High-Order Output-Based Adaptive Methods for Steady and Unsteady Aerodynamics

Krzysztof J. Fidkowski, *University of Michigan*

37^{th} Advanced VKI CFD Lecture Series
von Karman Institute, Rhode-St-Genèse, Belgium

December 9–12, 2013

Outline

Outline

Introduction

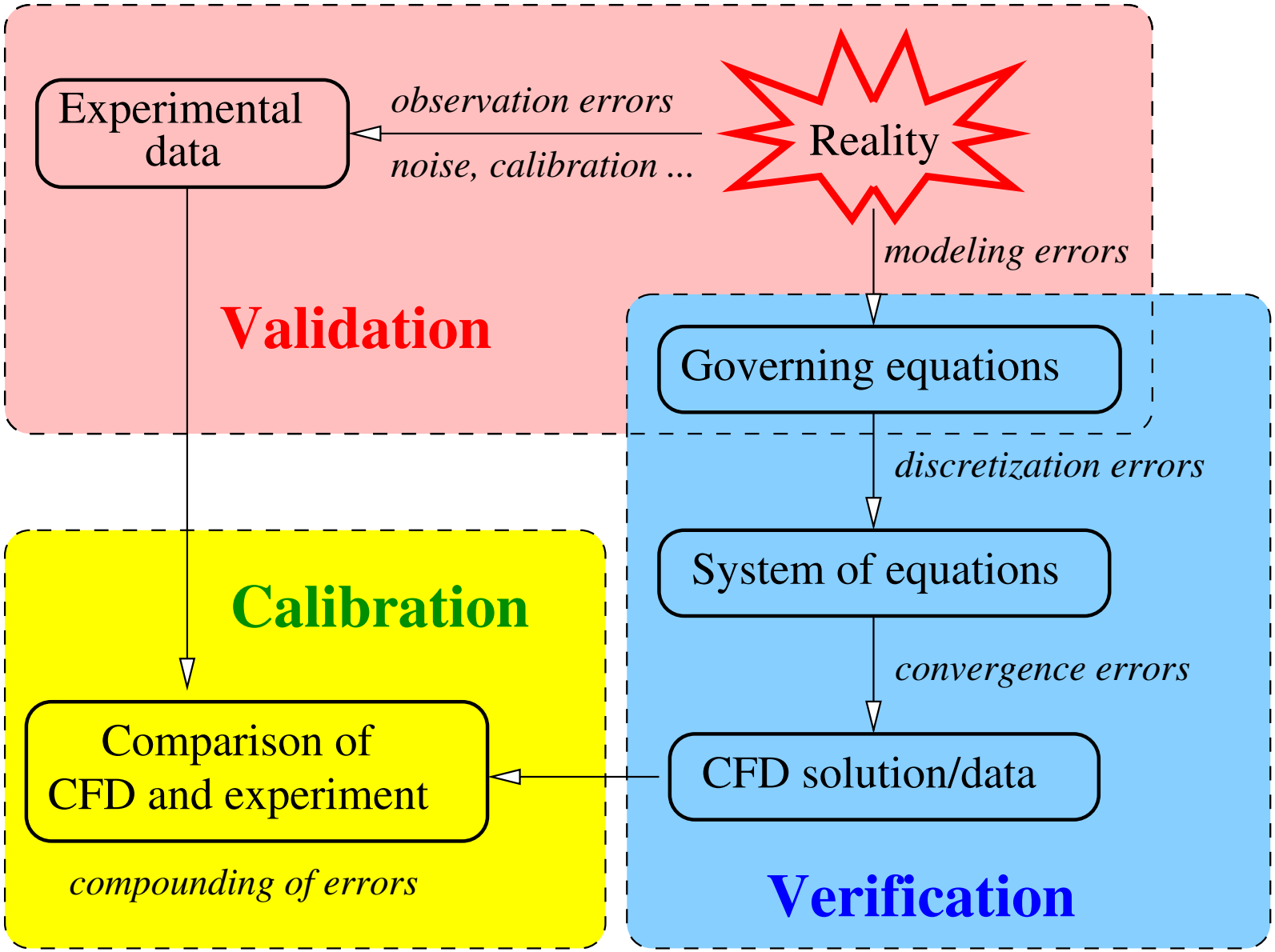
Complex CFD simulations made possible by

- Increasing computational power
- Improvements in numerical algorithms

New liability: ensuring accuracy of computations

- Management by expert practitioners is not feasible for increasingly-complex flow fields
- Reliance on best-practice guidelines is an open-loop solution: numerical error is unchecked for novel configurations
- Output calculations are not yet sufficiently robust, even on relatively standard simulations

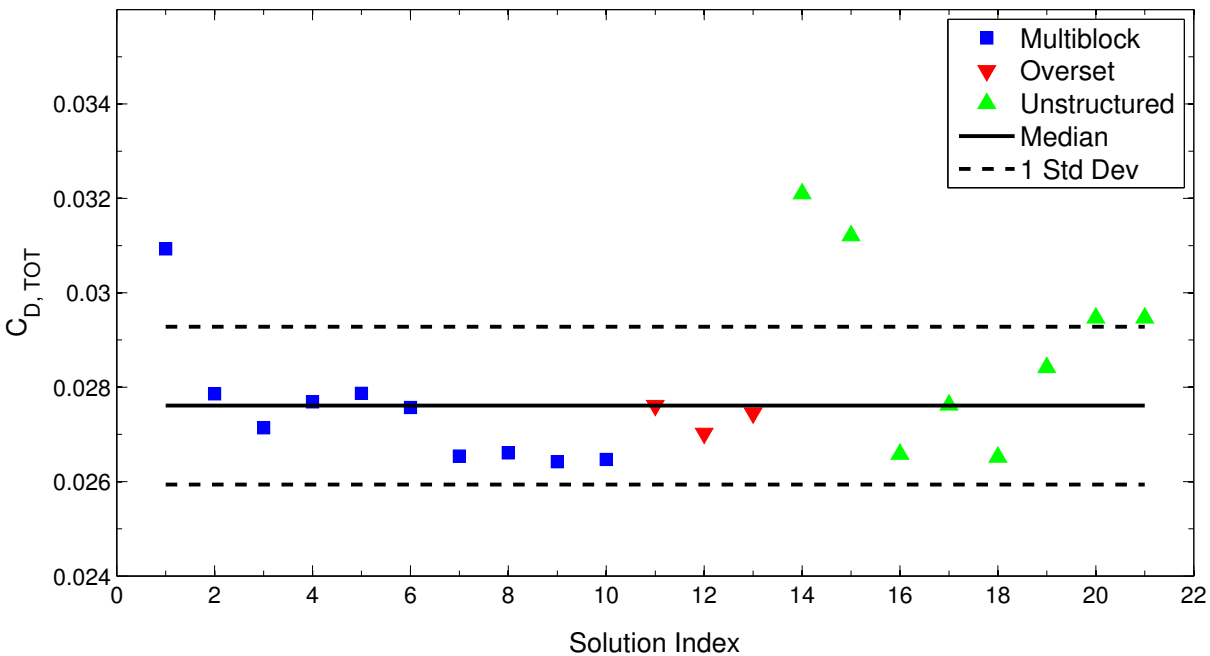
Errors in simulations come from various sources



Verification is important

AIAA Drag Prediction Workshop III (2006)

- Wing-body geometry, $M = 0.75, C_L = 0.5, Re = 5 \times 10^6$
- Drag computed with various state of the art CFD codes

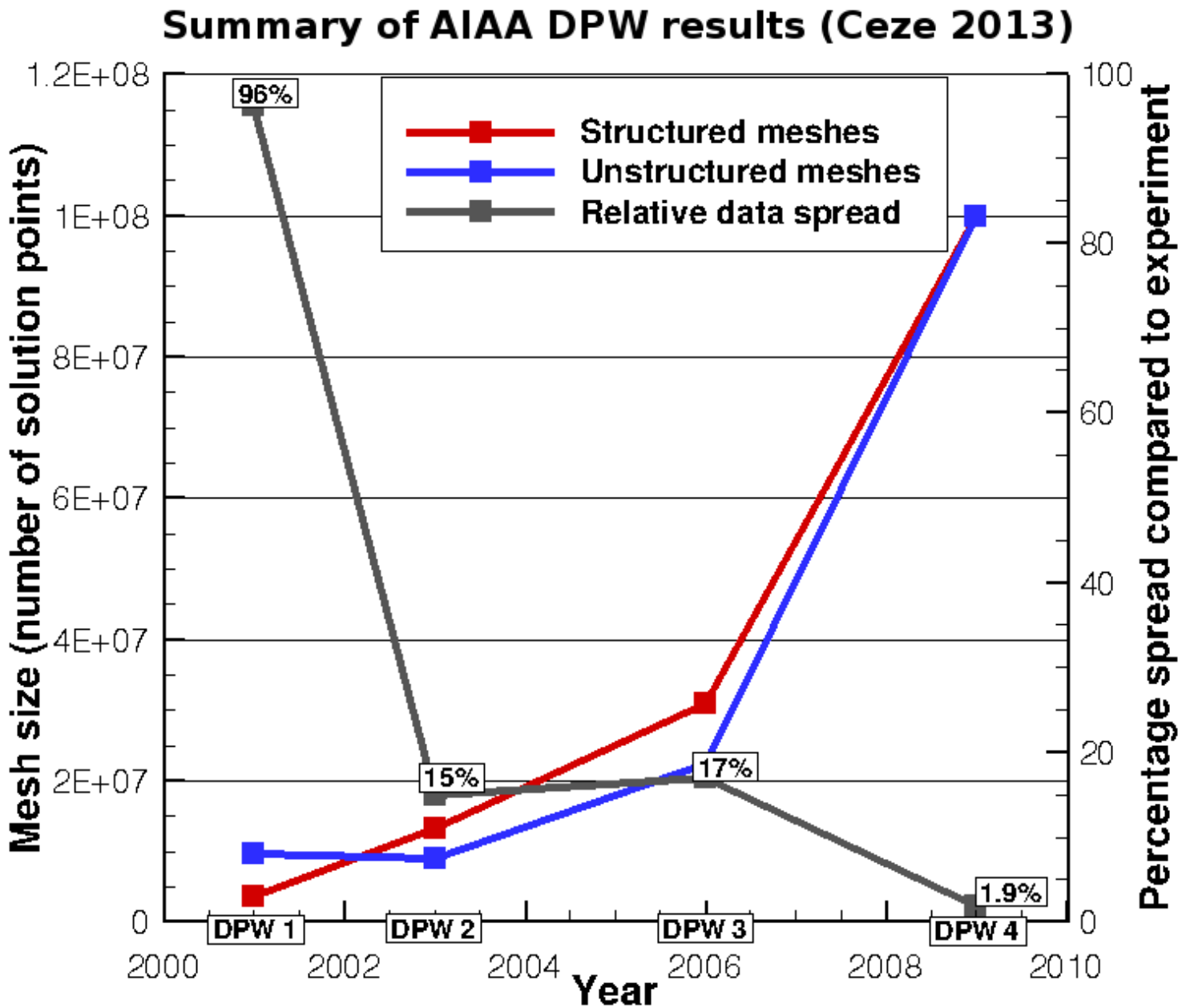


Differences in:

- Physical models
- Discretization
- **Mesh size distribution**

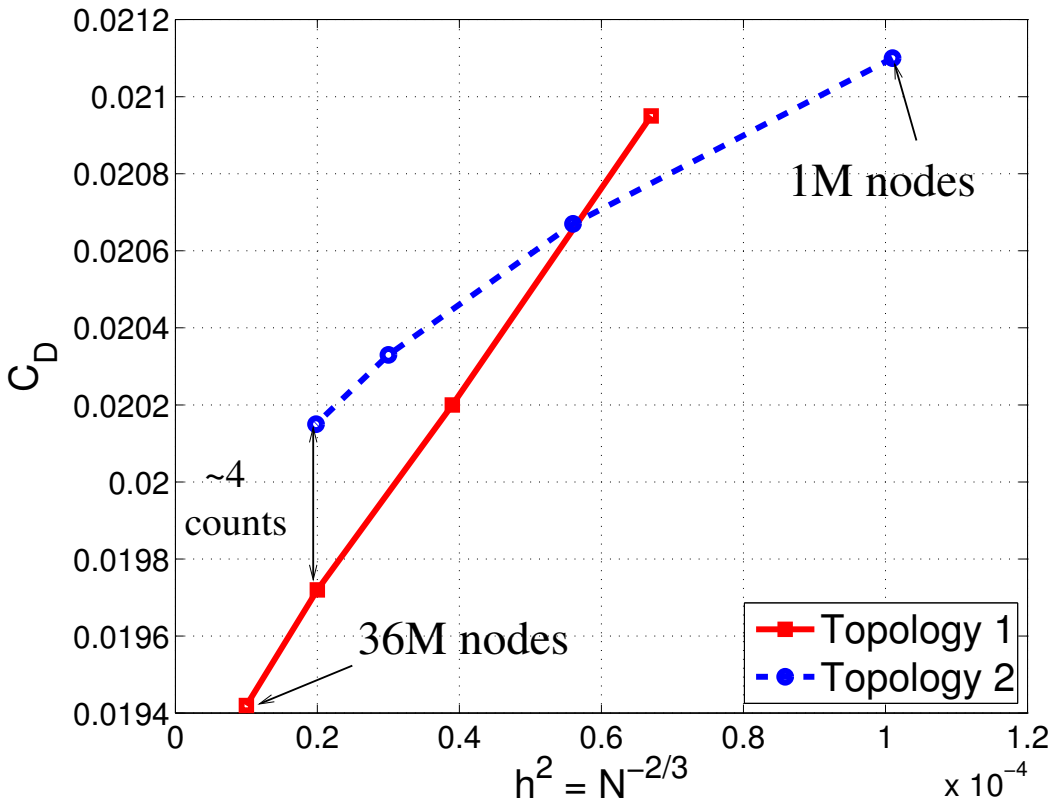
1 drag count ($.0001 C_D$) $\approx$ 4-8 passengers for a large aircraft

Numerical errors have come down, at a large cost



Uniform grid refinement may be misleading

- DPW III wing-alone case: $M_\infty = 0.76, Re = 5 \times 10^6$
- Two mesh sequences generated using best-practice mesh-generation guidelines [Mavriplis, 2007]
- Run on same code (turbulence model, solver, etc)



Verification = control of numerical error

- Dominant source is **discretization error** (i.e. lack of appropriate mesh resolution)
- Controlling error means answering
  1. How much error is present? (**error estimation**)
  2. How can this error be reduced? (**mesh adaptation**)
- Possible strategies:

	Error estimation?	Effective adaptation?
Resource exhaustion	**No**	**No**
Expert assessment	**Maybe**	**Maybe**
Convergence studies	**Yes**	**No**
Comparison to experiments	**Yes**	**No**
Feature–based adaptation	**No**	**Maybe**
Output–based methods	**Yes**	**Yes**

Improving CFD robustness

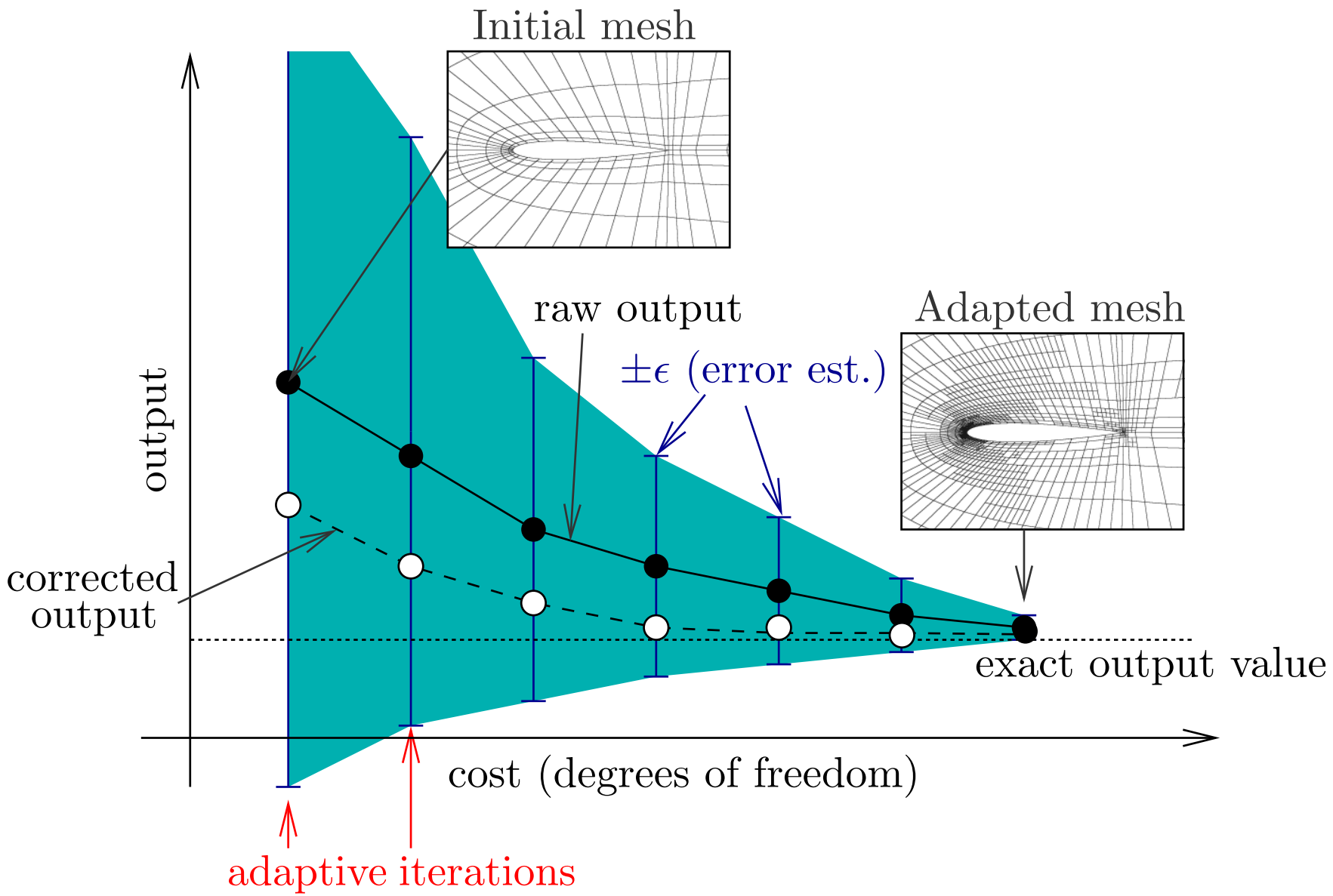
Error estimation

- Error estimates on outputs of interest are necessary for confidence in CFD results
- Mathematical theory exists for obtaining such estimates
- Recent works demonstrate the success of this theory for aerospace applications

Mesh adaptation

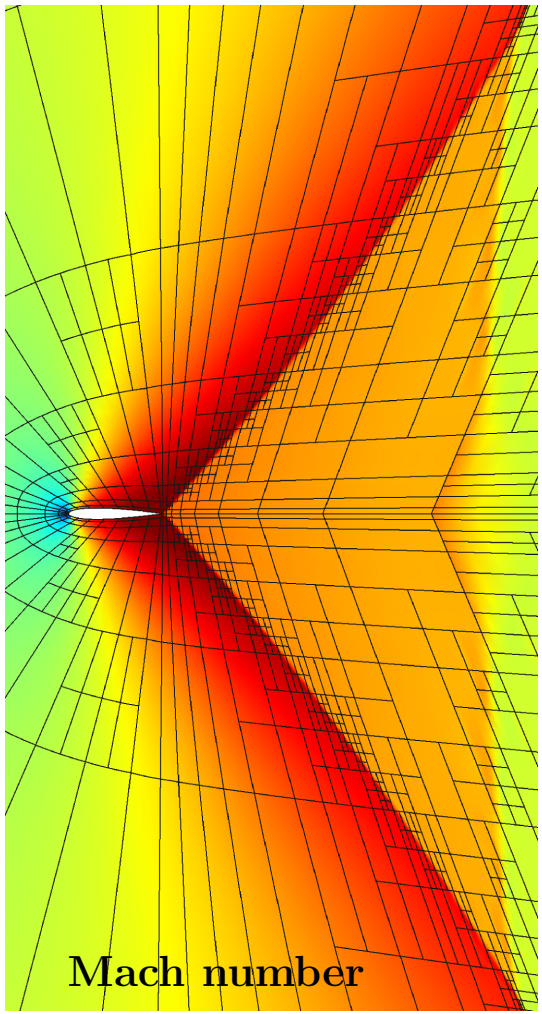
- Error estimation alone is not enough
- Engineering accuracy for complex aerospace simulations demands mesh adaptation to control numerical error
- Automated adaptation improves robustness by closing the loop in CFD analysis

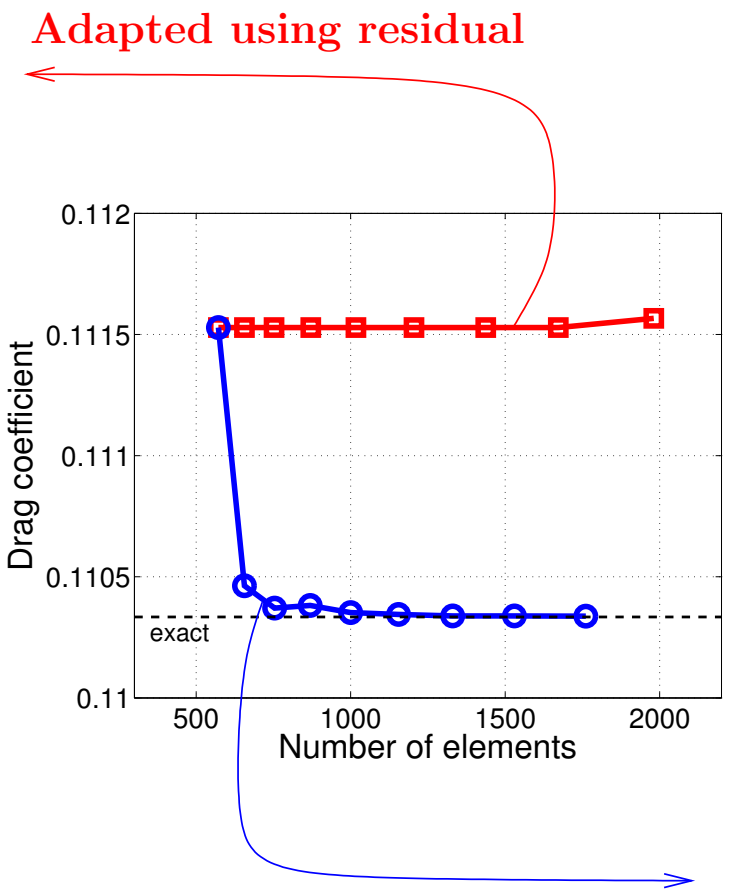
A typical output-adaptive result

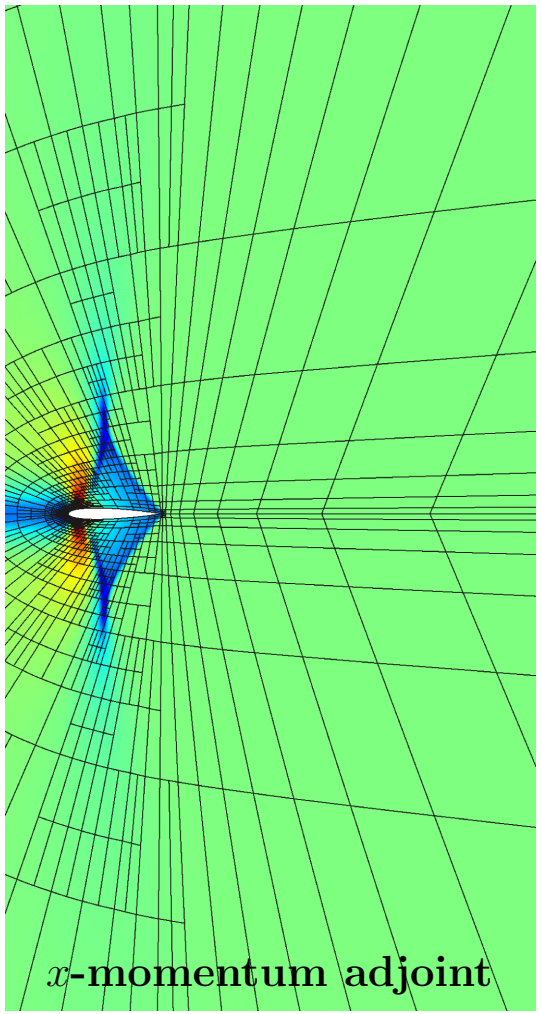


Why not just adapt "obvious" regions?

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

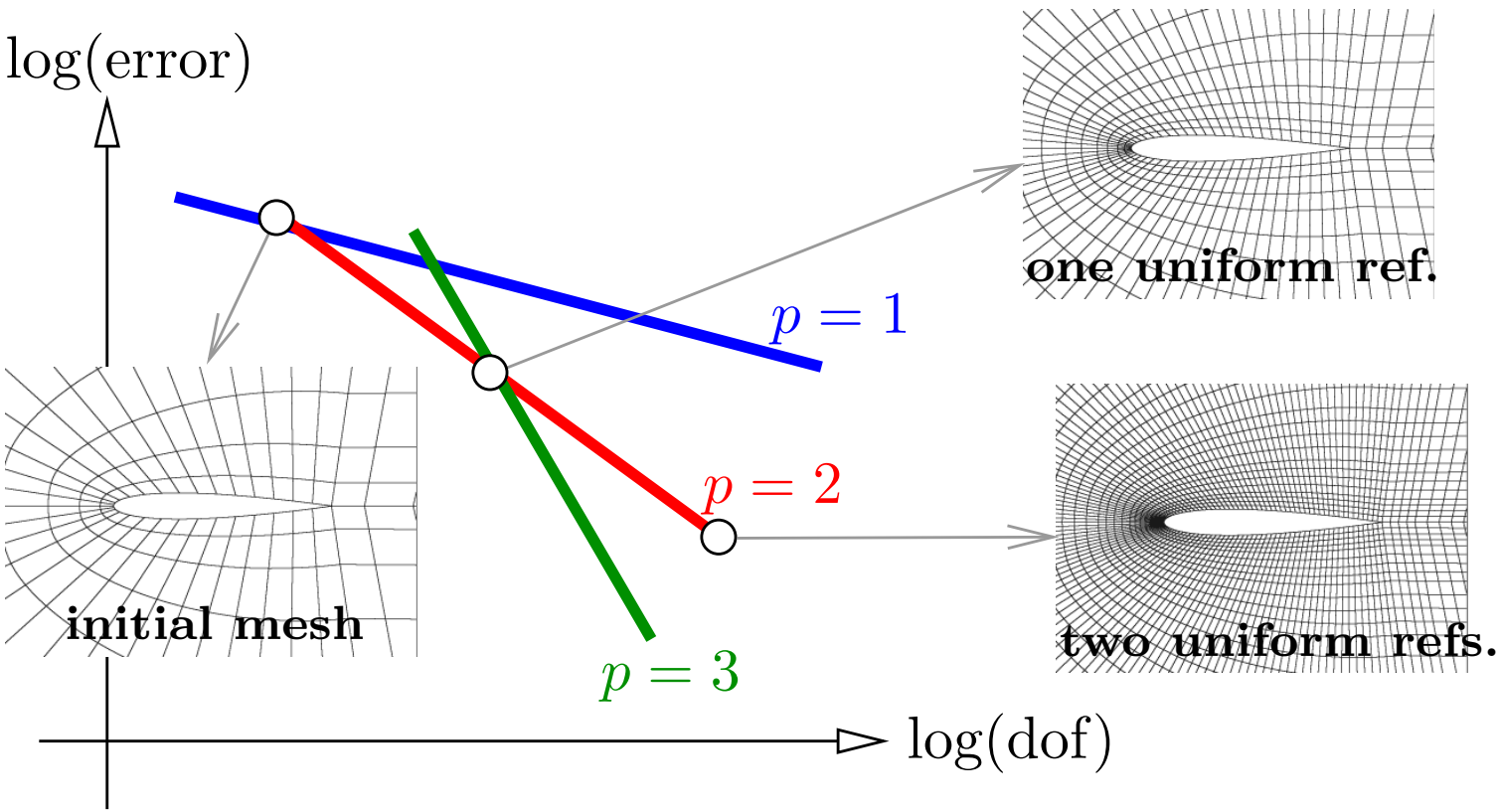






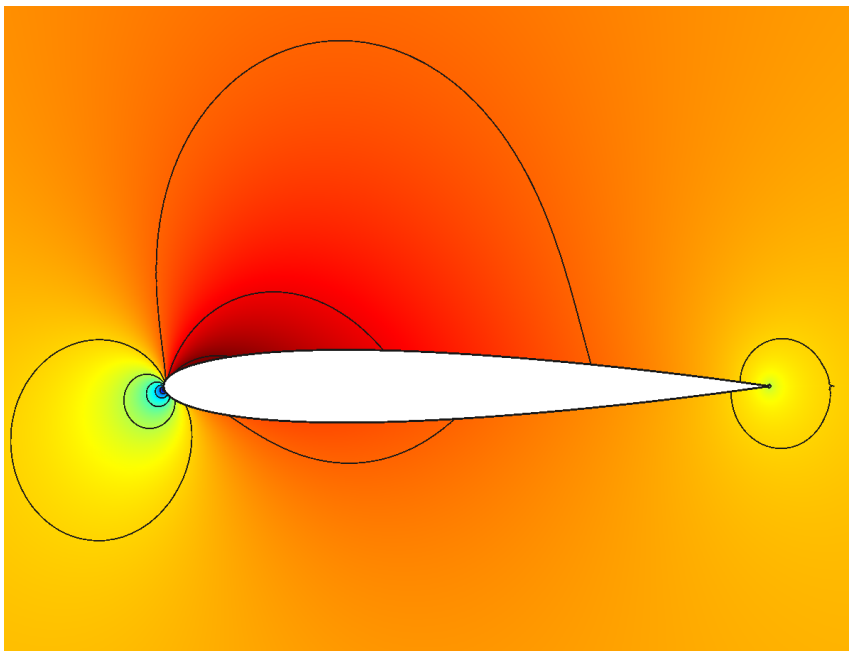
The role of high order

- High-order methods: errors converge faster than 2^{nd}-order
- Typically choose high-order methods for “smooth” problems, where we expect to see convergence plots that look like:

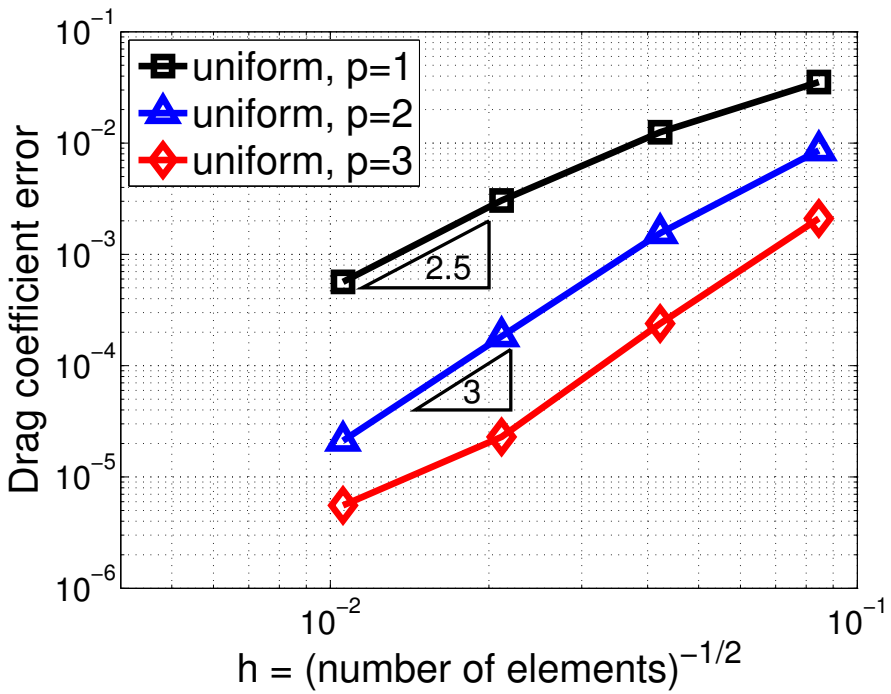


Can aero applications benefit from high order?

- Question considered by recent high-order CFD workshops
- Aerospace applications usually have both smooth and singular features (shocks, trailing edges)
- Singularities can limit observed rates



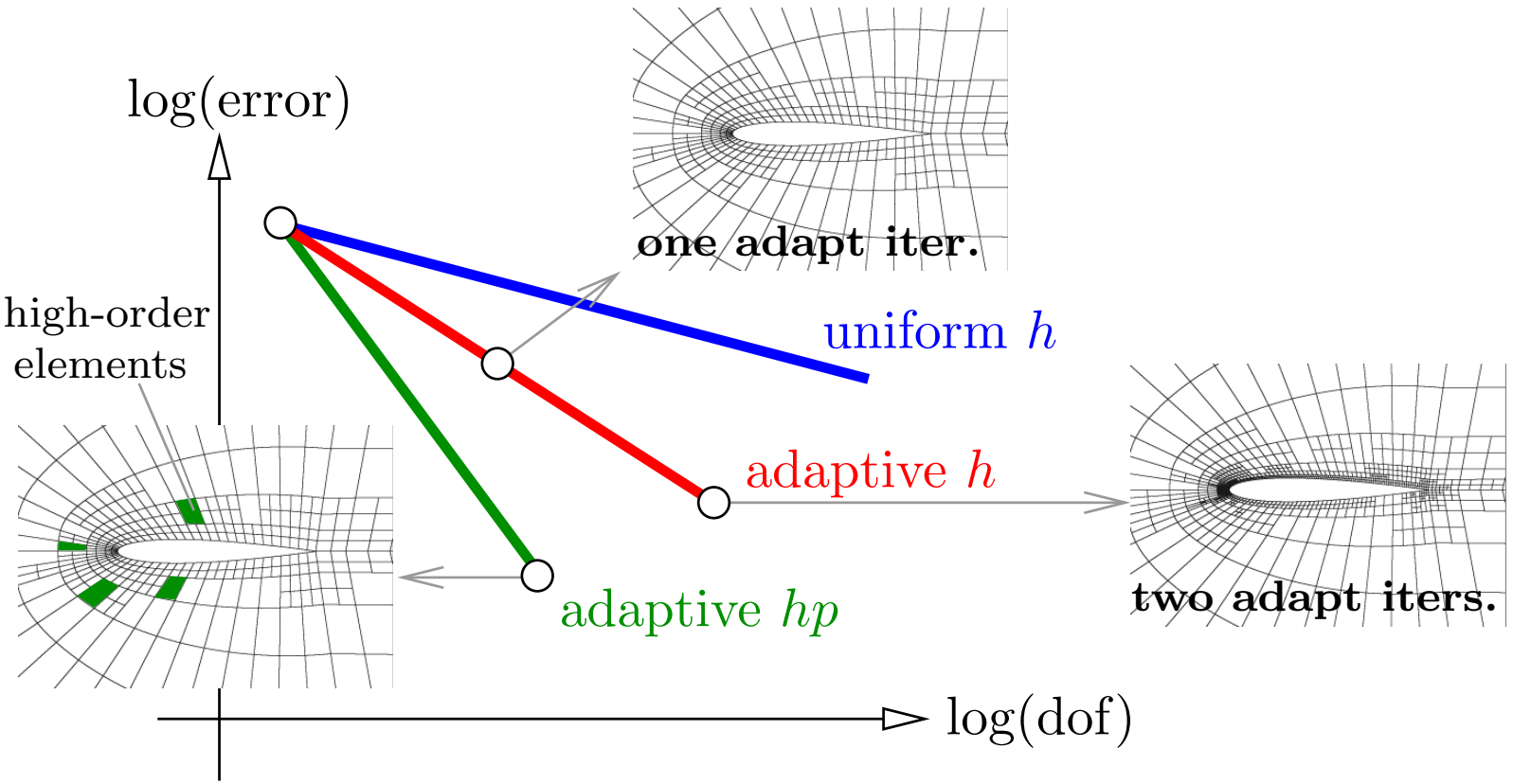
Mach numbers (0–0.7), Euler flow over a NACA 0012 airfoil



Drag error convergence

High-order in mesh adaptation

- Adaptation can isolate singularities with small elements
- In many high-order methods, local p-enrichment is possible
- High-order just becomes another refinement tool for efficiently improving accuracy



Outline

Conservation equations

- PDE:

$$\boxed{\partial_t \mathbf{u} + \partial_i \mathbf{H}_i(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}}$$

- $i : 1 \leq i \leq d$ indexes the spatial dimension d (implied sum)
- $\mathbf{u} \in \mathbb{R}^s$ is the state vector
- $\mathbf{H}_i \in \mathbb{R}^s$ is the i^{th} component of the total flux

$$\mathbf{H}_i = \underbrace{\mathbf{F}_i(\mathbf{u})}_{\text{inviscid flux}} + \underbrace{\mathbf{G}_i(\mathbf{u}, \nabla \mathbf{u})}_{\text{viscous flux}}$$

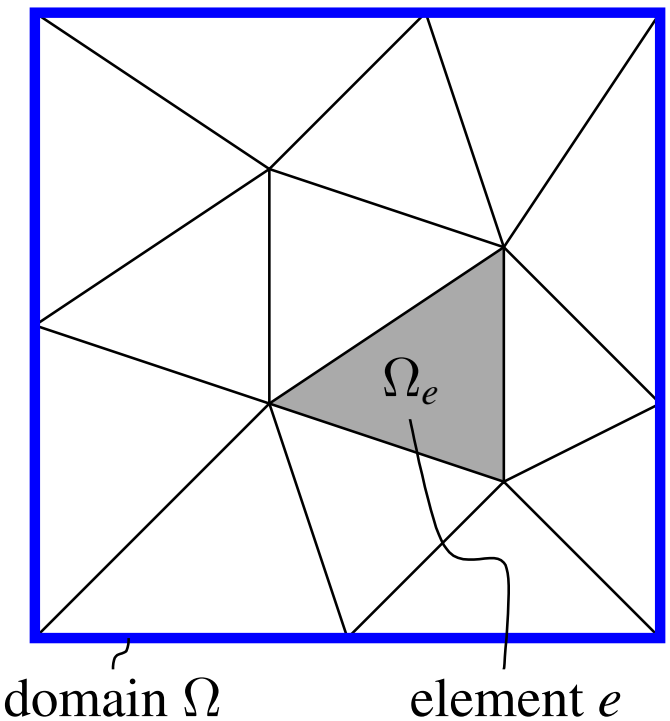
- The viscous flux is

$$\mathbf{G}_i(\mathbf{u}, \nabla \mathbf{u}) = -\mathbf{K}_{ij}(\mathbf{u})\, \partial_j \mathbf{u}$$

Solution approximation

Polynomials of order p_e on each element:

$$\mathbf{u}_h(\vec{x}) \approx \sum_{e=1}^{N_e} \sum_{n=1}^{N_{pe}} \mathbf{U}_{en} \phi_{en}(\vec{x})$$



N_e	=	# of elements
N_{p_e}	=	# of basis fcns on element e
$\phi_{en}(\vec{x})$	=	n^{th} basis fcn of order p_e on e
p_e	=	approximation order on element e
$\mathbf{U}_{en}$	=	vector of s coefficients on n^{th} basis function on element e

Weak form

- Multiply the PDE by test functions $\mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h$ to get

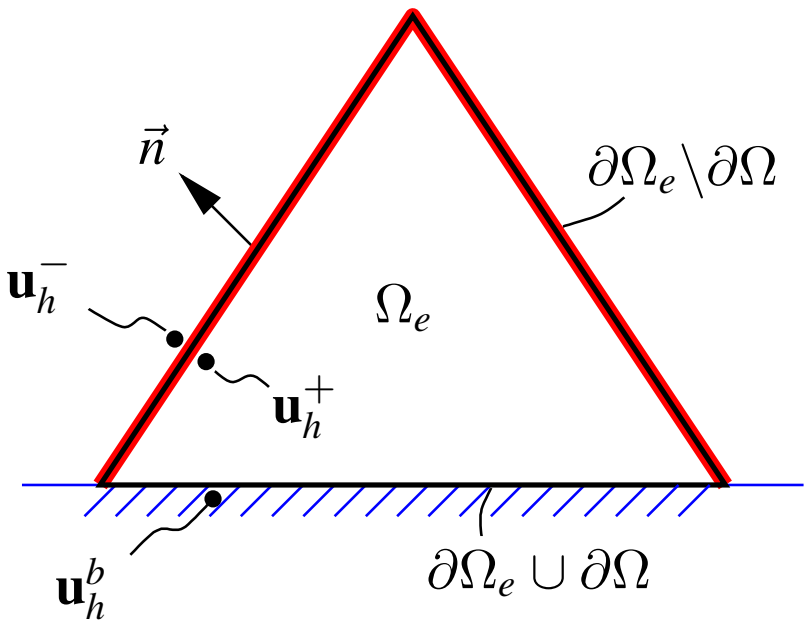
$$\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h) = 0, \quad \forall \mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h$$

- Integrating by parts and using BR2, we obtain

$$\begin{aligned}\mathcal{R}_h(\mathbf{u}_h, \mathbf{v}_h|_{\Omega_e}) \quad = \quad & \int_{\Omega_e} \mathbf{v}_h^T \partial_t \mathbf{u}_h \, d\Omega - \int_{\Omega_e} \partial_i \mathbf{v}_h^T \mathbf{H}_i \, d\Omega \\ & + \underbrace{\int_{\partial\Omega_e} \mathbf{v}_h^{+T} \left(\widehat{\mathbf{F}} + \widehat{\mathbf{G}} \right) ds}_{\text{interface/boundary flux}} \\ & + \underbrace{\int_{\partial\Omega_e} \partial_i \mathbf{v}_h^{+T} \widehat{\mathbf{K}}_{ij} \left(\mathbf{u}_h^+ - \widehat{\mathbf{u}}_h \right) ds}_{\text{adjoint-consistency term}}\end{aligned}$$

Fluxes

- $(\cdot)^+$ = quantity from element interior
- $(\cdot)^-$ = quantity from neighbor element
- $(\cdot)^b$ = quantity defined on a boundary
- $\widehat{(\cdot)}$ = an average quantity on a face
- BR2: unique state on an interior face is $\widehat{\mathbf{u}}_h = (\mathbf{u}_h^+ + \mathbf{u}_h^-)/2$



$$
\begin{aligned}
\widehat{\mathbf{F}} &= \widehat{\mathbf{F}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \vec{n}) \\
\widehat{\mathbf{G}} &= \widehat{\mathbf{G}}(\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla\mathbf{u}_h^+, \nabla\mathbf{u}_h^-, \vec{n}) \\
\widehat{\mathbf{F}}^b &= \widehat{\mathbf{F}}^b(\mathbf{u}_h^+, \mathrm{BC}, \vec{n}) \\
\widehat{\mathbf{G}}^b &= \widehat{\mathbf{G}}^b(\mathbf{u}_h^b, \nabla\mathbf{u}_h^+, \mathrm{BC}, \vec{n})
\end{aligned}
$$

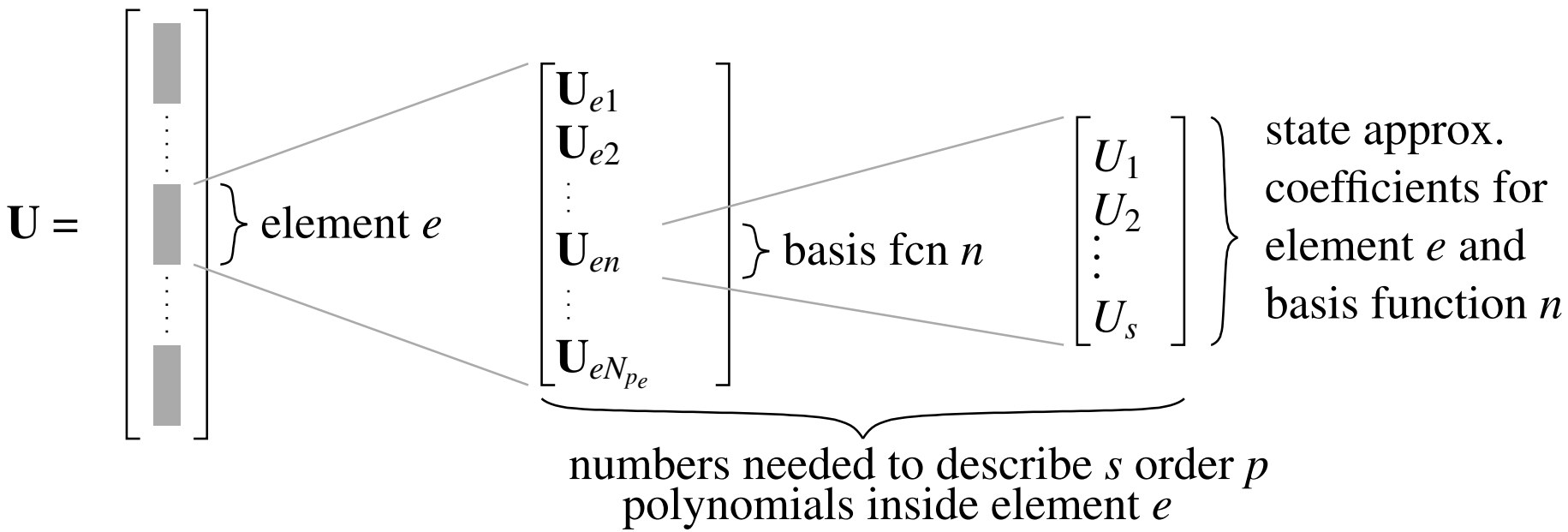
Discrete system

- Discrete residual on element e for n^{th} test function,

$$\mathbf{R}_{en} \equiv \{\mathcal{R}_h(\mathbf{u}_h, \phi_{en}\mathbf{e}_r)\}_{r=1\ldots s} \in \mathbb{R}^s$$

- We lump all residuals and states into single vectors (size N),

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



Verification using a manufactured solution

- How do we know if we coded the discretization correctly?
- Analytical solutions are scarce, especially for RANS
- Let's "make up" a solution,

$$\mathbf{u}(\vec{x}) = \mathbf{u}^{\mathrm{MS}}(\vec{x}) = \text{chosen by the user}$$

- Substituting $\mathbf{u}^{\mathrm{MS}}(\vec{x})$ into the PDE gives a remainder of

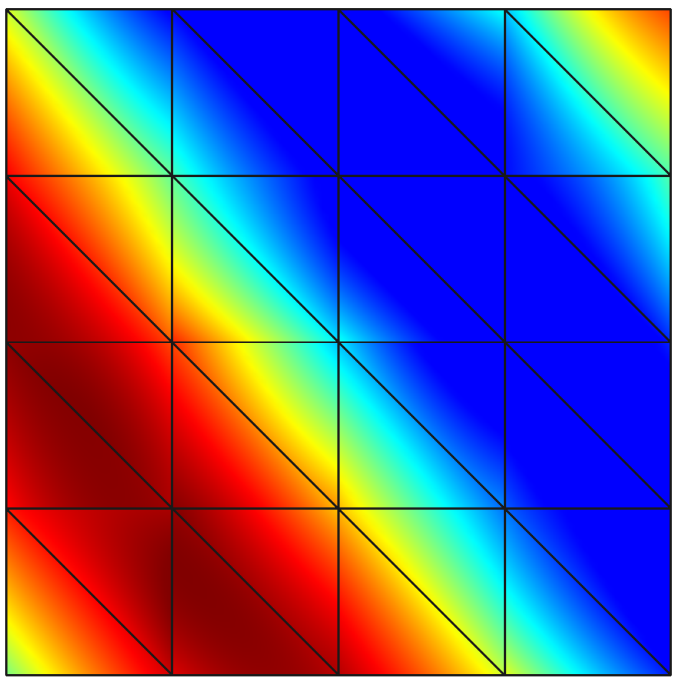
$$\mathbf{s}^{\mathrm{MS}} \equiv \partial_t \mathbf{u}^{\mathrm{MS}} + \partial_i \mathbf{H}_i(\mathbf{u}^{\mathrm{MS}}, \nabla \mathbf{u}^{\mathrm{MS}})$$

- Using this remainder as a negative source term gives a PDE that $\mathbf{u}^{\mathrm{MS}}$ *does* satisfy,

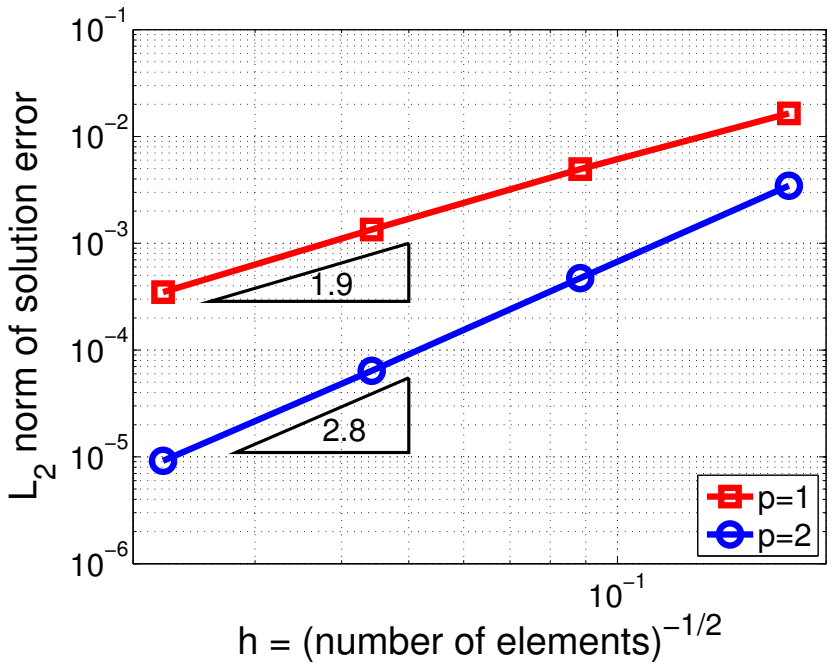
$$\boxed{\partial_t \mathbf{u}^{\mathrm{MS}} + \partial_i \mathbf{H}_i(\mathbf{u}^{\mathrm{MS}}, \nabla \mathbf{u}^{\mathrm{MS}}) - \mathbf{s}^{\mathrm{MS}} = \mathbf{0}}$$

Manufactured solution results for RANS

- Pick a sinusoidal variation, $\rho^{\text{MS}} = a_\rho + b_\rho \sin(c_\rho x + d_\rho y)$, and similarly for the other state components.
- Compute and discretize the source term, $\mathbf{s}^{\text{MS}}$
- Does solution on progressively-finer meshes approach $\mathbf{u}^{\text{MS}}$? Check with L_2 norm:



Manufactured solution, ρ



Spalart-Allmaras RANS test

Local sensitivities

- Suppose N_μ parameters affect our PDE, but we only have one scalar output, $J(\mathbf{U})$:

$$\underbrace{\boldsymbol{\mu}}_{\text{inputs} \in \mathbb{R}^{N_\mu}} \rightarrow \underbrace{\mathbf{R}(\mathbf{U}, \boldsymbol{\mu}) = 0}_{N \text{ equations}} \rightarrow \underbrace{\mathbf{U}}_{\text{state} \in \mathbb{R}^{N}} \rightarrow \underbrace{J(\mathbf{U})}_{\text{output (scalar)}}$$

- We are interested in how J changes with $\boldsymbol{\mu}$,

$$\frac{dJ}{d\boldsymbol{\mu}} \in \mathbb{R}^{1 \times N_\mu} = N_\mu \text{ sensitivities}$$

- Brute force approach: perturb each entry in $\boldsymbol{\mu}$ individually, re-solve the PDE, and measure the perturbation in the output

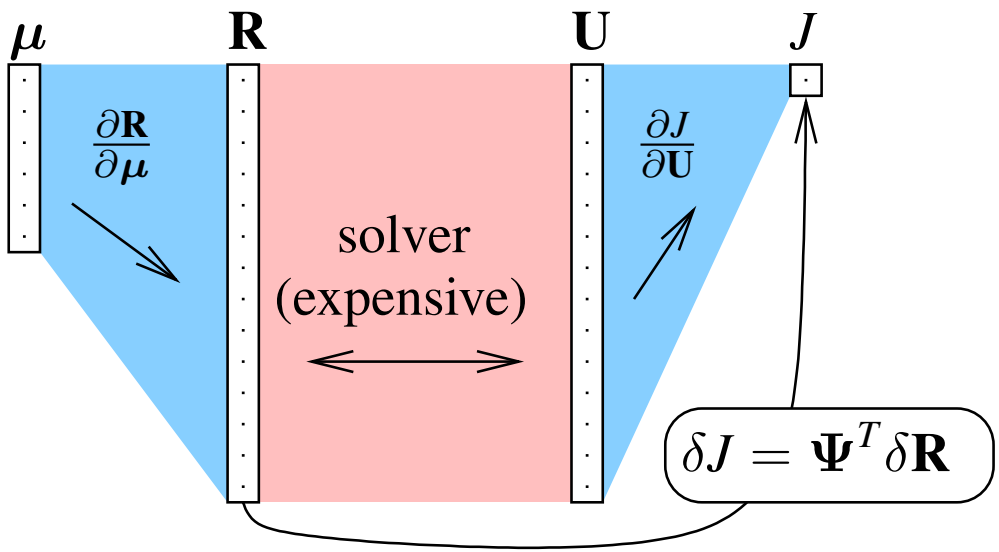
This is inefficient for large N_μ

The discrete adjoint

- We can efficiently compute sensitivities using a discrete adjoint vector, $\boldsymbol{\Psi} \in \mathbb{R}^N$,

$$\frac{dJ}{d\boldsymbol{\mu}} = \boldsymbol{\Psi}^T \frac{\partial \mathbf{R}}{\partial \boldsymbol{\mu}}$$

- Each entry in $\boldsymbol{\Psi}$ is the sensitivity of J to residual source perturbations in the corresponding entry in $\mathbf{R}$



The discrete adjoint equation

- Consider a small perturbation $\delta\mathbf{R}$ to the residual
- The resulting (linearized) state perturbation, $\delta\mathbf{U}$ satisfies

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\delta\mathbf{U} + \delta\mathbf{R} = 0$$

- Also linearizing the output we have,

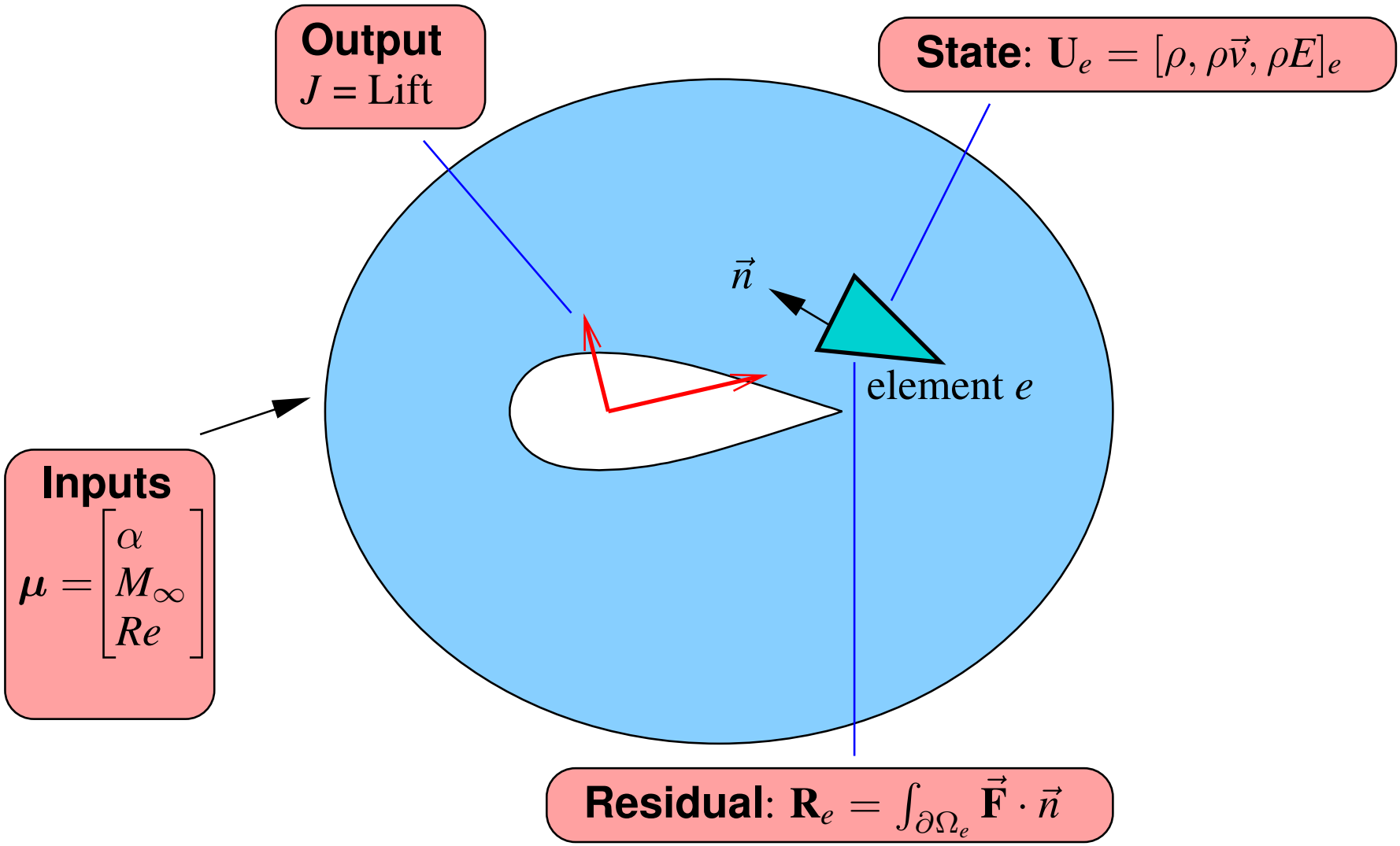
$$\delta J = \underbrace{\frac{\partial J}{\partial \mathbf{U}}\delta\mathbf{U}}_{\text{adjoint definition}} = \overbrace{\boldsymbol{\Psi}^T \delta\mathbf{R} = -\boldsymbol{\Psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\delta\mathbf{U}}^{\text{linearized equations}}$$

- Requiring the above to hold for arbitrary perturbations yields the linear *discrete adjoint equation*

$$\boxed{\left(\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right)^T \boldsymbol{\Psi} + \left(\frac{\partial J}{\partial \mathbf{U}}\right)^T = 0}$$

Adjoints in aerodynamics

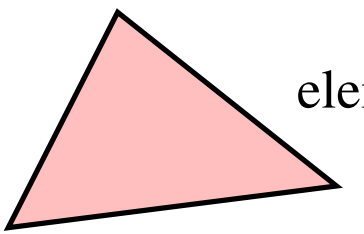
Consider flow over an airfoil:



Output sensitivity to residuals: the adjoint

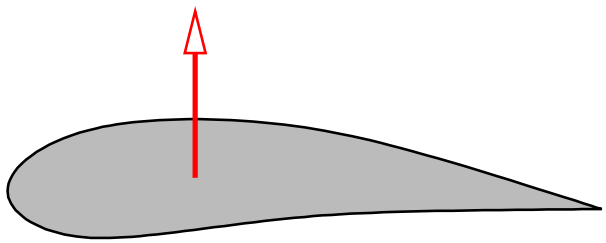
The lift adjoint Ψ is the sensitivity of lift to residual sources.

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



element e

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

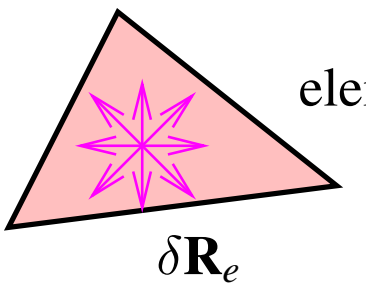


state $\mathbf{U}$

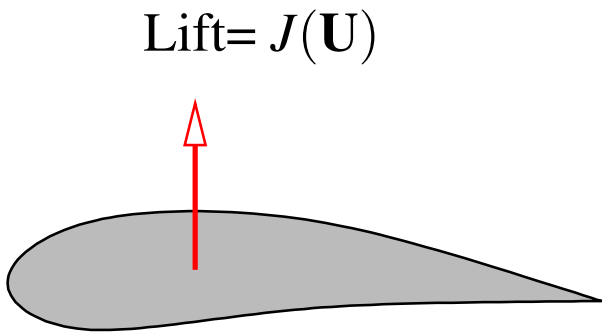
Output sensitivity to residuals: the adjoint

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

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



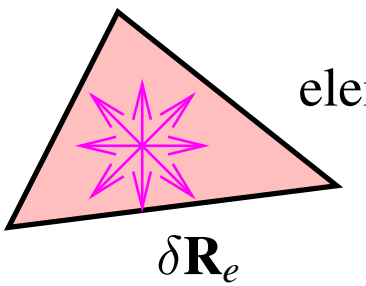
What if we add a residual source, $\delta\mathbf{R}_e$?



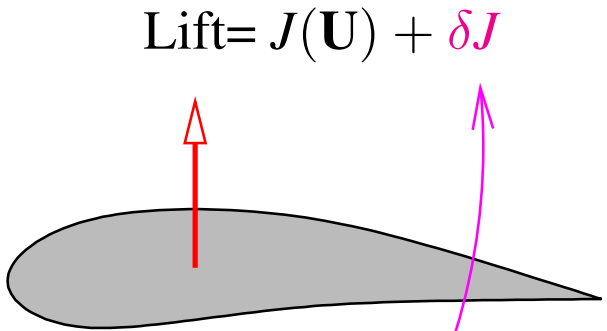
Output sensitivity to residuals: the adjoint

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

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



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

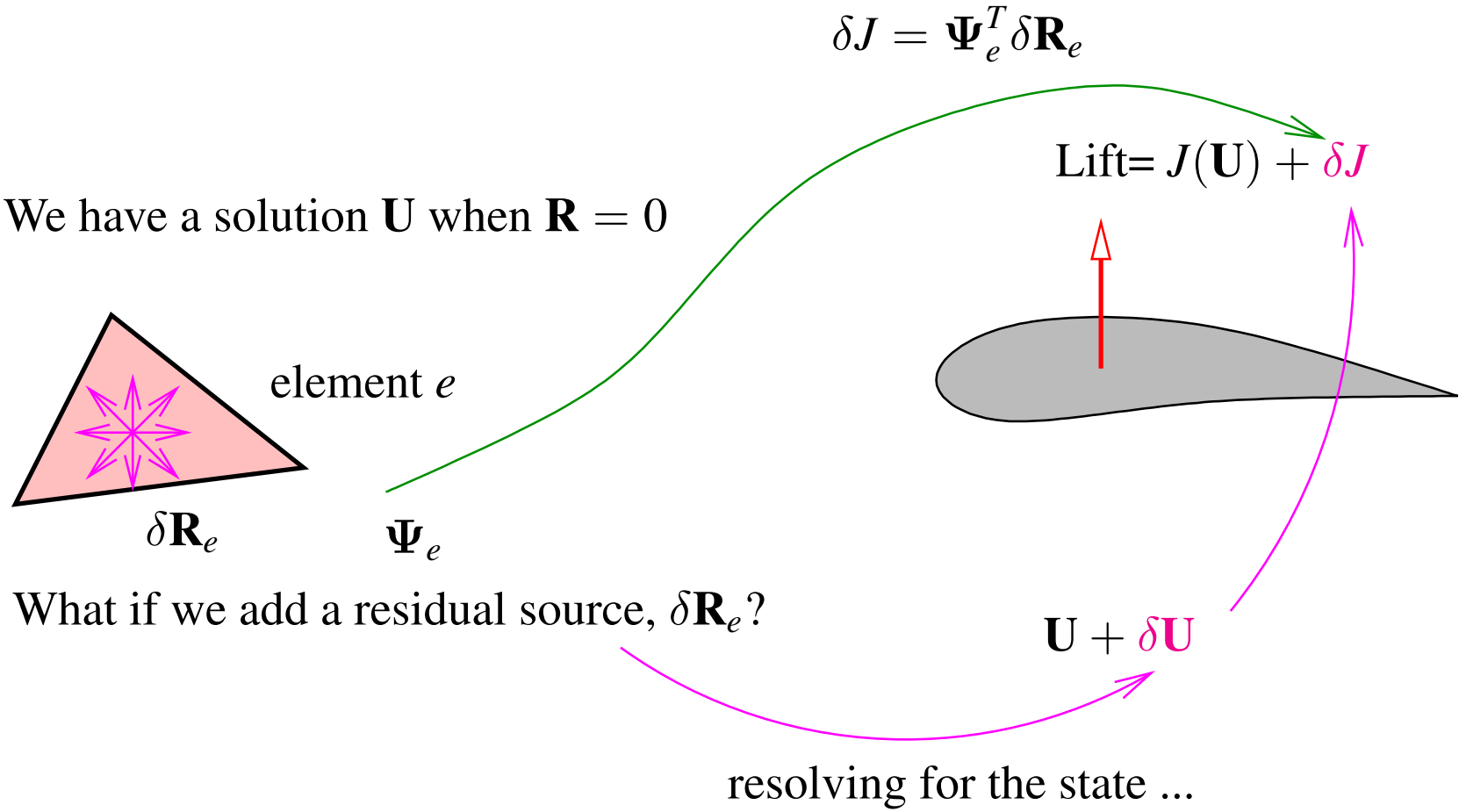


$\mathbf{U} + \delta\mathbf{U}$

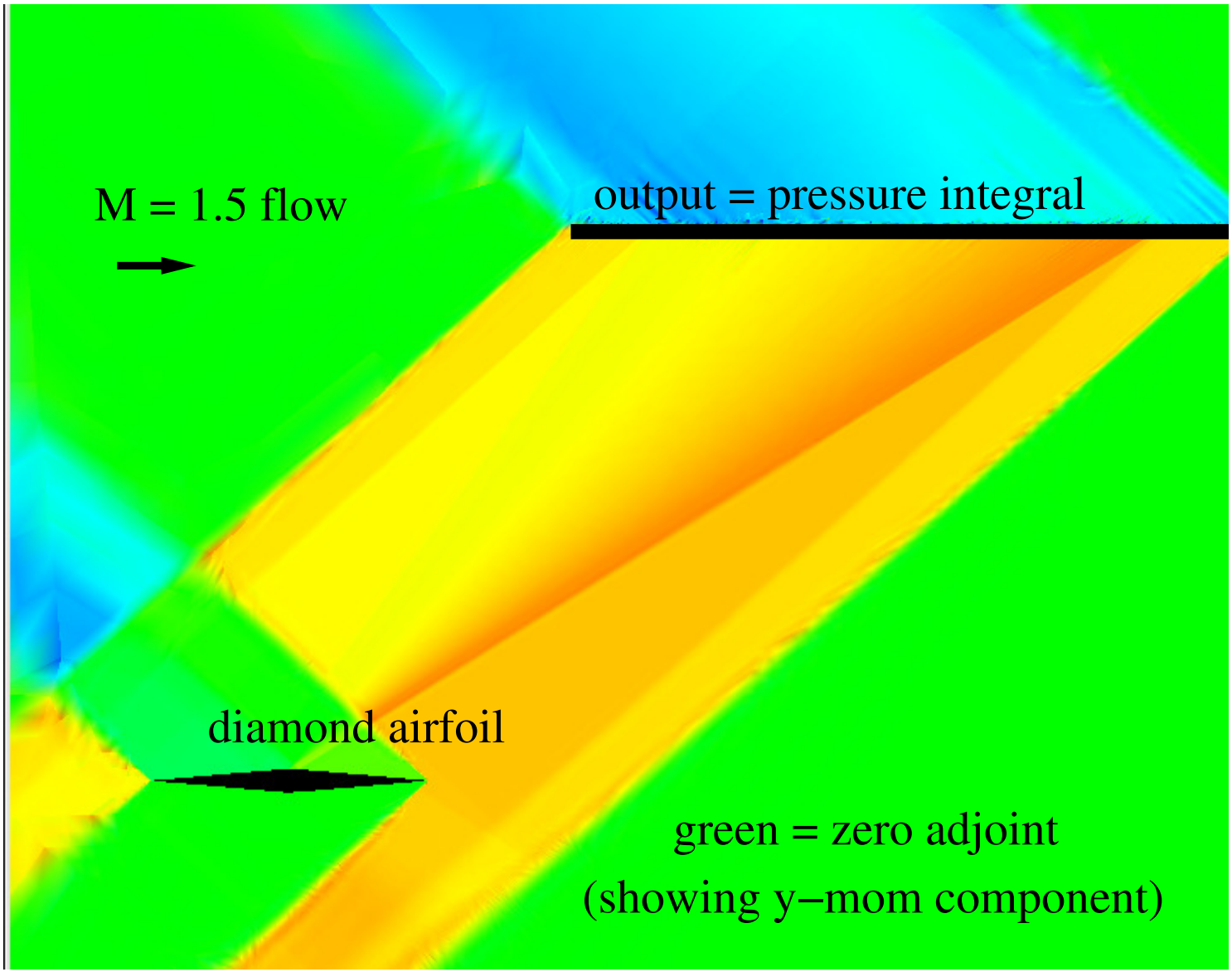
resolving for the state ...

Output sensitivity to residuals: the adjoint

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

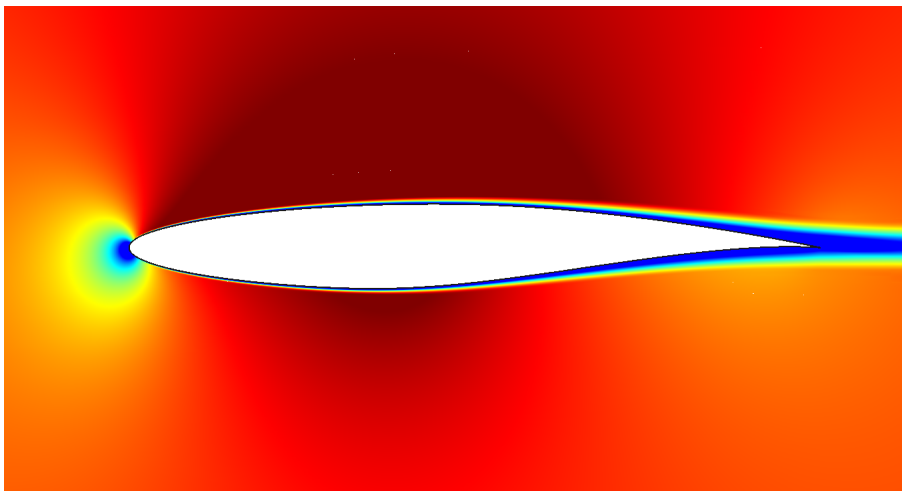


Sample steady adjoint solution

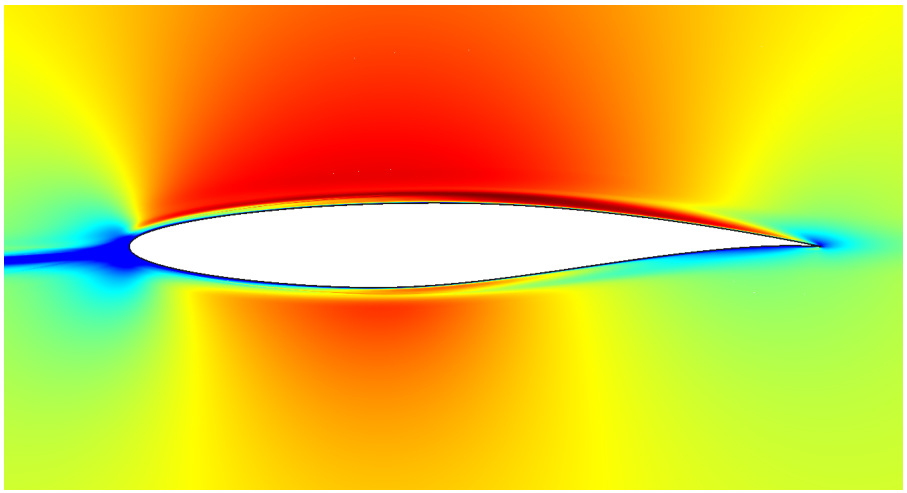


Another steady adjoint solution

RAE 2822, $M_\infty = 0.5$, $Re = 10^5$, $\alpha = 1^\circ$



x-momentum primal state



cons. of x-mom drag adjoint

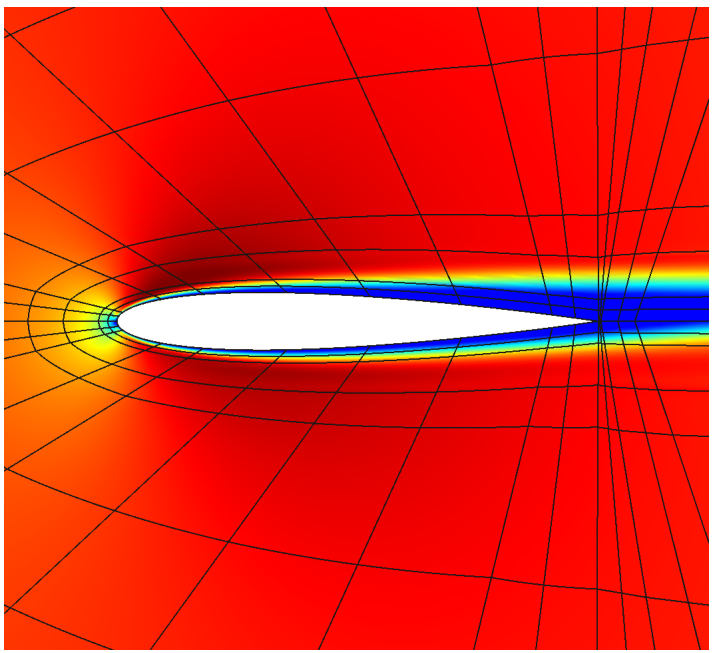
- Adjoint shares similar qualitative features with primal
- Note wake “reversal” in adjoint solution
- The discrete adjoint solution approximates the continuous adjoint when the discretization is *adjoint consistent*

Adjoint verification

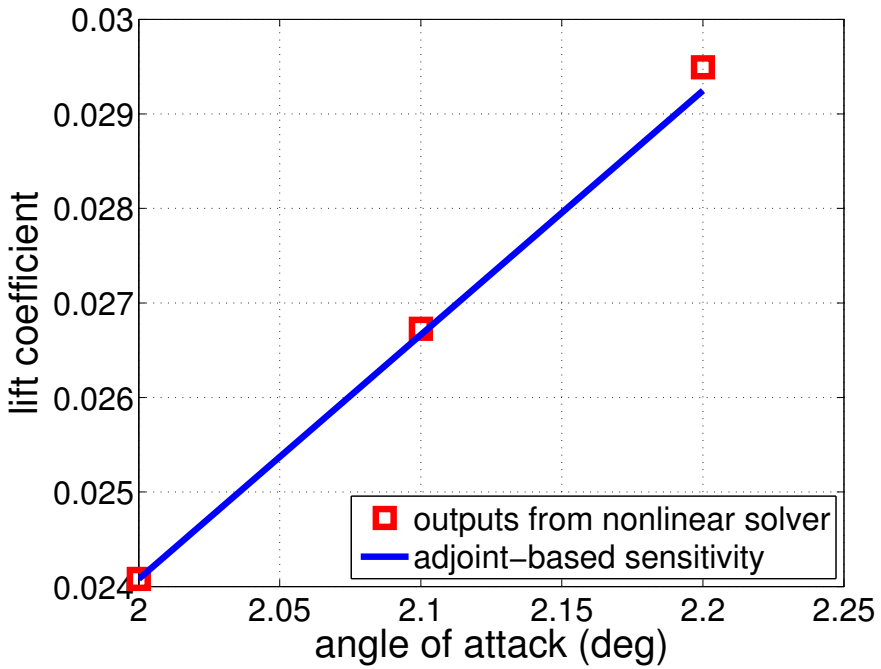
- We can verify the discrete adjoint with a sensitivity analysis,

$$\frac{dJ}{d\alpha} = \mathbf{\Psi}^T \frac{\partial \mathbf{R}}{\partial \alpha} + \frac{\partial J}{\partial \alpha}$$

- Compare to finite-difference sensitivity calculation
- Example: NACA 0012 airfoil in $Re = 5000$ flow



Mach number contours



Lift coefficient sensitivity

Adjoint implementation

- The discrete adjoint, $\mathbf{\Psi}$, is obtained by solving a linear system
- This system involves linearizations about the primal solution, $\mathbf{U}$, which is generally obtained first
- When the full Jacobian matrix, $\frac{\partial \mathbf{R}}{\partial \mathbf{u}}$, and an associated linear solver are available, the transpose linear solve is straightforward
- When the Jacobian matrix is not stored, the discrete adjoint solve is more involved: all operations in the primal solve must be linearized, transposed, and applied in reverse order
- In unsteady discretizations, the adjoint must be marched backward in time from the final to the initial state

Outline

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: $\delta J = J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h)$

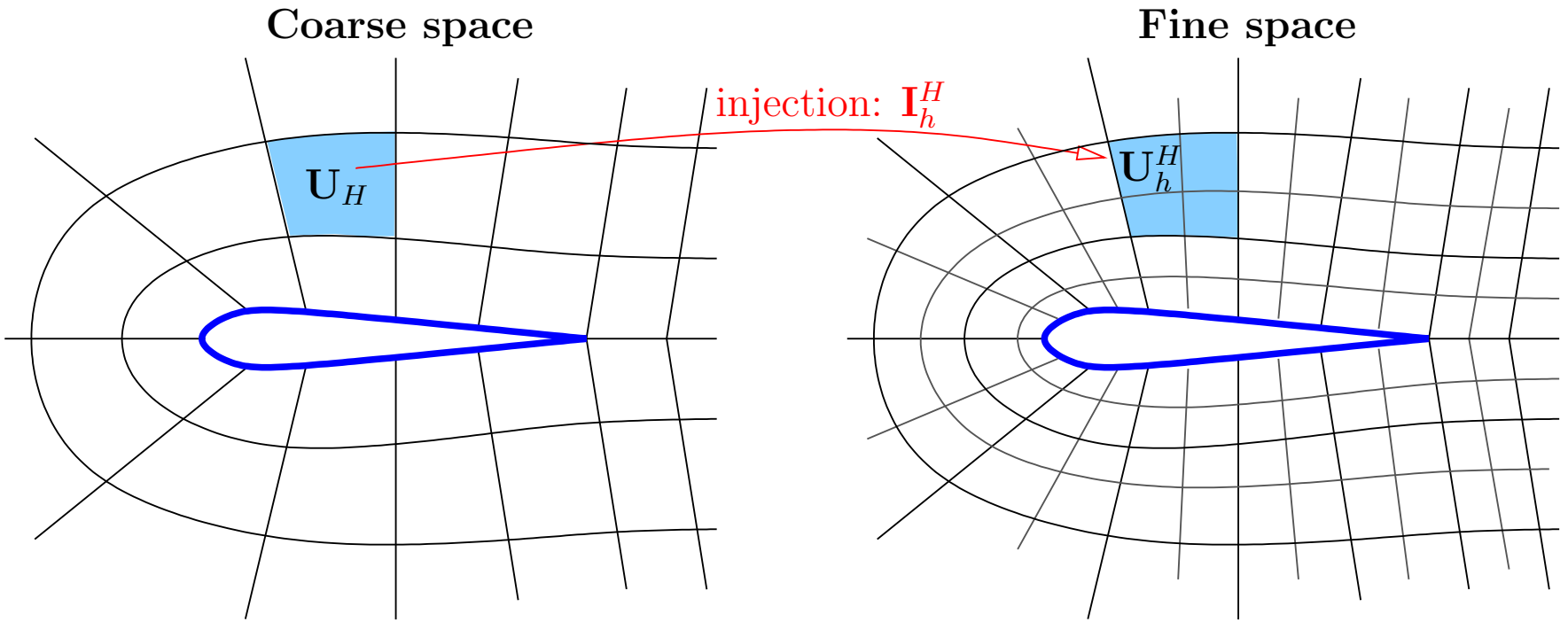
This is the difference in J relative to a finer discretization (h)

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

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

Fine-space injection

- The fine space can arise from h or p refinement
- Define an injection of the coarse state into the fine space



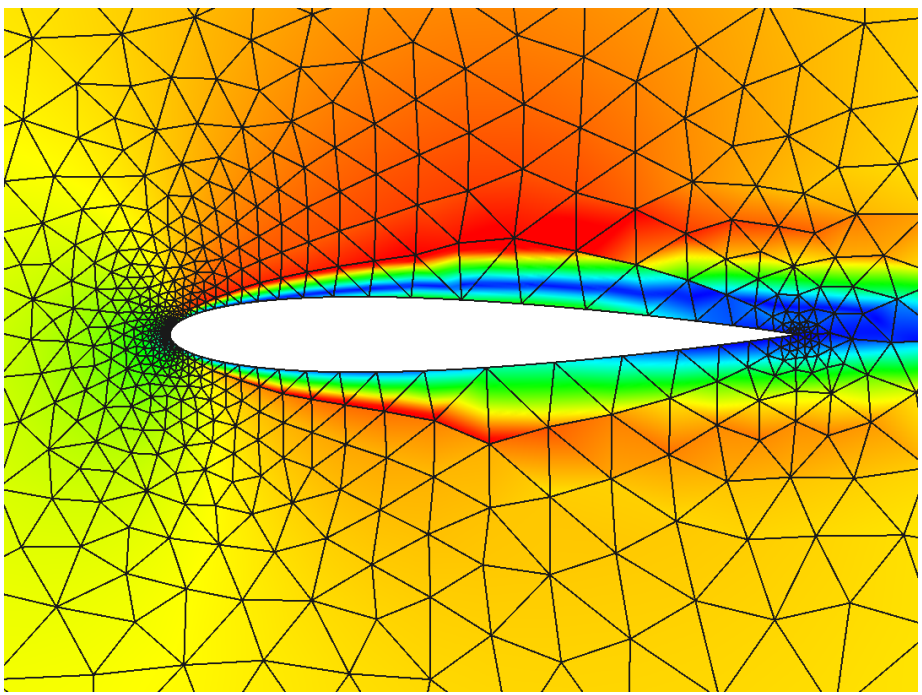
- $\mathbf{U}_h^H$ will generally not satisfy the fine-space equations,

$$\mathbf{R}_h(\mathbf{U}_h^H) \neq \mathbf{0}$$

Fine-space residuals

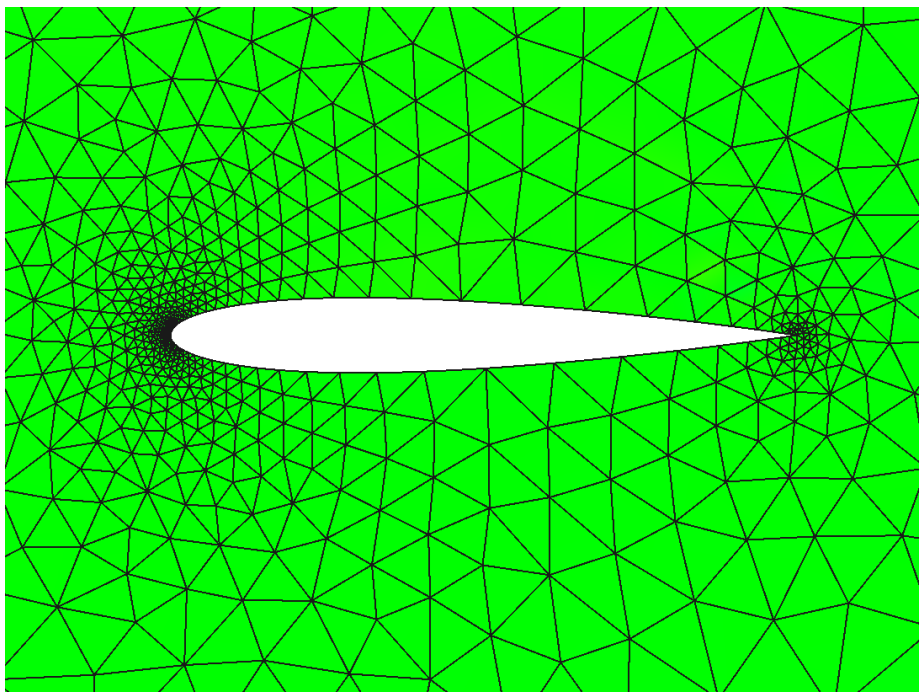
- A finer space (e.g. order enrichment) can uncover residuals in a converged solution
- Example: NACA 0012 at $\alpha = 2^\circ$ in $Re = 5000$, $M_\infty = 0.5$ flow

Coarse space state, $\mathbf{U}_H$



$p_H = 1$

Coarse space residual, $\mathbf{R}_H(\mathbf{U}_H)$

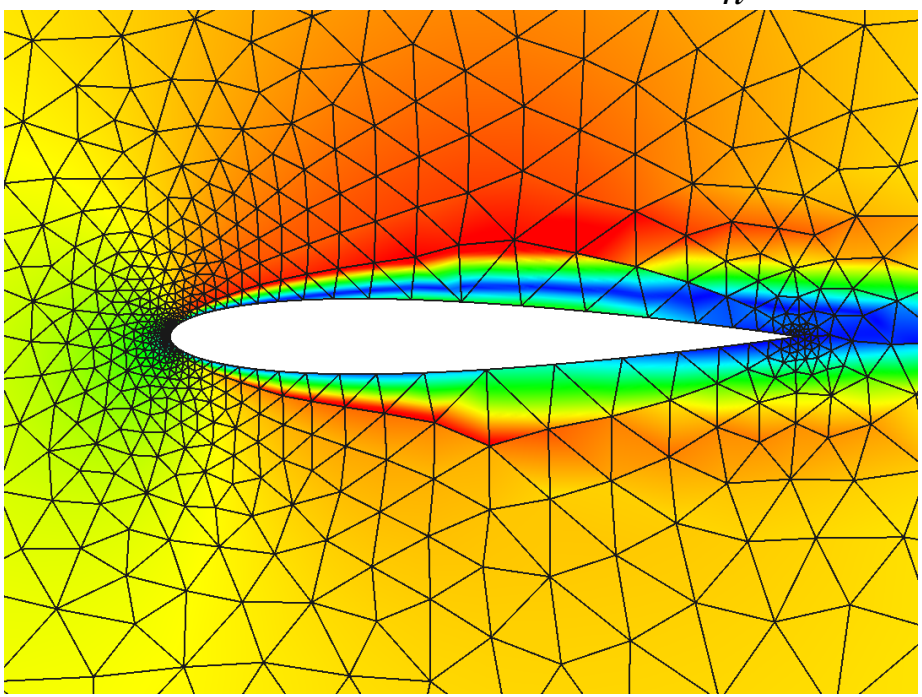


Zero as expected

Fine-space residuals

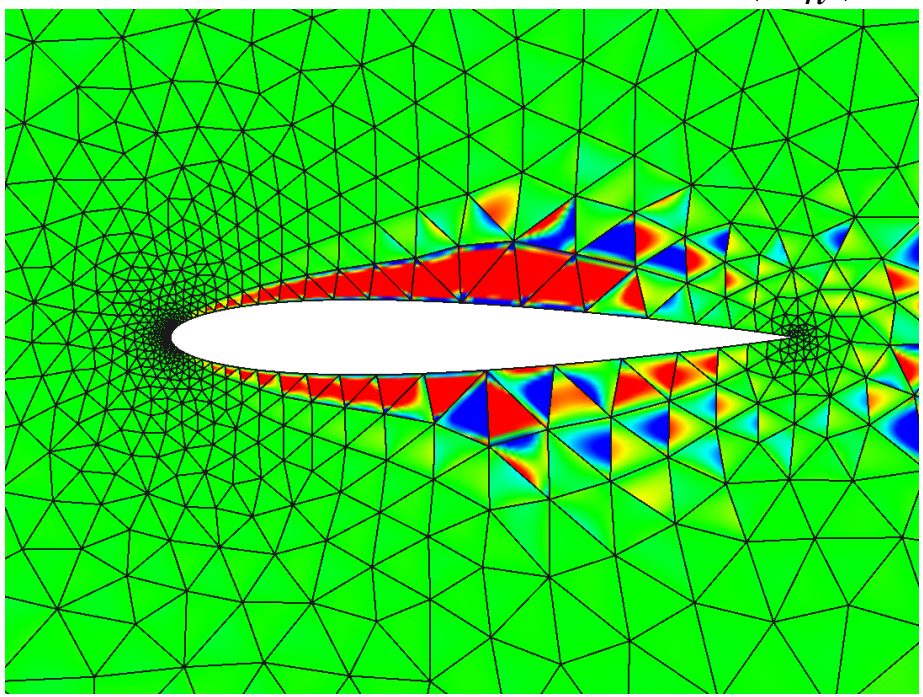
- A finer space (e.g. order enrichment) can uncover residuals in a converged solution
- Example: NACA 0012 at $\alpha = 2^\circ$ in $Re = 5000$, $M_\infty = 0.5$ flow

Injected state, $\mathbf{U}_h^H$



$p_h = 2$

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



Nonzero: new info

The adjoint-weighted residual

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

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

- The fine-space adjoint, $\mathbf{\Psi}_h$, then tells us to expect an output perturbation of

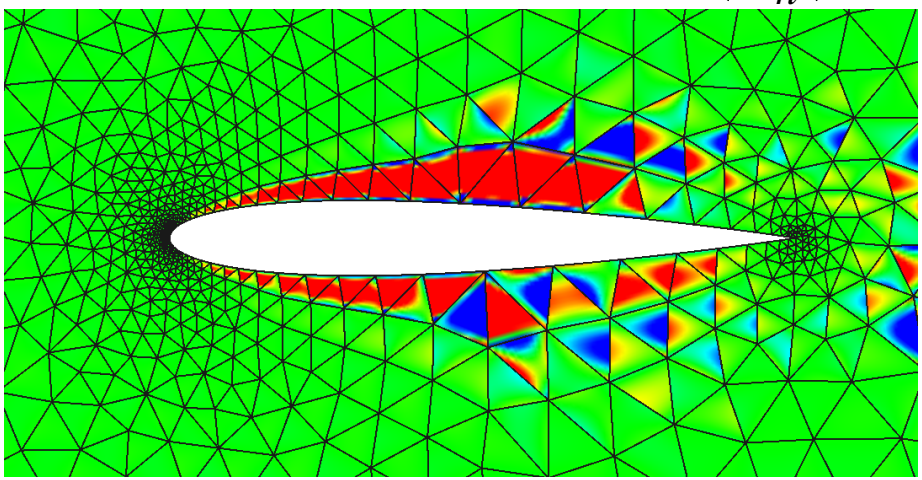
$$\underbrace{J_h(\mathbf{U}_h^H) - J_h(\mathbf{U}_h)}_{\approx \delta J} = \mathbf{\Psi}_h^T \delta\mathbf{R}_h = -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)$$

- This equation assumes small perturbations (e.g. if nonlinear)
- In summary, we have an *adjoint-weighted residual* error estimate,

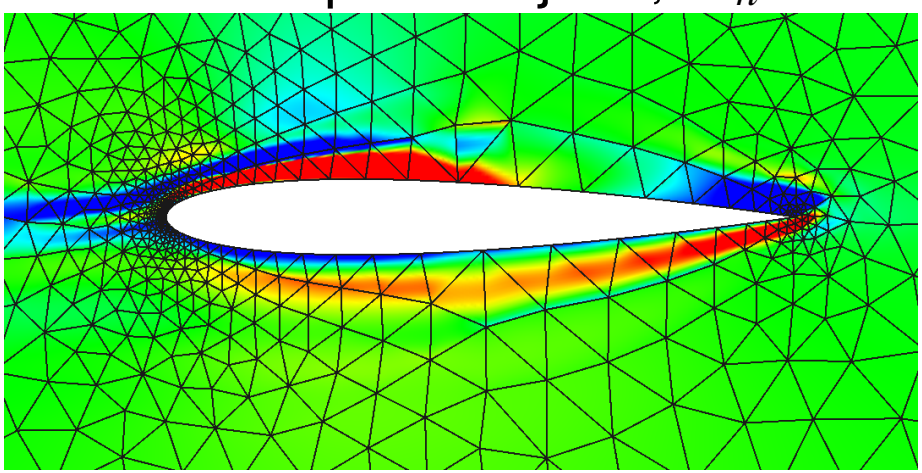
$$\boxed{\delta J \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H)}$$

Adjoint-weighted residual example

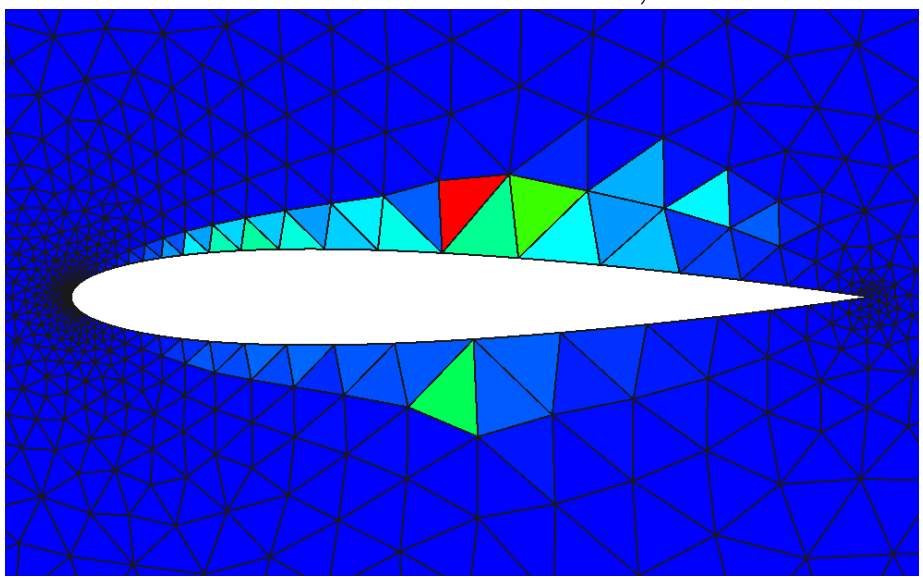
Fine space residual, $\mathbf{R}_h(\mathbf{U}_h^H)$



Fine space adjoint, Ψ_h



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



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

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

Two more definitions

Corrected output

$$J_H^{\text{corrected}} = J_H - \delta J$$

- Should converge faster than J_H
- Remaining error = error left in corrected output

Error effectivity

$$\eta_H = \frac{J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h)}{J_H(\mathbf{U}_H) - J}$$

- J = exact output
- We want η_H close to 1
- Effectivity is affected by choice of fine space

Error convergence tests

- Expect

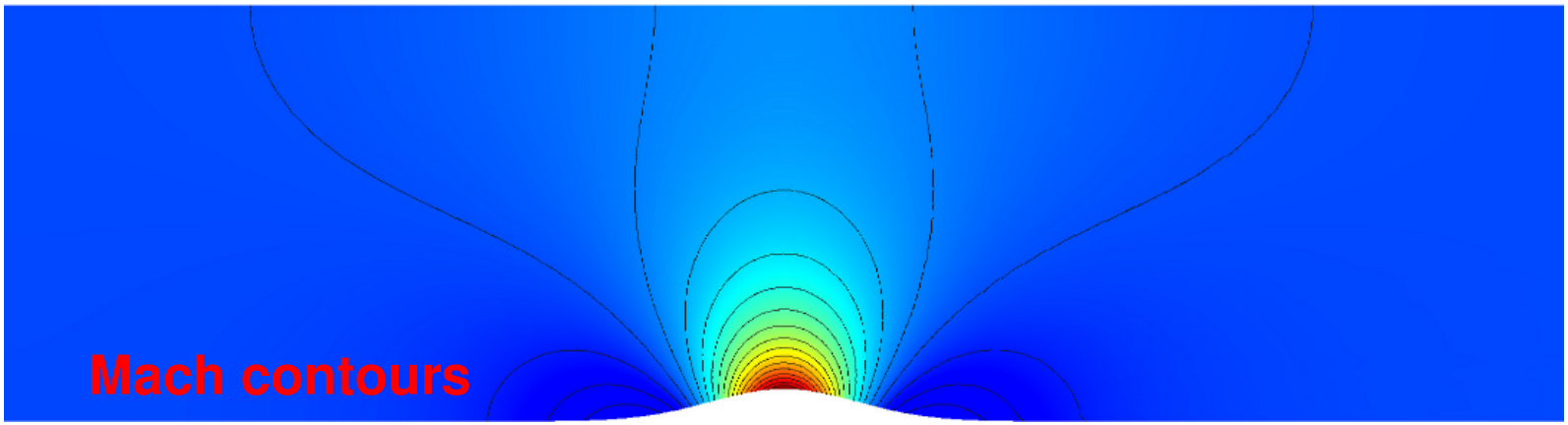
$$\text{error} = Ch^k, \quad \text{as } h \to 0$$

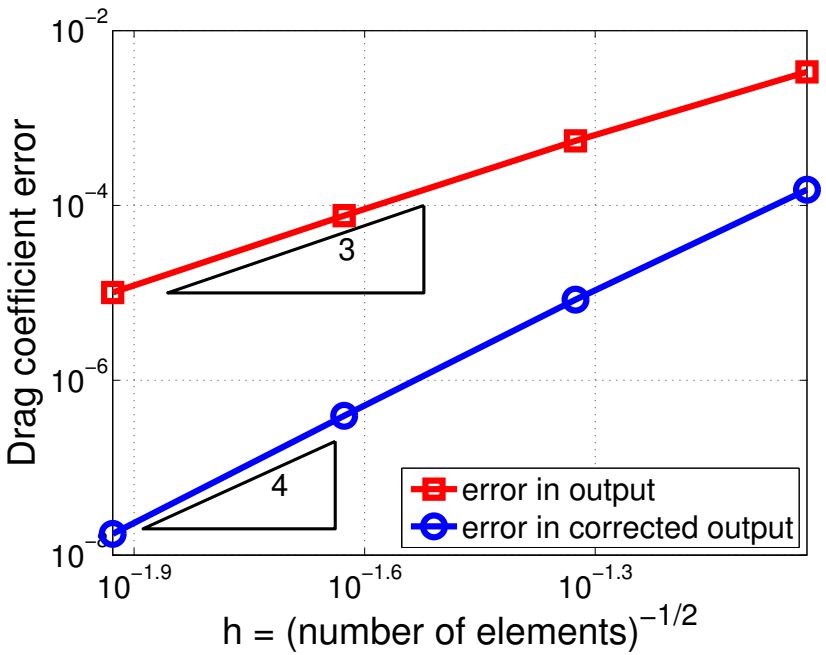
- k = rate of convergence
- h = measure of element size (precise value not important)
- For 2D uniform-refinement studies, can use $h = \sqrt{1/N_e}$
- Taking a log of the above equation,

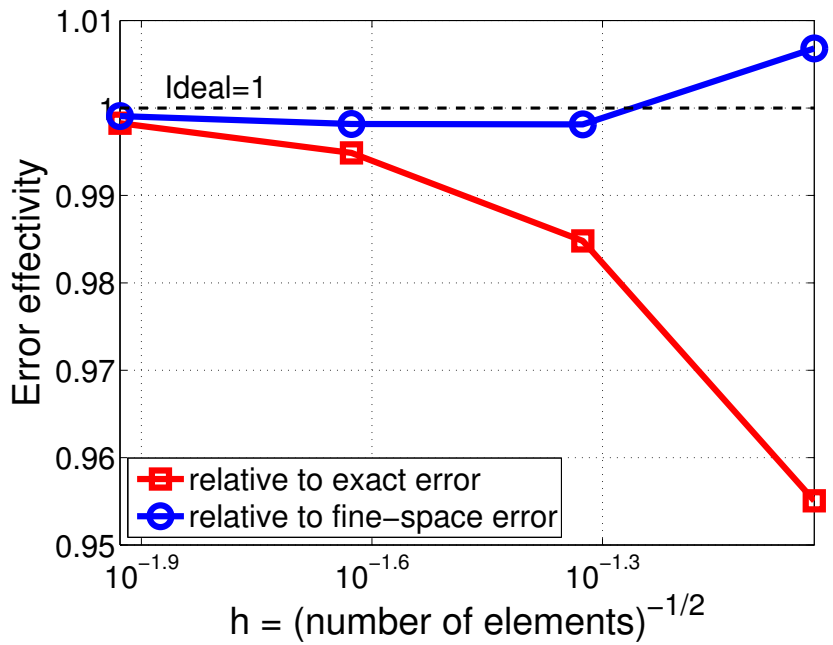
$$\log(\text{error}) \quad = \quad \log C + k \log\left(\sqrt{\frac{1}{N_e}}\right)$$

- We can measure k by plotting log(error) versus log(h)

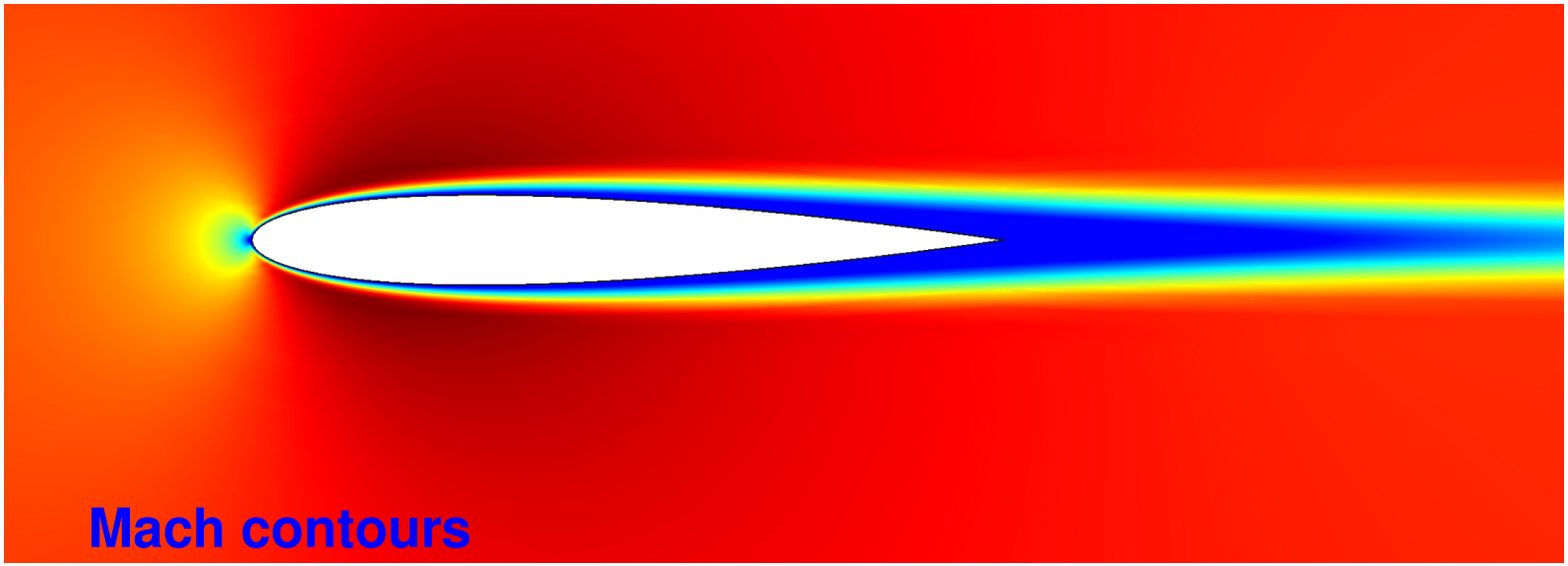
Drag error in inviscid flow over a bump

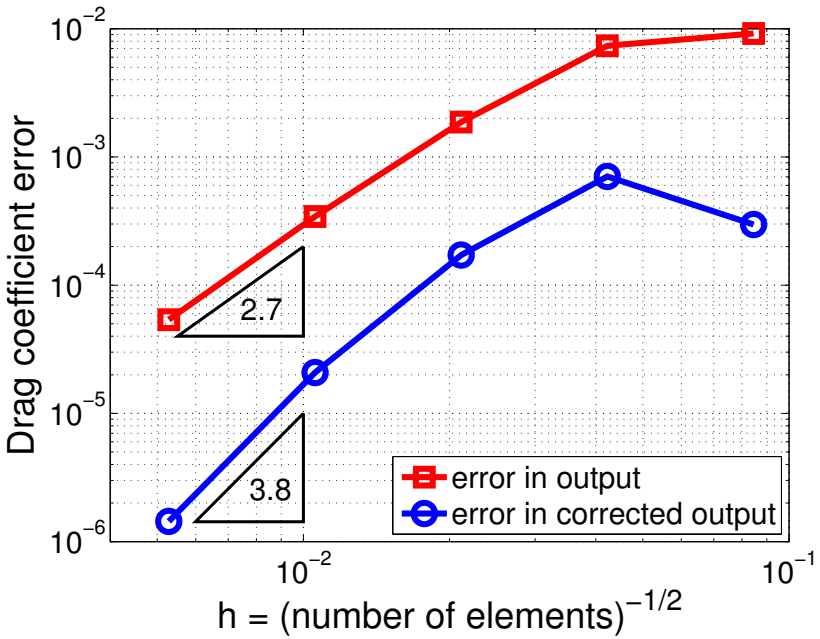


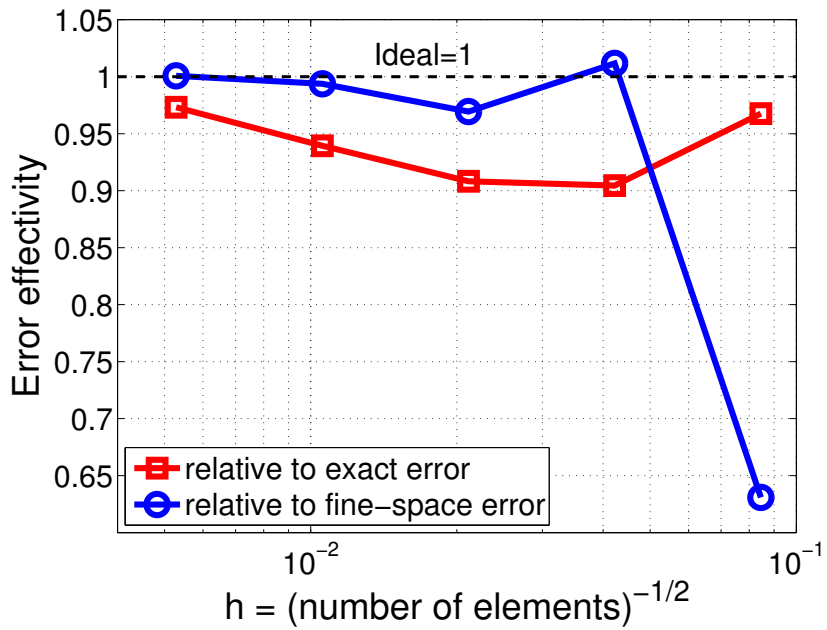




Drag error in viscous flow over an airfoil





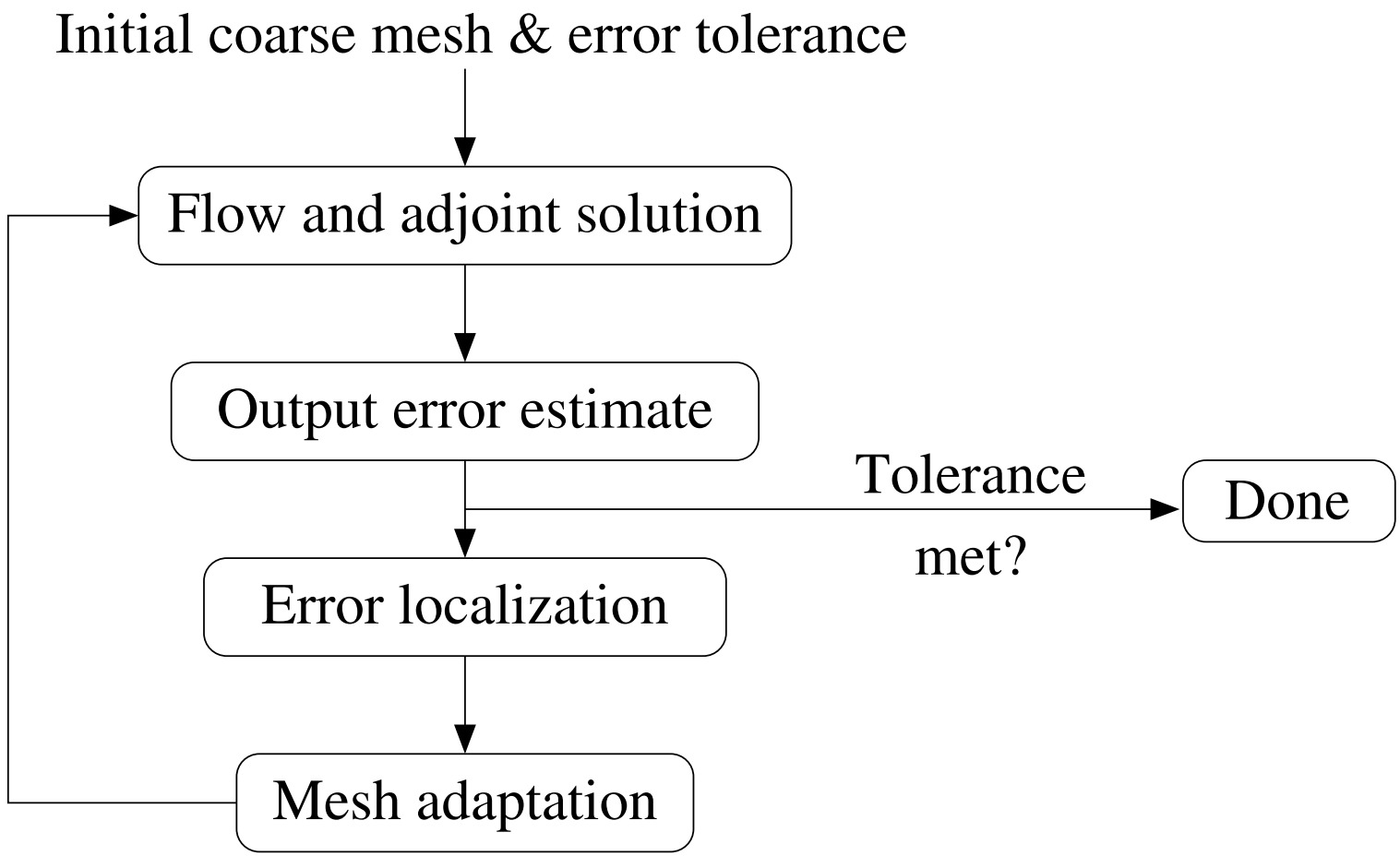


Error estimation summary

1. Solve the coarse-discretization forward and adjoint problems: $\mathbf{U}_H$ and $\mathbf{\Psi}_H$
2. Pick a fine discretization "h" (mesh refinement or order enrichment)
3. Calculate or approximate $\mathbf{\Psi}_h$ = adjoint on the fine space
4. Project $\mathbf{U}_H$ onto the fine discretization and calculate the residual $\mathbf{R}_h(\mathbf{U}_h^H)$
5. Weight the fine-space residual with the fine-space adjoint to obtain the output error estimate
6. The computed output error δJ is an estimate of the true error, not a bound

Outline

Mesh adaptation



Error localization

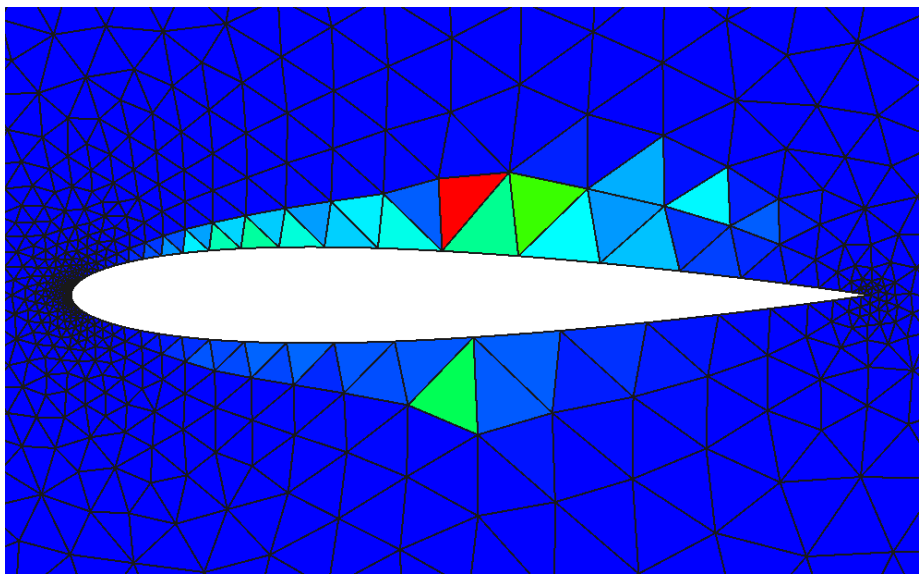
- Recall that the adjoint-weighted residual expression for the output error involves a sum over elements (e)

$$J_H(\mathbf{U}_H) - J_h(\mathbf{U}_h) \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) \quad = \quad -\sum_e \mathbf{\Psi}_{he}^T \mathbf{R}_{he}(\mathbf{U}_h^H)$$

- The absolute-value of each element's contribution to the error is the **error indicator** on that element

$$\epsilon_e \quad \equiv \quad \left| \mathbf{\Psi}_{he}^T \mathbf{R}_{he}(\mathbf{U}_h^H) \right|$$

Right : plot of error indicator for a viscous DG simulation, $p_H = 1$, $p_h = 2$



Output-based mesh adaptation

Motivating ideas

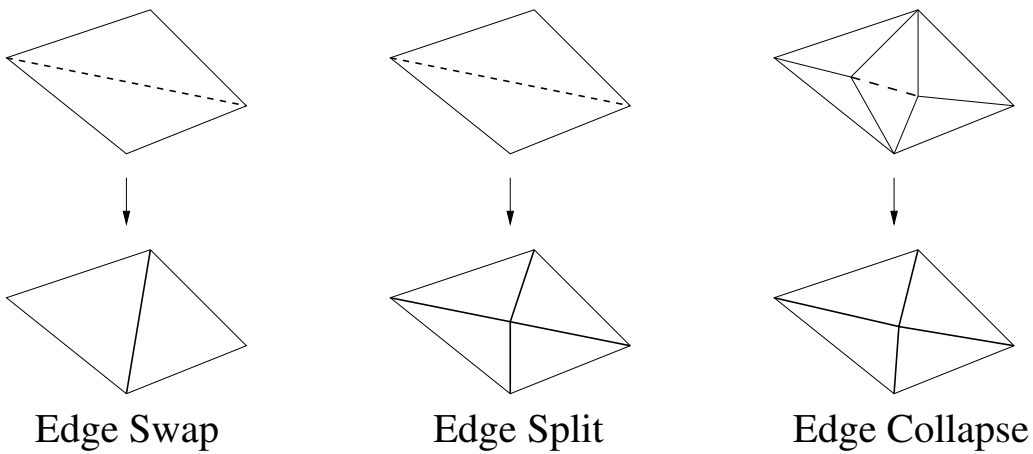
- The error indicator (ϵ_e) identifies elements with large adjoint-weighted residuals
- Locally refining a mesh reduces local residuals
- So we can reduce the output error by refining those elements that have a high ϵ_e

Adaptation choices

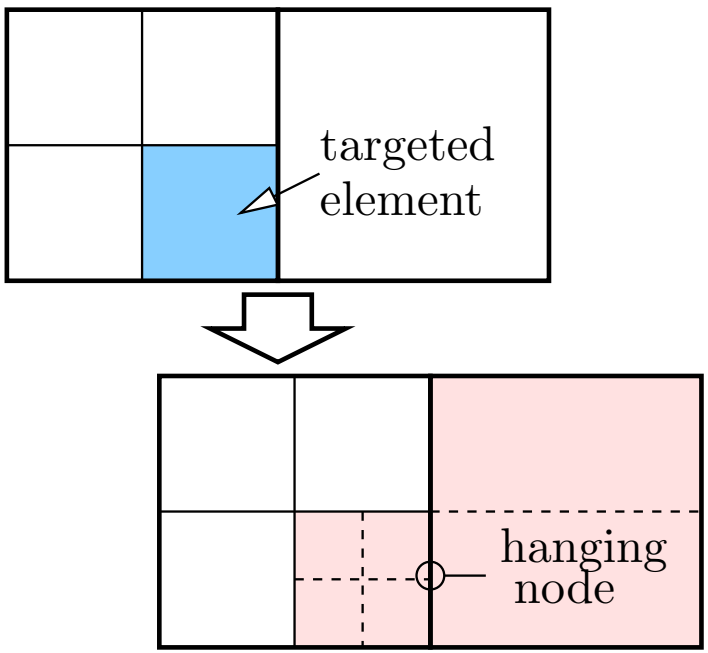
- Local refinement versus global re-meshing
- Which/how many elements should be targeted?
- Isotropic versus anisotropic refinement
- h, p, or hp mechanics
- Should coarsening be allowed?

Local mesh modification

- Modify the mesh incrementally (mesh generation is hard)
- Often more robust than global re-meshing
- With node movement, can be flexible for unstructured meshes
- Hanging nodes easily supported in DG



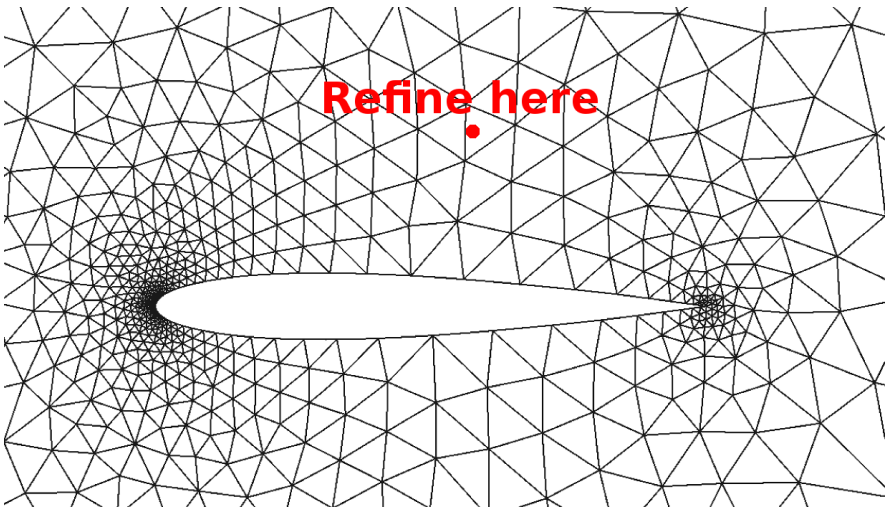
Unstructured local mesh operators



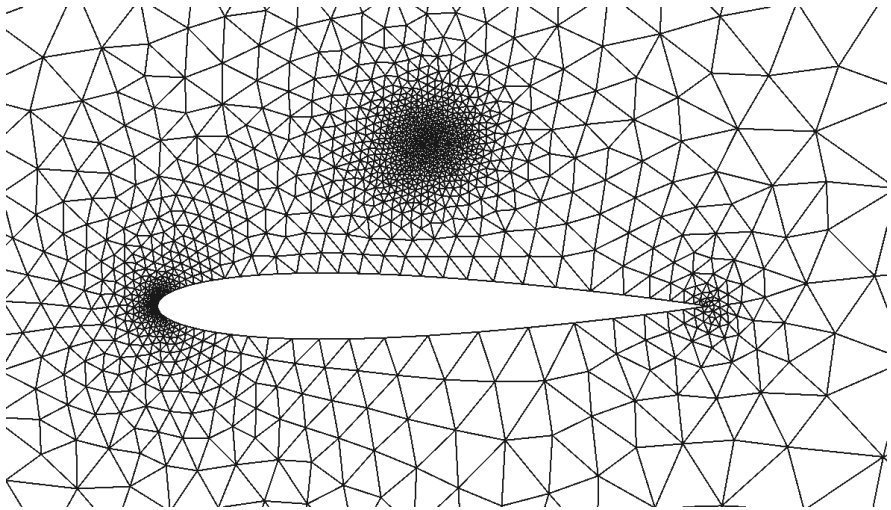
Hanging-node refinement

Global re-meshing

- Entire mesh is re-generated
- Current mesh still plays a role in defining a *Riemannian metric*
- Useful software in 2D: Bi-dimensional Anisotropic Mesh Generator (BAMG)
- Example of refinement near a single point:



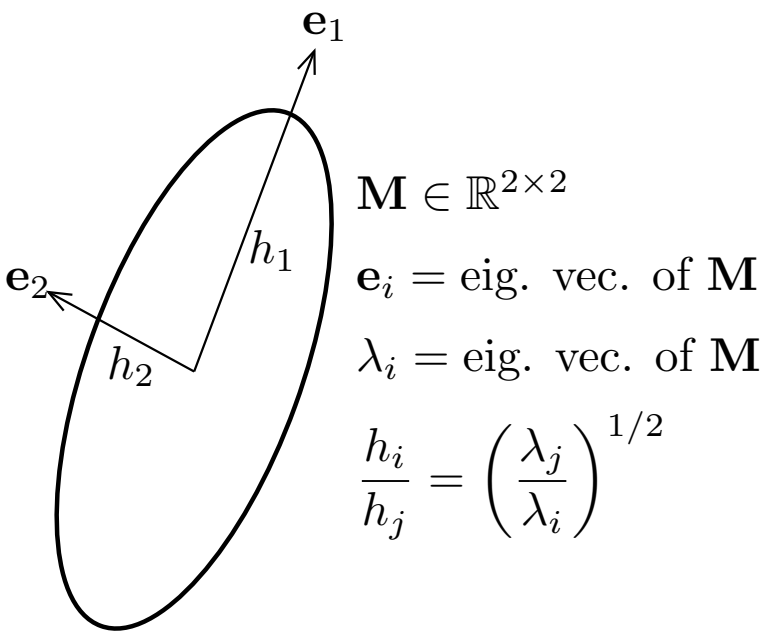
Original mesh



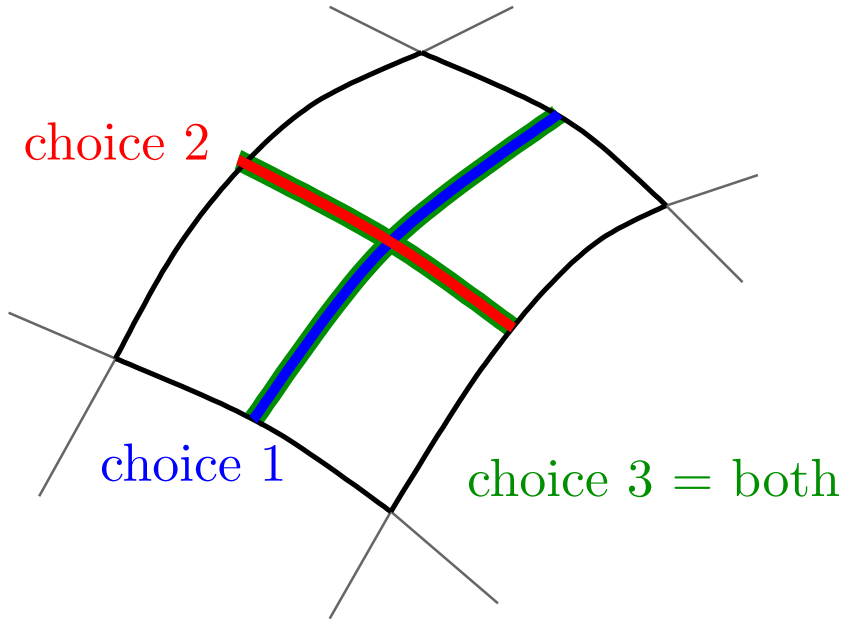
After refinement

Incorporating anisotropy

- Crucial for high-Reynolds number simulations, esp. in 3D
- Can come in via a metric or discrete hanging-node "slices"



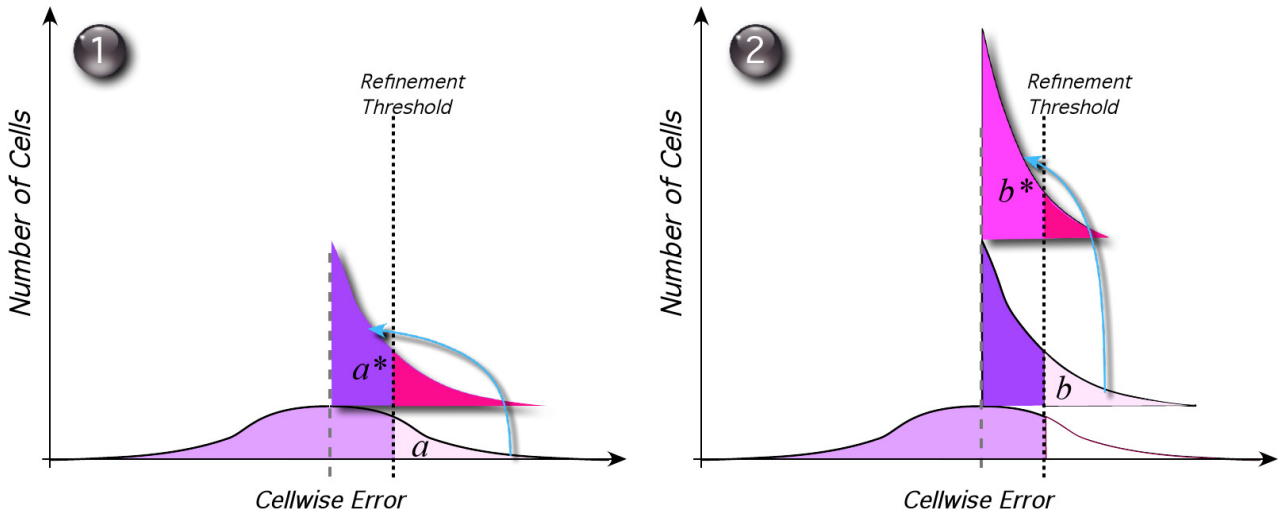
Anisotropic metric in 2D



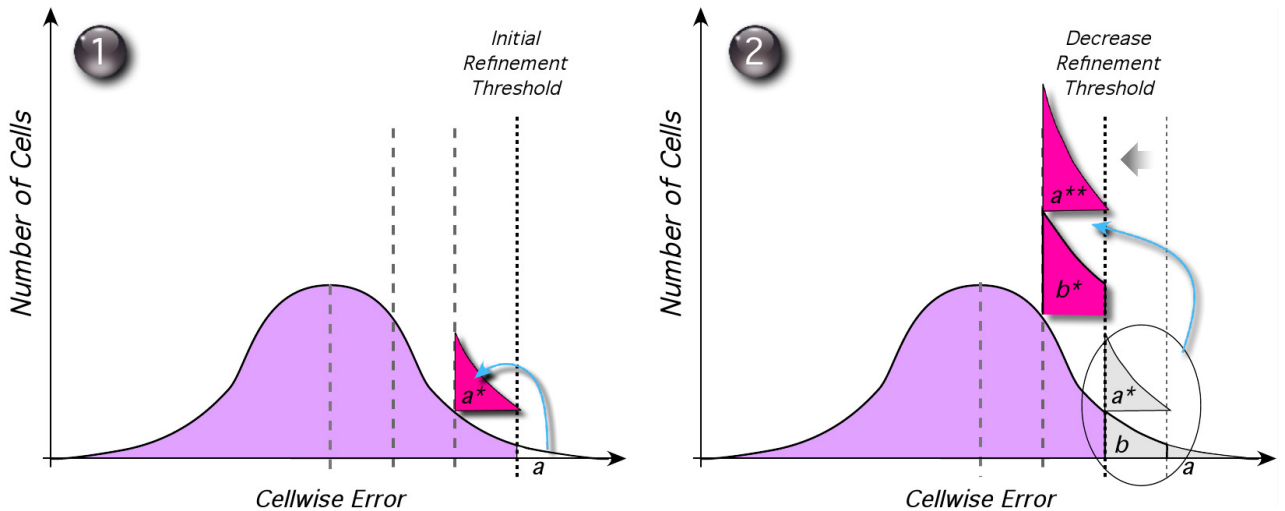
Hanging-node choices

- Assess need for anisotropy by
  - Looking at derivatives of a scalar quantity (Mach number)
  - Solving local sub-problems to determine impact of anisotropy directly on the output error

Targeting strategies [Nemec *et al*, 2008]



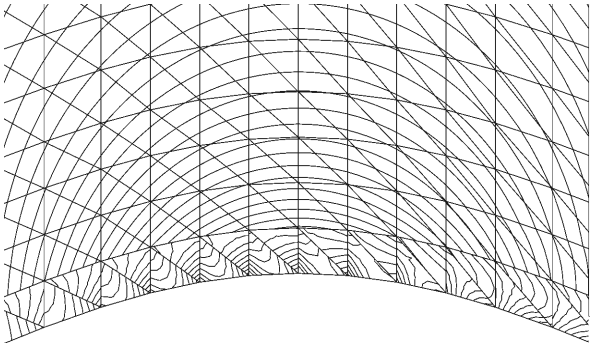
Constant threshold: refine all elements above a constant error indicator



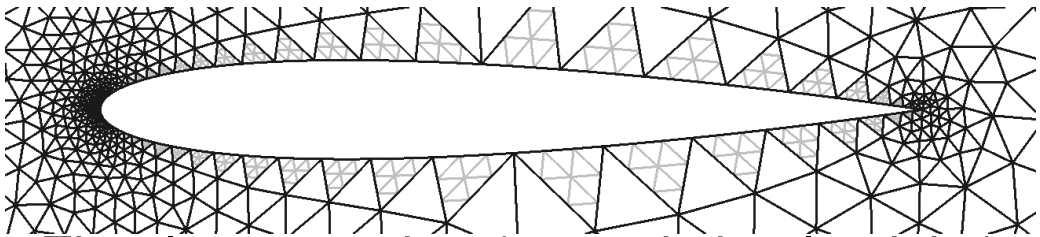
Decreasing threshold: threshold decreases with each iteration

Curved boundaries

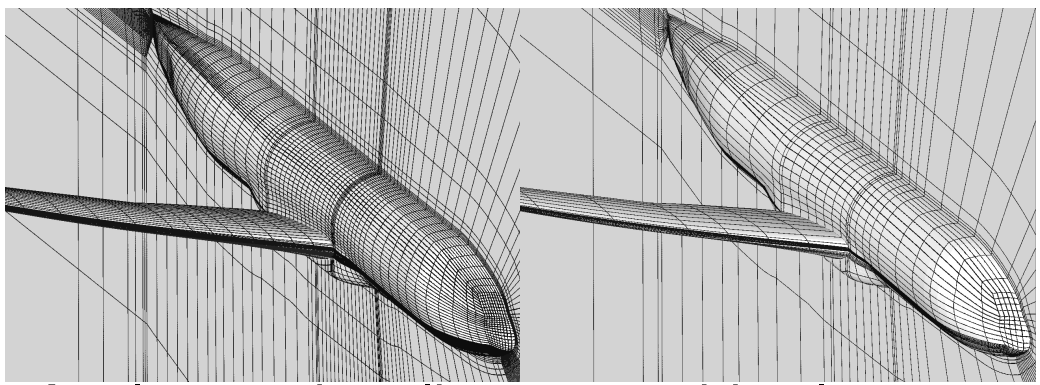
- DG needs an accurate representation of curved boundaries
- Curving elements is not easy
- Tangling is hard to avoid, especially in 3D anisotropic elements



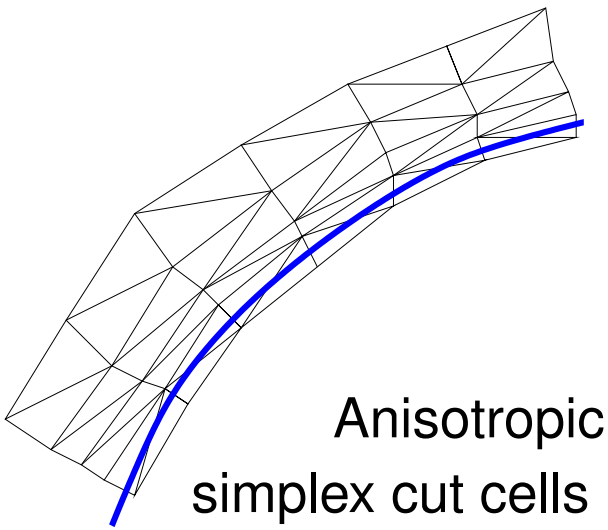
pressure contours
$p = 2$ Euler flow over a linear-element bump representation



First-layer curving (extend via elasticity)



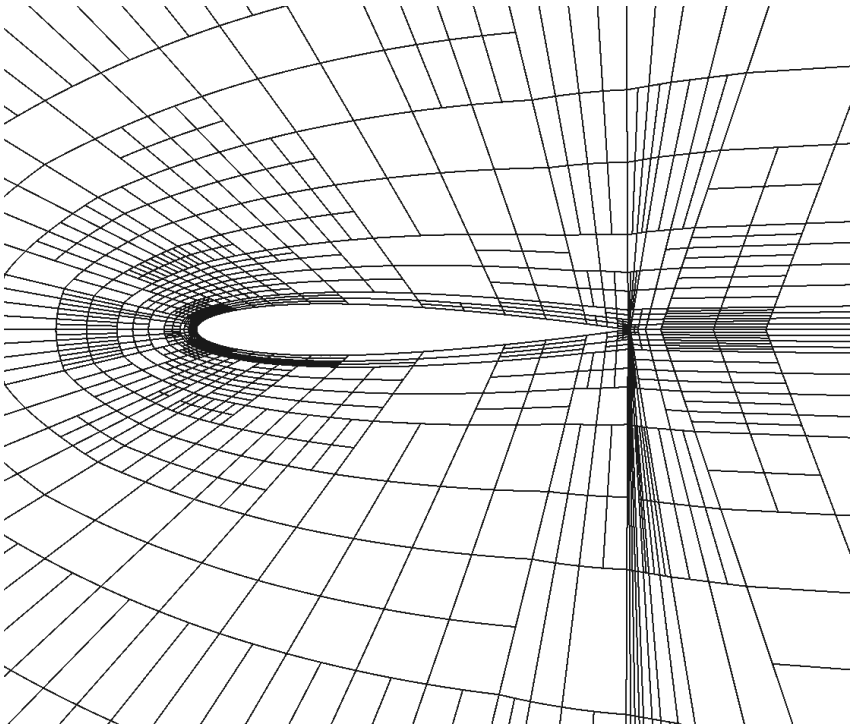
Agglomeration: linear $\rightarrow$ cubic elements



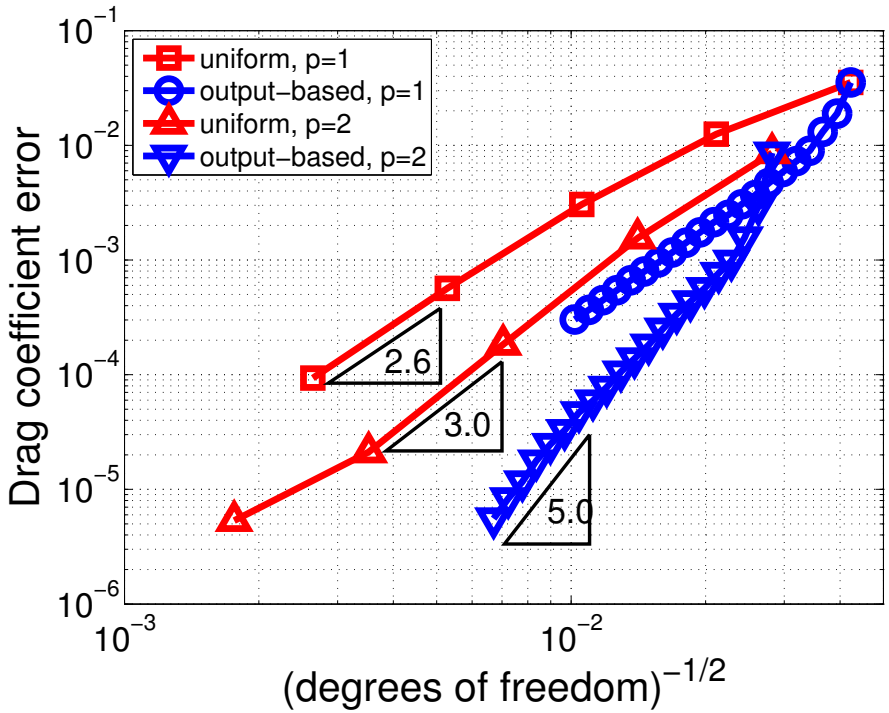
Anisotropic simplex cut cells

Inviscid flow over an airfoil

NACA 0012, Euler, $M_\infty = 0.5$, $\alpha = 2°$



Final drag-adapted mesh for $p = 2$

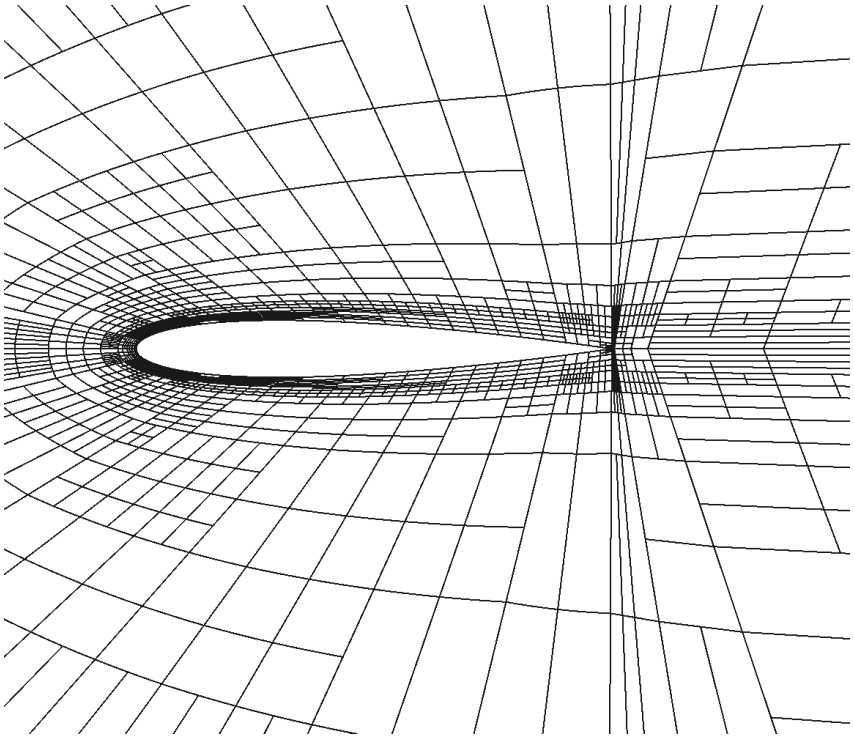


Drag convergence

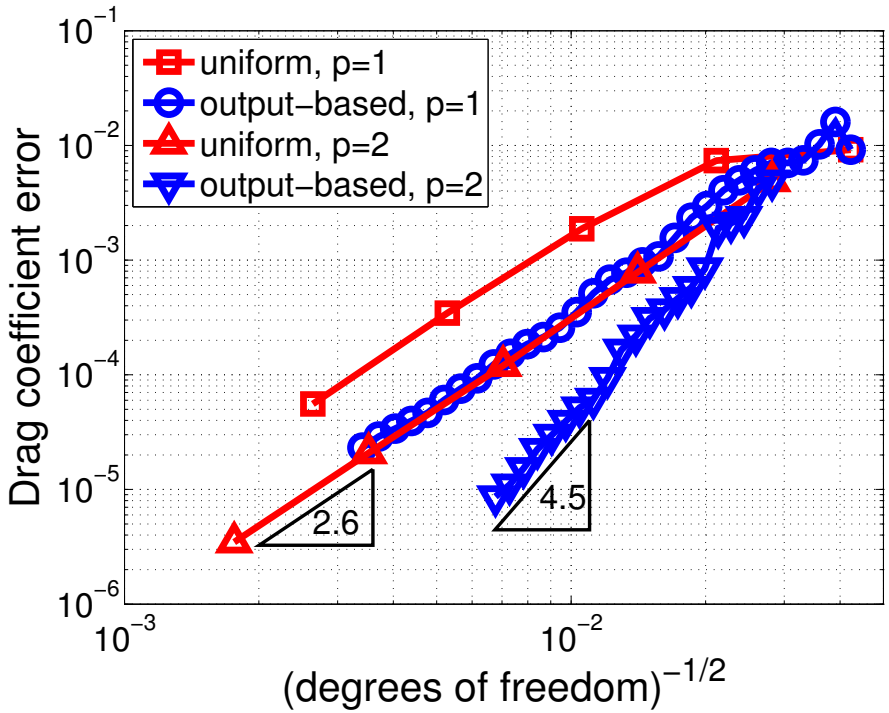
- We obtain Ψ_h approximately (adaptation unaffected)
- Adaptive refinement "quarantines" the trailing edge singularity and uncovers the superconvergent $2p + 1$ rate

Viscous flow over an airfoil

NACA 0012, Navier-Stokes, $M_\infty = 0.5$, $Re = 5000$, $\alpha = 0°$



Final drag-adapted mesh for $p = 2$

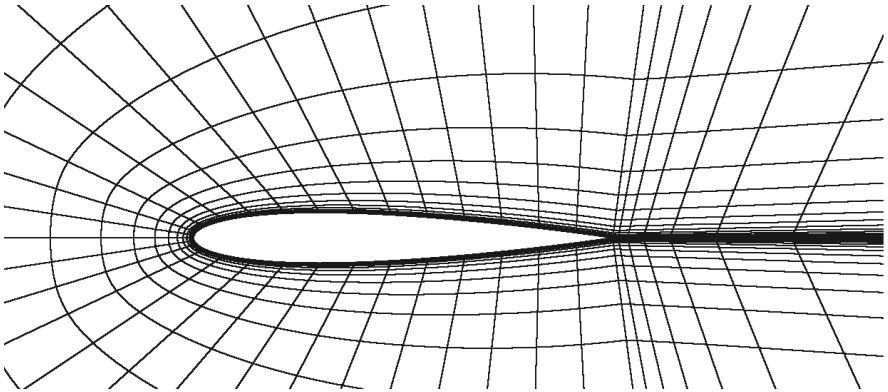


Drag convergence

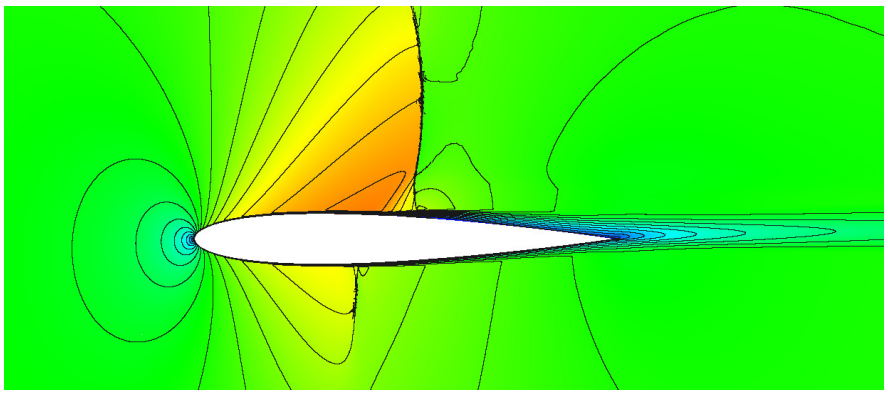
- Rate in uniform refinement limited by high-order singularities
- Adaptive refinement uncovers a superconvergent rate $(2p)$

Transonic RANS flow over an airfoil

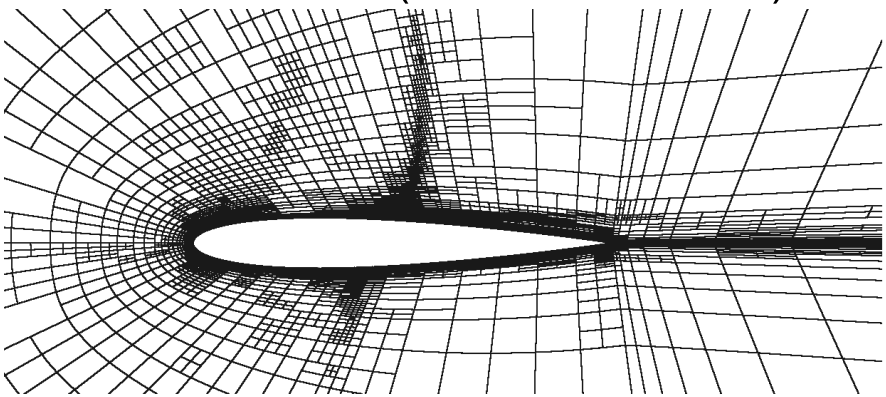
NACA 0012, Navier-Stokes + SA, $M_\infty = 0.8$, $Re = 100k$, $\alpha = 1.25°$



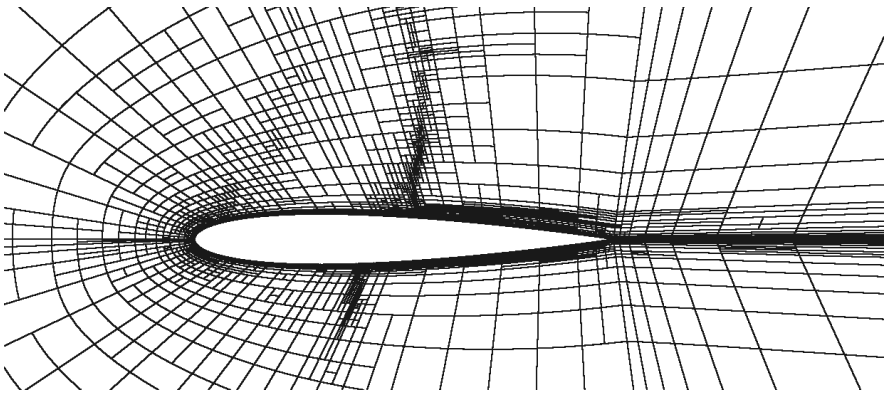
Initial mesh (1740 elements)



Mach number contours



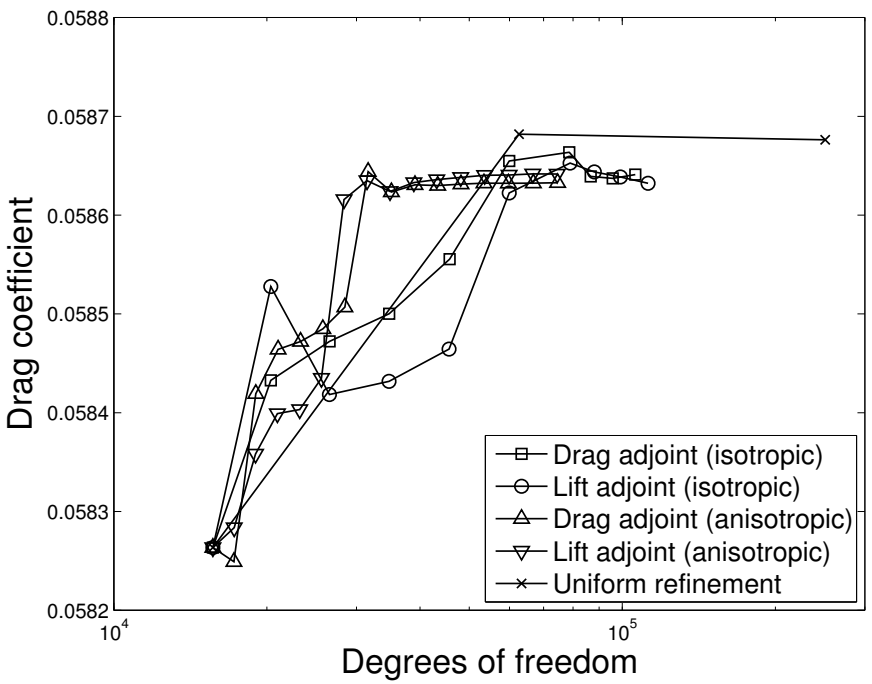
Adapted isotropic mesh



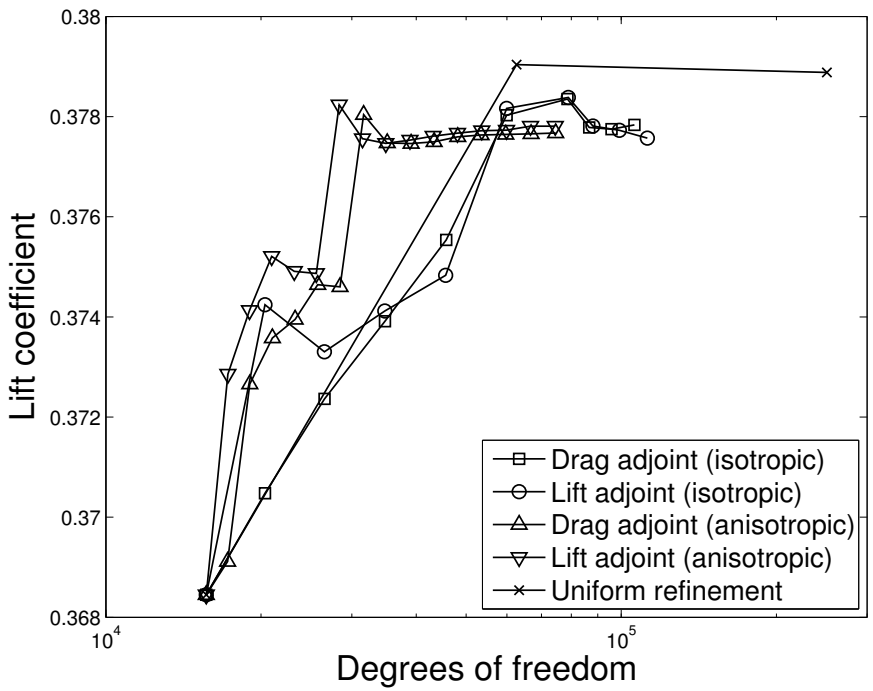
Adapted anisotropic mesh

Transonic RANS flow over an airfoil (ctd.)

- Fine space adjoint solved approximately
- Anisotropic adaptation driven by local solves on discrete refinement choices
- Outputs from uniform refinement overshoot exact values



Drag output

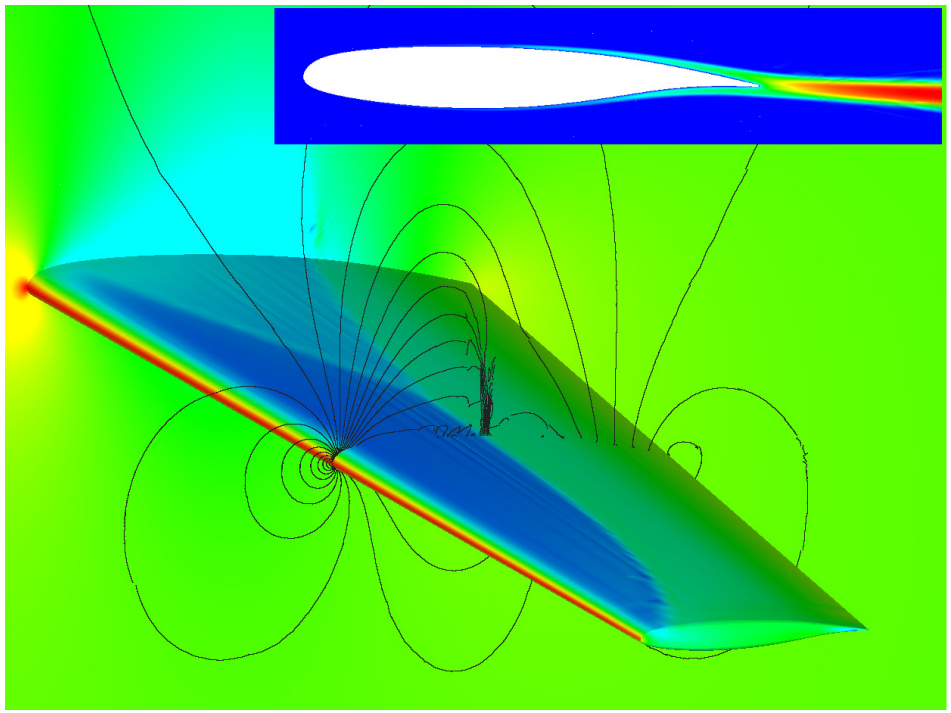


Lift output

Transonic RANS flow over a wing

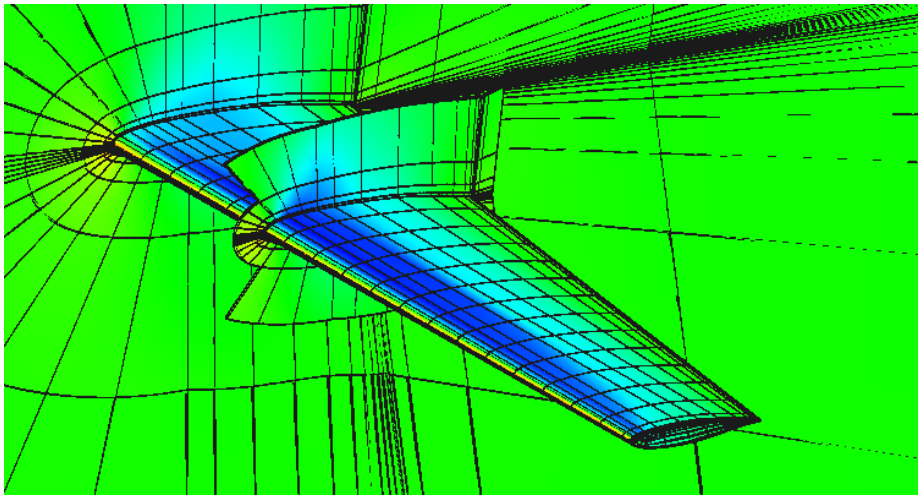
DPW III wing-alone case: $M_\infty = 0.76, Re = 5 \times 10^6$

- Initial mesh: cubic hex elements generated by agglomeration of linear multiblock meshes (first element $y^+ \approx 1$)
- Artificial viscosity shock capturing
- Spalart-Allmaras turbulence model with negative $\tilde{\nu}$ modification [Oliver & Allmaras]
- Drag-adaptive simulation using hp discrete choice algorithm (Ceze + Fidkowski, 2013)

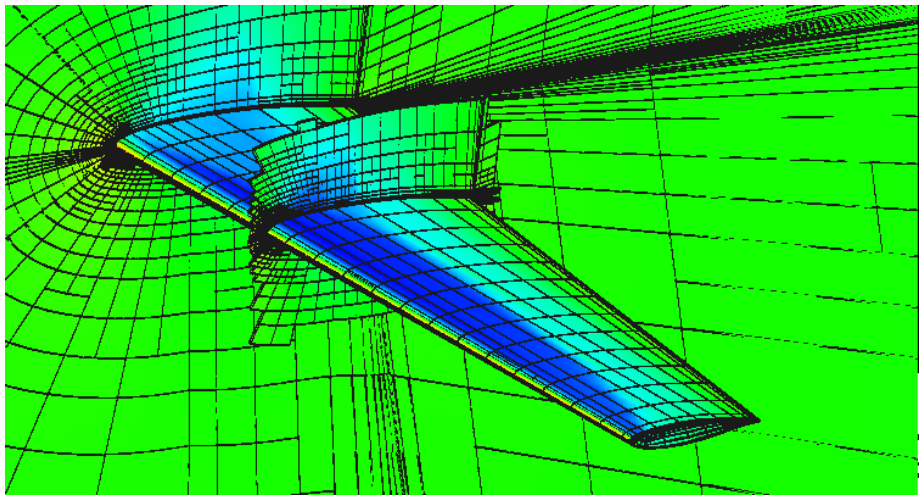


Contours of c_p and $\tilde{\nu}$

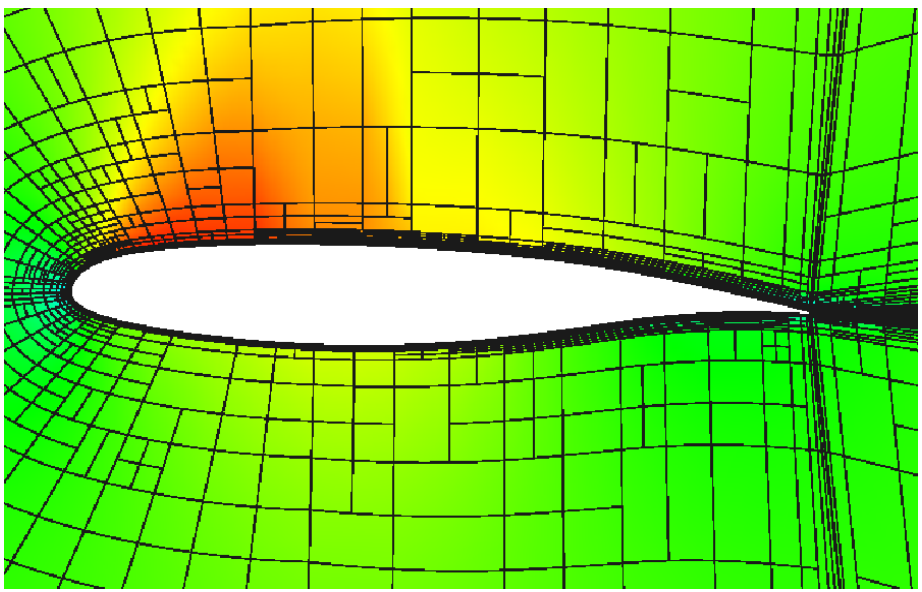
DPW wing: adapted meshes



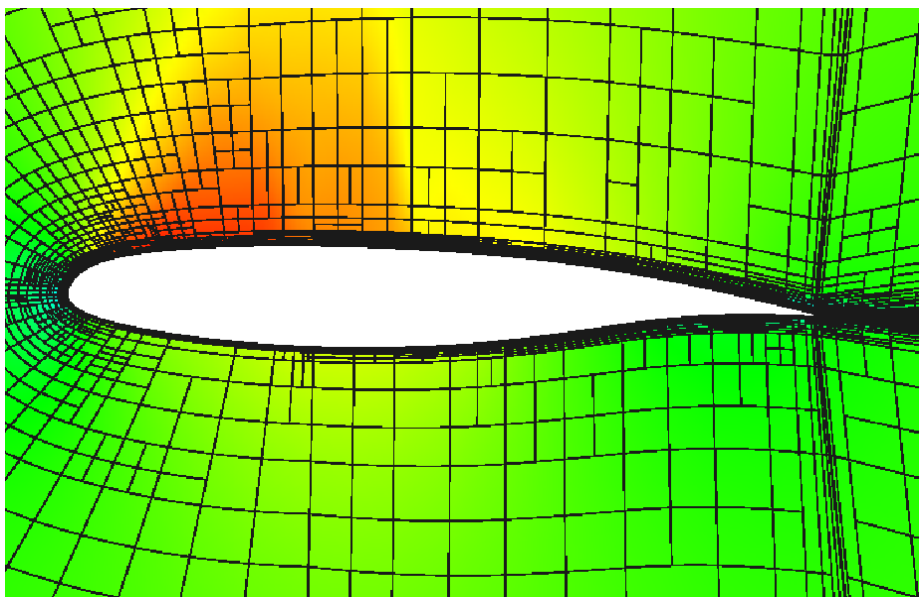
Original mesh, with c_p contours



7^{th} drag-adapted mesh

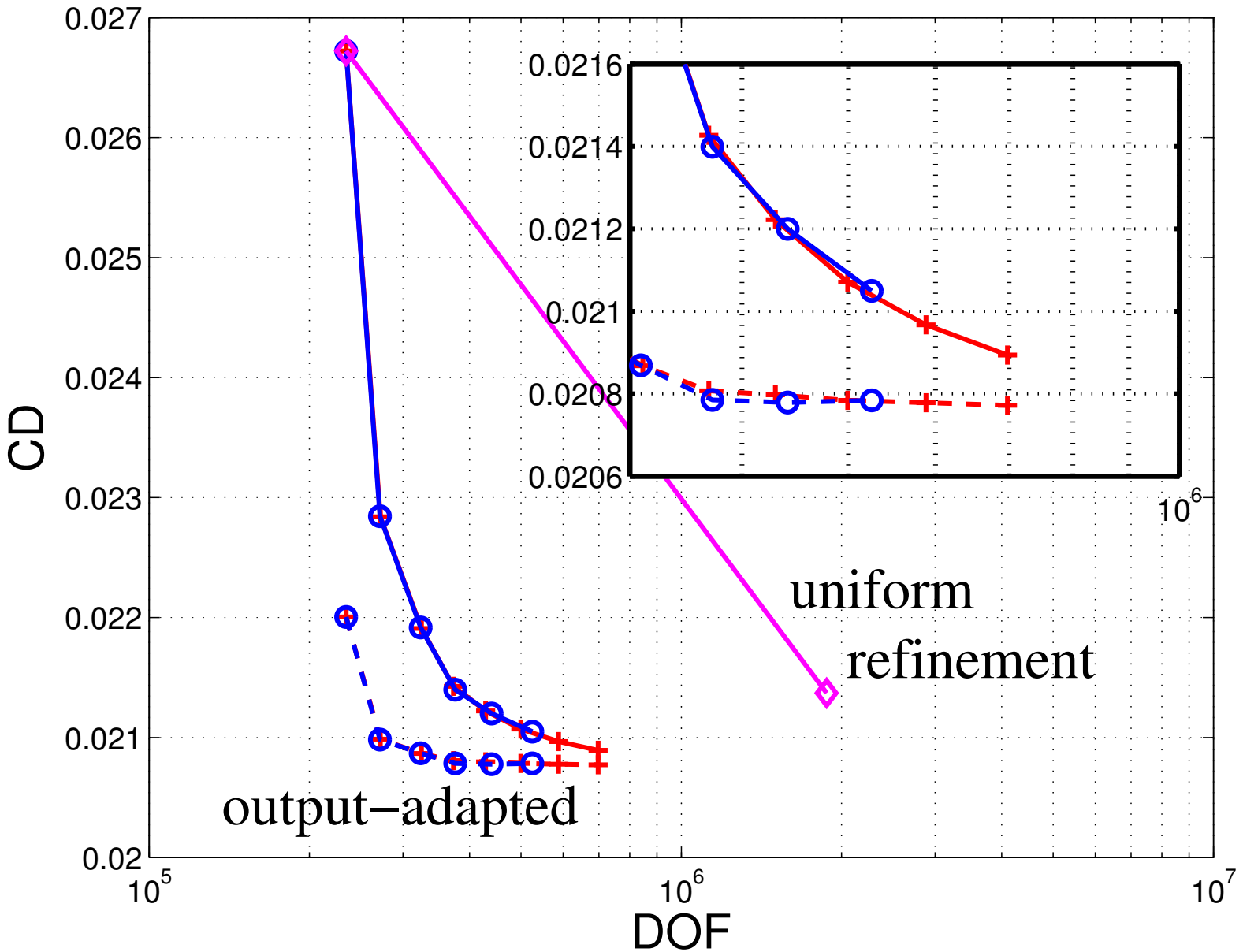


Mach/mesh using DOF cost

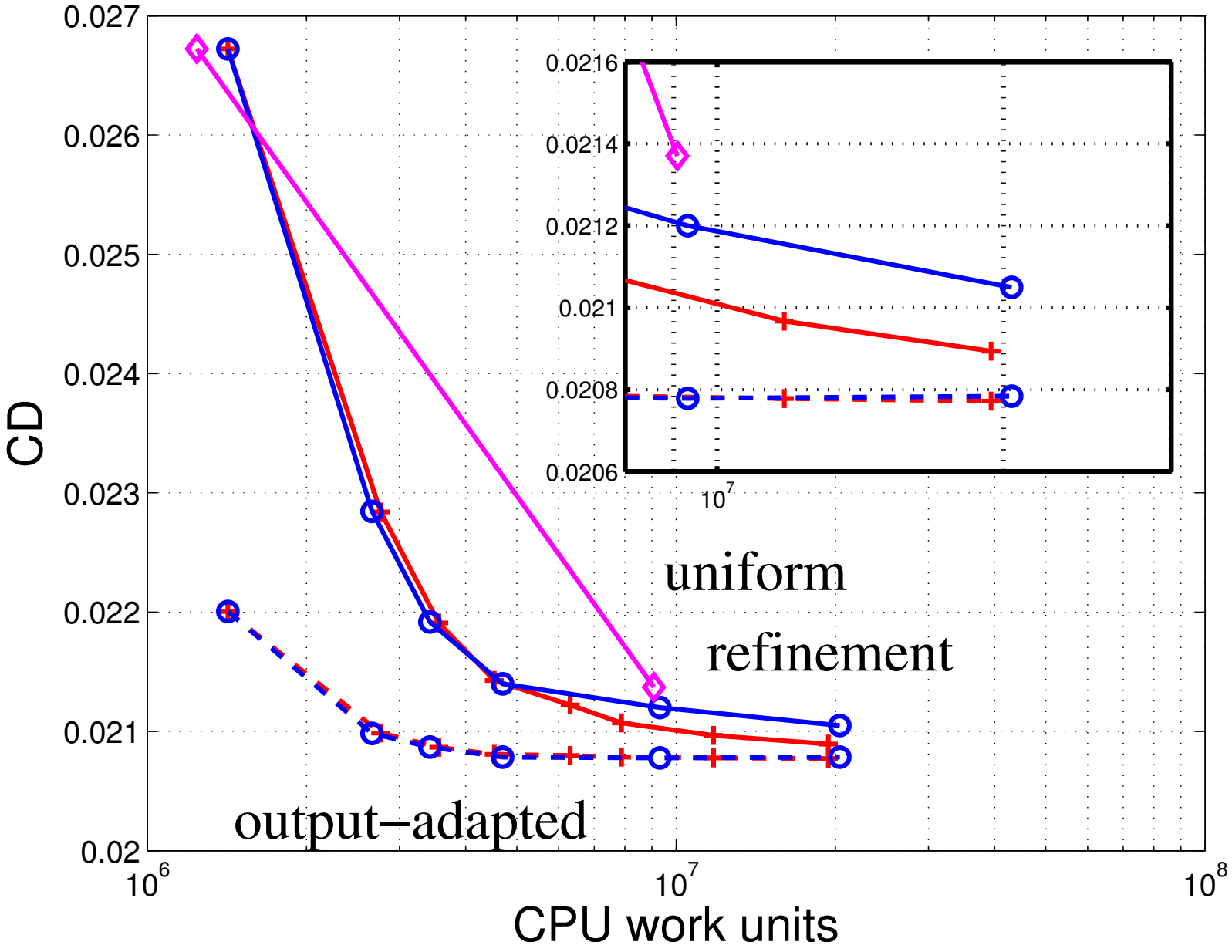


Mach/mesh using non-zero entries cost

DPW wing: comparison to uniform refinement



DPW wing: comparison to uniform refinement



Outline

A simple time discretization

- Discretizing space only gives

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) = \mathbf{0},$$

where $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the mass matrix,

$$\mathbf{M}_{ij} = \mathbf{I}_s \int_\Omega \phi_i \phi_j \, d\Omega$$

- Discretizing time via backward Euler, we have

$$\underbrace{\mathbf{M}\frac{\mathbf{U}^m - \mathbf{U}^{m-1}}{\Delta t} + \mathbf{R}(\mathbf{U}^m)}_{\text{unsteady residual: } \mathbf{R}^m} = \mathbf{0}$$

m = time node index

Adjoint discretization

- Unsteady sensitivity chain

$$\boldsymbol{\mu} \rightarrow \boxed{\mathbf{R}^m(\mathbf{U}^n, \boldsymbol{\mu}) = \mathbf{0}} \rightarrow \mathbf{U}^n \rightarrow J(\mathbf{U}^n)$$

- Adjoint equation

$$\sum_{m=1}^{N_t} \left(\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}^n} \right)^T \boldsymbol{\Psi}^m + \left(\frac{\partial J}{\partial \mathbf{U}^n} \right)^T = \mathbf{0}$$

Primal unsteady Jacobian

$$\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}^n} = \begin{bmatrix} \star & & & & & \\ \star & \star & & & & \\ & \star & \star & & & \\ & & \star & \star & & \\ & & & \star & \star & \\ & & & & \star & \star \end{bmatrix}$$

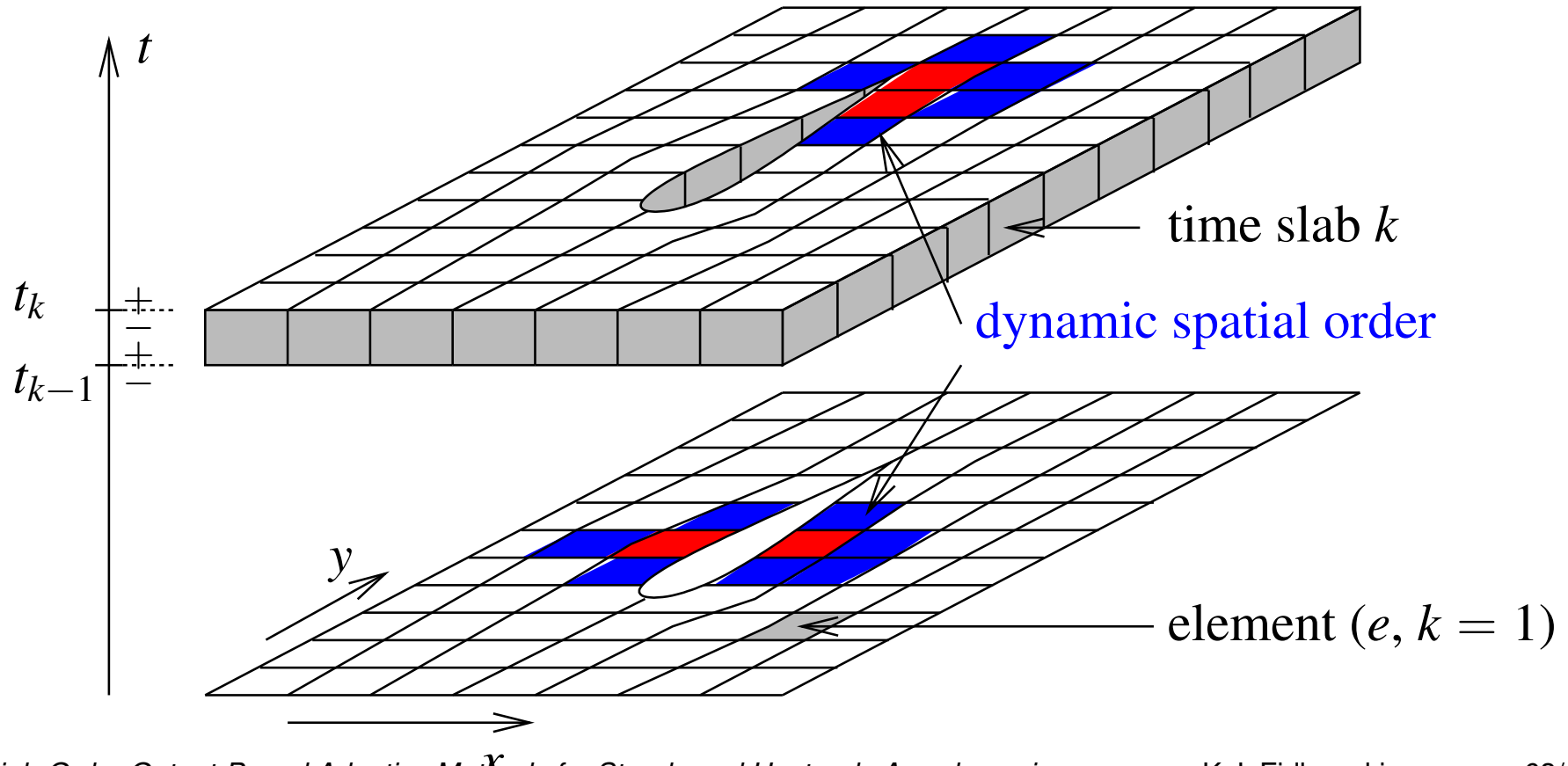
Adjoint unsteady Jacobian

$$\left(\frac{\partial \mathbf{R}^m}{\partial \mathbf{U}^n} \right)^T = \begin{bmatrix} \star & \star & & & & \\ & \star & \star & & & \\ & & \star & \star & & \\ & & & \star & \star & \\ & & & & \star & \star \\ & & & & & \star \end{bmatrix}$$

Jacobian transpose $\Rightarrow$ backwards time-marching for the adjoint

Discontinuous Galerkin in time

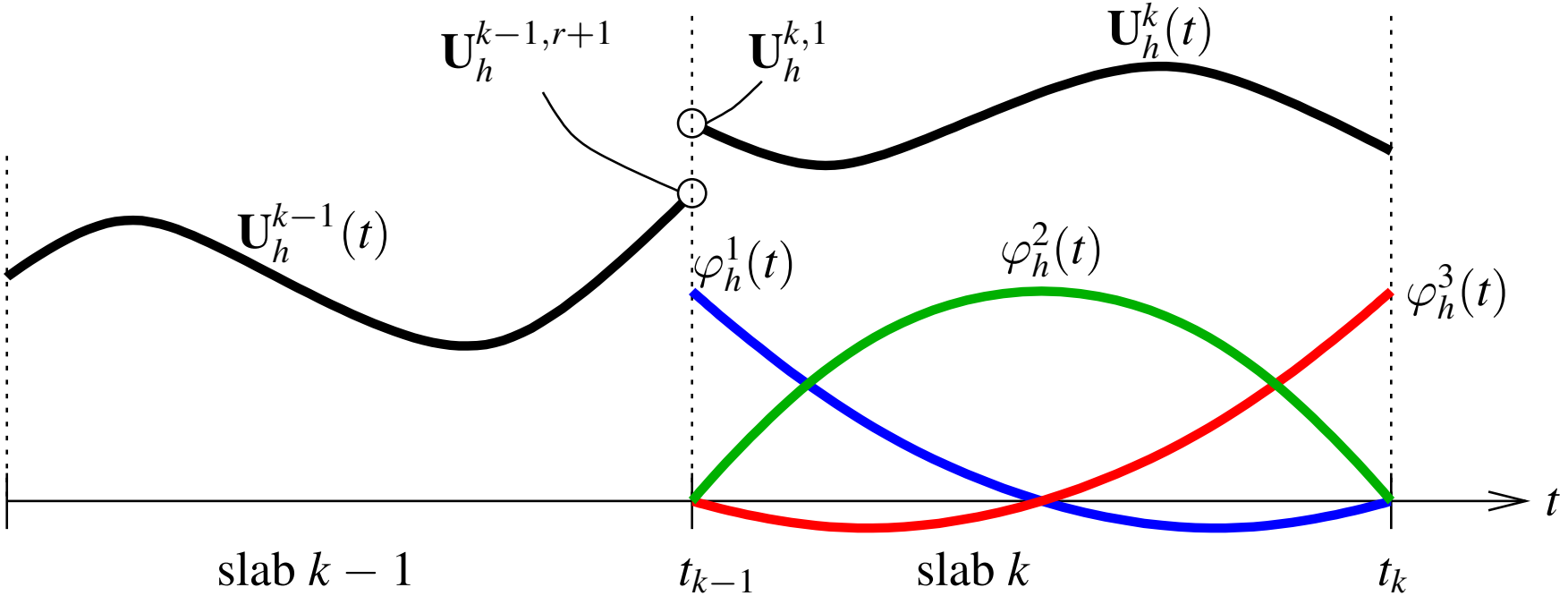
- Finite element in time: time intervals $\rightarrow$ "slabs"
- Order r temporal representation in each slab
- Spatial order can vary in time (dynamic order)
- End-of-slab solution provides initial condition for the next slab



DG-in-time equations

Multiplying the PDE by temporal test functions φ_h^m and integrating by parts gives $(r+1)$ unsteady residual vectors, enumerated by m,

$$\mathbf{R}_h^{km} \equiv \underbrace{a^{mn}\mathbf{M}_h^{k,k}\mathbf{U}_h^{kn} - \varphi_h^m(t_{k-1})\mathbf{M}_h^{k,k-1}\mathbf{U}_h^{k-1,r+1}}_{\text{from } d\mathbf{U}_h/dt} + \int_{t_{k-1}}^{t_k} \varphi_h^m(t)\mathbf{R}_h\left(\mathbf{U}_h^k(t)\right) dt$$



DG-in-time adjoint

- Total number of time nodes: $N_t = N_k(r+1)$
- $\mathbf{\Psi}_H^{km}$ = the adjoint at time slab k, time node m
- Unsteady adjoint equation

$$\underbrace{\left(\frac{\partial \mathbf{R}_h^{km}}{\partial \mathbf{U}_h^{ln}}\right)^T \mathbf{\Psi}_h^{km} + \left(\frac{\partial J_h}{\partial \mathbf{U}_h^{ln}}\right)^T}_{\mathbf{R}_{\psi h}^{ln}(\mathbf{\Psi}_h^{km})} = 0$$

k, l = time slab indices $\qquad$ m, n = intra-slab time node indices

- Primal and adjoint systems can both be solved using a relatively cheap, inexact Newton method

Output error estimation

- The adjoint-weighted residual extends to unsteady systems,

$$\delta J \approx -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H) = -\sum_{k=1}^{N_k}\sum_{m=1}^{r+1}\left(\boldsymbol{\Psi}_h^{km}\right)^T \mathbf{R}_h^{km}(\mathbf{U}_h^H)$$

$$= -\sum_{k=1}^{N_k}\sum_{e=1}^{N_e}\underbrace{\sum_{m=1}^{r+1}\left(\boldsymbol{\Psi}_{he}^{km}\right)^T \mathbf{R}_{he}^{km}\left(\mathbf{U}_h^H\right)}_{\varepsilon_e^k = \text{contribution of } (e,k)}$$

- The error indicator for element e of time slab k is

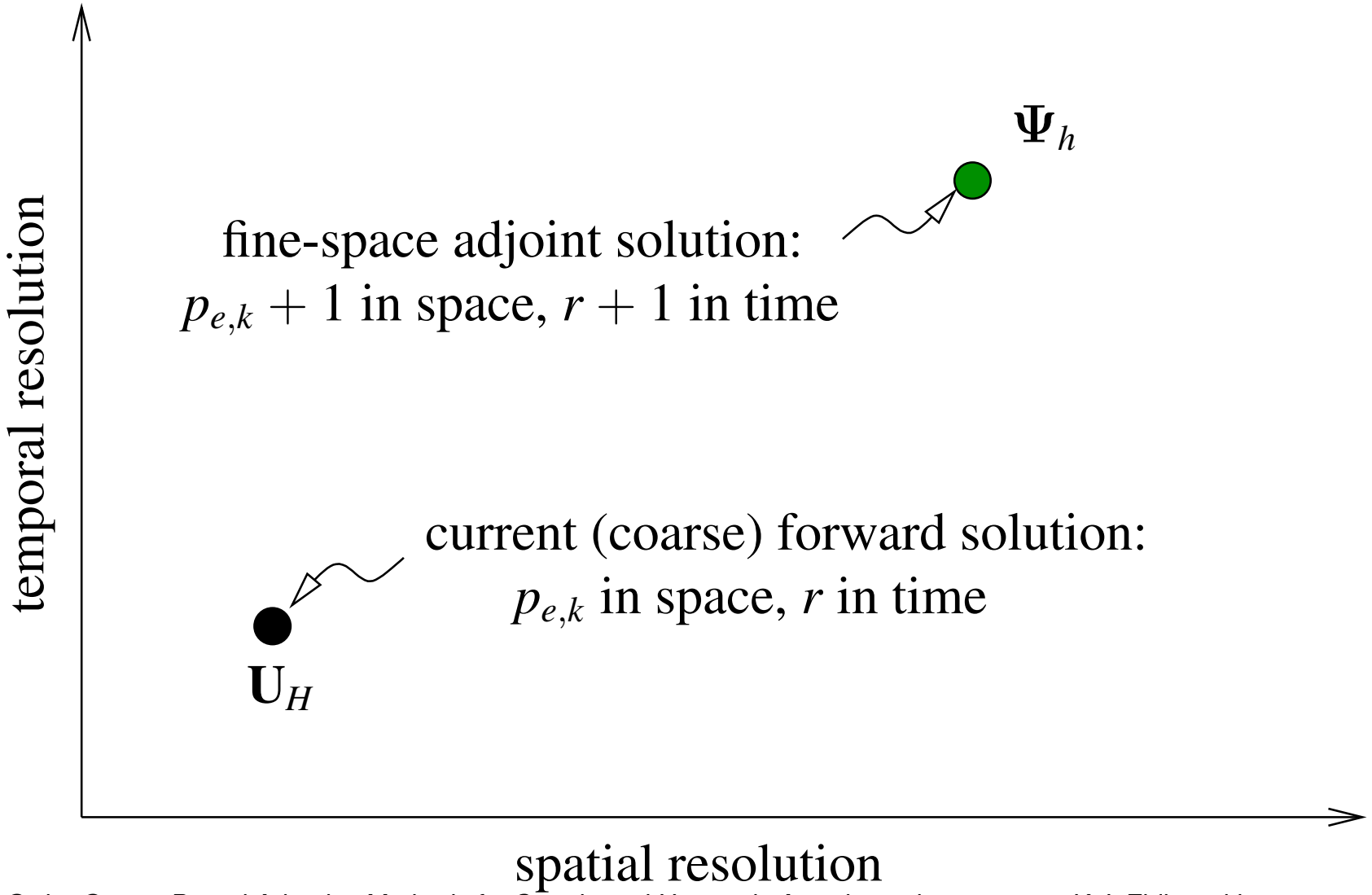
$$\epsilon_e^k = \left|\varepsilon_e^k\right|$$

- Sometimes also interested in a "conservative" error estimate

$$\text{sum of indicators} = \epsilon = \sum_{k=1}^{N_k}\sum_{e=1}^{N_e}\epsilon_e^k$$

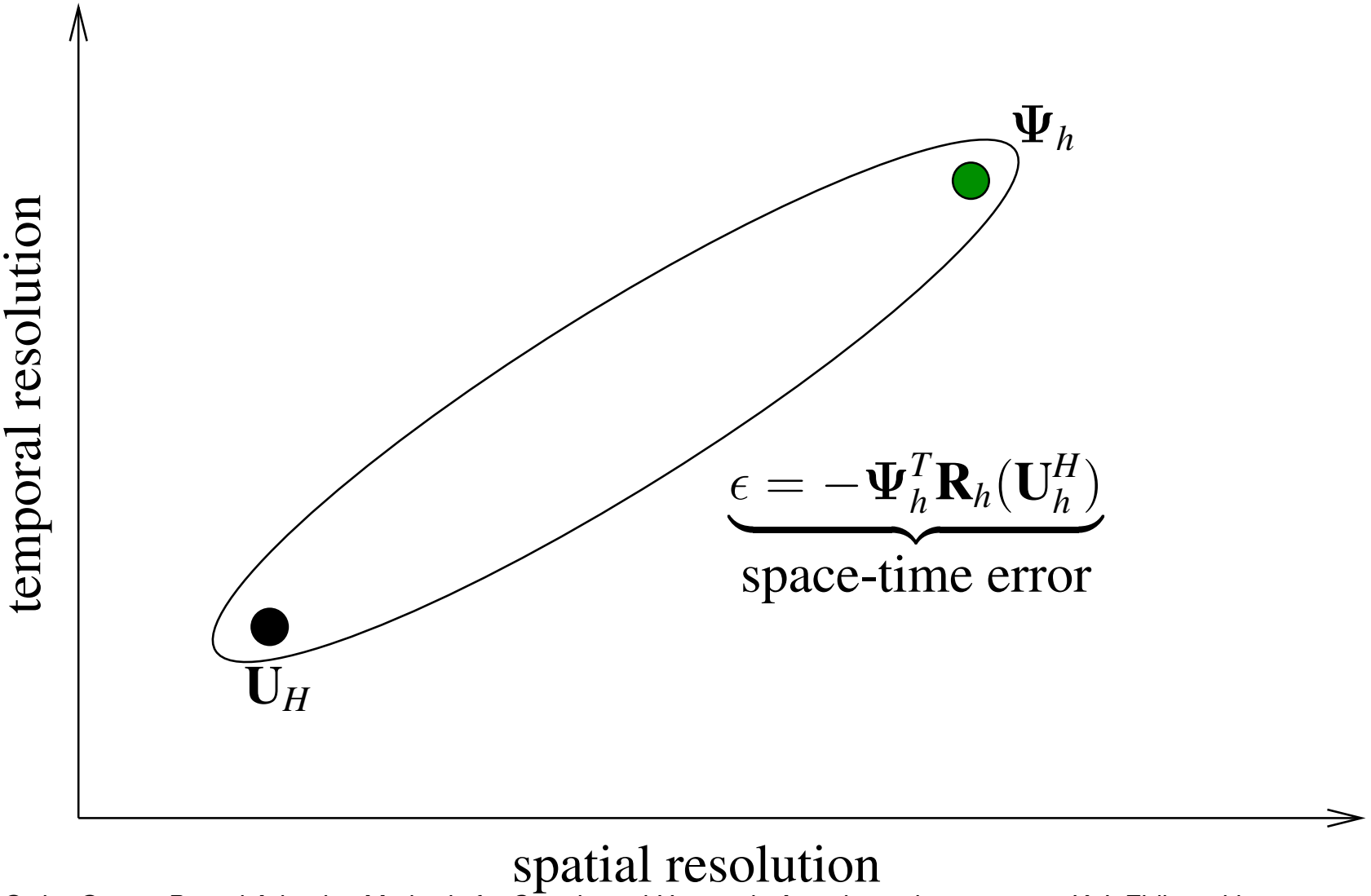
Space-time anisotropy measure

How much of the error is due to the spatial versus the temporal discretization?



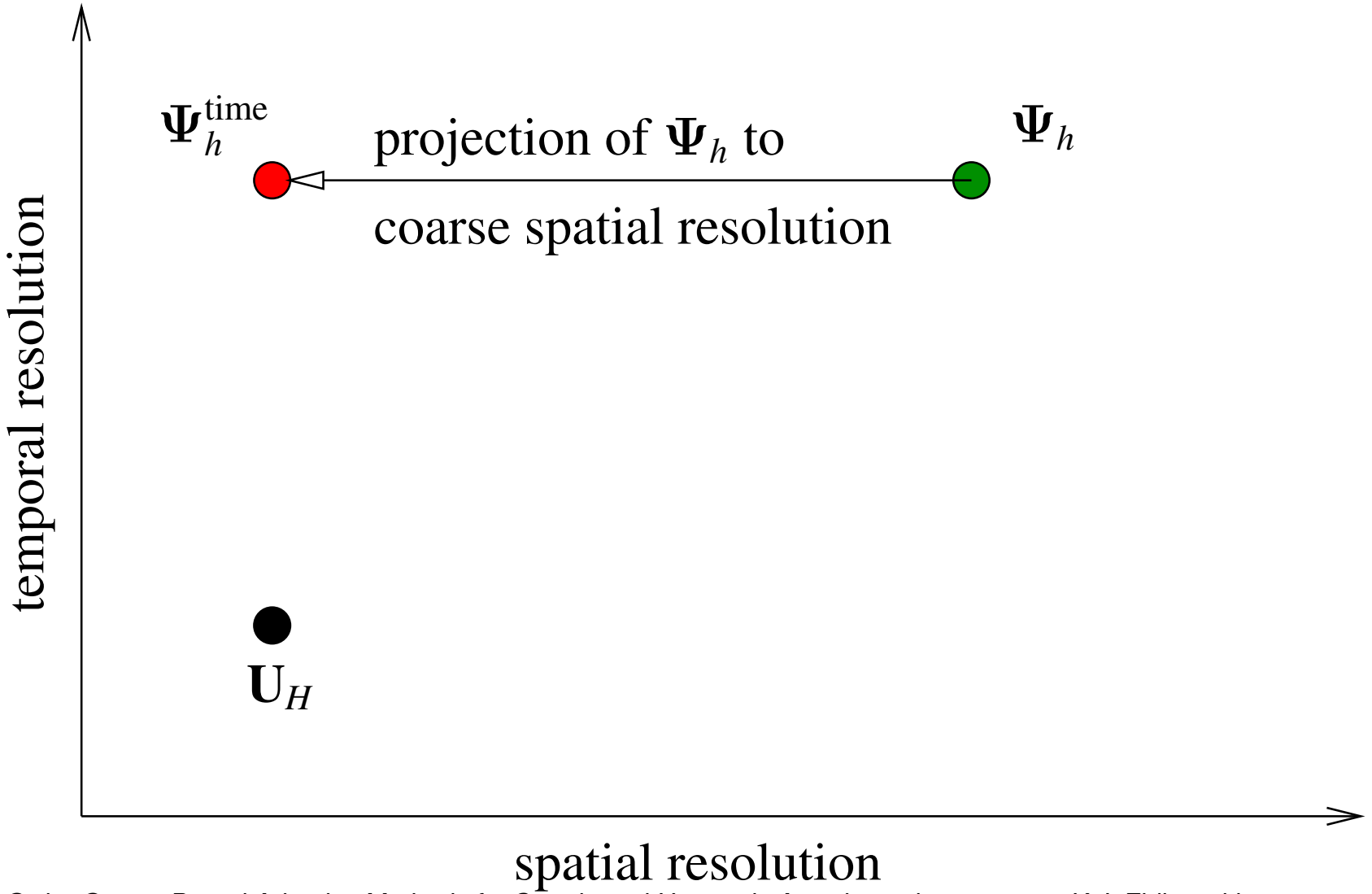
Space-time anisotropy measure

How much of the error is due to the spatial versus the temporal discretization?



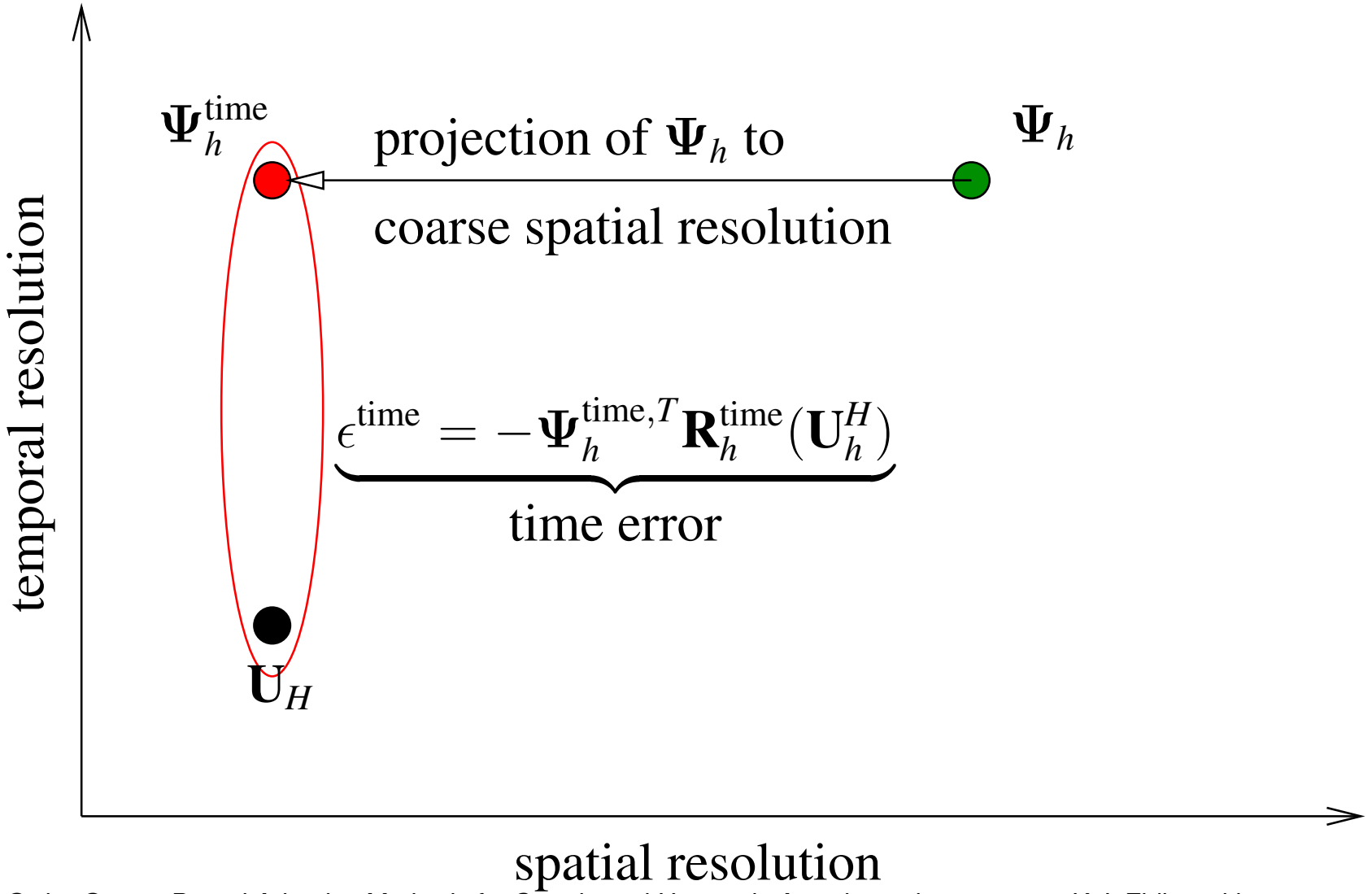
Space-time anisotropy measure

How much of the error is due to the spatial versus the temporal discretization?



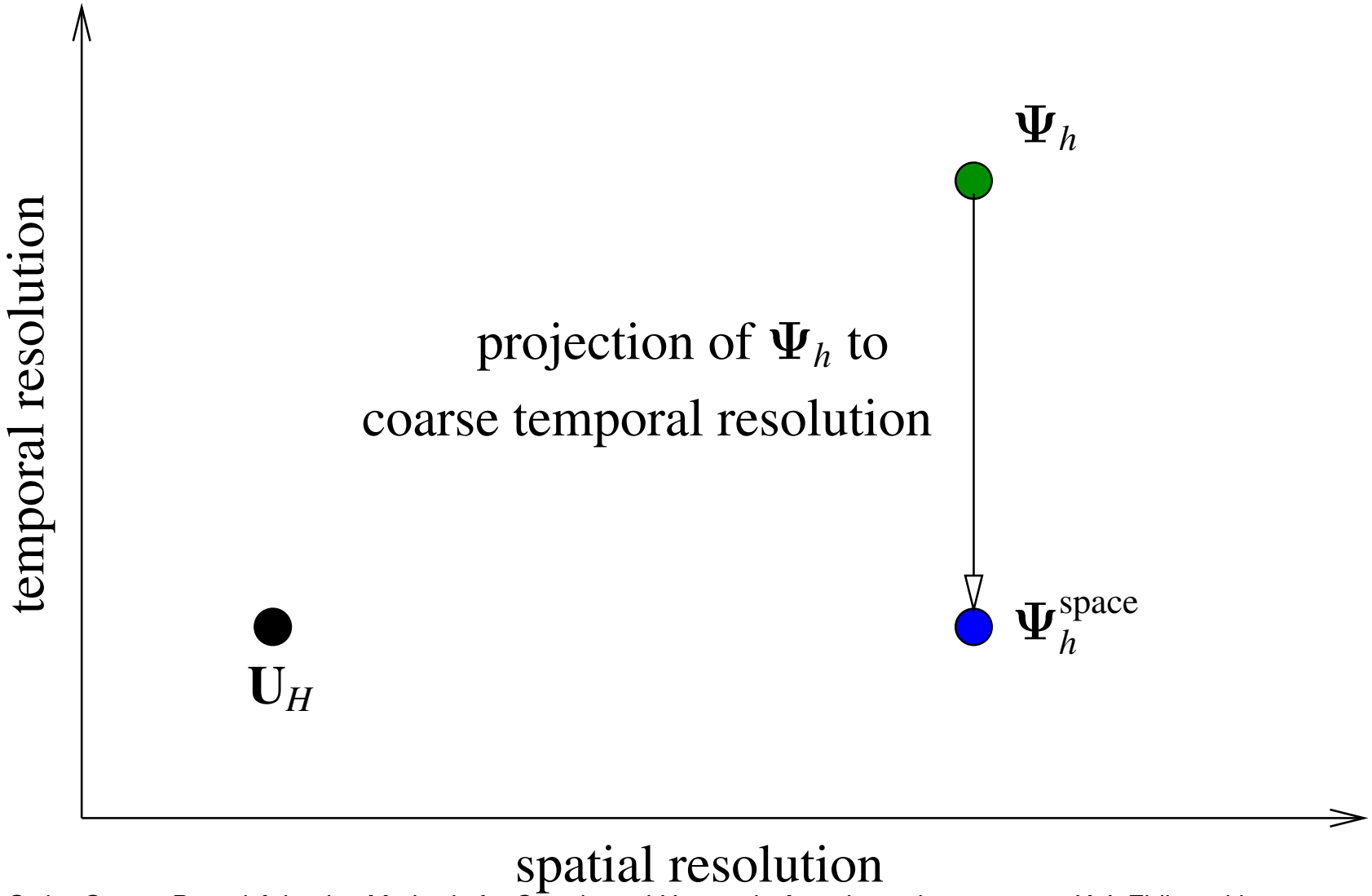
Space-time anisotropy measure

How much of the error is due to the spatial versus the temporal discretization?



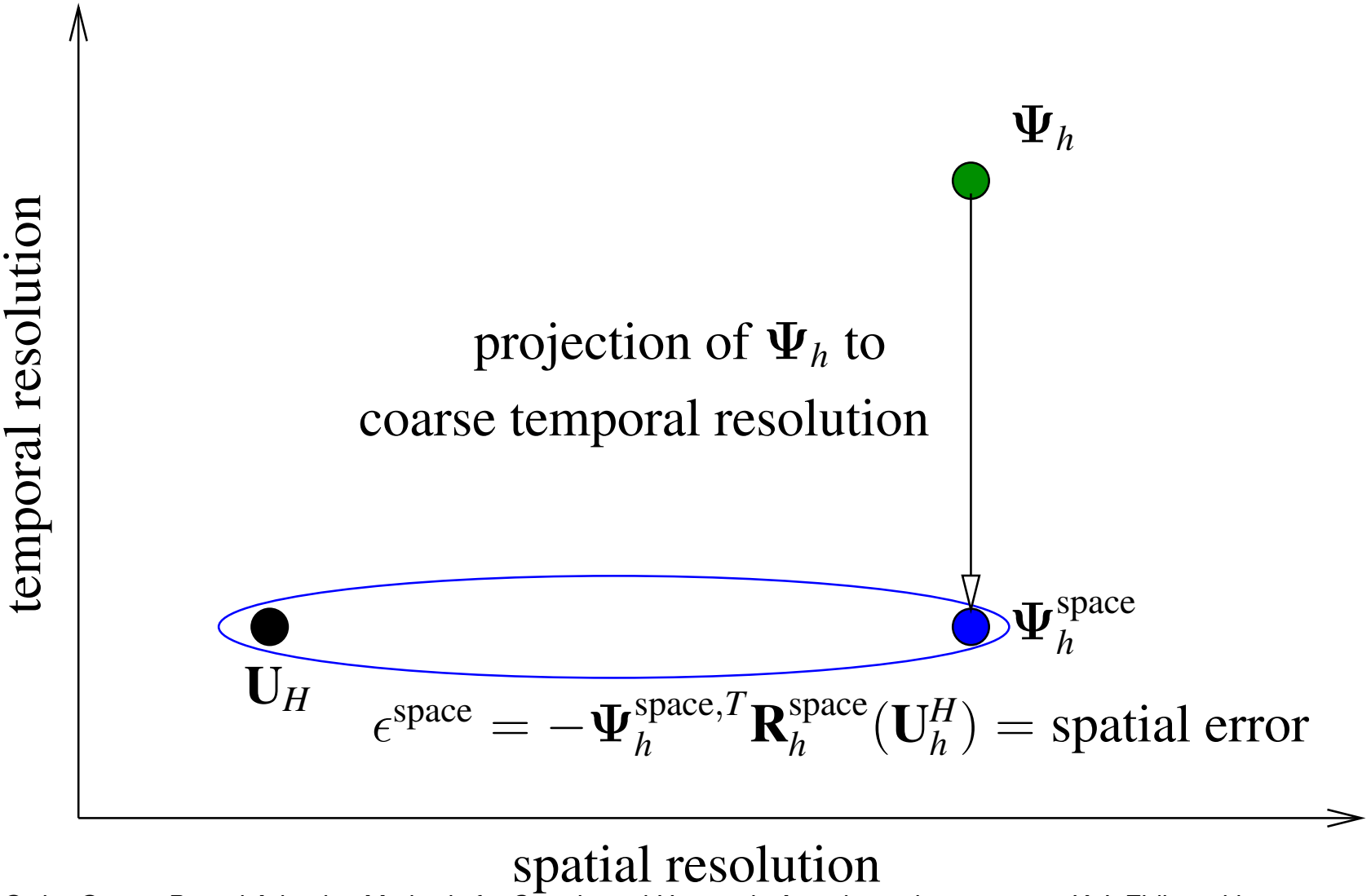
Space-time anisotropy measure

How much of the error is due to the spatial versus the temporal discretization?



Space-time anisotropy measure

How much of the error is due to the spatial versus the temporal discretization?



Space-time error indicators

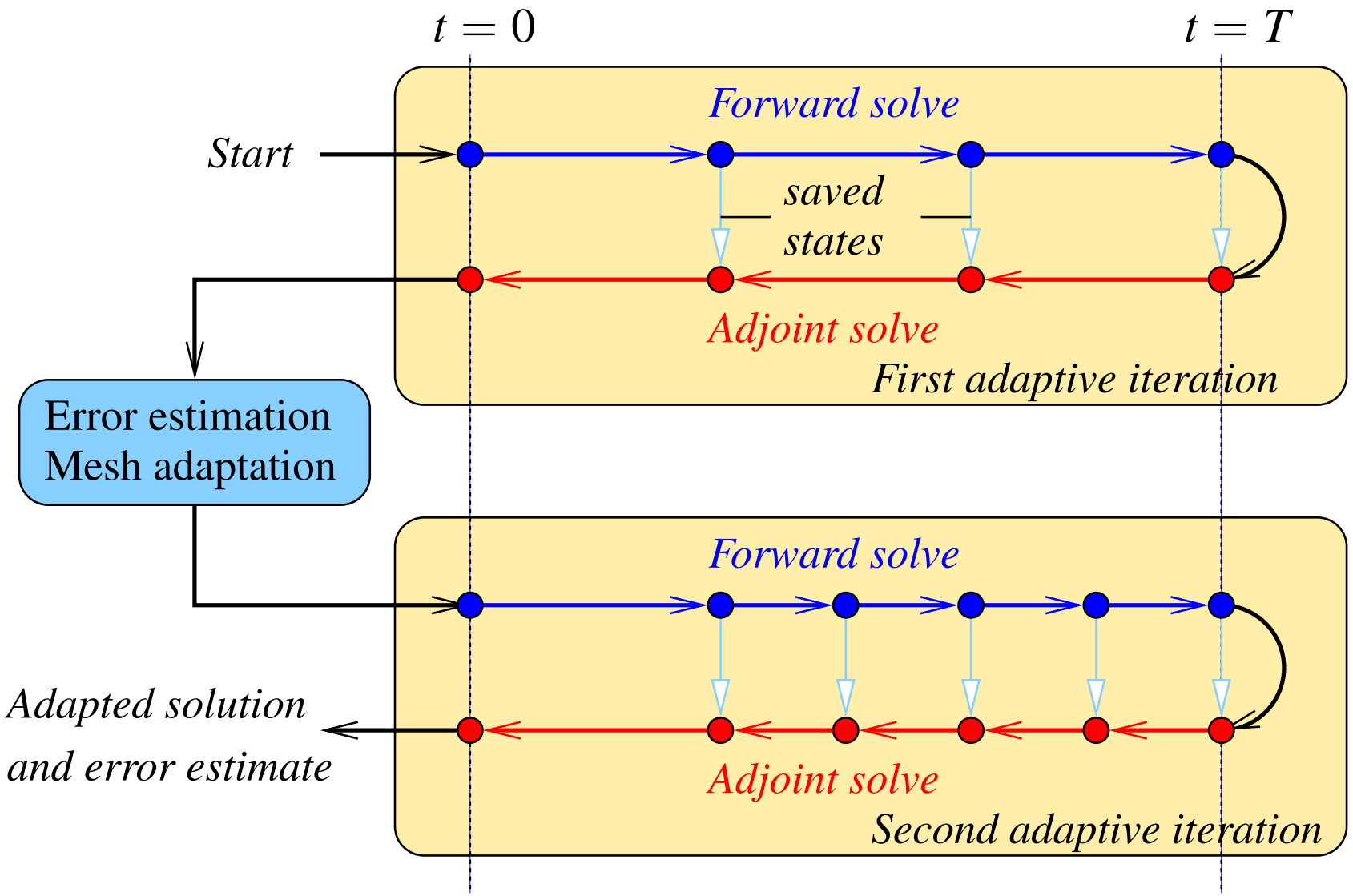
- One can use the projection-based anisotropy measure on each space-time element
- These projections give separate $\varepsilon_{e,k}^{\text{space}}$ and $\varepsilon_{e,k}^{\text{time}}$ estimates, which then yield spatial and temporal error fractions,

$$\beta_e^{k,\text{space}} = \frac{|\varepsilon_e^{k,\text{space}}|}{|\varepsilon_e^{k,\text{space}}| + |\varepsilon_e^{k,\text{time}}|}, \qquad \beta_e^{k,\text{time}} = 1 - \beta_e^{k,\text{space}}$$

- Adaptive indicators use total space-time error, $\epsilon_e^k = |\varepsilon_e^k|$

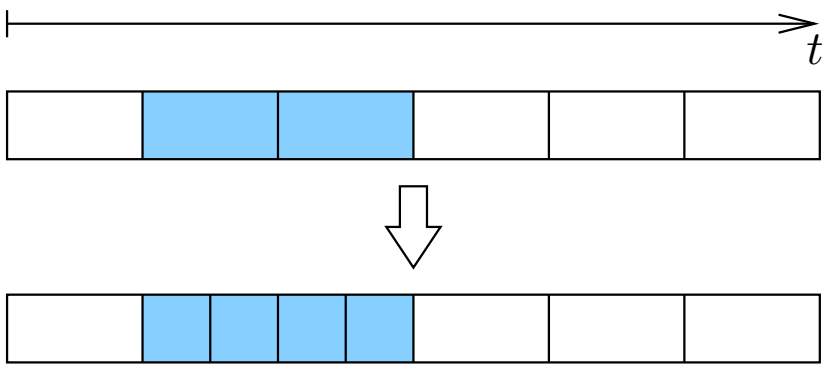
$$\begin{aligned}
\text{spatial indicator on space-time elem } e,k = \quad & \epsilon_e^{k,\text{space}} && = \epsilon_e^k \beta_e^{k,\text{space}} \\
\text{total spatial indicator on spatial elem } e = \quad & \epsilon_e^{\text{space}} && = \sum_k \epsilon_e^k \beta_e^{k,\text{space}} \\
\text{total temporal indicator on time slab } k = \quad & \epsilon^{k,\text{time}} && = \sum_e \epsilon_e^k \beta_e^{k,\text{time}}
\end{aligned}$$

Adaptive solution process

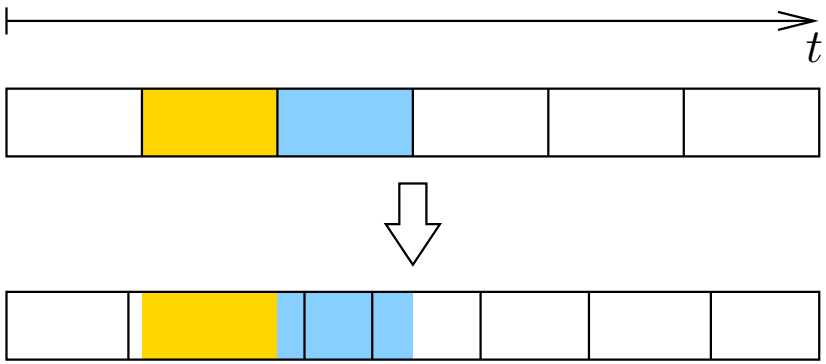


Temporal refinement

- Use bisection of time slabs if just refining
- If also coarsening, need to shuffle all time slabs
- Shuffle using a one-dimensional metric-based algorithm
- Example: refine blue slabs, coarsen gold slabs



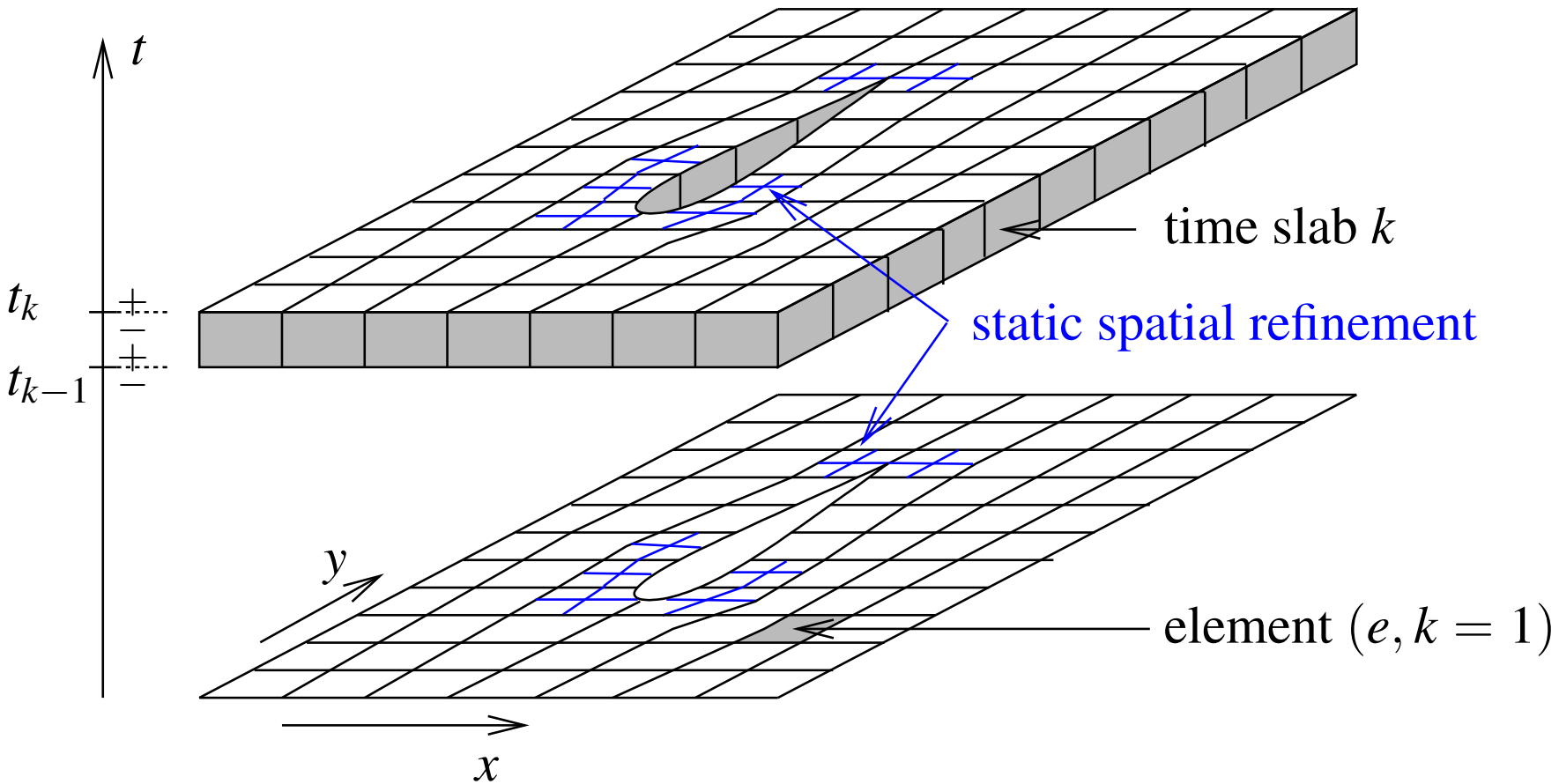
Time slab bisection



Time-slab shuffling

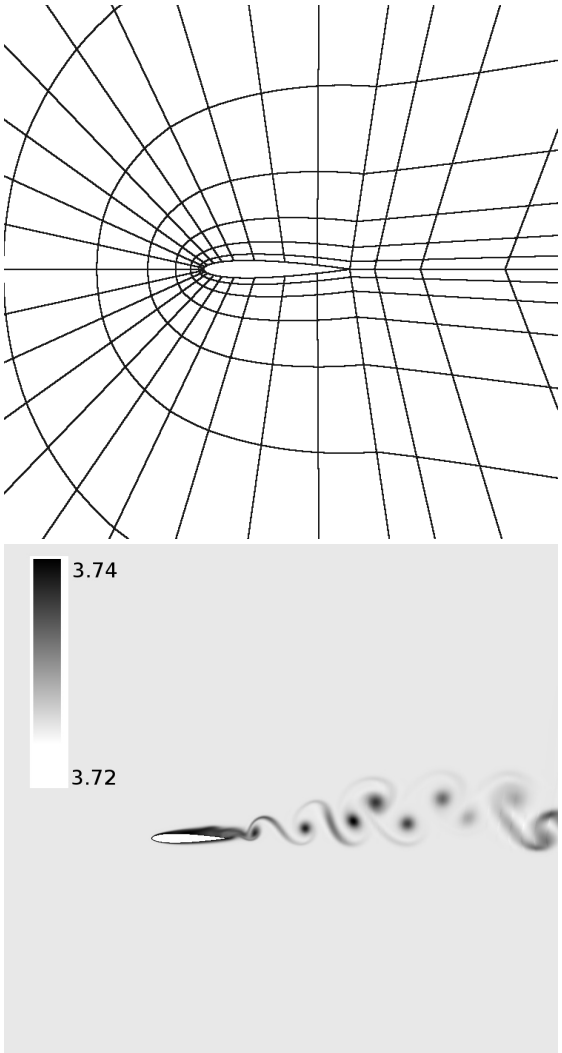
Static spatial refinement

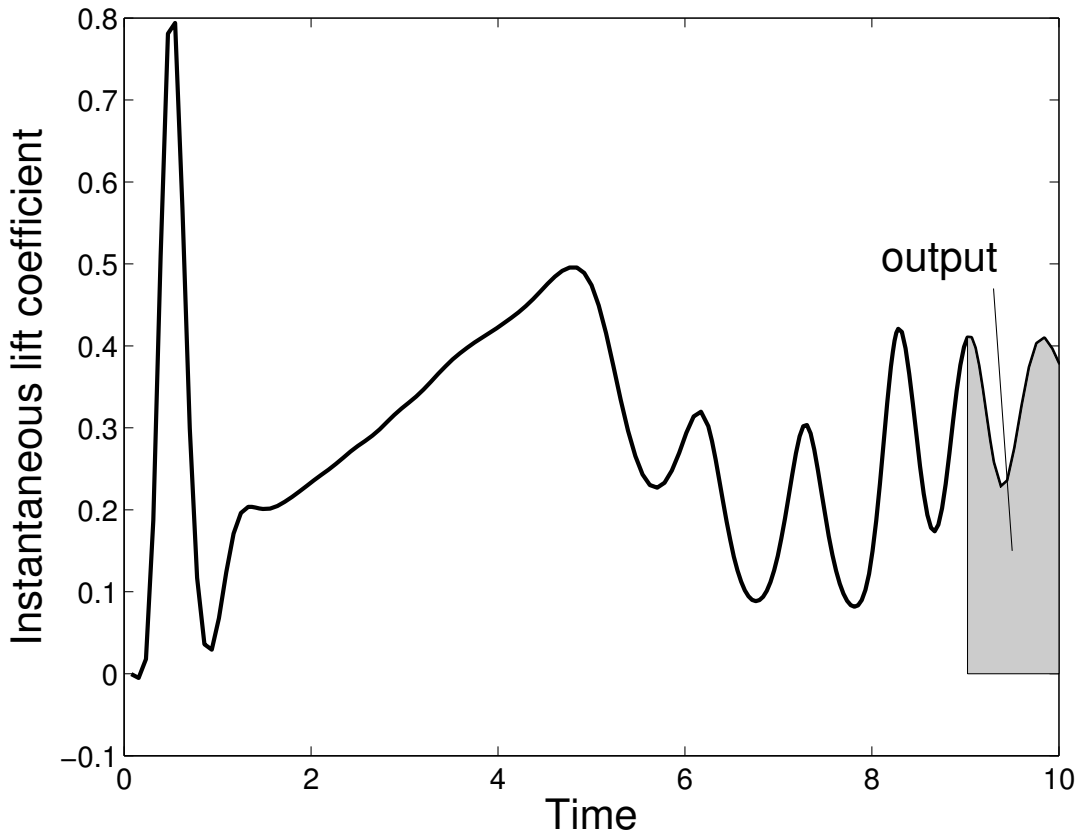
- As a simplification, let's first keep the spatial refinement constant in time
- On each adaptive iteration we h-refine some spatial elements
- This is surprisingly efficient for many problems



Static h-refinement, impulsively-started airfoil

NACA 0012, Navier-Stokes, $M_\infty = 0.5$, $Re = 5000$, $\alpha = 8°$

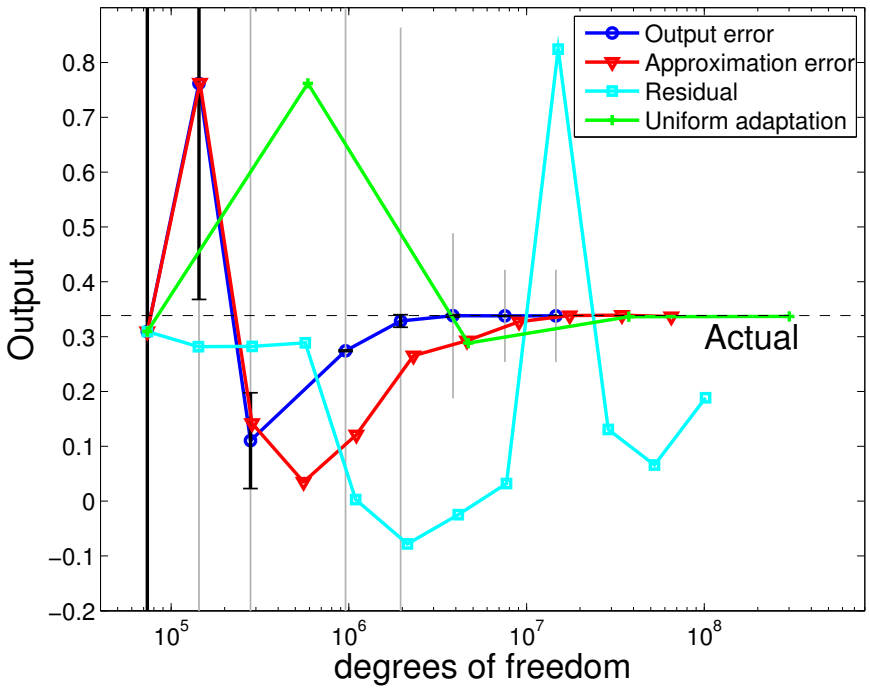




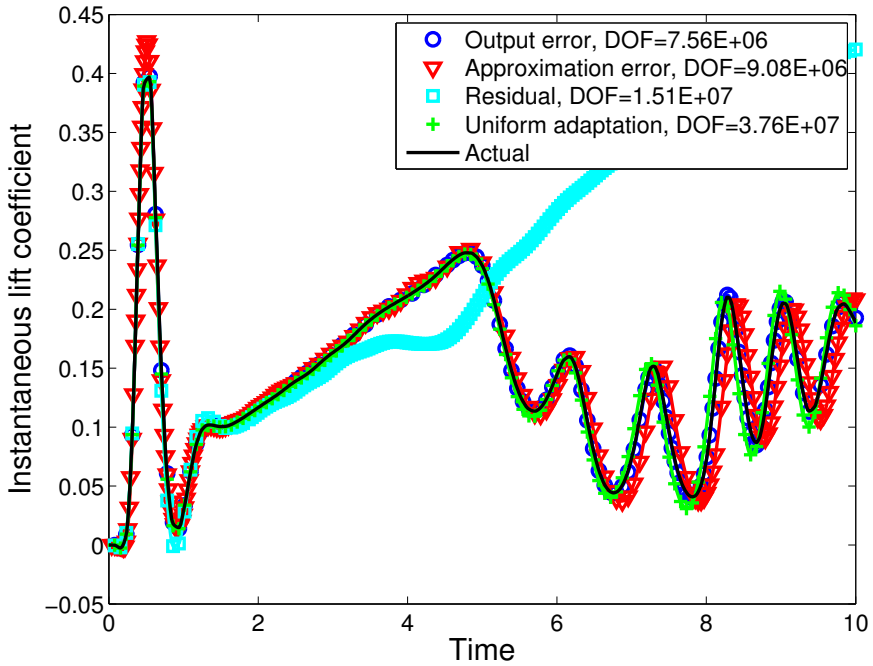
Time integral output definition; a vortex-shedding pattern has been established by the time of the output measurement

Impulsively-started airfoil: output convergence

Consider space-time refinement using various error indicators



Output convergence

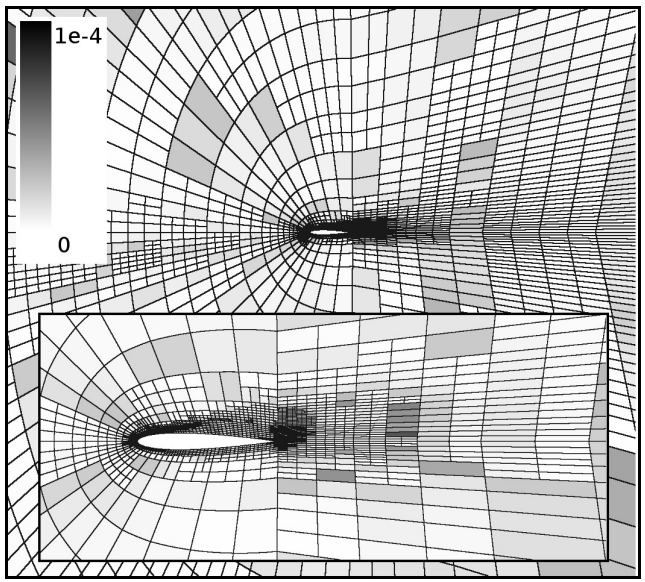


Output histories

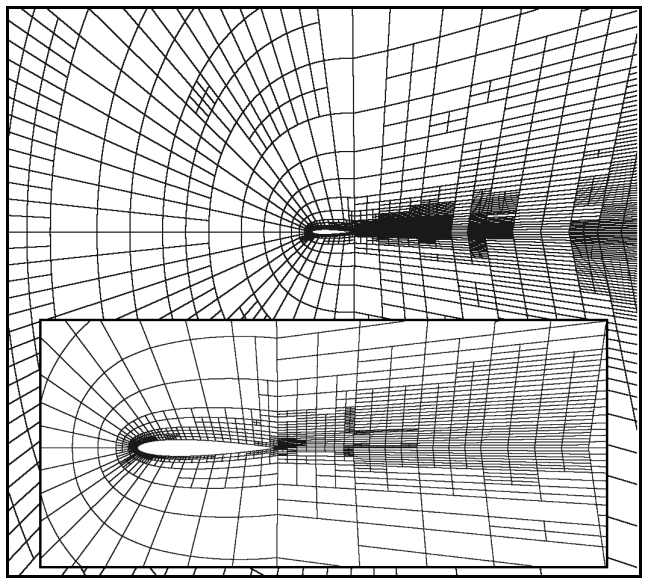
- Acoustic waves distract the unweighted residual indicator
- Local approximation error refinement performs well
- Output-based refinement is the most efficient in DOFs

Impulsively-started airfoil: adapted spatial meshes

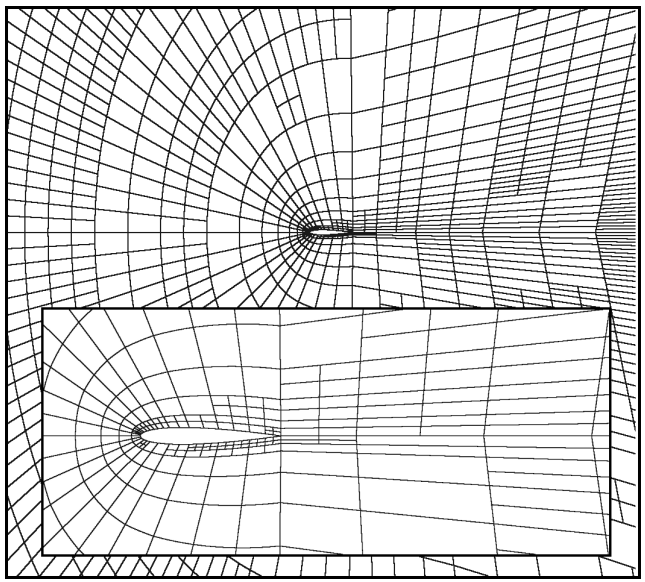
- Meshes shown at iterations with similar total dofs
- Spatially-marginalized output error estimate ϵ_e is shown on the elements of the output-adapted mesh



Adapted on output error (5956 elements)

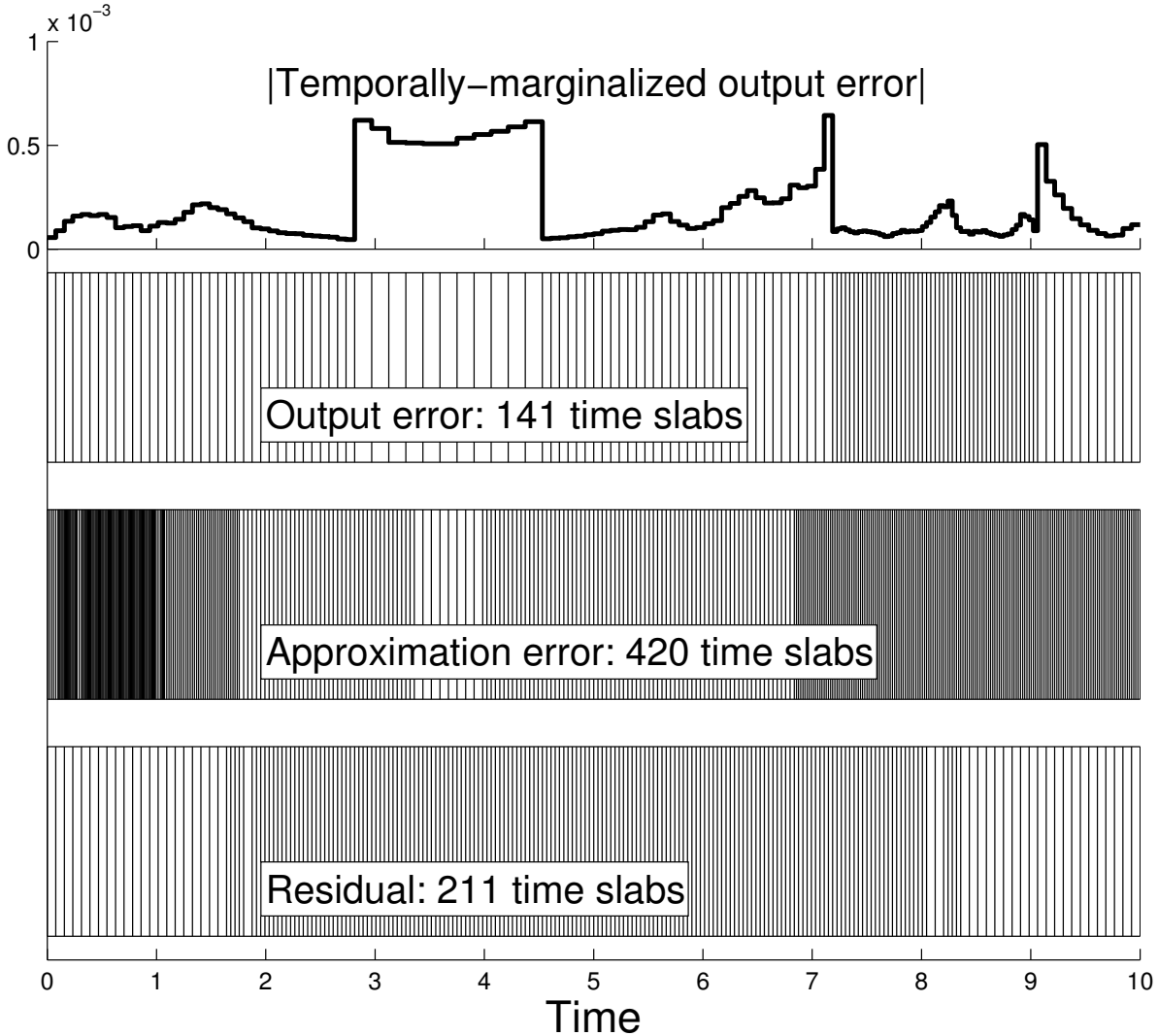


Adapted on approximation error (4585 elements)



Adapted on residual (7929 elements)

Impulsively-started airfoil: adapted time slabs

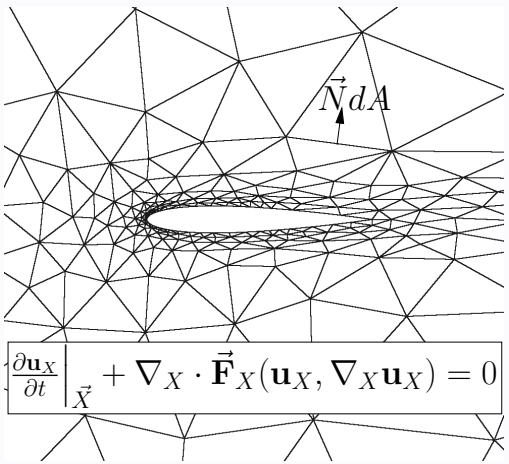


- Output indicator yields a fairly-uniform temporal refinement
- Approximation error focuses on the initial time (dynamics of the IC) and the latter 1/3 of the time, when the shed vortices develop
- Residual creates a mostly-uniform temporal mesh as it tracks acoustic waves

Deformable domains

ALE Idea: solve transformed PDE on a static reference domain

Reference domain: $\vec{X}, \mathbf{u}_X, \vec{\mathbf{F}}_X$



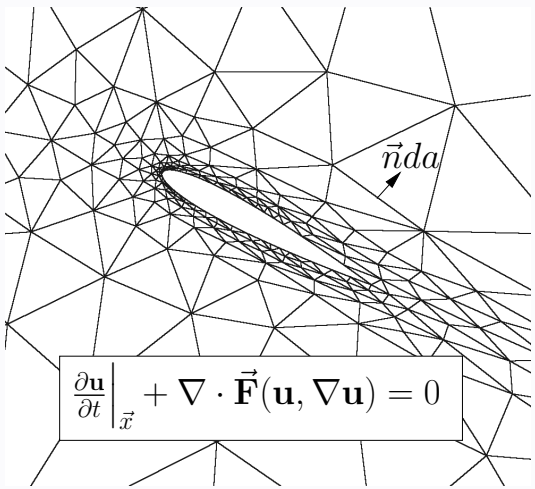
$\Rightarrow$

Mapping

$$
\begin{aligned}
\vec{X}, t &\Rightarrow \vec{x}(\vec{X}, t) \\
\underline{G} &= \frac{\partial \vec{x}}{\partial \vec{X}} \\
g &= \det(\underline{G}) \\
\mathbf{u}_X &= g\mathbf{u} \\
\vec{v}_G &= \frac{\partial \vec{x}}{\partial t} \\
\vec{\mathbf{F}}_X &= g\underline{G}^{-1}\vec{\mathbf{F}} - \mathbf{u}_X \underline{G}^{-1}\vec{v}_G \\
\vec{n}da &= g\underline{G}^{-T}\vec{N}dA \\
\vec{N}dA &= g^{-1}\underline{G}^{T}\vec{n}da
\end{aligned}
$$

$\Rightarrow$

Physical domain: $\vec{x}, \mathbf{u}, \vec{\mathbf{F}}$

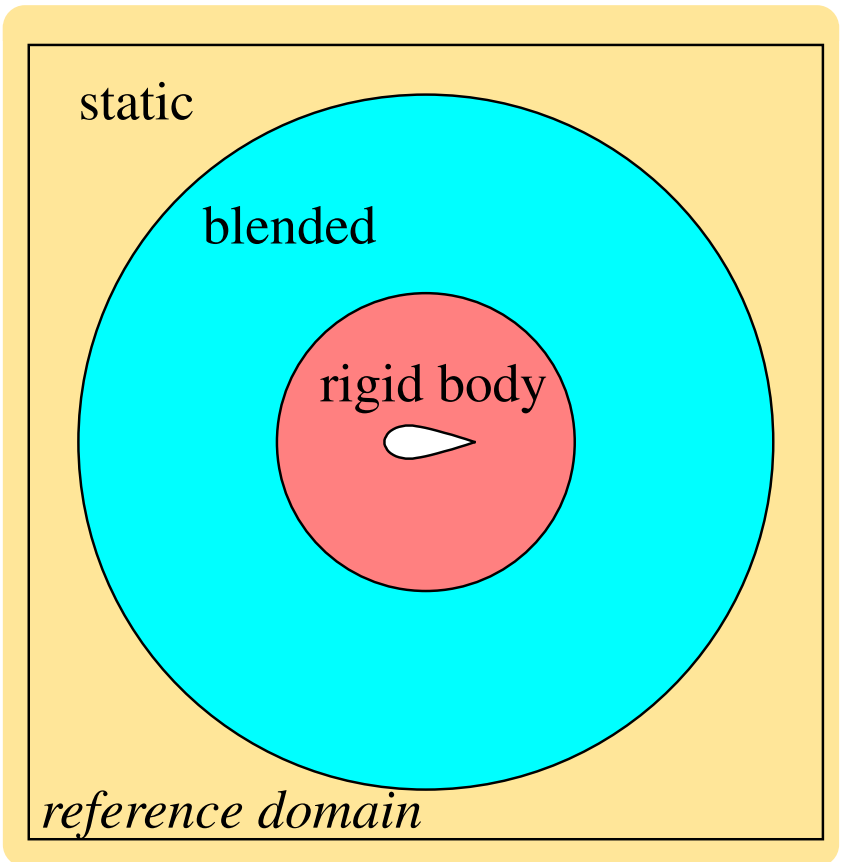


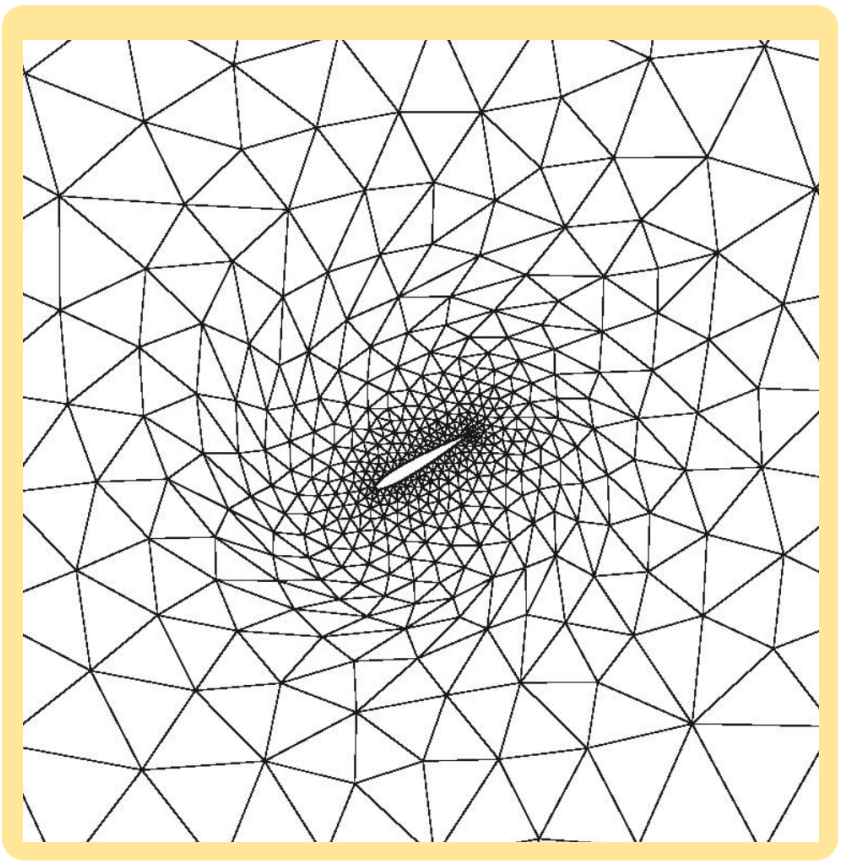
Key definitions

$\vec{X}$	=	reference-domain coordinates	$\mathbf{u}$	=	physical state
$\vec{x}$	=	physical-domain coordinates	$\mathbf{u}_X$	=	reference state
g	=	determinant of Jacobian matrix	$\vec{\mathbf{F}}$	=	physical flux vector
$\vec{v}_G$	=	grid velocity, $\partial\vec{x}/\partial t$	$\vec{\mathbf{F}}_X$	=	reference flux vector

Analytical motion technique

- Simple and quite general approach for deforming domains
- Near-field rigid body motion blends into a static farfield mesh
- Blending occurs via polynomial functions in the radial coordinate





Geometric conservation law (GCL)

- The ALE approach does not guarantee conservation:
  - Cannot always represent a constant physical solution
  - Time integration errors affect conservation
- A GCL by Persson et al (2009) relies on approximating $\mathbf{u}_{\bar{X}} = \bar{g}\mathbf{u}$ and solving an additional equation

$$\frac{\partial \bar{g}}{\partial t} - \nabla_X \cdot (g\mathcal{G}^{-1}\vec{v}_G) = 0$$

- This equation is local and cheap to solve, but ...
  - We need to integrate with higher quadrature rules
  - We now need to compute a GCL adjoint for error estimation

Pitching and plunging airfoils

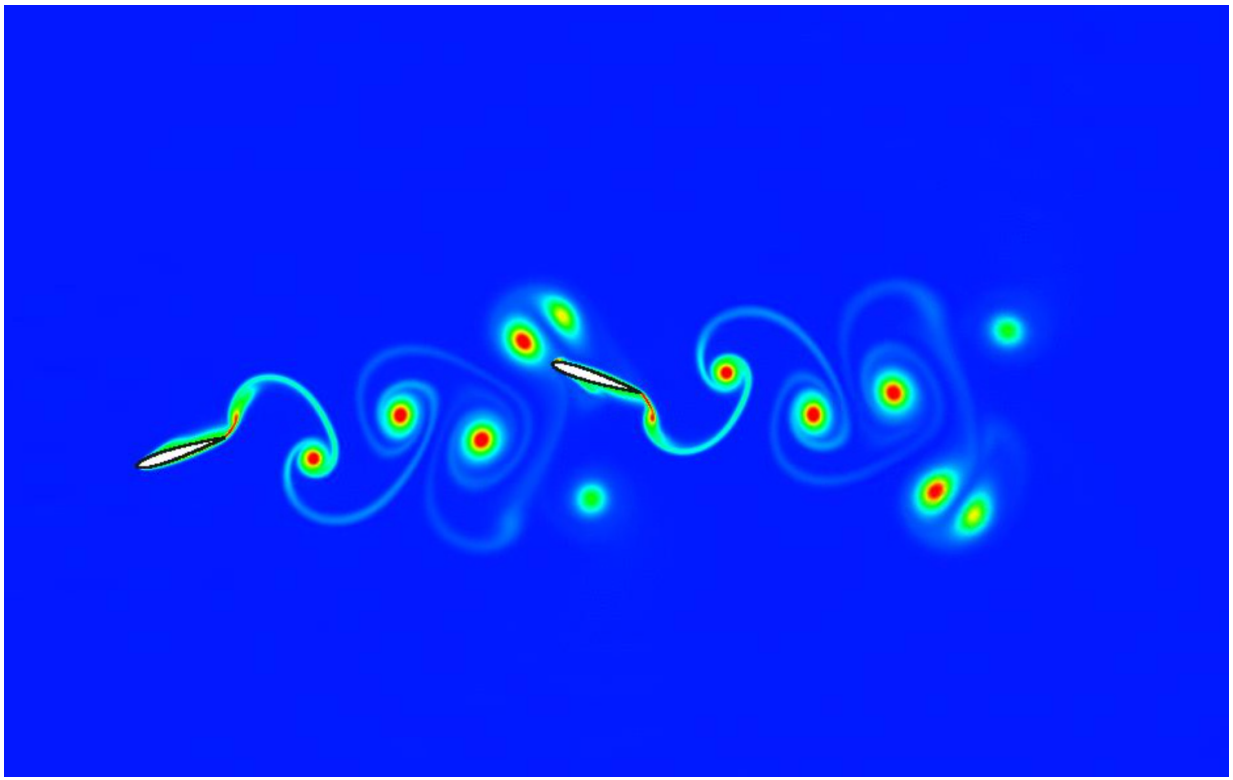
Consider two airfoils pitching and plunging in series

$$Re = 1200, \quad M_{inf} = 0.3, \quad Str = 2/3, \quad A_{pitch} = \pm 30\,^{\circ}, \quad A_{plunge} = 0.25c$$

Case params.

- $60c \times 60c$ domain
- $f_{growth} = 35\%$
- $f_{coarsen} = 5\%$

Output: Lift on the right airfoil integrated from $t = 7.25$ to 7.5 (the final time)

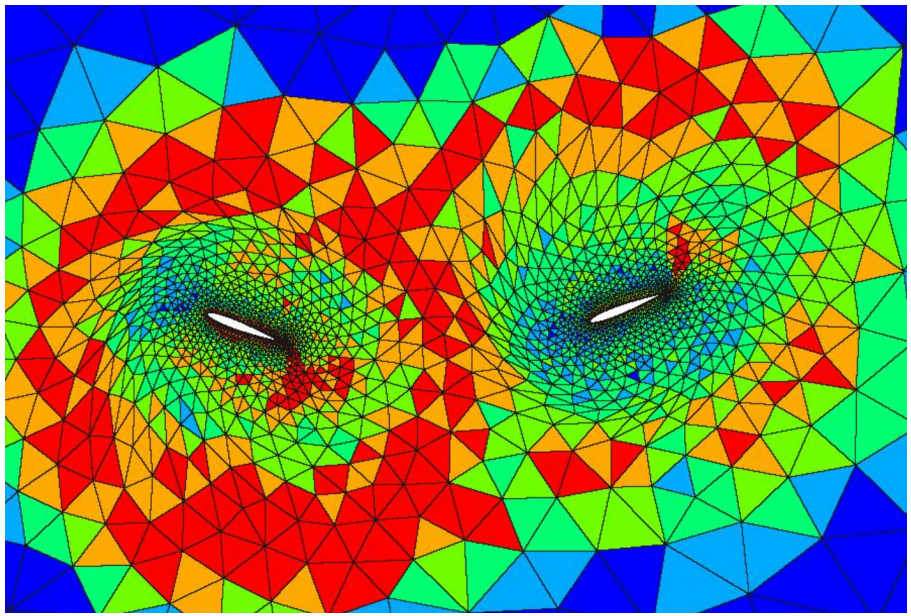


Entropy contours at the end of the simulation

Adapted mesh

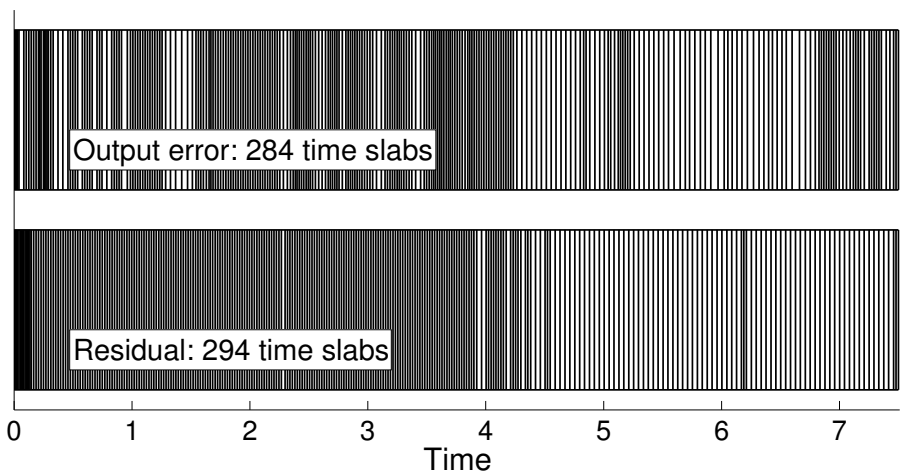
Output-based method

- Targets vortex shedding and larger elements near motion regions
- Refines earlier times, as well as final times over which output is integrated

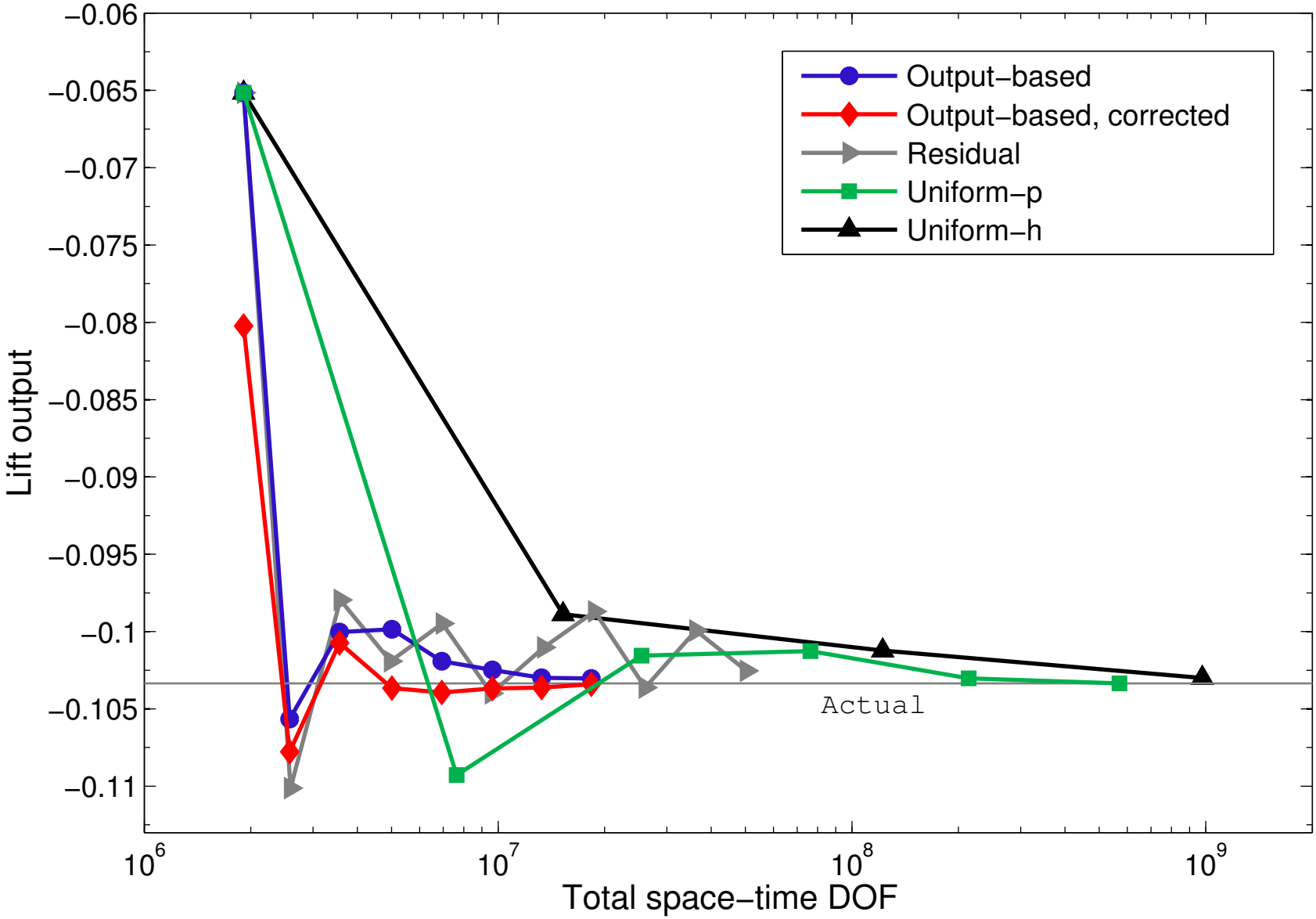


Residual-based method

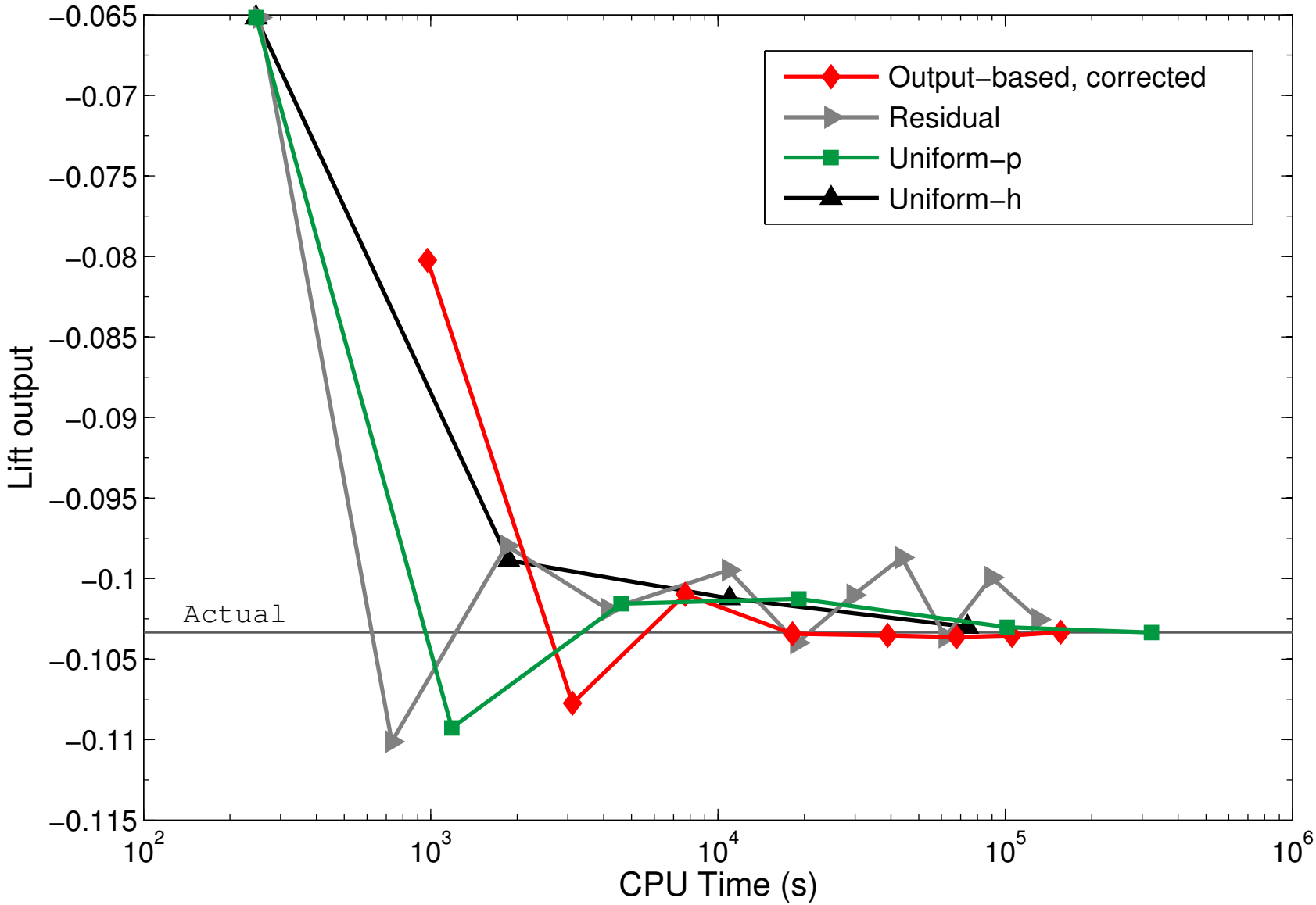
- Only adapts initial times
- Is again distracted by acoustic waves



Output convergence versus DOF

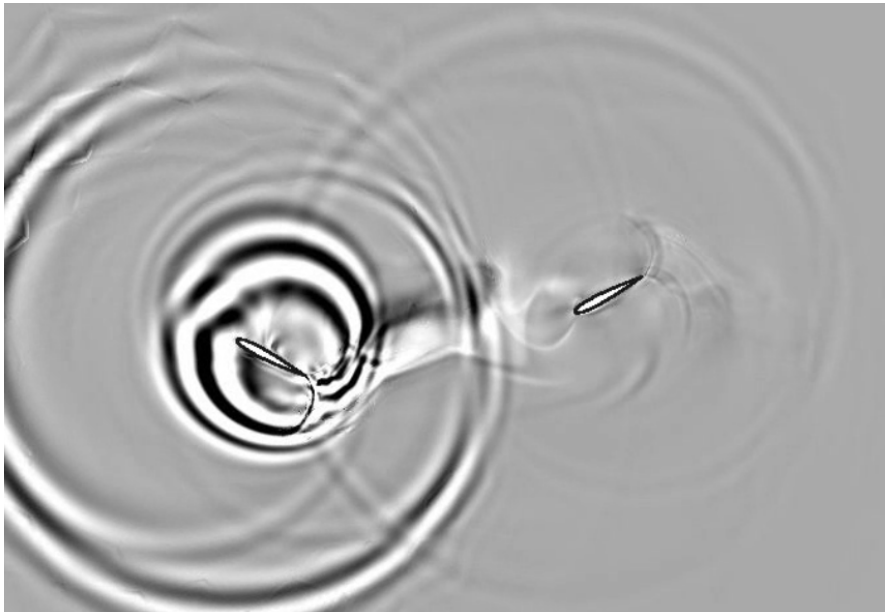


Output convergence versus CPU time



GCL adjoint

To see what's (in part) driving the adaptation, we look at contours of the GCL adjoint. Black and white regions indicate large output sensitivity.



- The output is very sensitive to initial vortex shedding from the first airfoil
- Acoustic rings and a convection path between the airfoils indicate two different modes of error propagation

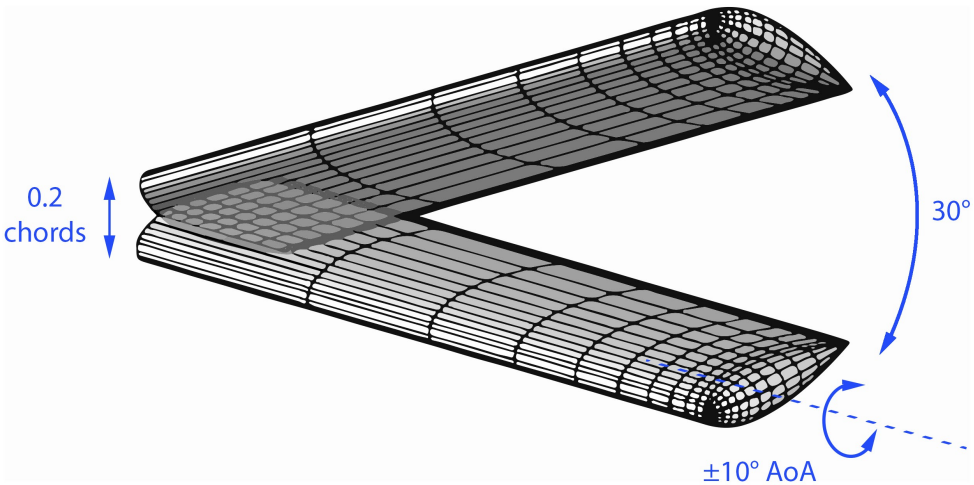
Three-dimensional flapping

We apply the adaptive strategy to a 3D flapping simulation

$$Re = 500, \quad M_{inf} = 0.3, \quad Str = 0.4, \quad A_{stroke} = \pm 30\,^{\circ}, \quad A_{pitch} = \pm 10\,^{\circ}$$

Case parameters

- Farfield at 20+ chords
- DG1 time scheme
- The order p is kept between 0 and 5
- $f_{growth} = 30\%$
- $f_{coarsen} = 5\%$

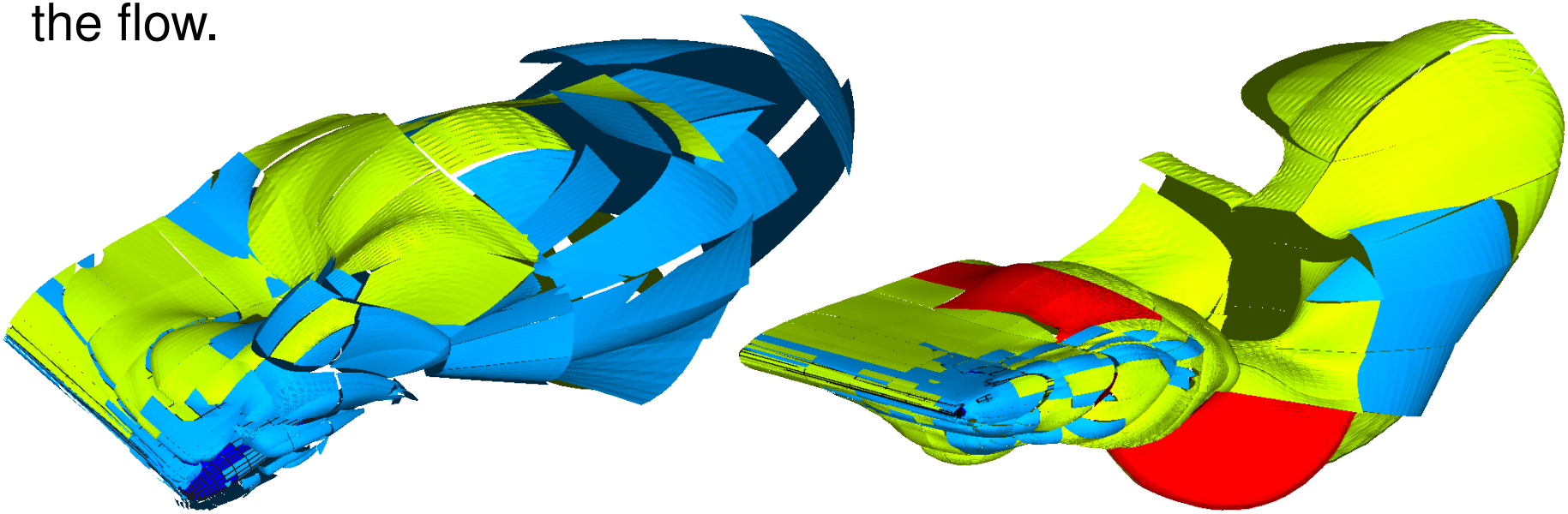


Wing geometry and kinematics

Output: Lift integrated over final 5% of simulation time

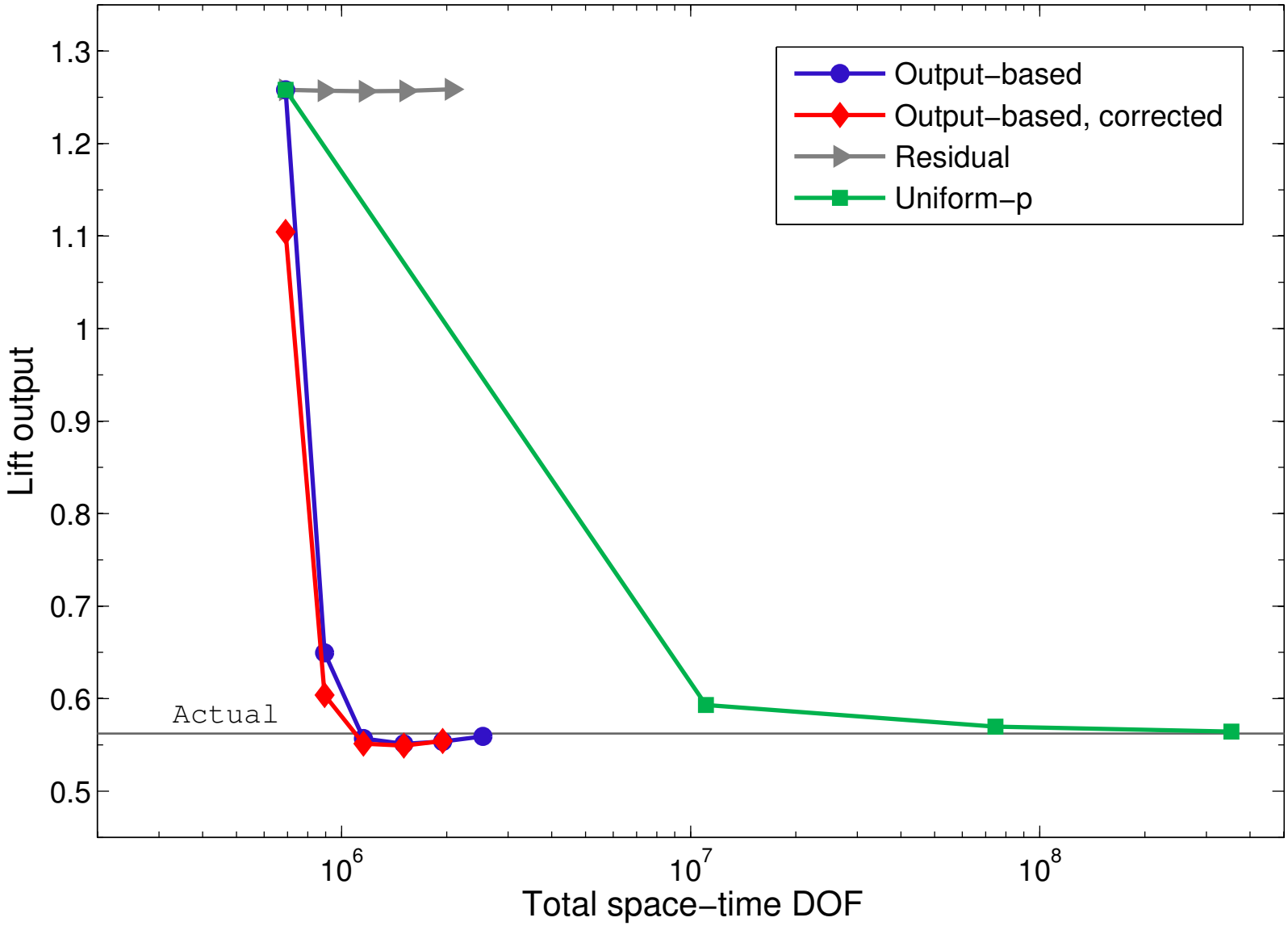
Adapted spatial meshes

Orders (0 to 3) plotted on entropy isosurfaces for two snapshots of the flow.

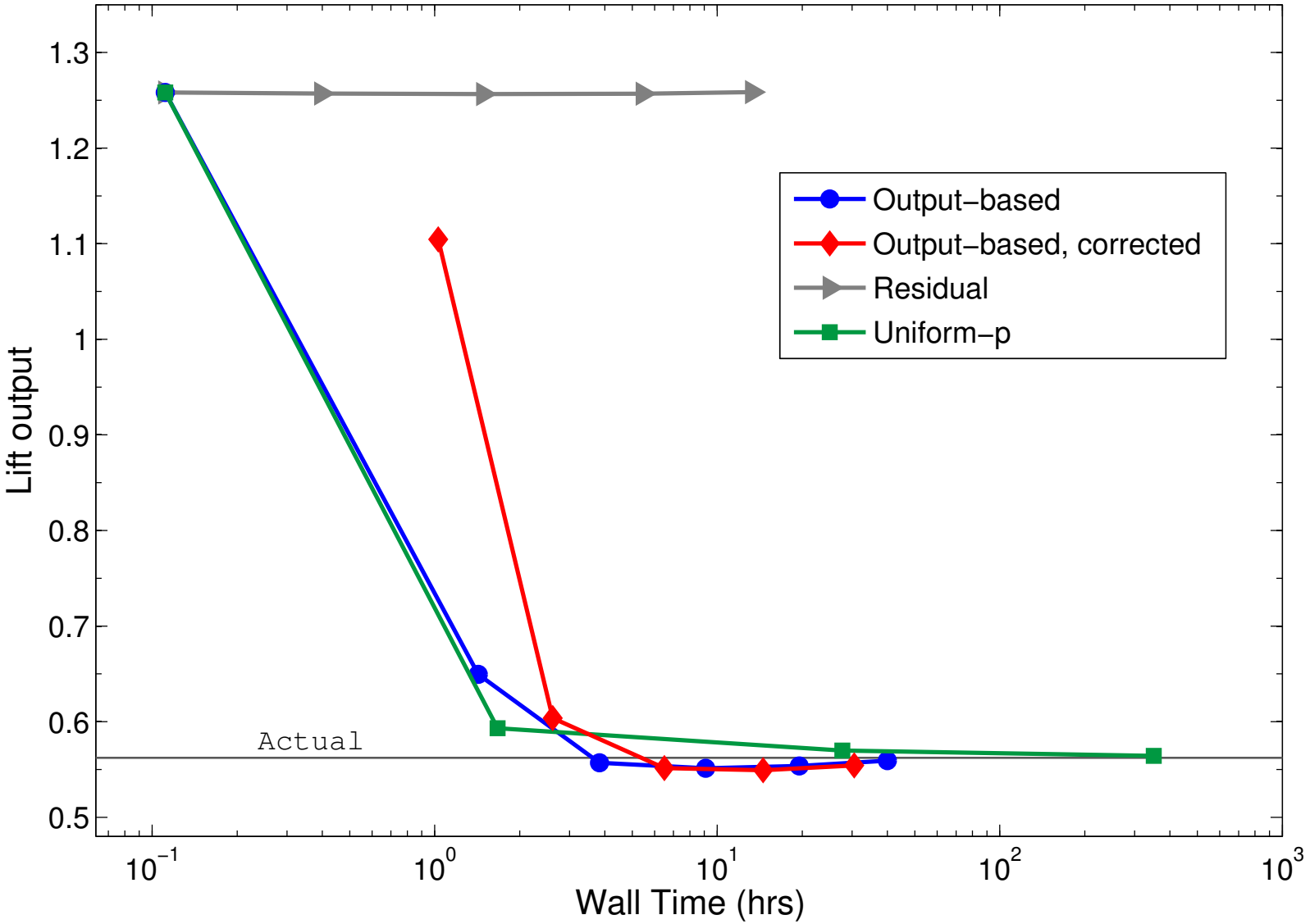


$t = 2.25T$ $t = 2.8T$

Output convergence versus DOF



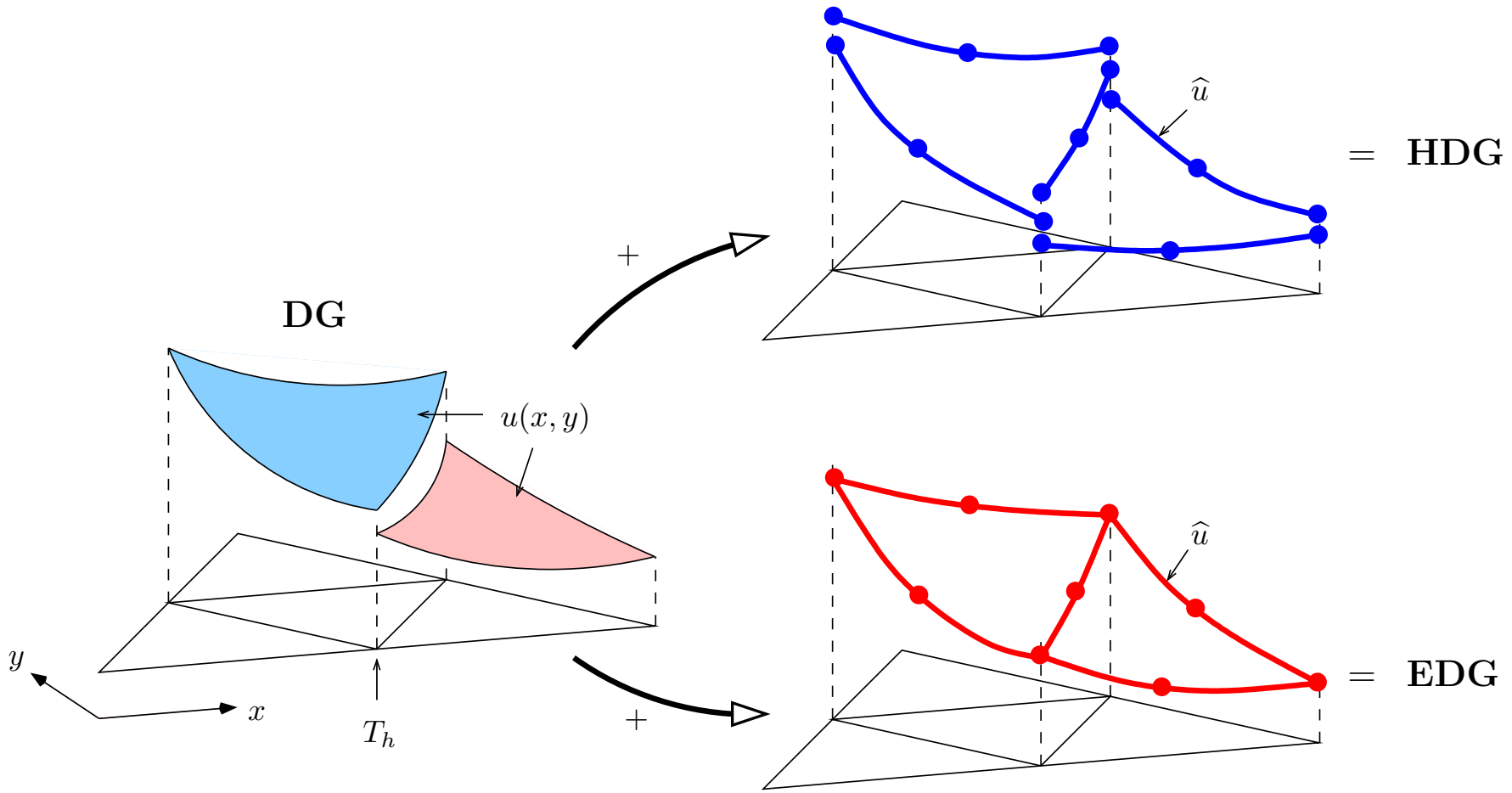
Output convergence versus CPU time



Outline

Hybridizing DG

In hybrid/mixed DG methods, we introduce *additional* unknowns on faces ($\widehat{u}$), with the intent that these will be the only globally-coupled unknowns.



Motivation: DOF count

The numbers in the below tables indicate approximately how many degrees of freedom (per equation of a system) we need per vertex of a typical mesh.

Triangles:

method	$p=1$	$p=2$	$p=3$
DG	6	12	20
CG	1	4	9
HDG	6	9	12
EDG	1	4	7

Quadrilaterals:

method	$p=1$	$p=2$	$p=3$
DG	4	9	16
CG	1	4	9
HDG	4	6	8
EDG	1	3	5

Tetrahedra:

method	$p=1$	$p=2$	$p=3$
DG	24	60	120
CG	1	8.2	27.4
HDG	36	72	120
EDG	1	8.2	27.4

Hexahedra:

method	$p=1$	$p=2$	$p=3$
DG	8	27	64
CG	1	8	27
HDG	12	27	48
EDG	1	7	19

HDG discretization

Introduce $\vec{\mathbf{q}}$ to obtain a system of PDEs

$$\begin{aligned} \vec{\mathbf{q}} - \nabla \mathbf{u} &= \mathbf{0} \\ \partial_t \mathbf{u} + \nabla \cdot \underbrace{\left[\vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \vec{\mathbf{q}}) \right]}_{\vec{\mathbf{H}}(\mathbf{u}, \vec{\mathbf{q}})} &= \mathbf{0} \end{aligned}$$

Trial functions	Test functions	Space
$\mathbf{u}$	$\mathbf{w}$	$\mathcal{V}_h$
$\vec{\mathbf{q}}$	$\vec{\mathbf{v}}$	$[\mathcal{V}_h]^d$
$\widehat{\mathbf{u}}$	$\boldsymbol{\mu}$	$\mathcal{M}_h$

$$\begin{aligned} \text{(on elements)} \quad \mathcal{V}_h &= \left\{ \mathbf{u} \in L^2(\Omega) : \mathbf{u}|_{\Omega_e} \in \mathcal{P}^{p_e}(\Omega_e) \ \forall \, \Omega_e \in T_h \right\} \\ \text{(on faces)} \quad \mathcal{M}_h &= \left\{ \widehat{\mathbf{u}} \in L^2(\mathcal{E}_h) : \widehat{\mathbf{u}}|_{\sigma_f} \in \mathcal{P}^{p_f}(\sigma_f) \ \forall \, \sigma_f \in \mathcal{E}_h \right\} \end{aligned}$$

HDG discretization (ctd.)

System of PDEs

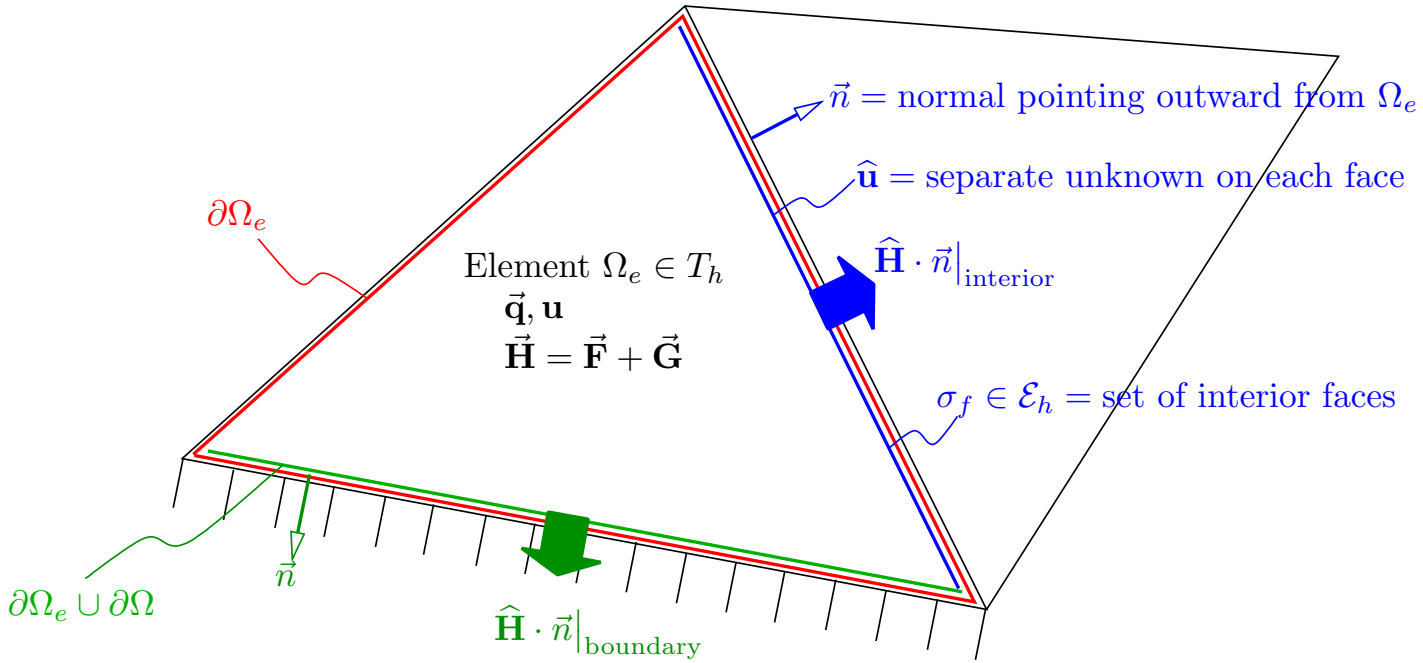
$$\begin{aligned} \vec{\mathbf{q}} - \nabla \mathbf{u} &= \mathbf{0} \\ \partial_t \mathbf{u} + \nabla \cdot \vec{\mathbf{H}}(\mathbf{u}, \vec{\mathbf{q}}) &= \mathbf{0} \end{aligned}$$

Weak form

$$\begin{aligned} (\vec{\mathbf{q}}, \vec{\mathbf{v}})_{T_h} + (\mathbf{u}, \nabla \cdot \vec{\mathbf{v}})_{T_h} - \langle \widehat{\mathbf{u}}, \vec{\mathbf{v}} \cdot \vec{n} \rangle_{\partial T_h} &= \mathbf{0}, \quad \forall \vec{\mathbf{v}} \in [\mathcal{V}_h]^d \\ (\partial_t \mathbf{u}, \mathbf{w})_{T_h} - \left(\vec{\mathbf{H}}, \nabla \mathbf{w}\right)_{T_h} + \left\langle \widehat{\vec{\mathbf{H}}} \cdot \vec{n}, \mathbf{w} \right\rangle_{\partial T_h} &= \mathbf{0}, \quad \forall \mathbf{w} \in \mathcal{V}_h \\ \left\langle \widehat{\vec{\mathbf{H}}} \cdot \vec{n}, \boldsymbol{\mu} \right\rangle_{\partial T_h \setminus \partial \Omega} &= \mathbf{0}, \quad \forall \boldsymbol{\mu} \in \mathcal{M}_h \end{aligned}$$

$$(\mathbf{u}, \mathbf{w})_{T_h} = \sum_{e=1}^{N_e} \int_{\Omega_e} \mathbf{u}^T \mathbf{w} \, d\Omega \qquad \langle \mathbf{u}, \vec{\mathbf{v}} \cdot \vec{n} \rangle_{\partial T_h} = \sum_{e=1}^{N_e} \int_{\partial \Omega_e} \mathbf{u}^{+T} \vec{\mathbf{v}}^{+} \cdot \vec{n}^{+} \, ds$$

HDG fluxes



$$\widehat{\mathbf{H}} \cdot \vec{n} = \vec{\mathbf{H}}(\widehat{\mathbf{u}}, \vec{\mathbf{q}}) \cdot \vec{n} + \underbrace{\mathbf{S}(\widehat{\mathbf{u}})(\mathbf{u} - \widehat{\mathbf{u}})}_{\text{stabilization}}$$

- Note, fluxes are *one-sided*: element-interior degrees of freedom are not directly coupled
- Stabilization borrows ideas from DG (e.g. Rusanov, Roe)

HDG approximation

$$\text{inside element } \Omega_e\text{: } \vec{\mathbf{q}}(\vec{x})\big|_{\Omega_e} = \sum_{i=1}^{d}\sum_{n=1}^{N_{pe}} \mathbf{Q}_{ein}\phi_n(\vec{x})\widehat{x}_i$$

$$\text{inside element } \Omega_e\text{: } \mathbf{u}(\vec{x})\big|_{\Omega_e} = \sum_{n=1}^{N_{pe}} \mathbf{U}_{en}\phi_n(\vec{x})$$

$$\text{on face } f\text{: } \hat{\mathbf{u}}(\vec{s})\big|_{\sigma_f} = \sum_{n=1}^{N_{pf}} \mathbf{\Lambda}_{fn}\mu_n(\vec{s})$$

- $\vec{\mathbf{q}}$ and $\mathbf{u}$ are approximated with the same basis and order, p_e on element e
- $\hat{\mathbf{u}}$ is approximated with order p_f on face f
- The only globally-coupled unknown vector will be $\mathbf{\Lambda}$

Residuals

Three types of residuals: two types inside an element, and one type on interior faces,

$$\begin{aligned}
\mathbf{R}^{Q}_{ein} &= \int_{\Omega_e} \mathbf{q}_i \phi_n \, d\Omega + \int_{\Omega_e} \mathbf{u} \partial_i \phi_n \, d\Omega - \int_{\partial\Omega_e} \widehat{\mathbf{u}} \phi_n n_i \, ds \\
\mathbf{R}^{U}_{en} &= \int_{\Omega_e} \partial_t \mathbf{u} \phi_n \, d\Omega - \int_{\Omega_e} \vec{\mathbf{H}} \cdot \nabla \phi_n \, d\Omega + \int_{\partial\Omega_e} \widehat{\mathbf{H}} \cdot \vec{n} \phi_n \, ds \\
\mathbf{R}^{\Lambda}_{fn} &= \int_{\sigma_f} \left\{ \widehat{\mathbf{H}} \cdot \vec{n} \big|_L + \widehat{\mathbf{H}} \cdot \vec{n} \big|_R \right\} \mu_n \, ds
\end{aligned}$$

Note, the integrand appearing in $\mathbf{R}^{\Lambda}$ can be re-written as

$$\widehat{\mathbf{H}} \cdot \vec{n} \big|_L + \widehat{\mathbf{H}} \cdot \vec{n} \big|_R = \left[\widehat{\mathbf{H}}(\widehat{\mathbf{u}}, \vec{\mathbf{q}}_L) - \widehat{\mathbf{H}}(\widehat{\mathbf{u}}, \vec{\mathbf{q}}_R) \right] \cdot \vec{n}_L + \mathbf{S}_L(\mathbf{u}_L - \widehat{\mathbf{u}}) + \mathbf{S}_R(\mathbf{u}_R - \widehat{\mathbf{u}})$$

$\Rightarrow$ the last set of equations is a weak statement of flux continuity

HDG residual Jacobian matrix

$$
\begin{array}{c|cccccc|c|c}
 & \mathbf{Q}_1 & \mathbf{U}_1 & \mathbf{Q}_2 & \mathbf{U}_2 & \cdots & \cdots & \cdots & \boldsymbol{\Lambda}_f & \cdots \\
\hline
\mathbf{R}_1^Q & \frac{\partial \mathbf{R}_1^Q}{\partial \mathbf{Q}_1} & \frac{\partial \mathbf{R}_1^Q}{\partial \mathbf{U}_1} & & & & & \cdots & \frac{\partial \mathbf{R}_1^Q}{\partial \boldsymbol{\Lambda}_f} & \cdots \\
\mathbf{R}_1^U & \frac{\partial \mathbf{R}_1^U}{\partial \mathbf{Q}_1} & \frac{\partial \mathbf{R}_1^U}{\partial \mathbf{U}_1} & & & & & \cdots & \frac{\partial \mathbf{R}_1^U}{\partial \boldsymbol{\Lambda}_f} & \cdots \\
\mathbf{R}_2^Q & & & \frac{\partial \mathbf{R}_2^Q}{\partial \mathbf{Q}_2} & \frac{\partial \mathbf{R}_2^Q}{\partial \mathbf{U}_2} & & & \cdots & \frac{\partial \mathbf{R}_2^Q}{\partial \boldsymbol{\Lambda}_f} & \cdots \\
\mathbf{R}_2^U & & & \frac{\partial \mathbf{R}_2^U}{\partial \mathbf{Q}_2} & \frac{\partial \mathbf{R}_2^U}{\partial \mathbf{U}_2} & & & \cdots & \frac{\partial \mathbf{R}_2^U}{\partial \boldsymbol{\Lambda}_f} & \cdots \\
\vdots & & & & & \ddots & & \cdots & \vdots & \cdots \\
\vdots & & & & & & \ddots & \cdots & \vdots & \cdots \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdot & \vdots & \cdot \\
\mathbf{R}_f^\Lambda & \frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{Q}_1} & \frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{U}_1} & \frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{Q}_2} & \frac{\partial \mathbf{R}_f^\Lambda}{\partial \mathbf{U}_2} & \cdots & \cdots & \cdots & \frac{\partial \mathbf{R}_f^\Lambda}{\partial \boldsymbol{\Lambda}_f} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdot & \vdots & \cdot
\end{array}
$$

Static condensation

$$\text{Jacobian matrix} = \left[\begin{array}{cc|c} \frac{\partial \mathbf{R}^{\mathcal{Q}}}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^{\mathcal{Q}}}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^{\mathcal{Q}}}{\partial \mathbf{\Lambda}} \\ \frac{\partial \mathbf{R}^{U}}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^{U}}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^{U}}{\partial \mathbf{\Lambda}} \\ \hline \frac{\partial \mathbf{R}^{\Lambda}}{\partial \mathbf{Q}} & \frac{\partial \mathbf{R}^{\Lambda}}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}^{\Lambda}}{\partial \mathbf{\Lambda}} \end{array}\right] = \left[\begin{array}{cc} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{array}\right]$$

- Let $\mathbf{QU} = [\mathbf{Q}, \mathbf{U}]^T$
- $\mathbf{A} = \frac{\partial \mathbf{R}^{\mathcal{QU}}}{\partial \mathbf{QU}}$ is easily invertible (element-local solves)
- Define a Schur-complement system matrix as

$$\mathbf{K} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$$

Static condensation (ctd.)

- At each Newton update, we need to solve the system

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{QU} \\ \Delta \mathbf{\Lambda} \end{bmatrix} + \begin{bmatrix} \mathbf{R}^{QU} \\ \mathbf{R}^{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

- Apply static condensation: i.e. hypothetically solve for $\mathbf{QU}$ from the first block and substitute into the second block,

$$\mathbf{K}\Delta\mathbf{\Lambda} + \underbrace{\mathbf{R}^{\Lambda} - \mathbf{C}\mathbf{A}^{-1}\mathbf{R}^{QU}}_{\widetilde{\mathbf{R}}^{\Lambda}} = \mathbf{0}$$

- Solve this (hopefully small) system for $\Delta\mathbf{\Lambda}$, and then back-substitute to get $\Delta\mathbf{QU}$

$$\Delta\mathbf{QU} = -\mathbf{A}^{-1}(\mathbf{B}\Delta\mathbf{\Lambda} + \mathbf{R}^{QU})$$

Adjoint discretization

- The adjoint system for output J is obtained by using the transpose of the primal Jacobian,

$$\begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix} \begin{bmatrix} \mathbf{\Psi}^{QU} \\ \mathbf{\Psi}^{\Lambda} \end{bmatrix} + \begin{bmatrix} \frac{\partial J}{\partial \mathbf{QU}}^T \\ \frac{\partial J}{\partial \mathbf{\Lambda}}^T \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

- Statically condensing out the element-interior degrees of freedom, we obtain the following system for the face adjoints,

$$\underbrace{\left[\mathbf{D}^T - \mathbf{B}^T \mathbf{A}^{-T} \mathbf{C}^T\right]}_{\mathbf{K}^T} \mathbf{\Psi}^{\Lambda} + \left[\frac{\partial J}{\partial \mathbf{\Lambda}}^T - \mathbf{B}^T \mathbf{A}^{-T} \frac{\partial J}{\partial \mathbf{QU}}^T \right] = \mathbf{0}$$

- Same solution procedure as primal: solve for $\mathbf{\Psi}^{\Lambda}$ first, then for $\mathbf{\Psi}^{QU}$

Error estimation and adaptation

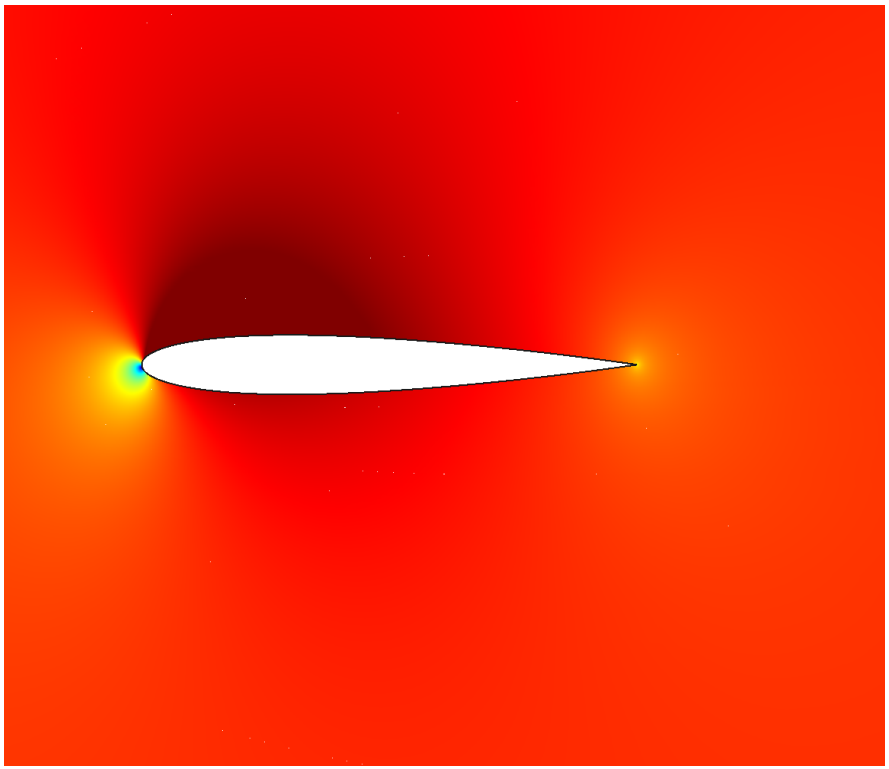
- The adjoint-weighted residual output error estimate applies to an HDG discretization,

$$\delta J \approx \underbrace{-(\boldsymbol{\Psi}_h^Q)^T \mathbf{R}_h^Q}_{\delta J^Q} \underbrace{-(\boldsymbol{\Psi}_h^U)^T \mathbf{R}_h^U}_{\delta J^U} \underbrace{-(\boldsymbol{\Psi}_h^\Lambda)^T \mathbf{R}_h^\Lambda}_{\delta J^\Lambda}$$

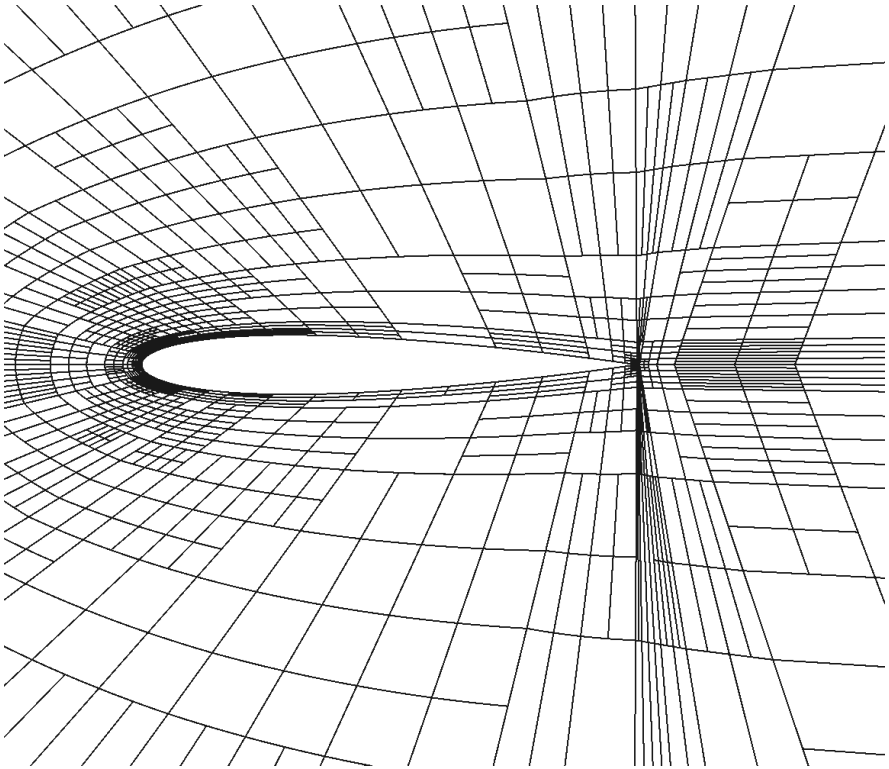
- Fine space = order increment on all elements *and* faces
- All residuals are evaluated using the coarse state injected into the fine space
- δJ^Λ is typically very small (sometimes zero)
- We only adapt on localizations of δJ^Q and δJ^U

Inviscid flow over an airfoil

NACA 0012, Euler, $M_\infty = 0.5$, $\alpha = 2^\circ$



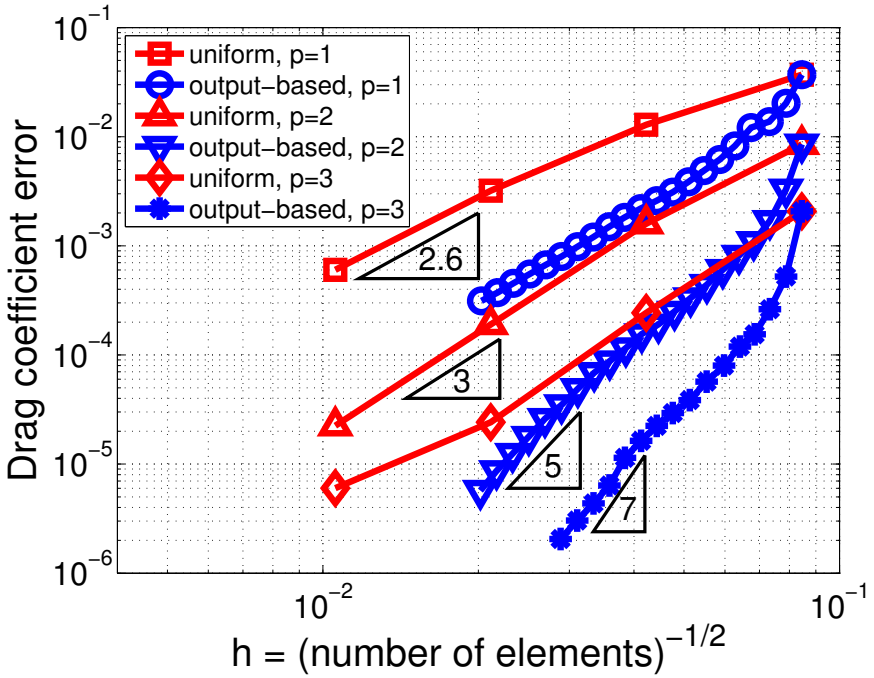
Mach contours (0–0.6)



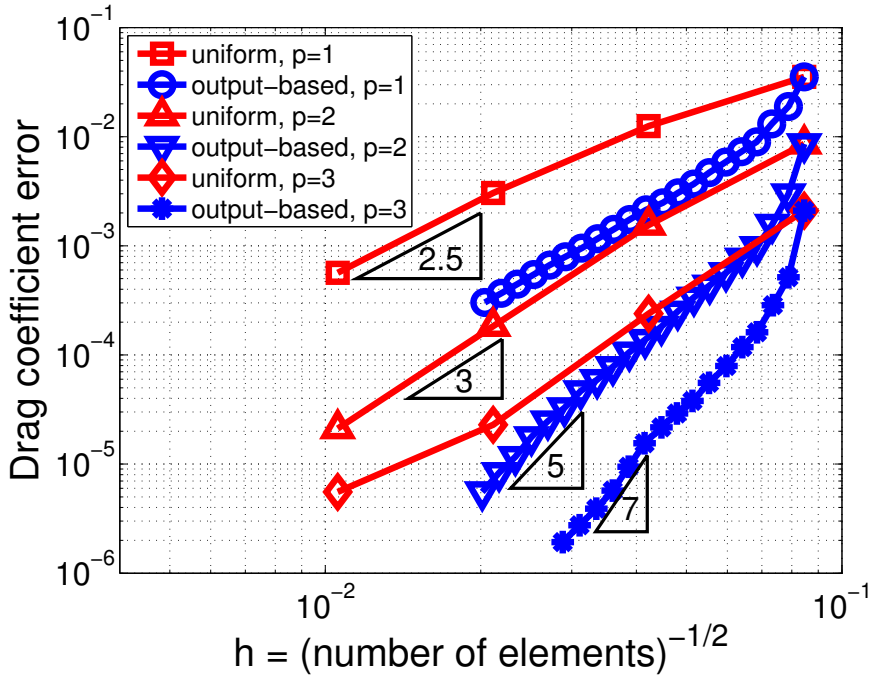
Final drag-adapted mesh for $p = 2$

- Compare adaptive refinements of DG and HDG
- Solve for Ψ_h approximately using block-Jacobi smoothing

Inviscid flow over an airfoil: results



HDG

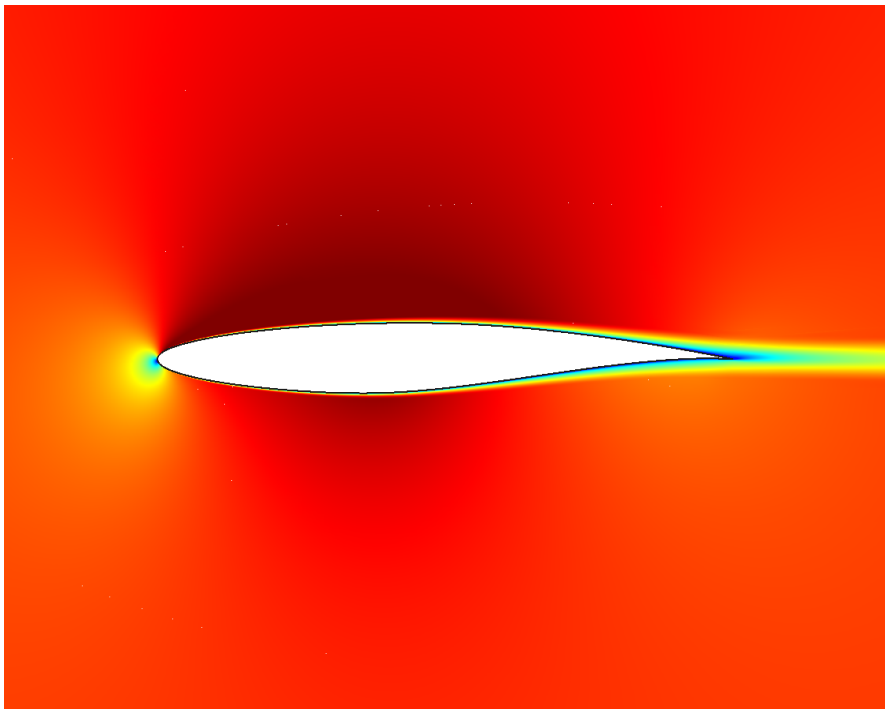


DG

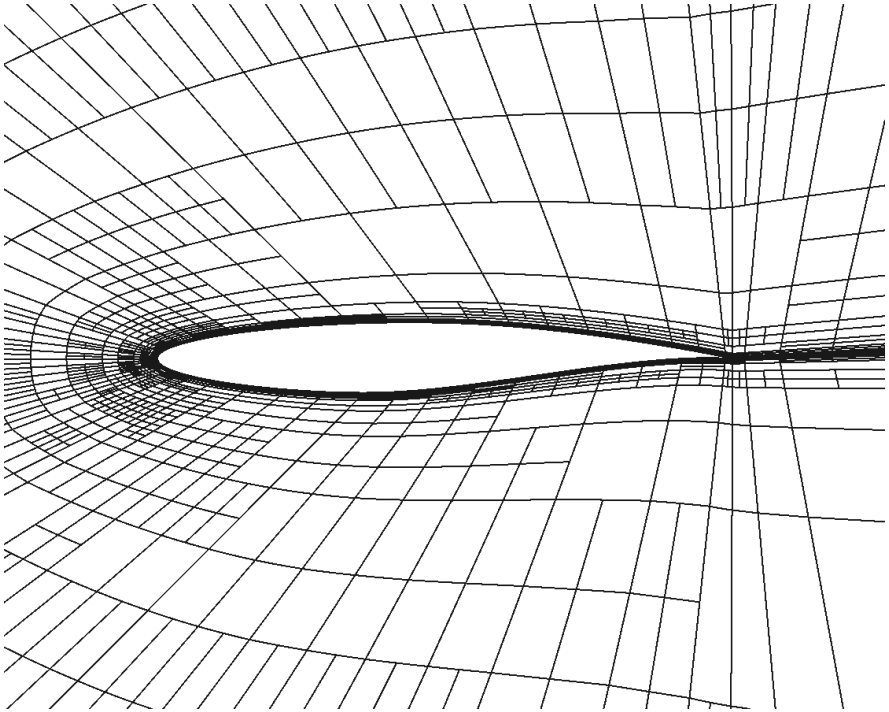
- In HDG, $\delta J^{\Lambda} = 0$ exactly in this case
- Results nearly identical between HDG and DG
- For high orders, HDG has an advantage of smaller global systems

RANS-SA flow over an airfoil

RAE 2822, RANS-SA, $M_\infty = 0.5$, $Re = 10^5$, $\alpha = 1^\circ$



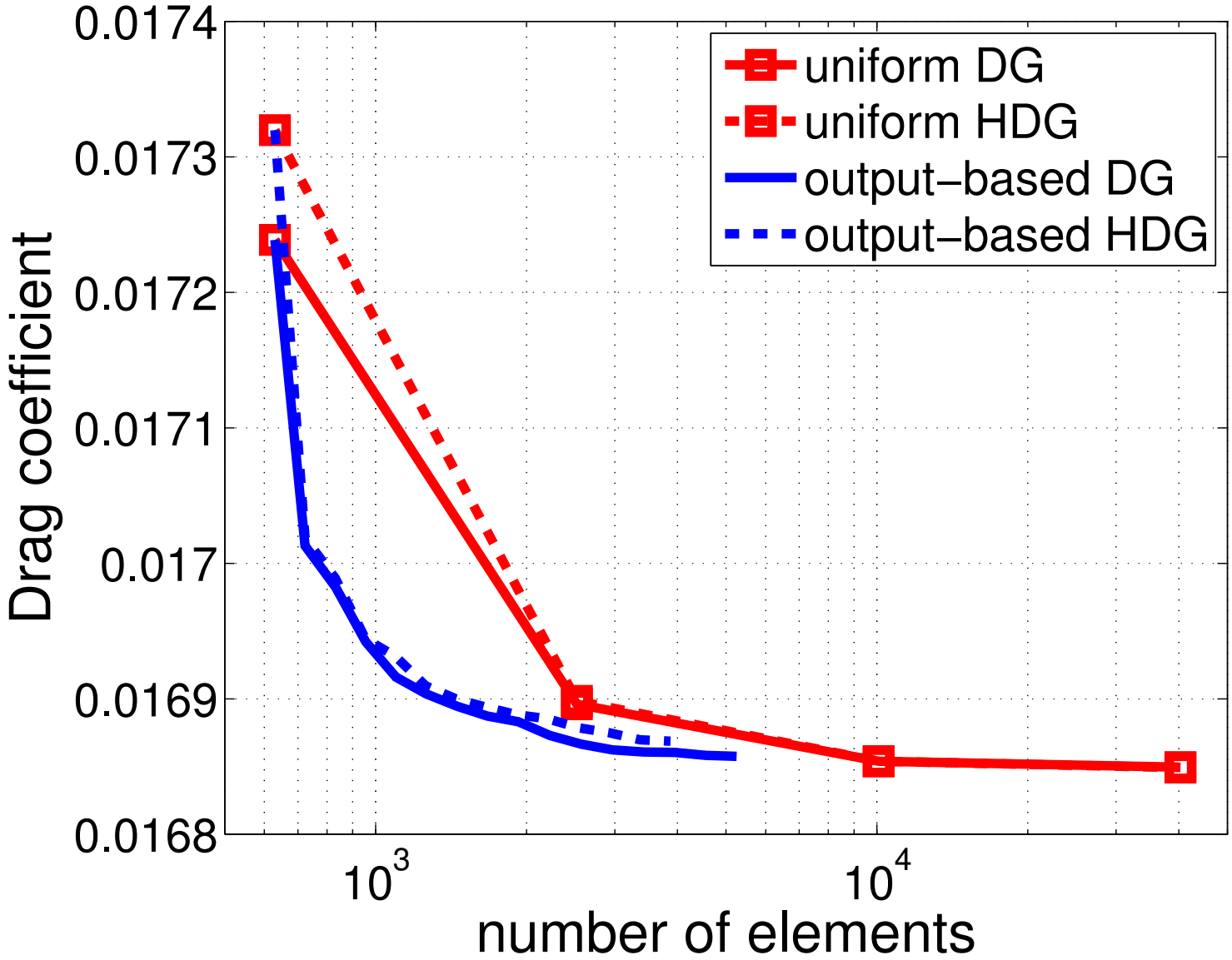
Mach contours (0–0.6)



Final drag-adapted mesh

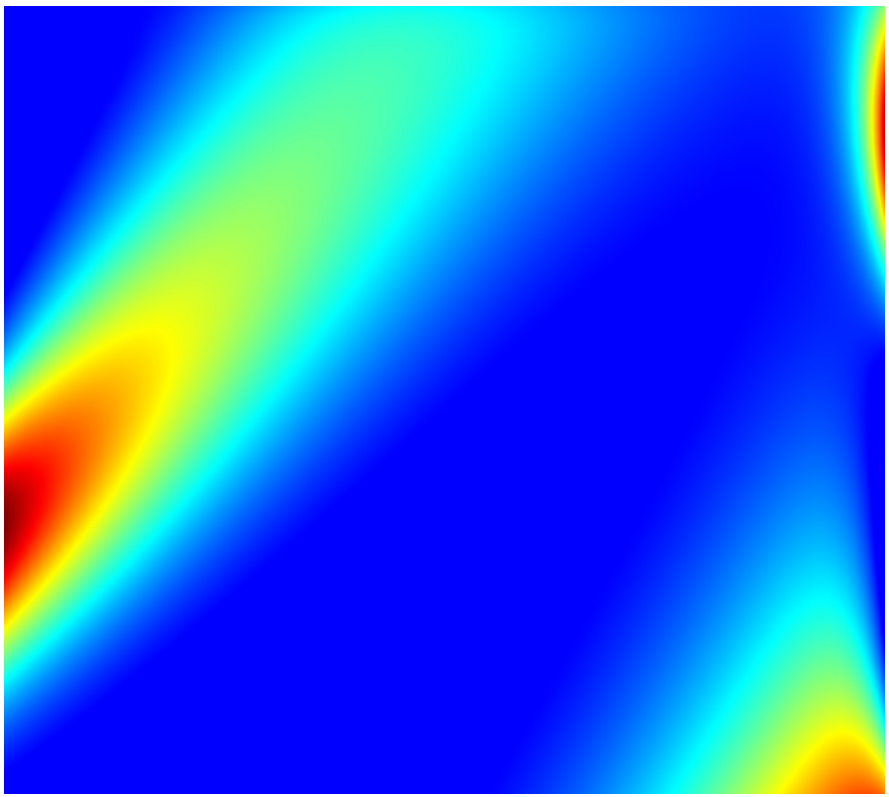
- Compare adaptive refinements of DG and HDG
- For solver robustness, the initial mesh is already tailored to resolve the boundary layer

RANS-SA flow over an airfoil: results

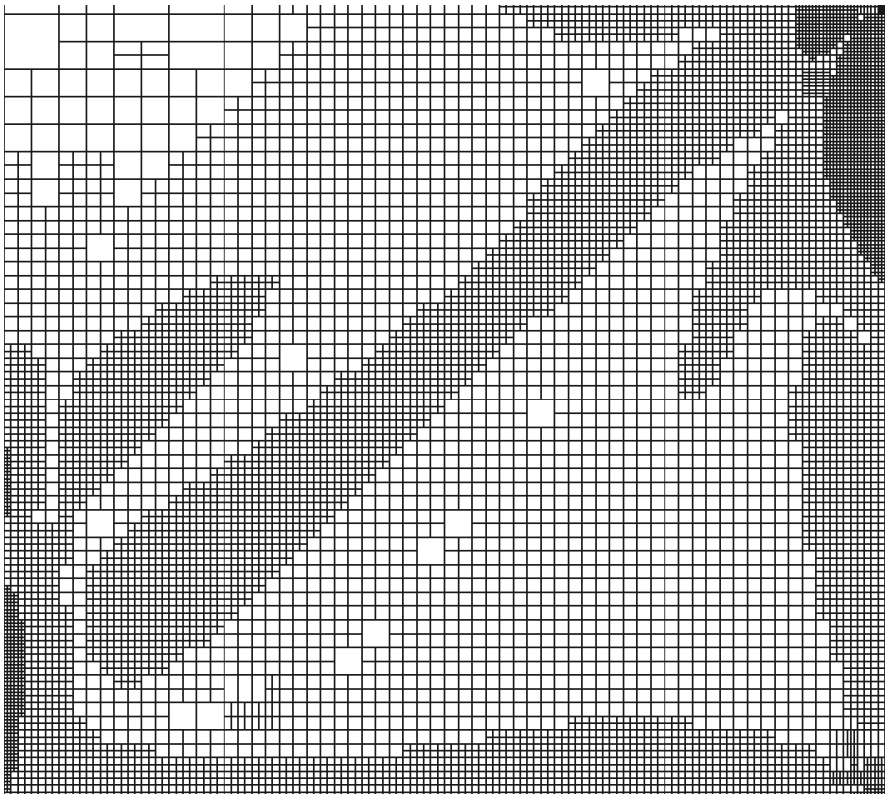


Scalar advection-diffusion

Scalar advection diffusion in a box, $Pe = 50$, Dirichlet BCs



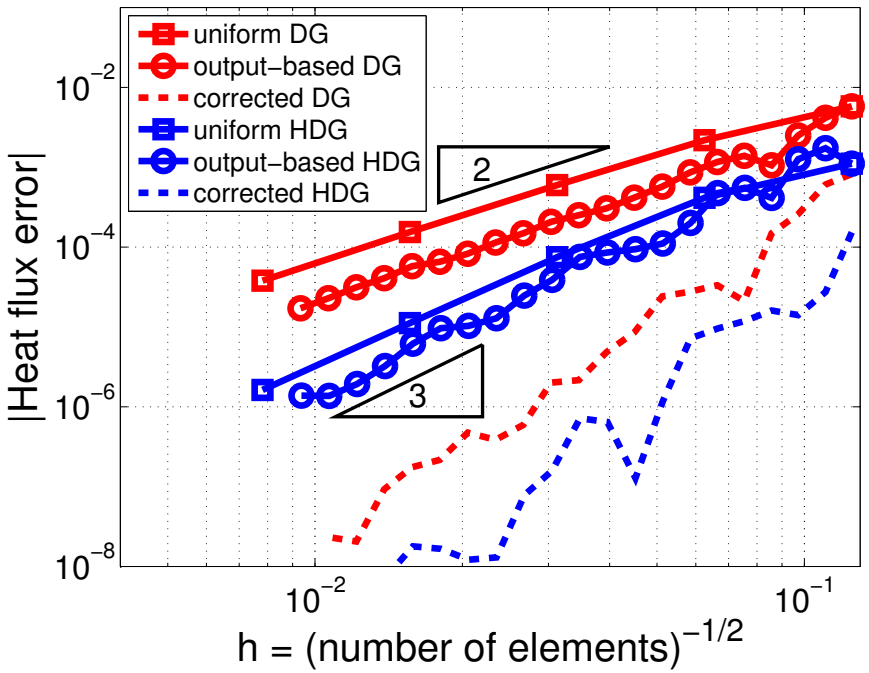
Scalar solution



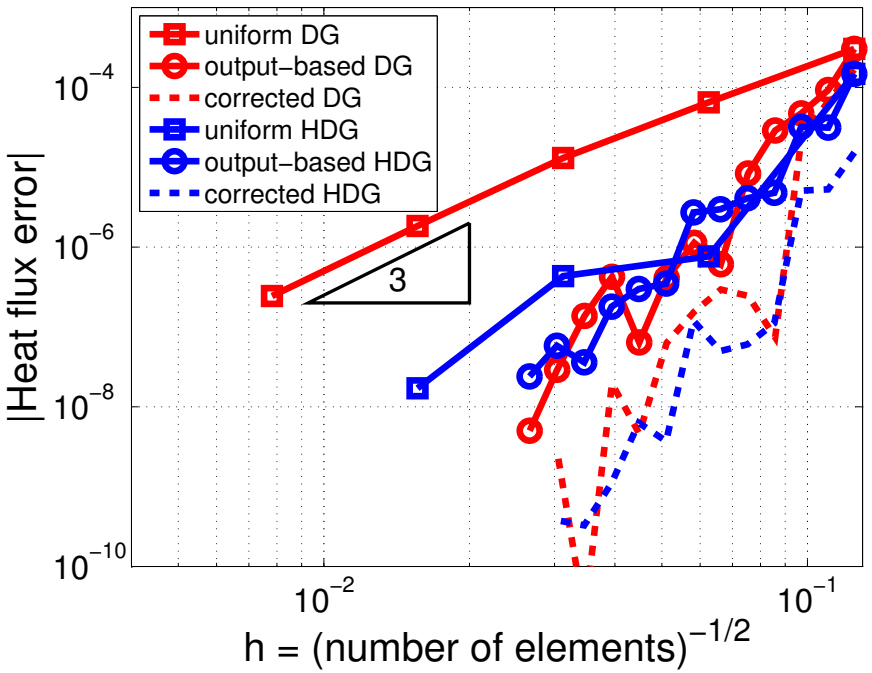
Final output-adapted mesh

- Output is the heat flux integral on the right boundary
- Compare adaptive refinements of DG and HDG

Scalar advection-diffusion: results



$p = 1$



$p = 2$

- DG and HDG now give different results
- Mixed formulation in HDG converges heat flux faster
- Output-adaptation buys an extra order of convergence when considering corrected values

Outline

Concluding remarks: summary

- Presented ideas, methods, and results for output-based adaptive aerodynamics simulations
- Used a discontinuous Galerkin (DG) finite element method for convective stability, hp adaptation
- Showed one way to address the cost of DG: hybridization
- Extended adjoint-weighted residual to unsteady and hybridized discretizations
- Showed results for 2D and 3D simulations of compressible flow, including cases with viscosity and Reynolds-averaged turbulence

Concluding remarks: key findings

- Adjoint error estimates improve robustness of CFD
- Output-based adaptation can quarantine singularities and recover optimal convergence rates
- For many steady problems, output-based adaptation saves DOFs and CPU time compared to uniform refinement or heuristic indicators
- Adaptation mechanics can be tricky, especially with curved anisotropic elements
- Unsteady problems add time as a dimension, and this generally helps adaptation
- Hybrid DG discretizations can reduce globally coupled DOFs at moderate-to-high orders

Acknowledgments

- Students and post-docs
- Collaborators
- Funding: Air Force, Department of Defense, Department of Energy, NASA, University of Michigan

— Thank you —