XFlow: A Solution-Adaptive Code

Krzysztof J. Fidkowski, *University of Michigan*

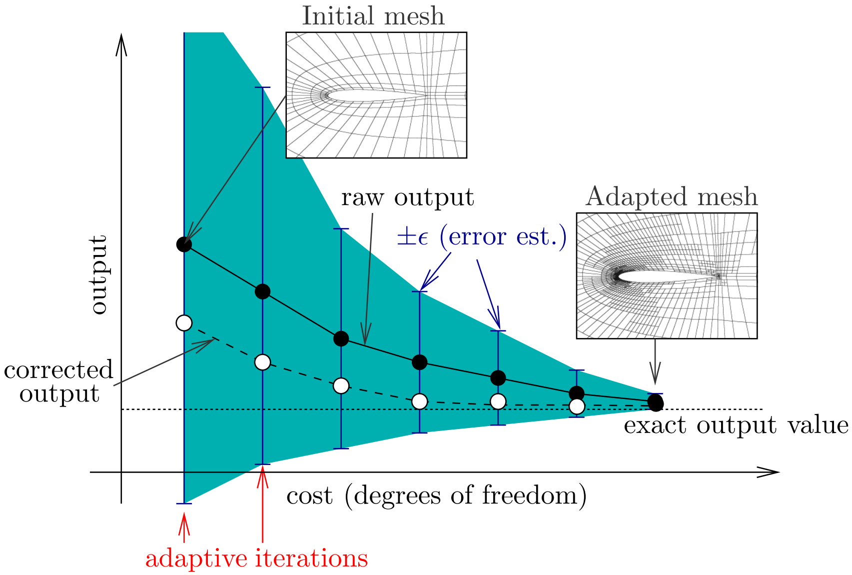
4^{th} International Workshop on High-Order CFD Methods
Crete, Greece

June 4-5, 2016

Code features

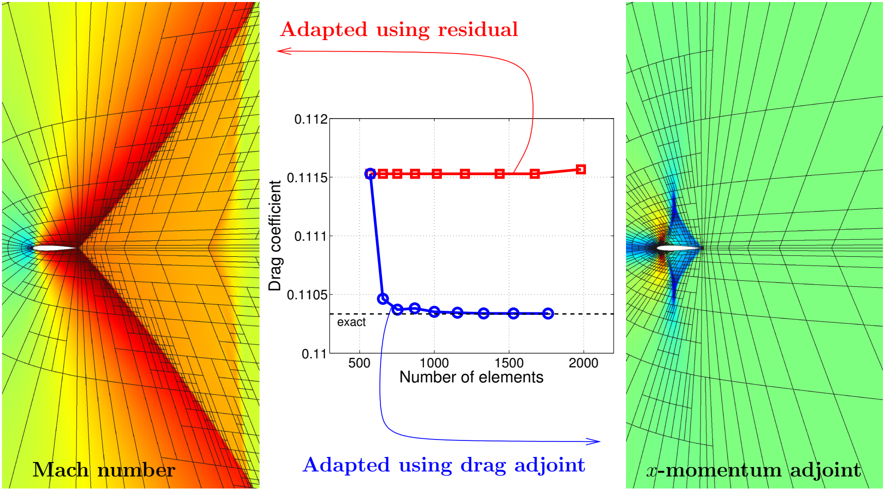
- DG and HDG discretizations
- C-code linked to ParMETIS, MPI
- Physics separate from numerics:
  - Compressible Navier-Stokes, RANS, shallow water, acoustics, scalar, radiation hydrodynamics
- Various time-stepping schemes:
  - RK, BDF, DIRK, (M)EBDF, SAMF, DG-in-time
- Fully-discrete and continuous-in-time adjoints for sensitivity studies and error estimation
- Structured and unstructured goal-oriented mesh and time-step adaptation

A typical output-adaptive result



Output-based adaptation is not always intuitive

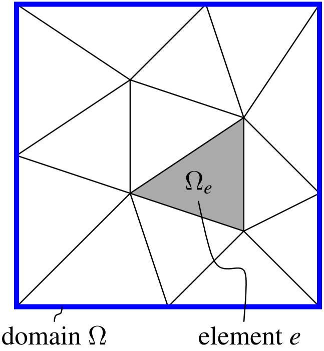
Fishtail shock in $M_\infty = 0.95$ inviscid flow over a NACA 0012 airfoil

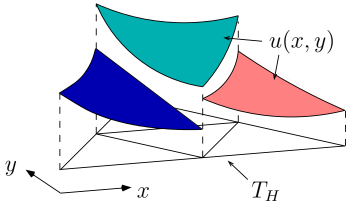


The discontinuous Galerkin method

- State vector: $\mathbf{u} = [\rho, \rho u_i, \rho E, \rho\tilde{\nu}]^T$
- PDE: $\partial_t \mathbf{u} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}, \nabla\mathbf{u}) + \mathbf{S}(\mathbf{u}, \nabla\mathbf{u}) = \mathbf{0}$
- Solution approximation on element e: $\mathbf{u}_h(\vec{x})\Big|_e \approx \sum_{j=1}^{n(p)} \mathbf{U}_{ej}\phi_j(\vec{x})$

$$\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s, \quad \mathcal{V}_h = \left\{ u \in L^2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p(\Omega_e) \; \forall\, \Omega_e \in T_h \right\}$$





N_e = # of elements
$n(p)$ = # of basis fcns
p = solution approximation order
$\phi_j(\vec{x})$ = j^{th} basis function

Nonlinear solver

- Newton-Raphson + pseudo-time continuation
- Linear system at each nonlinear iteration:

$$\left(\frac{\mathbf{M}}{\Delta t_a} + \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\bigg|_{\mathbf{U}_0} \right) \Delta \mathbf{U} + \mathbf{R}(\mathbf{U}_0) \quad = \quad \mathbf{0},$$

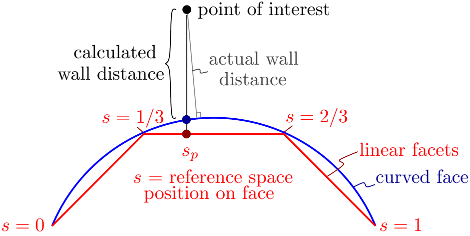
  $\mathbf{U}_0$ = initial guess, $\mathbf{M}$ = mass matrix,
- Δt_a is an artificial time step,

$$\Delta t_a = \mathrm{CFL}\, h / c_{\max}$$

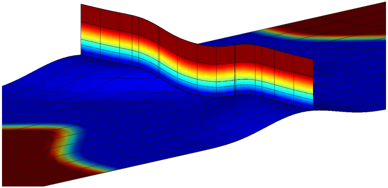
  $h = \text{volume}/(\text{surface area})$, $c_{\max} =$ max characteristic speed over quadrature points of the element
- State update is under-relaxed, $\mathbf{U} = \mathbf{U}_0 + \omega \Delta \mathbf{U}$, to keep it physical, via a line search

Wall distance calculation

- SA model requires d = distance to closest wall
- Store d via order p_{wd} approximation on each element
- Compute d at each order p_{wd} Lagrange node via brute force search to identify closest face, projection to faceted face representation, and snapping to the true geometry



calculation on curved elements

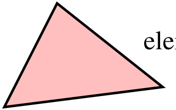


contours of wall distance

Output sensitivity to residuals: the adjoint

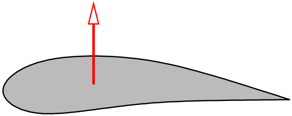
The lift adjoint $\mathbf{\Psi}$ is the sensitivity of lift to residual sources.

We have a solution $\mathbf{U}$ when $\mathbf{R} = 0$



element e

Lift= $J(\mathbf{U})$

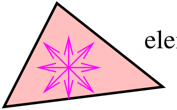


state $\mathbf{U}$

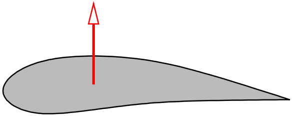
Output sensitivity to residuals: the adjoint

The lift adjoint $\boldsymbol{\Psi}$ is the sensitivity of lift to residual sources.

We have a solution $\mathbf{U}$ when $\mathbf{R} = 0$

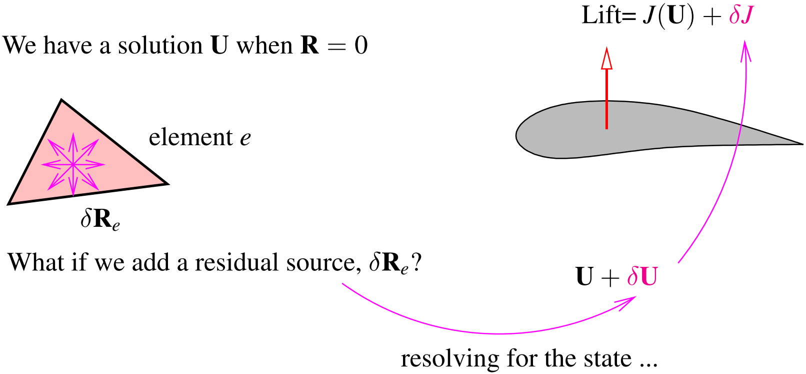


What if we add a residual source, $\delta\mathbf{R}_e$?



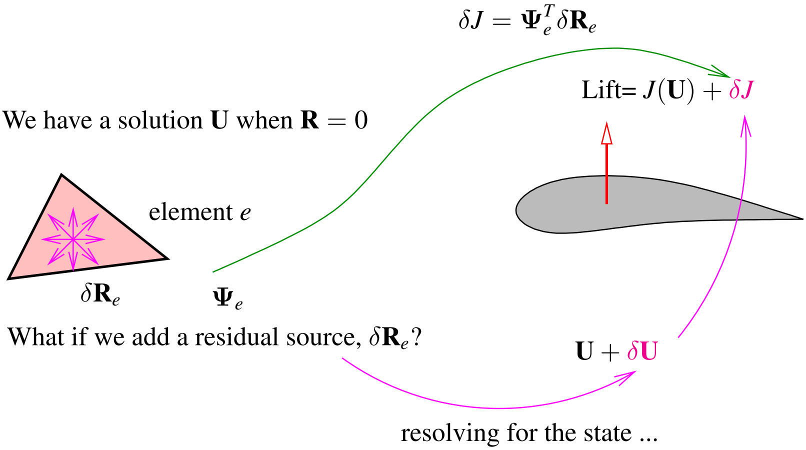
Output sensitivity to residuals: the adjoint

The lift adjoint Ψ is the sensitivity of lift to residual sources.



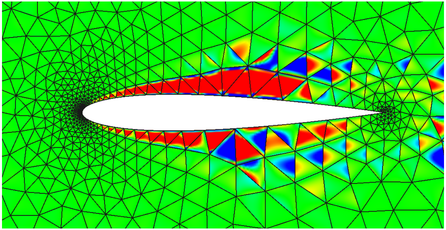
Output sensitivity to residuals: the adjoint

The lift adjoint $\mathbf{\Psi}$ is the sensitivity of lift to residual sources.

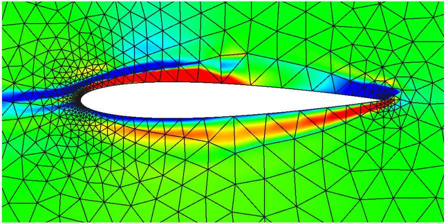


Adjoint-weighted residual as an error indicator

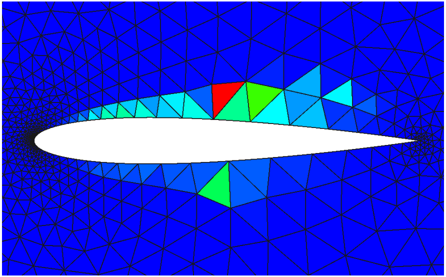
Fine space residual, $\mathbf{R}_h(\mathbf{U}_h^H)$



Fine space adjoint, $\mathbf{\Psi}_h$



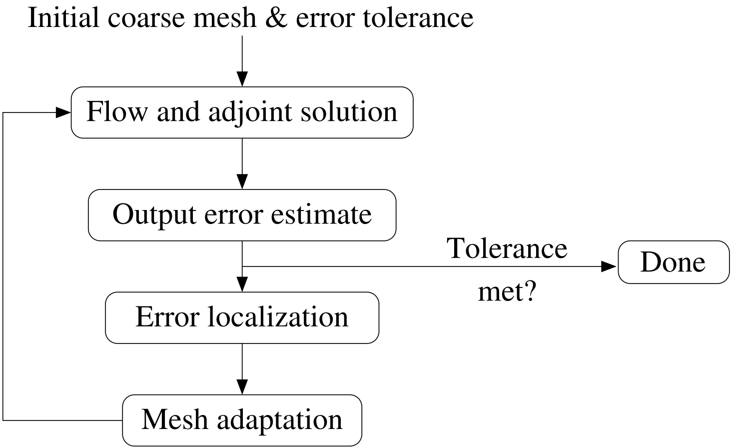
Error indicator, $\epsilon_e = |\mathbf{\Psi}_{h,e}^T \mathbf{R}_{h,e}(\mathbf{U}_h^H)|$



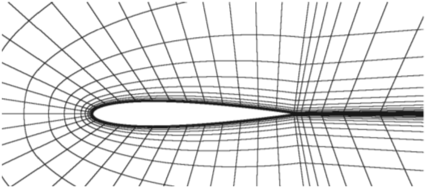
Output error: $\delta J \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)$

Idea: adapt where ϵ_e is high, to reduce the residual there

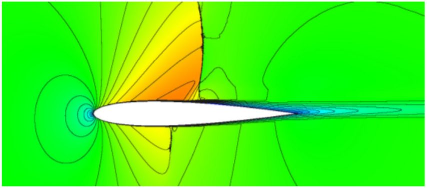
Mesh adaptation



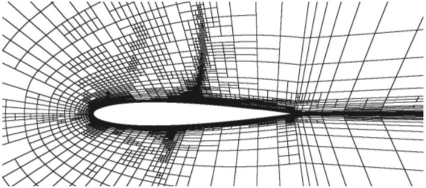
Transonic RANS flow over a NACA 0012



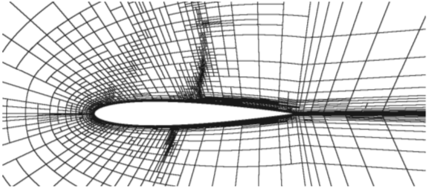
23.1: Initial mesh (1740 elements)



23.2: Mach number contours

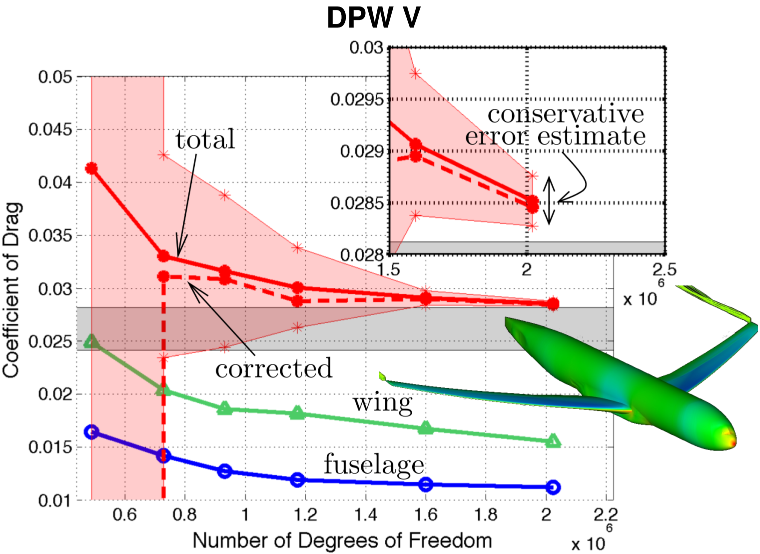


23.3: 6^{th} adapted mesh, isotropic (8,736 elements)



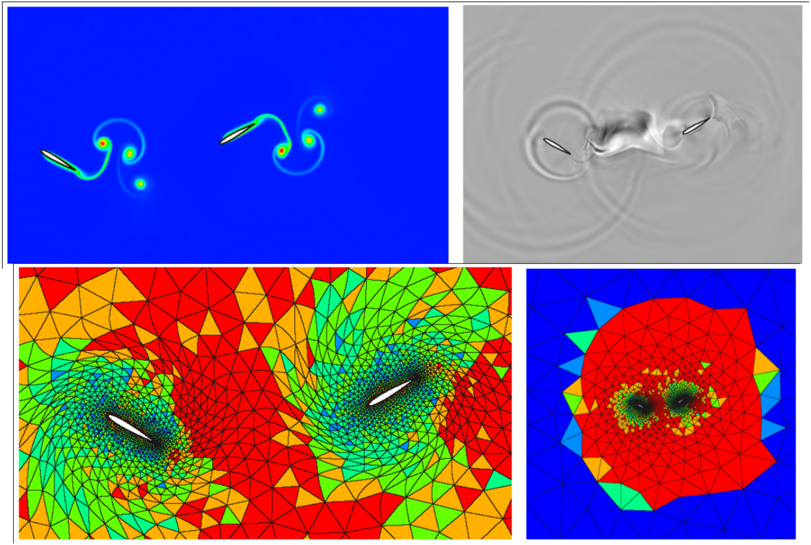
23.4: 10^{th} adapted mesh, anisotropic (4,816 elements)

h/p adaptive runs

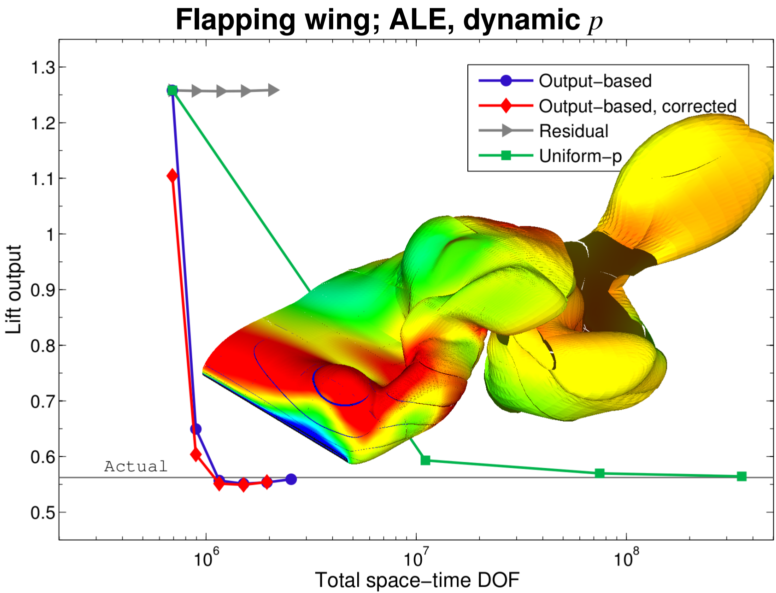


h/p adaptive runs

Staggered pitching/plunging airfoils; ALE, dynamic p



h/p adaptive runs



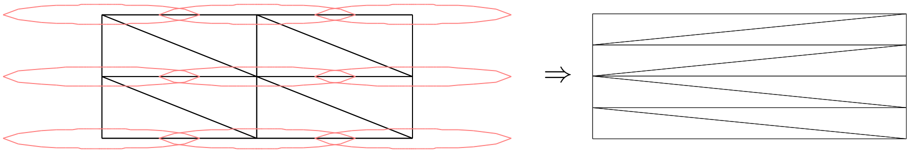
Mesh-conforming mesh generation

Idea

Make mesh in which each edge has the same metric length

$$\text{metric distance from } A \text{ to } B\text{: } \ell_{AB} = \int_A^B d\ell = \int_A^B \sqrt{d\vec{x}^T \mathcal{M} d\vec{x}}$$

- e.g. BAMG = Bi-dimensional Anisotropic Mesh Generator [1: Borouchaki, 1995]
- Input: background mesh and desired metric at nodes
- Output: metric-conforming mesh

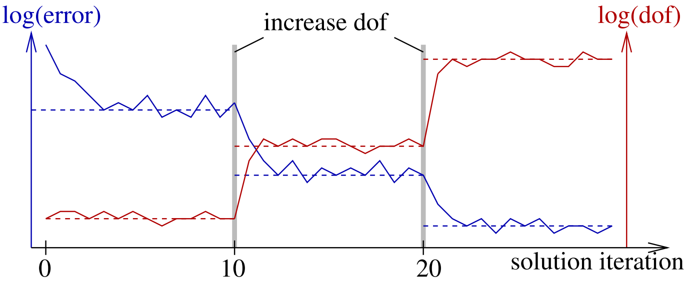


A mesh optimization algorithm [3: Yano, 2012]

- Given: current mesh, primal and adjoint solutions
- Determine: metric step matrix, $\mathcal{S}_v$, at each mesh vertex, v, that produces a mesh with the smallest output error at a fixed solution cost
- Key ingredients
  1. Error convergence model: $\mathcal{S}_v \rightarrow$ output error
  2. Cost model: $\mathcal{S}_v \rightarrow$ solution cost
  3. Iterative algorithm that equidistributes the marginal error-to-cost ratio
- Expect multiple iterations of optimization until error "bottoms out" at a fixed cost; can then increase allowable cost to further reduce error

Combining adaptation and optimization

1. Start with a coarse mesh at a certain cost = `dof`
2. Run multiple (~ 10) mesh optimization iterations at fixed cost
   - Each iteration requires primal and adjoint solves
   - Solves are quick since starting from good initial guesses
   - Error will drop, then stagnate/oscillate
   - Use results from final run or average of last few runs

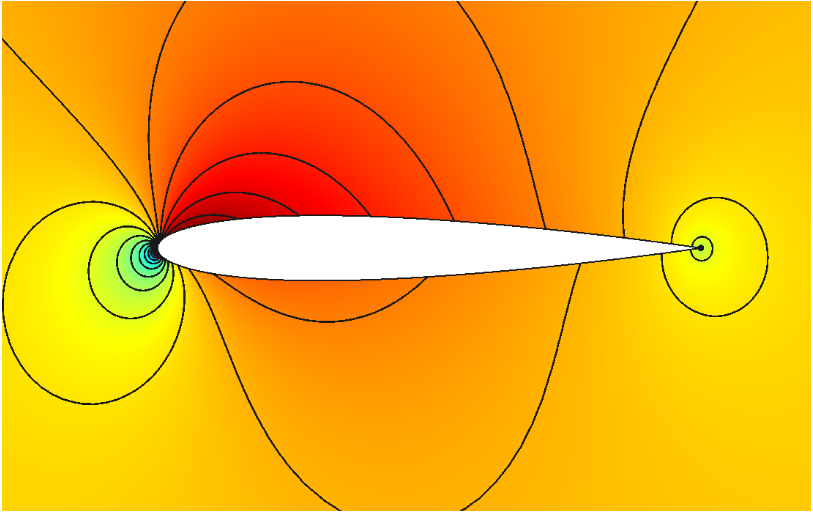


3. Increase `dof` cost by a prescribed factor if need more accuracy and can afford more cost; return to step 2

Example: NACA 0012 in inviscid flow

Euler equations, $M_\infty = 0.5, \alpha = 2^\circ, \gamma = 1.4$, output = drag

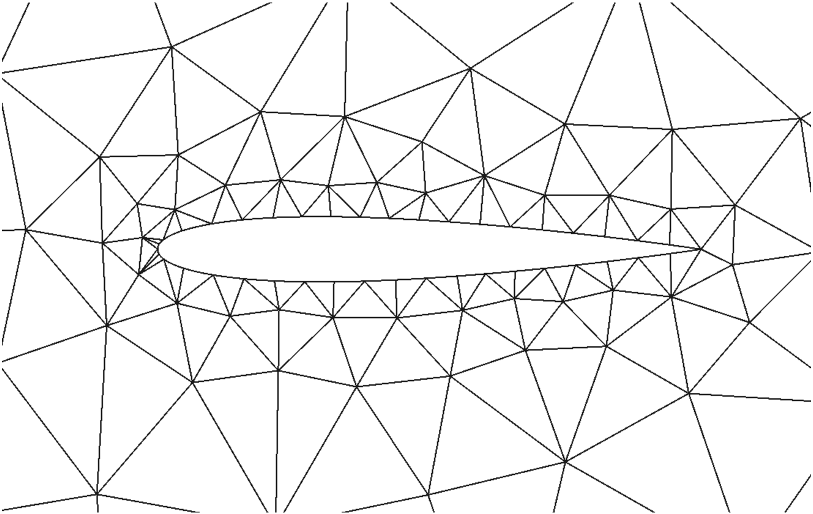
Mach number contours



Example: NACA 0012 in inviscid flow

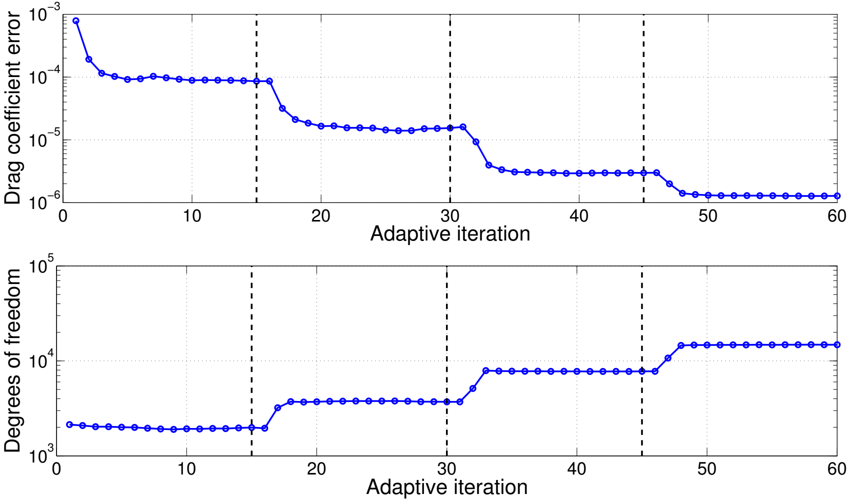
Euler equations, $M_\infty = 0.5, \alpha = 2^\circ, \gamma = 1.4$, output = drag

Initial mesh: 356 triangles, farfield @$2000c$



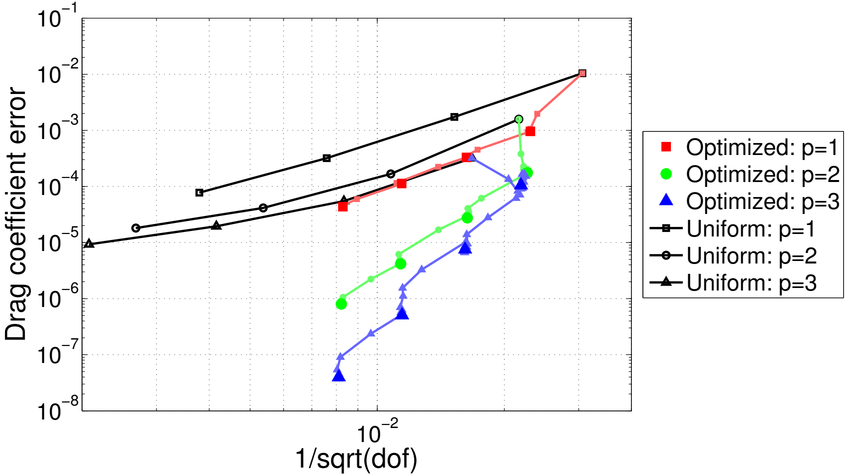
NACA 0012 in inviscid flow: sample run

$p = 2$, 15 optimization iterations at each `dof`

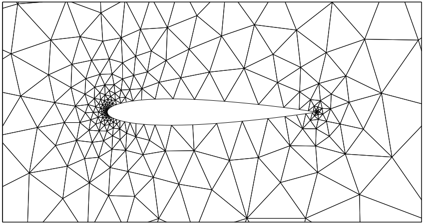


NACA 0012 in inviscid flow: output convergence

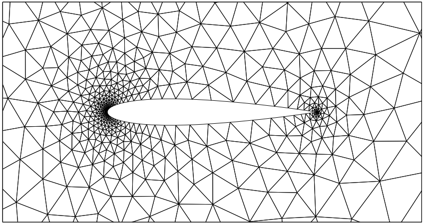
Compare to uniform refinement at different orders p



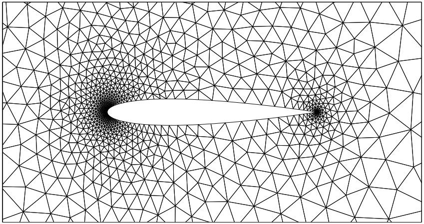
NACA 0012 in inviscid flow: optimized meshes



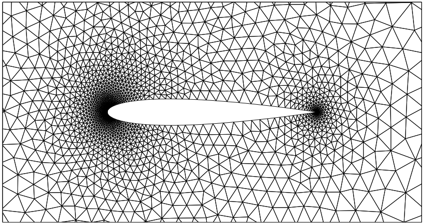
$p = 1$, dof = 2000



$p = 1$, dof = 4000

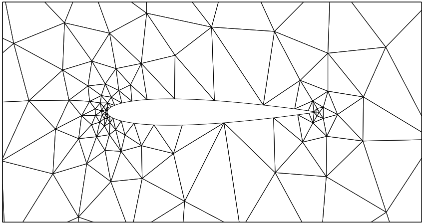


$p = 1$, dof = 8000

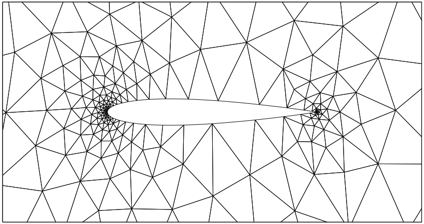


$p = 1$, dof = 16000

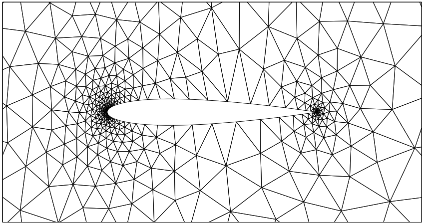
NACA 0012 in inviscid flow: optimized meshes



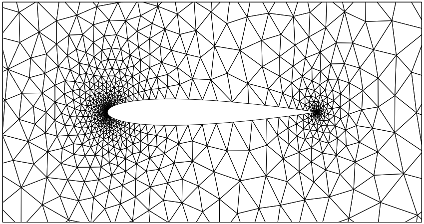
$p = 2$, dof = 2000



$p = 2$, dof = 4000

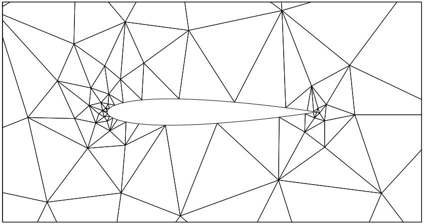


$p = 2$, dof = 8000

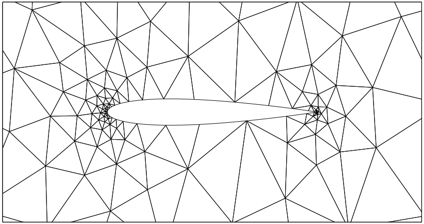


$p = 2$, dof = 16000

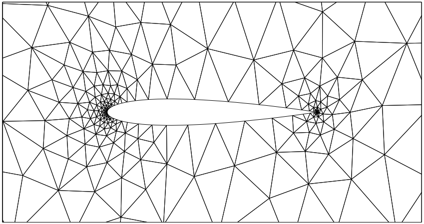
NACA 0012 in inviscid flow: optimized meshes



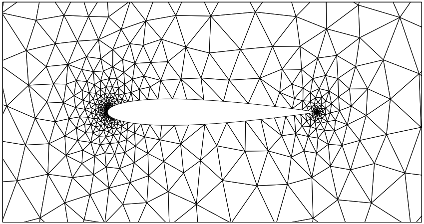
$p = 3$, dof = 2000



$p = 3$, dof = 4000



$p = 3$, dof = 8000

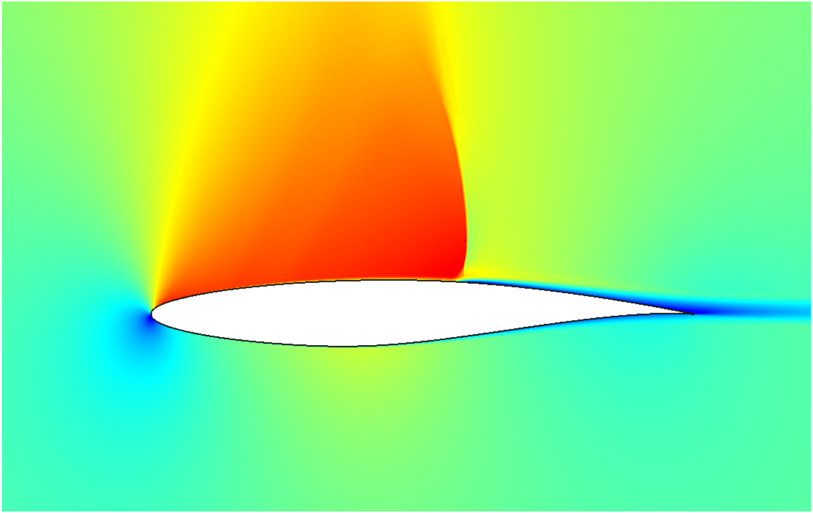


$p = 3$, dof = 16000

Example: RAE 2822 in transonic flow

RANS-SA, $M_\infty = 0.73, \alpha = 2.79°, Re = 6.5M$, output = drag

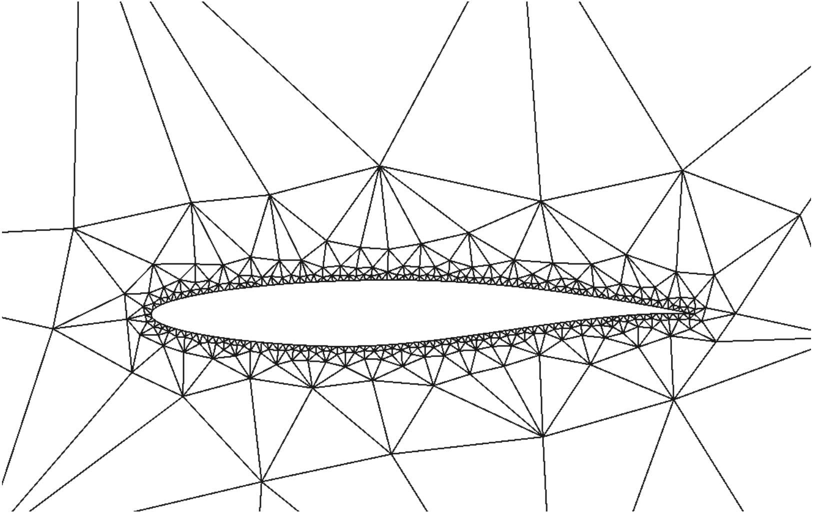
Mach number contours



Example: RAE 2822 in transonic flow

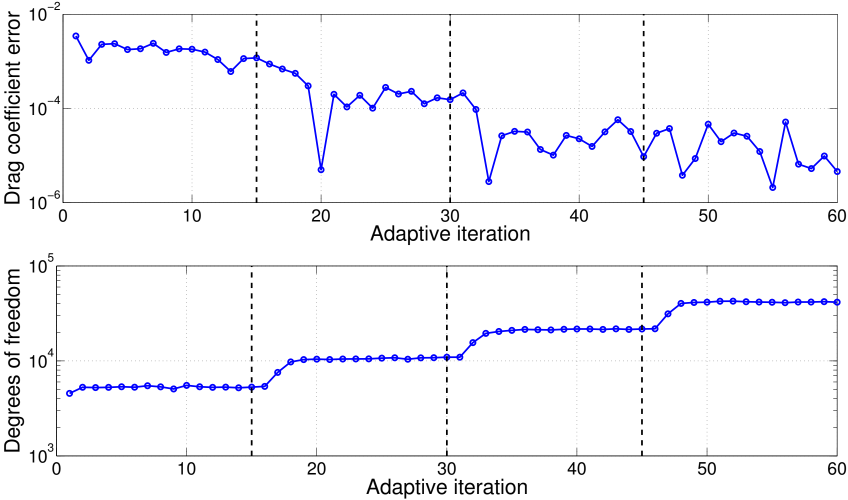
RANS-SA, $M_\infty = 0.73, \alpha = 2.79^\circ, Re = 6.5M$, output = drag

Initial mesh: 758 triangles, farfield @$2000c$



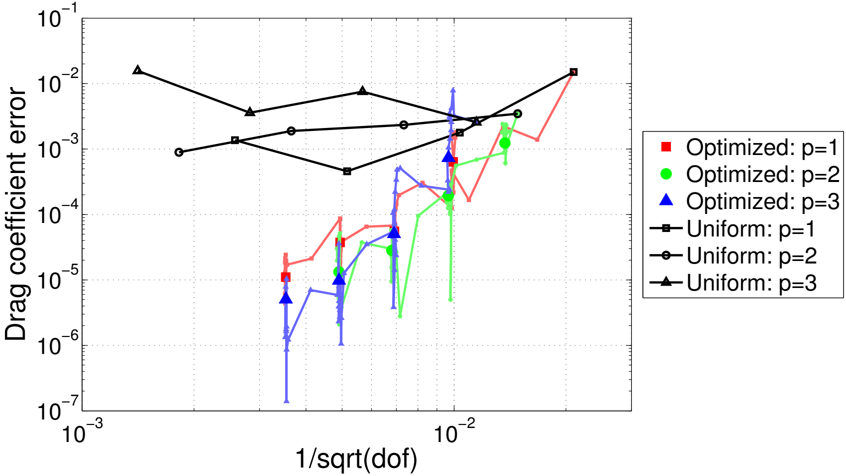
RAE 2822 in transonic flow: sample run

$p = 2$, 15 optimization iterations at each `dof`

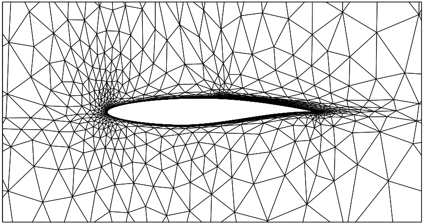


RAE 2822 in transonic flow: output convergence

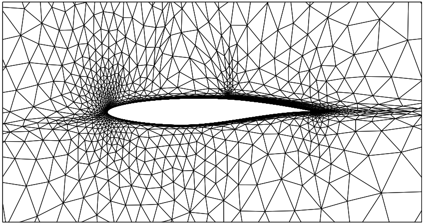
Compare to uniform refinement at different orders p



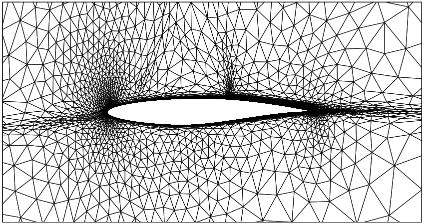
RAE 2822 in transonic flow: optimized meshes



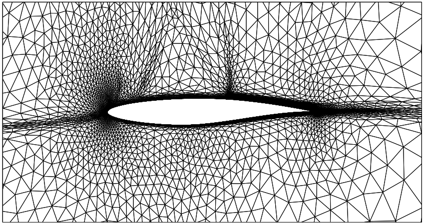
$p = 1$, `dof` = 5000



$p = 1$, `dof` = 10000

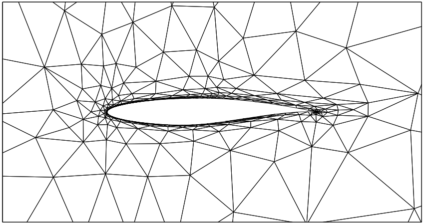


$p = 1$, `dof` = 20000

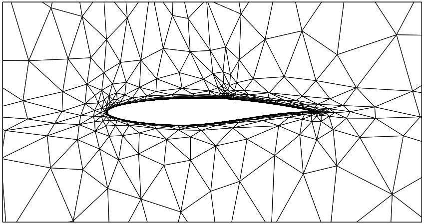


$p = 1$, `dof` = 40000

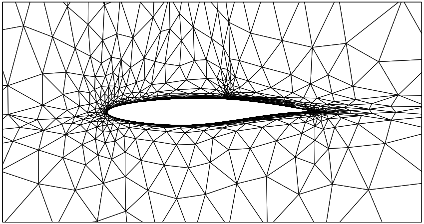
RAE 2822 in transonic flow: optimized meshes



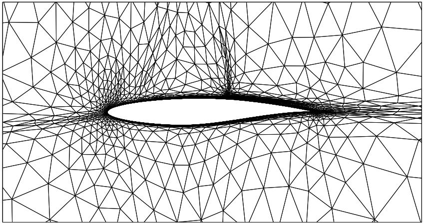
$p = 2$, dof = 5000



$p = 2$, dof = 10000

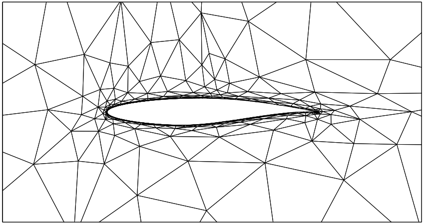


$p = 2$, dof = 20000

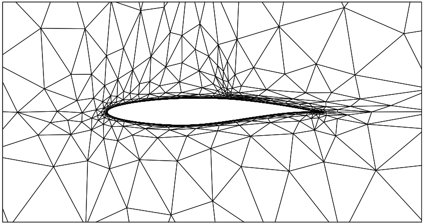


$p = 2$, dof = 40000

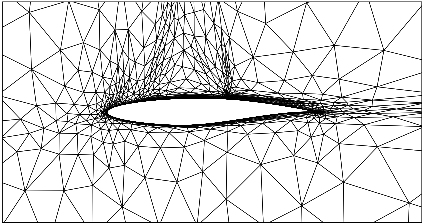
RAE 2822 in transonic flow: optimized meshes



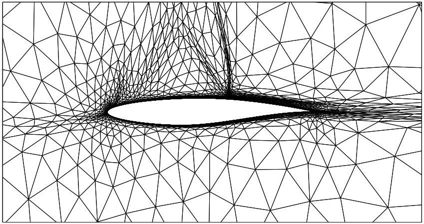
$p = 3$, `dof` = 5000



$p = 3$, `dof` = 10000



$p = 3$, `dof` = 20000



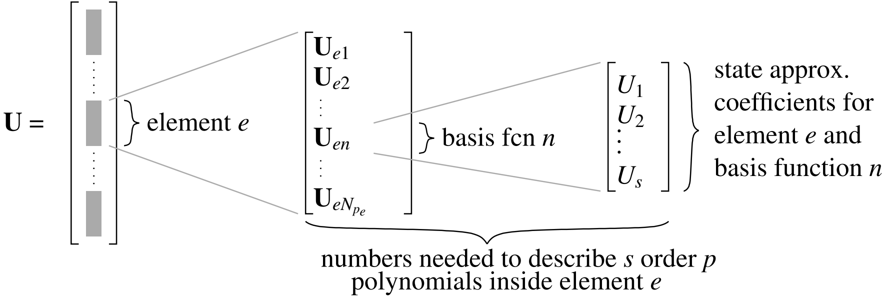
$p = 3$, `dof` = 40000

Backup Slides

Basis choice and DG system

- What basis functions to use?
  - DG $\Rightarrow \phi_j$ not tied to element shape
  - We can use full-order (tri) basis on quad elements
  - e.g. $p = 4$: 25 dofs for quad basis, 15 dofs for tri basis
- We lump all residuals and states into single vectors (size N),

$$\mathbf{R}(\mathbf{U}) = \mathbf{0}$$



Nonlinear solver: line search

1. Given: $\mathbf{U}_0$ and $\Delta\mathbf{U}$.
2. Compute $\omega^{\text{phys}} =$ maximum fraction such that $\mathbf{U}_0 + \omega^{\text{phys}}\Delta\mathbf{U}$ remains physical. This involves checks at quadrature points of each element.
3. Set $\omega = \min(1, \omega^{\text{phys}})$. If $\omega < 1$, set $\omega = \omega\beta^{\text{phys}}$, $\beta^{\text{phys}} < 1$.
4. While $\omega > \omega^{\text{min}}$ and $\|\mathbf{R}(\mathbf{U}_0 + \omega\Delta\mathbf{U})\| > \beta^{\text{residual}}\|\mathbf{R}(\mathbf{U}_0)\|$: set $w = w\beta^{\text{line}}$, where $\beta^{\text{line}} < 1$.
5. If $\omega < \omega^{\text{min}}$, do not update, and set $\text{CFL} = \text{CFL}\beta^{\text{CFL,decrease}}$.
6. If $\omega \geq \omega^{\text{min}}$, take the update: $\mathbf{U} = \mathbf{U}_0 + \omega\Delta\mathbf{U}$, and if $\omega = 1$, raise the CFL: $\text{CFL} = \text{CFL}\beta^{\text{CFL,increase}}$.

Parameters:

$$\beta^{\text{phys}} = 0.5,\ \ \beta^{\text{residual}} = 2.0,\ \ \beta^{\text{line}} = 0.5,\ \ \omega^{\text{min}} = 0.24,$$
$$\beta^{\text{CFL,increase}} = 1.2,\ \ \beta^{\text{CFL,decrease}} = 0.1.$$

Output error estimation

We want: $\delta J = J_H(\mathbf{U}_H) - J(\mathbf{U})$

This is the difference between J computed with the discrete system solution, $\mathbf{U}_H$, and J computed with the *exact* solution, $\mathbf{U}$

We'll settle for: $\delta J = J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h)$

This is the difference in J relative to a finer discretization (h)

$$\text{coarse space:} \quad \rightarrow \underbrace{\mathbf{R}_H(\mathbf{U}_H) = 0}_{N_H \text{ equations}} \rightarrow \underbrace{\mathbf{U}_H}_{\text{state} \in \mathbb{R}^{N_H}} \rightarrow \underbrace{J_H(\mathbf{U}_H)}_{\text{output (scalar)}}$$

$$\text{fine space:} \quad \rightarrow \underbrace{\mathbf{R}_h(\mathbf{U}_h) = 0}_{N_h \text{ equations}} \rightarrow \underbrace{\mathbf{U}_h}_{\text{state} \in \mathbb{R}^{N_h}} \rightarrow \underbrace{J_h(\mathbf{U}_h)}_{\text{output (scalar)}}$$

The adjoint-weighted residual

- $\mathbf{U}_h^H$ solves a *perturbed* fine-space problem

$$\text{find } \mathbf{U}_h' \text{ such that:} \quad \mathbf{R}_h(\mathbf{U}_h') \underbrace{-\mathbf{R}_h(\mathbf{U}_h^H)}_{\delta \mathbf{R}_h} = 0 \quad \Rightarrow \text{answer: } \mathbf{U}_h' = \mathbf{U}_h^H$$

- The fine-space adjoint, $\mathbf{\Psi}_h$, ($p+1$, solved exactly) then tells us to expect an output perturbation of

$$\underbrace{J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h)}_{\approx \delta J} = \mathbf{\Psi}_h^T \delta \mathbf{R}_h = -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)$$

- This equation assumes small perturbations (e.g. if nonlinear; linearization is about $\mathbf{U}_h^H$)
- In summary, we have an *adjoint-weighted residual*:

$$\boxed{\delta J \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)}$$

Mesh adaptation using a metric field

- Unstructured meshes offer more *geometric* and *adaptive* flexibility over structured ones
- Resolution information: size and shape of an element
- This can be encoded in a *metric field* [1: Borouchaki, 1995] [2: Pennec, 2006] over the domain
- We are interested in an adaptive method where the mesh is *regenerated* at each iteration using the current mesh and information from the solution
- Key ingredients:
  1. Metric-conforming mesh generator
  2. Solution-based metric specification

Error convergence model

- $\mathcal{E}_{e0}$ = current output error indicator on element e (from AWR)
- $\mathcal{S}_e$ = proposed metric step matrix on element e
- Model for error after metric modification with $\mathcal{S}_e$:

$$\boxed{\mathcal{E}_e = \mathcal{E}_{e0} \exp\left[\mathrm{tr}(R_e \mathcal{S}_e)\right]}$$

- R_e = error convergence rate tensor (identified by sampling)
- Note, this is a generalization to anisotropic shape changes of the more familiar isotropic model,

$$\mathcal{E}_e = \mathcal{E}_{e0} \left(\frac{h}{h_0}\right)^r = \mathcal{E}_{e0} \exp\left[r \log(h/h_0)\right]$$

- Sum over elements to get the total error on the mesh,

$$\boxed{\mathcal{E} = \sum_e \mathcal{E}_e}$$

Cost model

cost = degrees of freedom (`dof`) in solution approximation

- Assume p = approximation order = same for all elements
- $\mathcal{C}_{e0}$ = current cost on element e, e.g. $(p+1)(p+2)/2$
- New cost after application of step matrix $\mathcal{S}_e$,

$$\mathcal{C}_e = C_{e0} \underbrace{\exp\left[\frac{1}{2}\mathrm{tr}(\mathcal{S}_e)\right]}_{\mathrm{Area}_0/\mathrm{Area}}$$

- Note, the cost is just scaled by $\mathrm{Area}_0/\mathrm{Area}$ = # new elements occupying the original area of element e
- Sum over elements to get the total cost on the mesh,

$$\mathcal{C} = \sum_e \mathcal{C}_e$$

References I

[1] H. Borouchaki, P. George, F. Hecht, P. Laug, and E Saltel.
Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes.
INRIA-Rocquencourt, France. Tech Report No. 2741, 1995.

[2] Xavier Pennec, Pierre Fillard, and Nicholas Ayache.
A riemannian framework for tensor computing.
International Journal of Computer Vision, 66(1):41–66, 2006.

[3] Masayuki Yano.
An Optimization Framework for Adaptive Higher-Order Discretizations of Partial Differential Equations on Anisotropic Simplex Meshes.
PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.