Goal-Oriented Anisotropic Mesh Adaptation in the SU2 Framework

SU2 Conference 2020

Brian C. Munguía, Juan J. Alonso, and Adrien Loseille

Inria

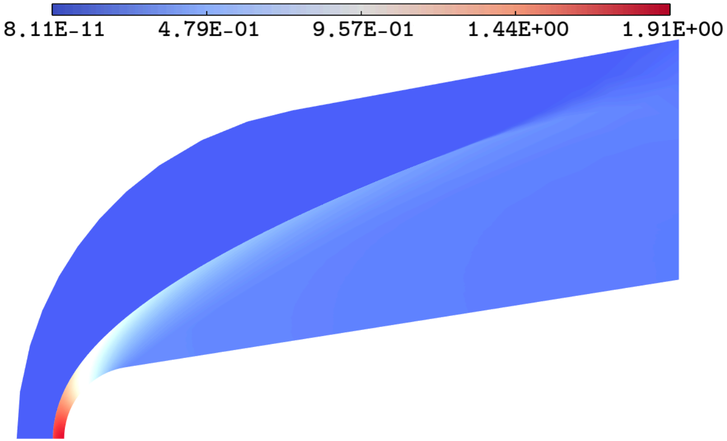
Stanford University

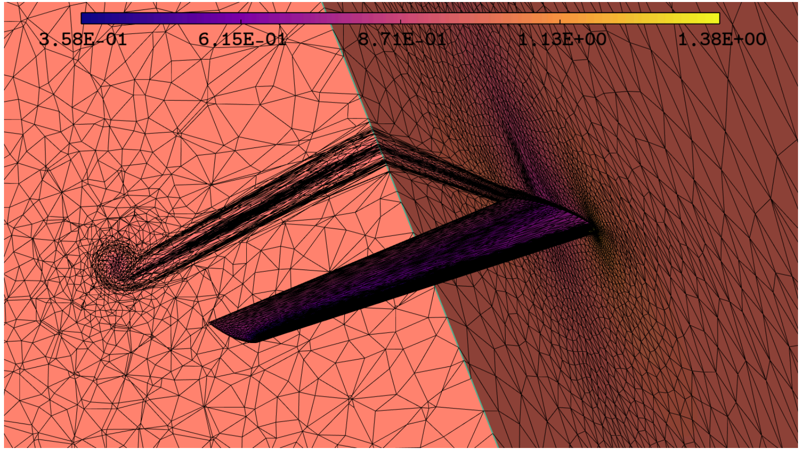
Overview

- Motivation
- Previous work
- Anisotropic mesh adaptation
  - Feature-based vs goal-oriented
  - Metric spaces
  - *A priori* error estimate
- Implementation
  - Adaptation framework
  - Modifications to NS and SST solver
  - SU2-AMG tutorial
- Mesh adaptation results
  - RAE 2822
- Conclusions and future work

Motivation

- Accuracy and robustness of high-fidelity analysis and design are highly dependent on discretization
- Mesh generation is a significant bottleneck in the CFD workflow
  [CFD Vision 2030]
- Physical phenomena are highly anisotropic in nature



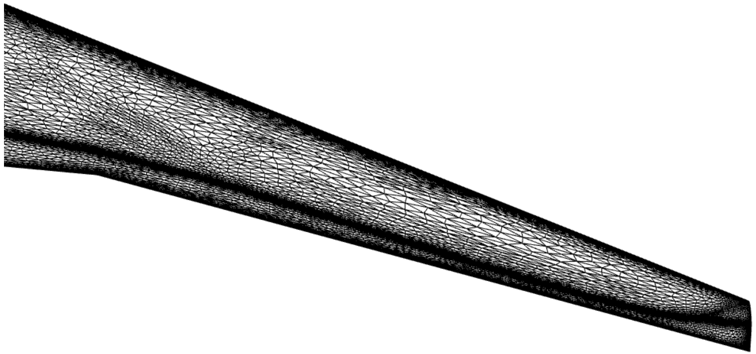


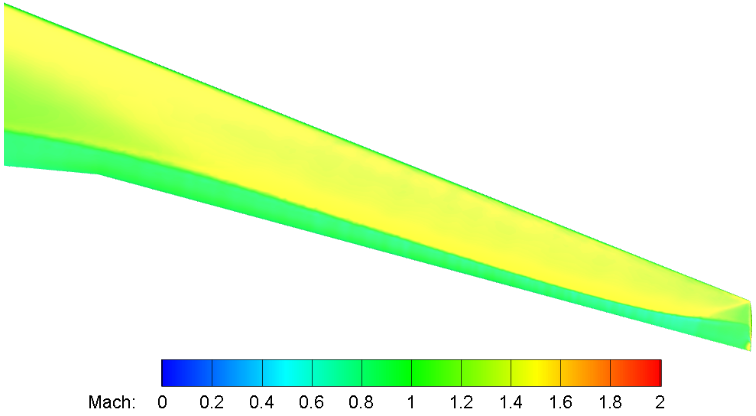
Previous work

- Grid adaptation for functional outputs using an *a posteriori* error estimate
  [Venditti and Darmofal, 2000]
- Development of *a priori* interpolation error estimates
  [Formaggia and Perotto, 2001]
- Continuous mesh framework with optimal metric for error control in $\mathbf{L}^p$-norm
  [Alauzet et al., 2006]
- *A priori* functional error estimate for 3D Euler equations and Spalart-Allmaras
  [Loseille et al., 2010], [Frazza, 2018]

Previous work in SU2

- Isotropic adaptation
  [Palacios et al., 2012], [Copeland et al., 2013]
  - Gradients
  - Adjoint-weighted residuals
- Feature-based anisotropic adaptation
  [Loseille et al., 2016]
- Goal-oriented adaptation for Euler equations and TNE2
  [Munguía et al., 2020]
  - Adjoint-weighted error in fluxes and chemical source terms

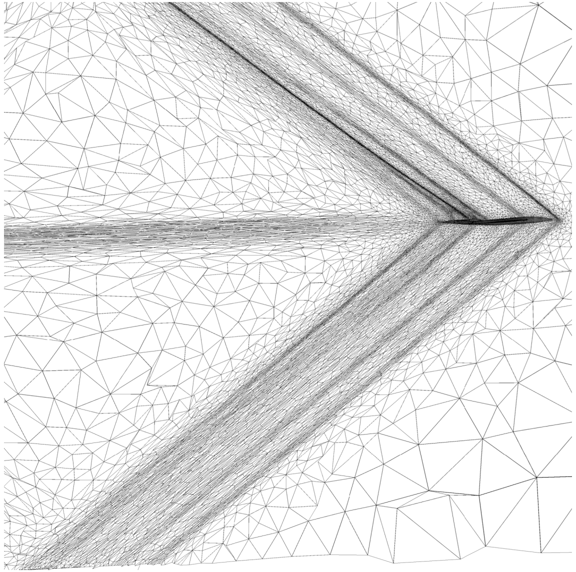




Feature-based vs goal-oriented adaptation

Feature-based

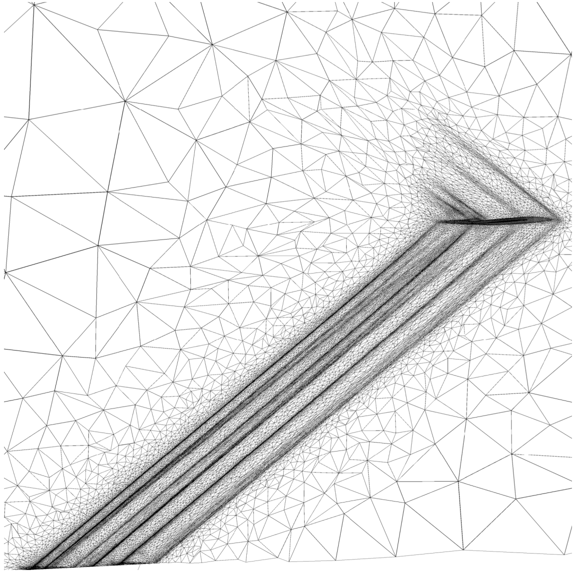
Best mesh to compute characteristics of a solution u



$\|u - \Pi_h u\|_{L^p(\Omega_h)}$

Goal-oriented

Best mesh to compute a functional $f(u)$



$\|f(u) - f(\Pi_h u)\|_{L^p(\Omega_h)}$

Inria

Stanford University

Metric space

- Linear elements → second-order interpolation error
- Riemannian metric space given by Hessians, e.g. in 2D:

$$\mathcal{M}(\boldsymbol{x}) = \begin{bmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

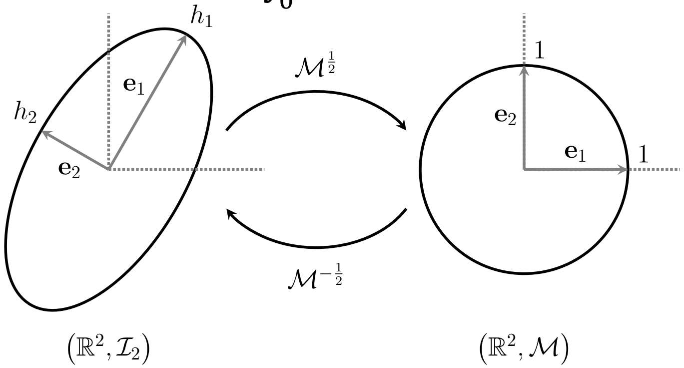
- Edge length in Euclidean metric space (i.e. $\mathcal{M}$ constant in space):

$$l_{\mathcal{M}}^2 = \vec{s}^T \mathcal{M} \vec{s} h^2 = (ax^2 + 2bxy + cy^2)h^2$$

$$\mathcal{M} = \mathcal{R} \begin{bmatrix} 1/h_1^2 & 0 \\ 0 & 1/h_2^2 \end{bmatrix} \mathcal{R}^T$$

- Edge length in Riemannian metric space:

$$l_{\mathcal{M}}(\boldsymbol{pq}) = \int_0^1 \sqrt{\boldsymbol{pq}^T \mathcal{M}(\boldsymbol{p} + t\boldsymbol{pq})\boldsymbol{pq}}\,\mathrm{d}t$$



Discrete-continuous duality

Discrete	Continuous
Element K	Metric tensor $\mathcal{M}$
Element volume $\lvert K \rvert$	$d^{-1} = \sqrt{\det \mathcal{M}(\boldsymbol{x})}$
Mesh X of Ω_h	Riemannian metric space $\mathbf{M} = \mathcal{M}(\boldsymbol{x})$ of Ω
Number of vertices N	Complexity $\mathcal{C}(\mathbf{M}) = \int_\Omega \sqrt{\det \mathcal{M}(\boldsymbol{x})}\, \mathrm{d}\Omega = \mathcal{N}$

Optimal metric

- Minimize error in $\mathbf{L}^p$-norm for problem of dimension d:

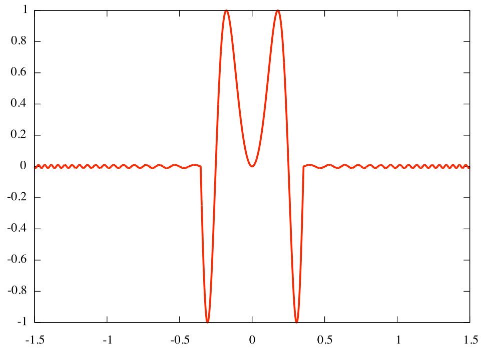
$$\mathcal{M}_{\mathbf{L}^p} = \underbrace{\mathcal{N}^{\frac{2}{d}}}_{1} \underbrace{\left(\int_\Omega (\det|\mathcal{H}(\boldsymbol{x})|)^{\frac{p}{2p+d}} \, \mathrm{d}\Omega \right)^{-\frac{2}{d}}}_{2} \underbrace{(\det|\mathcal{H}(\boldsymbol{x})|)^{-\frac{1}{2p+d}} |\mathcal{H}(\boldsymbol{x})|}_{3}$$

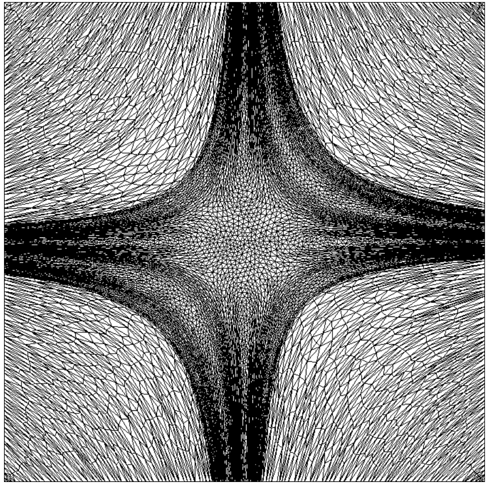
1. Desired complexity
2. Global normalization
3. Local normalization

Free parameters: $\mathcal{N}$ and p

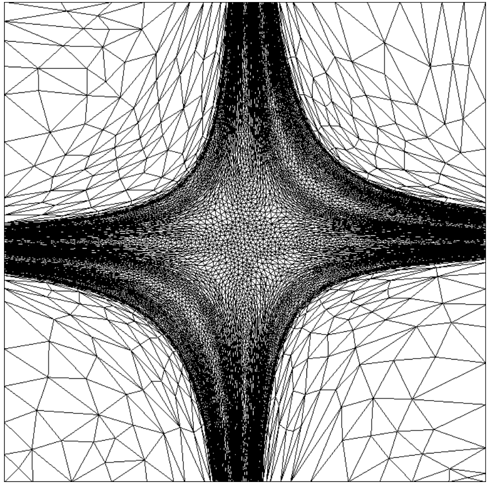
Optimal metric

- Choice of p dictated by problem
  - Larger p damps small amplitude variations
  - Previous work suggests $p = 1,2$ for Euler, $p = 4$ for RANS [Frazza, 2018]





$p = 1$



$p = 4$

Inviscid error estimate

- For Euler equations, can approximate error using flux and source differencing:

$$R(u) = \nabla \cdot F(u) - Q(u) = 0$$

$$\delta f \approx \int_{\Omega_h} \bar{u}^T \left[\left(\nabla \cdot F_h(u) - \nabla \cdot F(u) \right) - \left(Q_h(u) - Q(u) \right) \right] \mathrm{d}\Omega_h$$

- Integrating by parts and neglecting boundary terms:

$$\delta f \approx \int_{\Omega_h} \left[\nabla \bar{u}^T \left(F(u) - F_h(u) \right) + \bar{u}^T \left(Q(u) - Q_h(u) \right) \right] \mathrm{d}\Omega_h$$

- Adjoint-weighted Hessian:

$$\mathcal{H}(\boldsymbol{x}) = \left| \frac{\partial \bar{u}_j}{\partial x_i} \right| \left| H\left(F_i(u_j) \right) \right| + \left| \bar{u}_j \right| \left| H\left(Q(u_j) \right) \right|$$

- OK results for Euler and TNE2, but doesn’t properly weigh errors due to gradients

Viscous error estimate

- Approximate error by linearizing wrt conservative variables

$$\delta f \approx \int_{\Omega_\mathrm{h}} \bar{u}^\mathrm{T} \frac{\partial R}{\partial u} (u - u_h) \mathrm{d}\Omega_h$$

- First we need to manipulate terms of the following forms:

$(w\bar{u})^T \, (u - u_h)$	Zeroth-order e.g. source terms
$(w\bar{u})^T \frac{\partial (u - u_h)}{\partial x_i}$	First-order e.g. convective terms
$(w\bar{u})^T \frac{\partial^2 (u - u_h)}{\partial x_i \partial x_j}$	Second-order e.g. viscous terms

Viscous error estimate

- Neglecting boundary terms, we obtain the following form for viscous terms:

$$\int_{\Omega} (w\bar{u})^T \frac{\partial^2 (u - u_h)}{\partial x_i \partial x_j} \mathrm{d}\Omega \approx \int_{\Omega} \left(w \frac{\partial^2 \bar{u}}{\partial x_i \partial x_j} \right)^T (u - u_h) \mathrm{d}\Omega$$

- Weights w obtained by linearizing governing equations
- E.g. second-order energy error due to divergence of heat flux

$$\nabla^2 T = \frac{1}{\rho c_v} [\nabla^2(\rho c_v T) - c_v T \nabla^2 \rho]$$

$$\bar{H}_{\rho e} = -\frac{\lambda + \lambda_t}{\rho c_v} \left[(\bar{u}_{\rho e})_{xx} + (\bar{u}_{\rho e})_{yy} + (\bar{u}_{\rho e})_{zz} \right]$$

- Adjoint-weighted Hessian:

$$\mathcal{H}(\boldsymbol{x}) = (|\bar{C}_k| + |\bar{G}_k| + |\bar{H}_k|)|H(u_k)|$$

Simplification for SST error estimate

- Treat blending functions as constant

$$\phi = F_1\phi_1 + (1 - F_1)\phi_1$$

$$F_1 = \tanh(arg_1^4), F_2 = \tanh(arg_2^2)$$

$$arg_1 = \min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega}\right), \frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}d^2}\right]$$

$$arg_2 = \max\left(\frac{2\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega}\right)$$

- Don’t limit shear stress or production

$$\mu_t = \frac{\rho k a_1}{\max(\omega a_1, \Omega F_1)}$$

$$P_k = \max\left(\tau_{t,ij}\frac{\partial u_i}{\partial x_j}, 20\beta^*\rho k\omega\right)$$

- Linearize all other terms, including viscosity and thermal conductivity

Modifications to NS and SST solver

- Store conservative turbulent variables
  - Adjoints in `master/develop` obtained wrt turbulent primitives
  - Error estimate in terms of conservative primal and adjoint variables
- Replace solution clipping with under-relaxation
- Remove ω production limiter

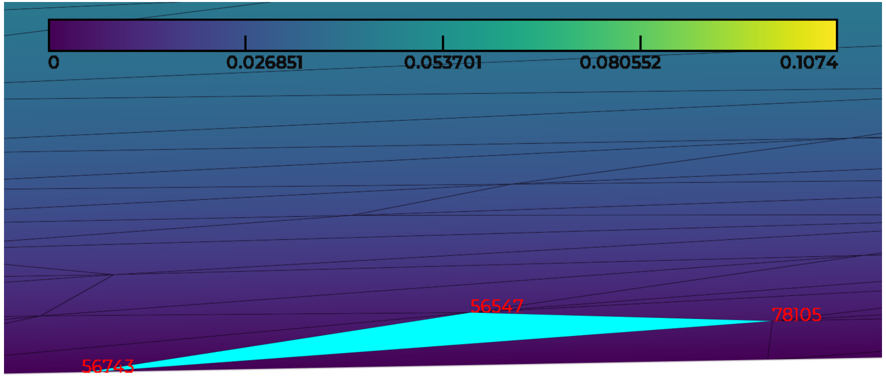
$$P_\omega = \frac{\gamma}{\nu_t} \tau_{t,ij} \frac{\partial u_i}{\partial x_j}$$

Modifications to NS and SST solver

- Use wall distance instead of "normal neighbor" distance in ω wall BC

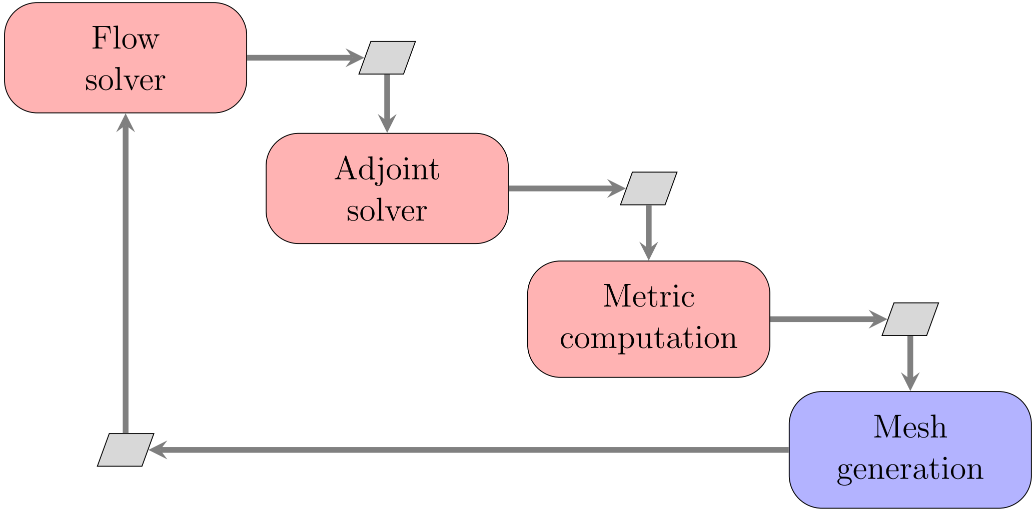
$$\omega_{wall} = \frac{60\nu}{\beta_1 (\Delta d_1)^2}$$

- Use over-relaxed gradient correction
  - More stable on highly non-orthogonal meshes [Jasak, 1996]
- Include Jacobians of
  - Green-Gauss gradients
  - Production and cross-diffusion
  - Laminar and eddy viscosity



Adaptation framework

- SU2: Flow, adjoint, and metric computation
- AMGIO: Conversion to and from GMF
- pyAMG: Mesh adaptation and solution interpolation
- “Stable” branch: `feature_adap`



Running SU2-AMG

- Currently limited to tri and tet meshes
- Background surface mesh
  - Fine representation of surface
  - SU2 or GMF
  - Defaults to initial mesh
- Ridge detection
  - Let AMG detect edges (3D) and corners (2D)
  - python script to append corners based on intersection of markers

```
PYADAP_BACK= rae2822_fine.su2
```

```
PYADAP_RDG= NO
```

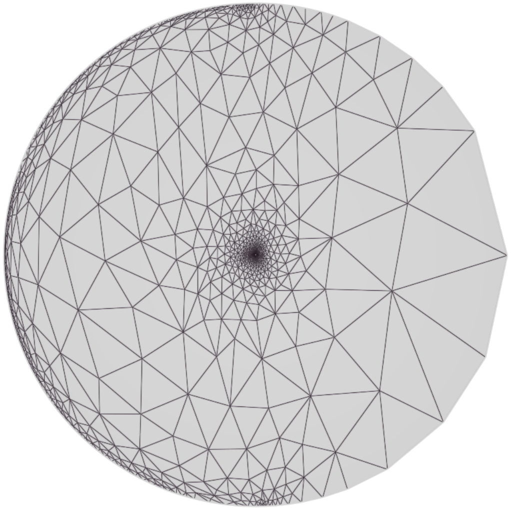
```
$ set_corner_points.py -f rae2822_rans.su2
NCORNERS= 2
1 0
1 512
```

Running SU2-AMG

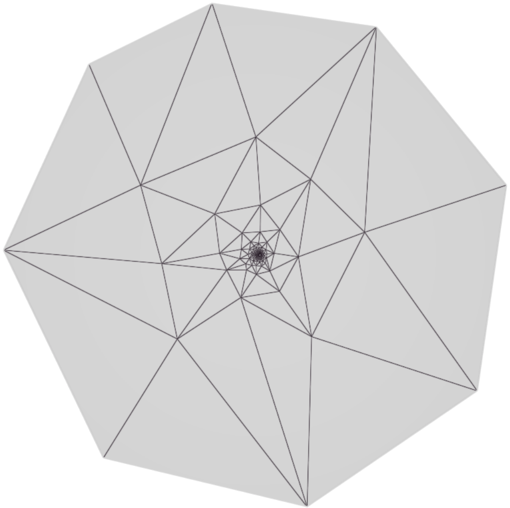
- Adaptation type
  - Goal, Mach, or pressure
- Target complexity
- Iterations per mesh level
- Norm

```
PYADAP_SENSOR= GOAL % MACH, PRES

PYADAP_COMPLEXITY= (30000, 60000, 120000)
PYADAP_SUBITE= (5, 5, 5)
PYADAP_NORM= 2.0
```



$p = 2$



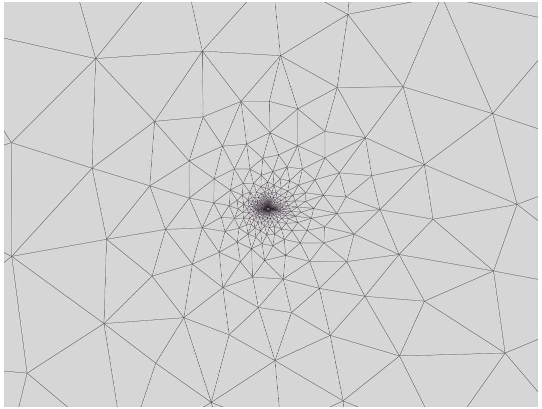
$p = 4$

Running SU2-AMG

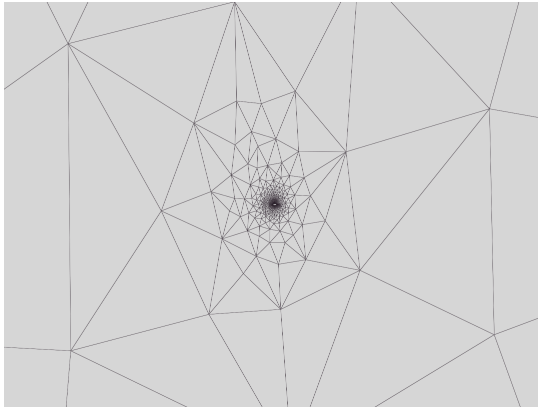
- Maximum cell size
- Minimum cell size
  - Very important for RANS
- Gradation parameter
  - Max ratio of neighboring cell sizes

```
PYADAP_HMAX= 500.0
PYADAP_HMIN= 1.0E-6

PYADAP_HGRAD= 3.0
```



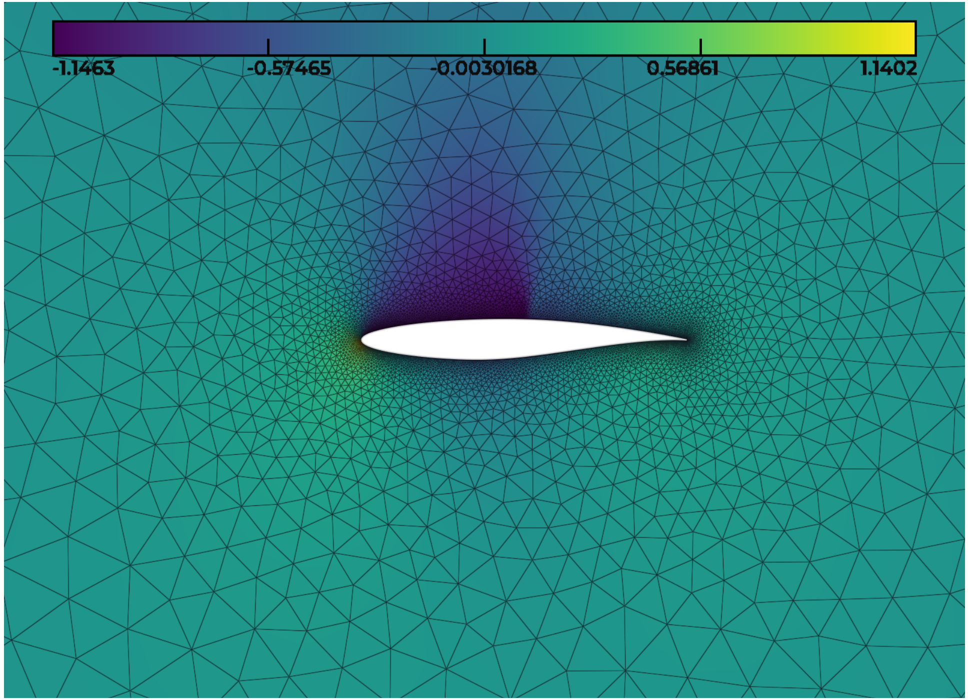
$h_{grad} = 1.5$



$h_{grad} = 3.0$

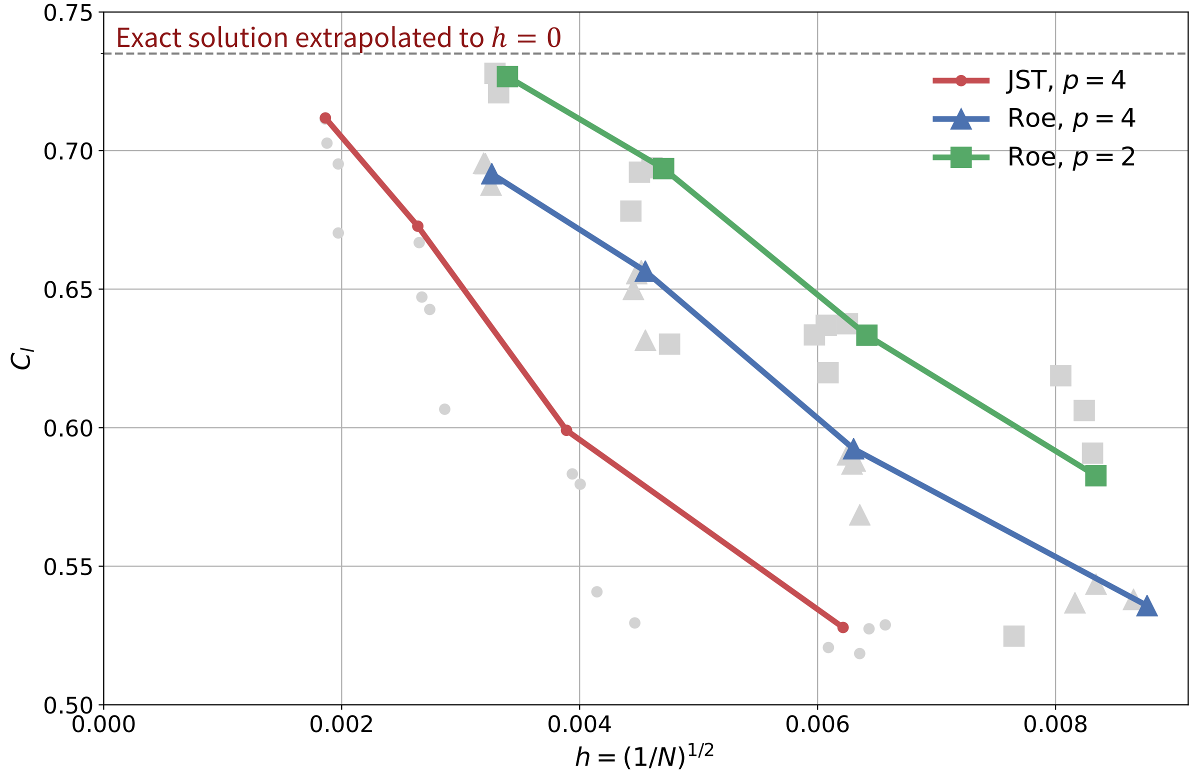
RAE 2822

- $M = 0.729$
- $Re = 6.5 \times 10^6$
- $\alpha = 2.31°$
- $f = c_l$
- $\mathcal{N} = 30000, 60000,$
  $120000, 240000$
- $h_{min} = 1.0 \times 10^{-6}$
- $h_{max} = 500$
- $h_{grad} = 3.0$



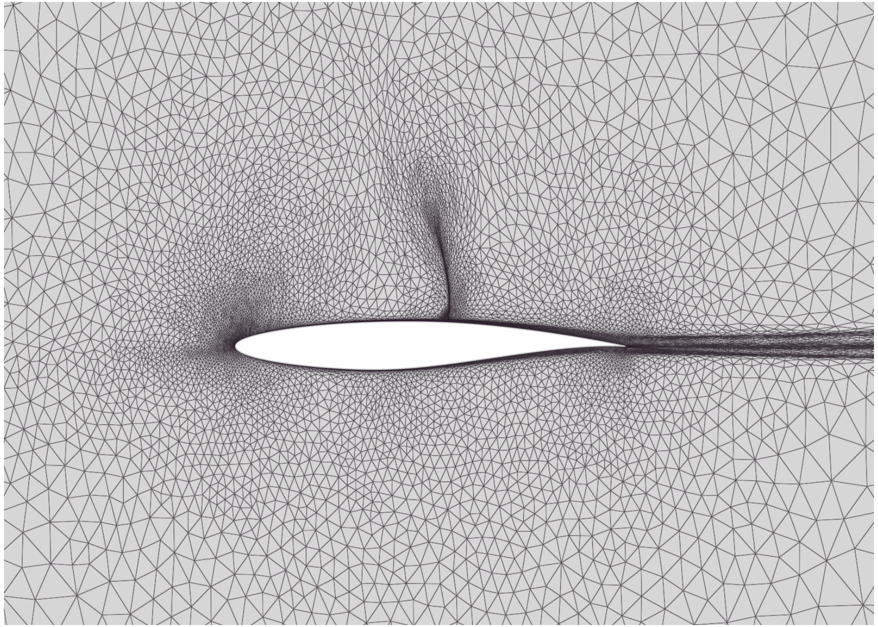
Initial mesh: 29996 points

RAE 2822

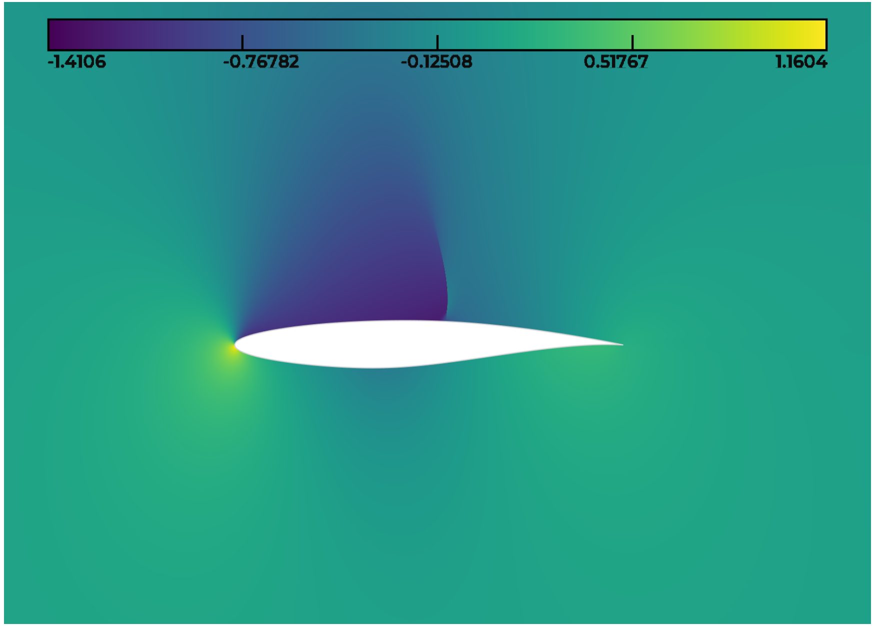


Exact solution extrapolated to $h = 0$
JST, $p = 4$
Roe, $p = 4$
Roe, $p = 2$
C_l
0.75
0.70
0.65
0.60
0.55
0.50
0.000
0.002
0.004
0.006
0.008
$h = (1/N)^{1/2}$

RAE 2822 ($p = 2$)

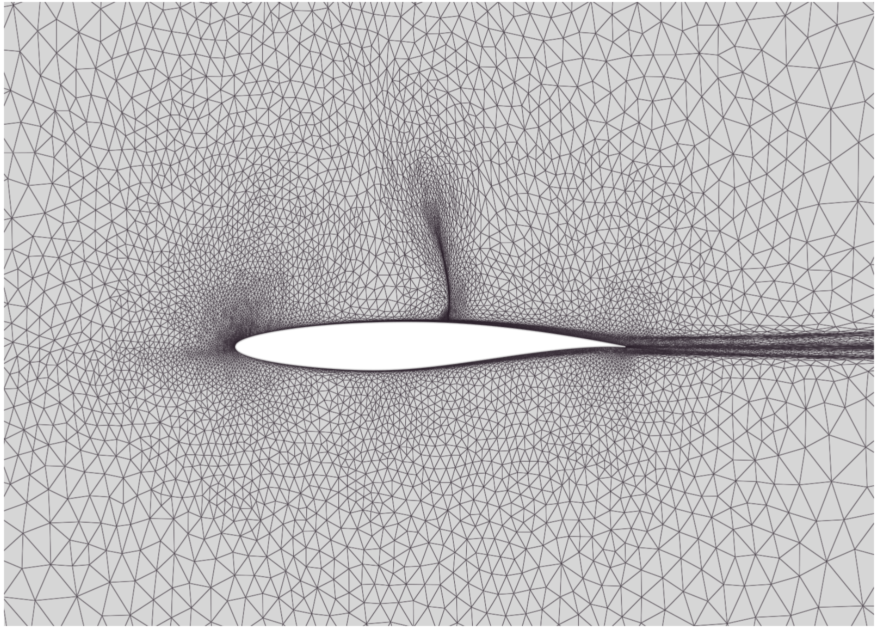


Final mesh: 83687 points

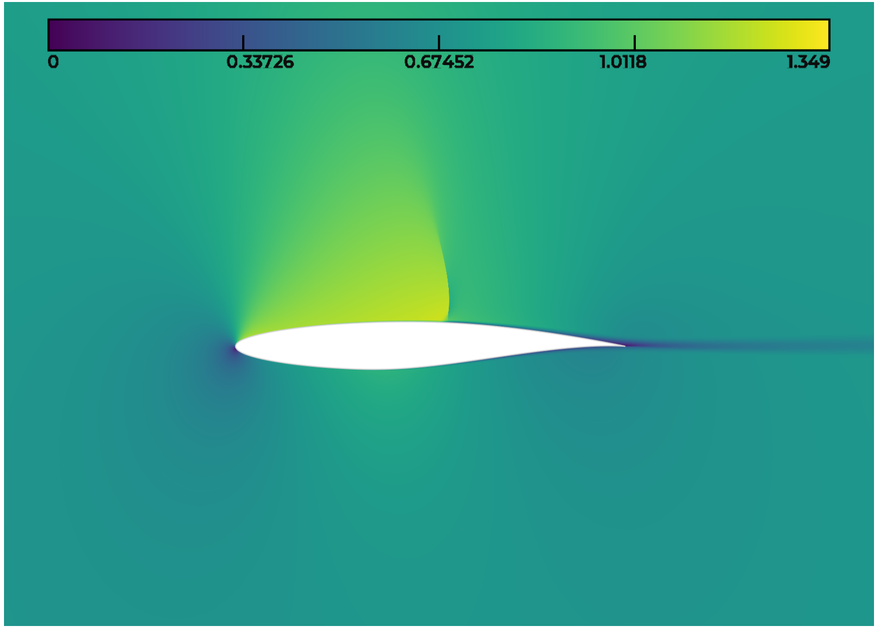


c_p

RAE 2822 ($p = 2$)

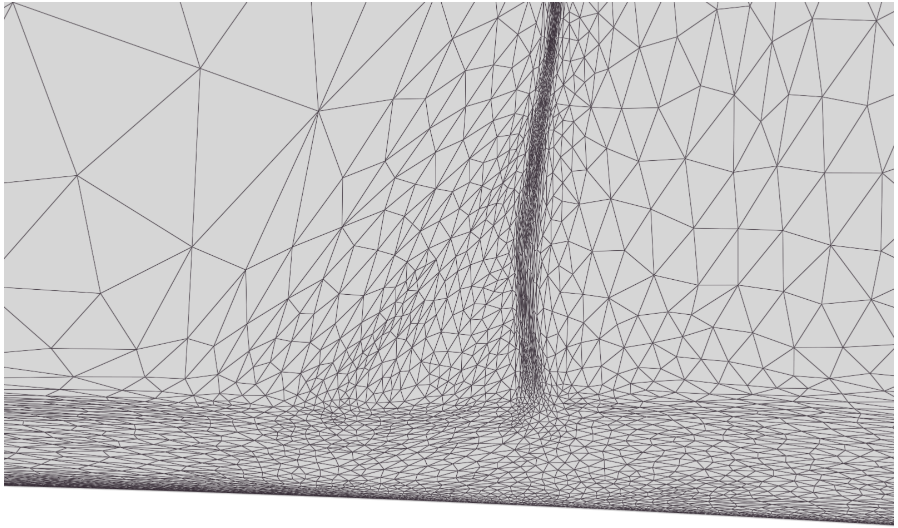


Final mesh: 83687 points

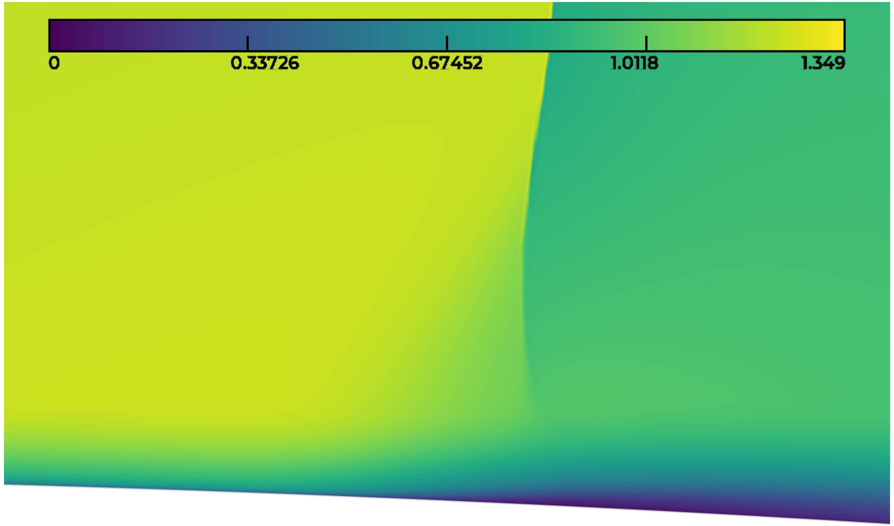


Mach number

RAE 2822 ($p = 2$)



Final mesh: 83687 points



Mach number

Stanford University

Conclusions

- Implemented error estimate and goal-oriented mesh adaptation
  - Euler
  - TNE2
    [Munguía et al., 2020]
  - SST
- Demonstrated ability to properly adapt for viscous, turbulent flows on RAE 2822
- Modifications to SST solver seem to have improved robustness on non-orthogonal meshes

Future work

- Thorough investigation of effects of adaptation parameters
  - Norm
  - Gradation
- Implement error estimates for other models
  - Spalart-Allmaras
  - Incompressible
  - Wall functions
- CAD-based projection
- Mesh-adaptive shape optimization

Acknowledgements

- Victorien Menier
- Loïc Frazza
- Walter Maier
- Jayant Mukhopadhaya
- Marco Fossati
- SU2 Development Team