Unstructured Grid Technology for CFD: Flow Solver Perspectives

Dimitri Mavriplis

University of Wyoming

Overview

- Unstructured mesh technology has come of age (finally)
  - Evidence from Drag Prediction Workshops
- Efficient solvers required for steady-state and time-implicit problems
  - (similar to structured meshes)
- Grid resolution and convergence
- AMR
- Sensitivity Analysis
- Higher-Order Methods

Unstructured Mesh Technology

- Current evidence suggests
  - USG accuracy comparable to structured grid accuracy on grid of similar resolution
    - Current best practice:
      - Prismatic elements in boundary layer regions
      - Tetrahedral elements in inviscid regions
      - Shock alignment for hypersonics may require hexahedral elements (Candler, Gnoffo)
      - Cell vs. Node schemes
        - » Tradeoffs still not fully understood

Unstructured Mesh Technology

- Current evidence suggests
  - USG efficiency similar to structured meshes
    - Factor 2 to 3 more costly per iteration
    - Better scalability: Homogeneous data-structures
    - Assuming similarly capable solution strategies
      - Implicit, Multigrid
      - Becomes more critical for finer meshes
        - » Steady-state
        - » Low reduced frequency problems (aeroelastics)

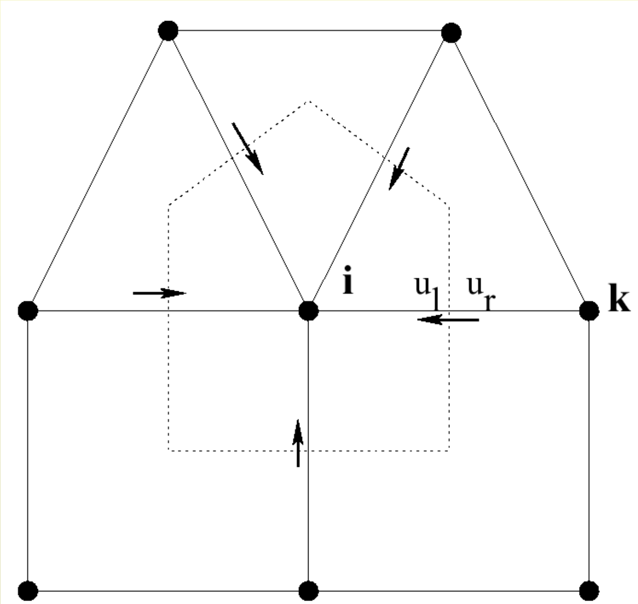
NSU3D Solver

- Governing Equations: Reynolds Averaged Navier-Stokes Equations
  - Conservation of Mass, Momentum and Energy
  - Turbulence models
    - Spalart-Allmaras 1 equation model
    - K-omega 2 equation model
- Vertex-Based Discretization
  - 2^{nd} order upwind finite-volume scheme
  - 6 or 7 variables per grid point
  - Flow equations fully coupled (5x5)
  - Turbulence equation uncoupled (1 or 2X2)

Spatial Discretization

- Mixed Element Meshes
  - Tetrahedra, Prisms, Pyramids, Hexahedra
- Control Volume Based on Median Duals
  - Fluxes based on edges

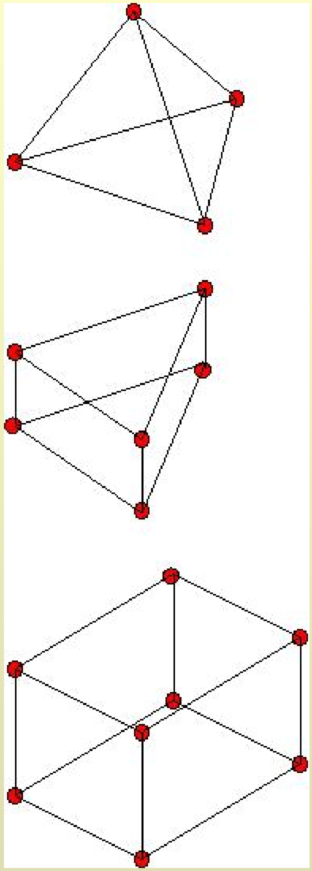
* $\mathbf{F}_{\mathrm{ik}} = f(\mathbf{u}_{\mathrm{left}}, \mathbf{u}_{\mathrm{right}})$
* $\mathbf{u}_{\mathrm{left}} = \mathbf{u}_{\mathrm{i}}, \mathbf{u}_{\mathrm{right}} = \mathbf{u}_{\mathrm{k}}$: *1st order accurate*
* $\mathbf{u}_{\mathrm{left}} = \mathbf{u}_{\mathrm{i}} + \frac{1}{2}\nabla \mathbf{u}_{\mathrm{i}}.\mathbf{r}_{\mathrm{ik}}$
* $\mathbf{u}_{\mathrm{right}} = \mathbf{u}_{\mathrm{k}} + \frac{1}{2}\nabla \mathbf{u}_{\mathrm{k}}.\mathbf{r}_{\mathrm{ki}}$: *2nd order accurate*
* ∇u_i *evaluated as contour integral around CV*

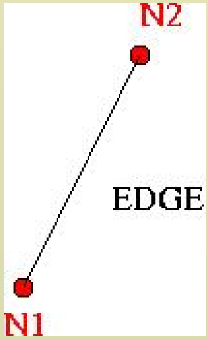


  - Various reconstruction options
    - Least squares, Green-Gauss,
    - biharmonic differences (matrix artificial dissipation)

Mixed-Element Discretizations

- Edge-based data structure
  - Building block for all element types
  - Reduces memory requirements
  - Minimizes indirect addressing / gather-scatter
  - Graph of grid = Discretization stencil
    - Implications for solvers, Partitioners





Solution Methodology

- To solve R(w) = 0 (steady or unsteady residual)
  - Newton's method: $$\left[\frac{\partial R}{\partial w}\right]\Delta w^{n+1} = -R(w^n)$$
  - Requires storage/inversion of Jacobian $\left[\frac{\partial R}{\partial w}\right]$ (too big for 2nd order scheme)
  - Replace with 1st order Jacobian
    - Stored as block Diagonals [D] (for each vertex) and off-diagonals [O] (2 for each edge)
  - Use block Jacobi or Gauss-Seidel to invert Jacobian at each Newton iteration using subiteration k:

$$[D]\Delta w^{k+1} = -R(w^n) - [O]\Delta w^k$$

Solution Methodology

- Corresponds to linear Jacobi/Gauss-Seidel in many unstructured mesh solvers

$$\left[D\right]\Delta w^{k+1} = -R(w^n) - \left[O\right]\Delta w^k$$

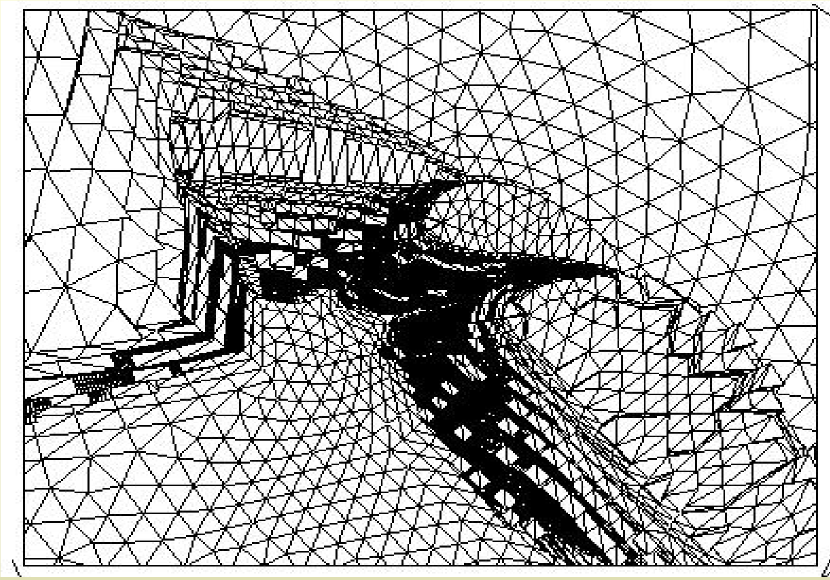
- Alternately, replace Jacobian simply by [D] (i.e. drop [O] terms) (Point implicit)

$$\left[D\right]\Delta w^{n+1} = -R(w^n)$$

  - Non-linear residual must now be updated at every iteration (no subiterations)
  - Corresponds to non-linear Jacobi/Gauss-Seidel
  - Converges at same rate as linear scheme (in index k) in the absence of strong non-linearities
    - Non-linear scheme requires ~ 150 words /cv
    - Linear schemes require ~500 words/cv : [O] ~ 350 words/cv
    - But non-linear scheme requires more cpu time because of more non-linear evaluations
  - NSU3D production code employs non-linear solver

Anisotropy Induced Stiffness

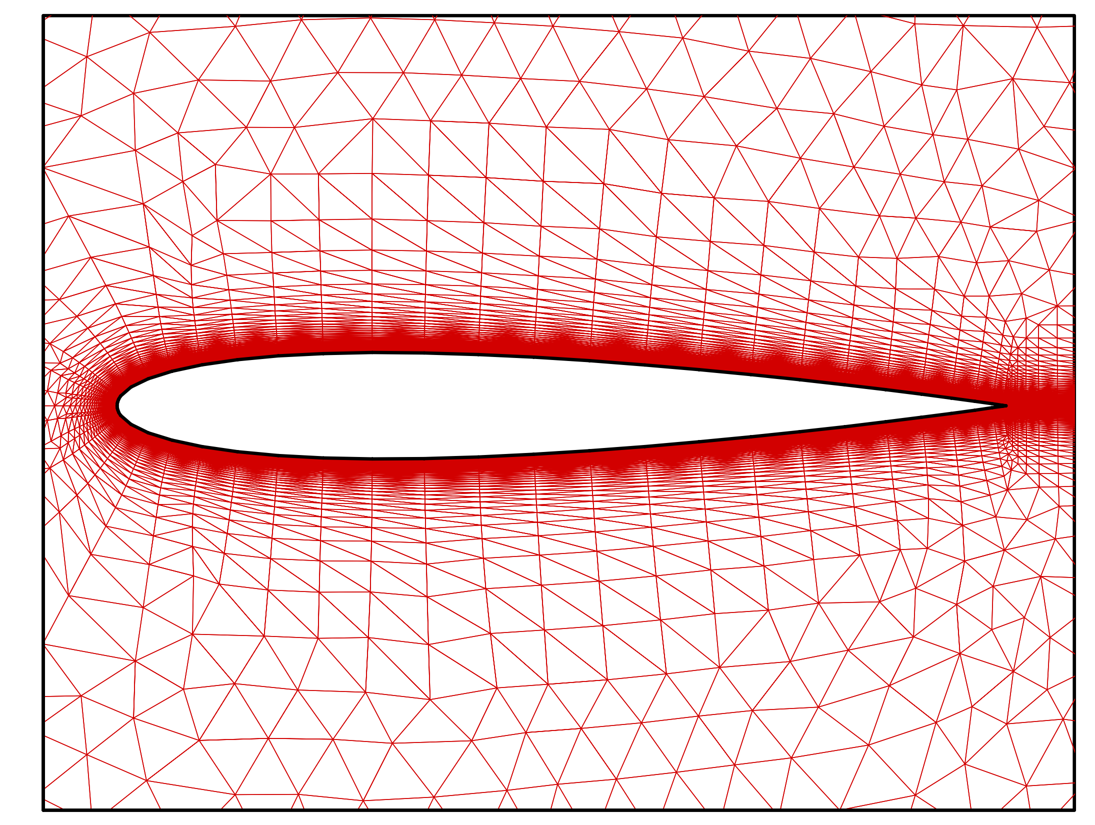
- Convergence rates for RANS (viscous) problems much slower then inviscid flows
  - Mainly due to grid stretching
  - Thin boundary and wake regions
  - Mixed element (prism-tet) grids



- Use directional solver to relieve stiffness
  - Line solver in anisotropic regions

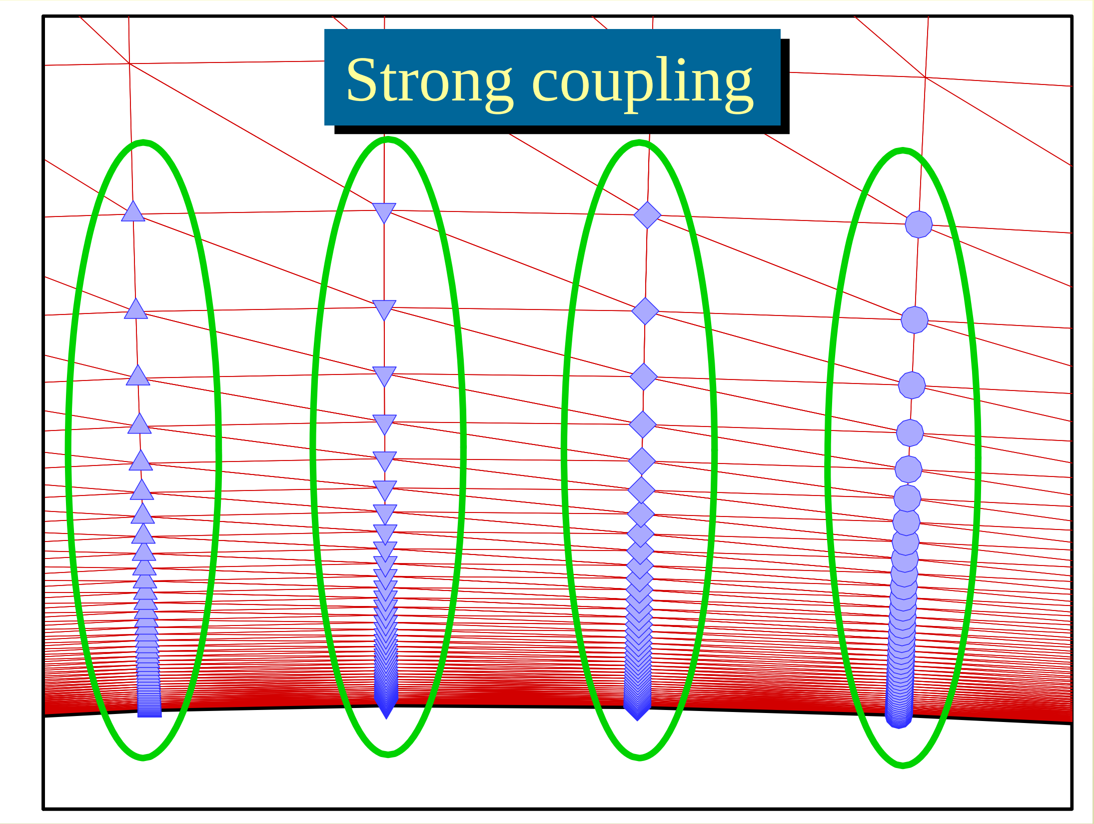
Method of Solution

- Line-implicit solver



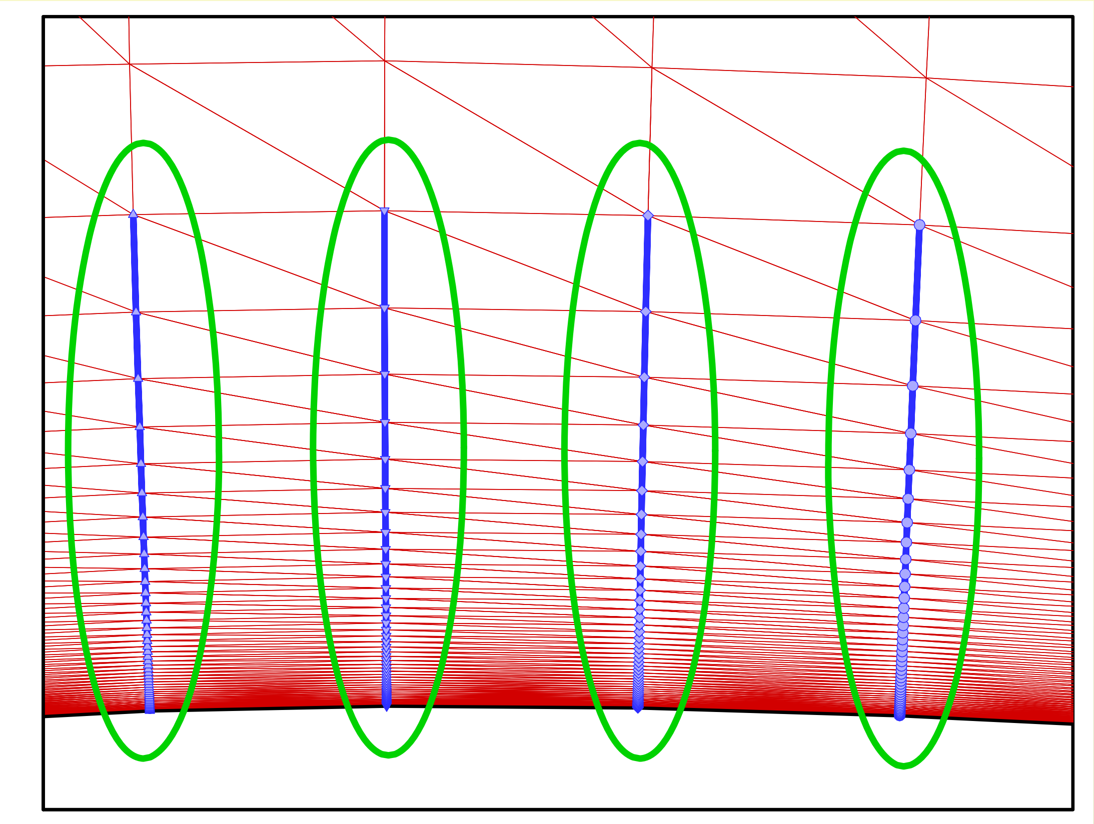
Method of Solution

- Line-implicit solver



Method of Solution

- Line-implicit solver

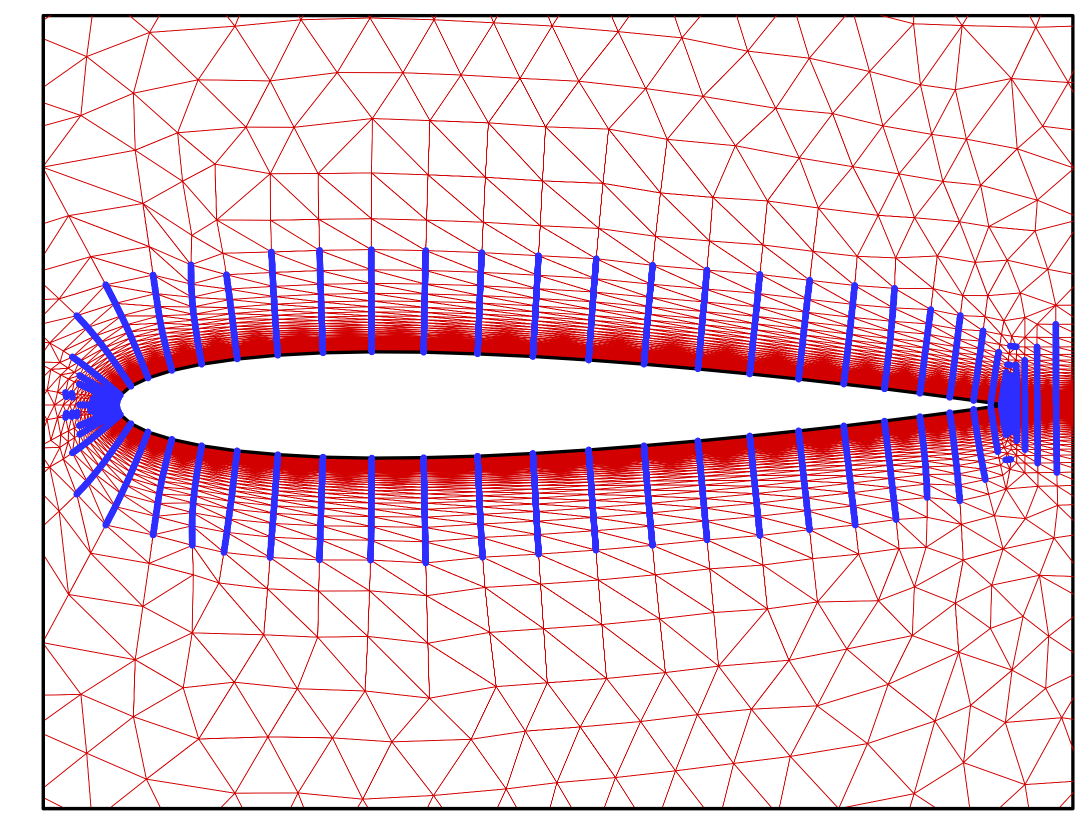


Method of Solution

- Line-implicit solver

$$[D]\Delta w^{n+1} = -R(w^n)$$

  - [D] now represents point Jacobians plus off-diagonals corresponding to edges in lines (~25 extra words/cv)**
  - Invert [D] using block tridiagonal algorithm**
  - Analogous approach possible for linear sub-iterative solver
  - Gauss-Seidel extensions
  - **: Non-linear line Jacobi used in production code

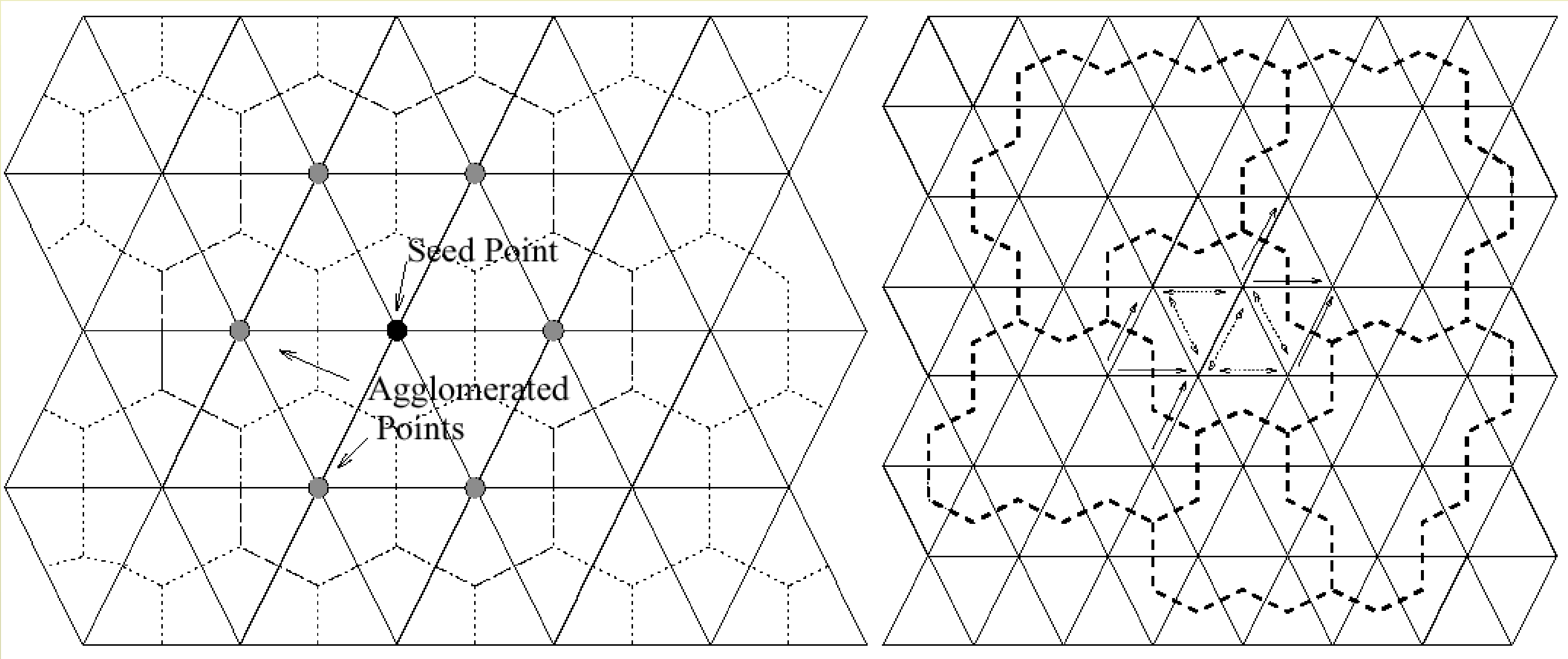


Method of Solution

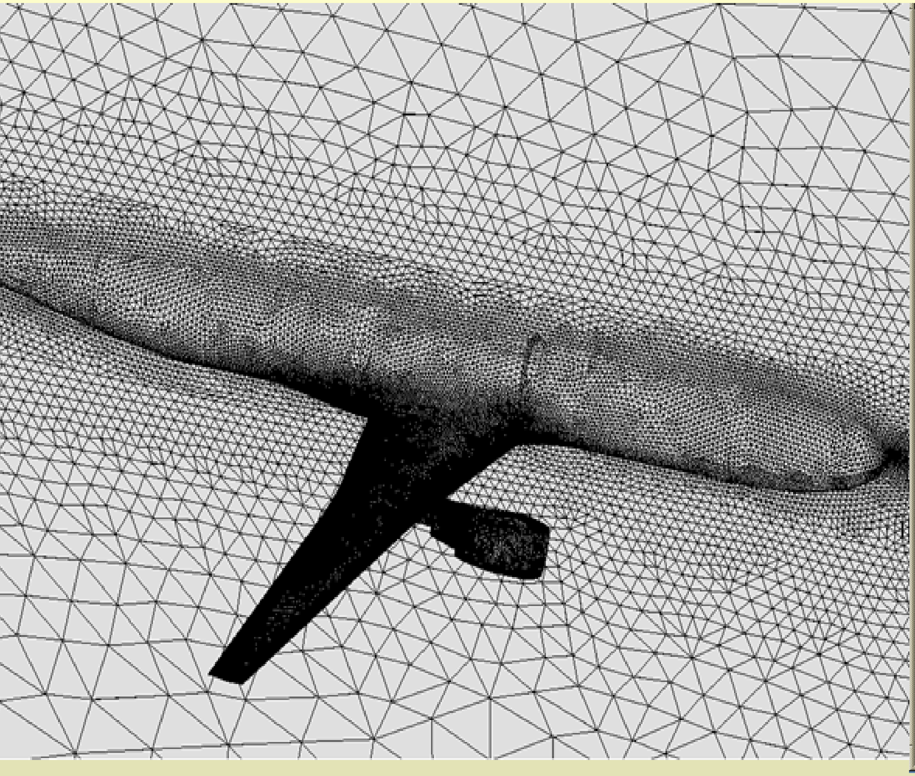
- Line Jacobi/Gauss-Seidel solver effective for time-dependent problems with small time step
  - Convergence rates independent of grid anisotropy
- For large time steps or steady-state problems require more global solver
- Use line solver as driver on each level of agglomeration multigrid algorithm

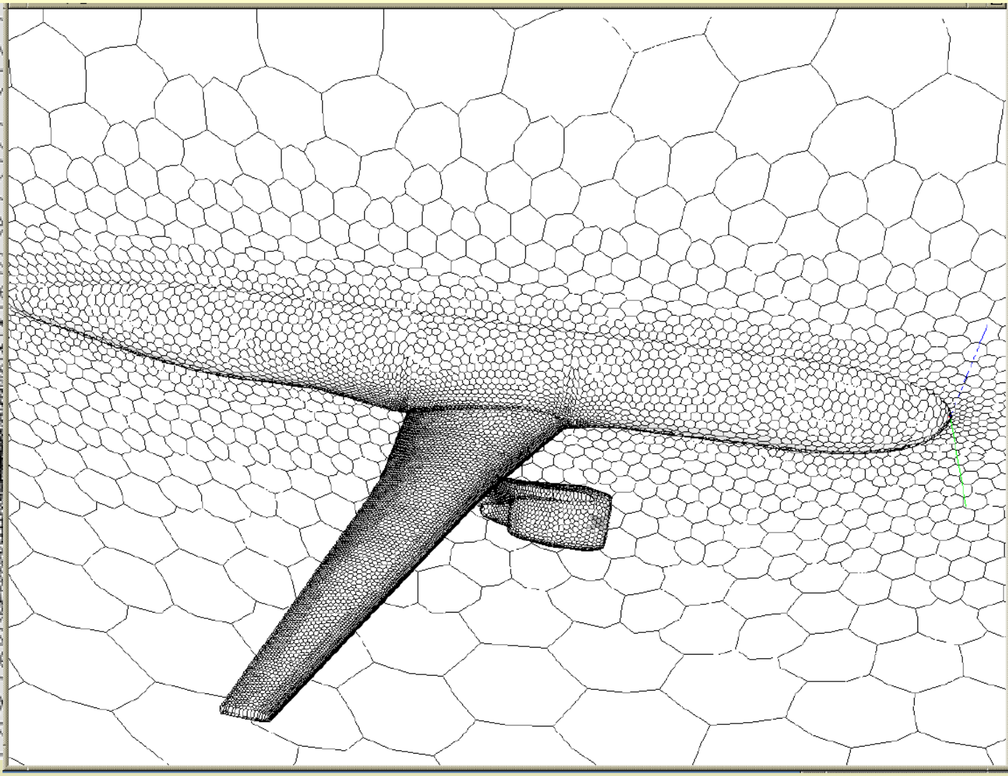
Agglomeration Multigrid

- Agglomeration Multigrid solvers for unstructured meshes
  - Coarse level meshes constructed by agglomerating fine grid cells/equations



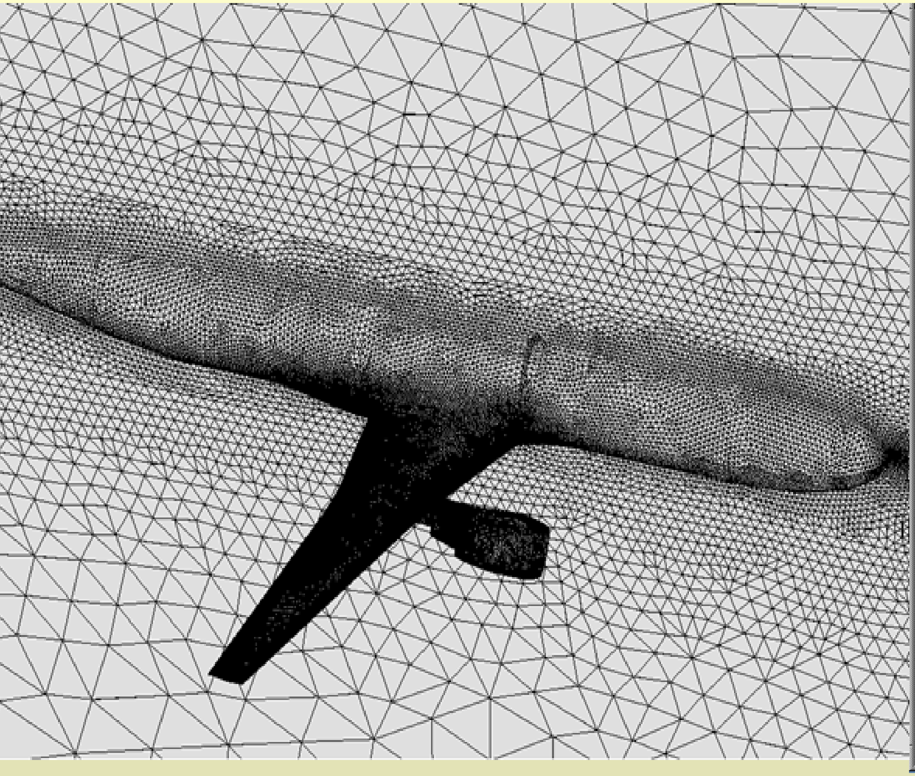
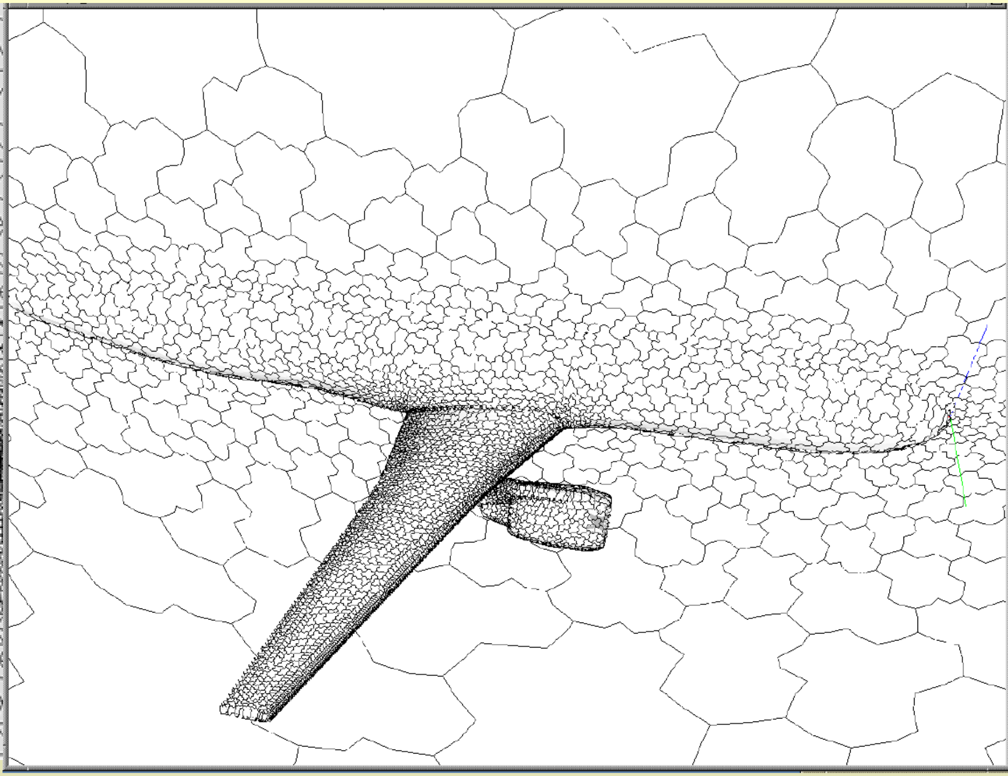
Agglomeration Multigrid





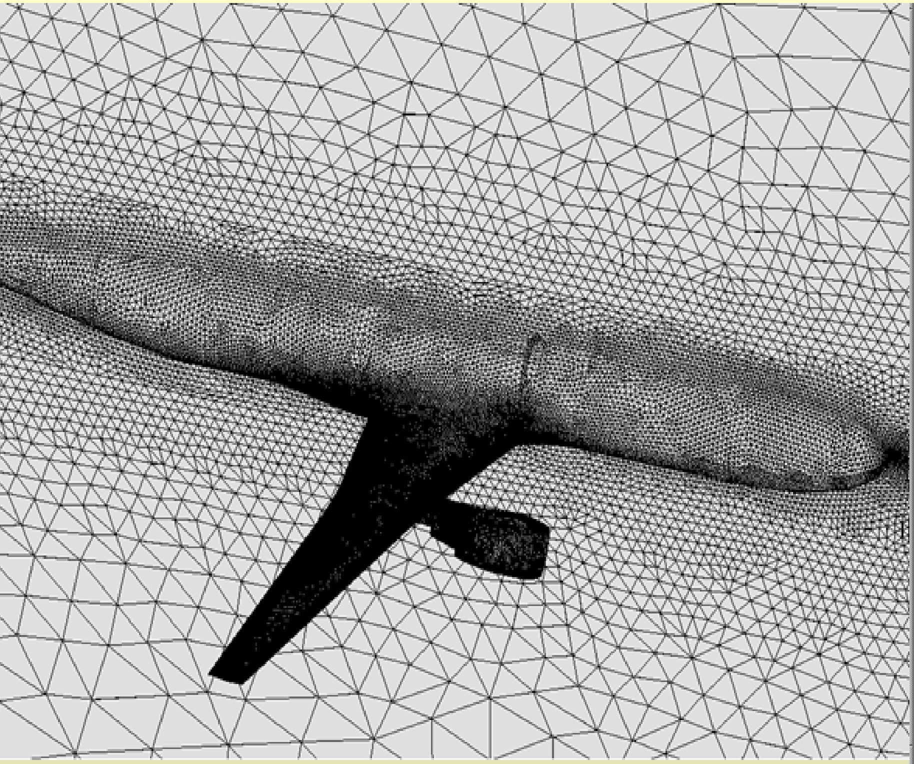
- Automated Graph-Based Coarsening Algorithm
- Coarse Levels are Graphs
- Coarse Level Operator by Galerkin Projection
- Grid independent convergence rates (order of magnitude improvement)

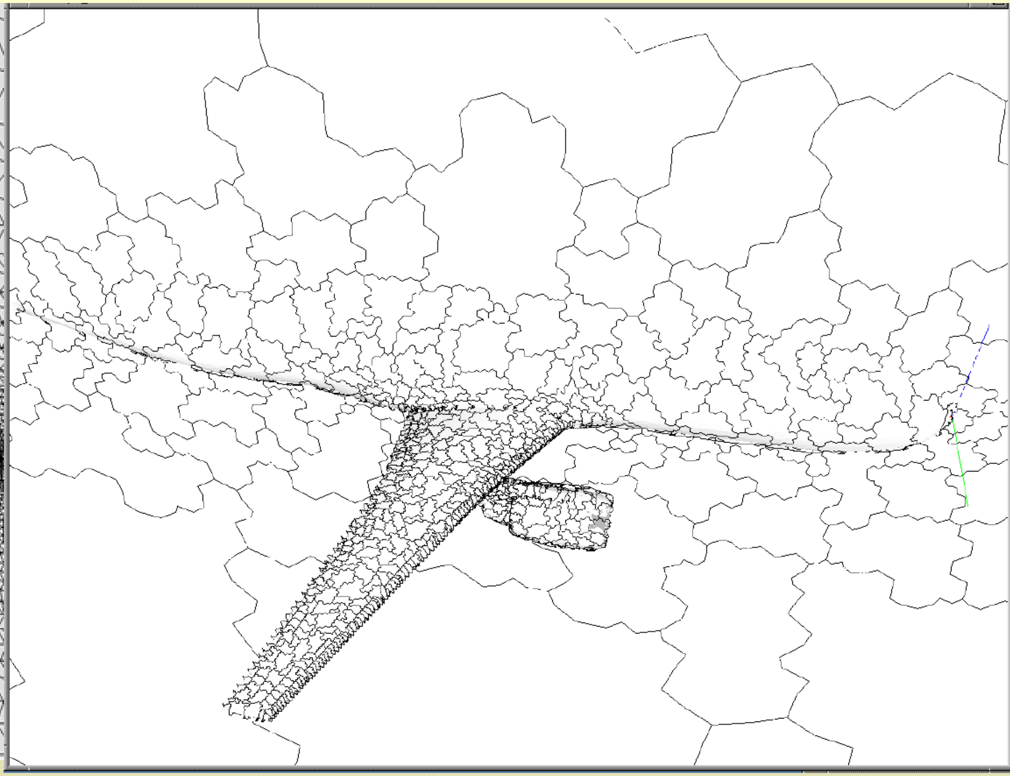
Agglomeration Multigrid

- •Automated Graph-Based Coarsening Algorithm
- •Coarse Levels are Graphs
- •Coarse Level Operator by Galerkin Projection
- •Grid independent convergence rates (order of magnitude improvement)

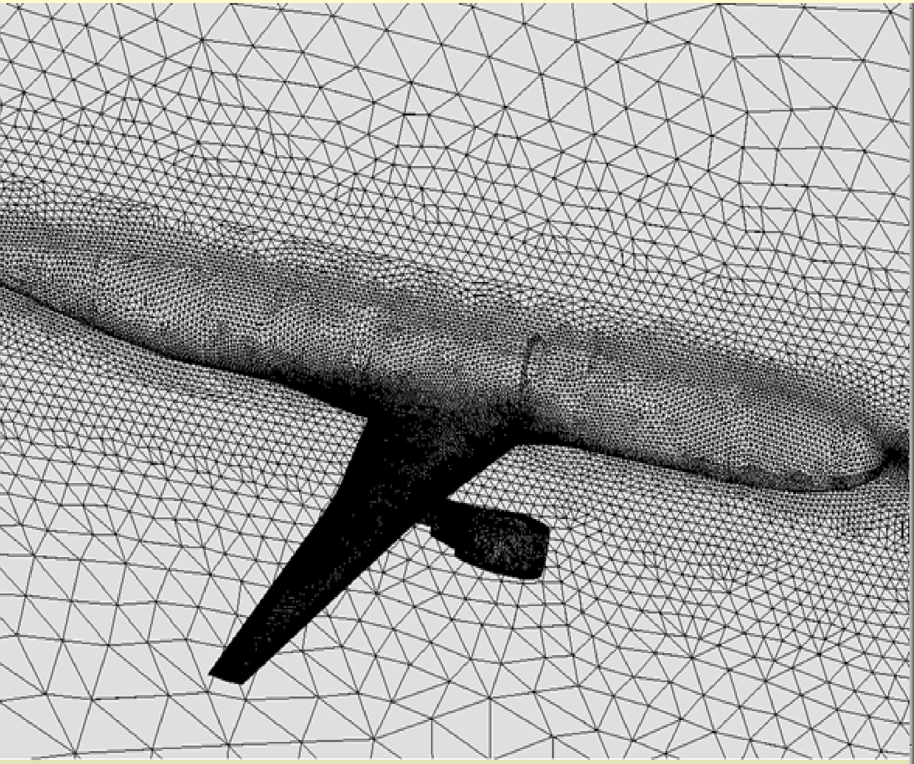
Agglomeration Multigrid





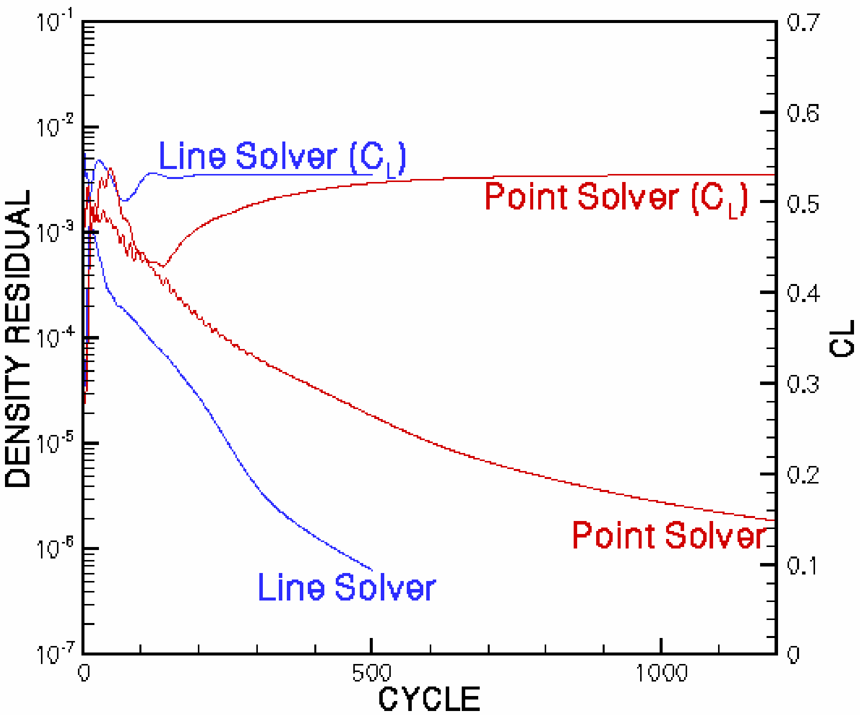
- Automated Graph-Based Coarsening Algorithm
- Coarse Levels are Graphs
- Coarse Level Operator by Galerkin Projection
- Grid independent convergence rates (order of magnitude improvement)

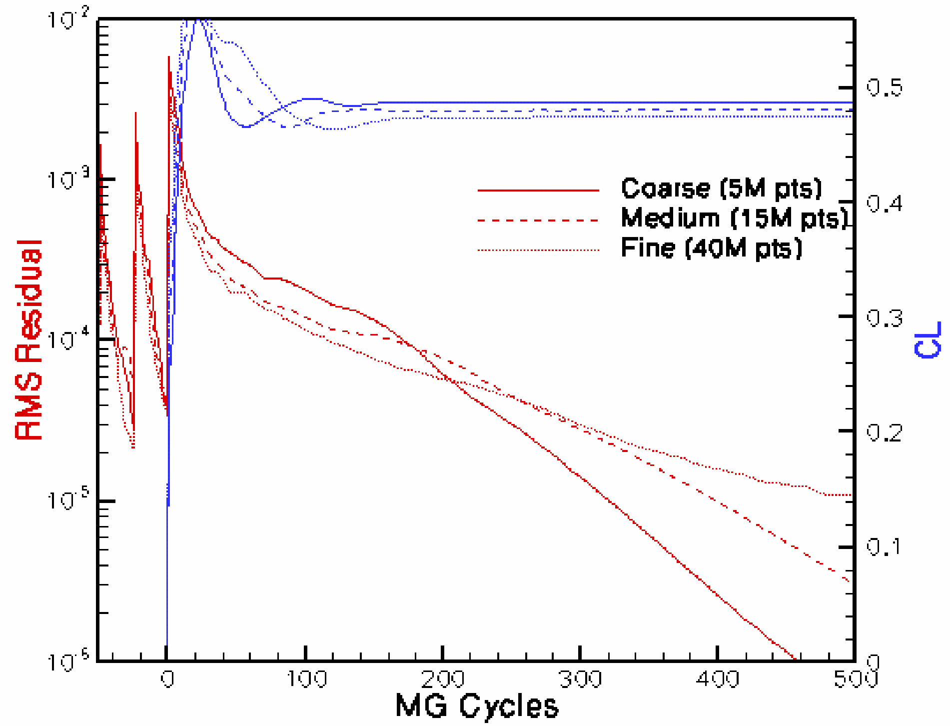
Agglomeration Multigrid



- Automated Graph-Based Coarsening Algorithm
- Coarse Levels are Graphs
- Coarse Level Operator by Galerkin Projection
- Grid independent convergence rates (order of magnitude improvement)

Line Solver Multigrid Convergence

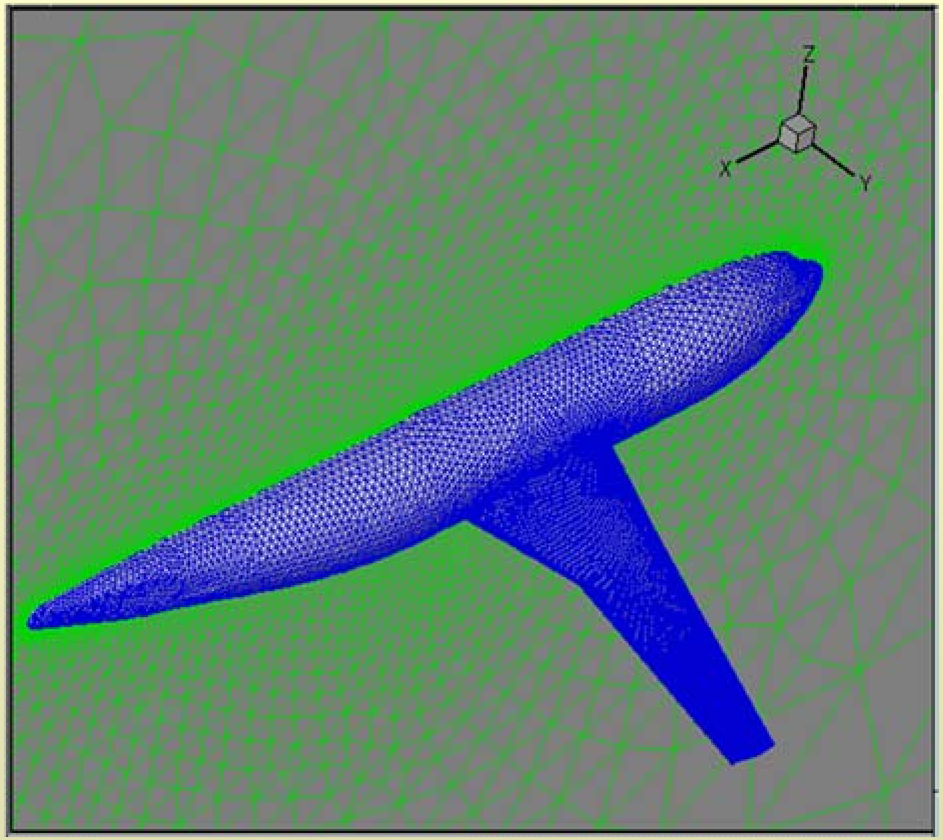


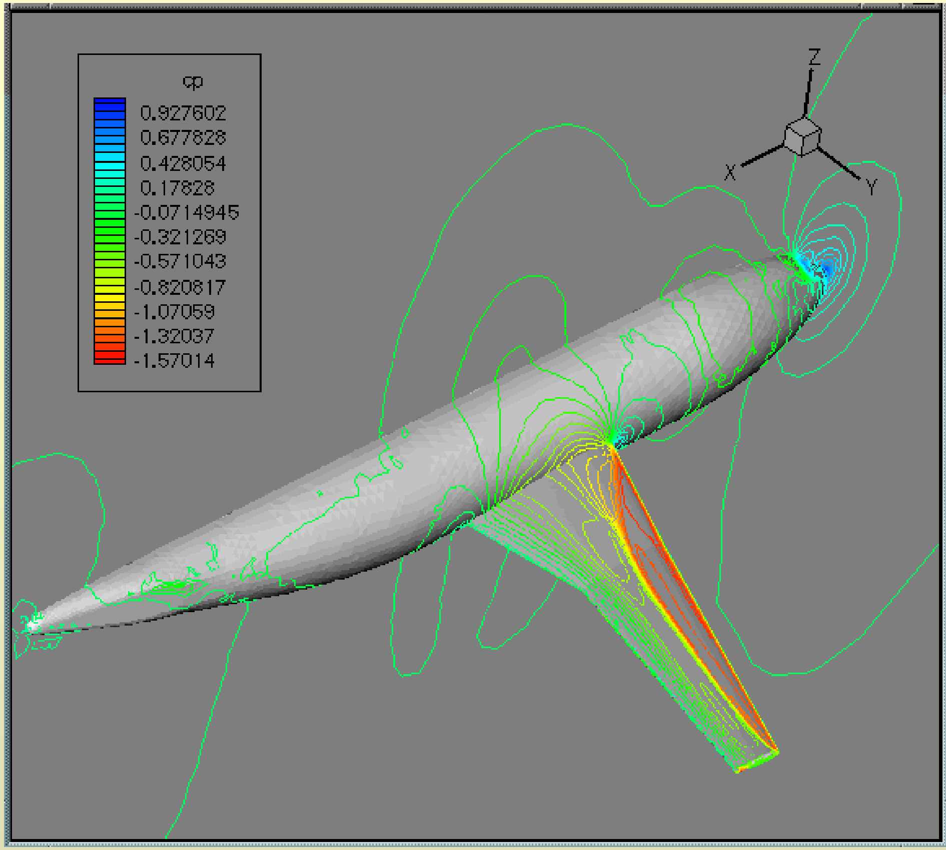


Line solver convergence insensitive to grid stretching

Multigrid convergence insensitive to grid resolution

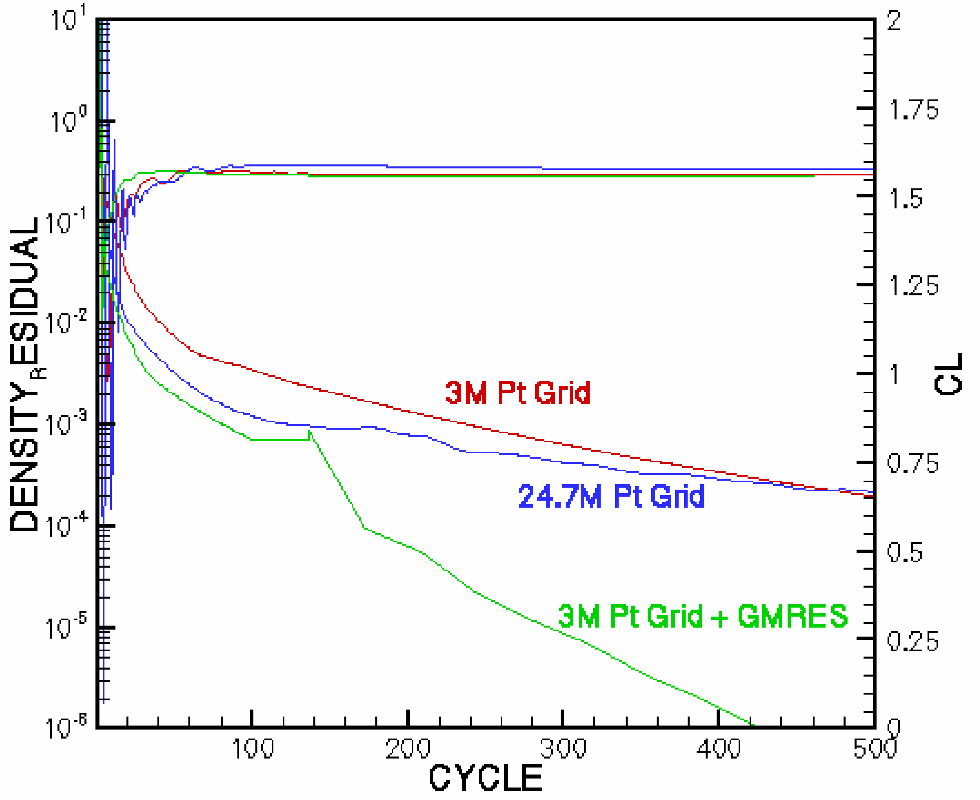
NSU3D TEST CASE

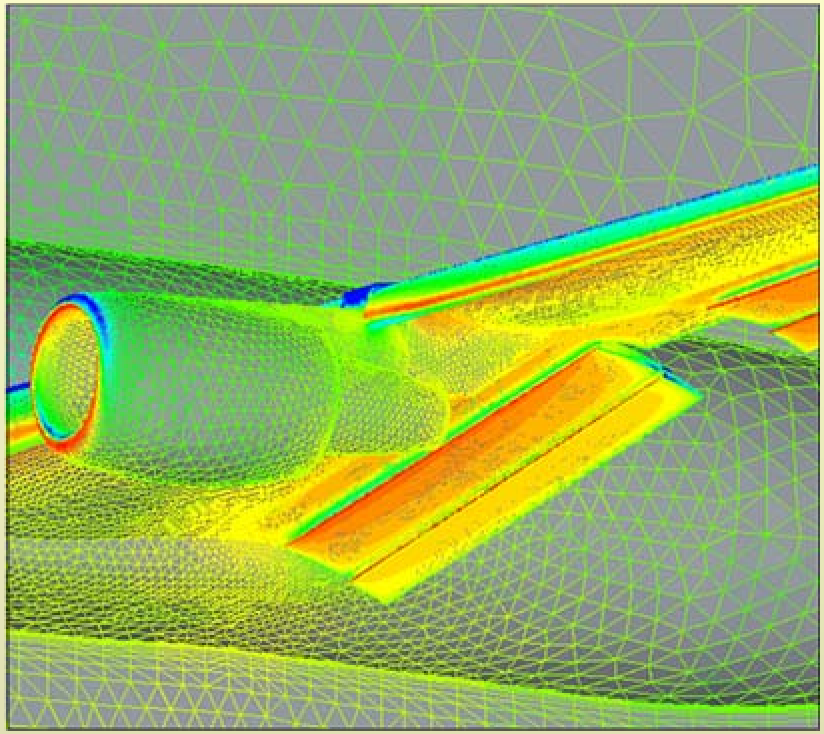




- Wing-Body Configuration
- 3 to 72 million grid points
- Transonic Flow
- Mach=0.75, Incidence = 0 degrees, Reynolds number=3,000,000

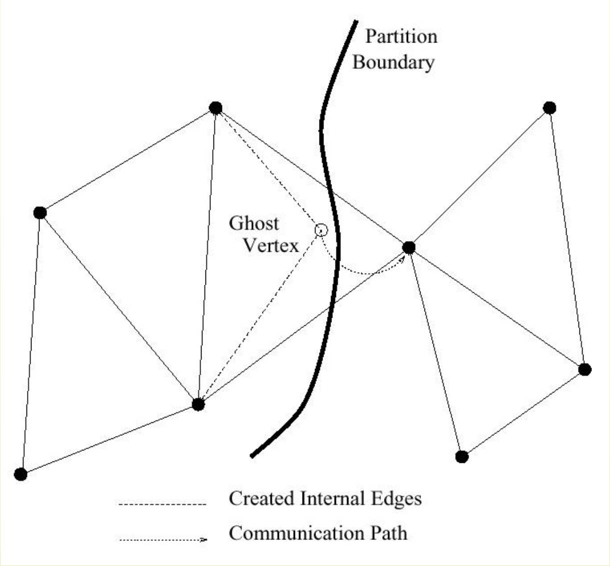
(Multigrid) Preconditioned Newton Krylov





- Mesh independent property of Multigrid
- GMRES effective (in asymptotic range) but requires extra memory

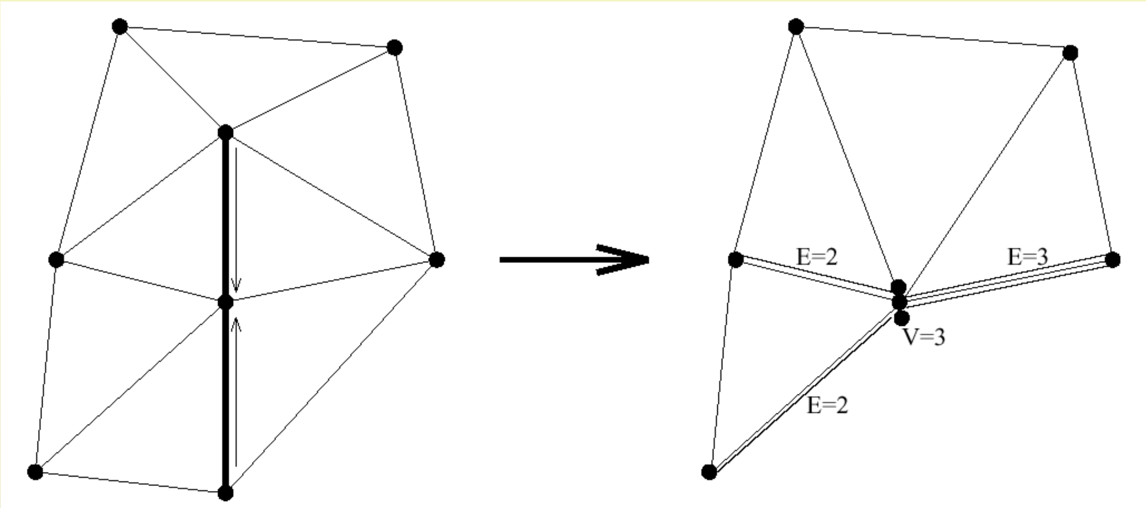
Parallelization through Domain Decomposition



- Intersected edges resolved by ghost vertices
- Generates communication between original and ghost vertex
  - Handled using MPI and/or OpenMP (Hybrid implementation)
  - Local reordering within partition for cache-locality

Partitioning

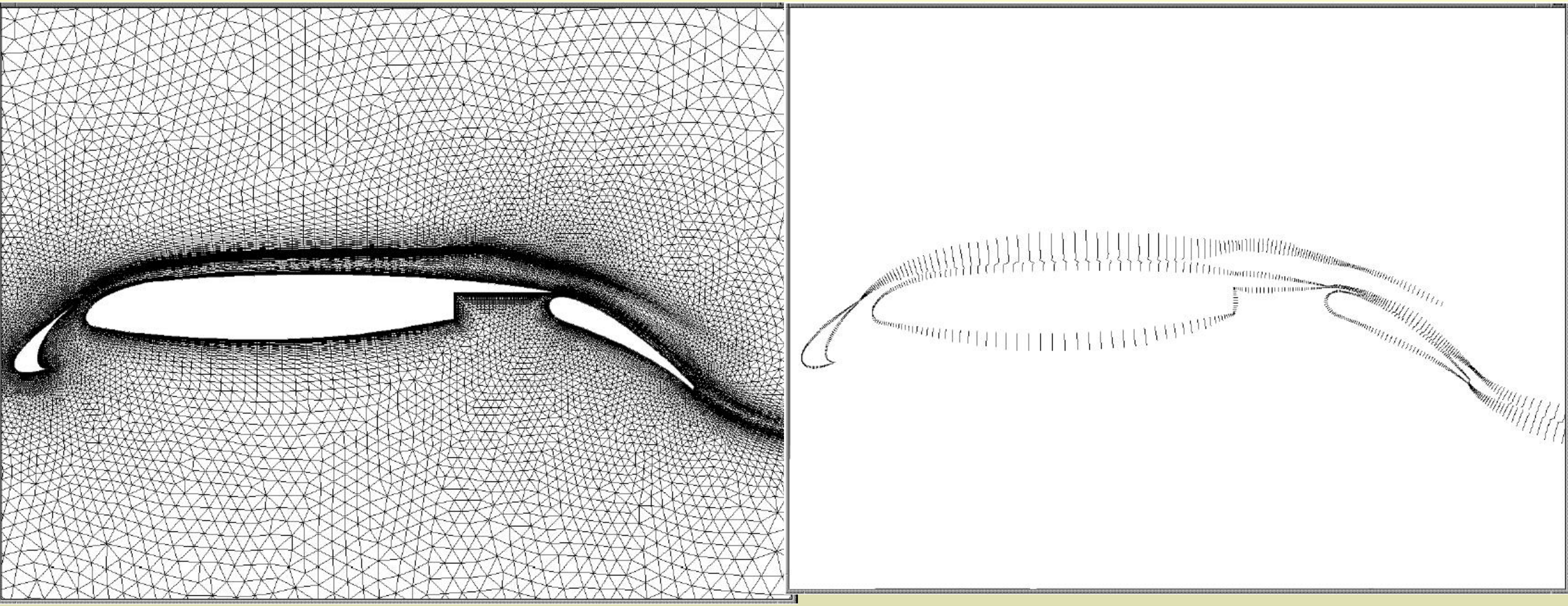
- (Block) Tridiagonal Lines solver inherently sequential
- Contract graph along implicit lines
- Weight edges and vertices



- Partition contracted graph
- Decontract graph
  - Guaranteed lines never broken
  - Possible small increase in imbalance/cut edges

Partitioning Example

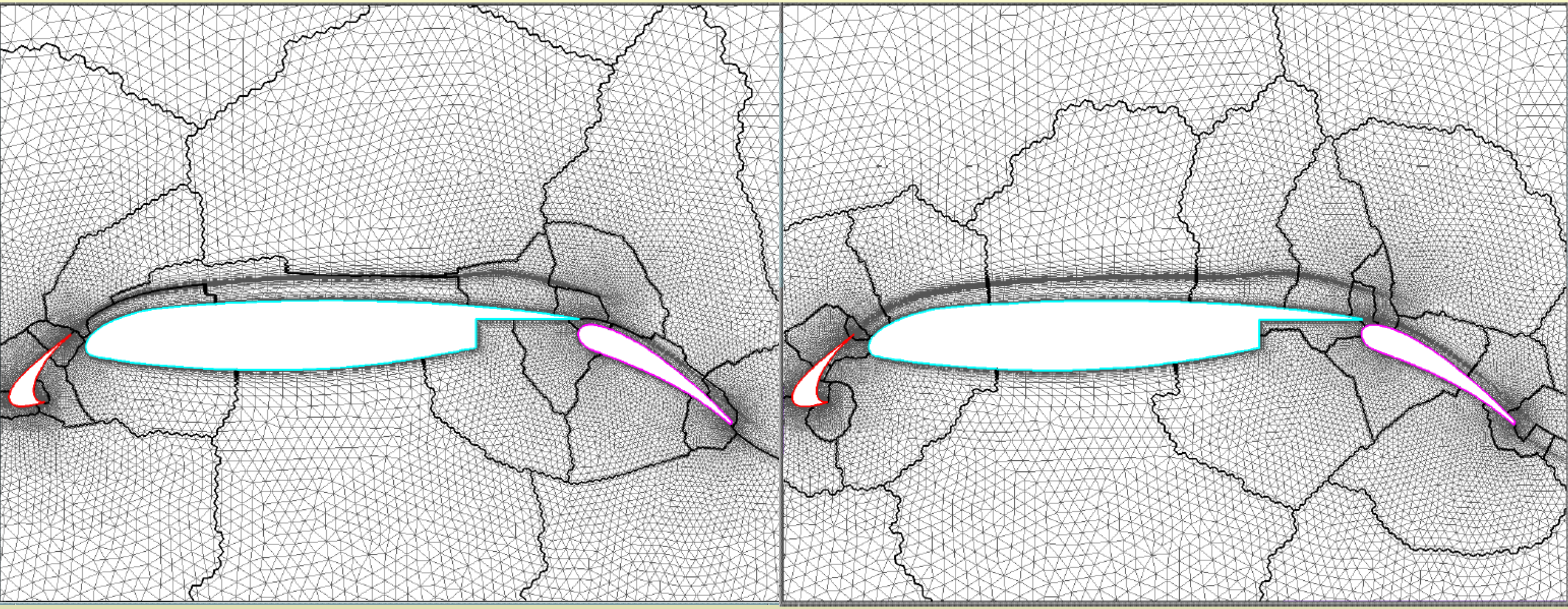
- 32-way partition of 30,562 point 2D grid



- Unweighted partition: 2.6% edges cut, 2.7% lines cut
- Weighted partition: 3.2% edges cut, 0% lines cut

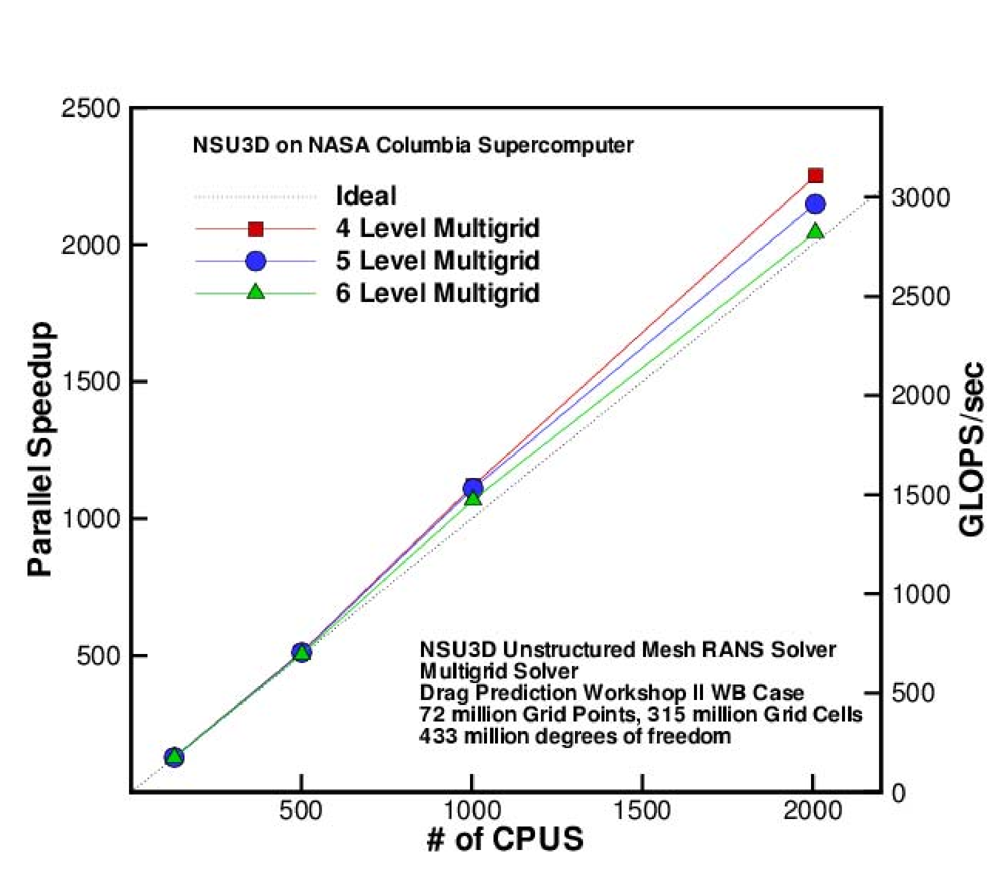
Partitioning Example

- 32-way partition of 30,562 point 2D grid



- Unweighted partition: 2.6% edges cut, 2.7% lines cut
- Weighted partition: 3.2% edges cut, 0% lines cut

Scalability



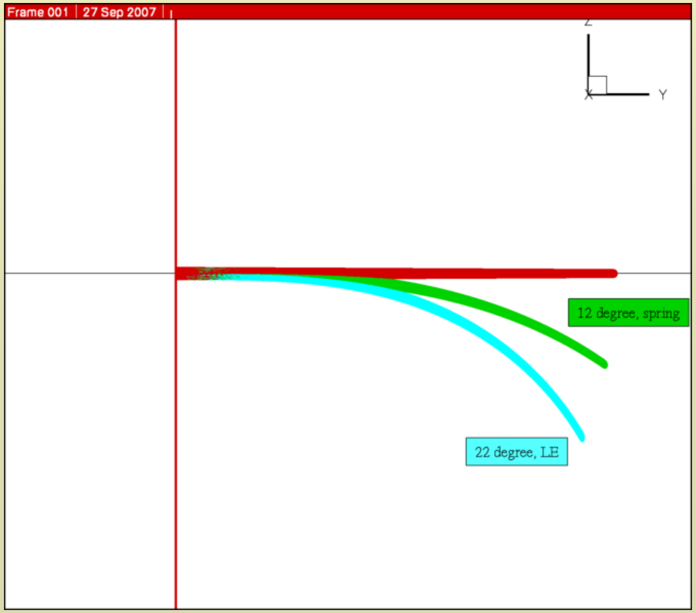
- Near ideal speedup for 72M pt grid on 2008 cpus of NASA Columbia Machine
  - Homogeneous Data-Structure
  - Near perfect load balancing
  - Near Optimal Partitioners

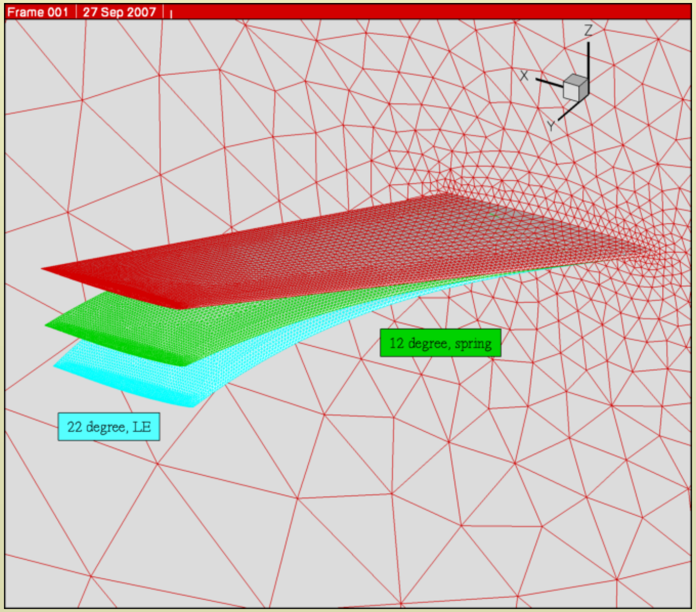
Mesh Motion

- Developed for MDO and Aeroelasticity Problems
- Emphasis on Robustness
  - Spring Analogy
  - Truss Analogy, Beam Analogy
  - Linear Elasticity: Variable Modulus
    - Regions of stiff E displace as solid body (no deformation)
- Emphasis on Efficiency
  - Edge based formulation
  - Gauss Seidel Line Solver with Agglomeration Multigrid

Mesh Deformation

- Mesh motion code tested for sample configurations
  - Bend until failure of mesh motion code
    - Spring analogy: Less robust, simpler
    - Linear elasticity analogy: More robust, more costly (x2)
      - Solved by multigrid in all cases

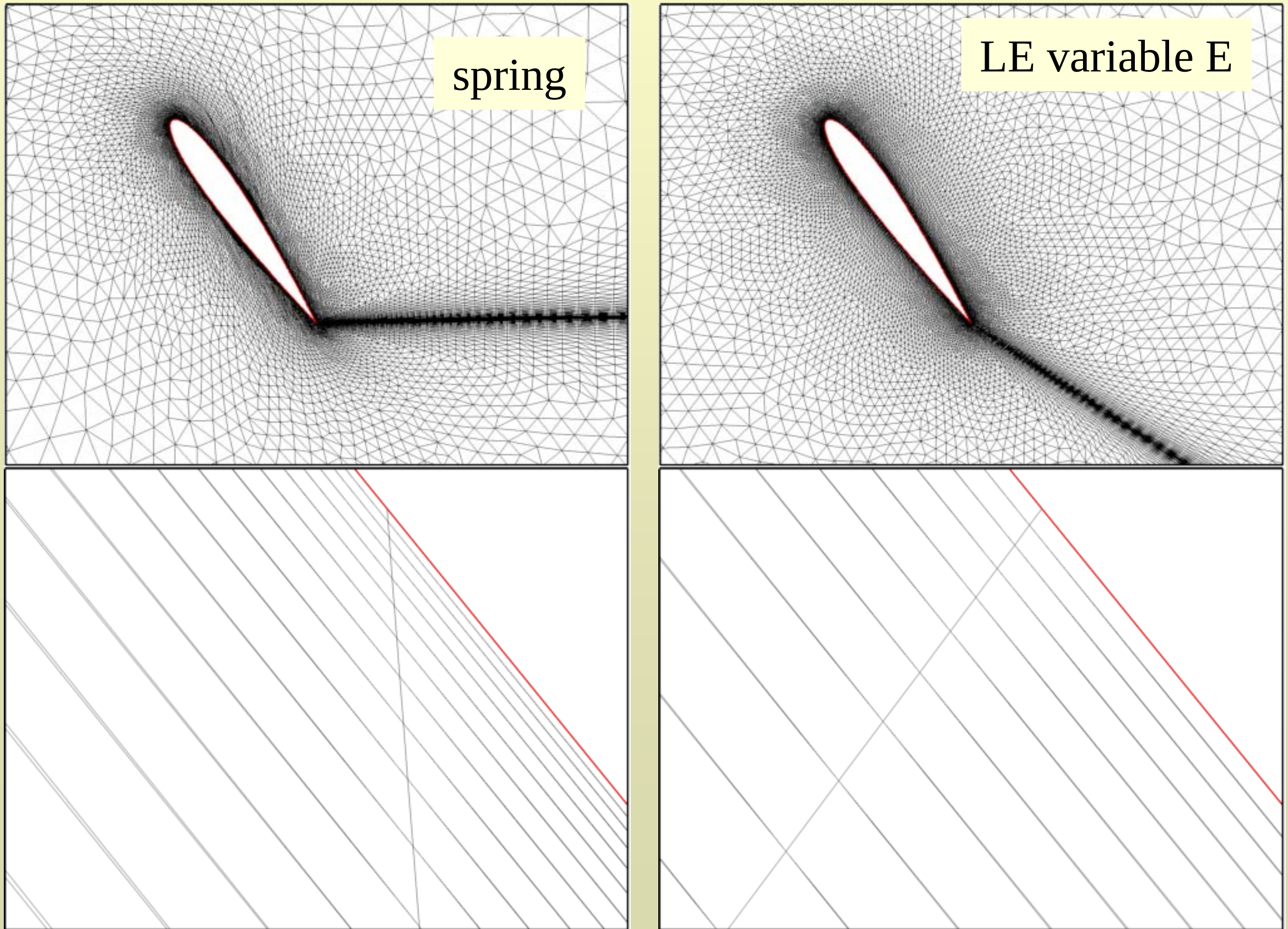




Mesh Motion Formulations

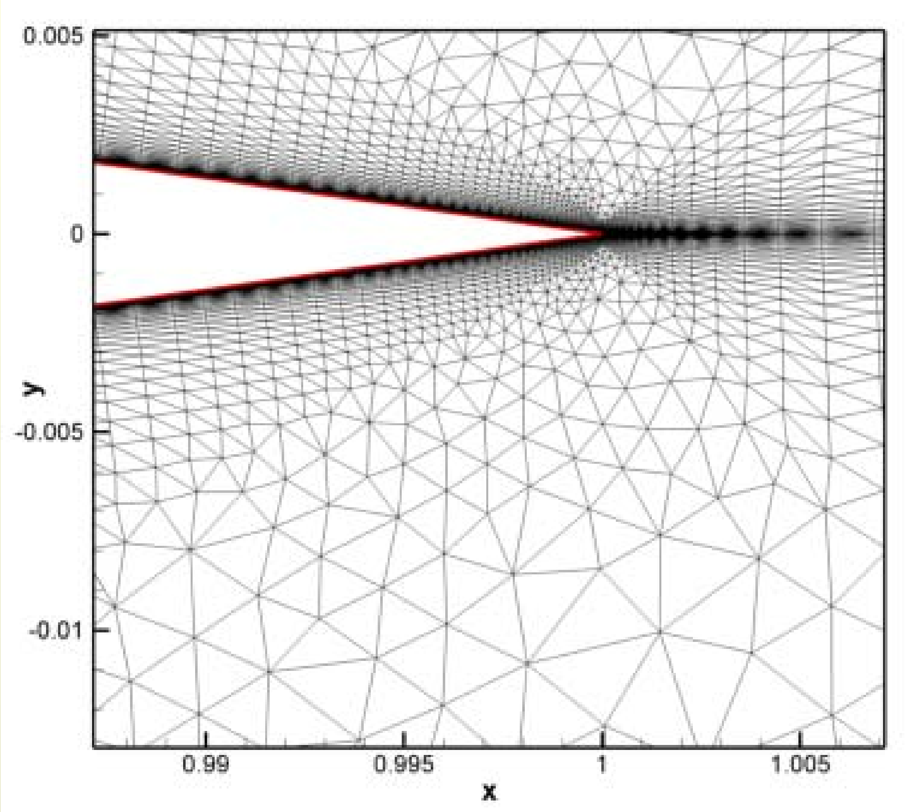
- *Linear elasticity approach preserves mesh orthogonality near surface (displaces mesh as solid body in regions of high E)*

- *Line solver displaces boundary layer mesh region rapidly in response to surface deflections*



Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

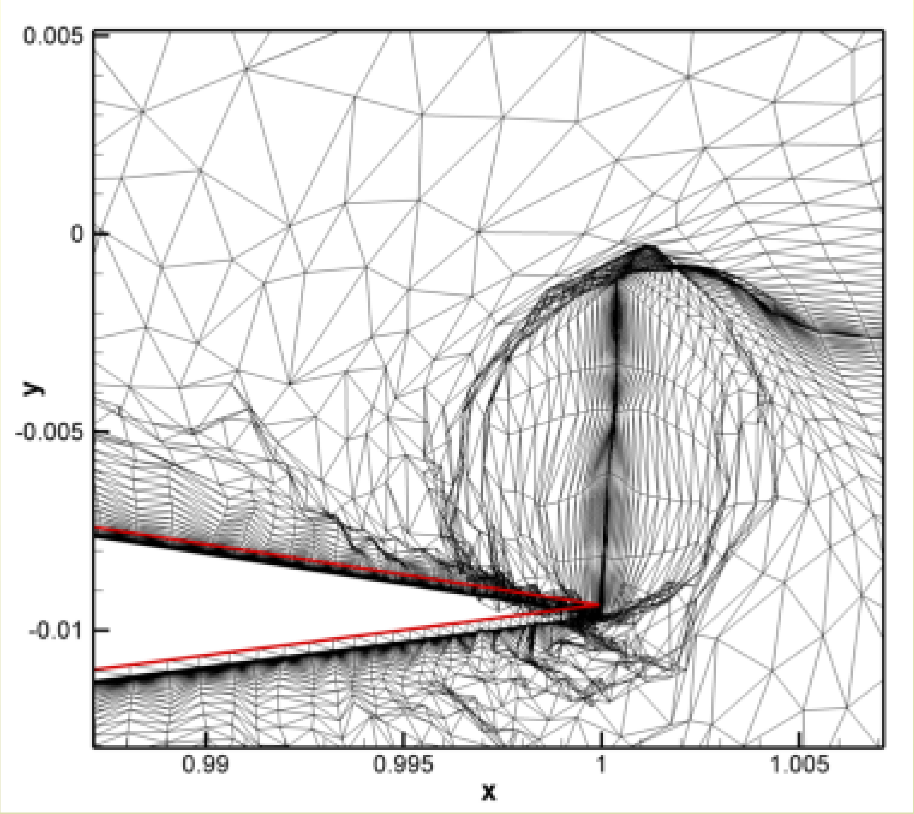


iter= 0

Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

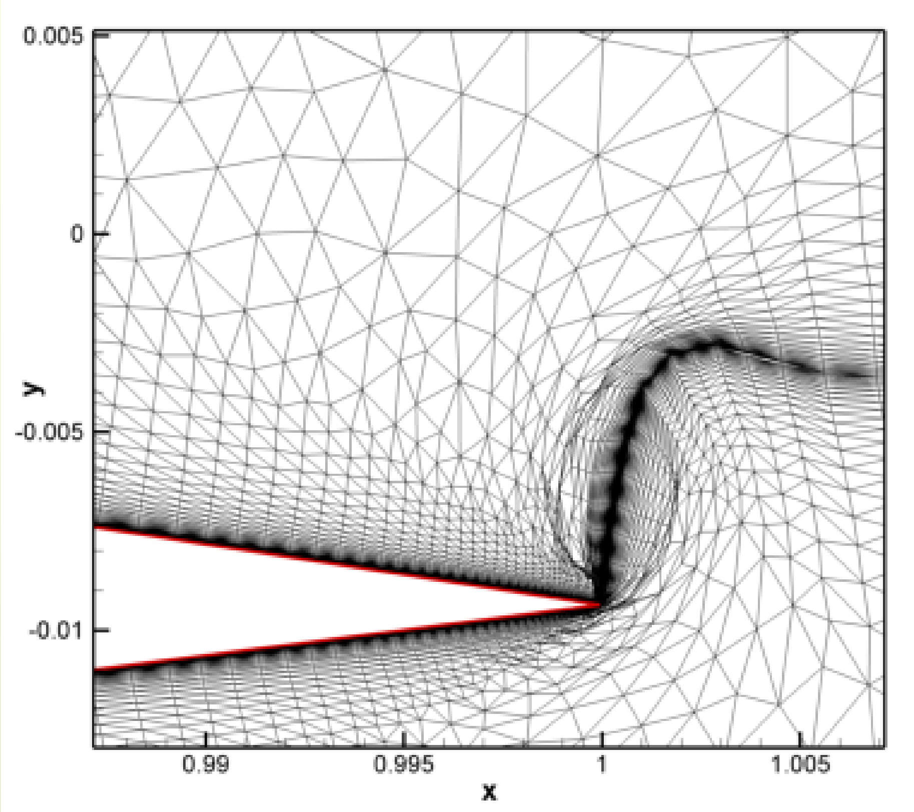


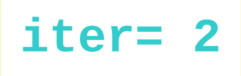
iter= 1

Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

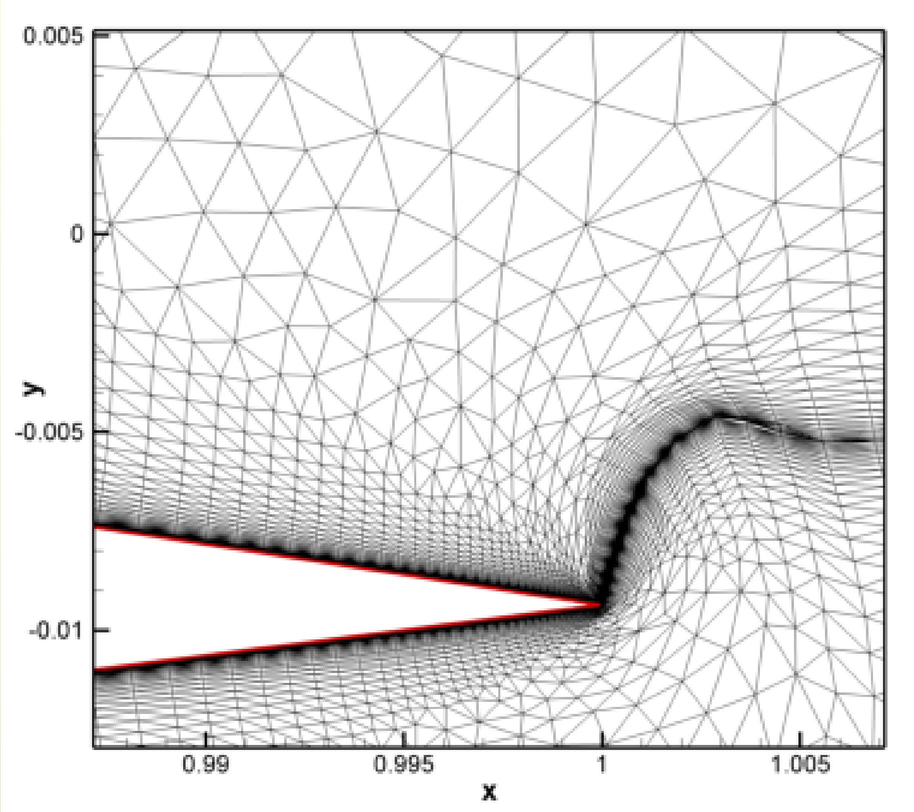




Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

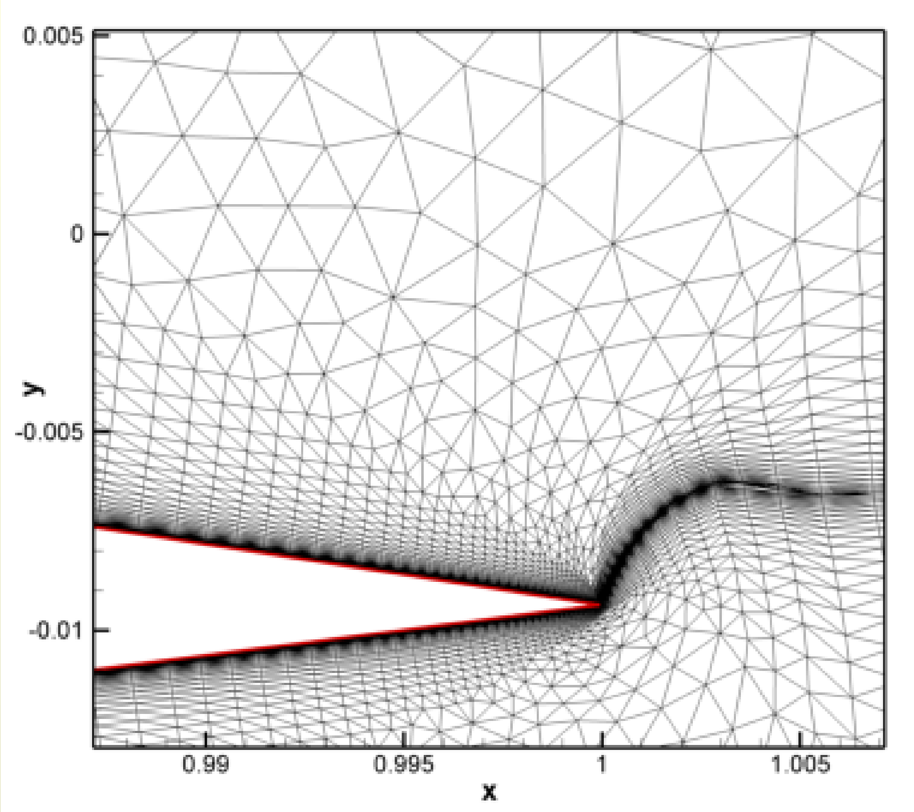


iter= 3

Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

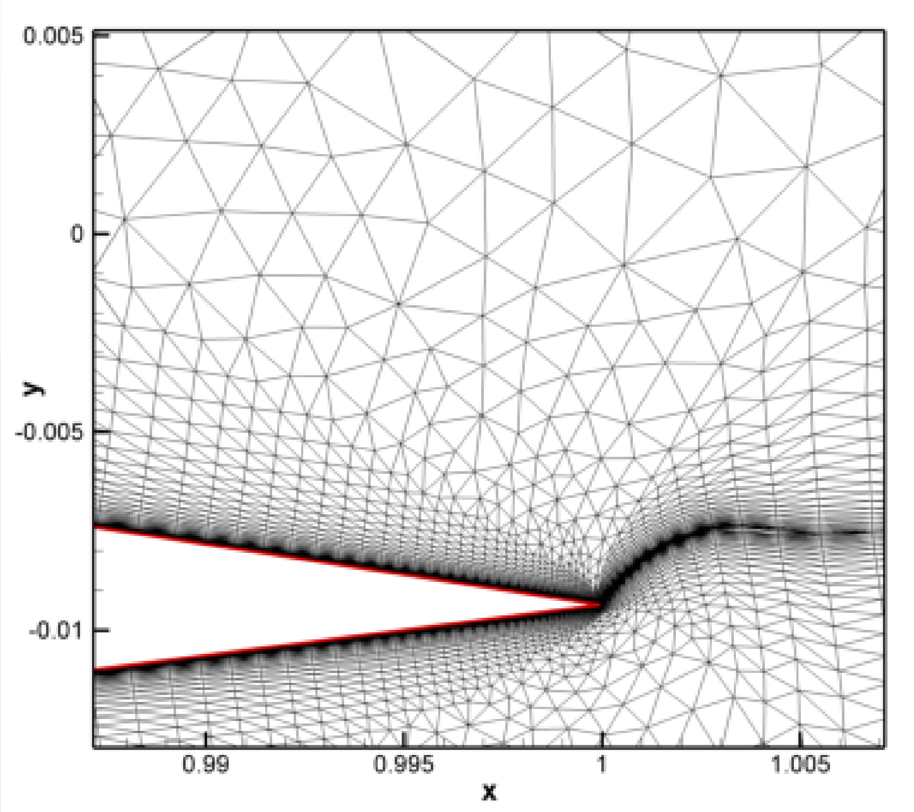


iter= 4

Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

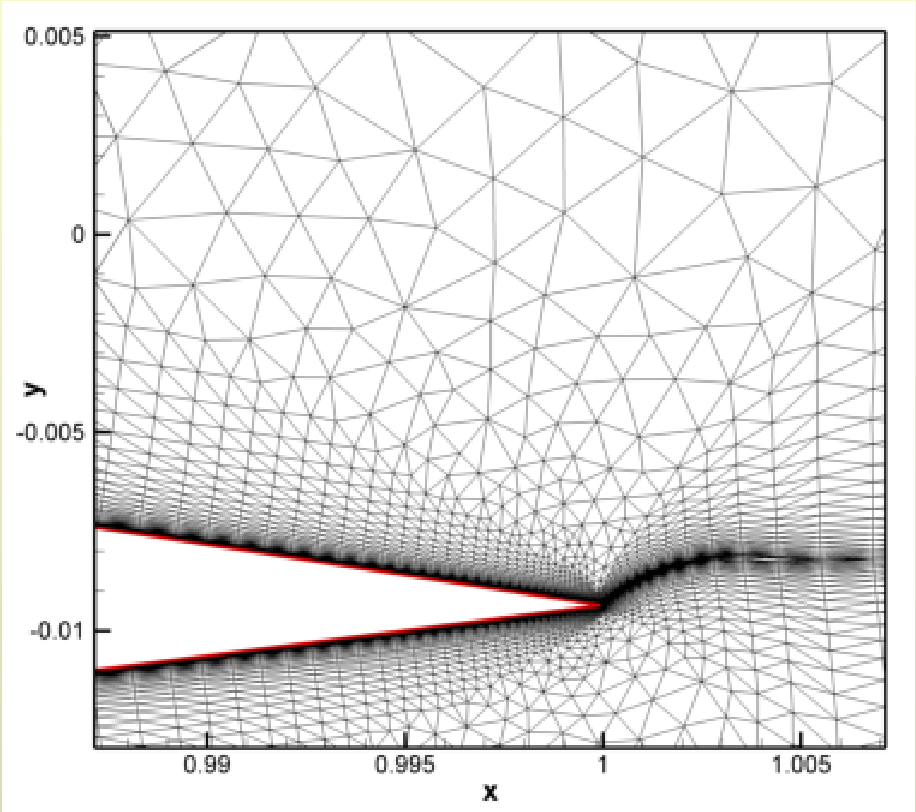


iter= 5

Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

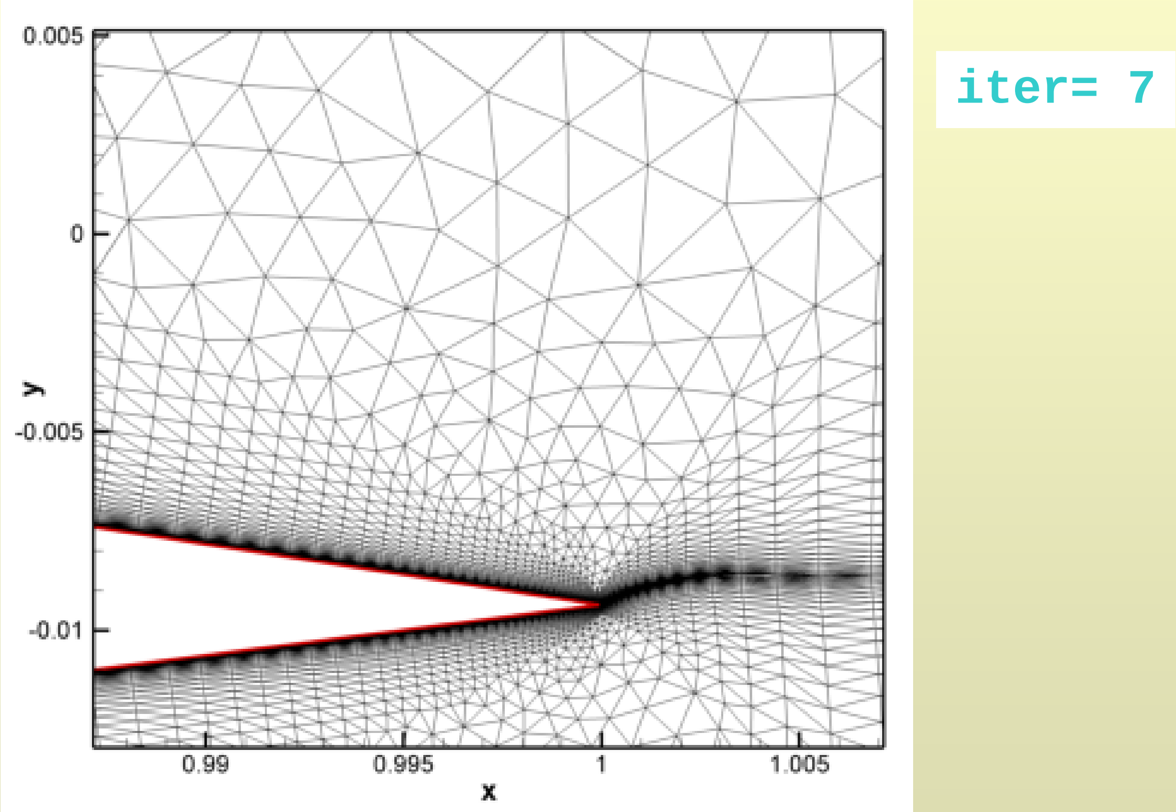


iter= 6

Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

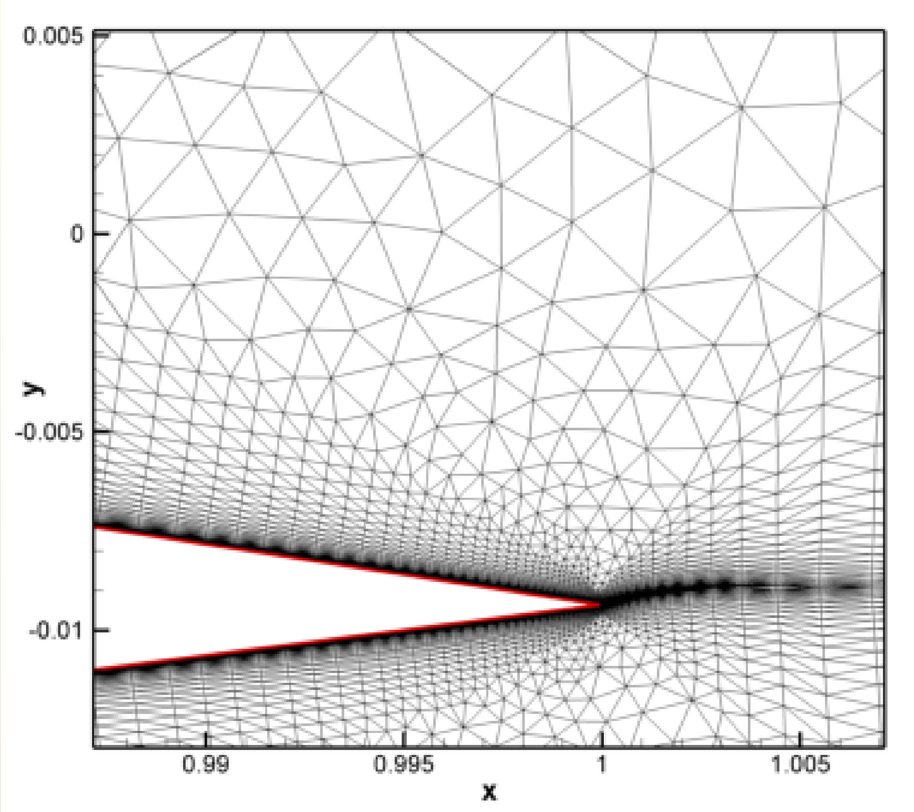
- Line solver + MG4 , first 10 iterations

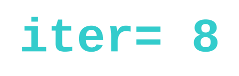


Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

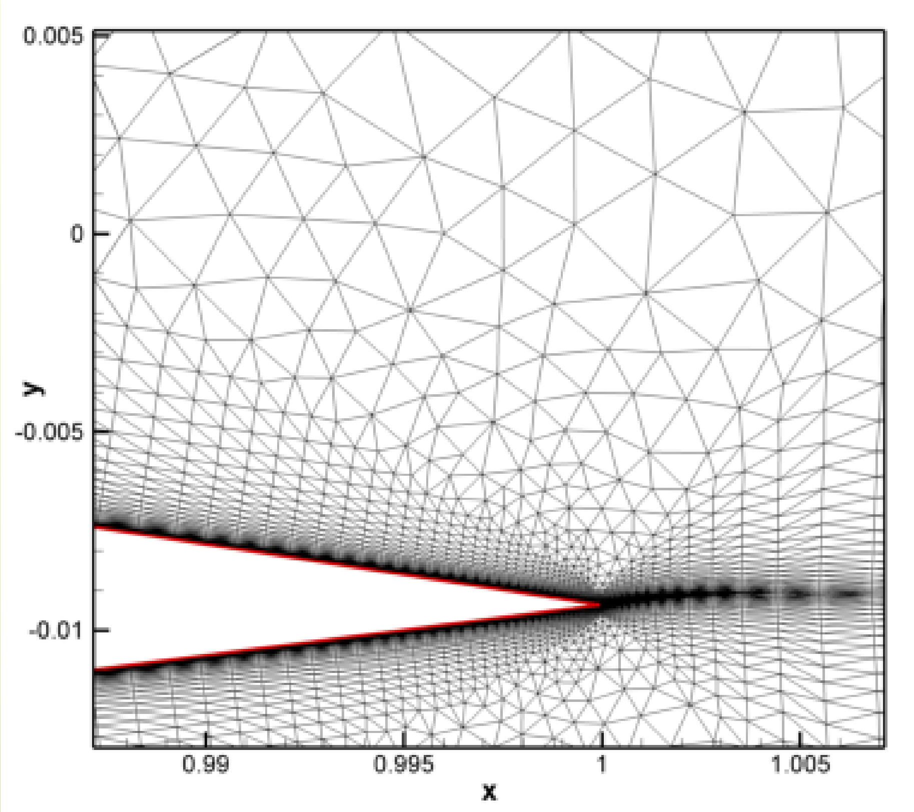


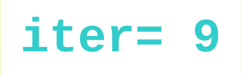


Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

- Line solver + MG4 , first 10 iterations

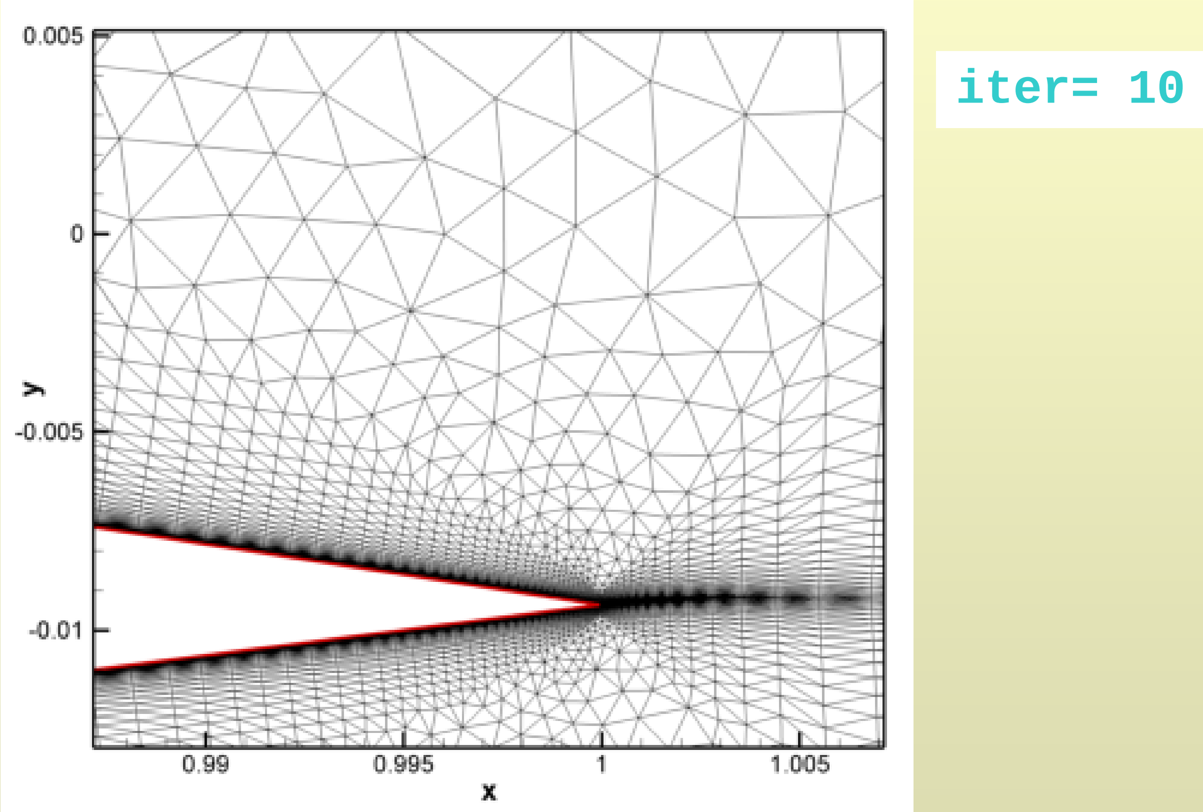




Viscous mesh, linear elasticity with variable E

Fast MG Solution of Mesh Motion Eqns

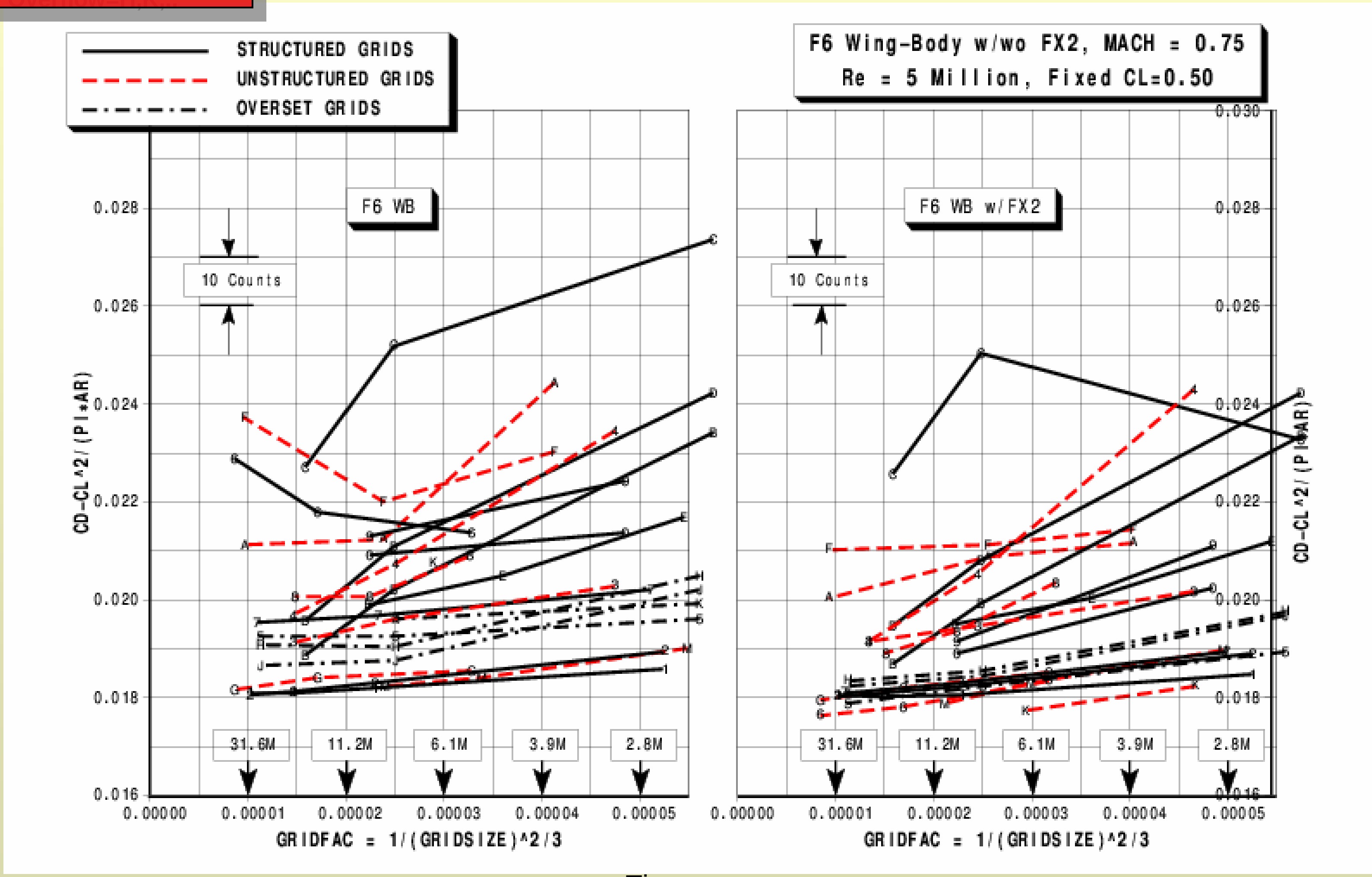
- Line solver + MG4 , first 10 iterations



Viscous mesh, linear elasticity with variable E

NSU3D=G
CFL3D=1,2
Overflow=H,K,..

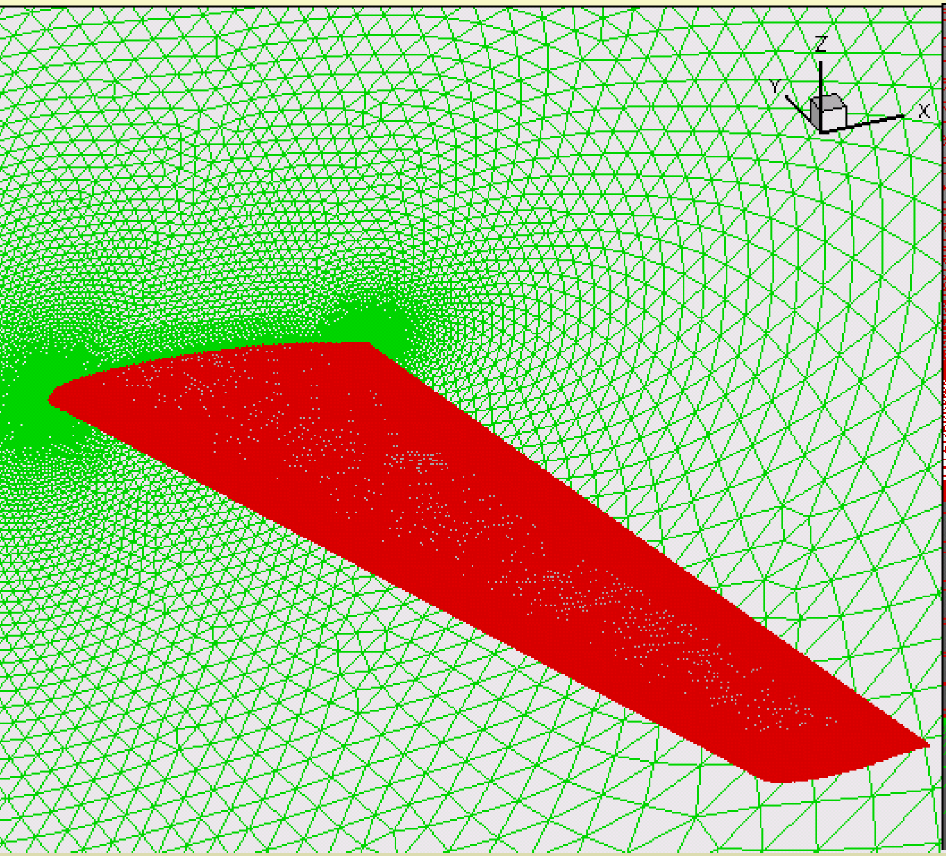
Grid Convergence – All Solutions

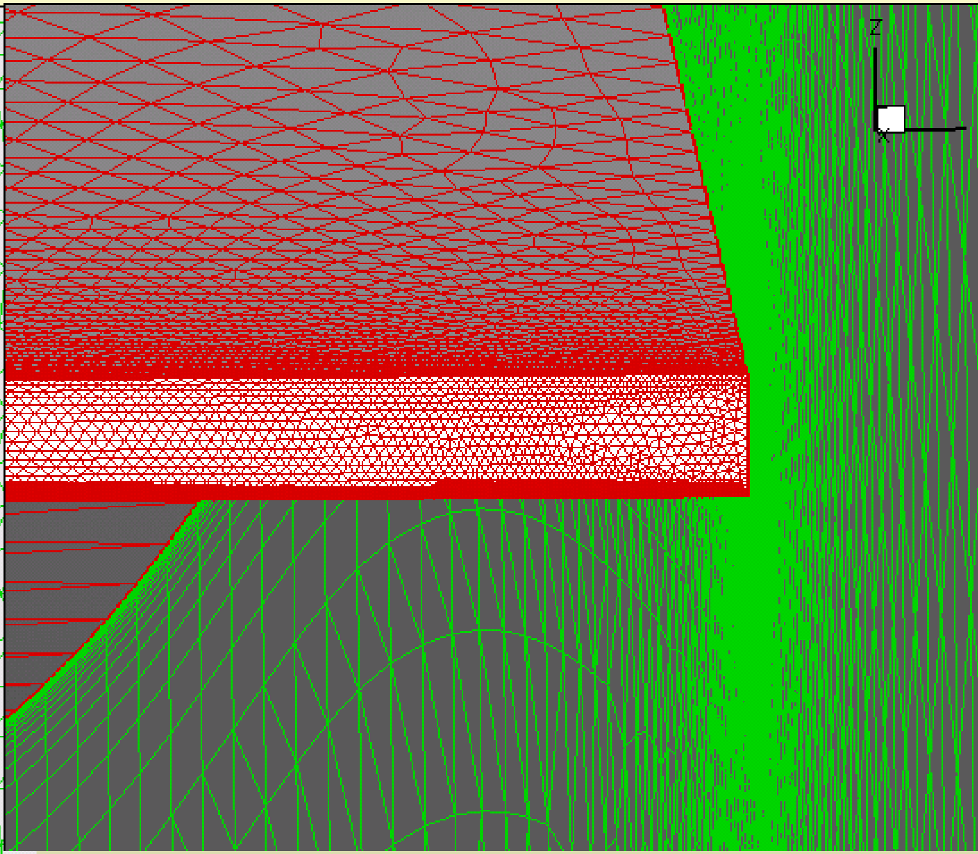


Unstructured vs Structured (Transonics)

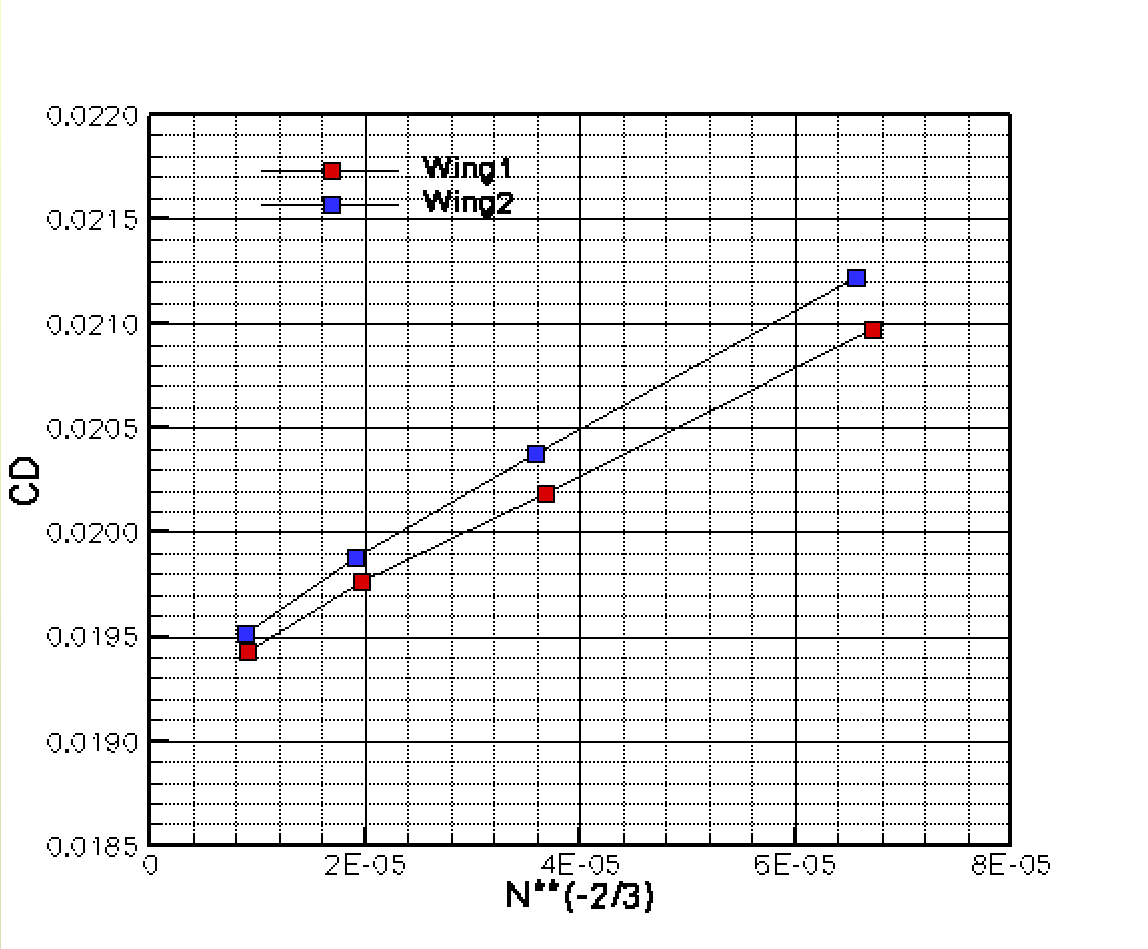
- Considerable scatter in both cases
- No clear advantage of one method over the other in terms of accuracy
- DPW3 Observation:
  - Core set of codes which:
    - Agree remarkably well with each other
    - Span all types of grids
      - Structured, Overset, Unstructured
    - Have been developed and used extensively for transonic aerodynamics

DPW3 Wing1-Wing2 Cases



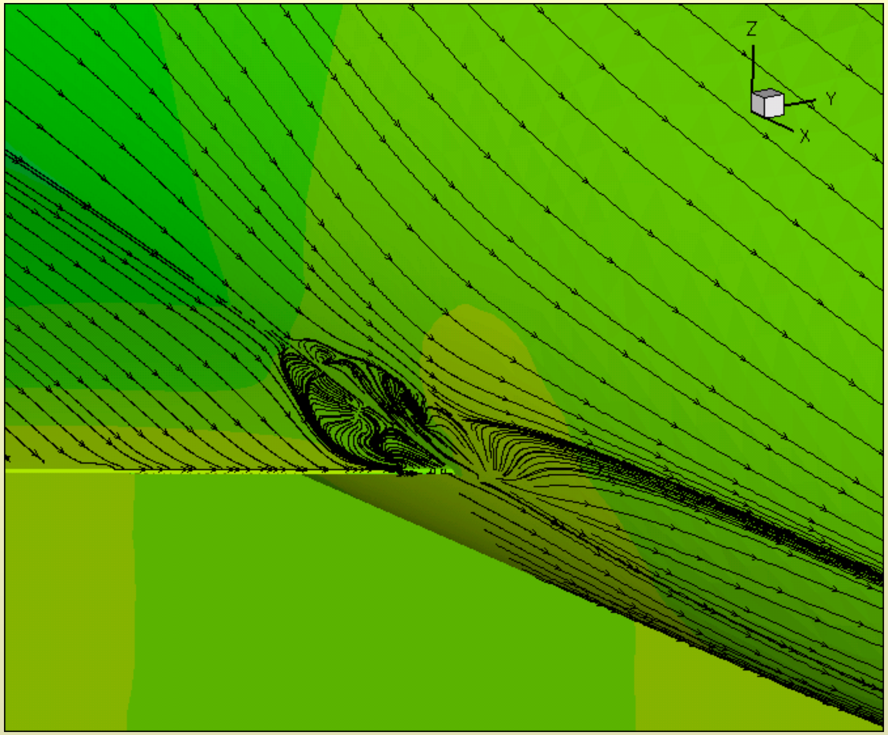


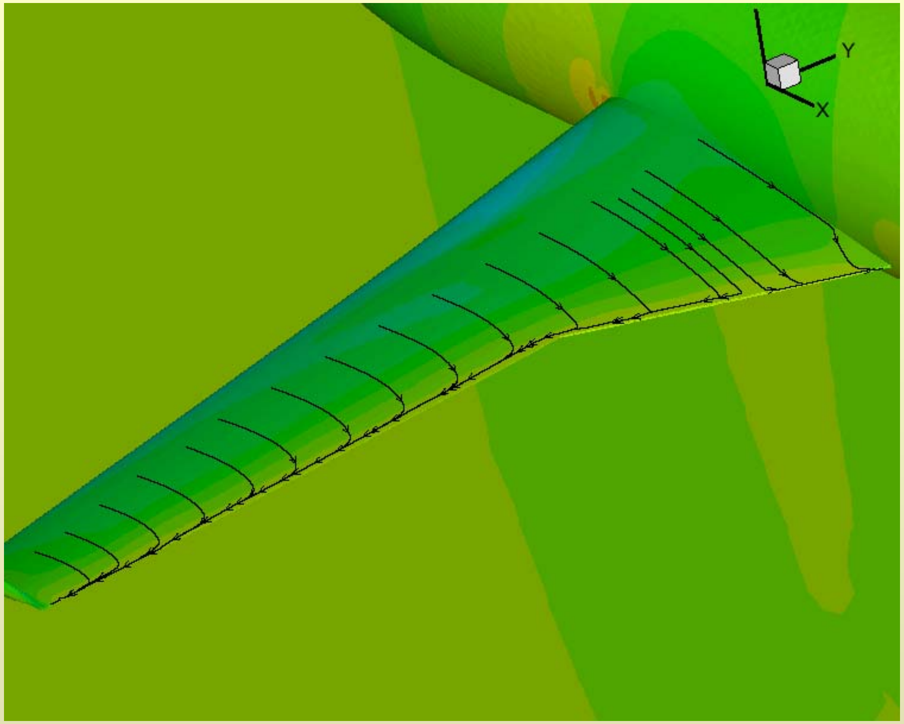
W1-W2 Grid Convergence Study (NSU3D)



•Apparently uniform grid convergence (attached flow cases)

Grid Resolution





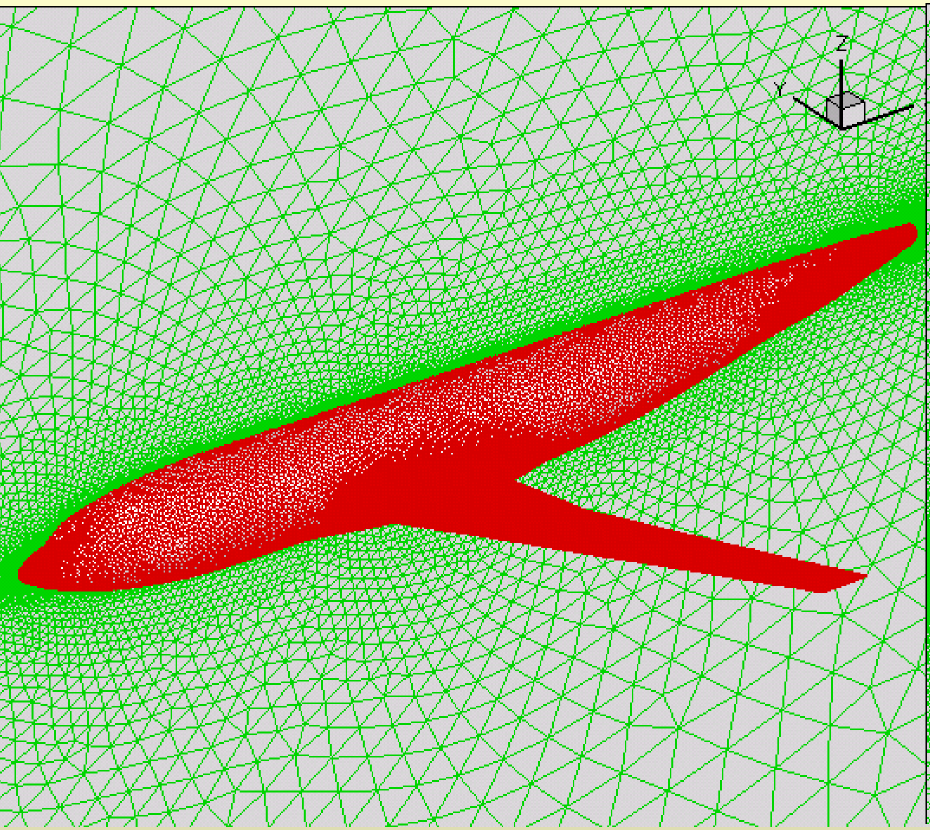
- Separated flow cases more demanding and often contradictory experiences

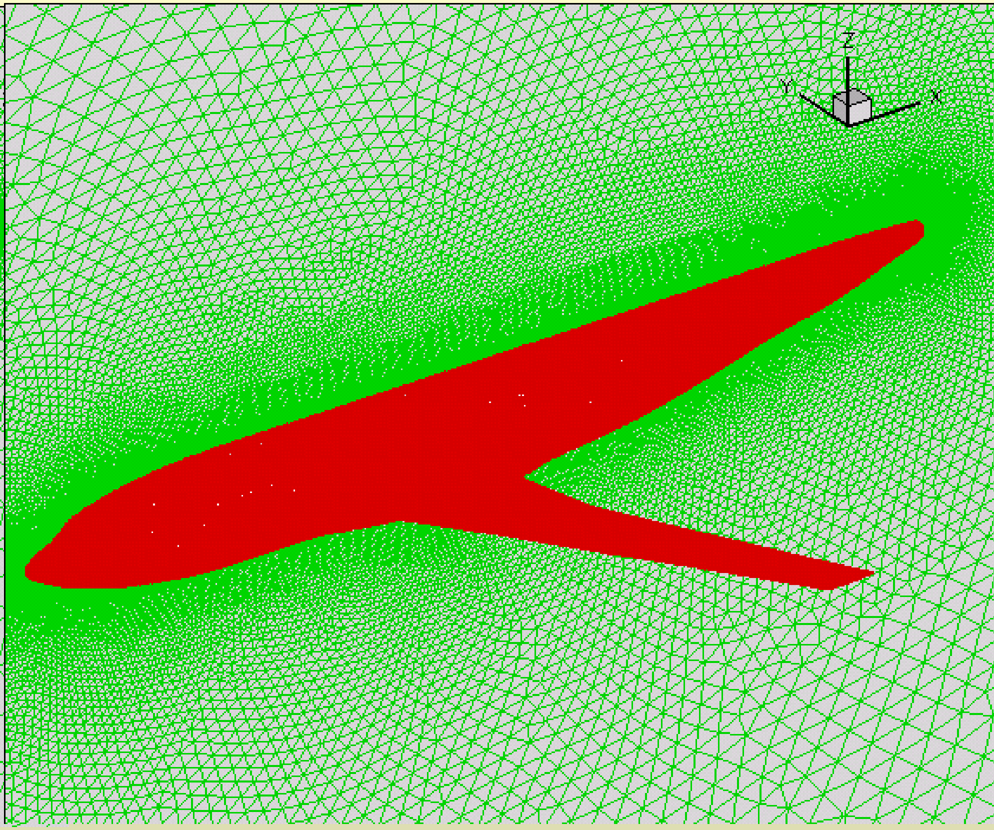
Grid Resolution Effects (DPW-3)

AIAA Paper 2008-0930

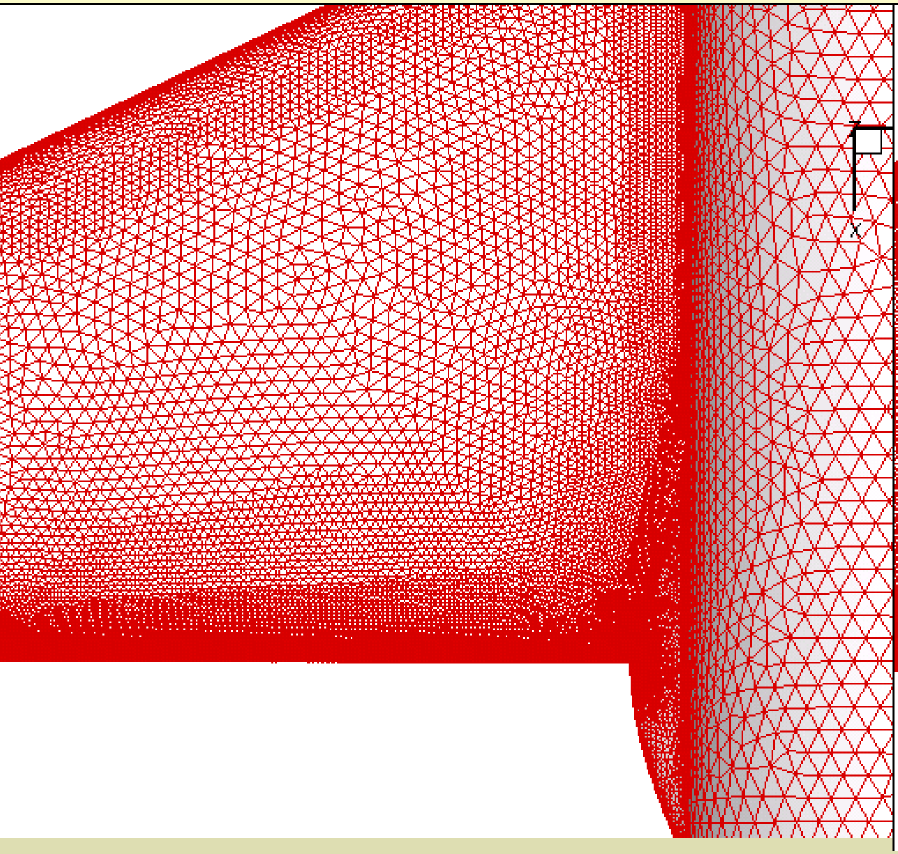
- SOB Separation increases with grid resolution
  - Boeing: Overset
  - Boeing: Unstructured
  - DLR: Unstructured
- SOB Separation remains constant with grid res.
  - Boeing: Block Structured
  - JAXA: Block Structured, Unstructured
- Trailing edge separation grows with grid res:
  - UW : Unstructured (NSU3D)
- Trailing edge separation constant with grid res:
  - JAXA: Structured, Unstructured
  - Boeing: Overset

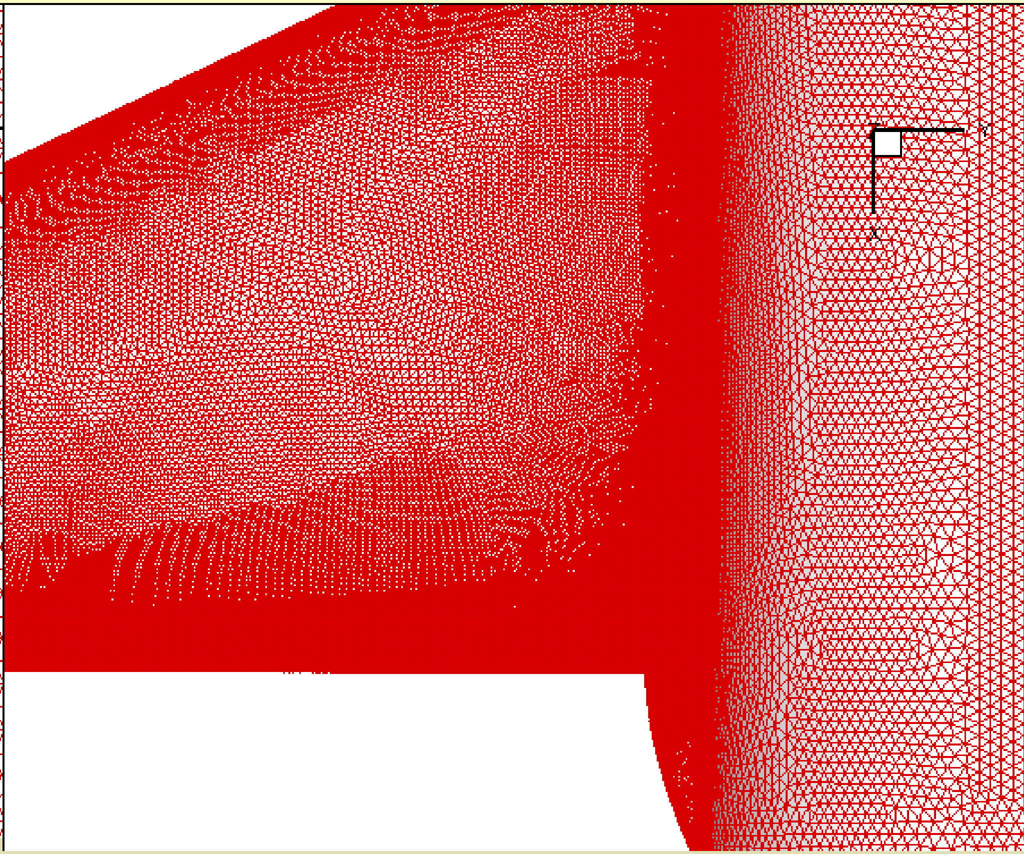
VGRID Node Centered (NASA)



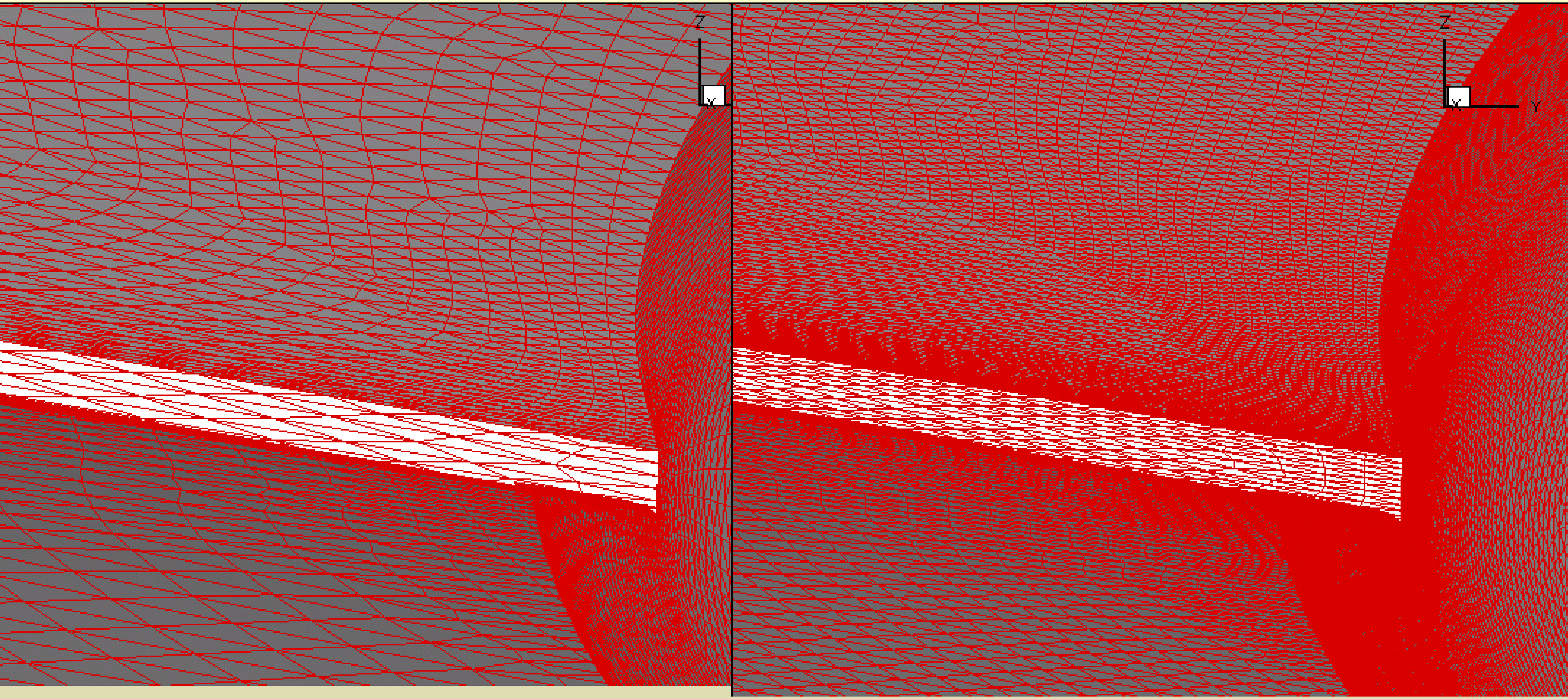


VGRID Node Centered (NASA)





VGRID Node Centered (NASA)



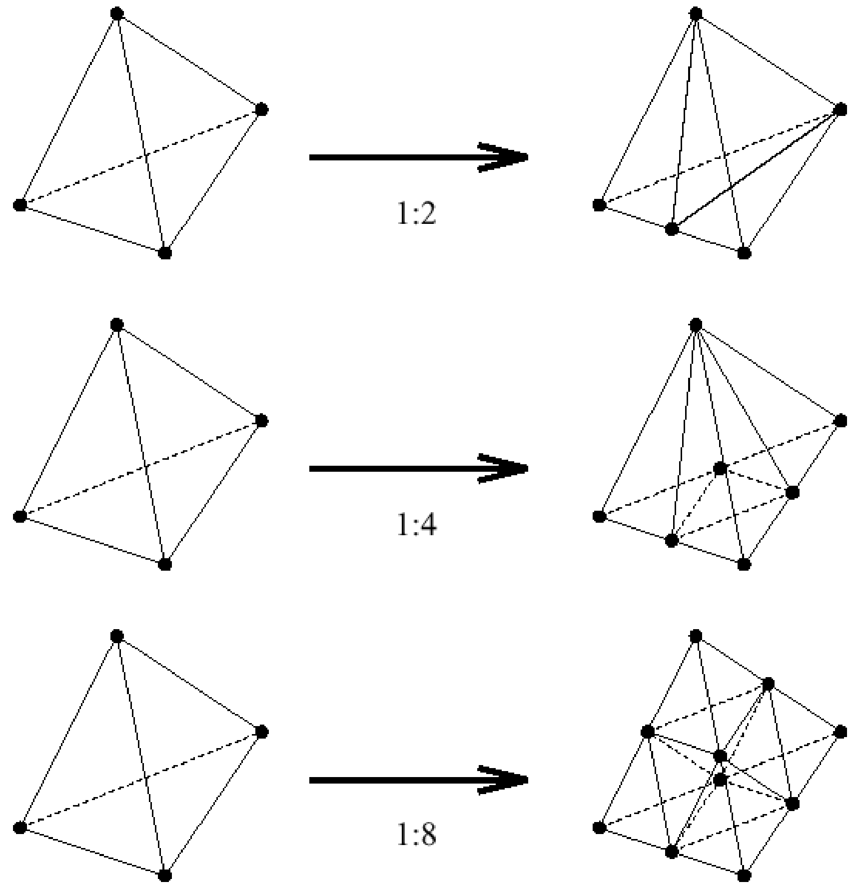
Grid Resolution

- Experimentation with much finer grids still required to understand behavior
  - Current largest grids ~ 100M pts
  - One more refinement ~ 10^9 pts
  - Implications for off-design conditions
    - Full flight envelope simulations
    - Stability and control
- AMR is obvious path for achieving higher resolution at manageable cost
  - Refinement criterion is roadblock
    - A-posteriori error estimation

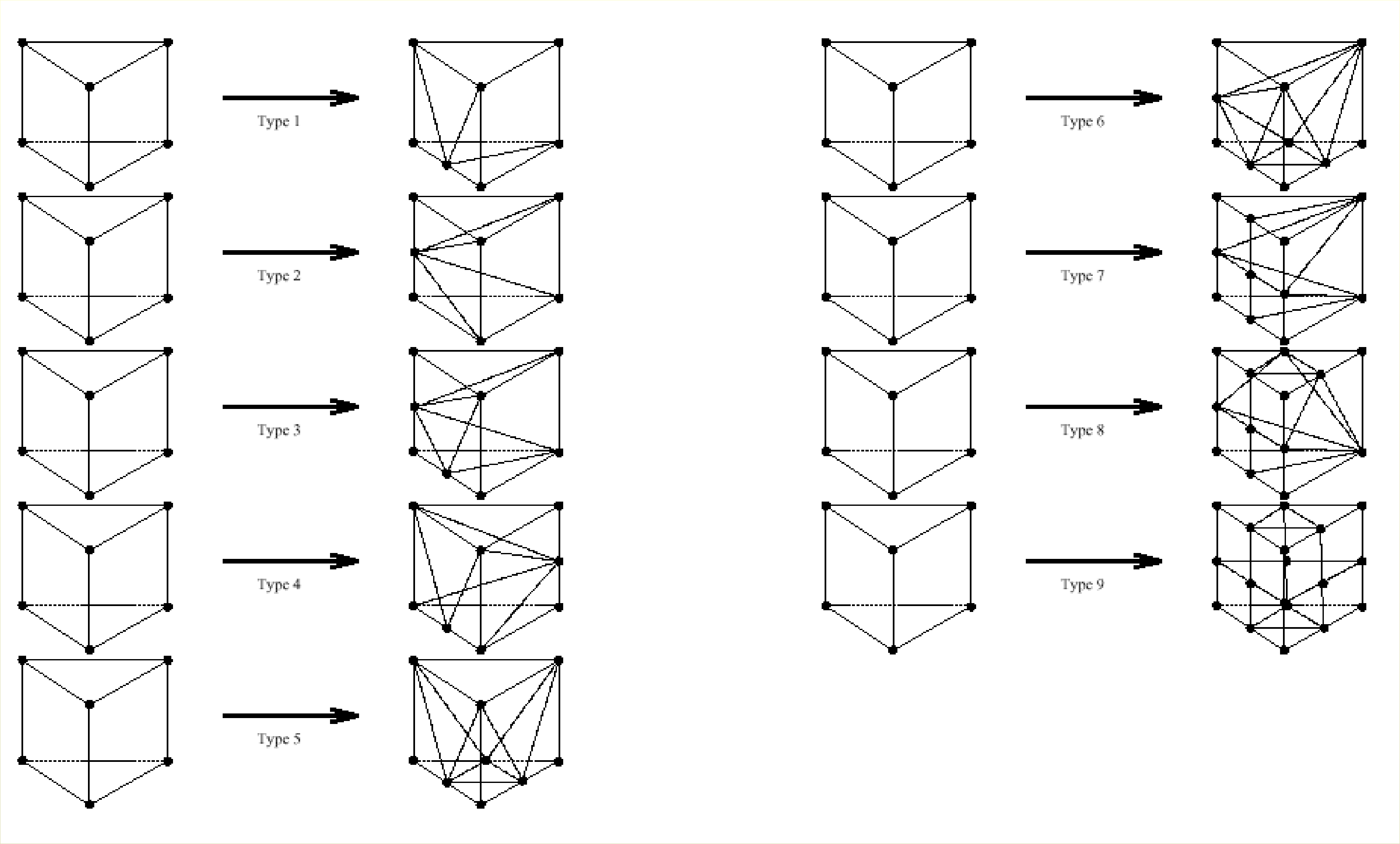
Adaptive Meshing Through Element Subdivision

- Standalone (sequential) code
- Flag Cells with Large Errors
- Subdivide these Cells (1:8)
- Adjust Cells in Transition Regions between Refined and Non-Refined Regions
  - (anisotropically refined cells)

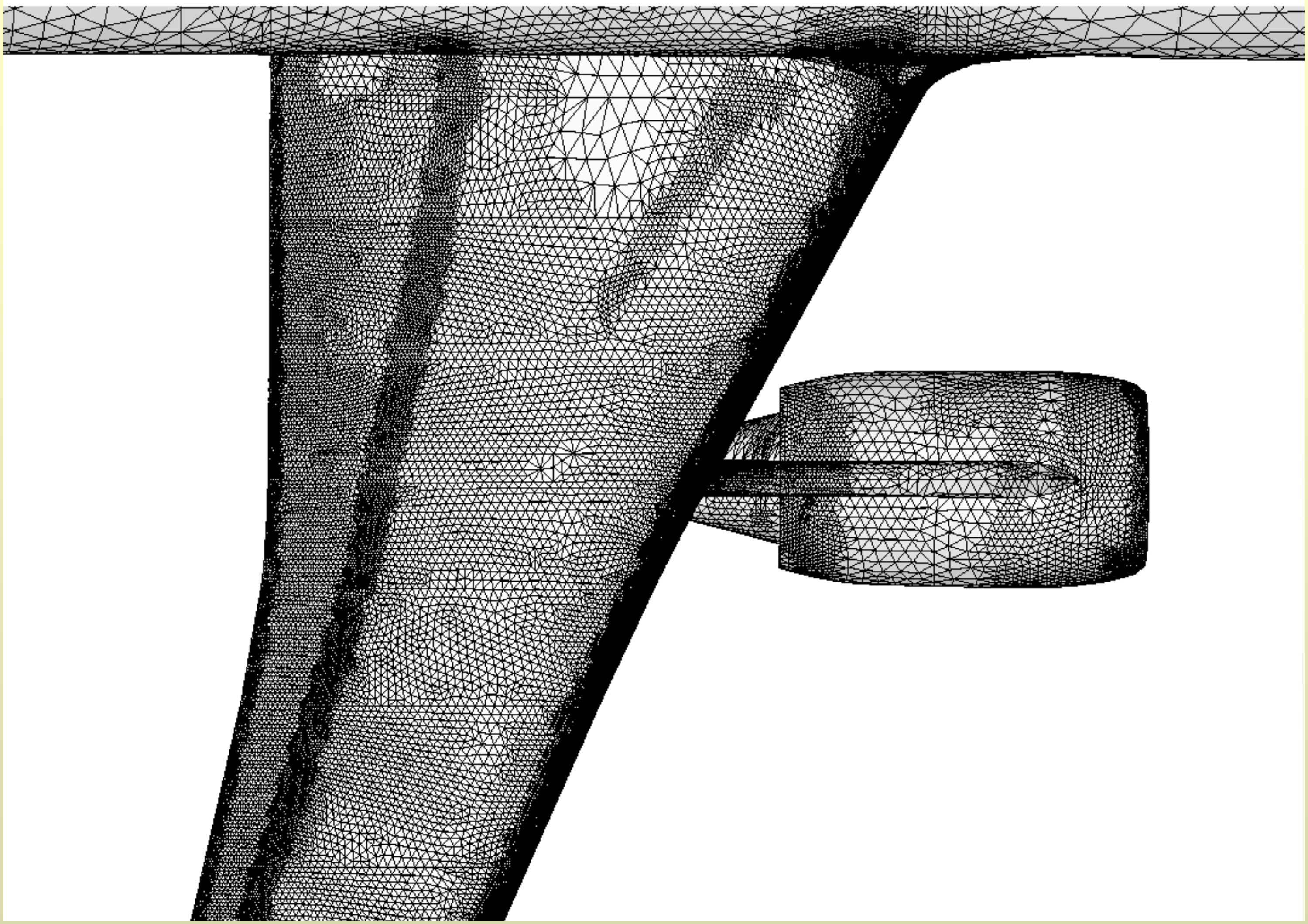
Subdivision Types for Tetrahedra



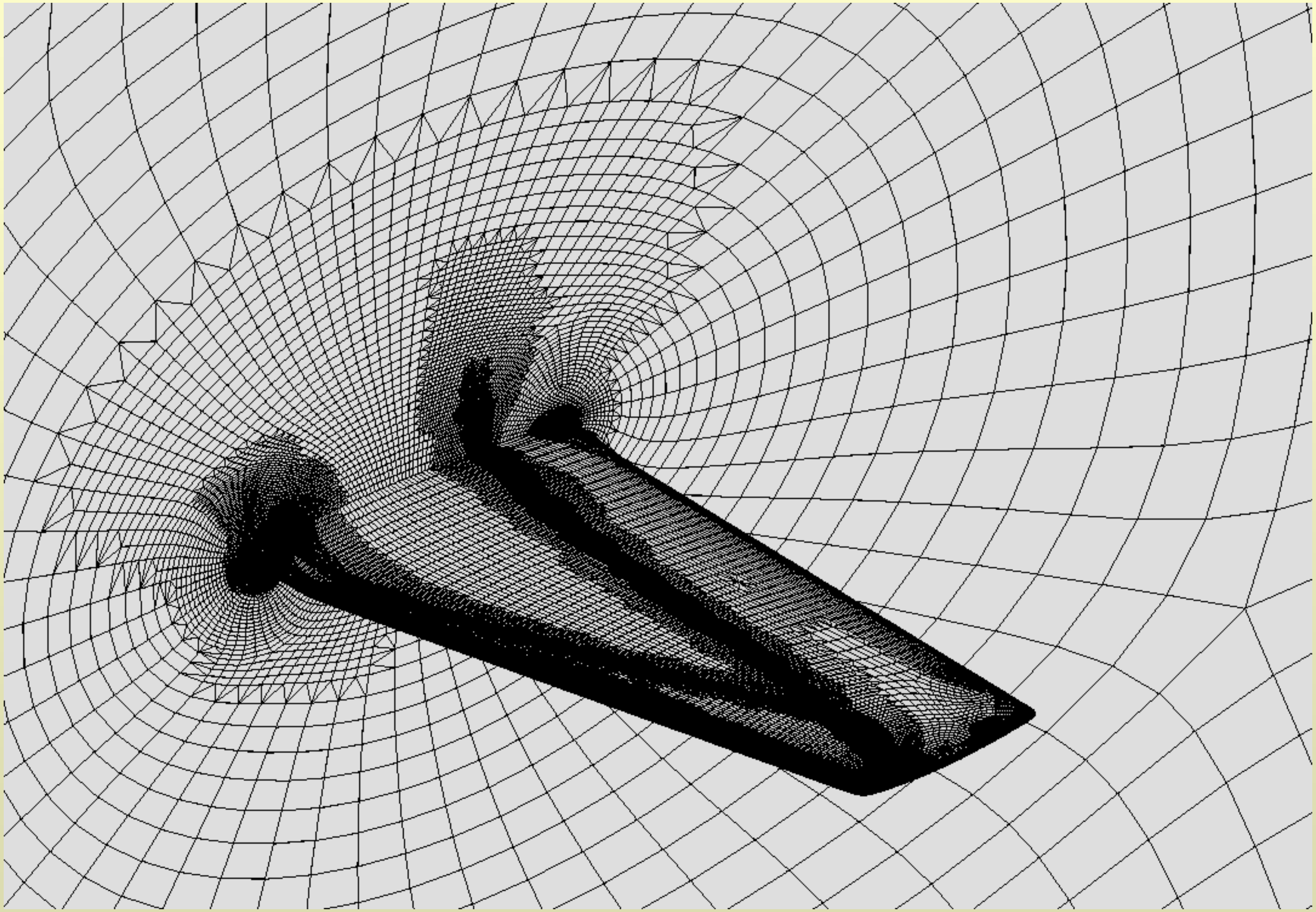
Subdivision Types for Prisms



Adaptive Tetrahedral Mesh by Subdivision



Adaptive Hexahedral Mesh by Subdivision



Adjoint-Based Error Estimation

- Complex simulations have multiple error sources
- Engineering simulations concerned with specific output objectives
- Adjoint methods / Goal Oriented Approach
  - Methodical approach for constructing discrete adjoint
  - Use for a posteriori error estimation
    - Spatial error
    - Temporal error
    - Other error sources
  - Use to drive adaptive process

Spatial Error Sensitivity

Fine level Taylor expansion of functional objective L:

$$L_h(U_h) = L_h(U_h^H) + \left[\frac{\partial L}{\partial U}\right](U_h - U_h^H)$$

h = fine grid (solution not available)
H = coarse grid (solution available)

Spatial Error Sensitivity

Fine level flow residual Taylor expansion

$$R_h(U_h) = \quad R_h(U_h^H) + \left[\frac{\partial R}{\partial U}\right]_{U_h^H} (U_h - U_h^H) = 0$$

$$U_h - U_h^H = -\left[\frac{\partial R}{\partial U}\right]^{-1} R_h(U_h^H)$$

Spatial Error Sensitivity

Fine level Taylor expansion of functional objective L:

$$L_h(U_h) = L_h(U_h^H) - \left[\frac{\partial L}{\partial U}\right]\left[\frac{\partial R}{\partial U}\right]^{-1} R_h(U_h^H)$$

$$\Downarrow$$

$$L_h(U_h) - L_h(U_h^H) = -\Lambda^T . R_h(U_h^H)$$

$$\left[\frac{\partial R}{\partial U}\right]^T \Lambda = -\left[\frac{\partial L}{\partial U}\right]^T$$ Adjoint problem

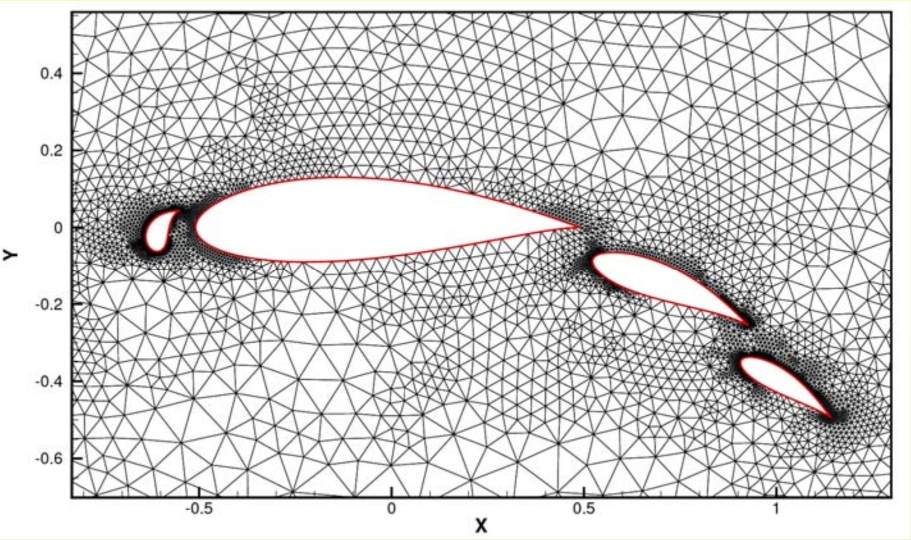
h = fine grid (solution not available)
H = coarse grid (solution available)

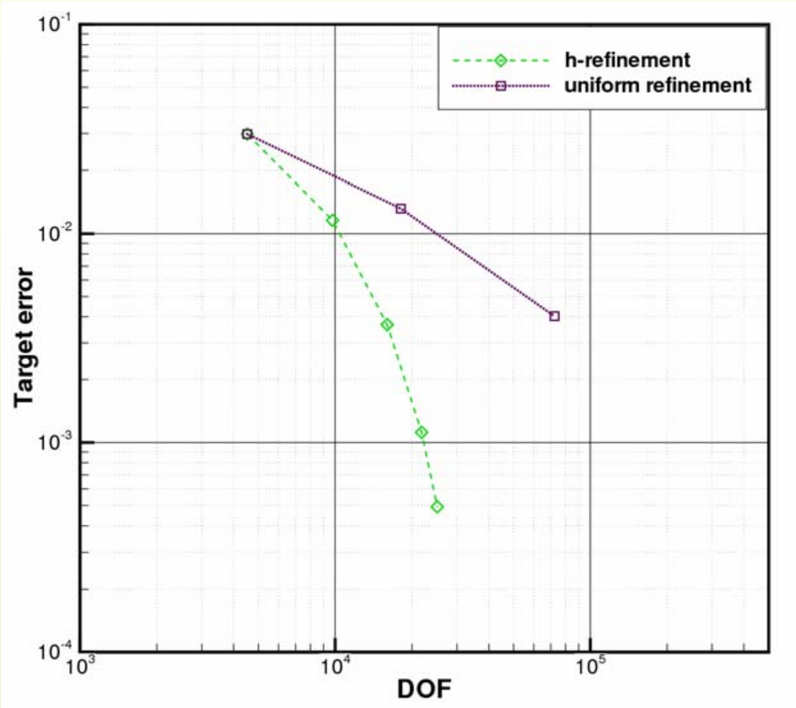
Spatial Error Sensitivity

$$L_h(U_h) - L_h(U_h^H) = -\Lambda^T . R_h(U_h^H)$$

- Compute objective on current grid
- Compute adjoint on current grid
- Project solution on to finer grid, evaluate residual (non-zero)
- Change in Objective on fine grid is inner product of adjoint with residual on fine grid
  - Cost: 1 flow solve, 1 adjoint solve on coarse grid
  - Gain: Prediction of objective on fine grid (8 times more grid points)
    - Use to certify sensitivity of solution to finer grid
    - Use to drive adaptive meshing (for specific objective)

Goal-Oriented Spatial Adaptivity (steady-state)





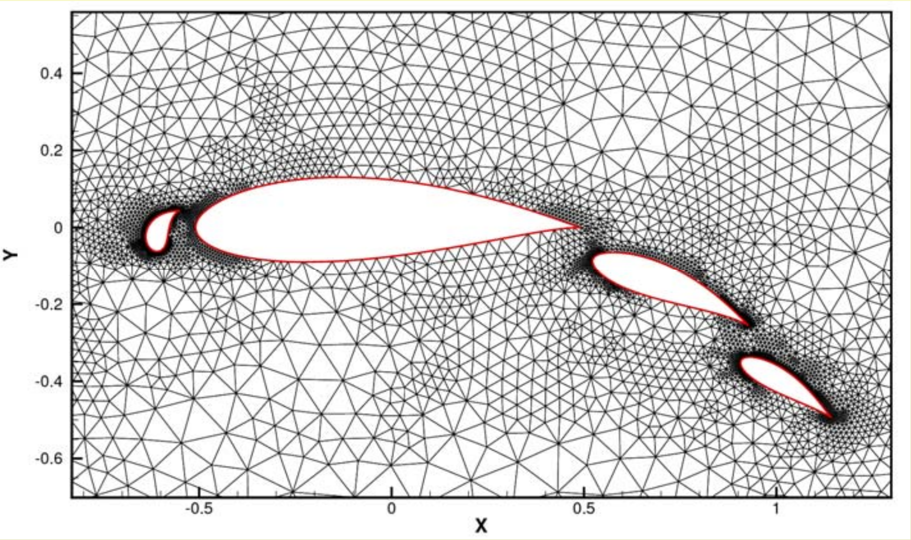
Adaptivity based on Drag

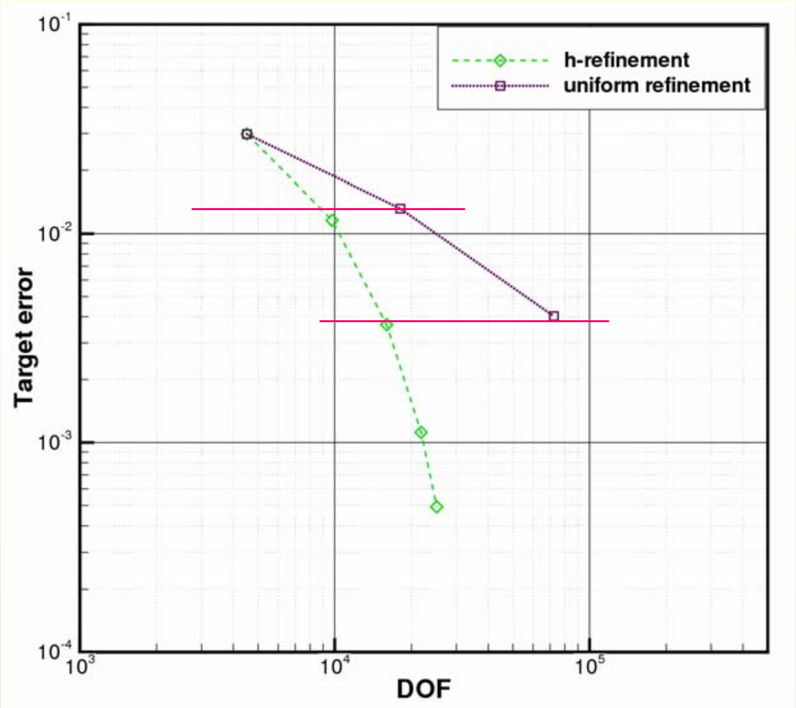
See also: Venditti and Darmofal; AIAA-2001

Giles et al; AIAA-2001

Park, M.   AIAA-2002

Goal-Oriented Spatial Adaptivity (steady-state)





Adaptivity based on Drag

See also: Venditti and Darmofal; AIAA-2001

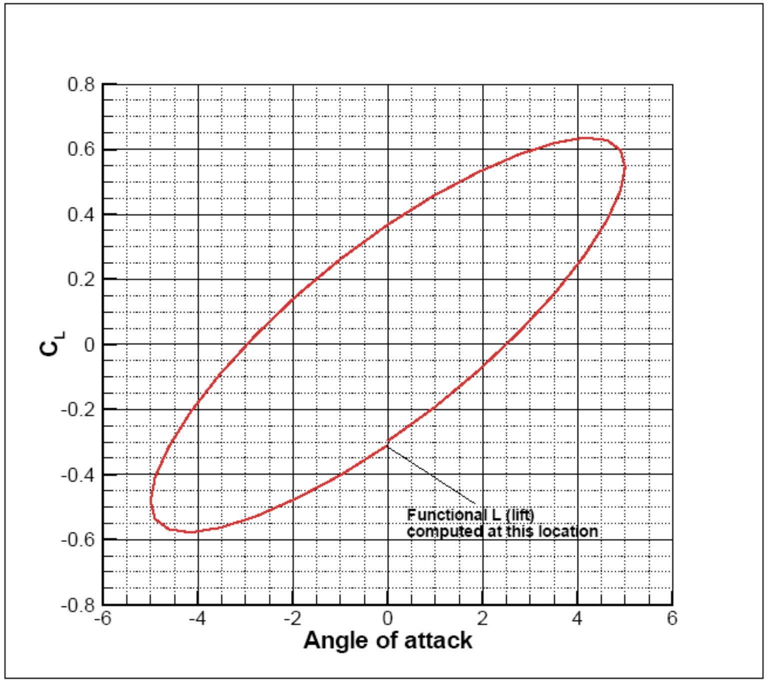
Giles et al; AIAA-2001

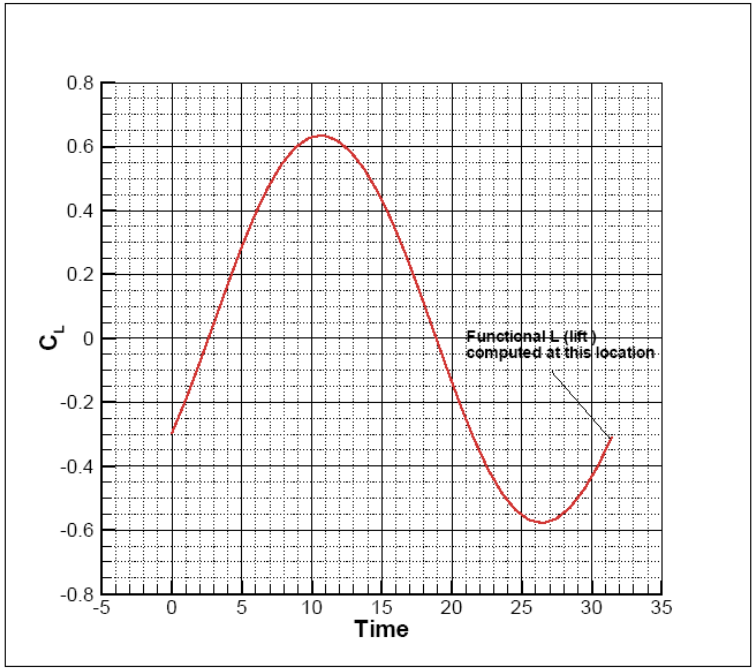
Park, M.   AIAA-2002

Error Estimation for Time Dependent Problems

Test Case Description

Sinusoidally pitching airfoil
Functional scalar is Lift after 1 period





Easily extended to estimate error in time-integrated Lift history

Summary of Temporal Resolution Error Evaluation

- Compute unsteady flow solution on coarse time domain
- Compute adjoint variables on coarse time domain
  - Integrating backward in time
- Project adjoint variables, flow solution and mesh solution onto fine time domain
- Temporal resolution error is then inner product of adjoint with corresponding non-zero residual on fine time domain
  - Distribution in time is used to drive adaptation

Validation

Description	Functional value	% Error vs. Target	% Predicted Error
Target functional - exact at 32 steps (fully converged)	-0.309957065250867	-	-
Fully converged flow and mesh at 16 steps	-0.285768366164898	+7.804	+7.463
Corrected for resolution from 16 to 32 steps	-0.3089006774509025	+0.341	-

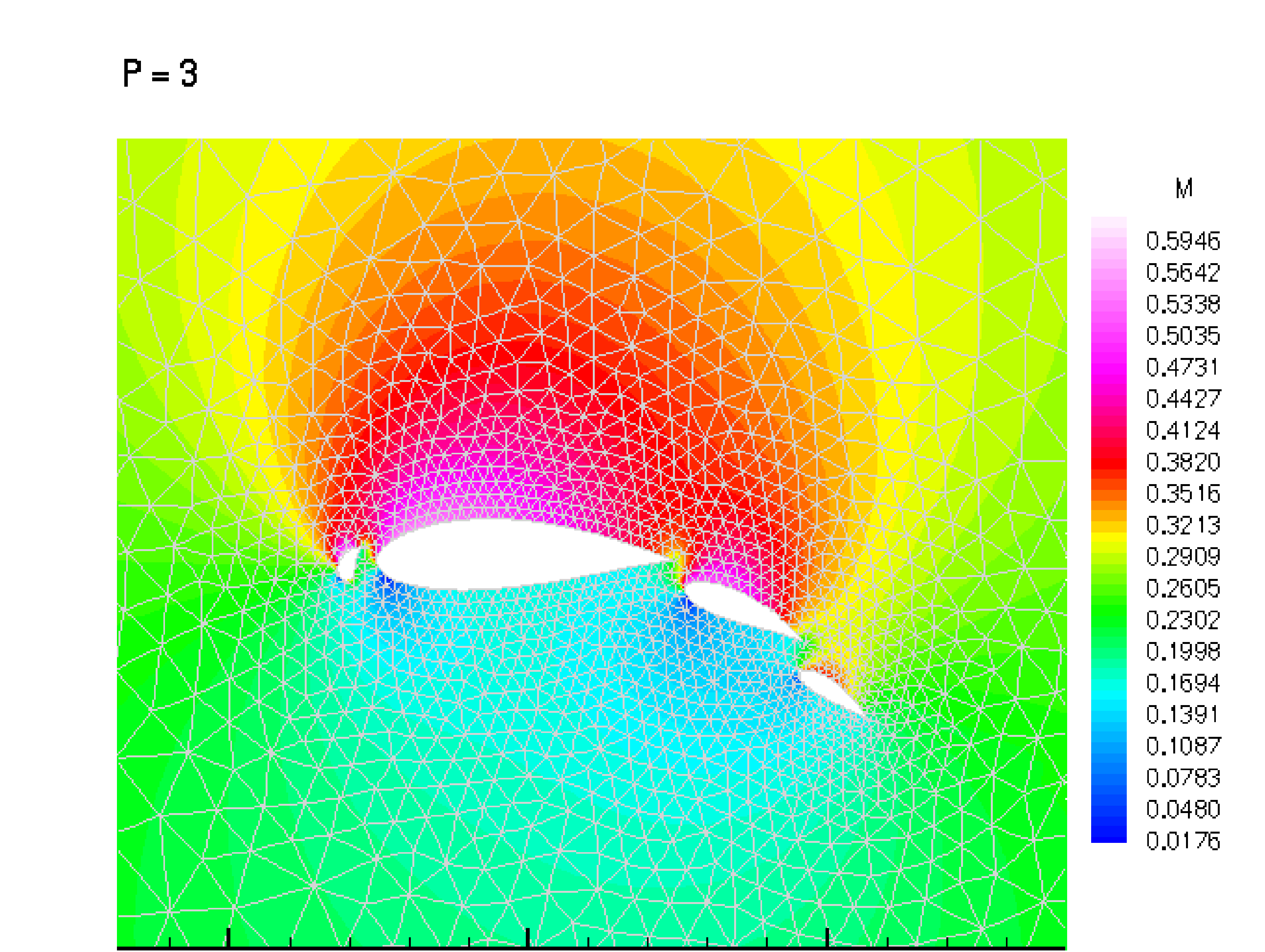
•Adjoint is linearization about current state (16 time steps) to predict objective value on modified state (32 time steps)

High Order Methods

- Higher order methods such as Discontinuous Galerkin best suited to meet high accuracy requirements
  - Asymptotic properties
- HOMs reduce grid generation requirements
- HOMs reduce grid handling infrastructure
  - Dynamic load balancing
- Compact data representation (data compression)
  - Smaller number of modal coefficients versus large number of point-wise values

- HOMs scale very well on massively parallel architectures

4-Element Airfoil (Euler Solution)



Single Grid Steady-State Implicit Solver

- Steady state

$$\boxed{R_p(U_p) = S_p}$$

- Newton iteration

$$\left[\frac{\partial R_p}{\partial U_p}\right]^n \Delta U_p^{n+1} = S_p - R_p(U_p^n)$$

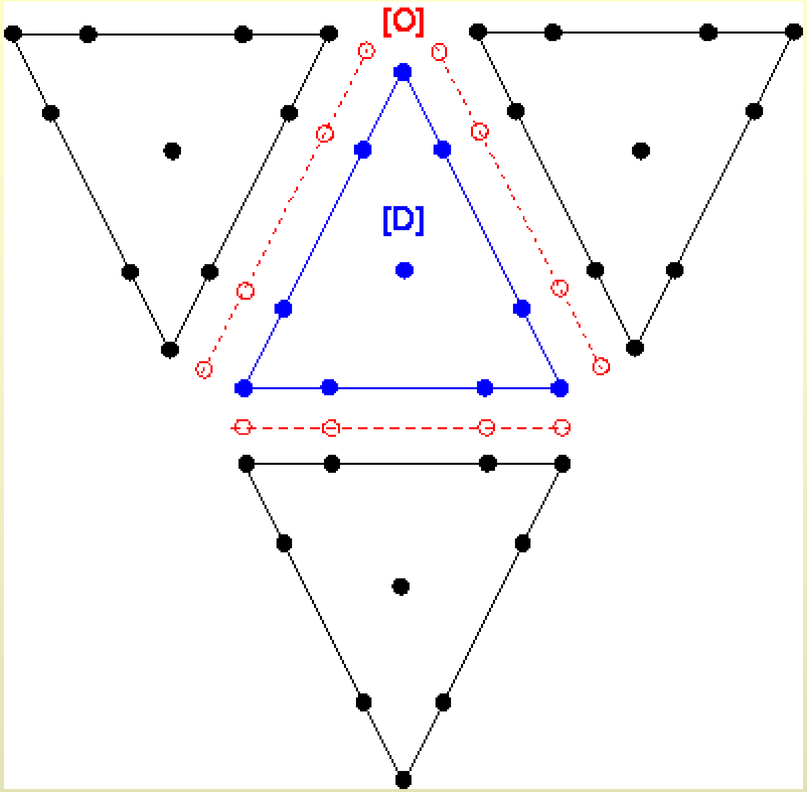
- Non-linear update

$$U_p^{n+1} = U_p^n + \Delta U_p^{n+1}$$

- [D] is Jacobian approximation

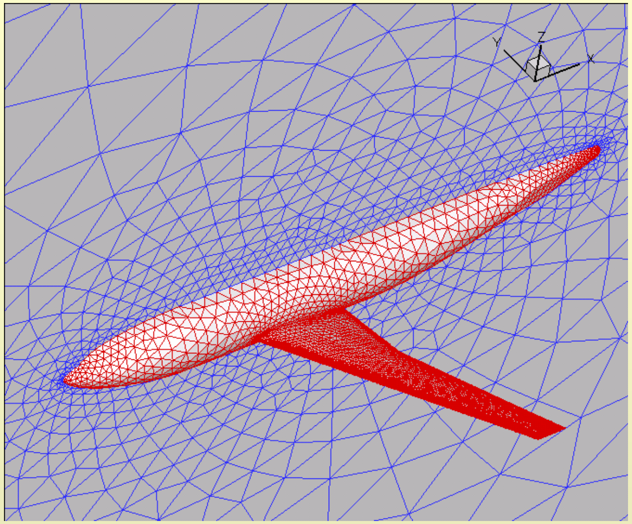
- Non-linear element-Jacobi (NEJ)

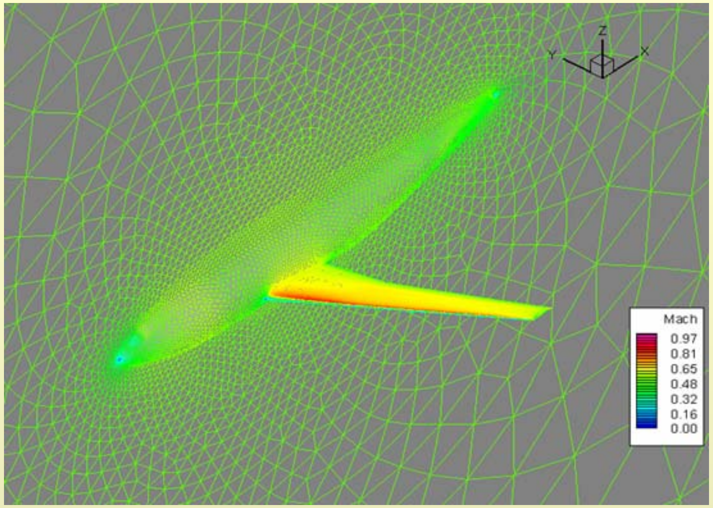
$$\boxed{\Delta U_p^{n+1} = \left[D_p^n\right]^{-1}\left(S_p - R_p(U_p^n)\right)}$$

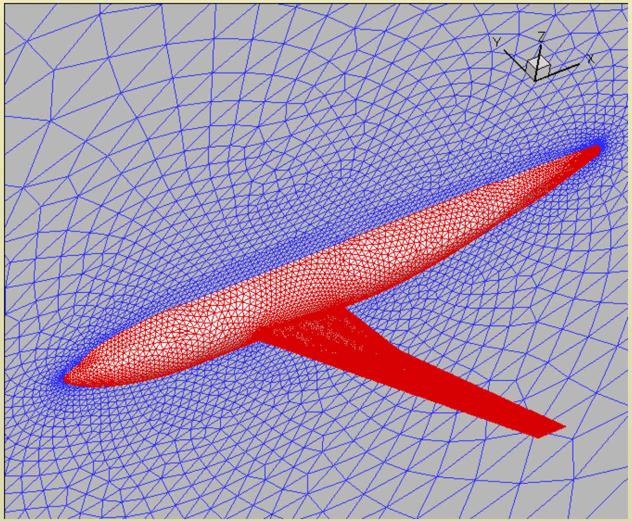


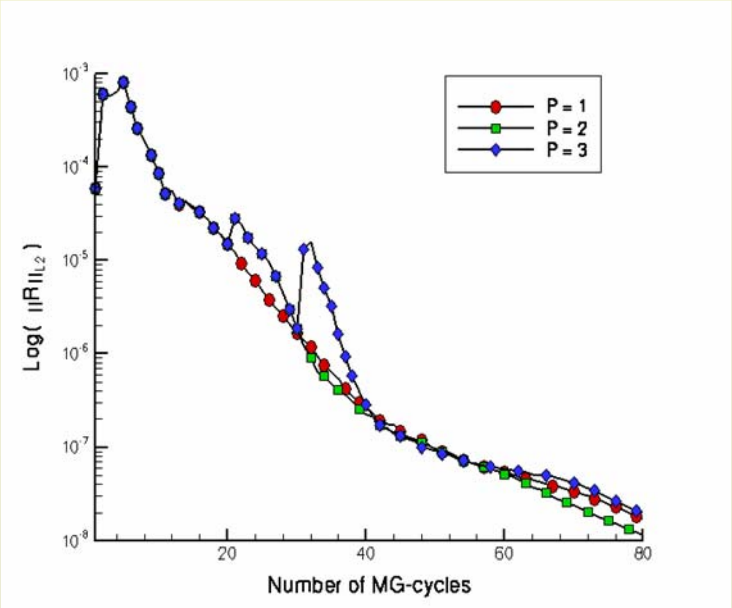
p	Size of [D]
1	5 x 5
2	20 x 20
3	50 x 50
4	100 x 100
5	280 x 280
6	420 x 420

3D High-Order DG Results (Inviscid)



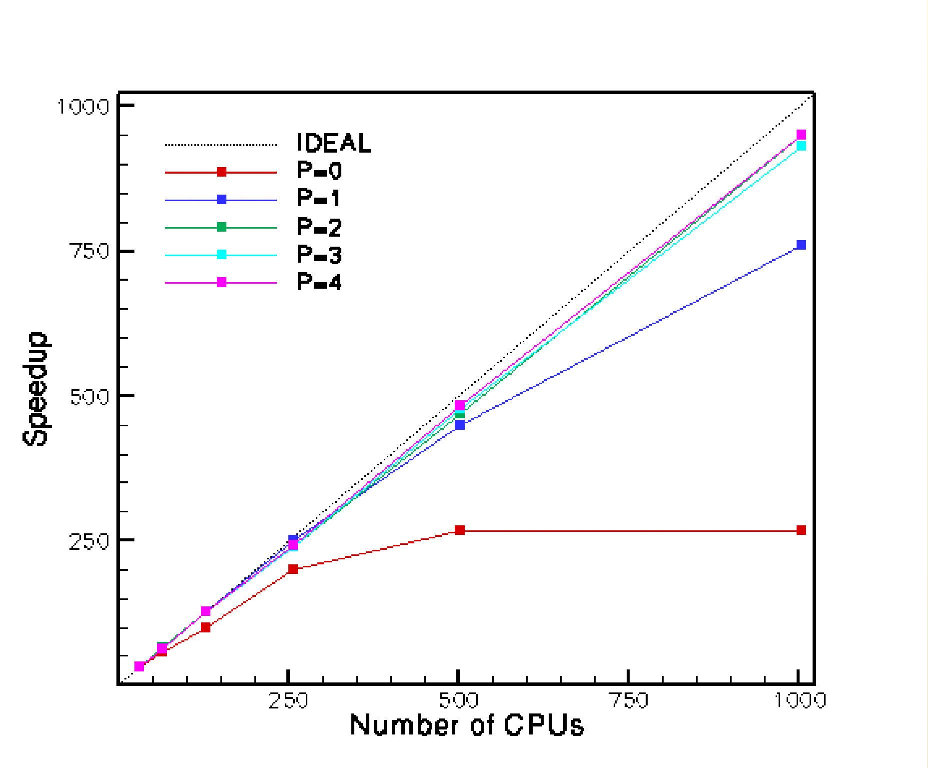






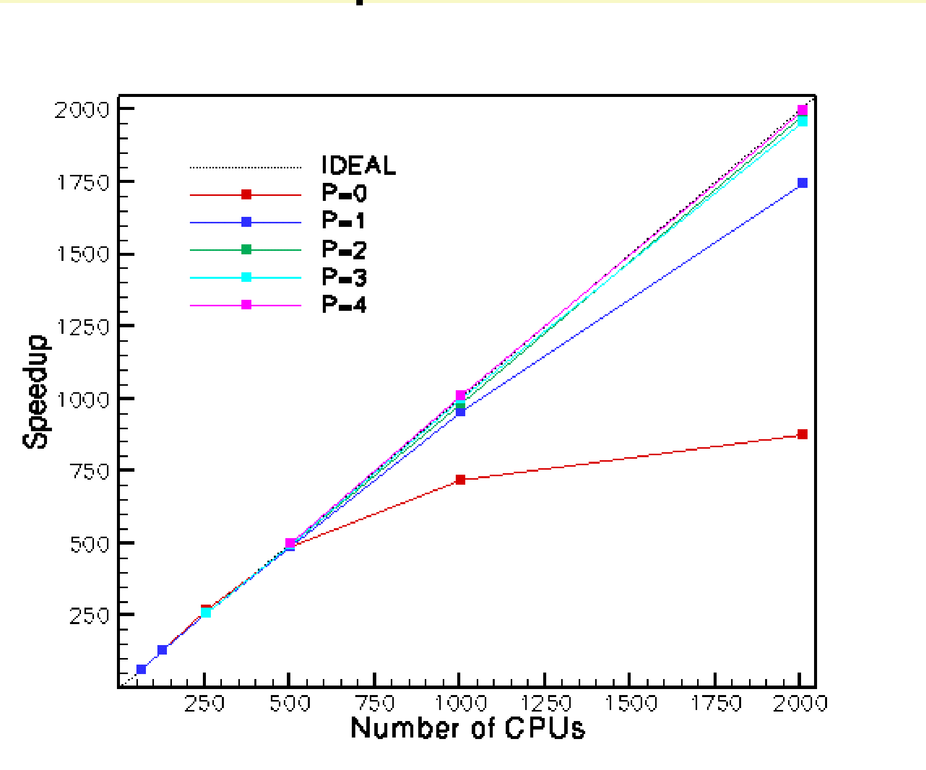
Parallel Performance: Speedup (1 MG-cycle)

185K pt mesh



- *p*=0 does not scale
- *p*=1 scales up to 500 proc.
- *p*>1 scales almost optimal

2.5M pt mesh



- *p*=0 does not scale
- *p*=1 scales up to 1000 proc.
- *p*>1 ideal scalability

Conclusions

- Current unstructured mesh solver technology is close to structured mesh solver capabilities
- All methods suffer from poorly quantified error sources
  - Particularly spatial discretization error
- Path forward will require new technology rooted in applied math
  - Sensitivity analysis techniques
  - Higher-order methods