

# Exploring Automated Adaptivity and Error Control

Marie E. Rognes\* and Anders Logg\*,<sup>†</sup>

\*Center for Biomedical Computing at Simula Research Laboratory, P.O. Box 134, 1325 Lysaker.

<sup>†</sup>Department of Informatics, University of Oslo, Norway.

**Abstract.** Goal-oriented adaptive error control provides a multitude of choices with regard to error estimation and discretization refinement strategies. In this note, we investigate the effect of such choices using a flexible framework for automated adaptive goal-oriented error control applied to a specific test case. The efficiency of the adaptive algorithm varies significantly with the choice of parameters. We observe that the optimal parameters match the default parameters provided by the framework.

**Keywords:** adaptivity, automation, mesh refinement, error control, FEniCS

**PACS:** 02.60.Cb, 02.60.Lj, 02.70.Dh

## INTRODUCTION

Goal-oriented adaptivity and error control for finite element discretizations seeks to find an approximate solution  $u_h$  of a variational problem  $F(u; v) = 0$  such that the error measured in a certain quantity of interest  $\mathcal{M}$  satisfies a given tolerance  $\varepsilon$ :

$$|\mathcal{M}(u) - \mathcal{M}(u_h)| \leq \varepsilon.$$

*Automated* goal-oriented error control was recently added to the FEniCS Project [1, 2] (in DOLFIN 0.9.8) based on an automation [3] of the framework for goal-oriented error control presented in [4, 5]. In this note, we examine how to control and explore the strategies used by the adaptive algorithm. In particular, the flexibility of the framework allows for the evaluation, comparison, and optimization of different adaptive algorithms.

## ILLUSTRATING AUTOMATED GOAL-ORIENTED ERROR CONTROL

For illustration, we consider a stationary Navier–Stokes flow in a two-dimensional channel with an obstacle, driven by a boundary pressure. The computational domain  $\Omega$  is defined as  $\Omega = \Omega_C \setminus \Omega_O$ , where  $\Omega_C = (0, 4) \times (0, 1)$  and  $\Omega_O = (1.4, 1.6) \times (0, 0.5)$ . We define the Neumann (inflow/outflow) boundary  $\partial\Omega_N = \{(x, y) \in \partial\Omega : x = 0 \text{ or } x = 4\}$  and the Dirichlet (no-slip) boundary  $\partial\Omega_D = \partial\Omega \setminus \partial\Omega_N$ . A natural mixed variational formulation for the stationary Navier–Stokes equations reads: find  $(u, p) \in V$  such that  $F((u, p); (v, q)) = 0$  for all  $(v, q) \in V$ , where

$$F((u, p); (v, q)) = \nu \langle \text{grad } u, \text{grad } v \rangle + \langle \text{grad } u \cdot u, v \rangle - \langle p, \text{div } v \rangle + \langle \text{div } u, q \rangle + \langle \bar{p}n, v \rangle_{\partial\Omega_N}.$$

where  $V = \{H^1(\Omega; \mathbb{R}^2) : v|_{\partial\Omega_D} = 0\} \times L^2(\Omega)$ ,  $\langle \cdot, \cdot \rangle$  denotes the  $L^2(\Omega)$  inner product, and  $n$  is the boundary normal. Above,  $\nu$  is the (kinematic) viscosity and  $\bar{p}$  is a given boundary condition at the inflow/outflow boundary. For this specific test case, we let  $\nu = 0.02$  and take  $\bar{p} = 1$  at  $x = 0$  and  $\bar{p} = 0$  at  $x = 4$ . The quantity of interest  $\mathcal{M}$  is the outflux at  $x = 4$ ,

$$\mathcal{M}(u, p) = \int_{x=4} u \cdot n \, ds.$$

The system is stably discretized using the Taylor–Hood element; that is, the velocity and pressure spaces are discretized by continuous piecewise quadratic vector fields and continuous piecewise linears, respectively. The complete DOLFIN [2] code for solving this discrete variational problem adaptively and with control of the error is given below.

The adaptive solve, invoked by the final call to `pde.solve`, is based on a fully automated, iterative algorithm. First, the discrete solution  $(u_h, p_h)$  is computed on the initial mesh  $\mathcal{T}_h$ , and an a posteriori error estimate  $\eta_h$  of the approximation is automatically generated based on the given variational formulation  $F$  and the goal functional  $\mathcal{M}$ . If the approximation is deemed sufficiently accurate; that is, if  $\eta_h < \varepsilon$ , the process is stopped. If not, a set of error

indicators  $\{\eta_T\}_{T \in \mathcal{T}_h}$  is automatically computed, and the cells of the mesh are marked for refinement (or not) based on these. The mesh is then locally refined, and the algorithm repeats with the new mesh.

```

from dolfin import *

# Import mesh
mesh = Mesh("channel_with_flap.xml.gz")

# Define function spaces (Taylor-Hood)
W = VectorFunctionSpace(mesh, "CG", 2) * FunctionSpace(mesh, "CG", 1)

# Define unknown and test function(s)
(v, q) = TestFunctions(W)
w = Function(W)
(u, p) = (as_vector((w[0], w[1])), w[2])

# Prescribed force
p0 = Expression("(4.0 - x[0])/4.0")

# Define variational form
n = FacetNormal(mesh)
a = (0.02*inner(grad(v), grad(u)) + inner(v, grad(u)*u) - div(v)*p + q*div(u))*dx
L = - p0*dot(v, n)*ds

# Define boundary and boundary condition
def noslip(x, on_boundary):
    return (x[1] < DOLFIN_EPS or x[1] > 1.0 - DOLFIN_EPS) or \
        (on_boundary and abs(x[0] - 1.5) < 0.1 + DOLFIN_EPS)
bcs = [DirichletBC(W.sub(0), Constant((0.0, 0.0)), noslip)]

# Define goal
M = u[0]*ds(0)
class Outflow(SubDomain):
    def inside(self, x, on_boundary):
        return x[0] > 4.0 - DOLFIN_EPS

# Define adaptive variational problem
pde = AdaptiveVariationalProblem(a - L, bcs=bcs, goal_functional=M, u=w,
                                goal_exterior_domain=Outflow())

# Compute solutions to within given tolerance (1.e-3)
(u, p) = pde.solve(1.e-3).split()

```

## CONTROLLING THE ADAPTIVE ALGORITHM

A number of optional parameters may be specified to control the behavior of the adaptive algorithm, including the choice of error estimate and error indicators, the marking strategy, and the refinement fraction. Here, we will illustrate how to compare different error indicators and mesh marking strategies by prescribing parameters for the adaptive variational problem.

The generated a posteriori error estimate is based on a representation of the error in terms of residual contributions, automatically generated from the variational formulation and the approximate solution, and factors involving an appropriate dual solution:

$$|\mathcal{M}(u) - \mathcal{M}(u_h)| \approx r(w) = \sum_{T \in \mathcal{T}_h} \int_T R_T \cdot w \, dx + \int_{\partial T} R_{\partial T} \cdot w \, ds. \quad (1)$$

Here,  $R_T$  is a residual contribution from the interior of the cell  $T$  and  $R_{\partial T}$  is a residual contribution from the boundary of  $T$ . For the precise formulation and more details, see [3].

This representation leads to (at least) three choices for error indicators  $\{\eta_T\}_{T \in \mathcal{T}_h}$  and error estimates  $\eta_h$ :

$$\eta_T^{\text{er}} = \left| \int_T R_T \cdot w \, dx + \int_{\partial T} R_{\partial T} \cdot w \, ds \right|, \quad \eta_h^{\text{er}} = r(w) \quad (\text{error\_representation}) \quad (2)$$

$$\eta_T^{\text{dwr}} = \left| \int_T R_T \cdot w \, dx + \left[ \int_{\partial T} R_{\partial T} \cdot w \, ds \right] \right|, \quad \eta_h^{\text{dwr}} = \sum \eta_T^{\text{dwr}} \quad (\text{dual\_weighted\_residual}) \quad (3)$$

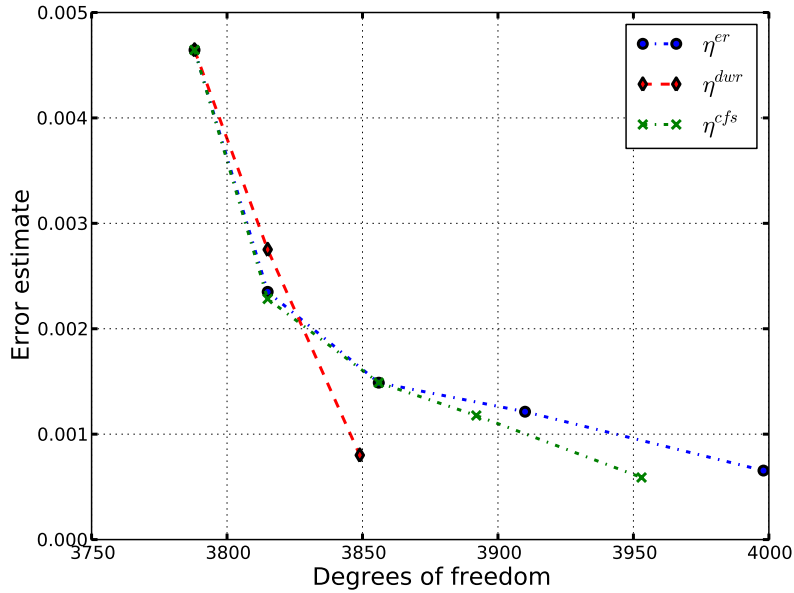
$$\eta_T^{\text{cfs}} = \left| \int_T R_T \cdot w \, dx \right| + \left| \left[ \int_{\partial T} R_{\partial T} \cdot w \, ds \right] \right|, \quad \eta_h^{\text{cfs}} = \sum \eta_T^{\text{cfs}} \quad (\text{cell\_facet\_split}) \quad (4)$$

where  $[\cdot]$  denotes an averaging of interior facet contributions, cf. [3].

The default error estimate is  $\eta_h^{\text{er}}$  and the default indicators are  $\{\eta_T^{\text{dwr}}\}$ , but the choice of error estimate and indicators can be adjusted by parameters of the adaptive variational problem. For instance, in order to use the error indicators labeled `cell_facet_split`, the following line of code could be added before the call to `pde.solve` in the above listed code.

```
pde.parameters["error_estimation"]["indicator"] = "cell_facet_split"
```

The choice of indicator plays a role for the adaptive algorithm, as is illustrated in Figure 1. We see that although all three indicators perform reasonably well, the use of the  $\eta^{\text{dwr}}$  indicators results in significantly less iterations than the others.



**FIGURE 1.** Error estimate versus number of degrees of freedom for adaptively refined meshes according to different error indicator strategies. Each data point corresponds to an iteration in the adaptive algorithm.

Currently, the adaptive variational framework includes four different mesh marking strategies. Each marking strategy takes as input a set of error indicators  $\{\eta_T\}_{T \in \mathcal{T}_h}$  and returns a set of markers; that is, a boolean value for each cell, indicating whether the cell is to be refined or not. For a parameter  $\alpha \in (0, 1]$ , the marking strategies are defined as follows:

**Dörfler [6]:** Sort  $\{\eta_T\}$  such that  $\eta_{T_i} \geq \eta_{T_{i+1}}$  for  $i = 1, \dots, |\mathcal{T}_h| - 1$ . Let  $m$  be the smallest integer such that  $\sum_{i=1}^m \eta_{T_i} \geq (1 - \alpha) \sum_{T \in \mathcal{T}_h} \eta_T$ . Mark  $\{\eta_{T_i}\}_{i=1}^m$ .

**Equidistribution:** Mark cell  $T$  if  $\eta_T > \frac{\alpha \varepsilon}{|\mathcal{T}_h|}$ .

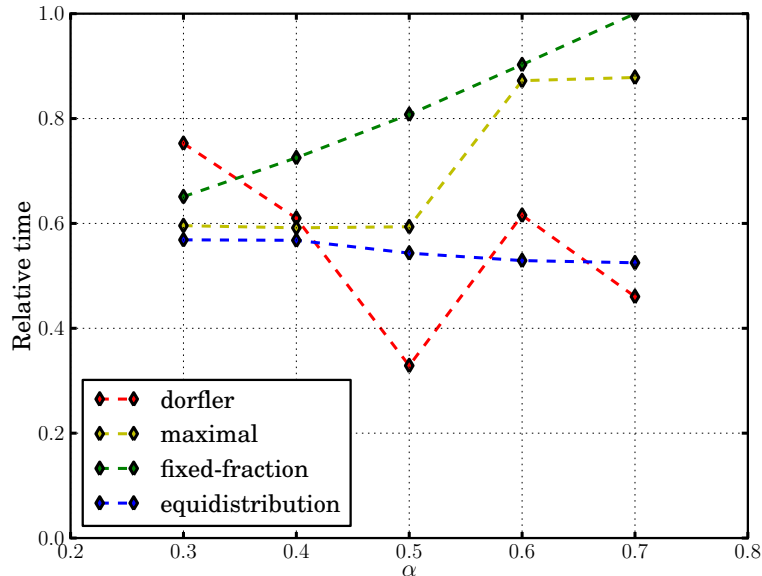
**Fixed-fraction:** Sort  $\{\eta_T\}$  such that  $\eta_{T_i} \geq \eta_{T_{i+1}}$  for  $i = 1, \dots, |\mathcal{T}_h| - 1$ . Let  $m$  be the smallest integer such that  $m \geq \alpha |\mathcal{T}_h|$ . Mark  $\{\eta_{T_i}\}_{i=1}^m$ .

**Maximal:** Let  $\eta_{max} = \max_T \eta_T$ . Mark cell  $T$  if  $\eta_T > \alpha \eta_{max}$ .

The default marking strategy is a Dörfler marking with a refinement fraction of  $\alpha = 0.5$ . However, the choice of marking strategy and fraction can easily be adjusted. For instance, the following lines result in the use of a maximal marking strategy with refinement fraction  $\alpha = 0.7$ .

```
pde.parameters["marking"]["strategy"] = "maximal"
pde.parameters["marking"]["fraction"] = 0.7
```

Thus, the computational efficiency of the different marking strategies can easily be compared, cf. Figure 2. We note that the default strategy, the Dörfler strategy with fraction  $\alpha = 0.5$ , performs the best in this test case.



**FIGURE 2.** Relative total computational time for the adaptive algorithm (relative to the longest time measured) versus refinement fraction  $\alpha$  for different mesh marking strategies.

In conclusion, we have illustrated how to solve a nonlinear stationary variational problem with fully automated adaptive goal-oriented error control, and how to adjust the parameters of the adaptive algorithm. Both the choice of error indicators and mesh marking strategy clearly affect the computational efficiency of the process.

## ACKNOWLEDGMENTS

This work is supported by an Outstanding Young Investigator grant from the Research Council of Norway, NFR 180450. This work is also supported by a Center of Excellence grant from the Research Council of Norway to the Center for Biomedical Computing at Simula Research Laboratory.

## REFERENCES

1. The FEniCS Project (2010), URL <http://www.fenics.org>.
2. A. Logg, and G. N. Wells, *ACM Transactions on Mathematical Software* **37**, 20:1–20:28 (2010).
3. M. E. Rognes, and A. Logg, *Submitted to journal* (2010).
4. K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Computational Differential Equations*, Cambridge University Press, 1996.
5. R. Becker, and R. Rannacher, *Acta Numerica* **10**, 1–102 (2001).
6. W. Dörfler, *SIAM Journal on Numerical Analysis* **33**, 1106–1124 (1996).