$\mathcal{M}((\sigma, u, \eta)) = \int_\Gamma g\, \sigma \cdot n \cdot t \, \mathrm{d}s$$

Adaptivity pays off for the nonlinear gel problem

$$F(v;x) = \langle S(x), \nabla(v) \rangle, \quad S = \frac{\partial W}{\partial F}, \quad W = \phi_I \mu_E \left(\frac{1}{2}(\|F\|^2 - \|I\|^2) + \beta^{-1}((\det F)^{-\beta} - 1) \right) + (\det F)\,(a\phi \ln \phi + b(1-\phi)\ln(1-\phi) + c\phi(1-\phi) + c_{FH})$$

$$\mathcal{M}(x) = \int_\Gamma S(x)^2_{\mathrm{nt}} \mathrm{dX}$$






```
from dolfin import *

# Mesh and function space
mesh = UnitSquare(12, 12)
V = VectorFunctionSpace(mesh, "CG", 1)

# Deformation
x0 = Expression(("x[0]", "x[1]"))
x = interpolate(x0, V)

# Deformation gradient
F = grad(x)
F = variable(F)

# Volume fraction
phi_I = 0.8
phi = phi_I*inv(det(F))

# Elastic potential
W_E = 0.5*((inner(F, F) - 2) + (det(F)**(-2) - 1))

# Flory-Huggins potential
a = 4.28001624e-05; b = 0.0428001624; c = 0.010354878
W_FH = a*phi*ln(phi) + b*(1 - phi)*ln(1-phi) + c*phi*(1-phi)

# Total potential
scale = 1.e3; c_FH = 0.01338703463
W = scale*(phi_I*W_E + det(F)*(W_FH + c_FH))

# Define stress-tensor
S = diff(W, F)

# Define bilinear form
v = TestFunction(V)
B = inner(S, grad(v))*dx

# Define goal functional
M = S[0][1]*S[0][1]*ds(0)

# Define adaptive problem
pde = AdaptiveVariationalProblem(B, bcs=[...], M, u=x, ...)

# Solve problem
x = pde.solve(1.0)
```

Adaptivity pays off for the nonlinear gel problem