FEniCS Course

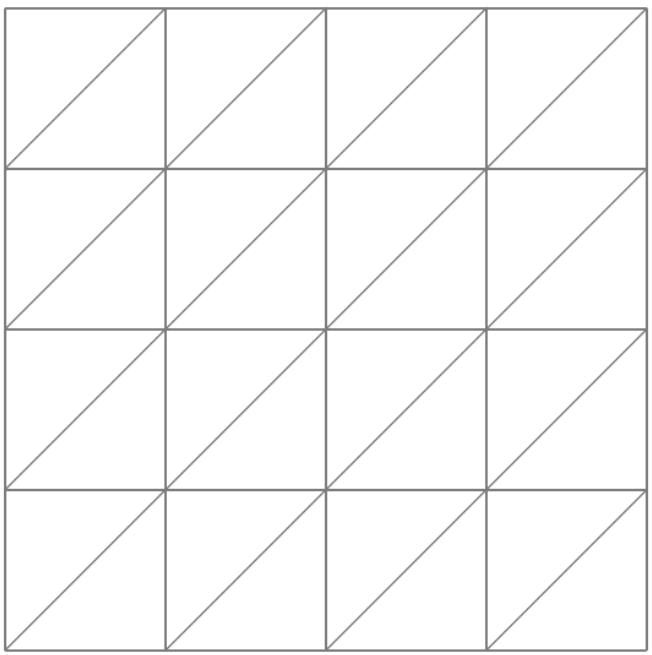
Lecture 11: A posteriori error estimates and adaptivity

Contributors
André Massing
Marie Rognes
Anders Logg

A priori estimates

If $u \in H^{k+1}(\Omega)$ and $V_h = P^k(\mathcal{T}_h)$ then

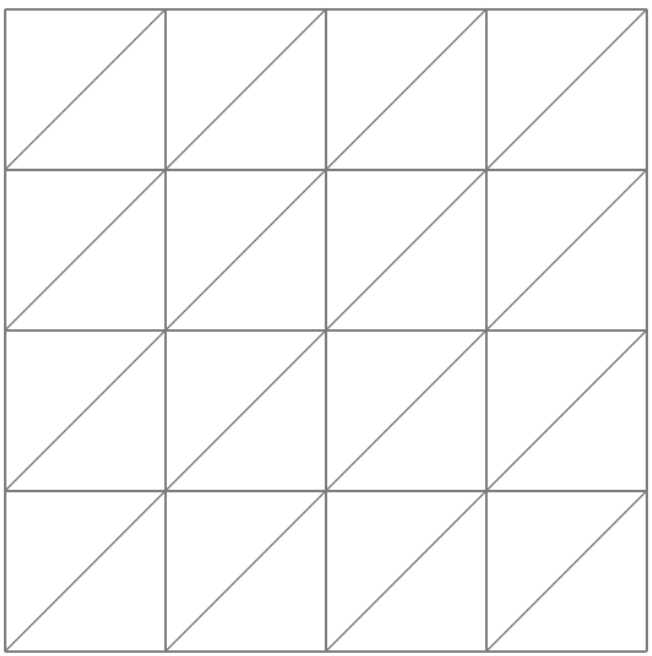
$$|||u - u_h||| \leqslant Ch^k \|u\|_{\Omega,k+1}$$
$$\|u - u_h\|_{L^2(\Omega)} \leqslant Ch^{k+1} \|u\|_{\Omega,k+1}$$

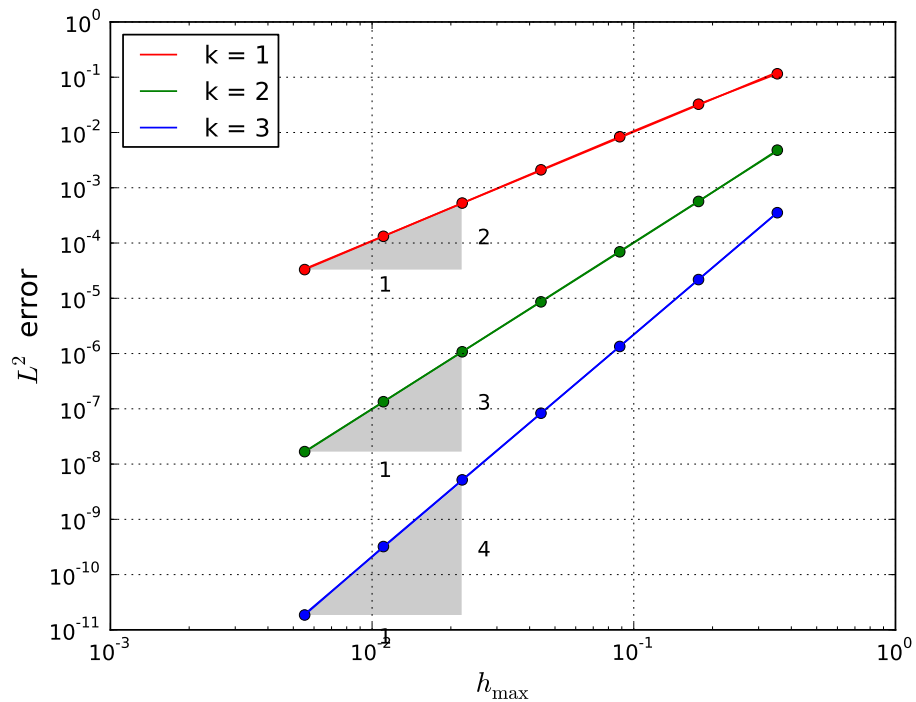


A priori estimates

If $u \in H^{k+1}(\Omega)$ and $V_h = P^k(\mathcal{T}_h)$ then

$$|||u - u_h||| \leqslant Ch^k \|u\|_{\Omega,k+1}$$
$$\|u - u_h\|_{L^2(\Omega)} \leqslant Ch^{k+1} \|u\|_{\Omega,k+1}$$

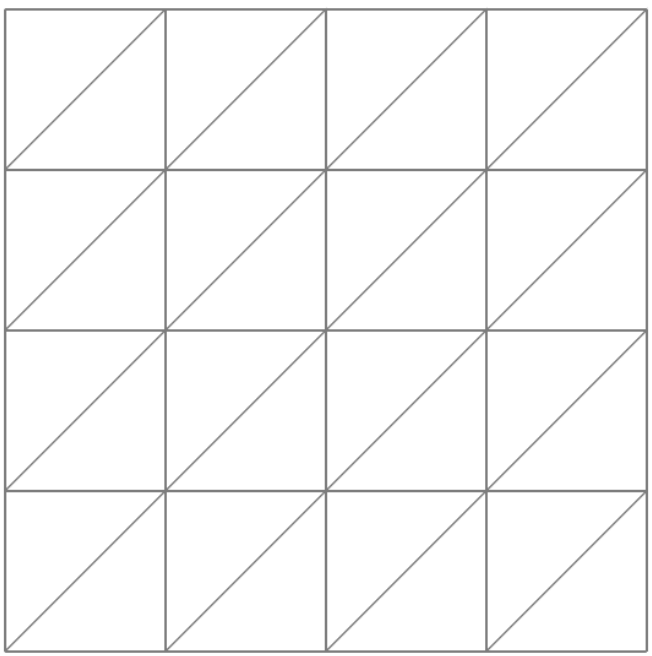


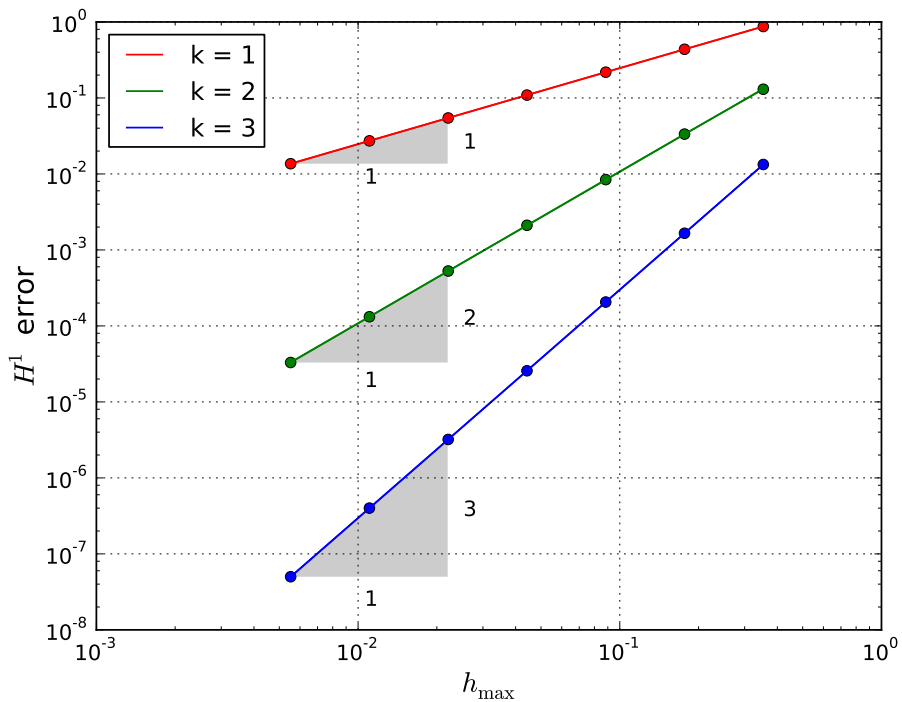


A priori estimates

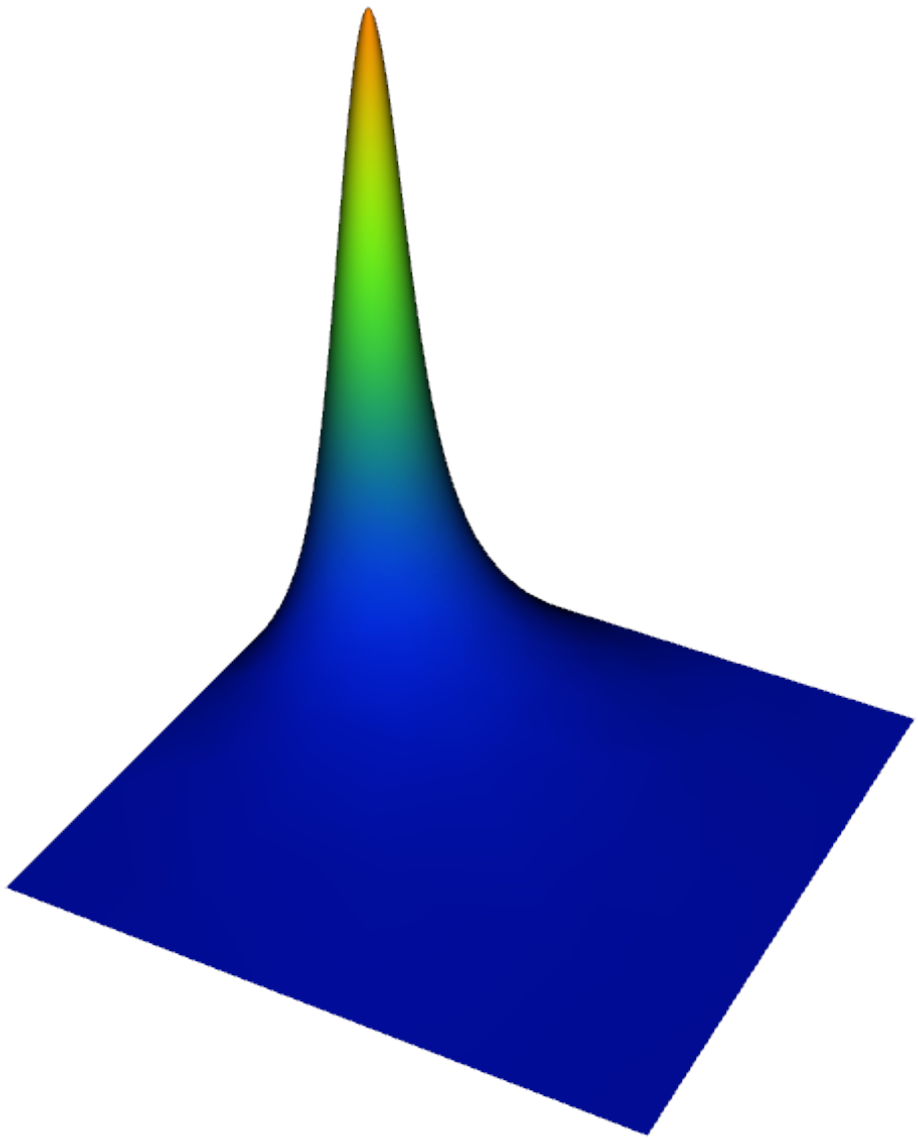
If $u \in H^{k+1}(\Omega)$ and $V_h = P^k(\mathcal{T}_h)$ then

$$|||u - u_h||| \leqslant Ch^k \|u\|_{\Omega,k+1}$$
$$\|u - u_h\|_{L^2(\Omega)} \leqslant Ch^{k+1} \|u\|_{\Omega,k+1}$$

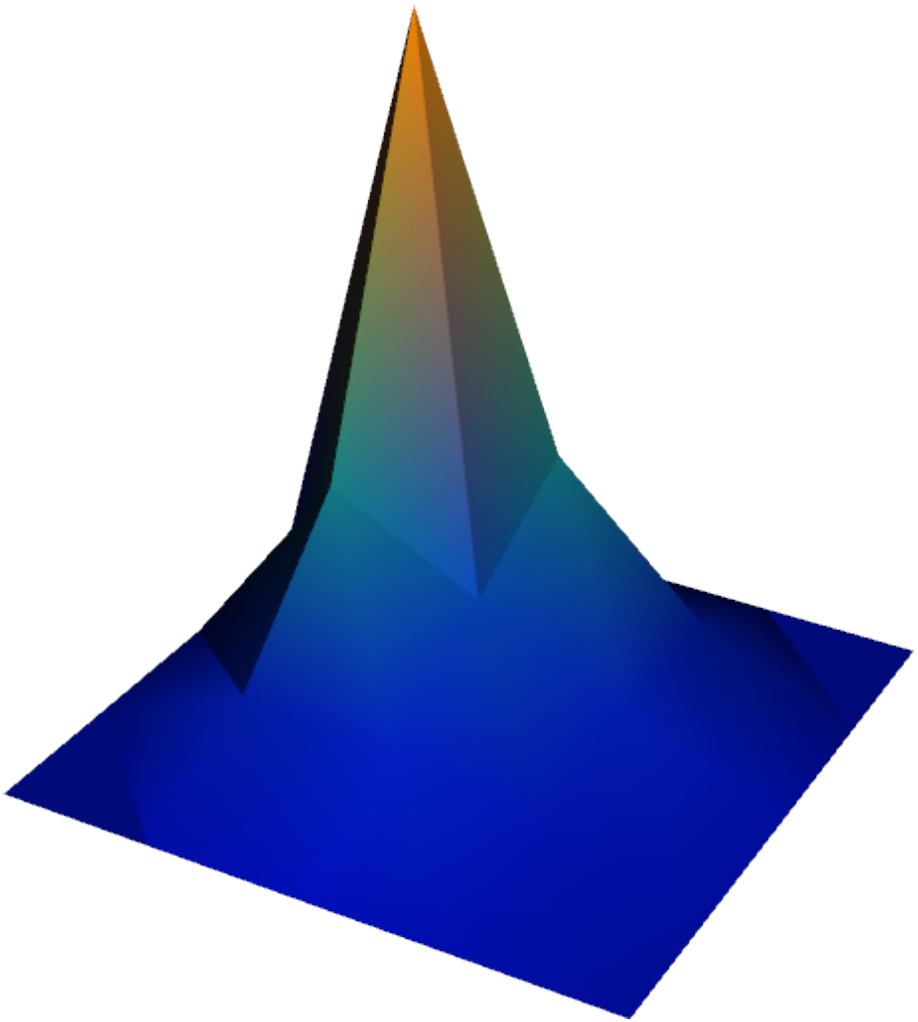


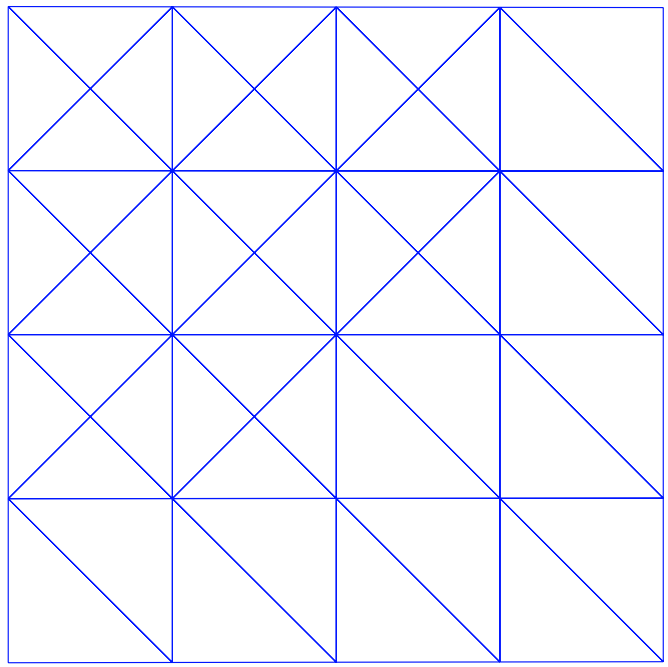


Mesh adaptation can yield more accurate results with less computational resources

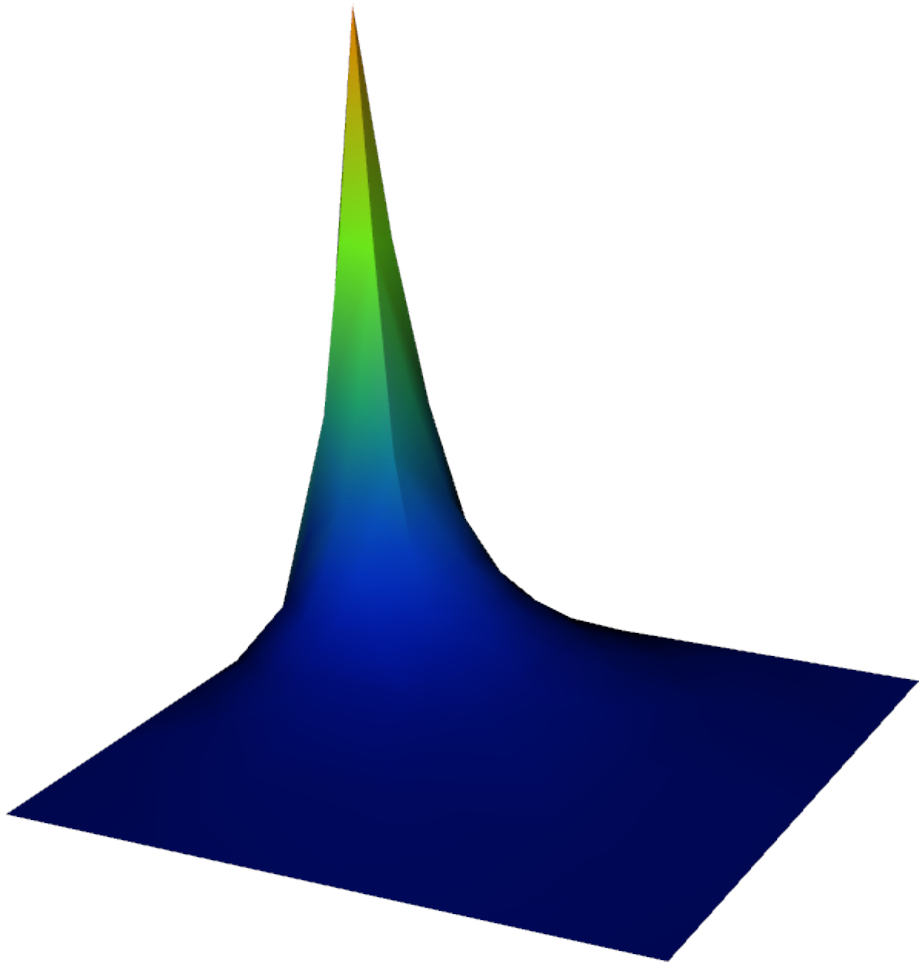


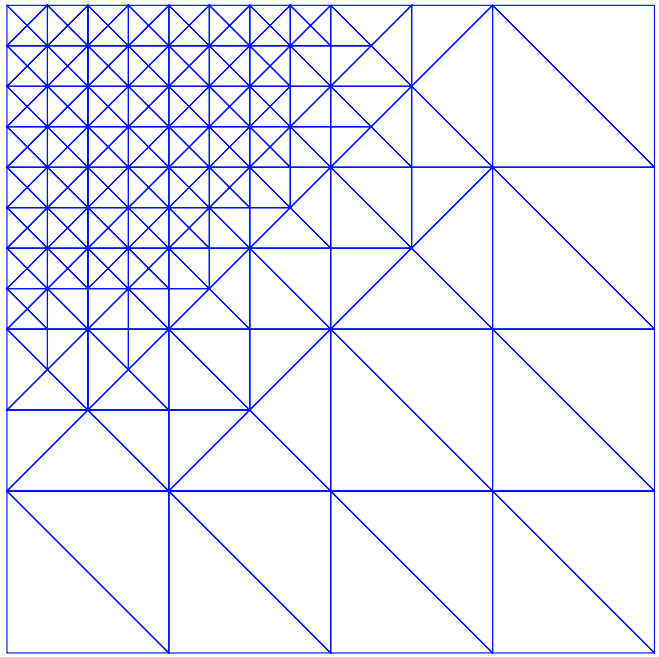
Mesh adaptation can yield more accurate results with less computational resources



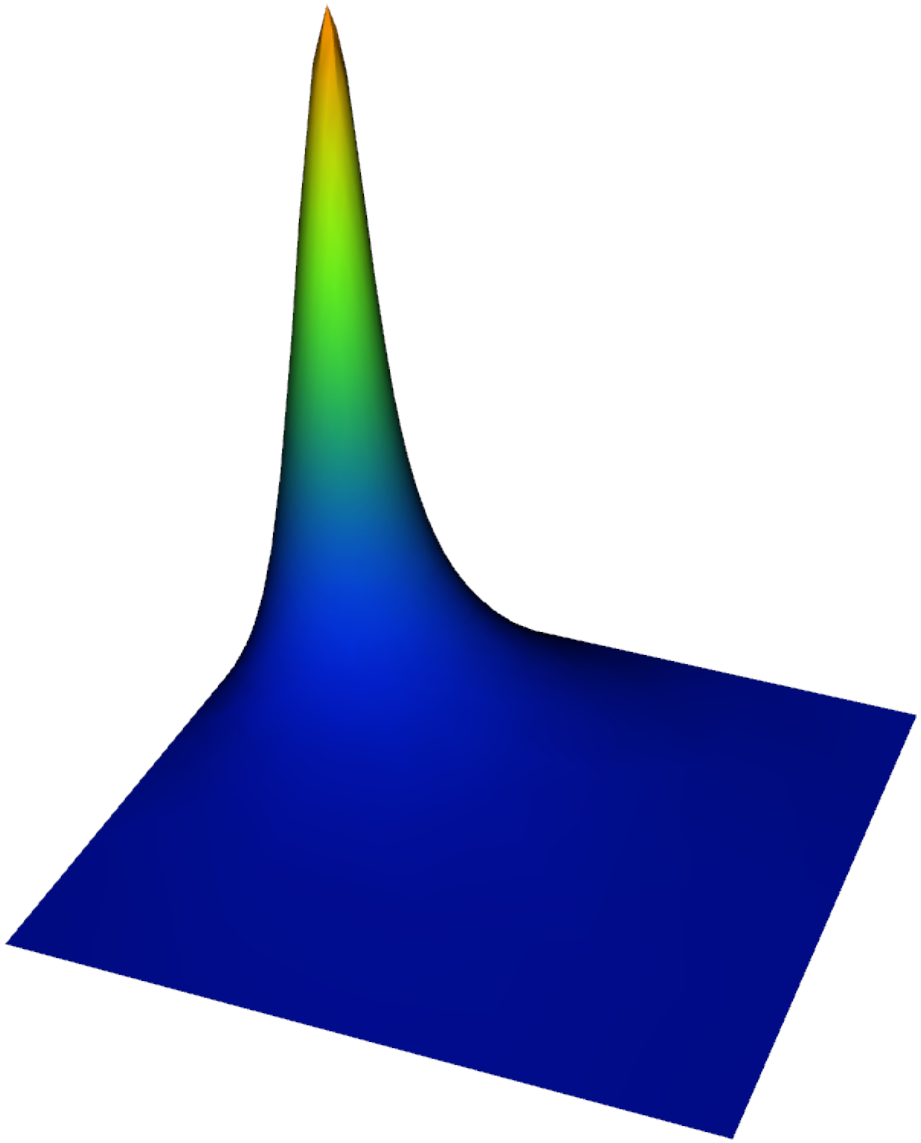


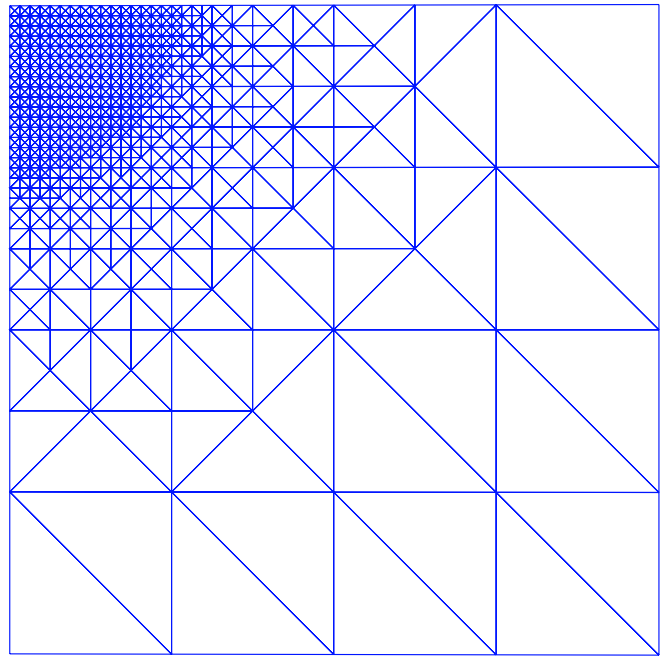
Mesh adaptation can yield more accurate results with less computational resources



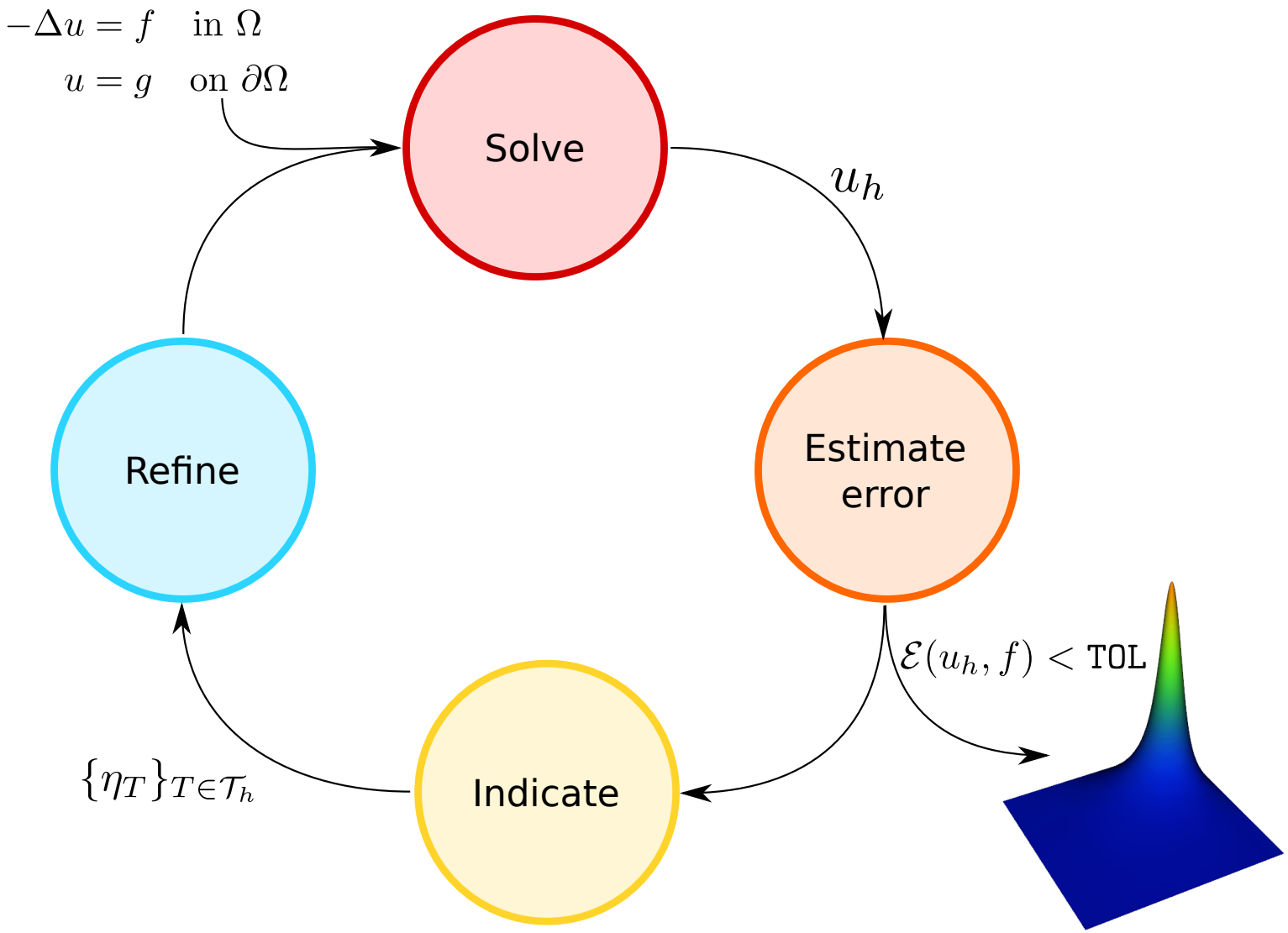


Mesh adaptation can yield more accurate results with less computational resources





Adaptive error control



Desired properties of error estimators

An error estimator $\mathcal{E} \sim \|u - u_h\|$ has to be

$$\text{computable} \quad \mathcal{E} = \mathcal{E}(u_h, f)$$

and should be

$$\text{reliable} \quad |||u - u_h||| \leqslant C\mathcal{E}(u_h, f)$$
$$\text{efficient} \quad c\mathcal{E}(u_h, f) \leqslant |||u - u_h|||$$
$$\text{local} \quad \mathcal{E}(u_h, f)^2 = \sum_{T \in \mathcal{T}_h} \rho_T(u_h, f)^2$$

The quality of $\mathcal{E}(u_h, f)$ is measured by the efficiency index η

$$\eta(\mathcal{E}(u_h, f)) = \frac{\|u - u_h\|}{\mathcal{E}(u_h, f)}$$

If $\eta(\mathcal{E}(u_h, f)) \to 1$ as $h \to 0$, the error estimator is asymptotically exact.

Types of error estimators

- Explicit residual-based error estimators
- Implicit error estimator based on local problems
- Gradient recovery estimators
- Hierarchic error estimators
- Goal-oriented error estimators

Explicit residual based error estimators

Model problem

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

Residual equation

$$R(u_h, f; v) = (\nabla(u - u_h), \nabla v) = (f, \nabla v) - (\nabla u_h, \nabla v) \quad \forall\, v \in H_0^1(\Omega)$$

Recall Galerkin orthogonality

$$R(u_h, f; v_h) = 0 \quad \forall\, v_h \in \widehat{V}_h$$

Interpolation operator $\pi_h : V \to V_h$

$$\begin{aligned} R(v) &= (f, v - \pi_h v) - (\nabla u_h, \nabla(v - \pi_h v)) \\ &= \sum_{T \in \mathcal{T}_h} (f + \Delta u_h, v - \pi_h v)_T - \sum_{T \in \mathcal{T}_h} (\nabla u_h \cdot n_T, v - \pi_h)_{\partial T} \\ &= \sum_{T \in \mathcal{T}_h} (f + \Delta u_h, v - \pi_h v)_T - \sum_{F \in \partial \mathcal{F}^i} (\llbracket \nabla u_h \cdot n_T \rrbracket, v - \pi_h)_F \end{aligned}$$

Explicit residual based error estimators

Starting from

$$R(v) = \sum_{T \in \mathcal{T}_h} (f + \Delta u_h, v - \pi_h v)_T - \sum_{F \in \partial \mathcal{F}^i} ([\![\nabla u_h \cdot n_T]\!], v - \pi_h)_F$$

and using the quasi-interpolant by Clement, which satisfies

$$\|v - \pi_h\|_{0,T} \leq C_1 h_T \|v\|_{\omega(T)}$$
$$\|v - \pi_h\|_{0,F} \leq C_2 h_F^{1/2} \|v\|_{\omega(F)},$$

one obtains

$$|R(v)| \leqslant C \|v\|_1 \left\{ \sum_{T \in \mathcal{T}_h} h_T^2 \|f + \Delta u_h\|^2 + \sum_{F \in \mathcal{F}^i} h_E \|[\![\nabla u_h]\!] \cdot n\|_F^2 \right\}^{1/2}$$

Explicit, residual based error estimators

Define

$$\begin{aligned}
&\text{Element residual} && r_T := f + \Delta|_T \\
&\text{Facet residual} && r_F := [\![\nabla u_h \cdot n]\!]|_T \\
&\text{Error indicators} && \rho_T^2 := h_T^2 \|r_T\|_T^2 + \frac{1}{2} \sum_{F \in \partial T} h_F \|r_F\|_T^2
\end{aligned}$$

Poincaré inequality gives $\|v\|_1 \sim \|\nabla v\|$ an thus

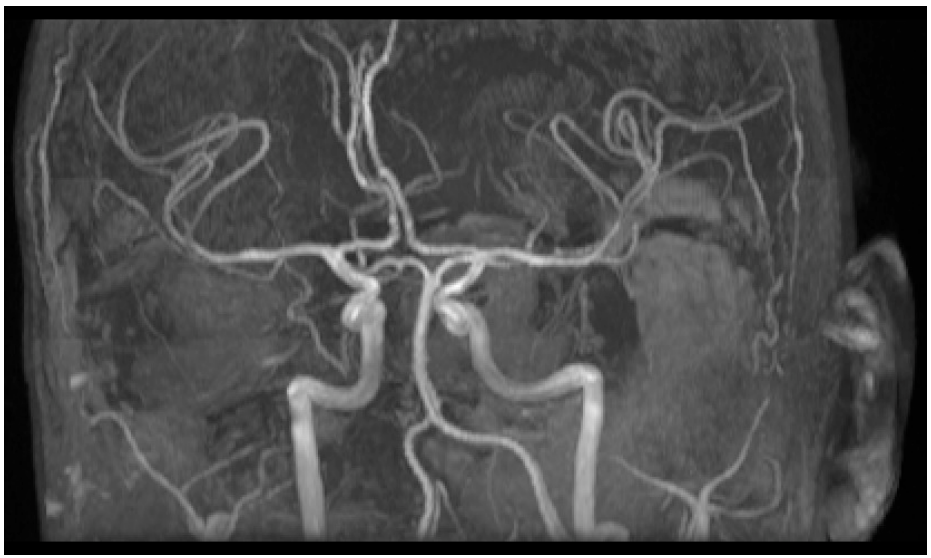
$$\begin{aligned}
\|u - u_h\|_1 &\leqslant \sup_{v \in V} \frac{(\nabla(u - u_h), \nabla v)}{\|v\|_1} = \sup_{v \in V} \frac{|R(v)|}{\|v\|_1} \\
&\leqslant C \Big(\sum_{T \in \mathcal{T}_h} \rho_T^2 \Big)^{1/2}
\end{aligned}$$

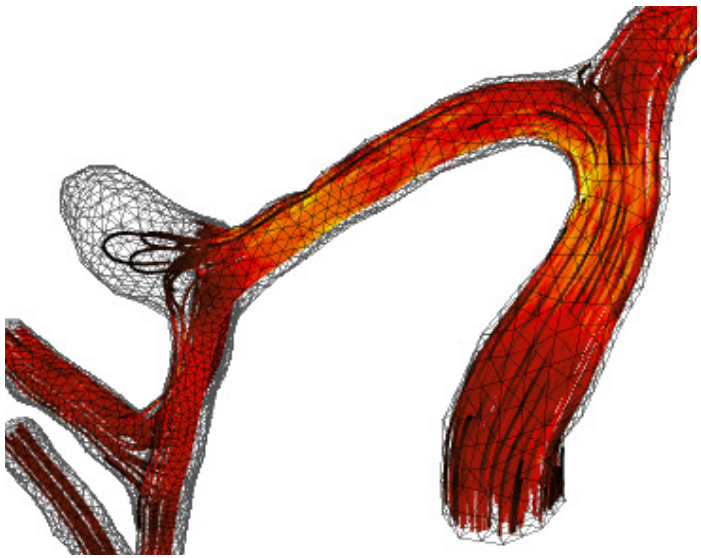
proving the reliability of the error estimator defined by $\{\rho_T\}_T$.

Goal-oriented error control

What is goal-oriented error control?

The scientist's viewpoint





Shear stress at vessel wall?

[Valen-Sendstad '08]

What is goal-oriented error control?

The mathematician's viewpoint

Input

- PDE: find $u \in V$ such that $a(u, v) = L(v) \quad \forall\, v \in V$
- 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

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

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

1. Recall residual $\qquad R(v) := L(v) - a(u_h, v)$
2. Introduce dual problem

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

3. Dual solution + residual $\implies$ error

$$\mathcal{M}(u) - \mathcal{M}(u_h) = \mathcal{M}(u - u_h) = a^*(z, u - u_h) = a(u - u_h, z)$$

$$= L(z) - a(u_h, z) = R(z) = R(z - z_h)$$

4. A good dual approximation $\tilde{z}_h$ gives computable error estimate $\qquad \eta_h = r(\tilde{z}_h)$
5. Error indicators ... ?

The dual-weighted residual method for the Poisson problem

Start with representation

$$|\mathcal{M}(u) - \mathcal{M}(u_h)| = |R(u - u_h, z - z_h)|.$$

Integrate by parts as for the classical energy-norm estimate gives

$$|\mathcal{M}(u) - \mathcal{M}(u_h)| \leqslant \sum_{T \in \mathcal{T}_h} \rho_T \omega_T,$$

where ρ_T is resembles the standard element residual

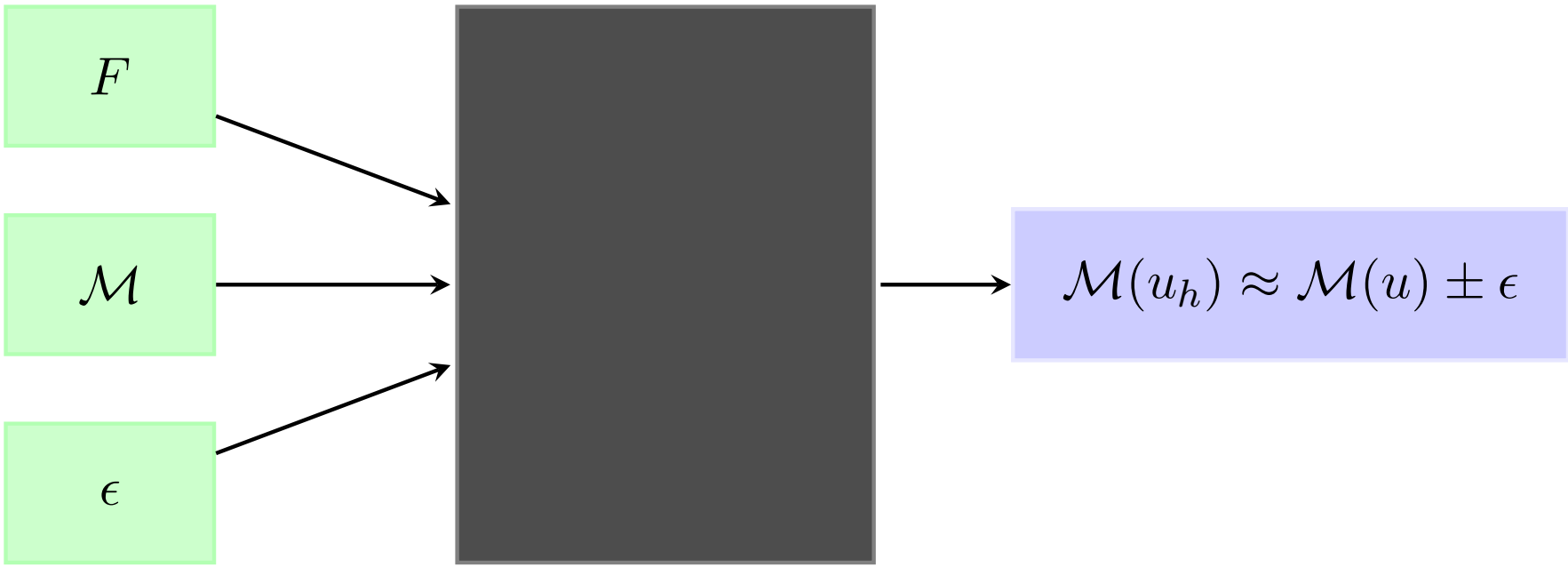
$$\rho_T = \|r_T\|_T + h_T^{1/2} \|r_F\|_{\partial T}$$

and ω_T is a weight

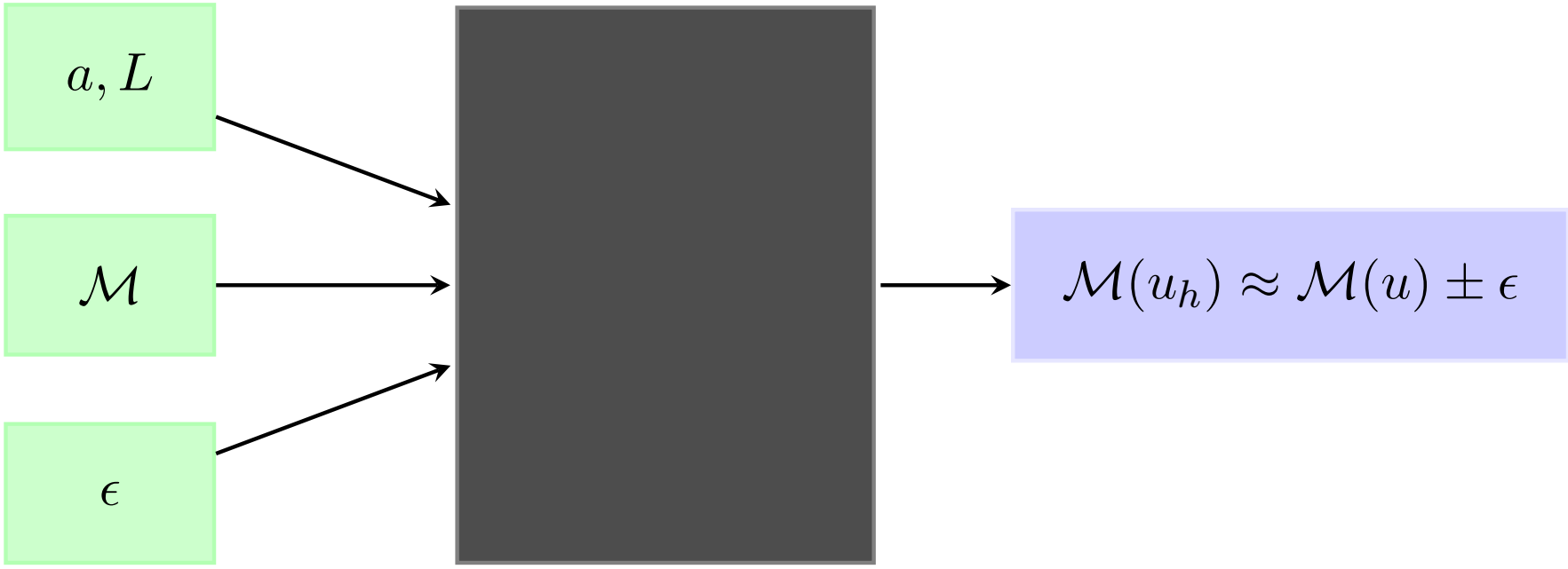
$$\omega_T = \|z - z_h\|_T + h_T^{1/2} \|z - z_h\|_{\partial T}$$

derived from the dual solution z.

What is automated goal-oriented error control?



What is automated goal-oriented error control?



FEniCS/DOLFIN

Python code

```
pde = VariationalProblem(a, L, bc)
pde.solve(u, tol, M)
```

Automated goal-oriented adaptivity – A complete example

Python code

```
from fenics import *


# Create mesh and define function space
mesh = UnitSquareMesh(8, 8)
V = FunctionSpace(mesh, "Lagrange", 1)


# Define boundary condition
u0 = Function(V)
bc = DirichletBC(V, u0, "x[0] < DOLFIN_EPS ||
    x[0] > 1.0 - DOLFIN_EPS")
```

Automated goal-oriented adaptivity – A complete example

Define variational problem:

Python code

```
u = TrialFunction(V)
v = TestFunction(V)
f = Expression("10*exp(-(pow(x[0] - 0.5, 2) +
    pow(x[1] - 0.5, 2)) / 0.02)",
                degree=1)
g = Expression("sin(5*x[0])", degree=1)
a = inner(grad(u), grad(v))*dx
L = f*v*dx + g*v*ds
```

Automated goal-oriented adaptivity – A complete example

Define function for the solution:

Python code

```
u = Function(V)
```

Define goal functional (quantity of interest) and tolerance:

Python code

```
M = u*dx

tol = 1.e-5
```

Automated goal-oriented adaptivity – A complete example

Solve equation a = L with respect to u and the given boundary conditions, such that the estimated error (measured in M) is less than tol

Python code

```
solver_parameters = {"error_control":
                     {"dual_variational_solver":
                      {"linear_solver": "cg"}}}
solve(a == L, u, bc, tol=tol, M=M,
    solver_parameters=solver_parameters)
```

Automated goal-oriented adaptivity – A complete example

Alternative, more verbose version (+ illustrating how to set parameters)

Python code

```
problem = LinearVariationalProblem(a, L, u, bc)
solver = AdaptiveLinearVariationalSolver(problem)
solver.parameters["error_control"]["dual_variational_solver"]["linear_sol
    = "cg"
solver.solve(tol, M)
```

Extract solutions on coarsest and finest mesh:

Python code

```
plot(u.root_node(), title="Solution on initial mesh")
plot(u.leaf_node(), title="Solution on final mesh")
interactive()
```

Useful FEniCS tools

Uniform mesh refinement:

Python code

```
mesh = refine(mesh)
```

Adaptive mesh refinement:

Python code

```
cell_markers = CellFunction("bool", mesh)
cell_markers.set_all(False)

for c in cells(mesh):
    if <something>:
        cell_markers[c] = True

mesh = refine(mesh, cell_markers)
```

The FEniCS challenge!

Compute the solution of the Poisson problem on the unit square with right-hand side $f = e^{-100(x^2+y^2)}$ and homogeneous Dirichlet boundary conditions.

Try to compute the solution as accurately as possible, using adaptive mesh refinement. Use any one of the techniques described in the lecture notes, or invent your own refinement strategy.

To check your answer, plot the L^2 norm as function of the refinement level. You should get an answer that approaches `8.888e-05`.

Hint: Refine in the lower left corner.

The student who computes the most accurate solution will be rewarded with an exclusive FEniCS surprise!