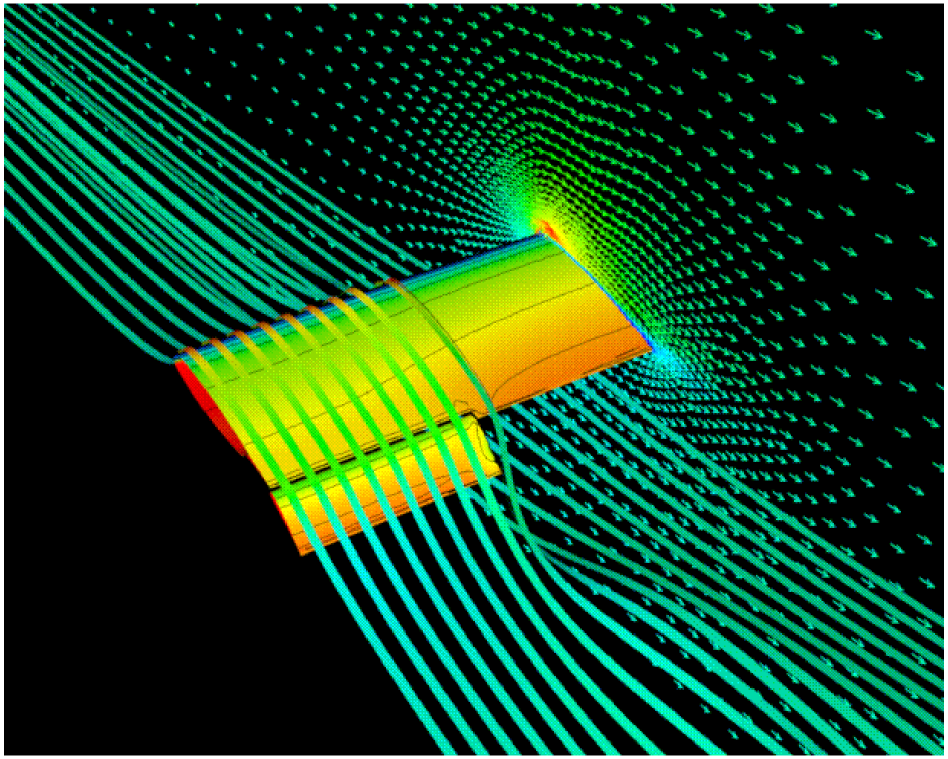
Aerodynamic Design Optimization Using the Navier-Stokes Equations



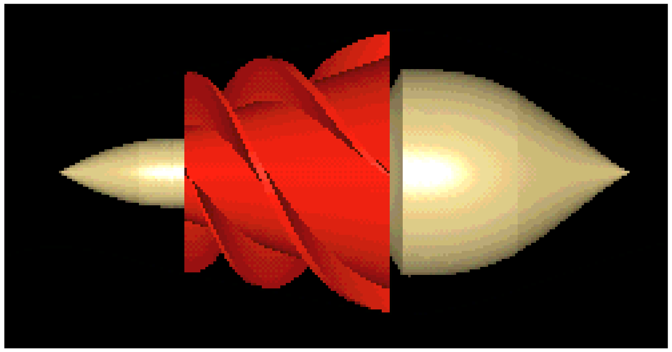
Eric J. Nielsen
Eric.J.Nielsen@nasa.gov
http://fun3d.larc.nasa.gov

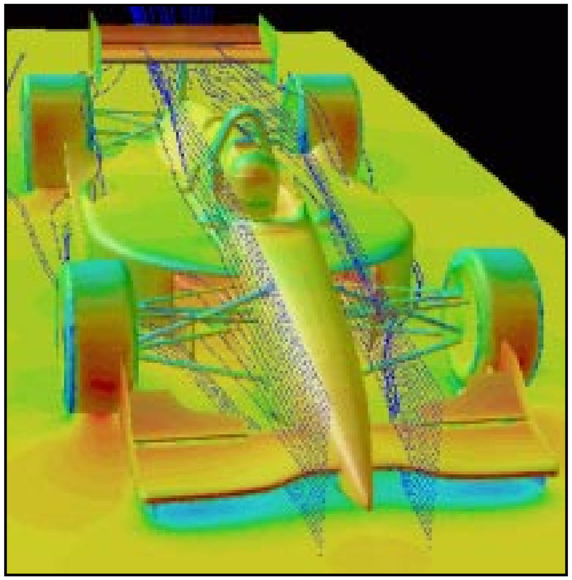
Computational Modeling and Simulation Branch
NASA Langley Research Center
Hampton, Virginia USA

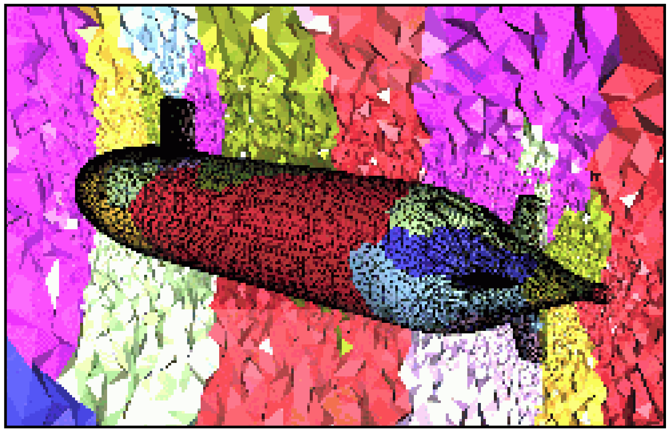
Motivation

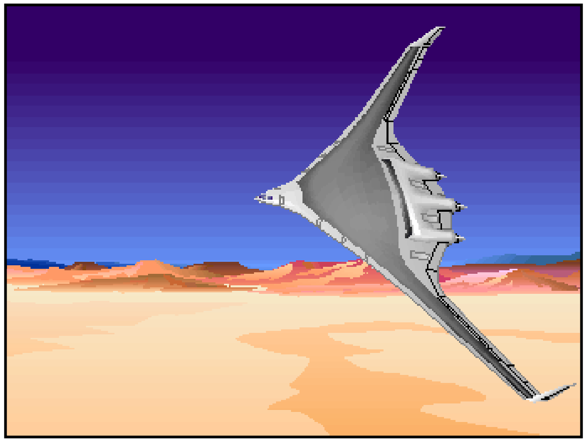
To enable rapid, high-fidelity design optimization in the early stages of the design cycle and to discover new aerodynamic concepts

- Complex flow physics
- Interaction through the flowfield
- Global cost functions
- Prior knowledge of the flowfield not required
- Design of any vehicle where fluid mechanics is important
- Design of experiments



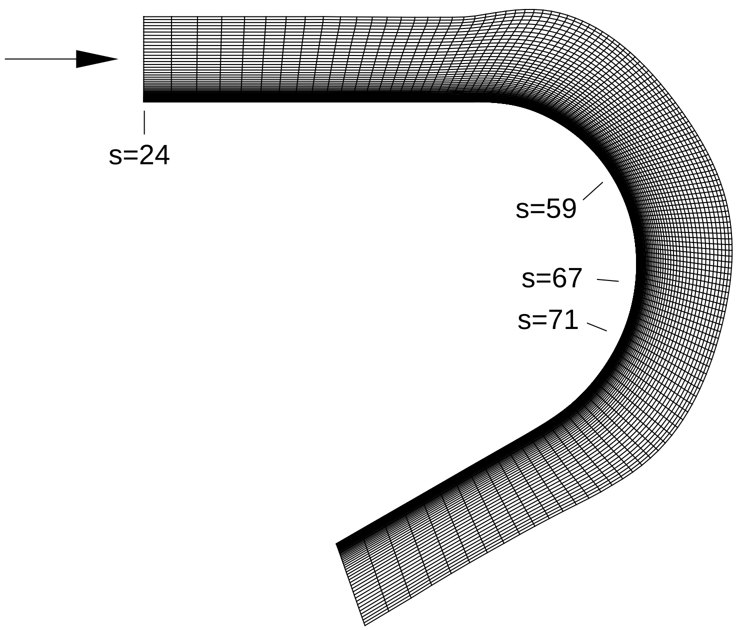




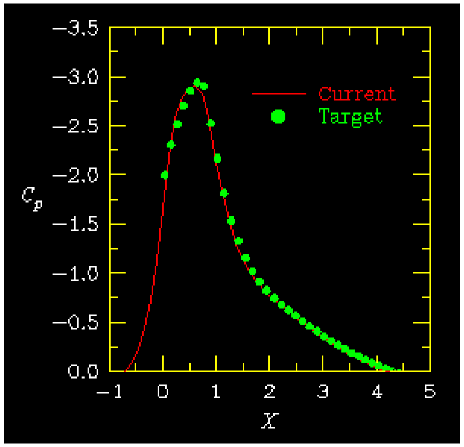


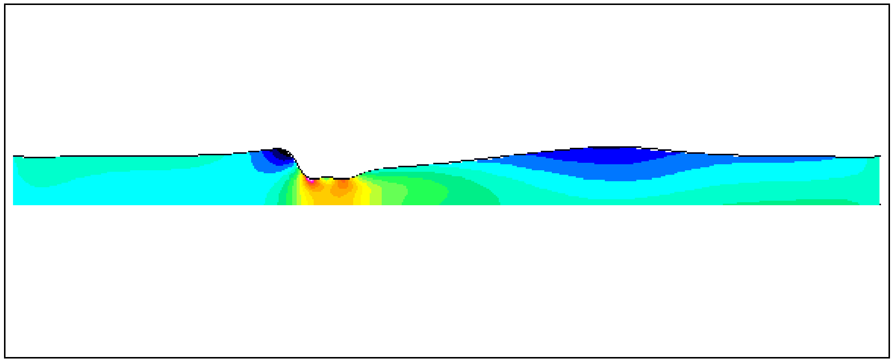
Design One Tunnel Wall to Give a Pressure Distribution on Another

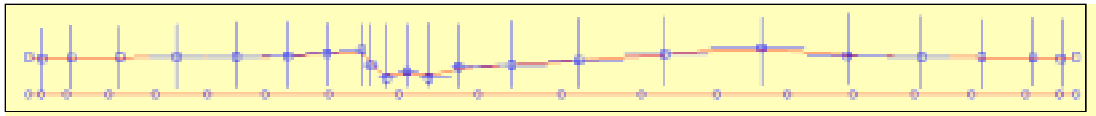
Design Constant Pressure Inner Wall for Turbulence Modeling Study



Design Upper Wall to Give "Stratford-like" Distribution on Lower Wall







The Aerodynamic Optimization Problem

$$min \;\; f(x, u(x))$$

$$\text{s.t.} \quad C_E(x, u(x)) = 0, \quad C_I(x, u(x)) \le 0, \quad x_L \le x \le x_U$$

where given x, $u(x)$ is computed via $A(x, u(x)) = 0$

- Often assumed that f, C_E, C_I, ∇ are readily available — *here this is not the case!*
- A computational aero simulation is used to obtain f, C_E, C_I, ∇:

  f may be C_L, C_D, $\dot{q}$, etc.

  C_E, C_I may be C_L, C_M, etc.

  These are typically integrated quantities of $u(x)$.

- Question of validation
  - Error bounds
  - Useful in adaptation $\longrightarrow$ decrease in expense of evaluation

The Reynolds-Averaged Navier-Stokes Equations

$$V\frac{\partial \boldsymbol{Q}}{\partial t} + \oint_{\Omega}(\vec{\boldsymbol{F}}_i \cdot \hat{\boldsymbol{n}})d\Omega - \oint_{\Omega}(\vec{\boldsymbol{F}}_v \cdot \hat{\boldsymbol{n}})d\Omega = 0 \qquad \boldsymbol{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix} \qquad \vec{\boldsymbol{F}}_i = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ (E+p)u \end{bmatrix}\hat{\boldsymbol{i}} + \begin{bmatrix} \rho v \\ \rho v u \\ \rho v^2 + p \\ \rho v w \\ (E+p)v \end{bmatrix}\hat{\boldsymbol{j}} + \begin{bmatrix} \rho w \\ \rho w u \\ \rho w v \\ \rho w^2 + p \\ (E+p)w \end{bmatrix}\hat{\boldsymbol{k}}$$

$$\vec{\boldsymbol{F}}_v = \boldsymbol{f}_v\hat{\boldsymbol{i}} + \boldsymbol{g}_v\hat{\boldsymbol{j}} + \boldsymbol{h}_v\hat{\boldsymbol{k}} \quad \boldsymbol{f}_v = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{bmatrix} \quad \boldsymbol{g}_v = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{zy} - q_y \end{bmatrix} \quad \boldsymbol{h}_v = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \end{bmatrix}$$

$$\tau_{xx} = (\mu + \mu_t)\frac{M_\infty}{Re}\frac{2}{3}[2u_x - (v_y + w_z)] \quad \tau_{yy} = (\mu + \mu_t)\frac{M_\infty}{Re}\frac{2}{3}[2v_y - (u_x + w_z)] \quad \tau_{zz} = (\mu + \mu_t)\frac{M_\infty}{Re}\frac{2}{3}[2w_z - (u_x + v_y)]$$

$$\tau_{xy} = \tau_{yx} = (\mu + \mu_t)\frac{M_\infty}{Re}(u_y + v_x) \quad \tau_{xz} = \tau_{zx} = (\mu + \mu_t)\frac{M_\infty}{Re}(u_z + w_x) \quad \tau_{yz} = \tau_{zy} = (\mu + \mu_t)\frac{M_\infty}{Re}(v_z + w_y)$$

$$q_x = \frac{-M_\infty}{Re(\gamma - 1)}\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial a^2}{\partial x} \quad q_y = \frac{-M_\infty}{Re(\gamma - 1)}\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial a^2}{\partial y} \quad q_z = \frac{-M_\infty}{Re(\gamma - 1)}\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial a^2}{\partial z}$$

Perfect Gas
Eqn of State

$$p = (\gamma - 1)\left[E - \rho\frac{(u^2 + v^2 + w^2)}{2}\right] \qquad \mu = \frac{\hat{\mu}}{\hat{\mu}_\infty} = \frac{(1 + C^*)(\hat{T}/\hat{T}_\infty)^{3/2}}{\hat{T}/\hat{T}_\infty + C^*}$$

Sutherland's Law

Functions

Turbulence Model

Spalart-Allmaras One-Equation Model

$$\frac{D\tilde{\upsilon}}{Dt} = \frac{M_\infty}{\sigma Re}\left\{\nabla \cdot [(\upsilon + (1 + c_{b_2})\tilde{\upsilon})\nabla\tilde{\upsilon}] - c_{b_2}\tilde{\upsilon}\nabla^2\tilde{\upsilon}\right\}$$

$$-\frac{M_\infty}{Re}\left(c_{w_1} f_w - \frac{c_{b_1}}{\kappa^2} f_{t_2}\right)\left(\frac{\tilde{\upsilon}}{d}\right)^2 + c_{b_1}(1 - f_{t_2})\tilde{S}\tilde{\upsilon} + \frac{Re}{M_\infty} f_{t_1}\Delta U^2$$

$$f_{v_1} = \frac{\chi^3}{\chi^3 + c_{v_1}^3}$$

$$f_{v_2} = 1 - \frac{\chi}{1 + \chi f_{v_1}}$$

$$g = r + c_{w_2}(r^6 - r)$$

$$\chi = \frac{\tilde{\upsilon}}{\upsilon}$$

$$f_w = g\left(\frac{1 + c_{w_3}^6}{g^6 + c_{w_3}}\right)^{1/6}$$

$$r = \frac{M_\infty}{Re}\frac{\tilde{\upsilon}}{\tilde{S}\kappa^2 d^2}$$

$$\tilde{S} = S + \frac{M_\infty}{Re}\frac{\tilde{\upsilon}}{\kappa^2 d^2} f_{v_2}$$

$$\mu_t = \rho\upsilon_t = \rho\tilde{\upsilon} f_{v_1}$$

Functions

Flow Solver

- Solves the governing equations using a finite-volume node-based upwind implicit solution scheme on mixed-element unstructured grids
- Highly scalable MPI implementation using domain-decomposition
- Compressible and incompressible formulations; reacting-gas chemistry option being matured
- Spalart's one-equation turbulence model integrated to the wall, solved loosely or tightly coupled
- Time-accurate options
- Multigrid with point- and line-implicit smoothers

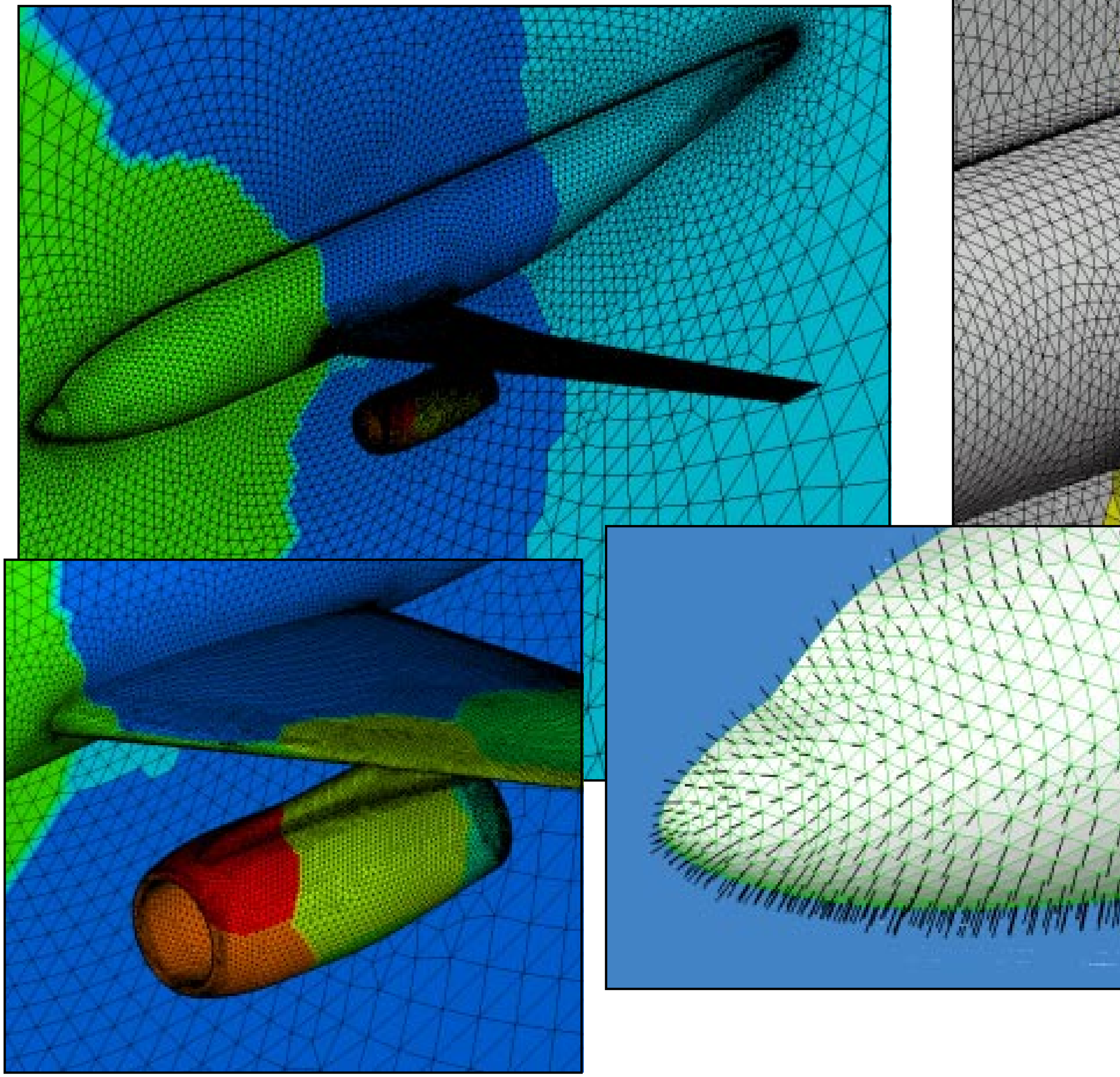
Pre-processor is ~80,000 lines of source code
Solver is ~115,000 lines of source code

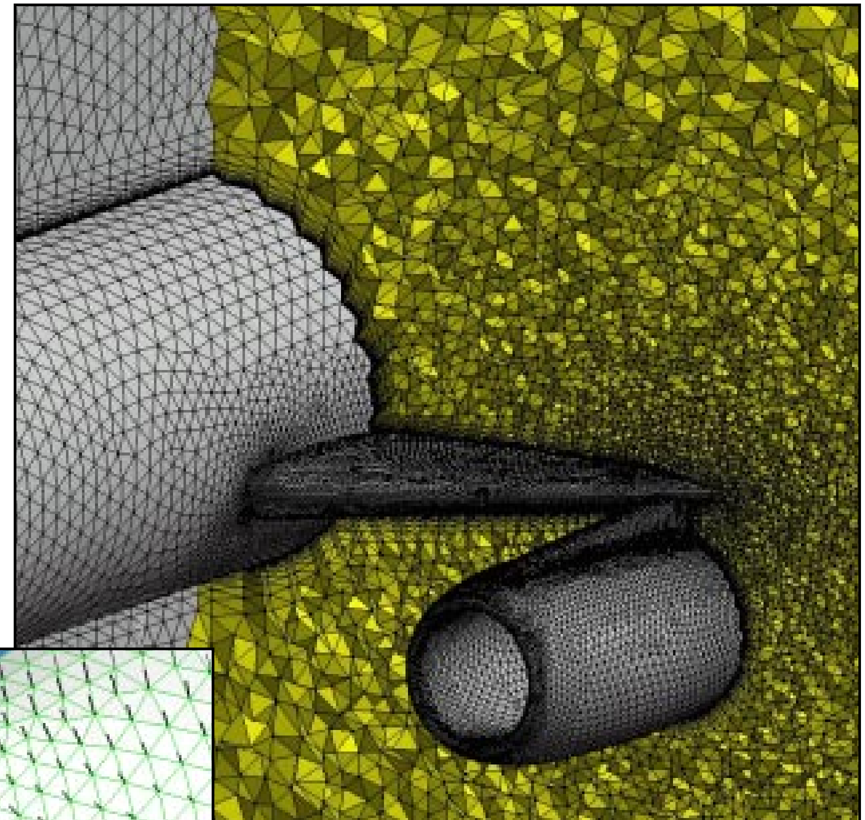
Problem Areas: Flow Solver

- Knobs always need adjusting
- Turbulence modeling

Functions

Grid Used for Typical RANS Computation





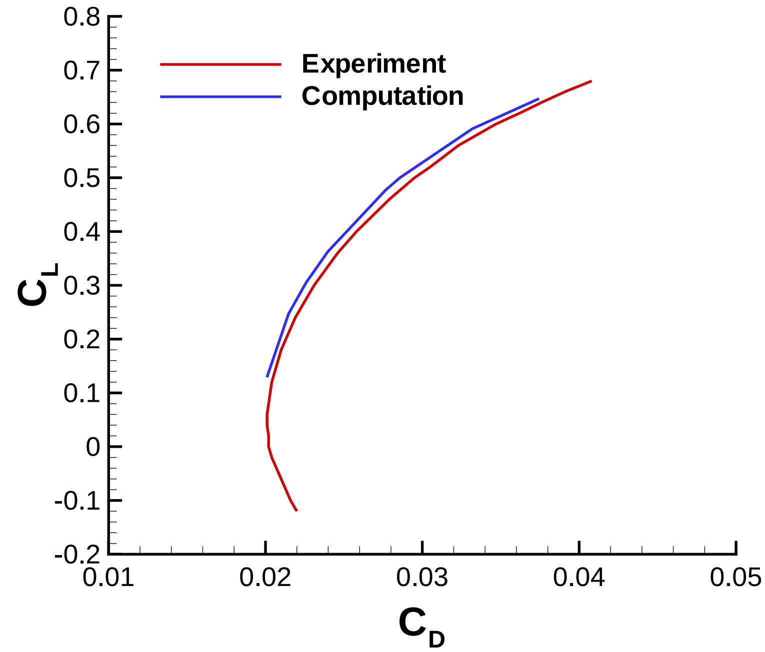
1,827,470 Nodes
10,715,204 Elements
Partitioned for 20 CPU’s

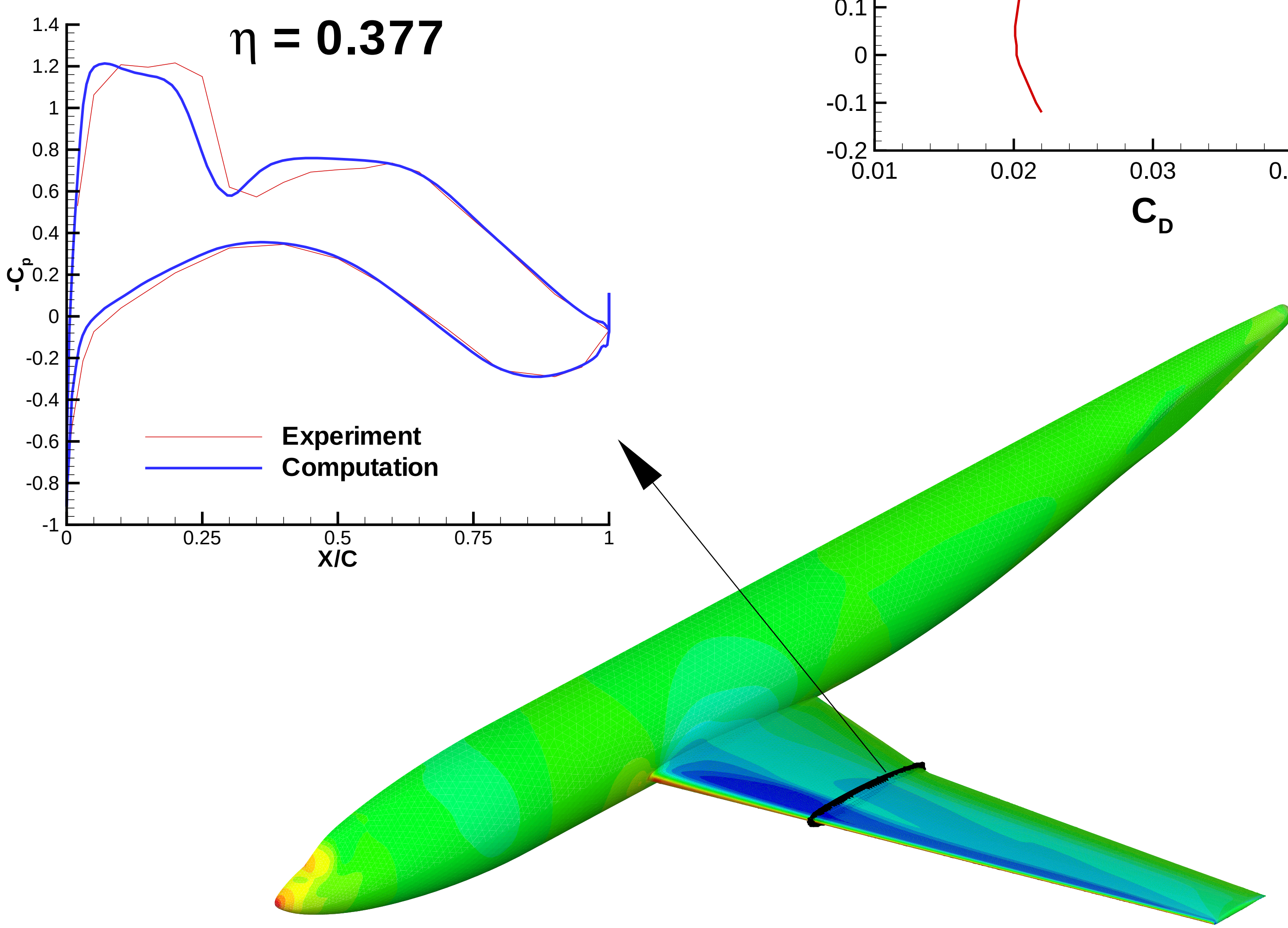
Grid Courtesy of E. Lee-Rausch and S. Pirzadeh

Results of a Typical RANS Analysis

$M_\infty = 0.75 \quad Re = 3\times10^6 \quad C_L = 0.5$

~30 Hours Wallclock Time on 20 CPU's



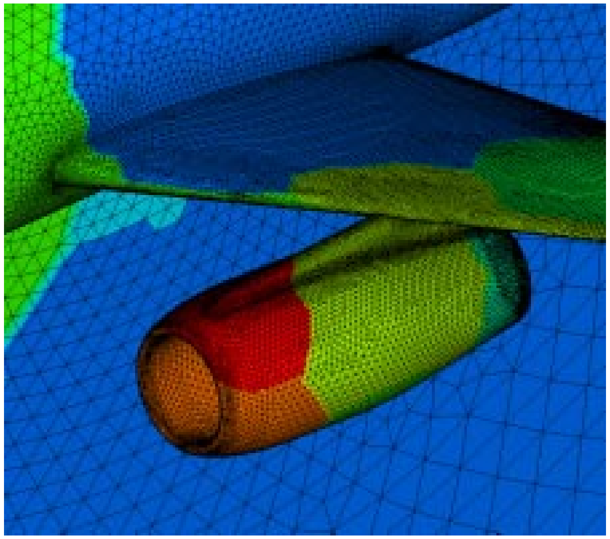


Functions

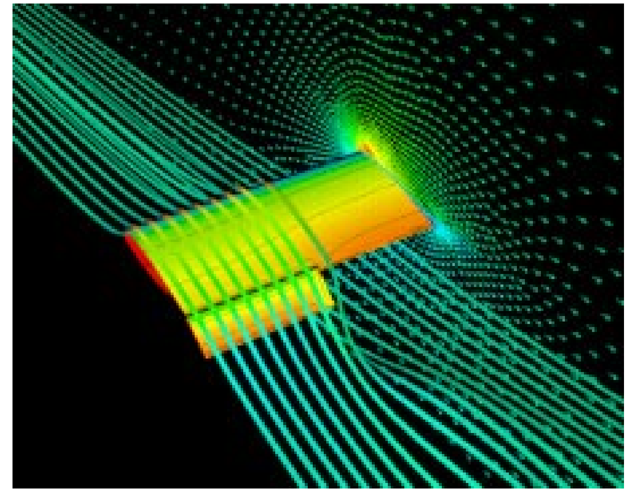
Solution Courtesy of E. Lee-Rausch

The Design Environment

Domain Decomposition



Flow Solver



Optimizer

$$min(f)$$

Parameterization

MASSOUD

(Samareh)

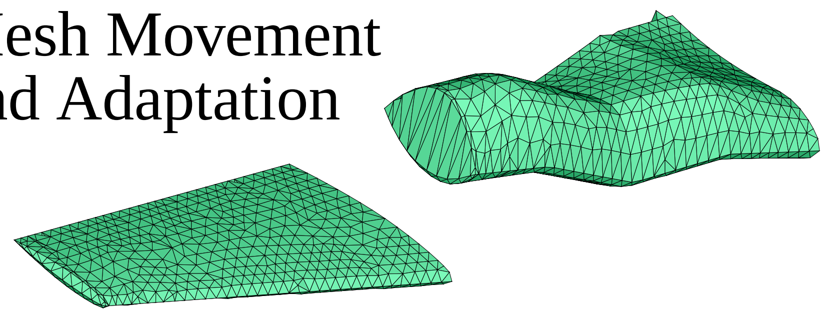
Adjoint Solver

$$\Lambda$$

Gradient Evaluation

$$\nabla f$$

Mesh Movement
and Adaptation



Obtaining Design Sensitivities: An Overview of Various Techniques

Finite-Differences

- Easy to implement
- Each variable must be perturbed independently
- Choice of step size is always an issue

Direct Differentiation

- Yields the most sensitivity information, but requires the solution of a large linear system of equations for each design variable

Complex Variables

- Yields similar information as direct approach, very little coding effort required

Adjoint Approach

- Solution of one linear system produces gradients of a cost function “independent” of the number of design variables

Differentiation Using Complex Variables

- Traditional Finite Difference

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

- Complex-Variable Approach (Lyness & Moler)

$$f(x+ih) = f(x) + ih f'(x) - \frac{h^2}{2} f''(x) - \frac{ih^3}{6} f'''(x) + \frac{h^4}{24} f^{iv}(x) + \ldots$$

$$\longrightarrow \quad f'(x) \approx \frac{Im[f(x+ih)]}{h}$$

Second-order accurate, incurs no subtractive cancellation error, and **requires hardly any coding**

- Ruby script "complexifies" current code base every night - handles all variable declarations, file I/O, MPI, operator overloading, etc.
- Resulting code is readable
- Extremely useful in tracking down hand-differentiated linearization bugs

Gradients

Adjoint and Design Equations

Define a Lagrangian function, L:

$$L(\boldsymbol{D}, \boldsymbol{Q}, \boldsymbol{X}, \Lambda) = f(\boldsymbol{D}, \boldsymbol{Q}, \boldsymbol{X}) + \Lambda^{T}\boldsymbol{R}(\boldsymbol{D}, \boldsymbol{Q}, \boldsymbol{X})$$

Now differentiate:

$$\frac{\partial L}{\partial \boldsymbol{D}} = \left\{\frac{\partial f}{\partial \boldsymbol{D}} + \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right]^{T}\frac{\partial f}{\partial \boldsymbol{X}}\right\} + \left[\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{D}}\right]^{T}\left\{\frac{\partial f}{\partial \boldsymbol{Q}} + \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{Q}}\right]^{T}\Lambda\right\} + \left\{\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}}\right]^{T} + \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right]^{T}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right]^{T}\right\}\Lambda$$

$$\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{Q}}\right]^{T}\Lambda = -\left(\frac{\partial f}{\partial \boldsymbol{Q}}\right)$$

Adjoint Equation

$$\frac{\partial L}{\partial \boldsymbol{D}} = \left\{\frac{\partial f}{\partial \boldsymbol{D}} + \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right]^{T}\frac{\partial f}{\partial \boldsymbol{X}}\right\} + \left\{\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}}\right]^{T} + \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right]^{T}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right]^{T}\right\}\Lambda$$

Sensitivity Equation

Gradients

Adaptation

Some Remarks on the Adjoint Equation

Key idea: Adjoint communicates high-fidelity physics information through the flowfield

- Requires complete linearizations of discrete residual wrt dependent variables (performed by hand in current work)
- Ignoring pieces of the linearizations is dangerous
- Careful construction of a time-marching algorithm based on the flow solver yields identical asymptotic convergence rates
- By including the ability to handle multiple RHS's, some of the overhead associated with multiple cost functions/constraints can be mitigated
- Adjoints provide a mathematically rigorous approach to error estimation and grid adaptation

Problem Areas: Adjoint Solver

- If nonlinear problem is unsteady, little hope for linearized version
- Can use stronger solvers, but some (most) problems are just unsteady

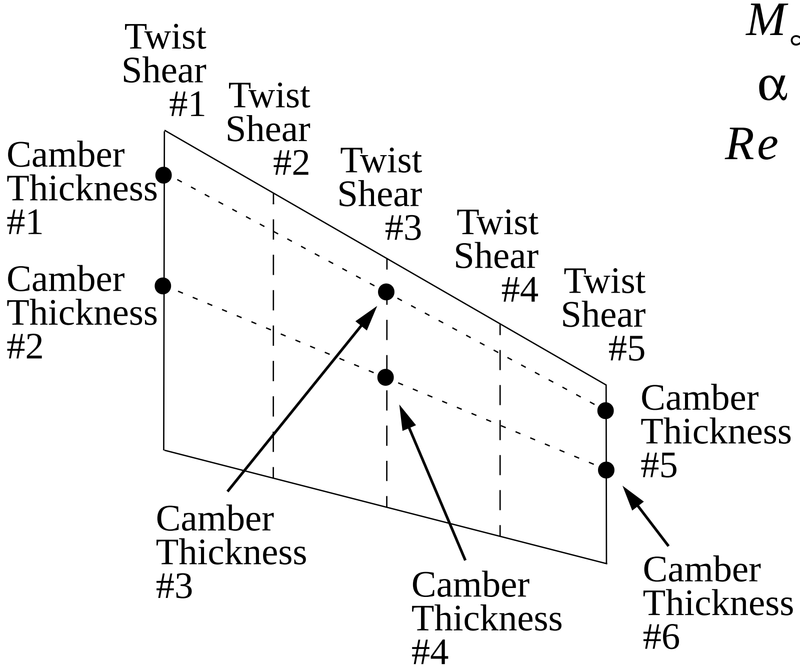
Gradients

Adaptation

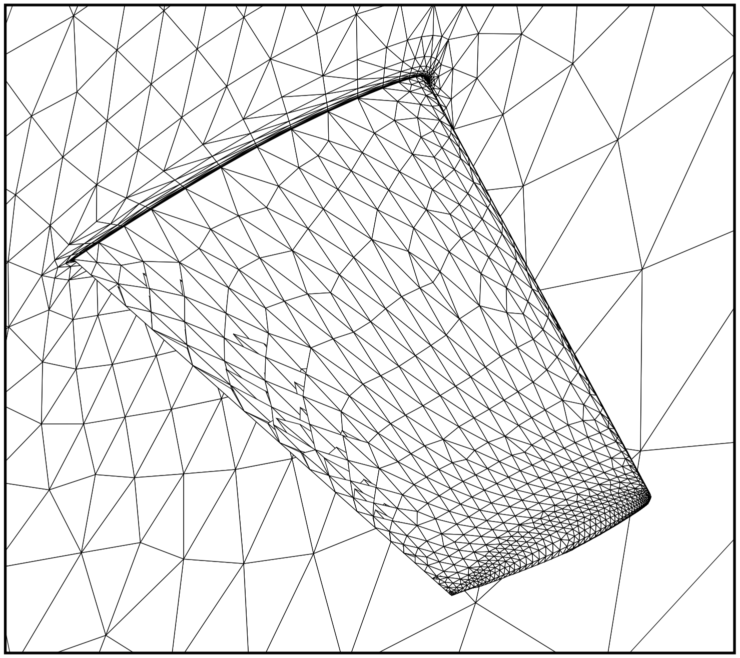
Consistency of Linearization

Three-Dimensional Turbulent Flow

		Camber	Thickness	Twist	Shear
C_L	Adjoint	0.956208938269467	-0.384940321071468	-0.010625997076936	-0.005505627646872
	Complex	0.956208938269046	-0.384940321071742	-0.010625997076937	-0.005505627647001
C_D	Adjoint	0.027595818243822	0.035539494383655	-0.000939653505699	-0.000389373578383
	Complex	0.027595818243811	0.035539494383619	-0.000939653505699	-0.000389373578412



$$M_\infty = 0.84$$
$$\alpha = 3.06°$$
$$Re = 5 \times 10^6$$



Gradients

Adjoint Methods for Error Estimation and Grid Adaptation

Traditional grid adaptation relies on solution gradients.
But what if the feature (e.g., shock) is in the wrong place to begin with?

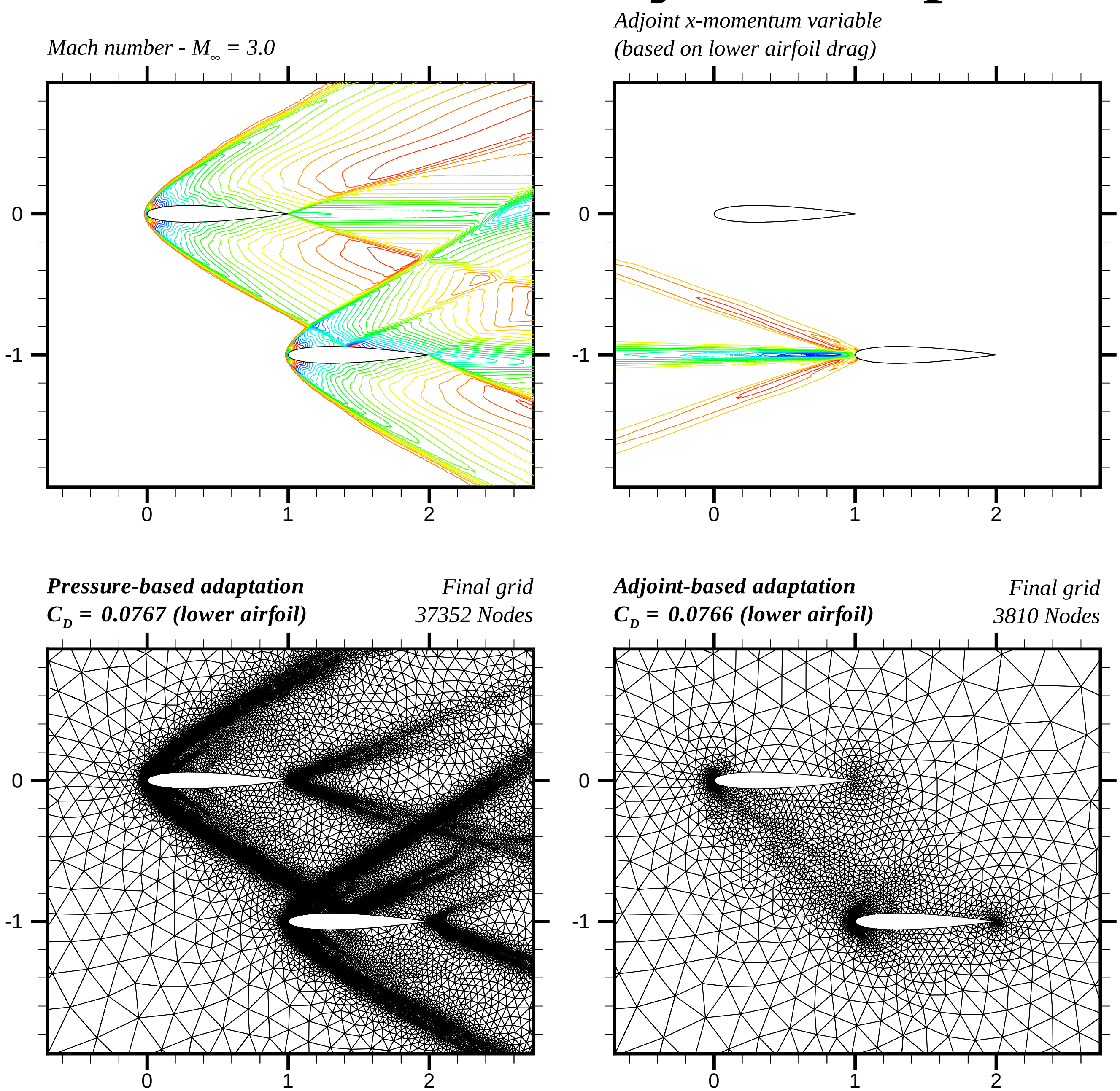
Adjoints avoid this problem and can be used to "tune" grids to accurately predict a given engineering quantity, such as lift or drag.

This can dramatically reduce the number of mesh points required for a given application, ***and produce the correct answer.***

Problem Areas: Grid Adaptation

- Highly anisotropic 3D adaptation mechanics need to be developed

Traditional Versus Adjoint Adaptation



Adaptation

Courtesy of D. Venditti and D. Darmofal, MIT

Sensitivity Evaluation Using Adjoint Variables

$$\frac{\partial L}{\partial \boldsymbol{D}} = \left\{\frac{\partial f}{\partial \boldsymbol{D}} + \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right]^T \frac{\partial f}{\partial \boldsymbol{X}}\right\} + \left\{\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{D}}\right]^T + \left[\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{D}}\right]^T \left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right]^T\right\}\Lambda$$

- Requires complete linearizations of
  - Discrete residual wrt grid (distance function, etc)
  - Surface parameterization wrt design variables
  - Mesh movement scheme wrt design variables
- Ignoring pieces of the linearizations is dangerous

Evaluating this expression is expensive and
is ***not*** independent of the number of design variables

Problem Areas: Gradient Evaluation

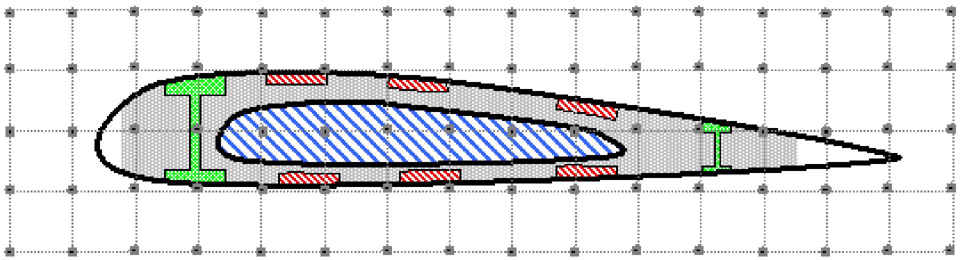
- Mesh sensitivities are extremely expensive - can an adjoint problem be formulated?

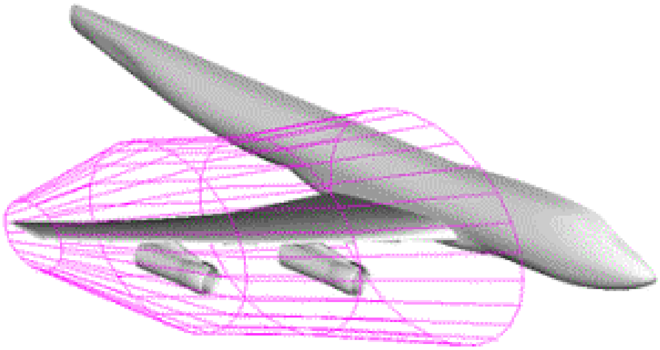
Gradients

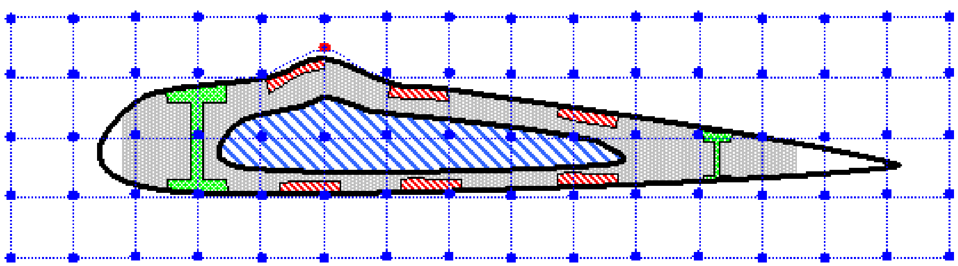
Geometric Parameterization Using MASSOUD

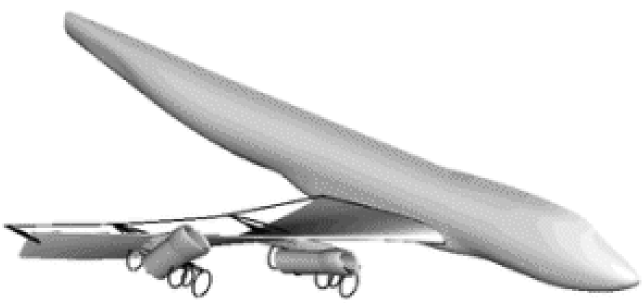
(Jamshid Samareh, NASA Langley)

- Parameterizes the changes in shape, not the shape itself - reduces the number of design variables
- Parameterizes the discipline grids - avoids manual grid regeneration
- Uses advanced soft object animation algorithms for deforming grids
- Analytic sensitivities available









Problem Areas: Parameterization

- What are good design variables to use?
- What design variables are going to cause mesh movement problems?
- How to pick bounds?

Functions

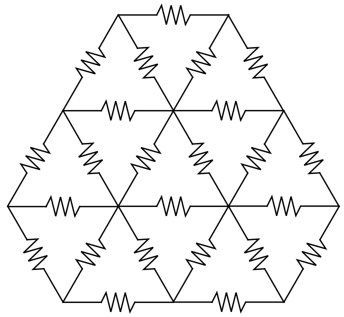
Gradients

Functions

Gradients

Mesh Movement Techniques

Spring Analogy



$$\sum_{j \in N_i} K_{ij}(\Delta \boldsymbol{x}_i - \Delta \boldsymbol{x}_j) = 0$$

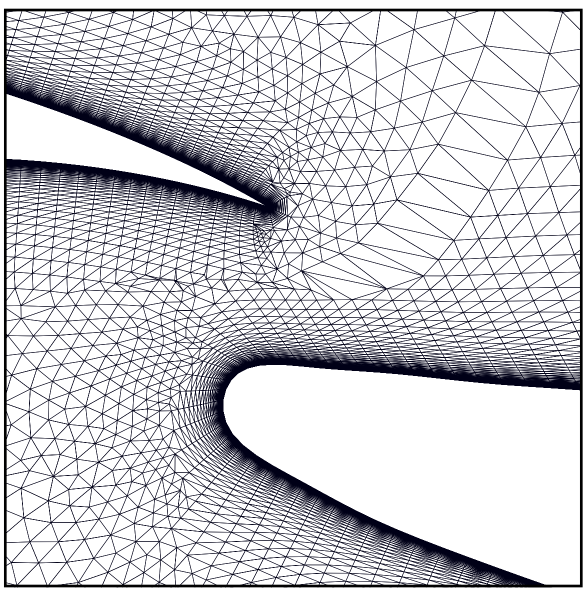
Linear Elasticity

$$\nabla^2 u + \frac{1}{1-2\nu}\frac{\partial}{\partial x}\nabla \cdot \vec{V} = 0$$

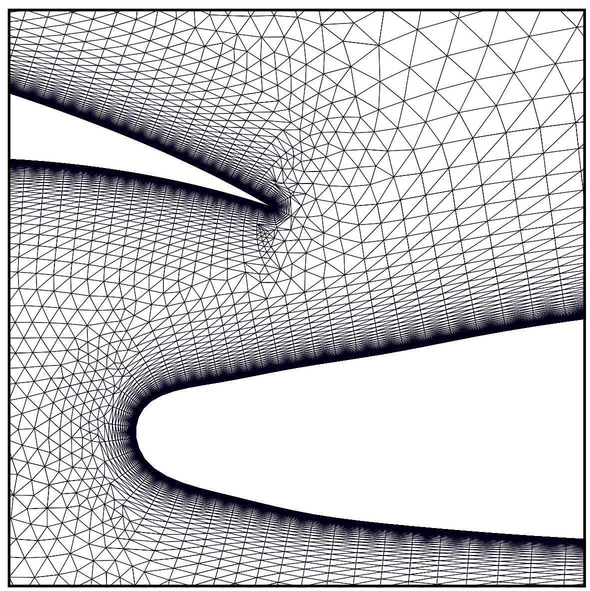
$$\nabla^2 v + \frac{1}{1-2\nu}\frac{\partial}{\partial y}\nabla \cdot \vec{V} = 0$$

$$\text{Set } \frac{1}{1-2\nu} = \text{aspect ratio}$$

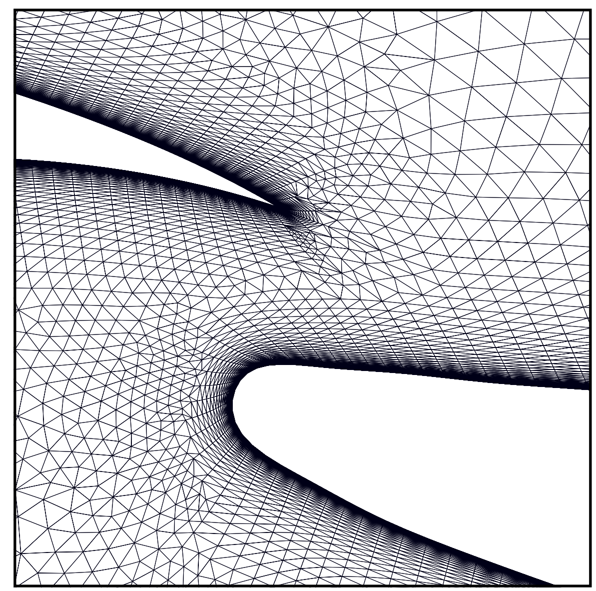
Distance/Springs



Original Mesh

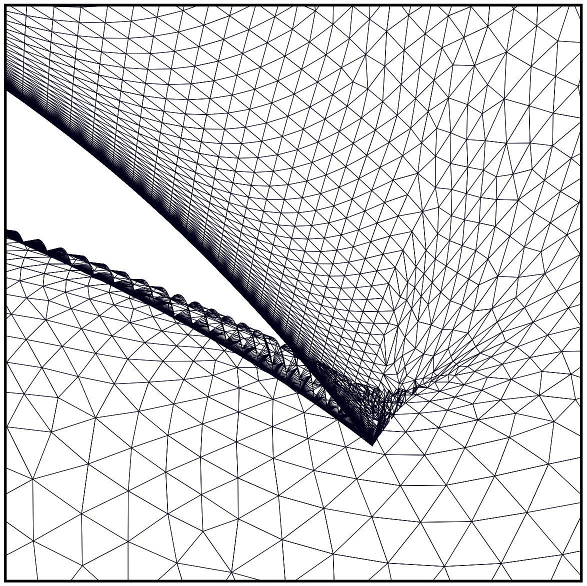


Linear Elasticity

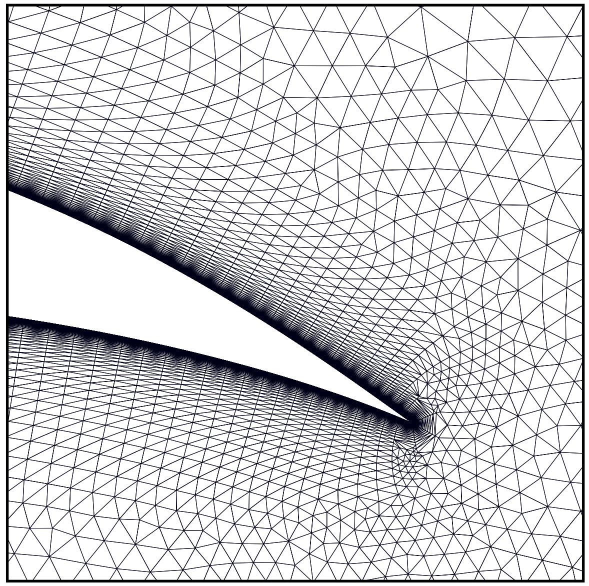


Mesh Movement Techniques

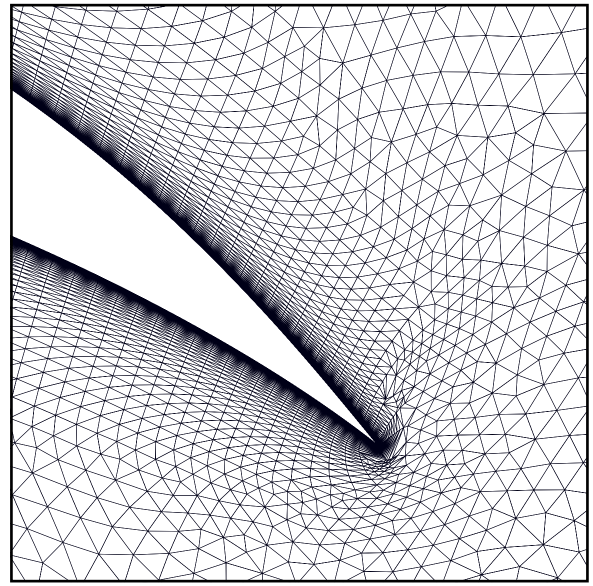
Distance/Springs



Original Mesh



Linear Elasticity



Problem Areas: Mesh Movement

- Extremely difficult to move realistic 3D grids - showstopper!
- High quality grids do not grow on trees - take what you can get
- Irreversible process causes problems for optimizer
- Grid regeneration introduces noise (even differentiable?)

Functions

Gradients

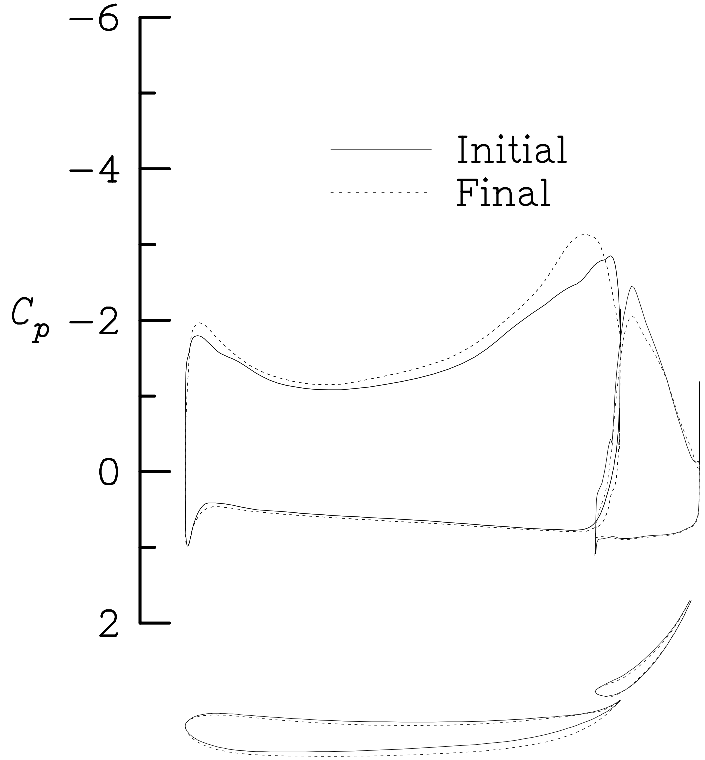
Design Example: Multielement Airfoil for Open-Wheel Racing Car

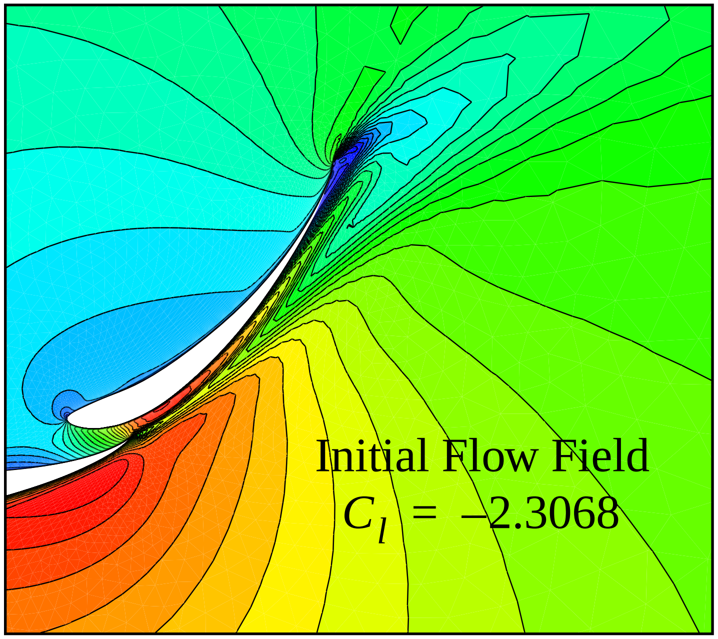
Goal: Increase the downforce

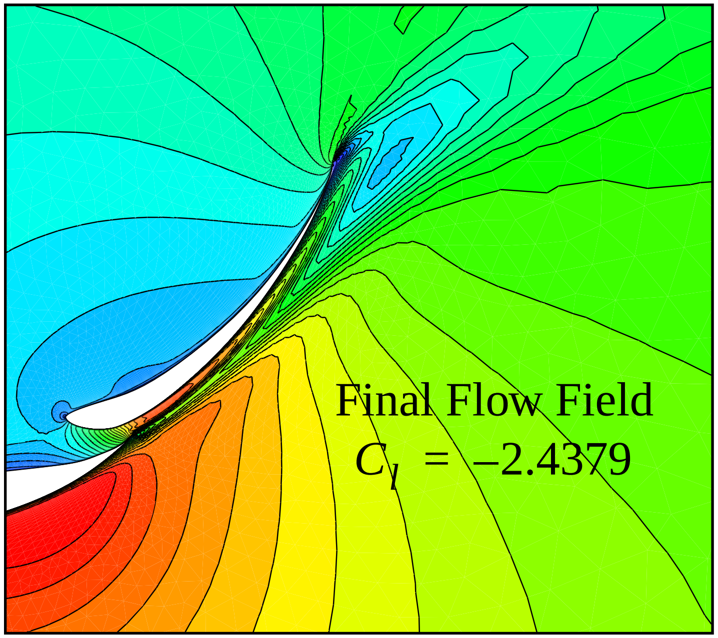
65 Design Variables, 25 Design Cycles

15,446 Nodes

$\alpha = 12°$ $Re = 2.4 \times 10^6$



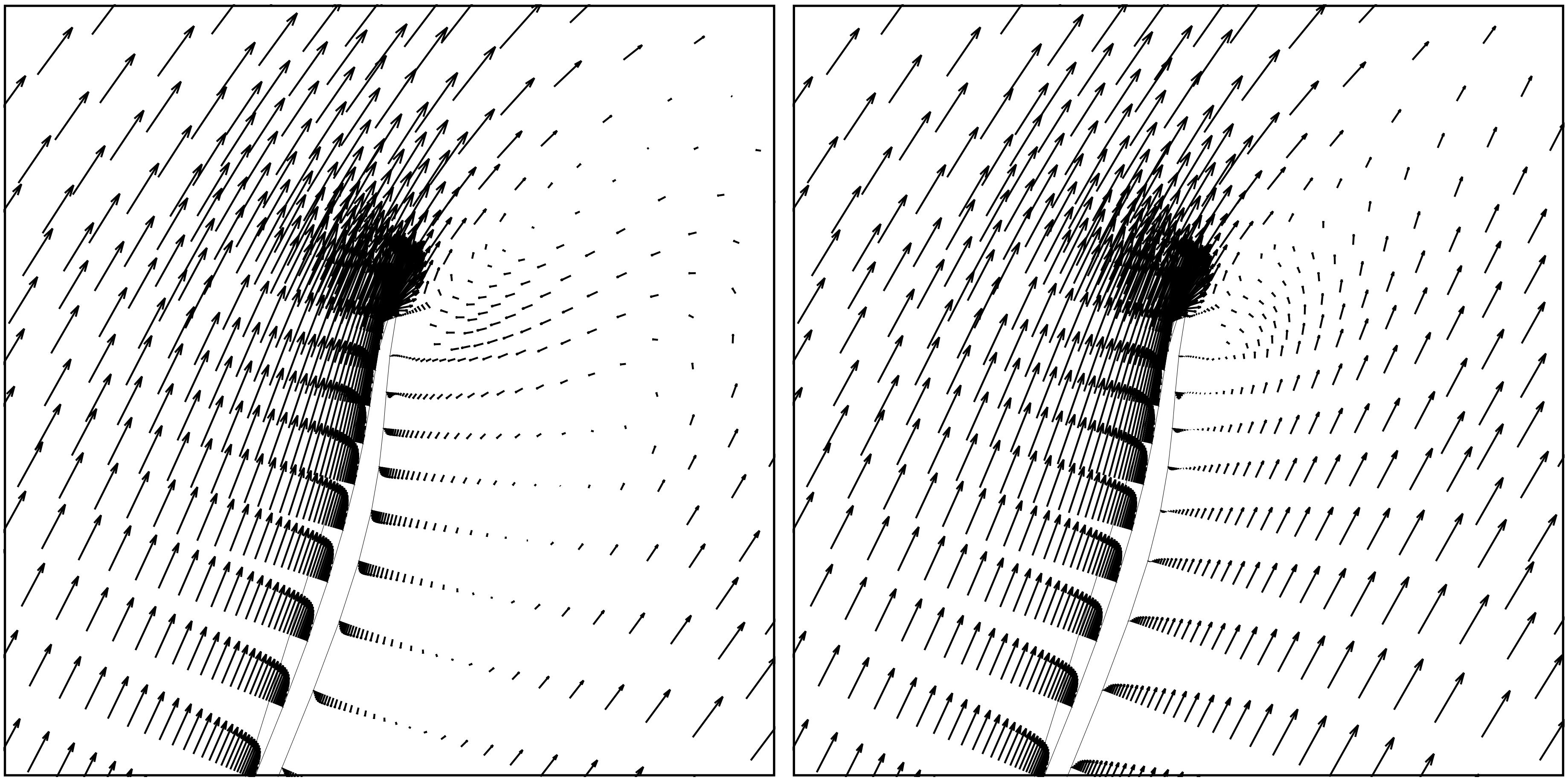




Multielement Airfoil for Open-Wheel Racing Car

Close-up of Velocity Vectors at Flap Trailing Edge

Initial Flow Field   Final Flow Field

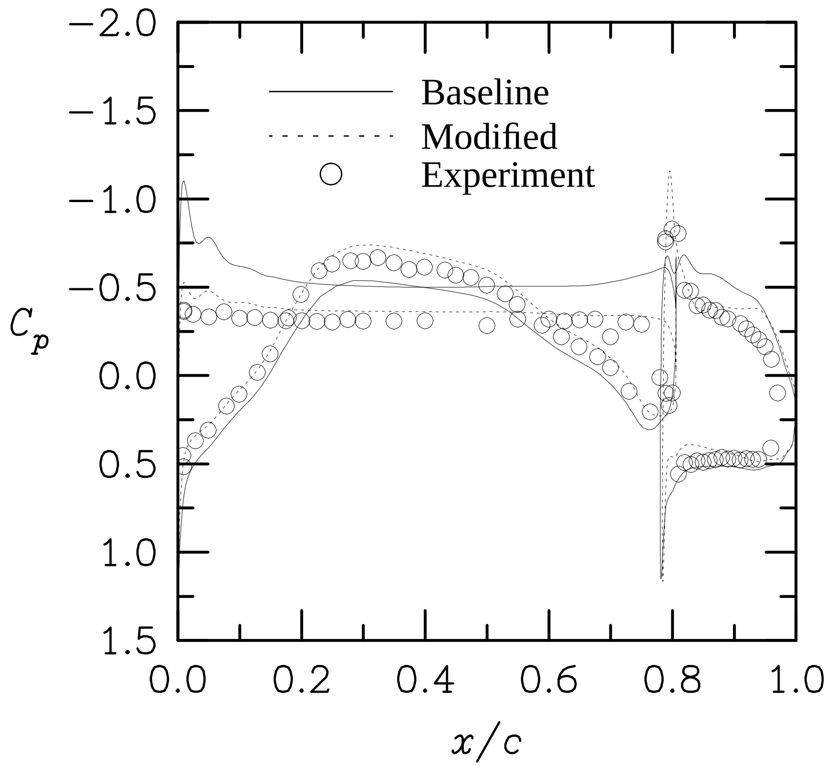


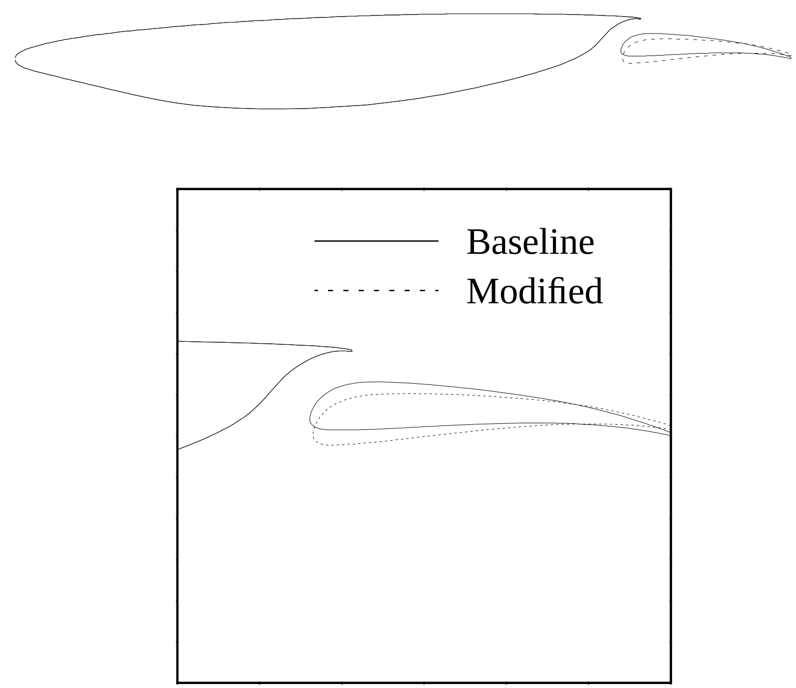
Application of Mesh Movement to a Multielement Airfoil Configuration

$$M_\infty = 0.7,\ \alpha = 1.5^\circ,\ Re = 30 \times 10^6$$

Experiment had non-uniform gap/overlap across span and deflected at high dynamic pressures

Objective: Determine flap position based on experimental pressures





Large Scale Design Case

Turbulent Flow Over Slotted Cruise Configuration

Goal: Reduce drag while maintaining lift

$M_\infty = 0.75 \qquad Re = 6.2 \times 10^6$

843,385 Nodes

4,796,360 Cells

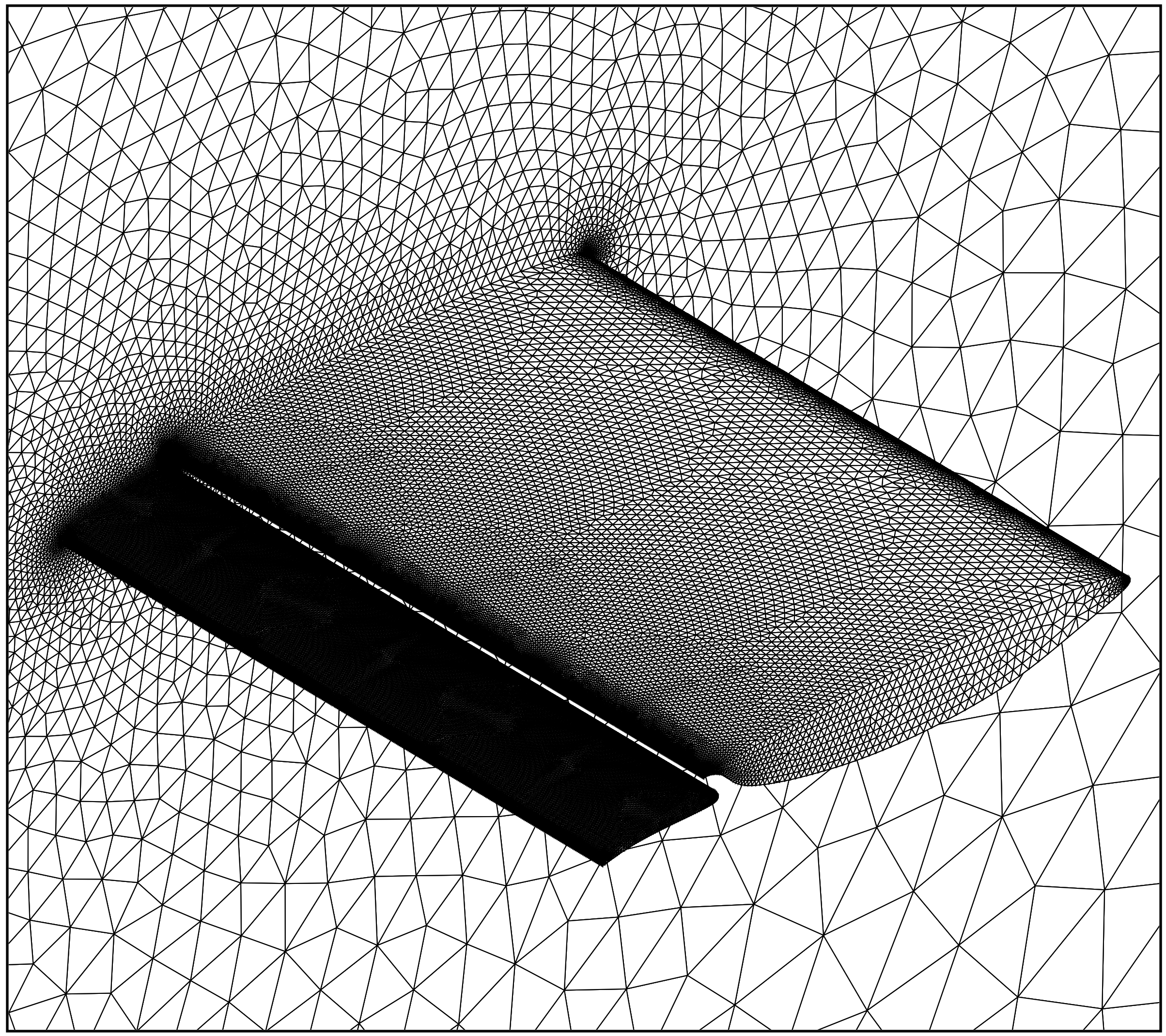
34 Design Variables

- Angle of attack
- 15 Camber values on each element
- Flap translation and rotation

5 Design cycles/16 CPU's

~6 Days wall clock time

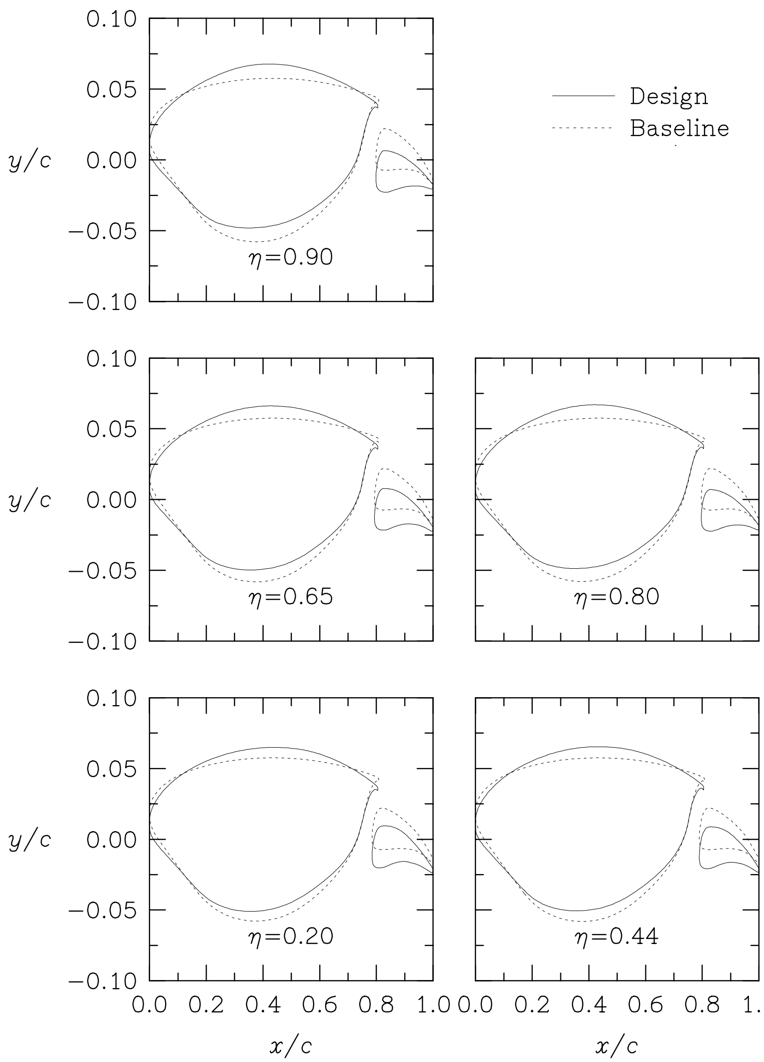
12 GB memory required

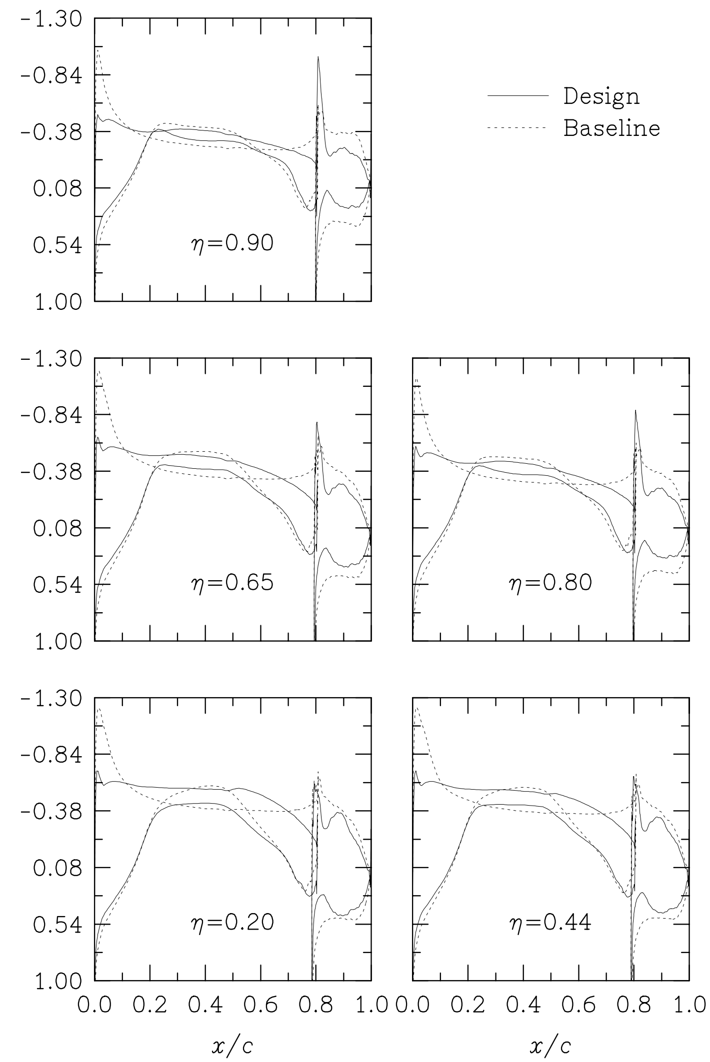


Large Scale Design Case

Turbulent Flow Over Slotted Cruise Configuration

Baseline Geometry	Modified Geometry
$\alpha = 2.81°$	$\alpha = 3.30°$
$C_L = 0.4375$	$C_L = 0.4374$
$C_D = 0.0399$	$C_D = 0.0378$





Typical Cost of a Single Design Cycle

Assuming 1,000,000 Mesh Points on 20 CPU's

Flow Solve	10 Hours
Adjoint Solve	8 Hours
20-30 Grid Sensitivities and Gradient Evaluations	10 Hours
Line Search with 5-6 Grid Moves and Flow Solves	20 Hours
Single Design Cycle	**~2 Days Wallclock Time**

Now consider:

- ~10 Design cycles in a given run
- Robust (i.e., multipoint) design
- Unsteady flows
- Reacting gas chemistry
- MDO

Future Directions / Possible Solutions

Flow and Adjoint Solutions

- Adaptive, “smart” algorithms
- Simultaneous convergence/error estimation monitoring
- Adjoint-based grid adaptation

Mesh Movement

- High quality initial grids
- Mesh untangling (Freitag)
- Quaternion-based approaches (Samareh)
- Adjoint-based sensitivities

Grid Adaptation

- Algorithms for highly anisotropic meshes

Optimization

- Model management (Alexandrov)
- Parallel algorithms

Hardware

- Massively parallel computing