# Mesh adaptation for finite volume methods

## With some hints, tricks, suggestions, better practice

Dr. J.-D. Müller
School of Engineering and Materials Science,
Queen Mary, University of London
j.mueller@qmul.ac.uk

About Flow Mesh Adapation and Modification Workshop
Version 1, June 2015

©: Jens-Dominik Müller, 2015

# Ingredients for mesh adaptation

**Sensor:** Estimates local error of the discretisation. Ideally should be isotropic: in which direction do we need to refine or coarsen the mesh.

**Weighting:** Weight the local errors with their effect on a goal function, i.e. the adjoint.

**Mesh modification:** How to change the mesh? And related components such as multigrid levels?

**Error correction:** Rather than refining the mesh, we can instead use the estimated error correct the linear error in the goal function. And refine only for the non-linear/non-computable part.

# Ingredients for mesh adaptation

**Sensor:** Estimates local error of the discretisation. Ideally should be isotropic: in which direction do we need to refine or coarsen the mesh.

**Weighting:** Weight the local errors with their effect on a goal function, i.e. the adjoint.

**Mesh modification:** How to change the mesh? And related components such as multigrid levels?

**Error correction:** Rather than refining the mesh, we can instead use the estimated error correct the linear error in the goal function. And refine only for the non-linear/non-computable part.

# Ingredients for mesh adaptation

**Sensor:** Estimates local error of the discretisation. Ideally should be isotropic: in which direction do we need to refine or coarsen the mesh.

**Weighting:** Weight the local errors with their effect on a goal function, i.e. the adjoint.

**Mesh modification:** How to change the mesh? And related components such as multigrid levels?

**Error correction:** Rather than refining the mesh, we can instead use the estimated error correct the linear error in the goal function. And refine only for the non-linear/non-computable part.

# Ingredients for mesh adaptation

**Sensor:** Estimates local error of the discretisation. Ideally should be isotropic: in which direction do we need to refine or coarsen the mesh.

**Weighting:** Weight the local errors with their effect on a goal function, i.e. the adjoint.

**Mesh modification:** How to change the mesh? And related components such as multigrid levels?

**Error correction:** Rather than refining the mesh, we can instead use the estimated error correct the linear error in the goal function. And refine only for the non-linear/non-computable part.

# Contents

# Contents

# Commonly used types of sensors for finite volume methods

**Truncation-error:** Use simple Taylor analysis to link errors to first or second gradients in the solution.

**Residual:** Evaluate the residual on a shifted control-volume, e.g. the element in cell-vertex or finite element methods.

**Artificial viscosity:** Evaluate the magnitude of A.V. by comparing control volume residuals with and without A.V.

**Sub-division:** Locally refine the mesh and linearly interpolate the solution, evaluate the residual on the refined grid.

# Truncation error estimation

- Gradient- or Hessian-based error estimators assume the error behaves as simple Taylor expansion terms that are truncated:

$$\left.\frac{\partial u}{\partial x}\right|_{bkwd} = \frac{u_i - u_{i-1}}{h} = \frac{\partial u}{\partial x} + h\frac{\partial^2 u}{\partial x^2} + O(h^2)$$

- Essentially analyses the discretisation in finite difference formulation, difficult to compute for higher-order finite volume schemes on unstructured grids.
- The actual truncation error depends on much more than the spatial gradients. Which variable?
- A mapping would be needed from errors in the primitive variables to residuals to allow adjoint weighting:

$$\Delta J = \frac{\partial J}{\partial u}\Delta u = \frac{\partial J}{\partial R}\frac{\partial R}{\partial u}\Delta u = v^T\frac{\partial R}{\partial u}\Delta u$$

# Truncation error estimation

- Gradient- or Hessian-based error estimators assume the error behaves as simple Taylor expansion terms that are truncated:

$$\left.\frac{\partial u}{\partial x}\right|_{bkwd} = \frac{u_i - u_{i-1}}{h} = \frac{\partial u}{\partial x} + h\frac{\partial^2 u}{\partial x^2} + O(h^2)$$

- Essentially analyses the discretisation in finite difference formulation, difficult to compute for higher-order finite volume schemes on unstructured grids.
- The actual truncation error depends on much more than the spatial gradients. Which variable?
- A mapping would be needed from errors in the primitive variables to residuals to allow adjoint weighting:

$$\Delta J = \frac{\partial J}{\partial u}\Delta u = \frac{\partial J}{\partial R}\frac{\partial R}{\partial u}\Delta u = v^T\frac{\partial R}{\partial u}\Delta u$$

# Truncation error estimation

- Gradient- or Hessian-based error estimators assume the error behaves as simple Taylor expansion terms that are truncated:

$$\left.\frac{\partial u}{\partial x}\right|_{bkwd} = \frac{u_i - u_{i-1}}{h} = \frac{\partial u}{\partial x} + h\frac{\partial^2 u}{\partial x^2} + O(h^2)$$

- Essentially analyses the discretisation in finite difference formulation, difficult to compute for higher-order finite volume schemes on unstructured grids.
- The actual truncation error depends on much more than the spatial gradients. Which variable?
- A mapping would be needed from errors in the primitive variables to residuals to allow adjoint weighting:

$$\Delta J = \frac{\partial J}{\partial u}\Delta u = \frac{\partial J}{\partial R}\frac{\partial R}{\partial u}\Delta u = v^T\frac{\partial R}{\partial u}\Delta u$$

# Truncation error estimation

- Gradient- or Hessian-based error estimators assume the error behaves as simple Taylor expansion terms that are truncated:

$$\left.\frac{\partial u}{\partial x}\right|_{bkwd} = \frac{u_i - u_{i-1}}{h} = \frac{\partial u}{\partial x} + h\frac{\partial^2 u}{\partial x^2} + O(h^2)$$

- Essentially analyses the discretisation in finite difference formulation, difficult to compute for higher-order finite volume schemes on unstructured grids.
- The actual truncation error depends on much more than the spatial gradients. Which variable?
- A mapping would be needed from errors in the primitive variables to residuals to allow adjoint weighting:

$$\Delta J = \frac{\partial J}{\partial u}\Delta u = \frac{\partial J}{\partial R}\frac{\partial R}{\partial u}\Delta u = v^T\frac{\partial R}{\partial u}\Delta u$$

# Contents

# Residual-based error estimation

- In finite element methods the mass matrix is not diagonalised ("mass-lumped") which links nodal and element residuals. This provides the Galerkin orthogonality: error and test functions are orthogonal.
- The residual evaluated at the node is driven to zero by time-stepping, but will typically not be zero over other shifted contours, e.g. the element.
- The residual error over the shifted integral converges as well, due to Galerkin orthogonality, but not at the same rate as at the node. It can hence be used formally to estimate the error.
- Weight this local residual with the adjoint.
- Error analysis in finite elements is well-developed, can be formalised well for linear problems.

# Residual-based error estimation

- In finite element methods the mass matrix is not diagonalised ("mass-lumped") which links nodal and element residuals. This provides the Galerkin orthogonality: error and test functions are orthogonal.
- The residual evaluated at the node is driven to zero by time-stepping, but will typically not be zero over other shifted contours, e.g. the element.
- The residual error over the shifted integral converges as well, due to Galerkin orthogonality, but not at the same rate as at the node. It can hence be used formally to estimate the error.
- Weight this local residual with the adjoint.
- Error analysis in finite elements is well-developed, can be formalised well for linear problems.

# Residual-based error estimation

- In finite element methods the mass matrix is not diagonalised ("mass-lumped") which links nodal and element residuals. This provides the Galerkin orthogonality: error and test functions are orthogonal.
- The residual evaluated at the node is driven to zero by time-stepping, but will typically not be zero over other shifted contours, e.g. the element.
- The residual error over the shifted integral converges as well, due to Galerkin orthogonality, but not at the same rate as at the node. It can hence be used formally to estimate the error.
- Weight this local residual with the adjoint.
- Error analysis in finite elements is well-developed, can be formalised well for linear problems.

# Residual-based error estimation

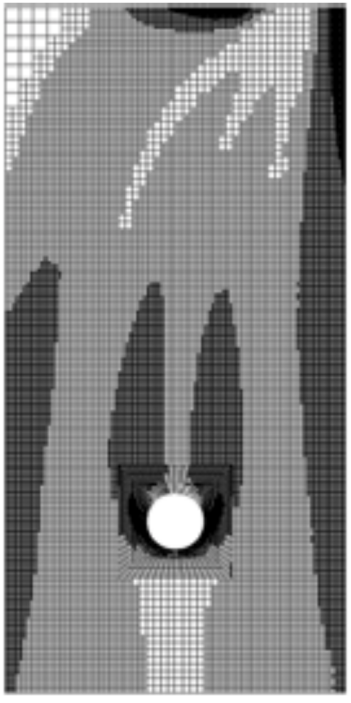
- In finite element methods the mass matrix is not diagonalised ("mass-lumped") which links nodal and element residuals. This provides the Galerkin orthogonality: error and test functions are orthogonal.
- The residual evaluated at the node is driven to zero by time-stepping, but will typically not be zero over other shifted contours, e.g. the element.
- The residual error over the shifted integral converges as well, due to Galerkin orthogonality, but not at the same rate as at the node. It can hence be used formally to estimate the error.
- Weight this local residual with the adjoint.
- Error analysis in finite elements is well-developed, can be formalised well for linear problems.

# Residual-based error estimation

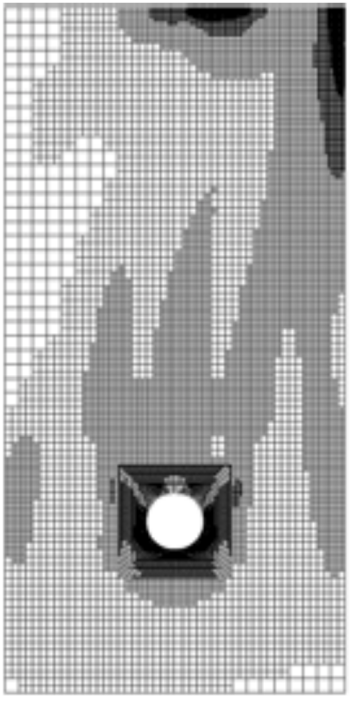
- In finite element methods the mass matrix is not diagonalised ("mass-lumped") which links nodal and element residuals. This provides the Galerkin orthogonality: error and test functions are orthogonal.
- The residual evaluated at the node is driven to zero by time-stepping, but will typically not be zero over other shifted contours, e.g. the element.
- The residual error over the shifted integral converges as well, due to Galerkin orthogonality, but not at the same rate as at the node. It can hence be used formally to estimate the error.
- Weight this local residual with the adjoint.
- Error analysis in finite elements is well-developed, can be formalised well for linear problems.

# Residual-based error estimation in h-p finite elements

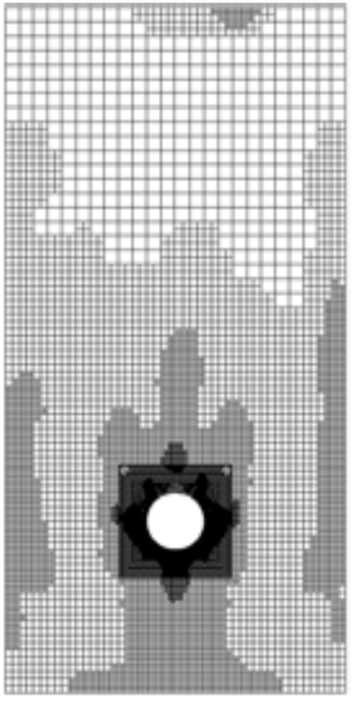
flow



vorticity



sensor: residual

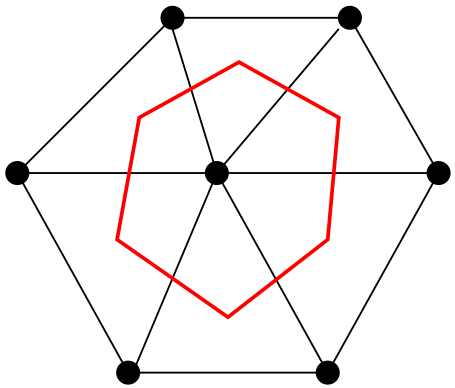


goal-oriented (Drag)

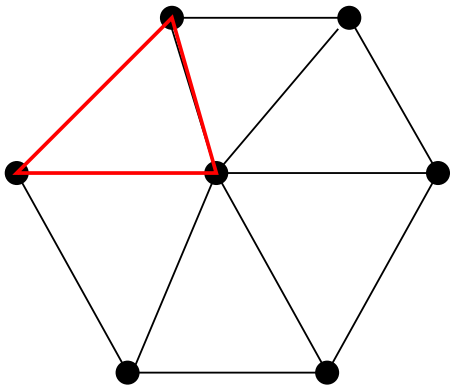
(Source: Becker, Rannacher: Acta Numerica, 2001)

# Application to finite volumes

- Extension to finite volumes: in node-centred discretisations evaluate the residual over the cell/element without any artificial viscosity terms (e.g. a central scheme), use the magnitude of this residual as a sensor.



$R_{Node} = 0$

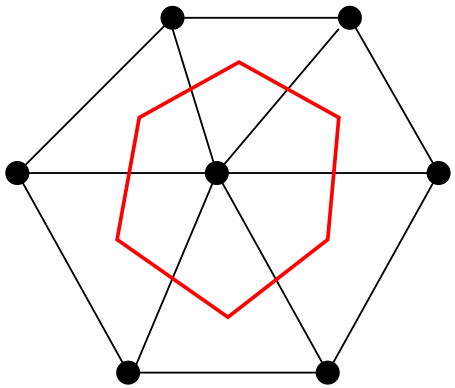


$R_{Cell} \neq 0$

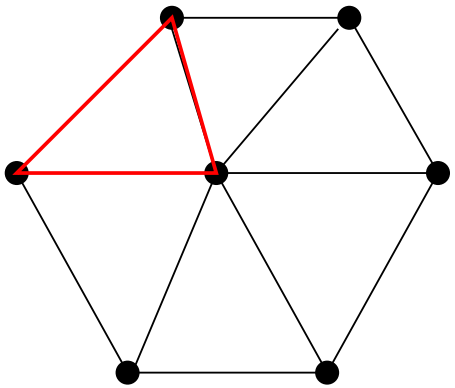
- Issue: super-convergence/noise. The FV does not use a mass matrix. Chequer-board modes in the cell residual cancel out at the nodes, the discretisation is transparent to these modes, they are not damped.

# Application to finite volumes

- Extension to finite volumes: in node-centred discretisations evaluate the residual over the cell/element without any artificial viscosity terms (e.g. a central scheme), use the magnitude of this residual as a sensor.



$R_{Node} = 0$

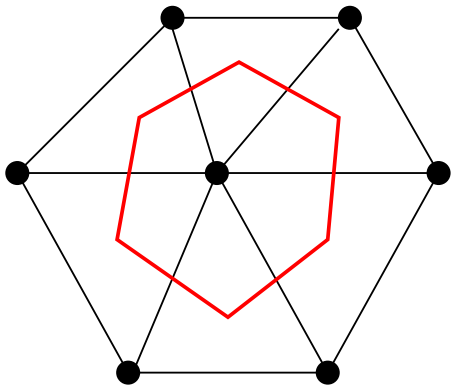


$R_{Cell} \neq 0$

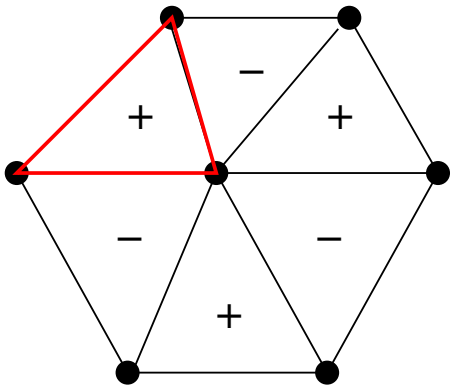
- Issue: super-convergence/noise. The FV does not use a mass matrix. Chequer-board modes in the cell residual cancel out at the nodes, the discretisation is transparent to these modes, they are not damped.

# Application to finite volumes

- Extension to finite volumes: in node-centred discretisations evaluate the residual over the cell/element without any artificial viscosity terms (e.g. a central scheme), use the magnitude of this residual as a sensor.



$R_{Node} = 0$




$R_{Cell} \neq 0$

- Issue: super-convergence/noise. The FV does not use a mass matrix. Chequer-board modes in the cell residual cancel out at the nodes, the discretisation is transparent to these modes, they are not damped.

# Adjoint-weighted residual adaptation in finite volumes

- Evaluate the cell-based residual $R_C(u_h)$, e.g. Euler Eq.:

$$R(u) \equiv \frac{\partial F(u)}{\partial x} + \frac{\partial G(u)}{\partial y} = 0$$

Residual error on $N$-noded element:

$$R_{Cell} = \iint_T \nabla \cdot (F_h, G_h)\, dA = \int_{\partial T} (F_h, G_h) \cdot \mathbf{n}\, ds$$
$$= \sum_{k=1}^{N} (\overline{F_h}, \overline{G_h})_k \cdot \mathbf{n}_k\, \Delta s_k$$

- and weight with the cell-averaged adjoint $\bar{v}$:

$$\delta J_{Cell} = \iint_{Cell} v_h^T R(u_h)\, dA \approx \overline{v_h}^T \iint_{Cell} R(u_h)\, dA. = v^T R_{Cell}$$

# Adjoint-weighted residual adaptation in finite volumes

- Evaluate the cell-based residual $R_C(u_h)$, e.g. Euler Eq.:

$$R(u) \equiv \frac{\partial F(u)}{\partial x} + \frac{\partial G(u)}{\partial y} = 0$$

Residual error on $N$-noded element:

$$R_{Cell} = \iint_T \nabla \cdot (F_h, G_h)\, dA = \int_{\partial T} (F_h, G_h) \cdot \mathbf{n}\, ds$$
$$= \sum_{k=1}^{N} (\overline{F_h}, \overline{G_h})_k \cdot \mathbf{n}_k\, \Delta s_k$$

- and weight with the cell-averaged adjoint $\bar{v}$:

$$\delta J_{Cell} = \iint_{Cell} v_h^T R(u_h)\, dA \approx \overline{v_h}^T \iint_{Cell} R(u_h)\, dA. = v^T R_{Cell}$$

# Adjoint-weighted residual adaptation in finite volumes

- Evaluate the cell-based residual $R_C(u_h)$, e.g. Euler Eq.:

$$R(u) \equiv \frac{\partial F(u)}{\partial x} + \frac{\partial G(u)}{\partial y} = 0$$

Residual error on $N$-noded element:

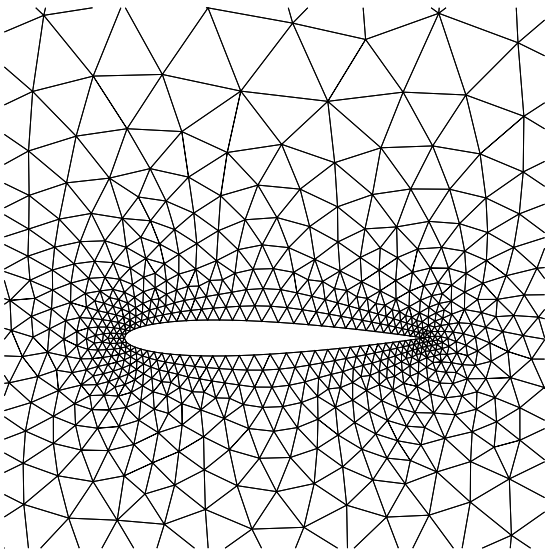
$$R_{Cell} = \iint_T \nabla \cdot (F_h, G_h)\, dA = \int_{\partial T} (F_h, G_h) \cdot \mathbf{n}\, ds$$
$$= \sum_{k=1}^{N} (\overline{F_h}, \overline{G_h})_k \cdot \mathbf{n}_k\, \Delta s_k$$

- and weight with the cell-averaged adjoint $\bar{v}$:

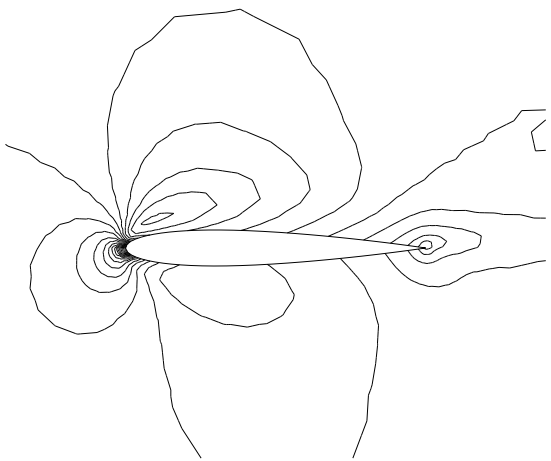
$$\delta J_{Cell} = \iint_{Cell} v_h^T R(u_h)\, dA \approx \overline{v_h}^T \iint_{Cell} R(u_h)\, dA. = v^T R_{Cell}$$

# Example: subsonic aerofoil in inviscid flow

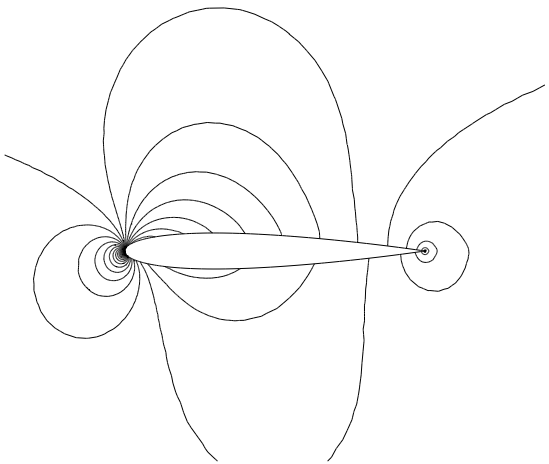
NACA 0012, Ma=0.4, $\alpha = 2°$



initial grid



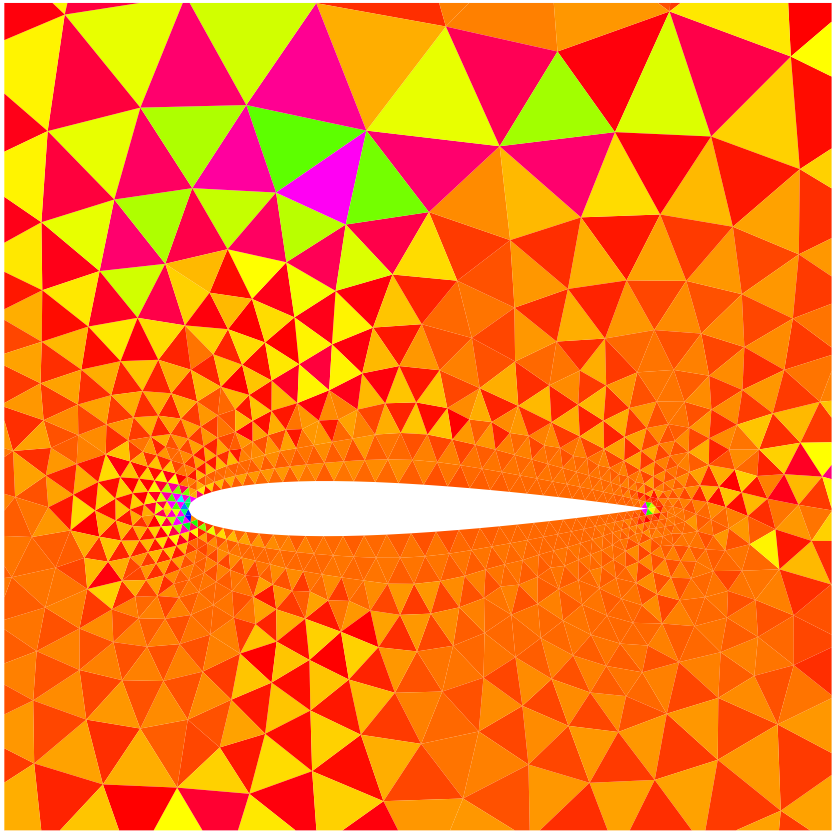
Mach, initial



Mach, 5 levels

# Residual sensor

Mass flux residual, NACA 0012, Ma=0.4, $\alpha = 2^\circ$

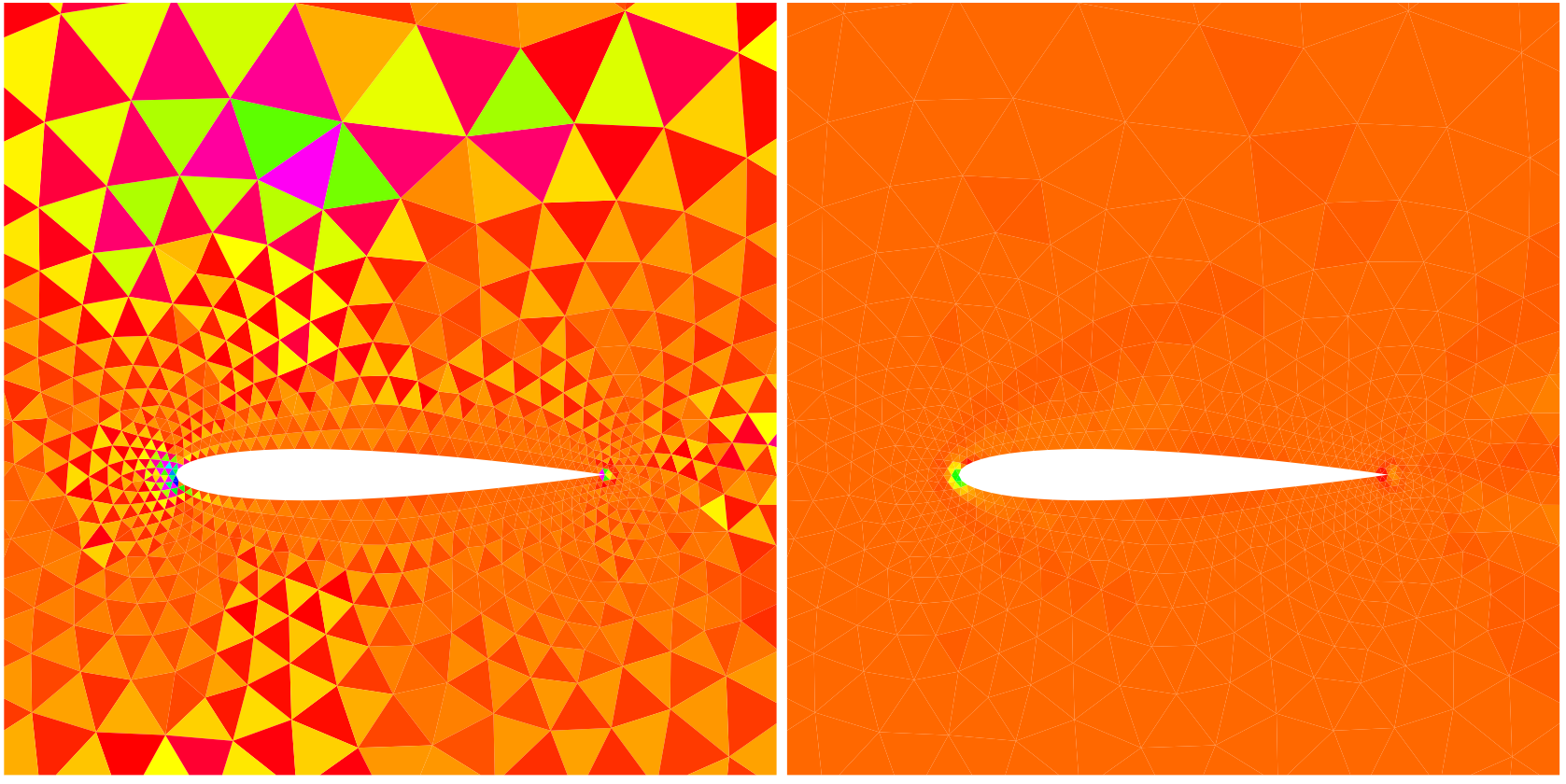


no smoothing

1 gather-scatter step

# Residual sensor

Mass flux residual, NACA 0012, Ma=0.4, $\alpha = 2^\circ$



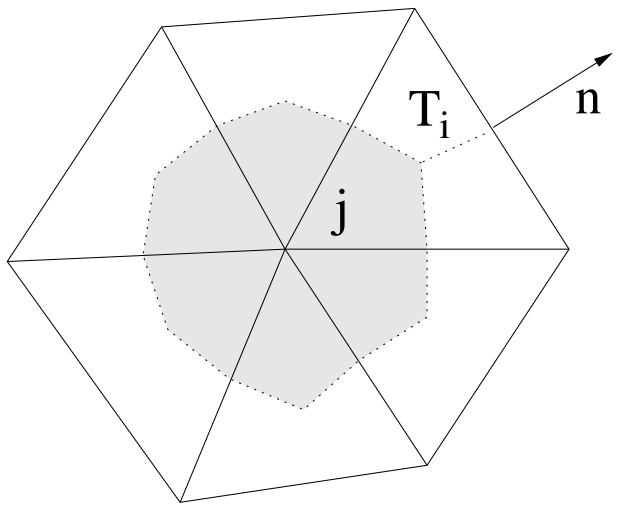
no smoothing

1 gather-scatter step

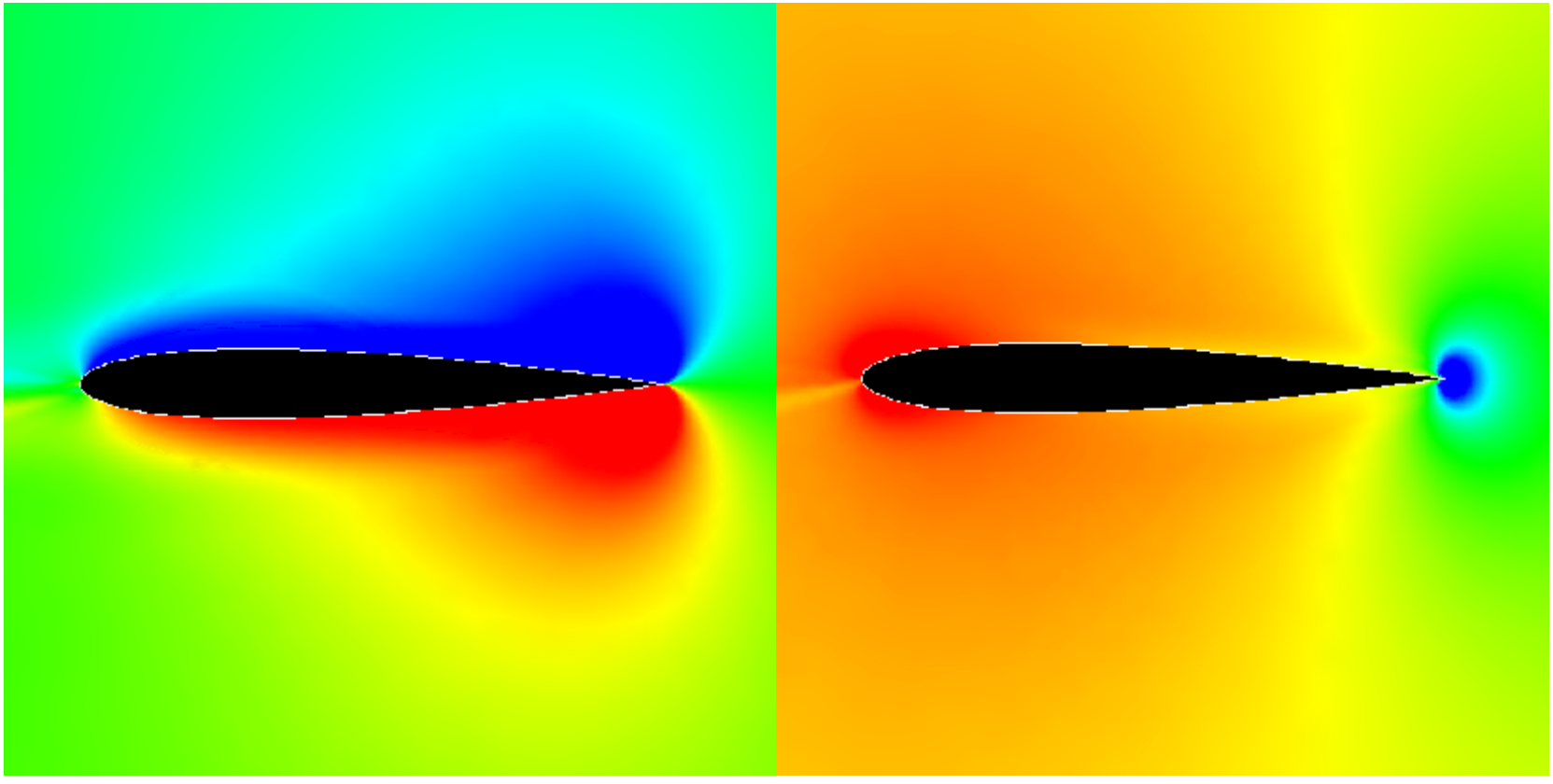
# Regularising element-based residual indicators

gather nodal residual: $$R_j^{Node} = \sum_{\text{Cells}} \frac{1}{3} R_i^{Cell}$$

scatter to interpolate on triangle: $$R_i^{tri} = \sum_{\text{Nodes}} \frac{A_j}{A_i} R_j^{node}$$



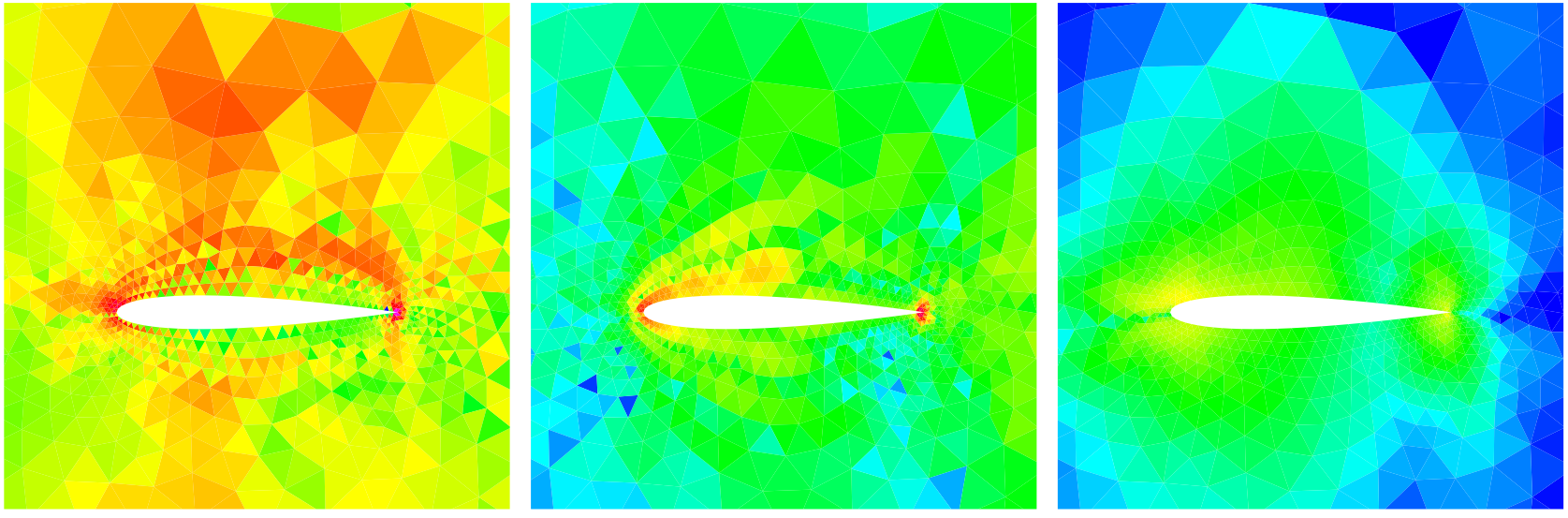
# Subsonic Airfoil: Adjoint Solution for Lift



mass flux v-velocity

# Smoothing $|v||f|$

NACA 0012, Ma=0.4, $\alpha = 2^\circ$



0 1 30 g-s steps

# Sensors Tested

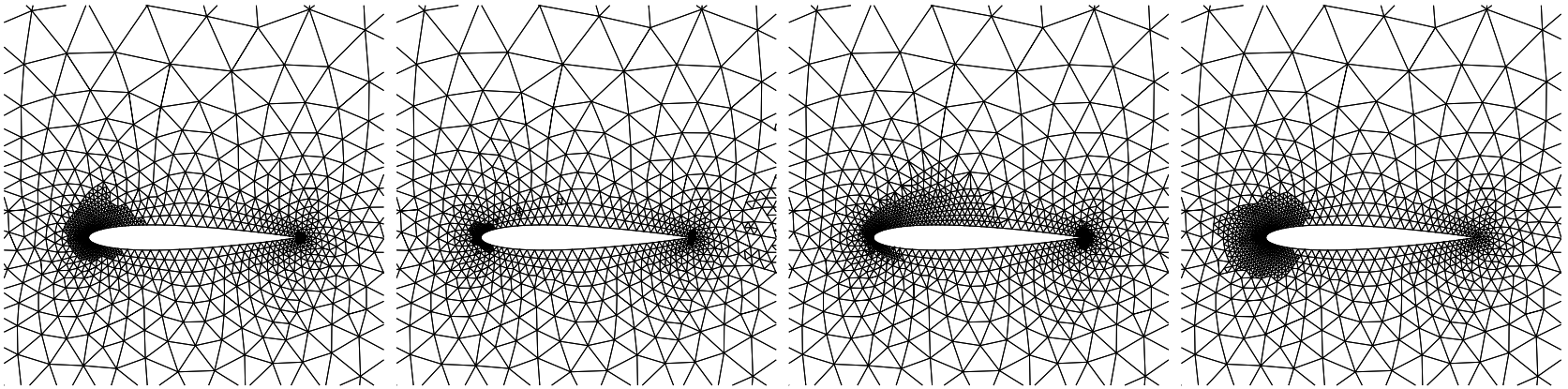
| | |
|---|---|
| $\lvert\Delta p\rvert$ | 1st differences of pressure along edges |
| $\lvert f\rvert$ | unweighted residual |
| $\lvert v\rvert\lvert f\rvert$ | adjoint weighted residual, 1 gather-scatter |
| $\lvert v\rvert\lvert f\rvert$, 30 g-s | adjoint weighted residual, 30 gather-scatter |

de-refinement and refinement

| | |
|---|---|
| either | one mean deviation below/above the average |
| or | bottom 10%, top 30% |

# Example: subsonic aerofoil in inviscid flow

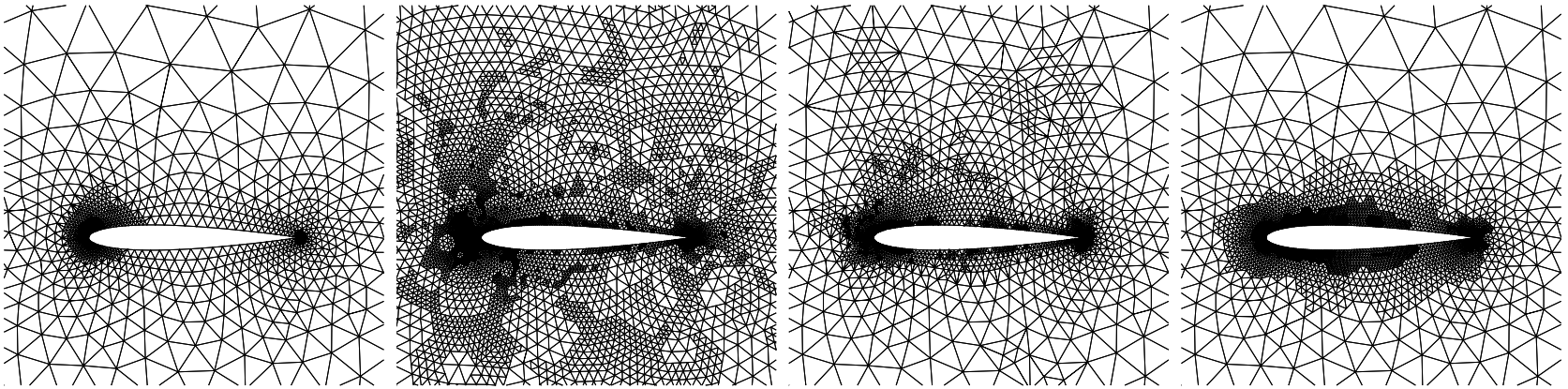
NACA 0012, Ma=0.4, $\alpha = 2°$, 1st level

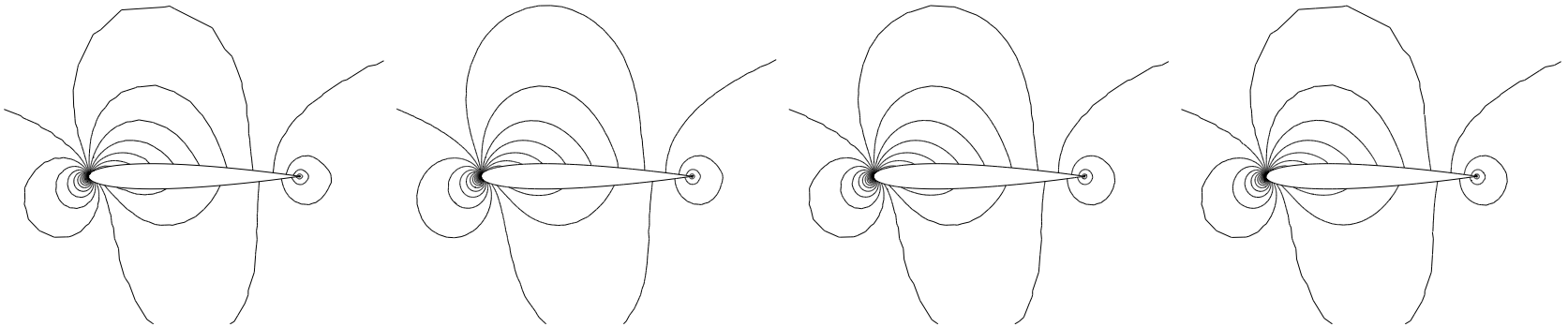


$|\Delta p|$ $|f|$ $|v||f|$ $|v||f|$, 30 g-s

# Example: subsonic aerofoil in inviscid flow

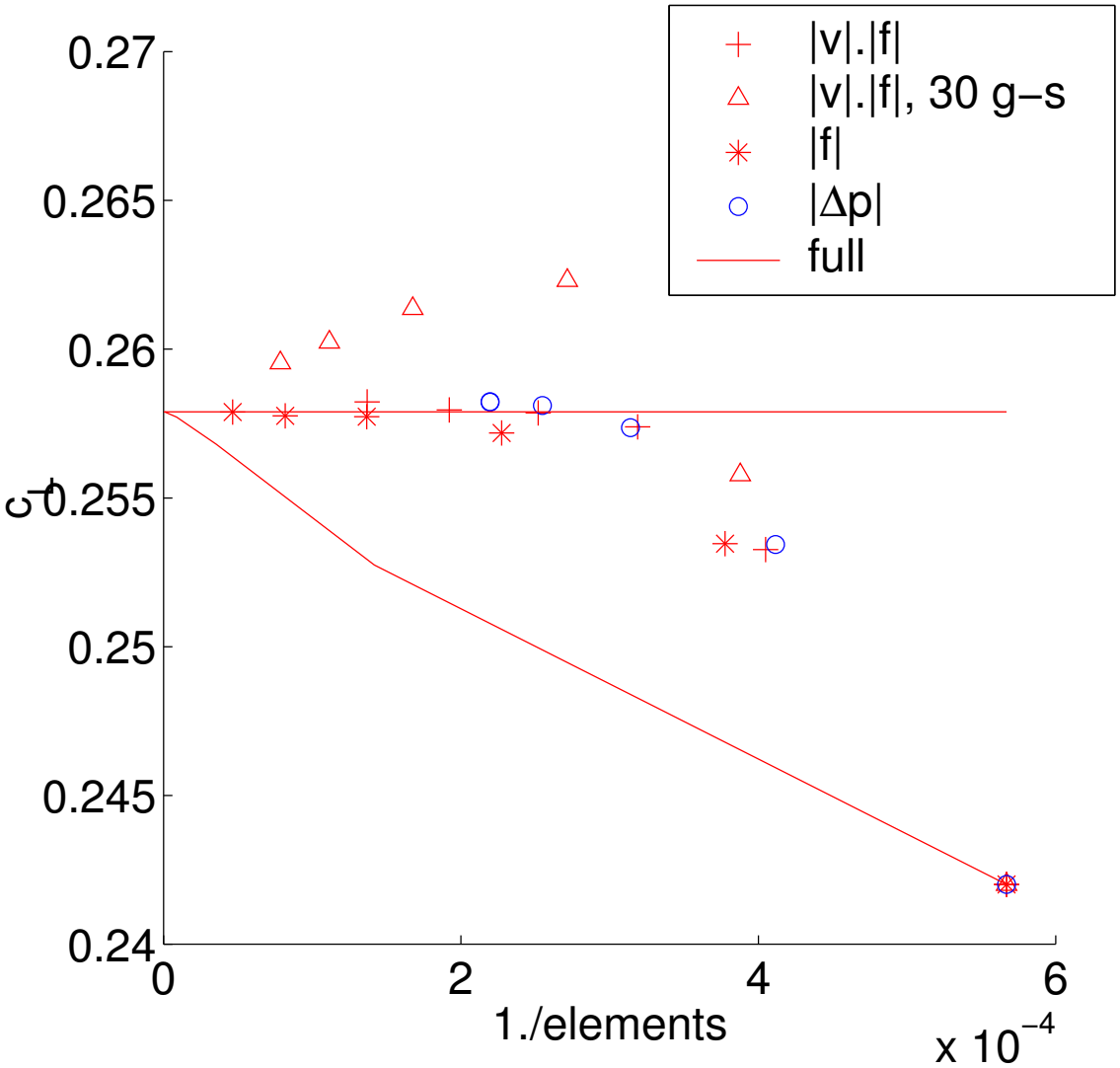
NACA 0012, Ma=0.4, $\alpha = 2°$, 5th level





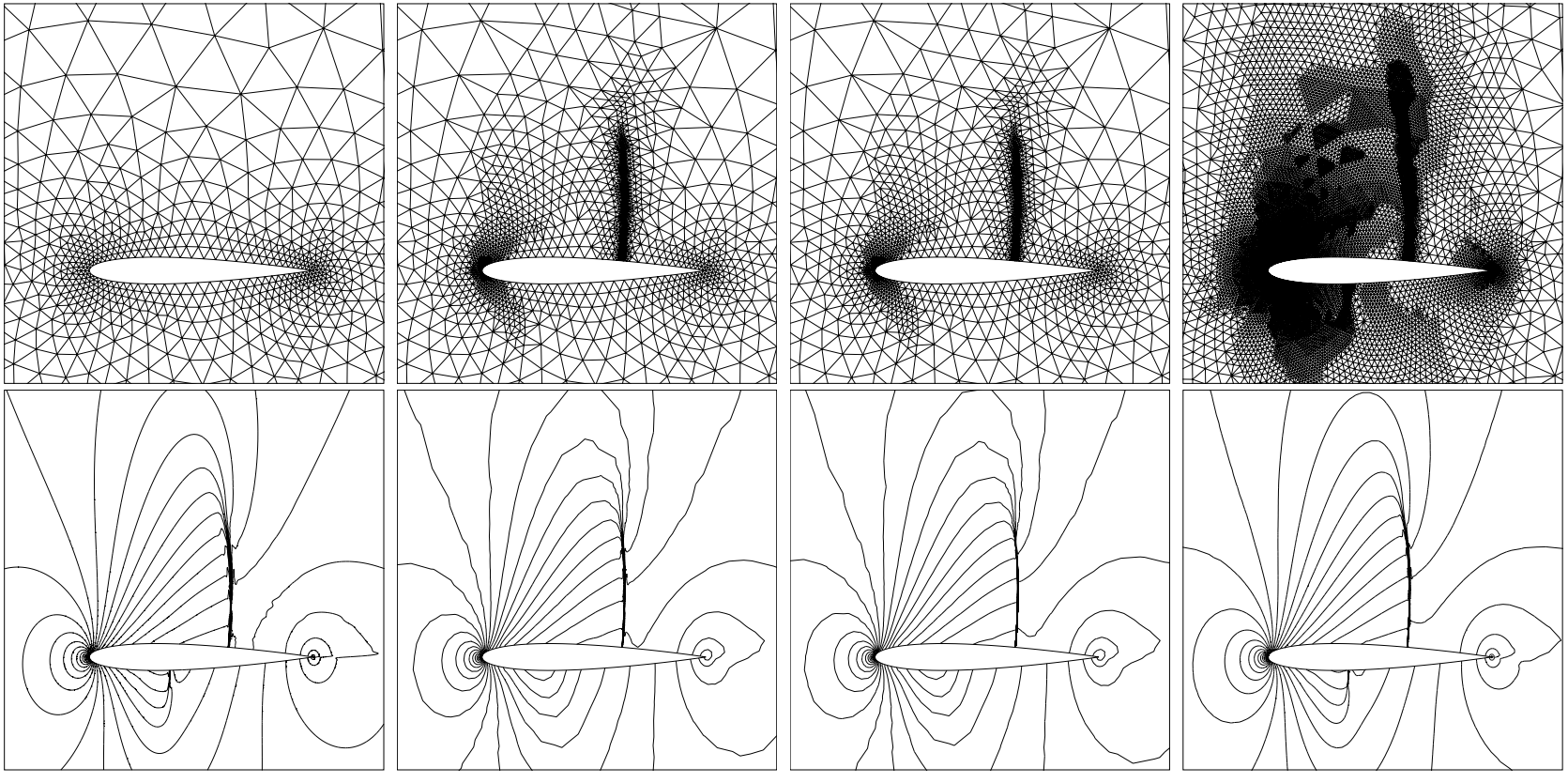
$|\Delta p|$ $|f|$ $|v||f|$ $|v||f|$, 30 g-s

# Subsonic Airfoil: lift vs. $h^2$



# Example: transonic aerofoil in inviscid flow

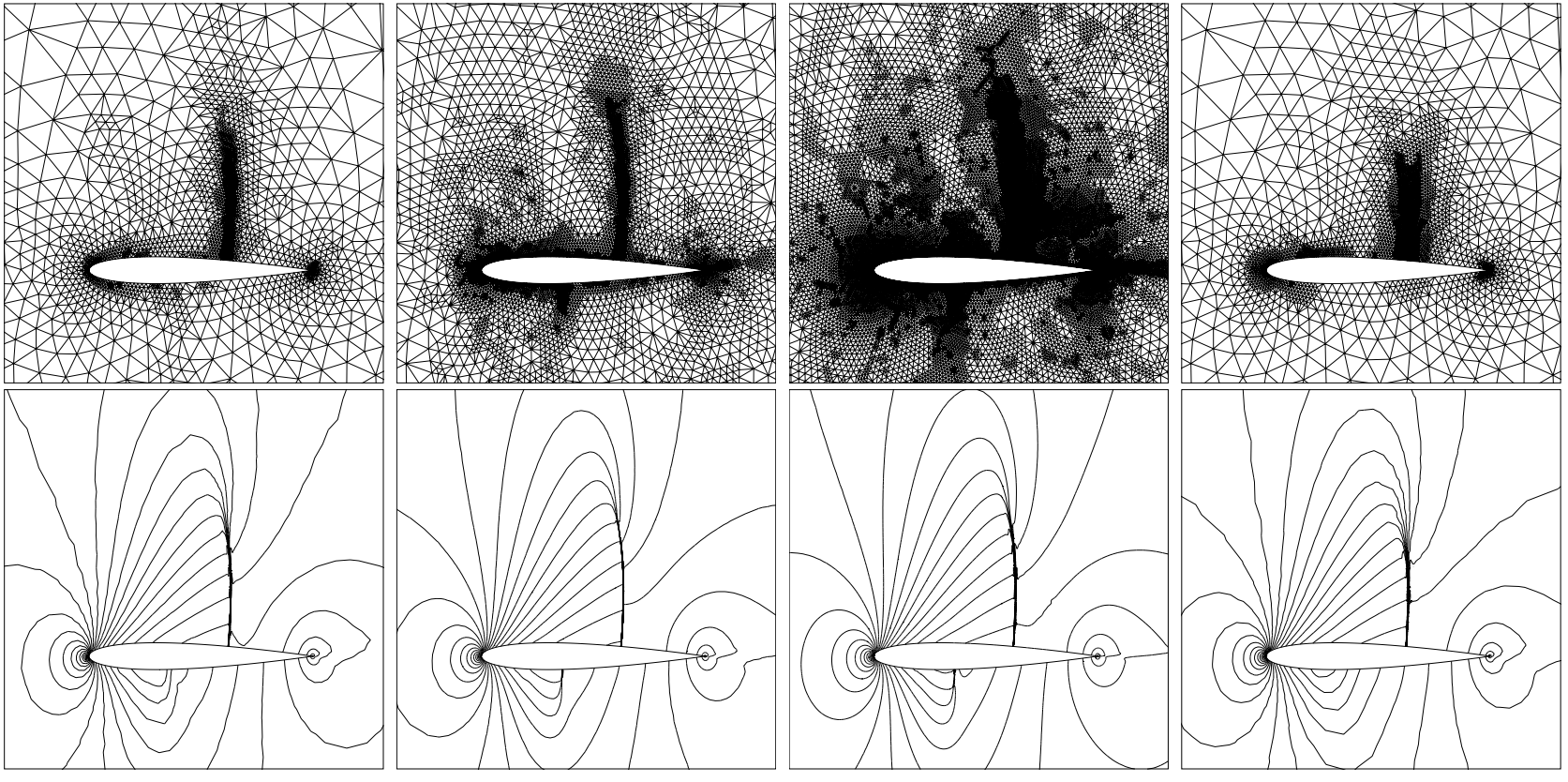
NACA 0012, Ma=0.8, $\alpha = 1.25°$, 5/9 levels, pressure-sensor



ini grid/full ref $|\Delta p|$ $|\Delta p|$, 9 lvls. $|\Delta p|$, 30% ref

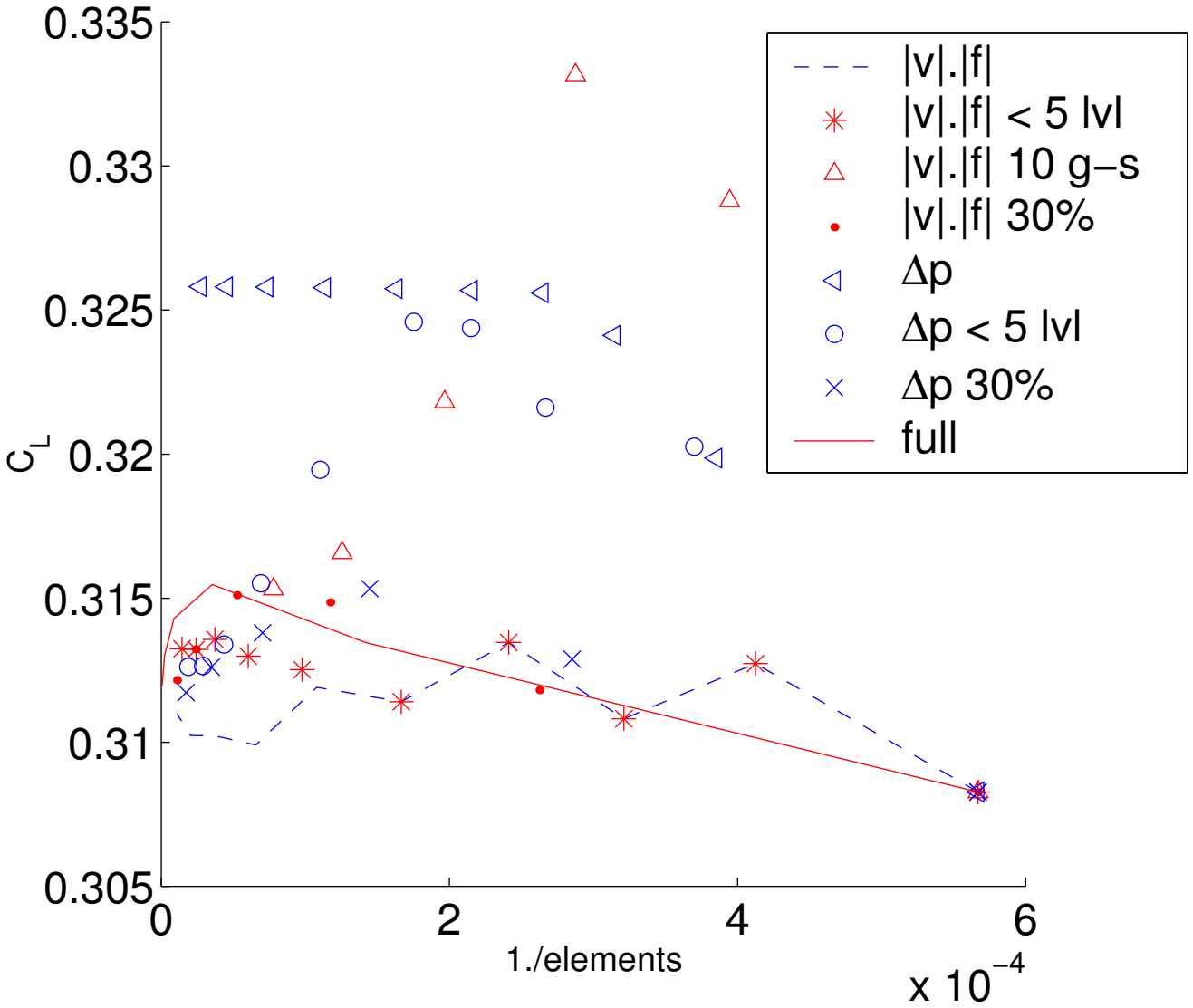
# Example: transonic aerofoil in inviscid flow

NACA 0012, Ma=0.8, $\alpha = 1.25°$, 5/9 levels, residual sensors

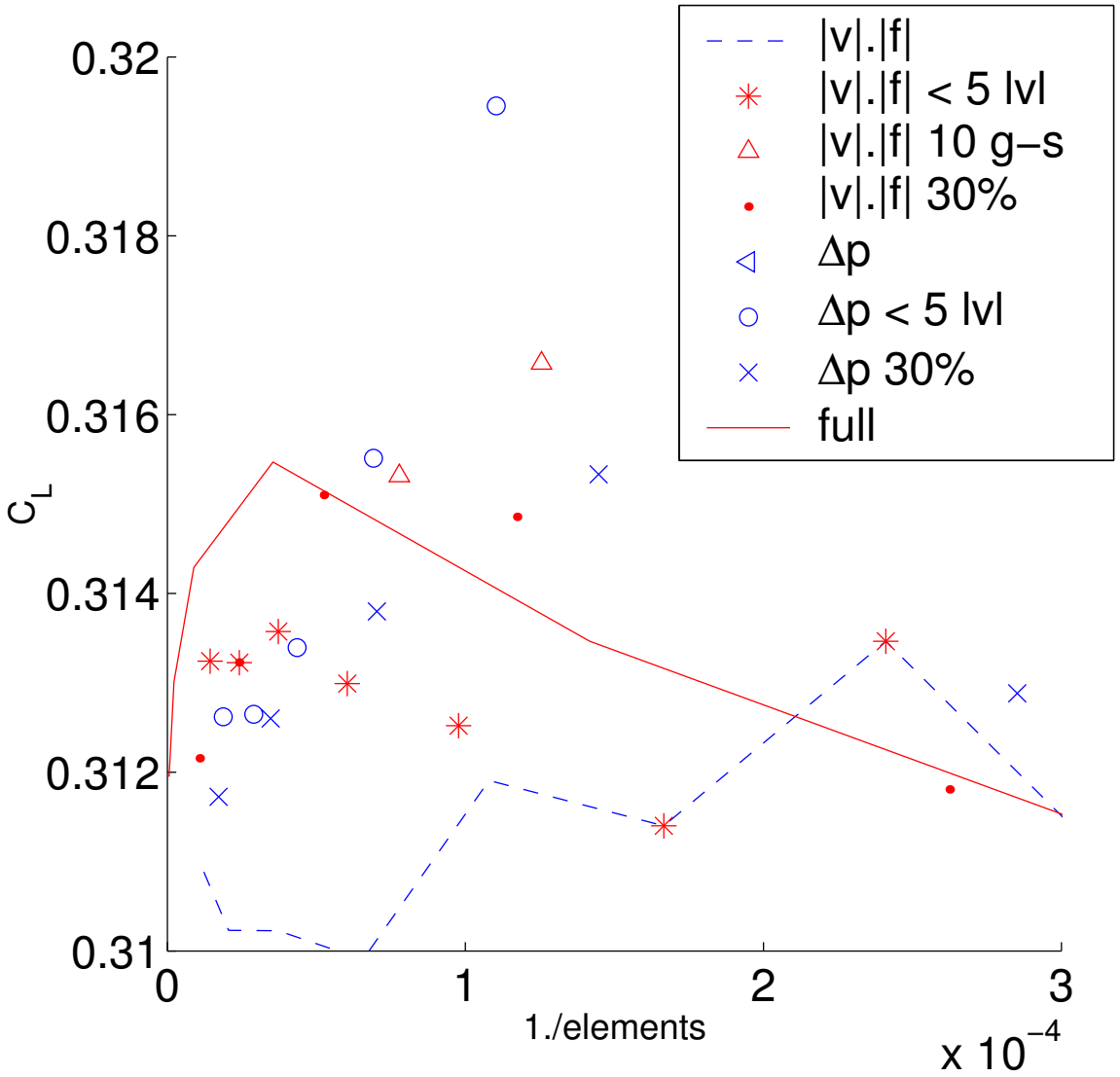


$|v||f|$ $|v||f|$, 9 lvls. $|v||f|$, 30% $|v||f|$, 10 g-s

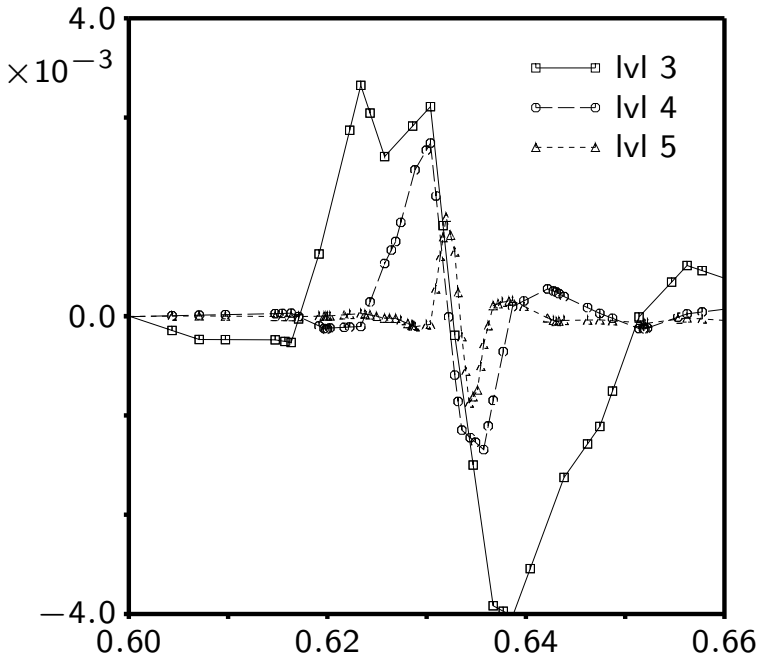
# Transonic Airfoil: lift vs. $h^2$



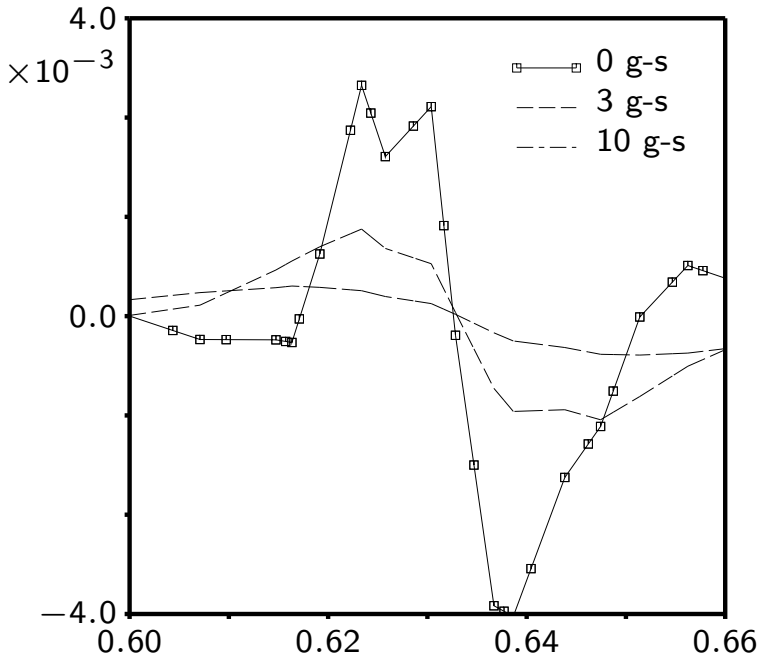
# Transonic Airfoil: lift vs. $h^2$



# Residual in the shock




0 g-s; levels 3, 4, 5




0, 3, 10 g-s; level 3

# Refining for shocks

- $O(1)$ error in flux at shock produces large residuals
- flux conservation means these sum to zero, approximately
- continuity of adjoint variables means their effects on functional also cancel, to leading order
- without limiting, too much adaptation goes into stronger shock
- smoothing the residual helps, but too much loses the adaptation of the weaker shock
- limiting the number of levels does a good job, but is not a very satisfactory solution

# Contents

# Error estimation using local refinement

- Venditti, Darmofal (JCP, 2002): Estimate the error by evaluating the residual of the solution of the current grid interpolated onto a subdivided grid.
- Correct the goal function with the "computable" correction, no solution on the refined mesh needed.
- Adapt the mesh to minimise the error in the computable correction (ECC) due to non-linearity, which can be approximated as the difference between the inner products of
  - adjoint solution-error and primal residual and
  - adjoint residual and the primal solution-error.

# Error estimation using local refinement

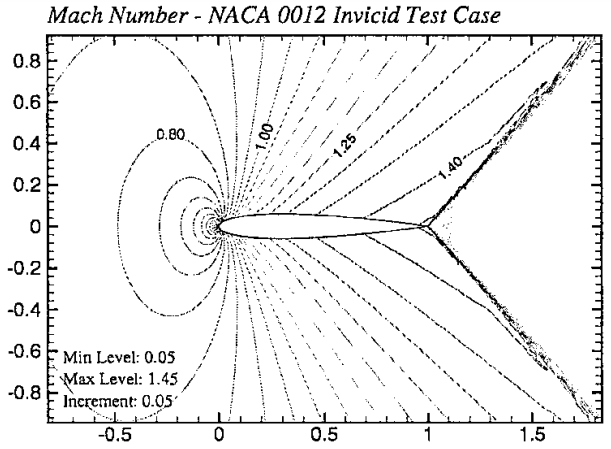
- Venditti, Darmofal (JCP, 2002): Estimate the error by evaluating the residual of the solution of the current grid interpolated onto a subdivided grid.
- Correct the goal function with the "computable" correction, no solution on the refined mesh needed.
- Adapt the mesh to minimise the error in the computable correction (ECC) due to non-linearity, which can be approximated as the difference between the inner products of
  - adjoint solution-error and primal residual and
  - adjoint residual and the primal solution-error.

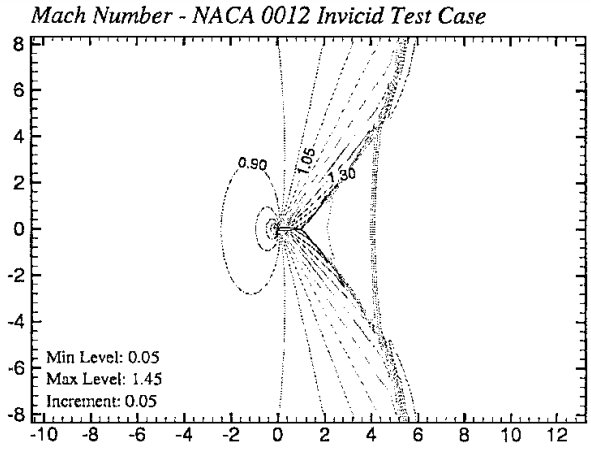
# Error estimation using local refinement

- Venditti, Darmofal (JCP, 2002): Estimate the error by evaluating the residual of the solution of the current grid interpolated onto a subdivided grid.
- Correct the goal function with the "computable" correction, no solution on the refined mesh needed.
- Adapt the mesh to minimise the error in the computable correction (ECC) due to non-linearity, which can be approximated as the difference between the inner products of
  - adjoint solution-error and primal residual and
  - adjoint residual and the primal solution-error.

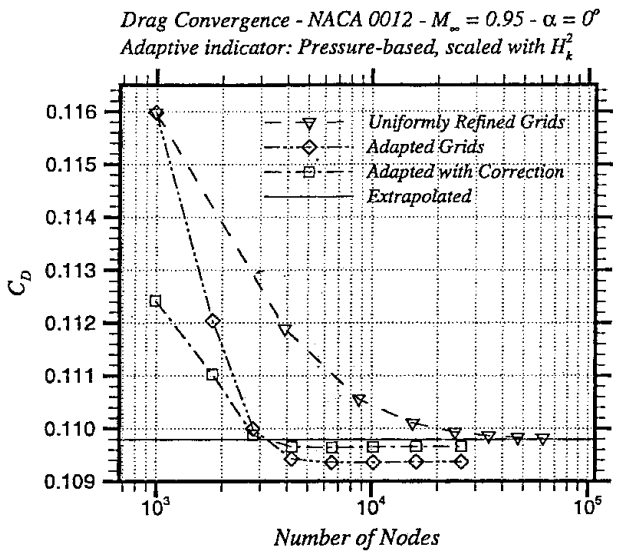
# Application of local refinement sensor

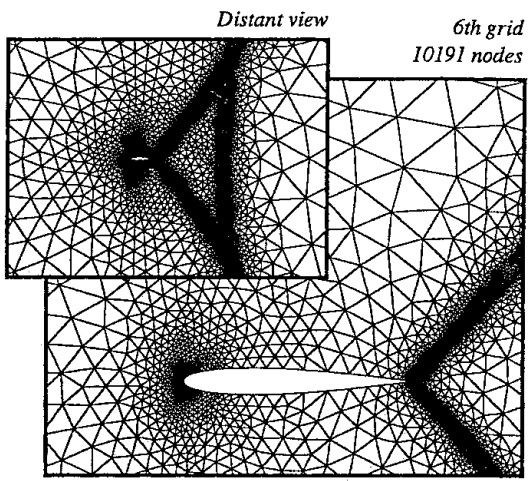
NACA 0012
Ma=0.95




pressure
sensor

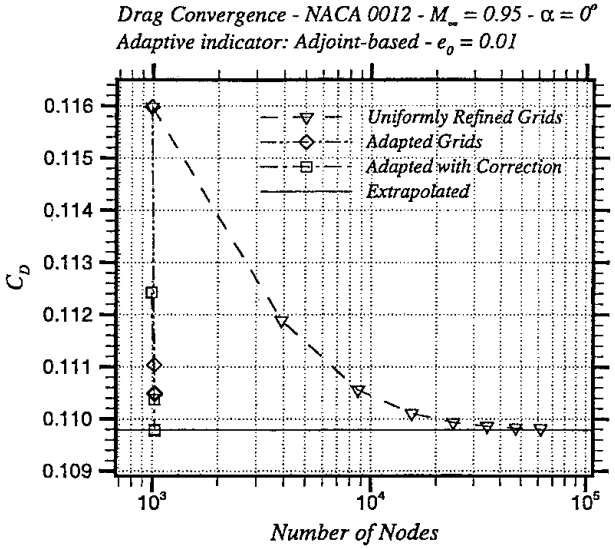


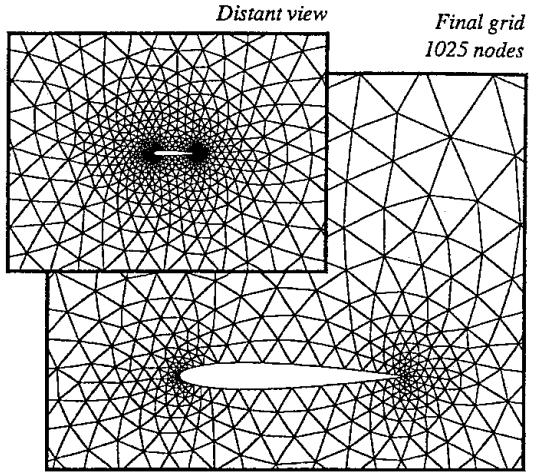




Source: Venditti, Darmofal, JCP 2002

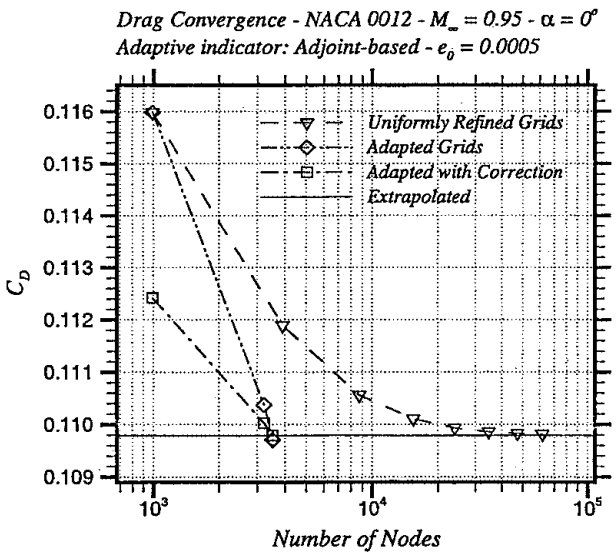
# Application of local refinement sensor

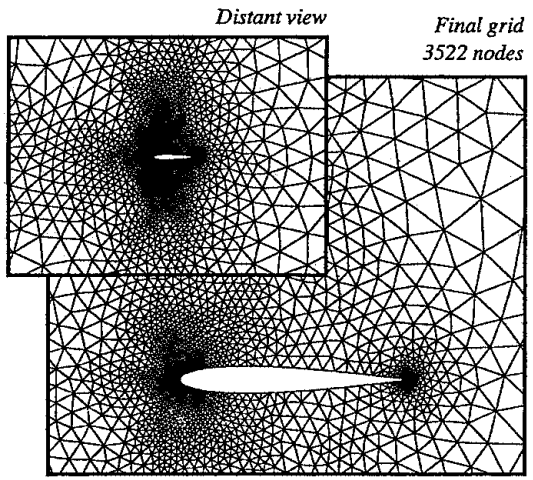
ECC sensor
e=0.01




ECC sensor
e=0.0005




Source: Venditti, Darmofal, JCP 2002

# Conservation-based estimation/Entropy variables

- Principle: compressible FV discretisations conserve the integrals of the continuity and momentum equations, but not entropy $s$
- Observation (Fidkowski, Roe, 2010): the flow solution in entropy variables is identical to the adjoint for entropy production of the Euler eq.
- Viscous effects and shocks will reduce $p_{tot}$ and $s$, but these contributions can be computed and subtracted from the total entropy production to obtain numerical entropy.
- Idea: use the numerical $s$ as a sensor.

# Conservation-based estimation/Entropy variables

- Principle: compressible FV discretisations conserve the integrals of the continuity and momentum equations, but not entropy $s$
- Observation (Fidkowski, Roe, 2010): the flow solution in entropy variables is identical to the adjoint for entropy production of the Euler eq.
- Viscous effects and shocks will reduce $p_{tot}$ and $s$, but these contributions can be computed and subtracted from the total entropy production to obtain numerical entropy.
- Idea: use the numerical $s$ as a sensor.

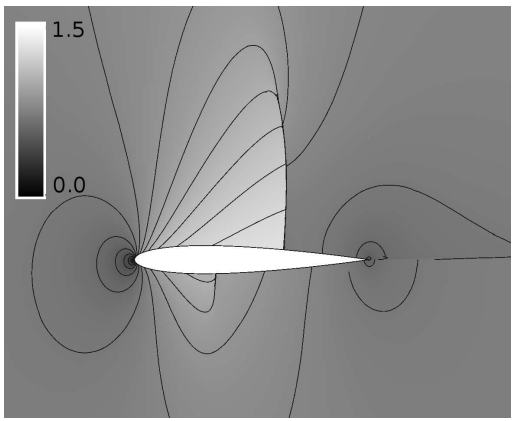
# Conservation-based estimation/Entropy variables

- Principle: compressible FV discretisations conserve the integrals of the continuity and momentum equations, but not entropy $s$
- Observation (Fidkowski, Roe, 2010): the flow solution in entropy variables is identical to the adjoint for entropy production of the Euler eq.
- Viscous effects and shocks will reduce $p_{tot}$ and $s$, but these contributions can be computed and subtracted from the total entropy production to obtain numerical entropy.
- Idea: use the numerical $s$ as a sensor.

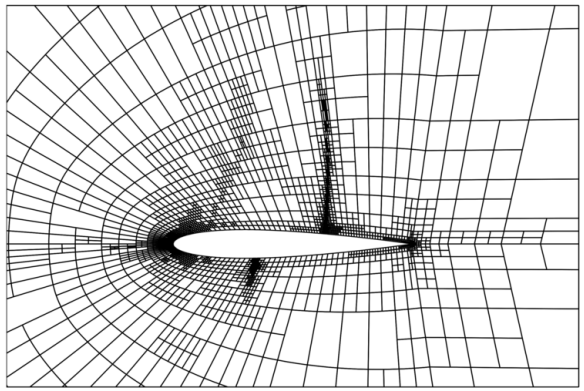
# Conservation-based estimation/Entropy variables

- Principle: compressible FV discretisations conserve the integrals of the continuity and momentum equations, but not entropy $s$
- Observation (Fidkowski, Roe, 2010): the flow solution in entropy variables is identical to the adjoint for entropy production of the Euler eq.
- Viscous effects and shocks will reduce $p_{tot}$ and $s$, but these contributions can be computed and subtracted from the total entropy production to obtain numerical entropy.
- Idea: use the numerical $s$ as a sensor.

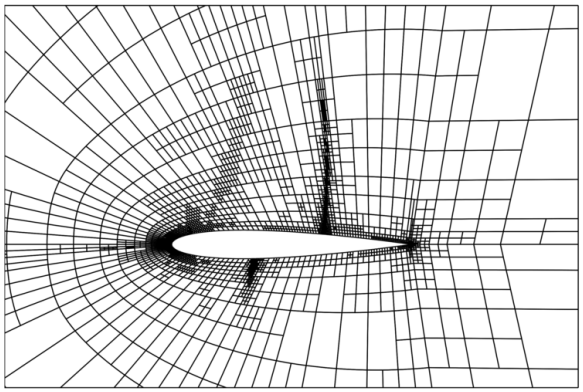
# Adaptation with entropy variables



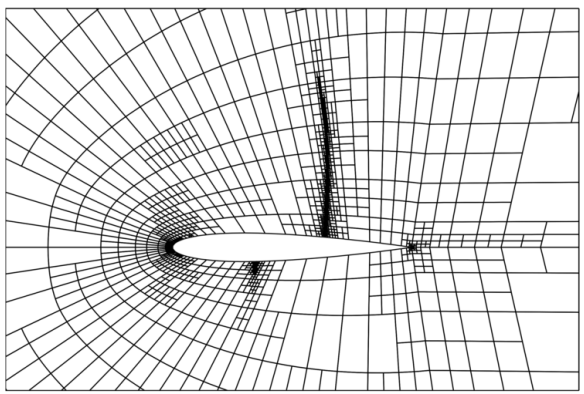

NACA 0012,
Ma=0.8, $\alpha = 1.25°$



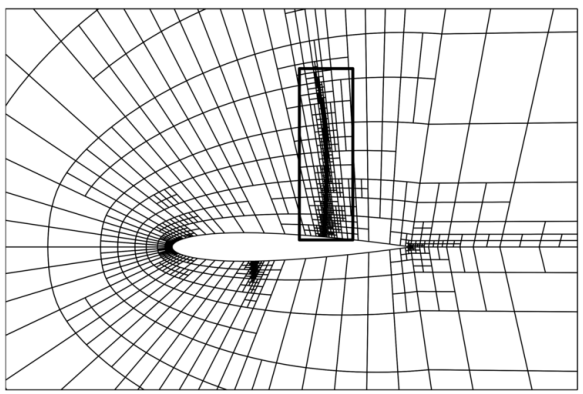
drag adjoint



lift adjoint



residual

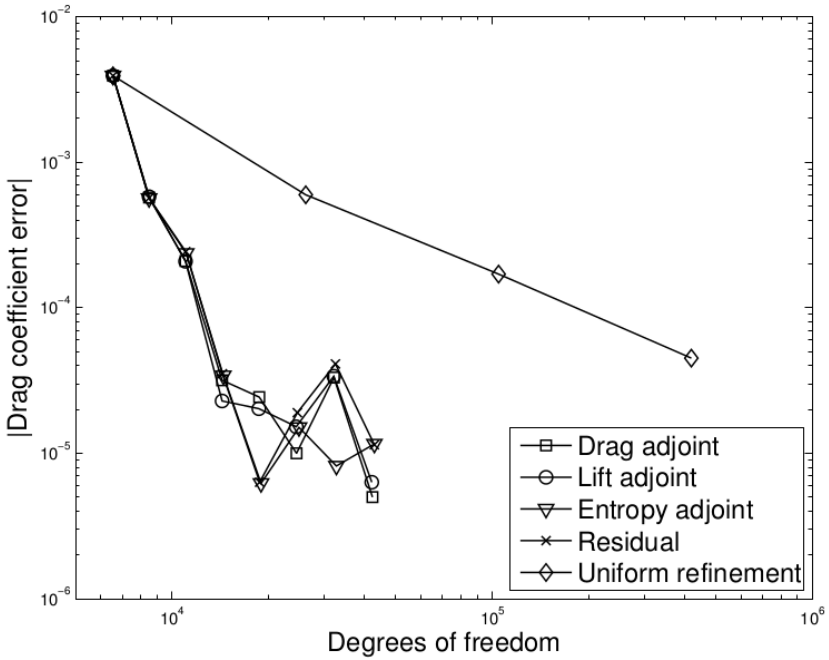


entropy

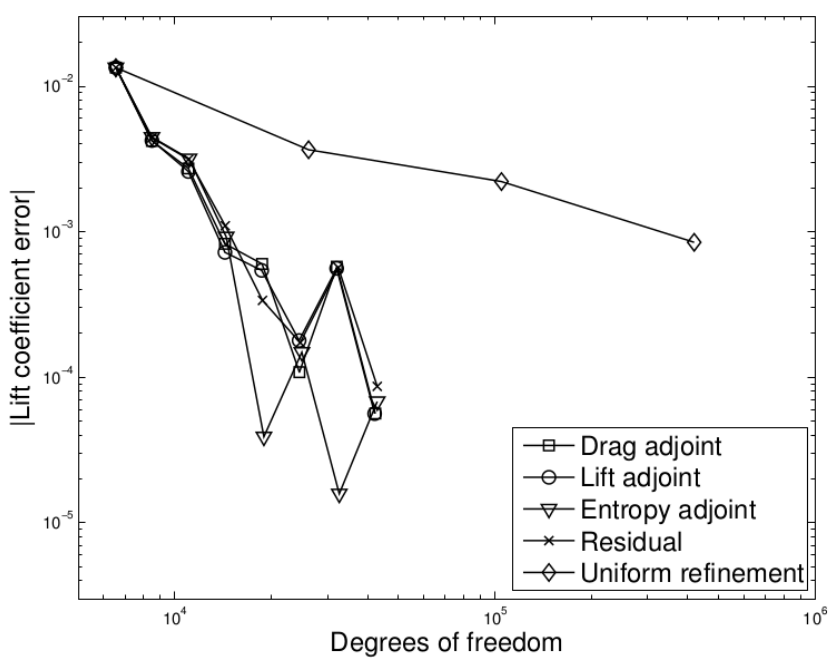
Source: Fidkowski, Roe, SIAM J Comp, 2010

# Adaptation with entropy variables

Convergence of lift and drag errors with various adaptation sensors:




drag output




lift output

Source: Fidkowski, Roe, SIAM J Comp, 2010

# Sensors based on Artificial Viscosity

- Dwight et al.: refinement is required where A.V. is high, de-refinement is possible where the solution is captured (nearly) exactly and A.V. is low.
- They use a JST scheme with scalar dissipation based on first and third-order pressure differences, multiplied with coefficients $\epsilon^{(2)}, \epsilon^{(4)}$.
- Use adjoint weighting to evaluate the sensitivity of the cost function w.r.t the value of the coefficients $\epsilon$.

# Sensors based on Artificial Viscosity

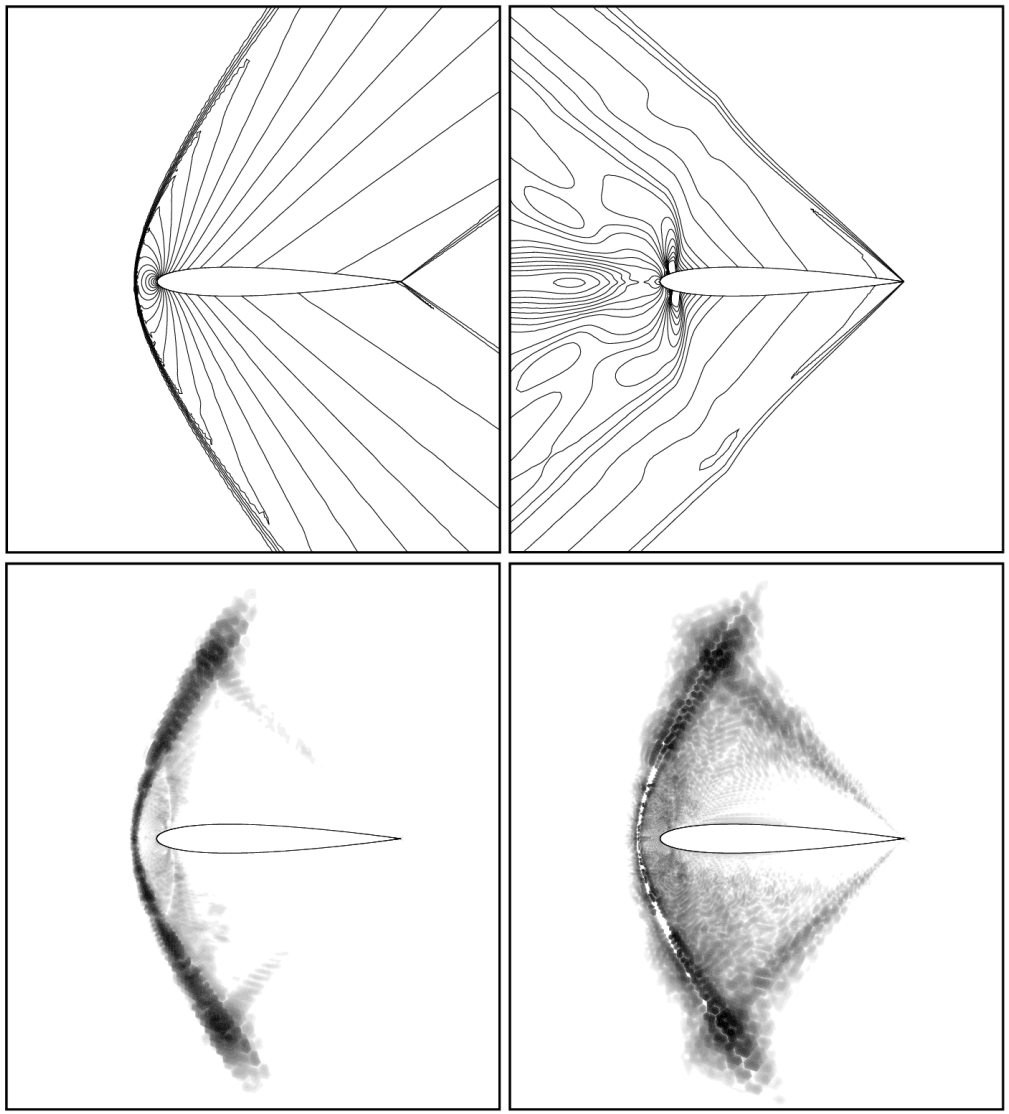
- Dwight et al.: refinement is required where A.V. is high, de-refinement is possible where the solution is captured (nearly) exactly and A.V. is low.
- They use a JST scheme with scalar dissipation based on first and third-order pressure differences, multiplied with coefficients $\epsilon^{(2)}, \epsilon^{(4)}$.
- Use adjoint weighting to evaluate the sensitivity of the cost function w.r.t the value of the coefficients $\epsilon$.

# Sensors based on Artificial Viscosity

- Dwight et al.: refinement is required where A.V. is high, de-refinement is possible where the solution is captured (nearly) exactly and A.V. is low.
- They use a JST scheme with scalar dissipation based on first and third-order pressure differences, multiplied with coefficients $\epsilon^{(2)}, \epsilon^{(4)}$.
- Use adjoint weighting to evaluate the sensitivity of the cost function w.r.t the value of the coefficients $\epsilon$.

# Application of sensor based on artificial viscosity

primal



adjoint

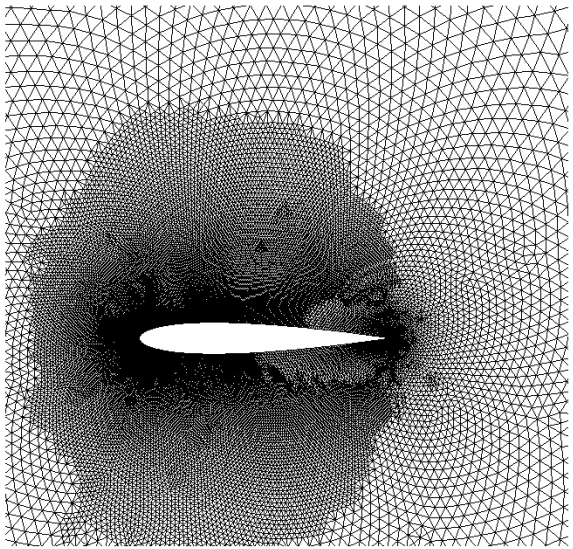
sens. to $\epsilon^{(2)}$

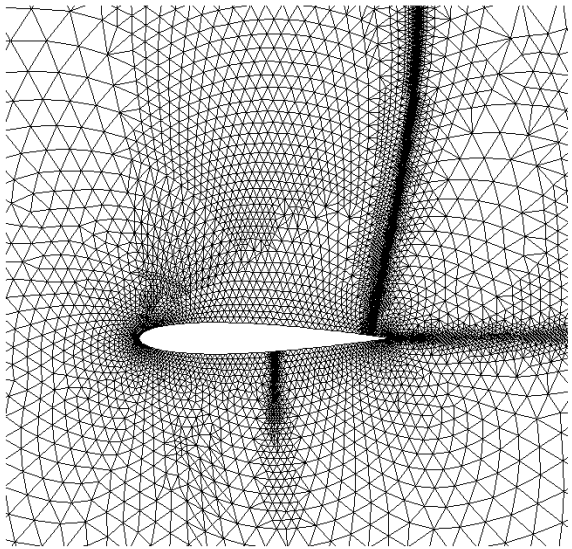
sens. to $\epsilon^{(4)}$

Source: Dwight, JCP, 2008

# Application of sensor based on artificial viscosity

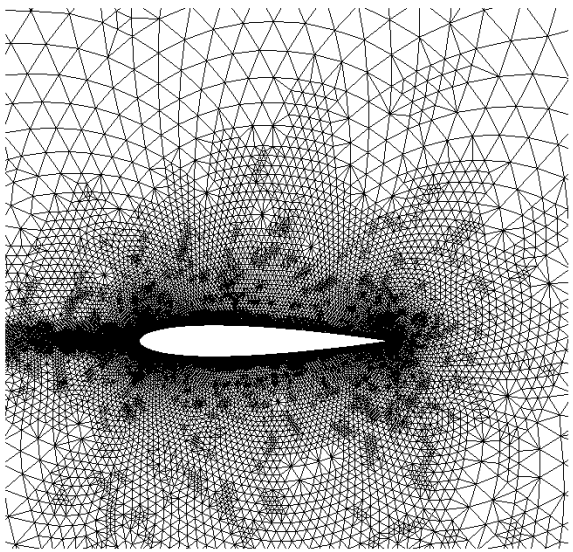
subsonic grad. sens.

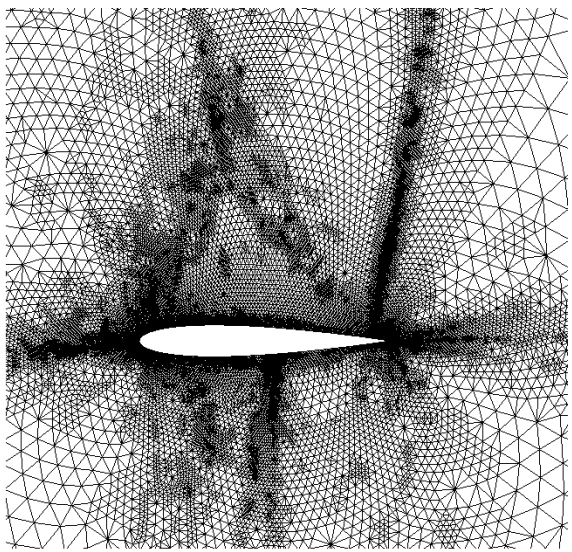




transonic grad. sens.

subsonic dissip. sens.



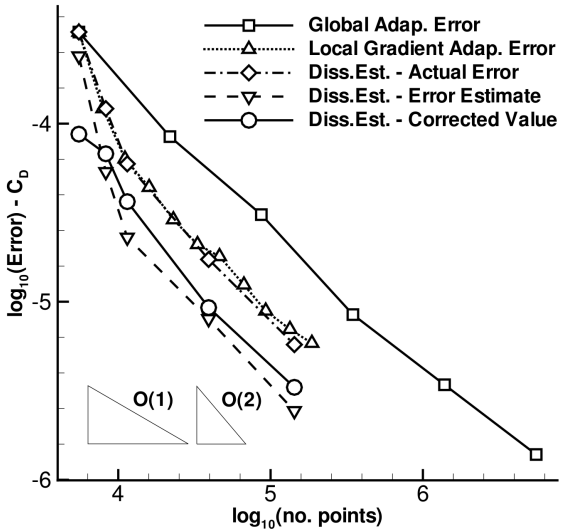


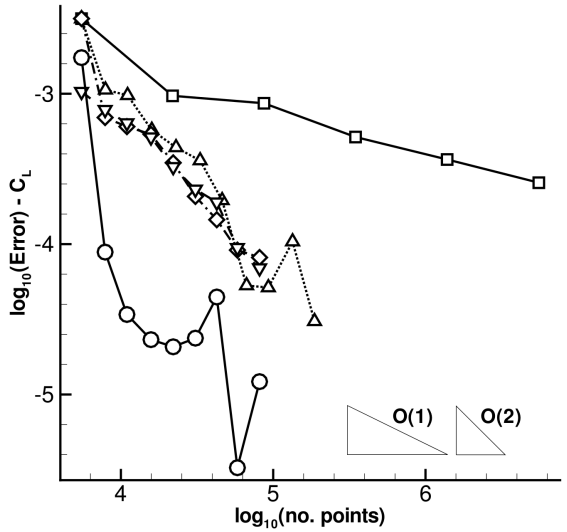
transonic dissip. sens.

Source: Dwight, JCP, 2008

# Adaptation with entropy variables

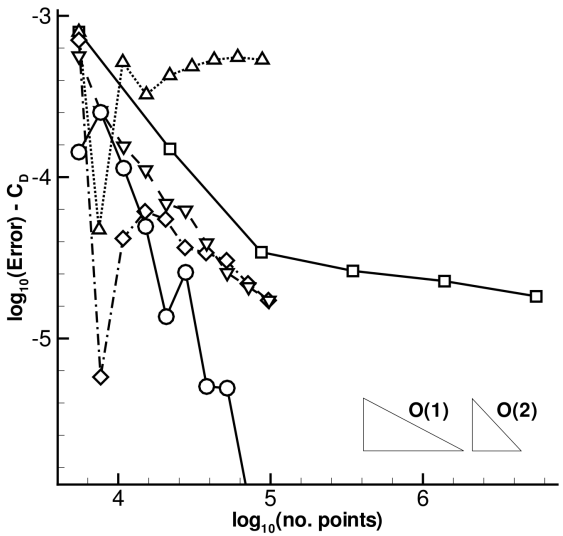
subs. drag

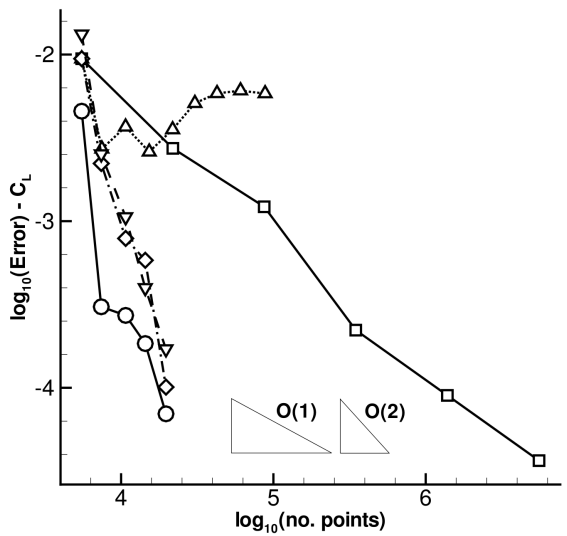



subs. lift

trans. drag




trans. lift

Source: Dwight, JCP, 2008

# Contents

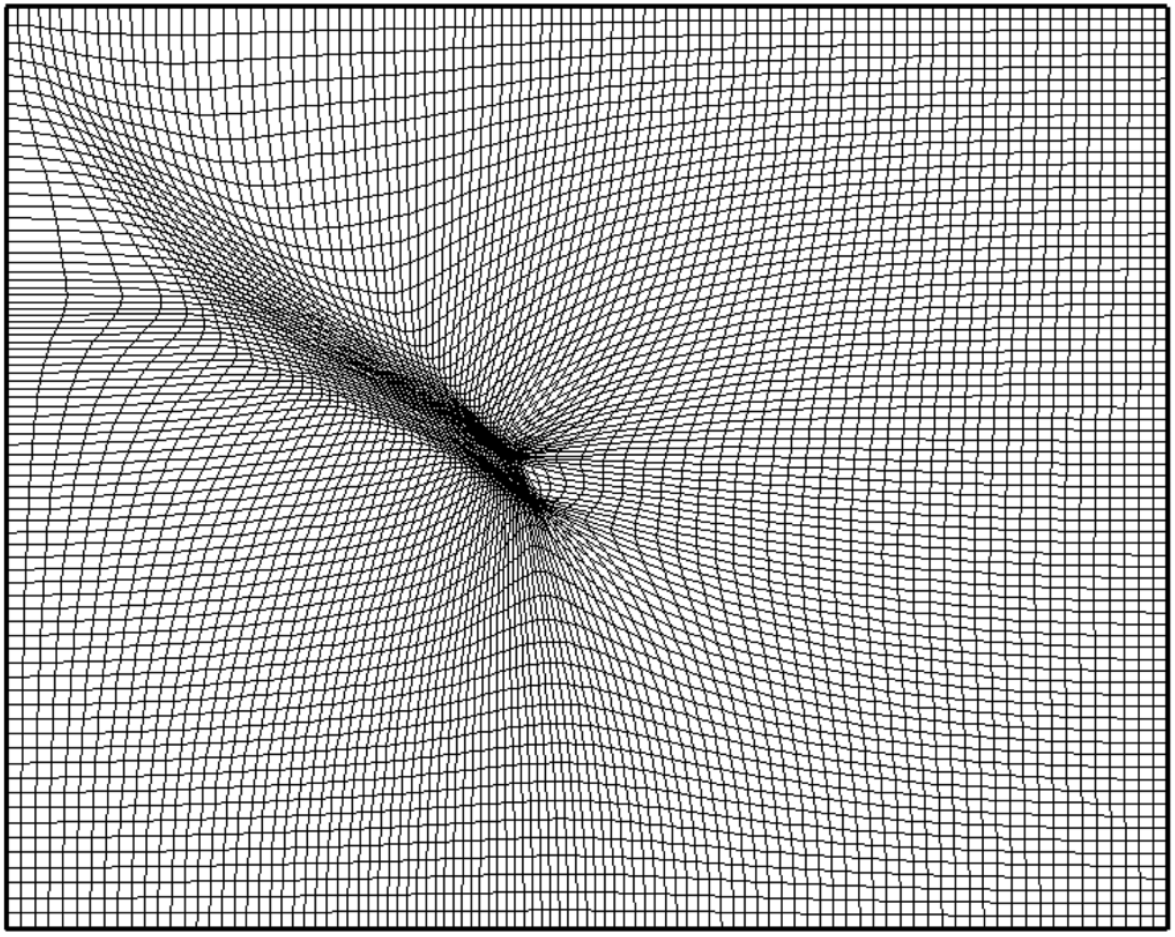
# Mesh adaptation methods

- Remeshing
- *r*-refinement
- *p*-refinement
- *h*-refinement

# r-refinement

- Equi-distribute the error in the mesh at constant cost:
  - Concentrate mesh into zones of interest/high error,
  - Coarsen mesh in zones of low error.
- If Topology is kept invariant: no change in data-structures. Low overheads for unsteady cases. Often already implemented in optimisation chains to respond to design changes.
- Mesh quality will degrade with aggressive mesh movent: can be combined with mesh repair, e.g. edge- or face-swapping for tetrahedra (MMG3D tool).

# r-refinement



Example of *r*-refinement to capture a pollutant plume.
Source: Garcia-Menendez et al., Atmosphere, 2011

# p-refinement

- Increase the order of accuracy of the scheme in regions of smooth variation.
- Useful for finite-element and discontinuous Galerkin methods. See later presentations by Prof. Fidkowski.
- Of limited use in F.V. methods since most practical FV methods are limited to second-order accuracy.
- Needs to be combined with *h* or *r*-refinement to capture discontinuous aspects of the solution.

# p-refinement

- Increase the order of accuracy of the scheme in regions of smooth variation.
- Useful for finite-element and discontinuous Galerkin methods. See later presentations by Prof. Fidkowski.
- Of limited use in F.V. methods since most practical FV methods are limited to second-order accuracy.
- Needs to be combined with $h$ or $r$-refinement to capture discontinuous aspects of the solution.

# p-refinement

- Increase the order of accuracy of the scheme in regions of smooth variation.
- Useful for finite-element and discontinuous Galerkin methods. See later presentations by Prof. Fidkowski.
- Of limited use in F.V. methods since most practical FV methods are limited to second-order accuracy.
- Needs to be combined with $h$ or $r$-refinement to capture discontinuous aspects of the solution.

# p-refinement

- Increase the order of accuracy of the scheme in regions of smooth variation.
- Useful for finite-element and discontinuous Galerkin methods. See later presentations by Prof. Fidkowski.
- Of limited use in F.V. methods since most practical FV methods are limited to second-order accuracy.
- Needs to be combined with *h* or *r*-refinement to capture discontinuous aspects of the solution.

# h-refinement

- global remeshing Majewski, Dobrzynski, Alauzet
- local remeshing
  - Delaunay triangulation
  - Mesh repair, edge/face swapping: Dobrzynski MMG3D, works well for simplex meshes (tets in 3D)
  - Optimisation-based smoothing Freitag, Gooch (Gamma, Mesquite)
  - Prism layers need special treatment due to element type and high aspect ratio.
- isotropic hierarchic refinement: 'AMR' using quad-trees on Cartesian meshes is relatively straightforward, can be effective for blast-type problems to track fronts. Difficulties with anisotropic layers.

# h-refinement

- global remeshing Majewski, Dobrzynski, Alauzet
- local remeshing
  - Delaunay triangulation
  - Mesh repair, edge/face swapping. Dobrzynski MMG3D, works well for simplex meshes (tets in 3D)
  - Optimisation-based smoothing Freitag, Gooch (Grumpp, Mesquite)
  - Prism layers need special treatment due to element type and high aspect ratio.
- isotropic hierarchic refinement: 'AMR' using quad-trees on Cartesian meshes is relatively straightforward, can be effective for blast-type problems to track fronts. Difficulties with anisotropic layers.

# h-refinement

- global remeshing Majewski, Dobrzynski, Alauzet
- local remeshing
  - Delaunay triangulation
  - Mesh repair, edge/face swapping. Dobrzynski MMG3D, works well for simplex meshes (tets in 3D)
  - Optimisation-based smoothing Freitag, Gooch (Grumpp, Mesquite)
  - Prism layers need special treatment due to element type and high aspect ratio.
- isotropic hierarchic refinement: 'AMR' using quad-trees on Cartesian meshes is relatively straightforward, can be effective for blast-type problems to track fronts. Difficulties with anisotropic layers.

# h-refinement

- global remeshing Majewski, Dobrzynski, Alauzet
- local remeshing
  - Delaunay triangulation
  - Mesh repair, edge/face swapping. Dobrzynski MMG3D, works well for simplex meshes (tets in 3D)
  - Optimisation-based smoothing Freitag, Gooch (Grumpp, Mesquite)
  - Prism layers need special treatment due to element type and high aspect ratio.
- isotropic hierarchic refinement: 'AMR' using quad-trees on Cartesian meshes is relatively straightforward, can be effective for blast-type problems to track fronts. Difficulties with anisotropic layers.

# h-refinement

- global remeshing Majewski, Dobrzynski, Alauzet
- local remeshing
  - Delaunay triangulation
  - Mesh repair, edge/face swapping. Dobrzynski MMG3D, works well for simplex meshes (tets in 3D)
  - Optimisation-based smoothing Freitag, Gooch (Grumpp, Mesquite)
  - Prism layers need special treatment due to element type and high aspect ratio.
- isotropic hierarchic refinement: 'AMR' using quad-trees on Cartesian meshes is relatively straightforward, can be effective for blast-type problems to track fronts. Difficulties with anisotropic layers.

# h-refinement

- global remeshing Majewski, Dobrzynski, Alauzet
- local remeshing
  - Delaunay triangulation
  - Mesh repair, edge/face swapping. Dobrzynski MMG3D, works well for simplex meshes (tets in 3D)
  - Optimisation-based smoothing Freitag, Gooch (Grumpp, Mesquite)
  - Prism layers need special treatment due to element type and high aspect ratio.
- isotropic hierarchic refinement: 'AMR' using quad-trees on Cartesian meshes is relatively straightforward, can be effective for blast-type problems to track fronts. Difficulties with anisotropic layers.

# h-refinement

- global remeshing Majewski, Dobrzynski, Alauzet
- local remeshing
  - Delaunay triangulation
  - Mesh repair, edge/face swapping. Dobrzynski MMG3D, works well for simplex meshes (tets in 3D)
  - Optimisation-based smoothing Freitag, Gooch (Grumpp, Mesquite)
  - Prism layers need special treatment due to element type and high aspect ratio.
- isotropic hierarchic refinement: 'AMR' using quad-trees on Cartesian meshes is relatively straightforward, can be effective for blast-type problems to track fronts. Difficulties with anisotropic layers.

# Anisotropic hierarchic grid adaptation

- Effective hierarchic grid adaptation methods need to
  - be able to operate on hybrid grids (tet, prism, pyramid, hex)
  - allow anisotropic refinement,
  - need to avoid propagation of mesh lines through the entire mesh,
  - preserve mesh quality,
  - have local operations (geometric neighbourhood only) to support parallel implementation.

# Anisotropic hierarchic grid adaptation: hip toolkit

Anisotropic hierarchic grid adaptation (Mueller, IJNMFD, 2002):

- Nodes are added on edges, faces or cells
  - Searchable data-structure for **adaptive edges** of cells which allows to communicate refinement of an edge to all neighbouring elements.
  - Nodes on faces are communicated with 'diagonal' adaptive edges
  - Cell-centre nodes are known when an element is refined.
- Define a limited set of canonical refinement patterns (and their rotations) that preserve element quality.
- Limit refinement differences between cells to at most 1 level
- Treat hanging nodes in a post-processing step, if needed.

# Anisotropic hierarchic grid adaptation: hip toolkit

Anisotropic hierarchic grid adaptation (Mueller, IJNMFD, 2002):

- Nodes are added on edges, faces or cells
  - Searchable data-structure for **adaptive edges** of cells which allows to communicate refinement of an edge to all neighbouring elements.
  - Nodes on faces are communicated with 'diagonal' adaptive edges
  - Cell-centre nodes are known when an element is refined.
- Define a limited set of canonical refinement patterns (and their rotations) that preserve element quality.
- Limit refinement differences between cells to at most 1 level
- Treat hanging nodes in a post-processing step, if needed.

# Anisotropic hierarchic grid adaptation: hip toolkit

Anisotropic hierarchic grid adaptation (Mueller, IJNMFD, 2002):

- Nodes are added on edges, faces or cells
  - Searchable data-structure for **adaptive edges** of cells which allows to communicate refinement of an edge to all neighbouring elements.
  - Nodes on faces are communicated with 'diagonal' adaptive edges
  - Cell-centre nodes are known when an element is refined.
- Define a limited set of canonical refinement patterns (and their rotations) that preserve element quality.
- Limit refinement differences between cells to at most 1 level
- Treat hanging nodes in a post-processing step, if needed.

# Anisotropic hierarchic grid adaptation: hip toolkit

Anisotropic hierarchic grid adaptation (Mueller, IJNMFD, 2002):

- Nodes are added on edges, faces or cells
  - Searchable data-structure for **adaptive edges** of cells which allows to communicate refinement of an edge to all neighbouring elements.
  - Nodes on faces are communicated with 'diagonal' adaptive edges
  - Cell-centre nodes are known when an element is refined.
- Define a limited set of canonical refinement patterns (and their rotations) that preserve element quality.
- Limit refinement differences between cells to at most 1 level
- Treat hanging nodes in a post-processing step, if needed.

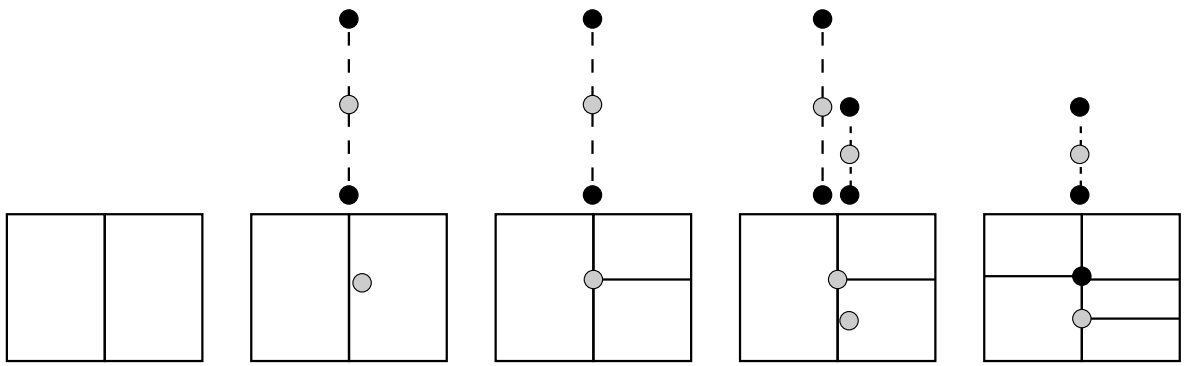
# Anisotropic hierarchic grid adaptation: hip toolkit

Anisotropic hierarchic grid adaptation (Mueller, IJNMFD, 2002):

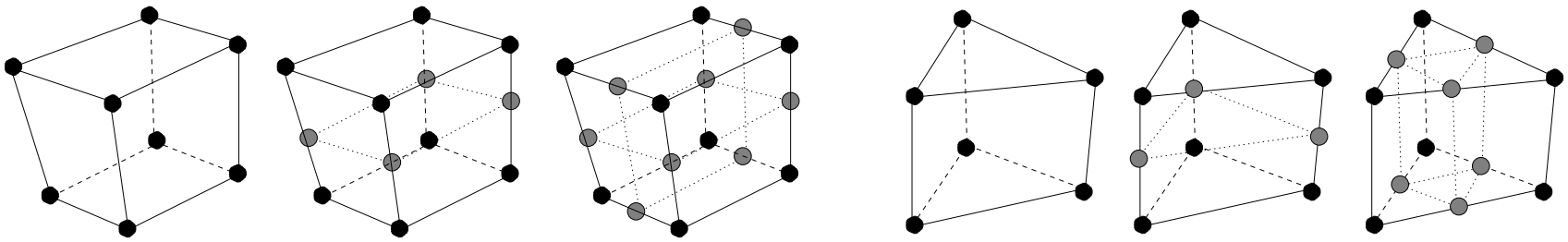
- Nodes are added on edges, faces or cells
  - Searchable data-structure for **adaptive edges** of cells which allows to communicate refinement of an edge to all neighbouring elements.
  - Nodes on faces are communicated with 'diagonal' adaptive edges
  - Cell-centre nodes are known when an element is refined.
- Define a limited set of canonical refinement patterns (and their rotations) that preserve element quality.
- Limit refinement differences between cells to at most 1 level
- Treat hanging nodes in a post-processing step, if needed.

# Adaptive edges

- Carry a mid-node
- Searchable by end- and mid-nodes
- are only removed if all surrounding elements are refined
- otherwise indicate a hanging node

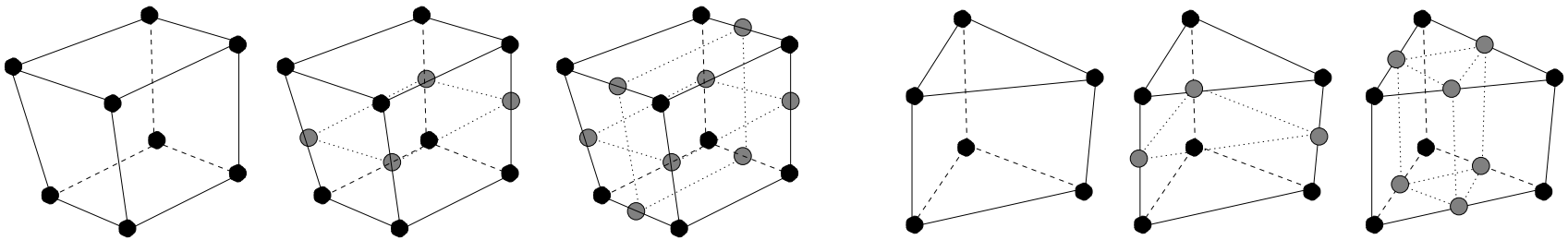


# Examples of anisotropic refinement patterns



- One instance of the pattern is pre-defined
- Secondary information such as child to edge or child to face information is computed from the child to node definition.
- All rotations are pre-computed and stored.

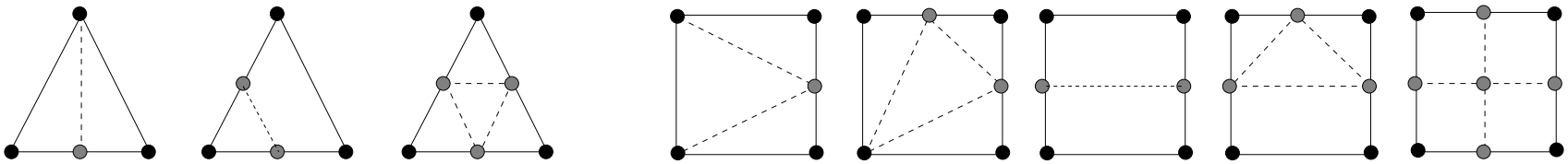
# Examples of anisotropic refinement patterns



- One instance of the pattern is pre-defined
- Secondary information such as child to edge or child to face information is computed from the child to node definition.
- All rotations are pre-computed and stored.

# Conservation through unique face tessellation

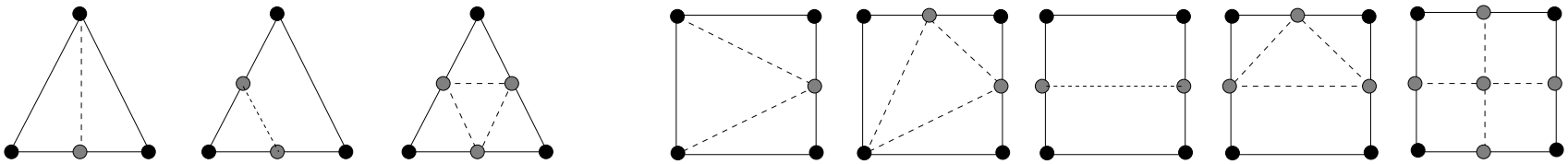
To compute a consistent conservative flux, the tessellation of the cell faces at the refinement interface (level difference) needs to be the same viewed from either side:



- In 3D tri and quad faces need to be considered
- Faces are rotated into a canonical position,
- Rules are defined for all possible canonical cases.

# Conservation through unique face tessellation

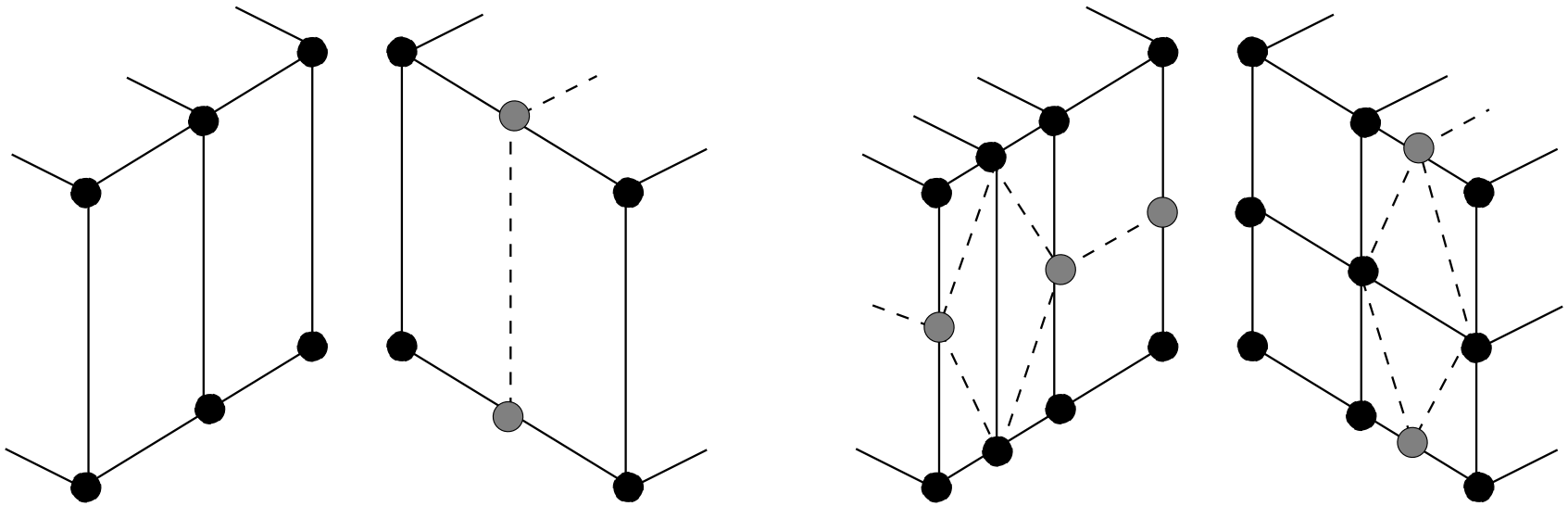
To compute a consistent conservative flux, the tessellation of the cell faces at the refinement interface (level difference) needs to be the same viewed from either side:



- In 3D tri and quad faces need to be considered
- Faces are rotated into a canonical position,
- Rules are defined for all possible canonical cases.

# Anisotropic cross-refinement

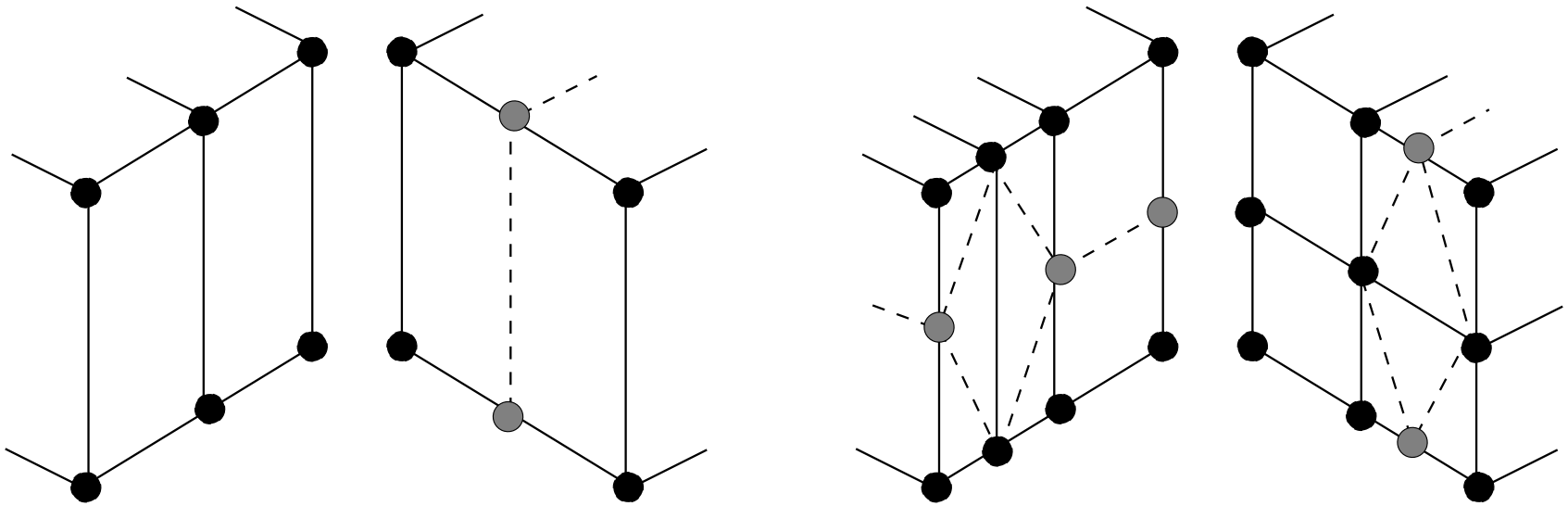
- Anisotropic refinement may switch direction at an interface,
- forcing isotropic refinement in that case is not an option, as it would result in propagation of the costly isotropic refinement through the mesh



- resolved by insertion of a 'hanging node' on both neighbour faces.

# Anisotropic cross-refinement

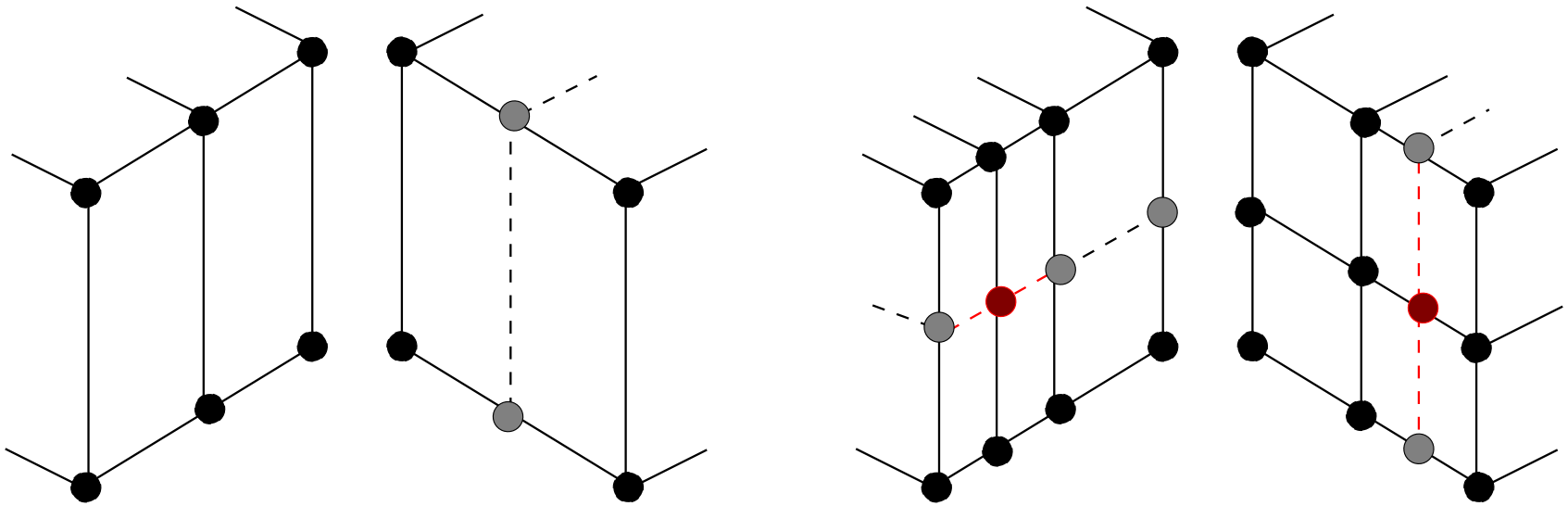
- Anisotropic refinement may switch direction at an interface,
- forcing isotropic refinement in that case is not an option, as it would result in propagation of the costly isotropic refinement through the mesh



- resolved by insertion of a 'hanging node' on both neighbour faces.

# Anisotropic cross-refinement

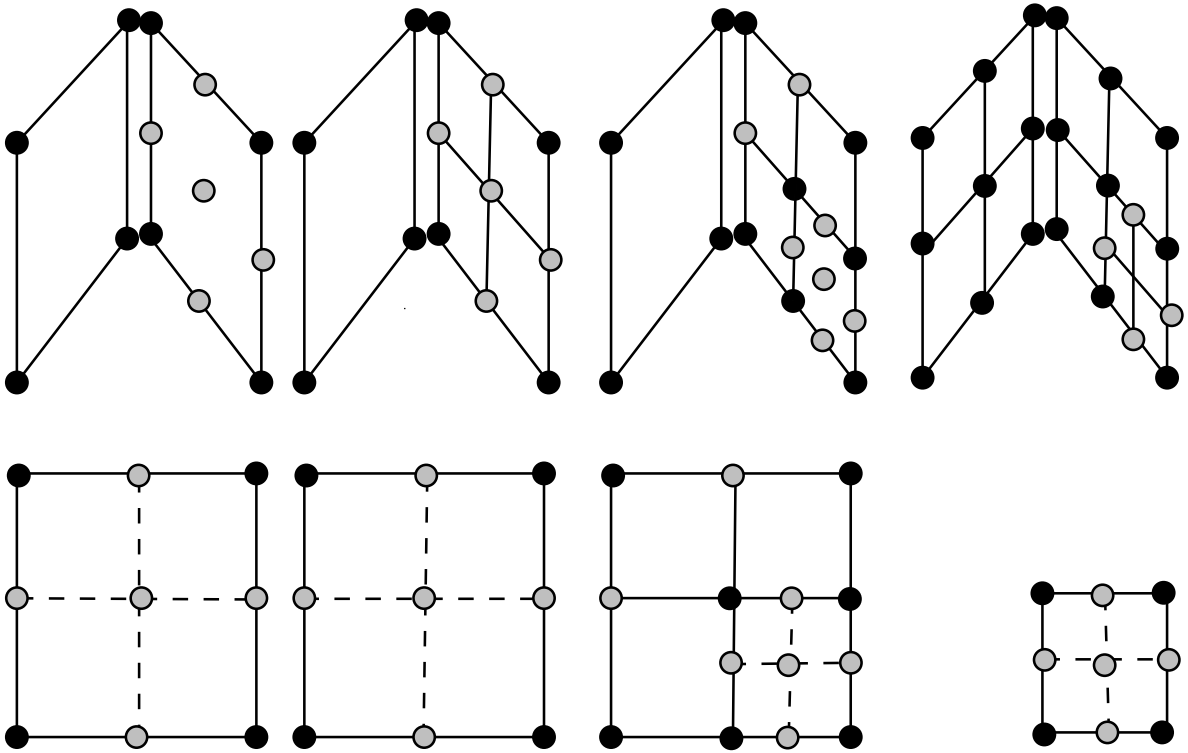
- Anisotropic refinement may switch direction at an interface,
- forcing isotropic refinement in that case is not an option, as it would result in propagation of the costly isotropic refinement through the mesh



- resolved by insertion of a 'hanging node' on both neighbour faces.

# Adaptive edges on quadrilateral faces

Edges between mid-nodes carry the central node on the face and force a unique tessellation.

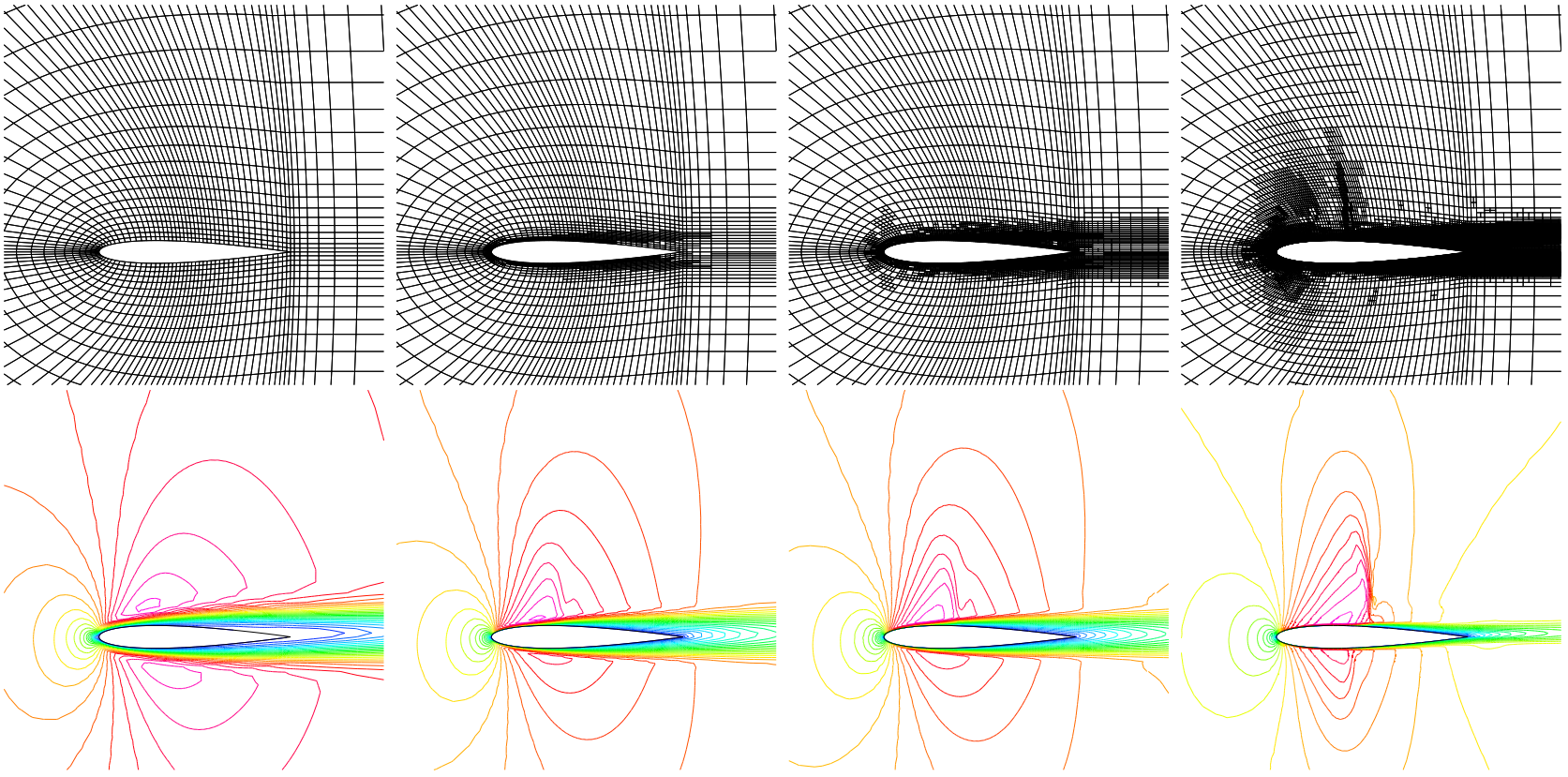


# Anisotropic refinement algorithm

1. Evaluate the anisotropic sensor field on edges,
2. Determine thresholds for refinement.
3. Loop over all elements,
   - 3.1 Flag all edges that have a sensor value above threshold,
   - 3.2 Flag all edges that will have more than one level difference,
   - 3.3 Find the smallest pattern that contains all flagged edges,
   - 3.4 Mark the element with this refinement pattern,
   - 3.5 Find/Create the adaptation edges for this pattern.
4. If new adaptation edges were created, goto 3,
5. Create the mid-nodes on the adaptation edges,
6. Create the children elements using those nodes,
7. Create boundary faces for the children.
8. Remove all adaptive edges that are not hanging.

# Example: Adaptation from Euler to Navier-Stokes grid

NACA 0012, Ma=0.8, $\alpha = 1.25°$, Re= $10^5$, adapted using first order velocity differences across edges.

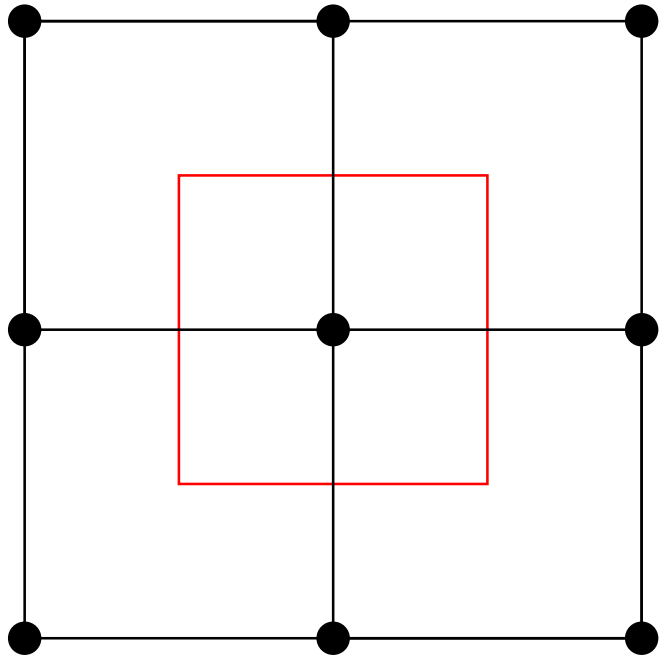


level 0 level 2 level 4 level 6

# Contents

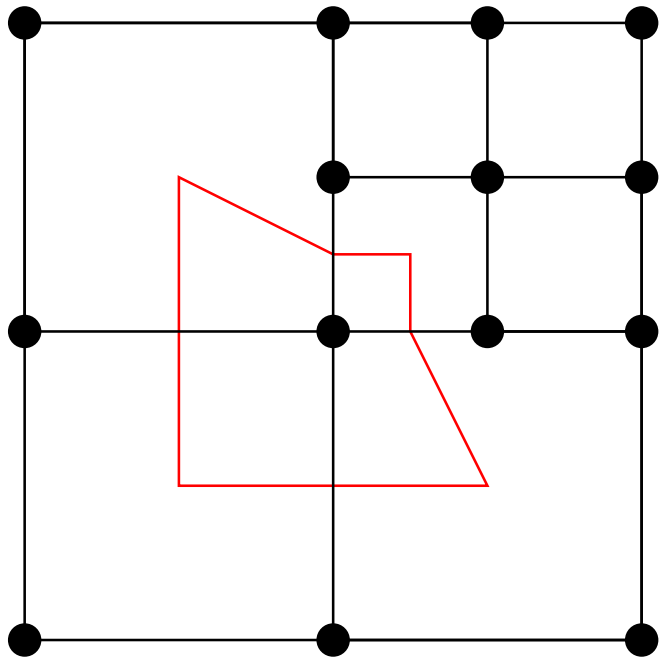
# Exactness of gradient and viscous operators

On a regular grid standard edge-based gradient operators are exact.



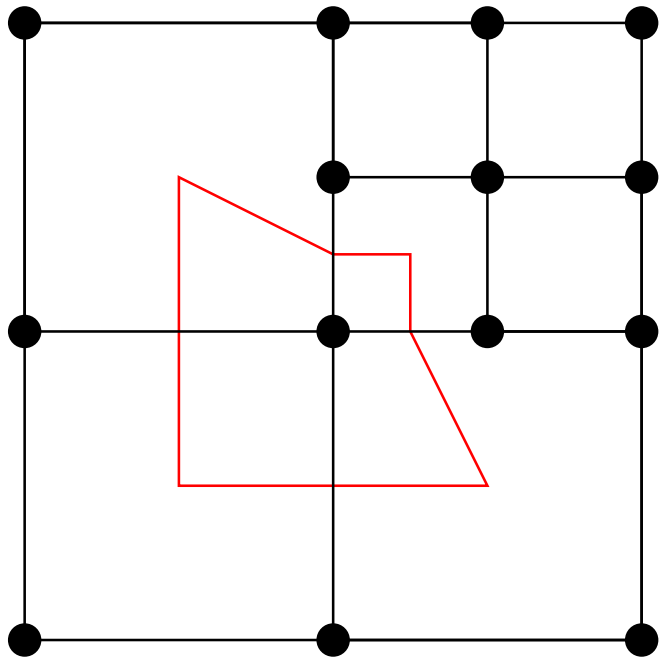
# Exactness of gradient and viscous operators

- On a refined, irregular grid edge-based gradient operators will have large errors, worse for 2nd derivatives.
- Efficient adaptation will need to have linearly-exact operators that can cope with such mesh distortions.
- E.g. cell-based gradients in vertex-centred schemes, or linearly exact dissipations in cell-vertex schemes.



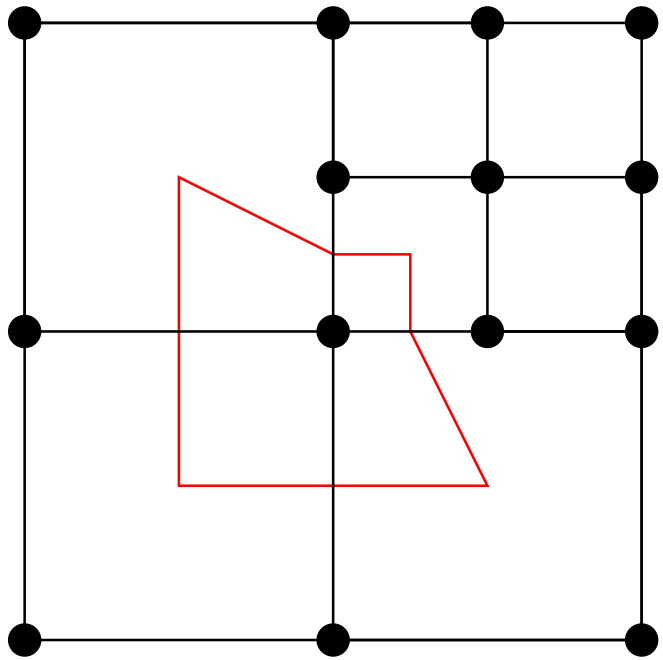
# Exactness of gradient and viscous operators

- On a refined, irregular grid edge-based gradient operators will have large errors, worse for 2nd derivatives.
- Efficient adaptation will need to have linearly-exact operators that can cope with such mesh distortions.
- E.g. cell-based gradients in vertex-centred schemes, or linearly exact dissipations in cell-vertex schemes.

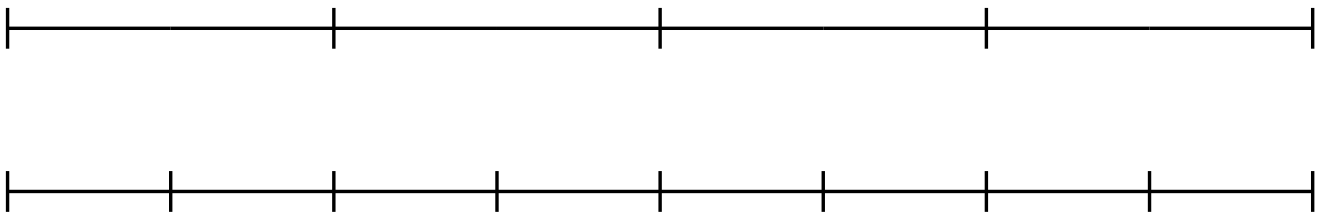


# Exactness of gradient and viscous operators

- On a refined, irregular grid edge-based gradient operators will have large errors, worse for 2nd derivatives.
- Efficient adaptation will need to have linearly-exact operators that can cope with such mesh distortions.
- E.g. cell-based gradients in vertex-centred schemes, or linearly exact dissipations in cell-vertex schemes.

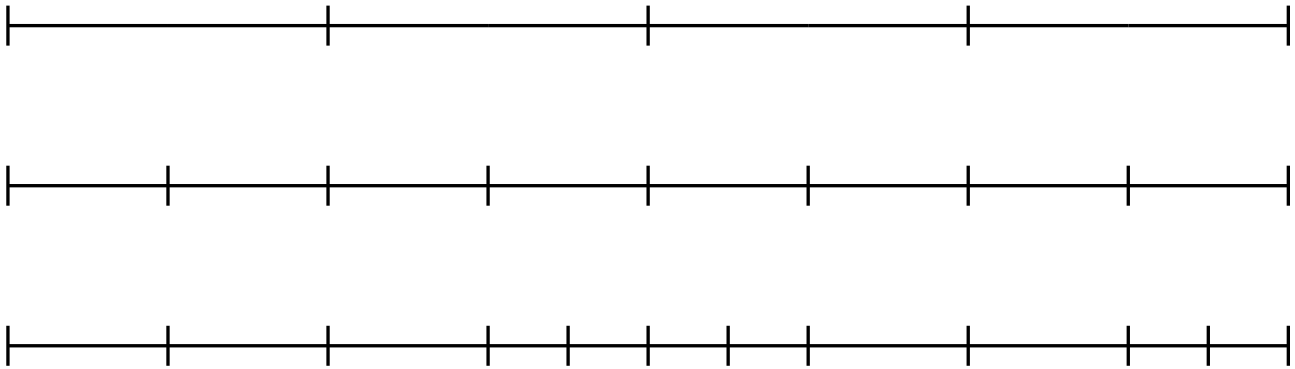


# Multigrid: How to define levels?



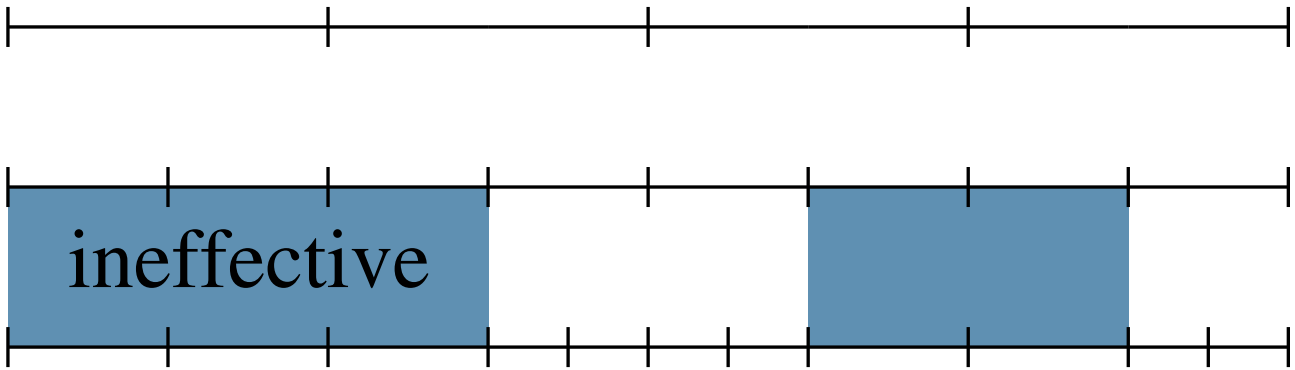
Normal, nested multigrid

# Multigrid: How to define levels?



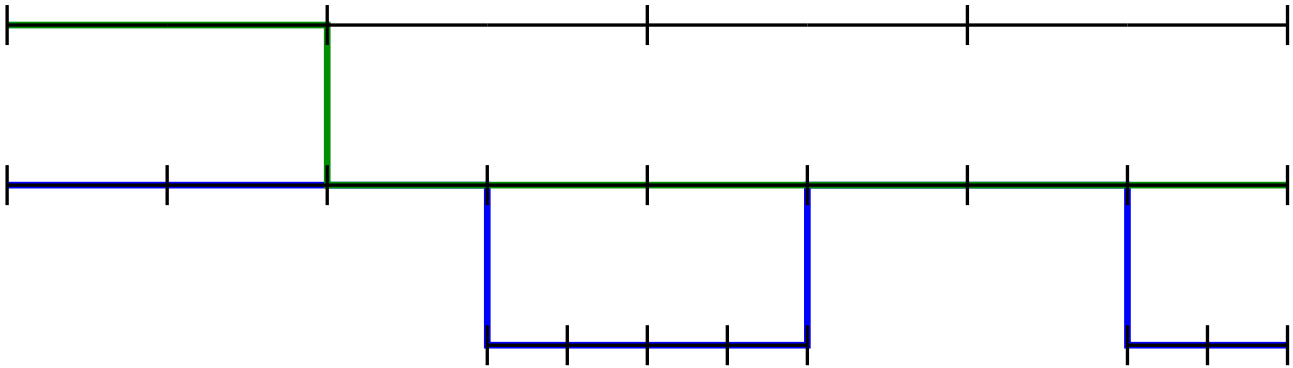
One level of refinement

# Multigrid: How to define levels?



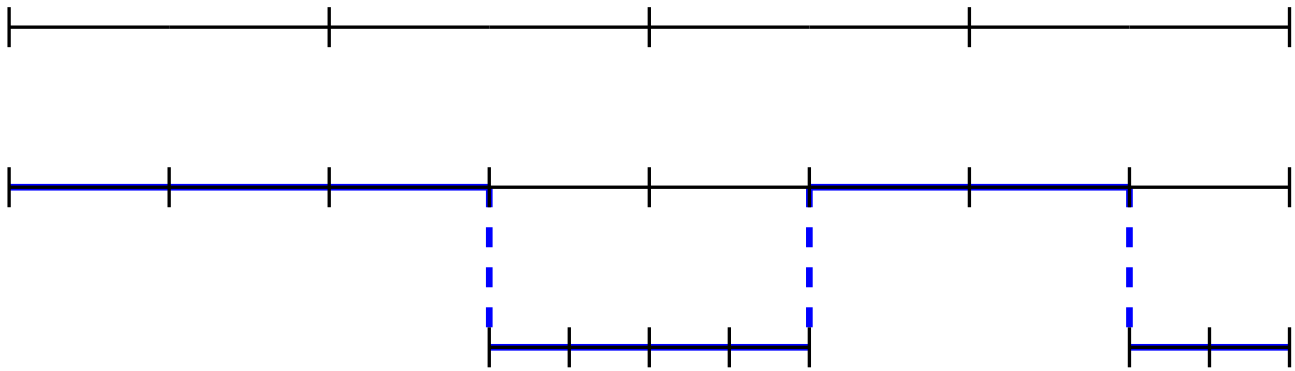

No multigrid coarsening in shaded areas, duplicated work

# Multigrid: How to define levels?



Redefining multigrid levels is costly, can be difficult may also include duplicated work

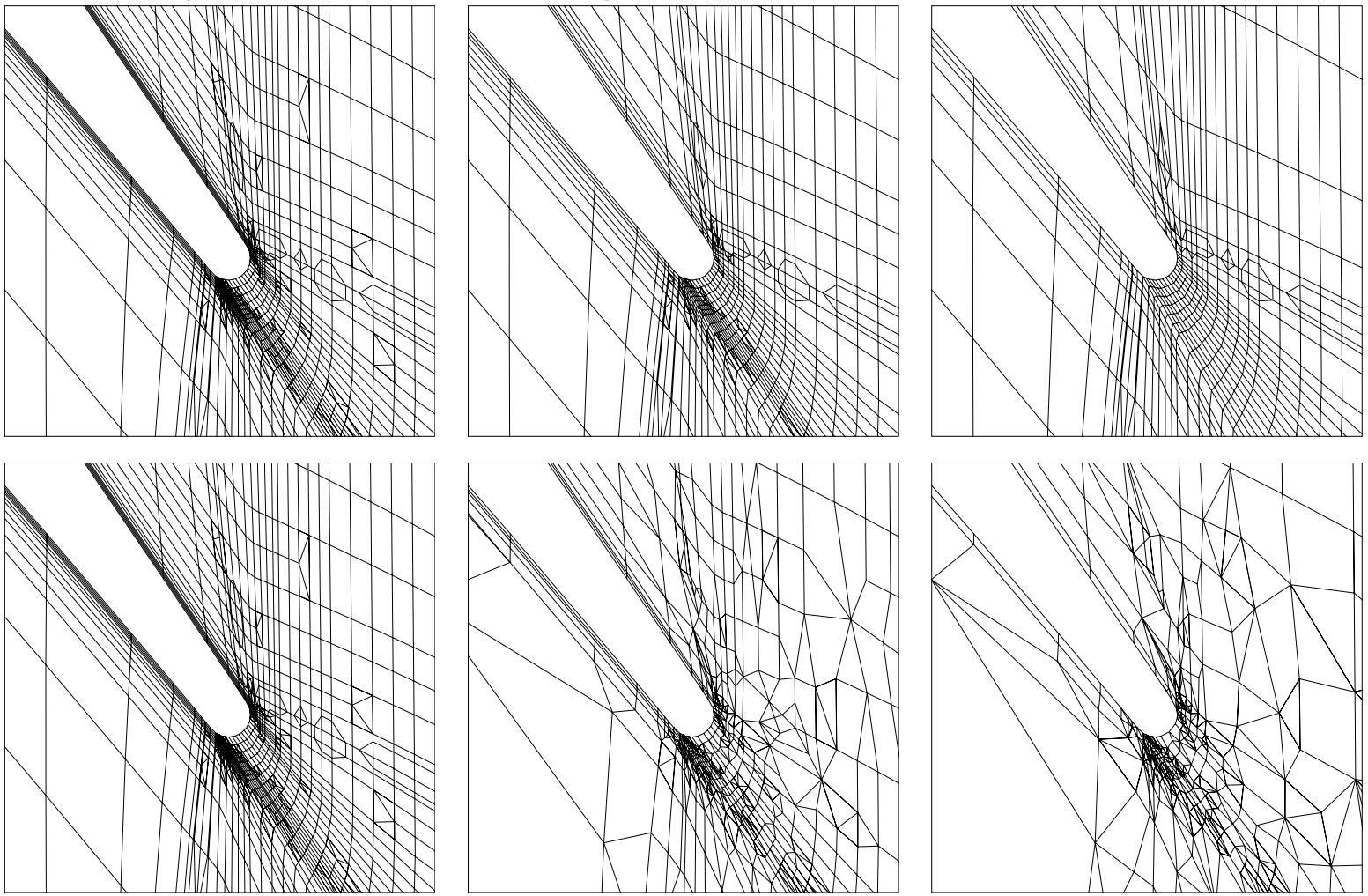
# Multigrid: How to define levels?



Zonal multigrid (Mavriplis, 1995): only solve for adapted grid on the adapted patch. Needs conservative interface, only loose coupling may affect convergence rate.

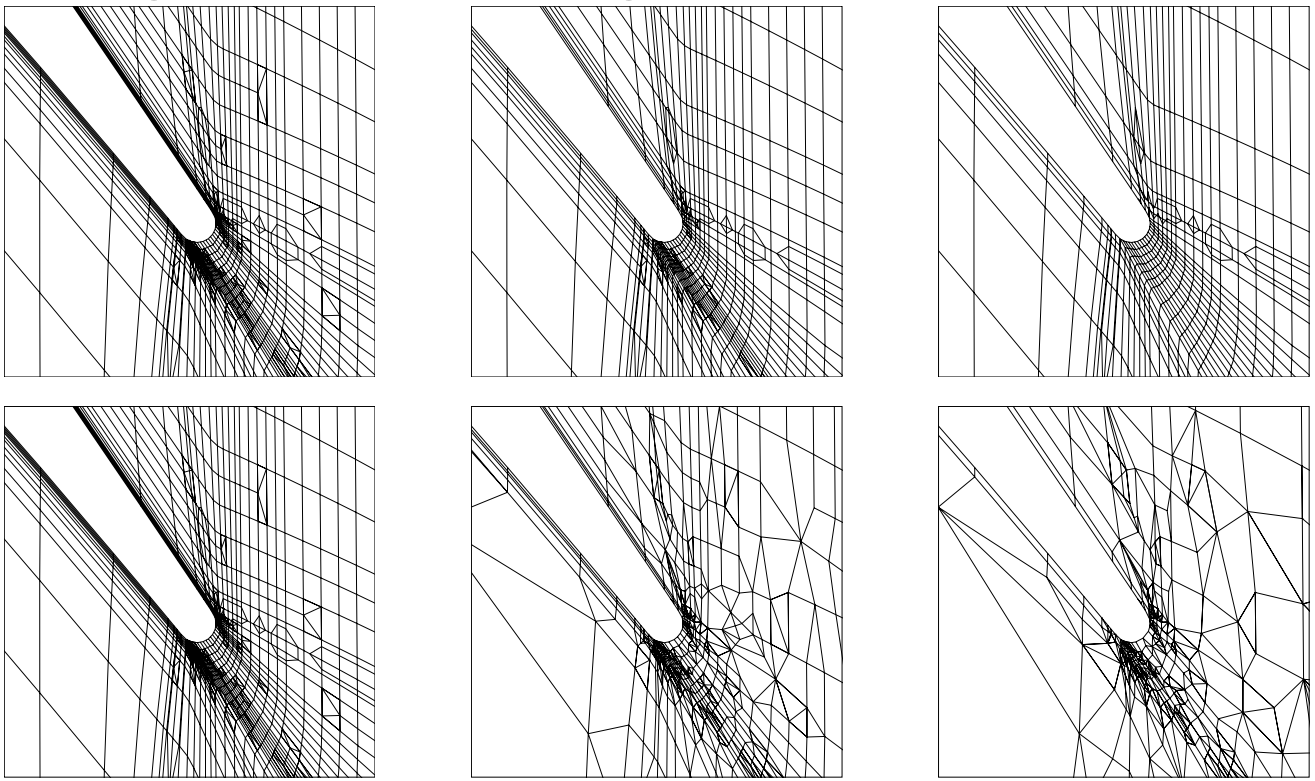
# Re-coarsening vs. using adaptive hierarchy for multi-grid

Two levels of anisotropic refinement compared to two levels of coarsening from the refined grid.



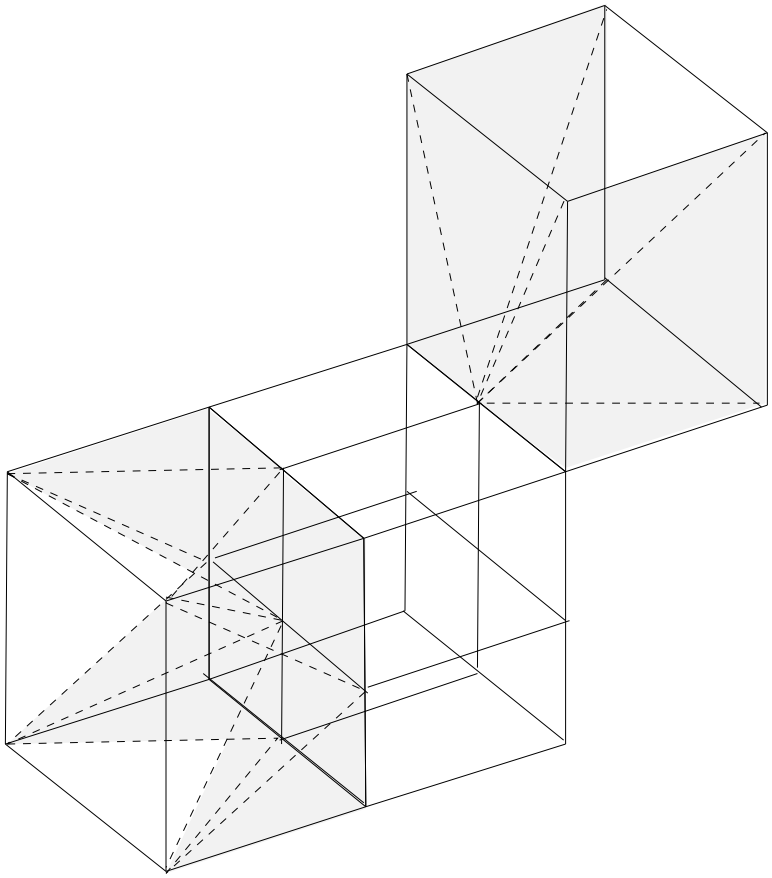
# Re-coarsening vs. using adaptive hierarchy for multi-grid

Two levels of anisotropic refinement compared to two levels of coarsening from the refined grid.



- "Un-adapted" coarse grids are better for the near-field, coarsening approach also coarsens the farfield.
- Both approaches need to be combined.

# Adaptation interfaces: hanging node treatment



**Buffering:** Tessellate the cell with hanging nodes into primitive elements, some cases may need insertion of a central node.

**Solver:** Support hanging nodes/polyhedral elements in the discretisation

# Hanging node treatment

Buffering:

+ No solver change needed.
- In 3D and very local refinement there may be more buffer elements than refined elements.
- Poor element quality at the interface.

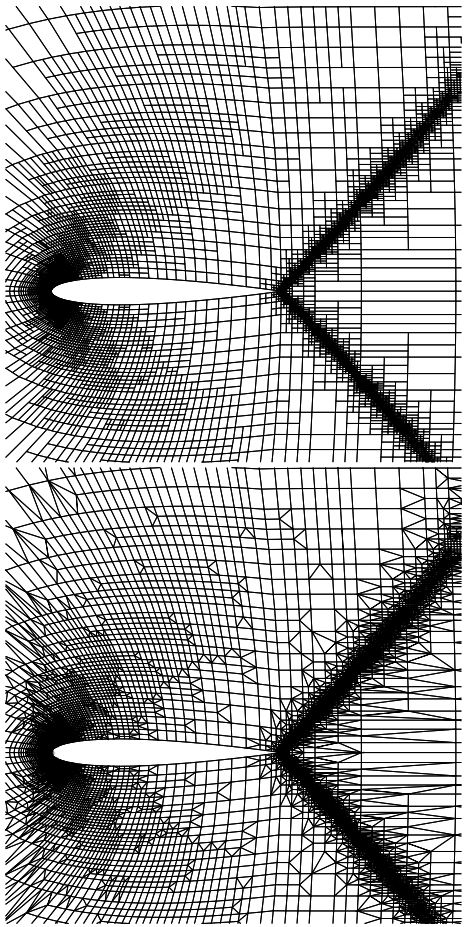
Solver treatment:

- Requires solver change.
+ Allows to define control volumes with better accuracy and smoothness
+ No additional elements.

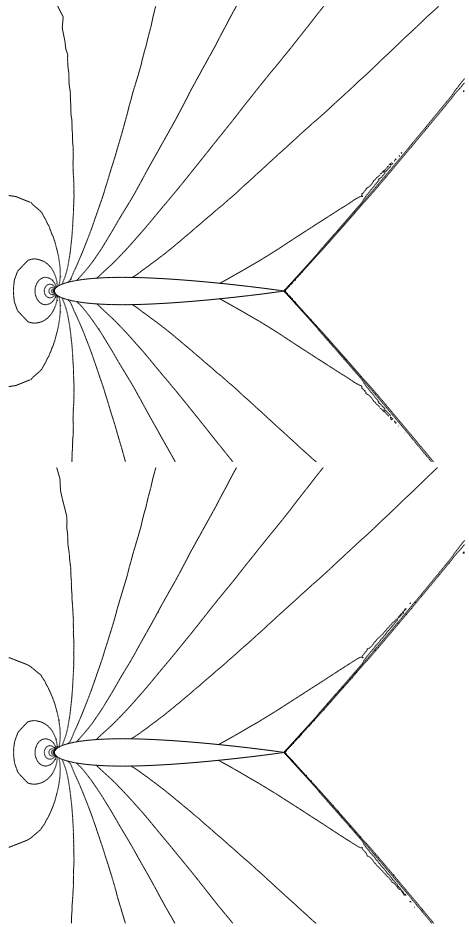
# Comparison of interface approaches

hanging nodes, linearly transparent dissipation

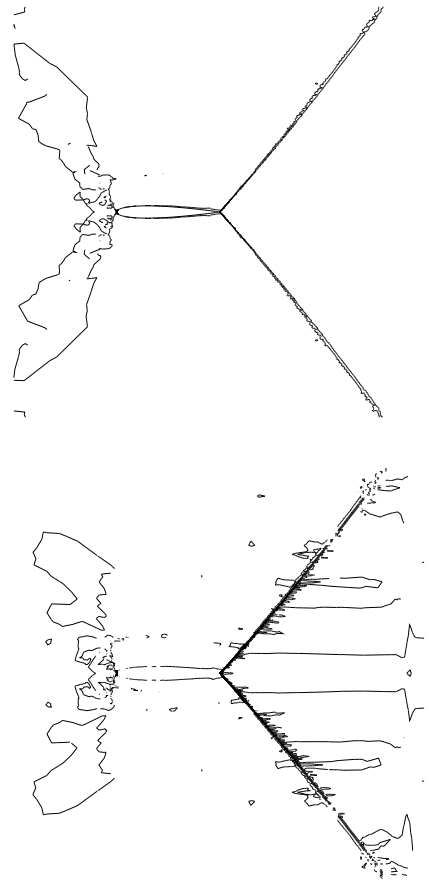
buffered: 50% more cells, 10% more CPU time in 2D, 30% in 3D.



grid



Mach



Entropy

# Contents

# Conclusions

- A variety of sensors is available for finite volume methods,
- Good sensors are expensive, but still don't offer error bounds for non-linear equations.
- Sensors need be anisotropic for efficiency.
- Sensors need to be adjoint-weighted for efficiency and robustness.
- remeshing provides the best grids and has least impact on the solver, but is costly, not suitable for unsteady adaptation.
- *r*-refinement can be part of a solution, suitable for unsteady, but needs to be combined with other techniques for enrichment.
- *h*-refinement can be local and effective for unsteady adapt., but has significant ramifications for the solver on grid regularity and hanging nodes.

# Conclusions

- A variety of sensors is available for finite volume methods,
- Good sensors are expensive, but still don't offer error bounds for non-linear equations.
- Sensors need be anisotropic for efficiency.
- Sensors need to be adjoint-weighted for efficiency and robustness.
- remeshing provides the best grids and has least impact on the solver, but is costly, not suitable for unsteady adaptation.
- *r*-refinement can be part of a solution, suitable for unsteady, but needs to be combined with other techniques for enrichment.
- *h*-refinement can be local and effective for unsteady adapt., but has significant ramifications for the solver on grid regularity and hanging nodes.

# Conclusions

- A variety of sensors is available for finite volume methods,
- Good sensors are expensive, but still don't offer error bounds for non-linear equations.
- Sensors need be anisotropic for efficiency.
- Sensors need to be adjoint-weighted for efficiency and robustness.
- remeshing provides the best grids and has least impact on the solver, but is costly, not suitable for unsteady adaptation.
- *r*-refinement can be part of a solution, suitable for unsteady, but needs to be combined with other techniques for enrichment.
- *h*-refinement can be local and effective for unsteady adapt., but has significant ramifications for the solver on grid regularity and hanging nodes.

# Conclusions

- A variety of sensors is available for finite volume methods,
- Good sensors are expensive, but still don't offer error bounds for non-linear equations.
- Sensors need be anisotropic for efficiency.
- Sensors need to be adjoint-weighted for efficiency and robustness.
- remeshing provides the best grids and has least impact on the solver, but is costly, not suitable for unsteady adaptation.
- $r$-refinement can be part of a solution, suitable for unsteady, but needs to be combined with other techniques for enrichment.
- $h$-refinement can be local and effective for unsteady adapt., but has significant ramifications for the solver on grid regularity and hanging nodes.

# Conclusions

- A variety of sensors is available for finite volume methods,
- Good sensors are expensive, but still don't offer error bounds for non-linear equations.
- Sensors need be anisotropic for efficiency.
- Sensors need to be adjoint-weighted for efficiency and robustness.
- remeshing provides the best grids and has least impact on the solver, but is costly, not suitable for unsteady adaptation.
- $r$-refinement can be part of a solution, suitable for unsteady, but needs to be combined with other techniques for enrichment.
- $h$-refinement can be local and effective for unsteady adapt., but has significant ramifications for the solver on grid regularity and hanging nodes.

# Conclusions

- A variety of sensors is available for finite volume methods,
- Good sensors are expensive, but still don't offer error bounds for non-linear equations.
- Sensors need be anisotropic for efficiency.
- Sensors need to be adjoint-weighted for efficiency and robustness.
- remeshing provides the best grids and has least impact on the solver, but is costly, not suitable for unsteady adaptation.
- *r*-refinement can be part of a solution, suitable for unsteady, but needs to be combined with other techniques for enrichment.
- *h*-refinement can be local and effective for unsteady adapt., but has significant ramifications for the solver on grid regularity and hanging nodes.

# Conclusions

- A variety of sensors is available for finite volume methods,
- Good sensors are expensive, but still don't offer error bounds for non-linear equations.
- Sensors need be anisotropic for efficiency.
- Sensors need to be adjoint-weighted for efficiency and robustness.
- remeshing provides the best grids and has least impact on the solver, but is costly, not suitable for unsteady adaptation.
- *r*-refinement can be part of a solution, suitable for unsteady, but needs to be combined with other techniques for enrichment.
- *h*-refinement can be local and effective for unsteady adapt., but has significant ramifications for the solver on grid regularity and hanging nodes.

# Acknowledgements

This presentation has been given within the **About Flow** project on

"Adjoint-based optimisation of industrial and unsteady flows".

`http://aboutflow.sems.qmul.ac.uk`

About Flow has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under Grant Agreement No. 317006.

Rolls-Royce engys esi get it right Inria informatics mathematics

Queen Mary University of London RWTH AACHEN UNIVERSITY

National Technical University of Athens