



A Comparison of Edge-Based Scheme Newton-Krylov Method Operators for Solving Compressible Flows

Thomas M. Smith

Russell W. Hooper, Curtis C. Ober and Alfred A. Lorber

Sandia National Laboratories

Albuquerque, NM

SIAM Conference on Computational Science and Engineering

February 12-15,

Orlando, FL



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000



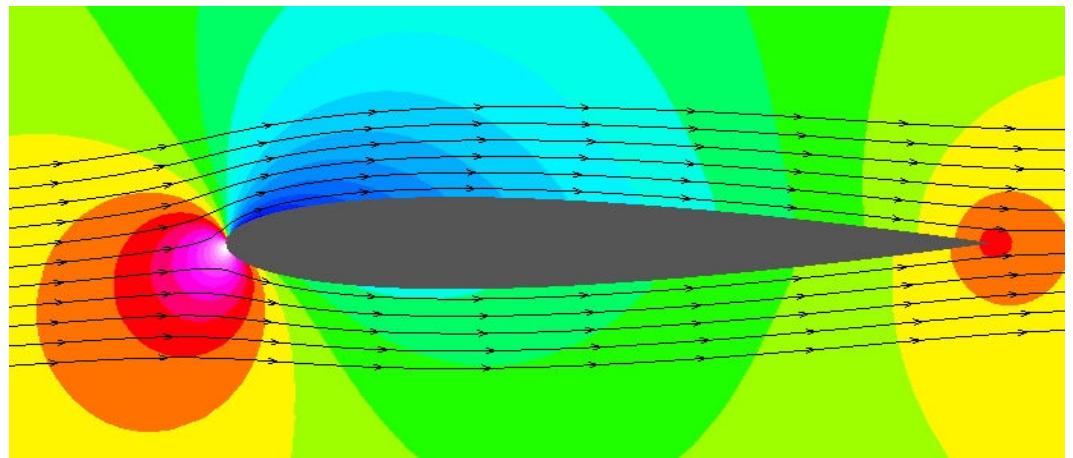


Primary Goal



Develop production capabilities to perform compressible flow simulations for the Nuclear Weapons Complex.

- Compressible subsonic through hypersonic
- Laminar through turbulent regimes
- Inviscid and viscous flows
- Steady state and transient
- Chemically reacting flow
- Multi-physics coupling
- Arbitrary body motion
- Mesh Adaptivity





SIERRA Framework



- SIERRA Framework
 - Unstructured mesh finite-element infrastructure
 - Parallel services
 - H-Adaptivity
 - Load balancing
 - I/O services (restart)
 - Scalable linear/nonlinear equation solver interface
 - Trilinos/Nox
 - Common interfaces and transfer operators for multi-physics code coupling
- SIERRA Code Management System (SCMS)
 - Configuration management (archiving, building, and testing software across multiple platforms)
 - Software Engineering infrastructure (requirements management, issue tracking, ...)





Edge-Based Assembly

- Spatial Discretization**

- Node centered finite-volume method
- Median dual mesh: volumes stored on nodes, area vector stored on edges
- 1st and 2nd order upwind advection

- Second-order extrapolation (MUSCL, van Leer)**

$$u_i^+ = u_i + \frac{1}{2} \Phi(\Delta_i^+, \Delta_i^-) \nabla u_i \bullet \Delta \mathbf{r}_{ij}$$

$$u_j^- = u_j - \frac{1}{2} \Phi(\Delta_j^+, \Delta_j^-) \nabla u_j \bullet \Delta \mathbf{r}_{ij}$$

$$\Delta \mathbf{r}_{ij} = (\mathbf{r}_j - \mathbf{r}_i)$$

$$\Phi_i \in [0, 1]$$

- Residual and matrix assembly**

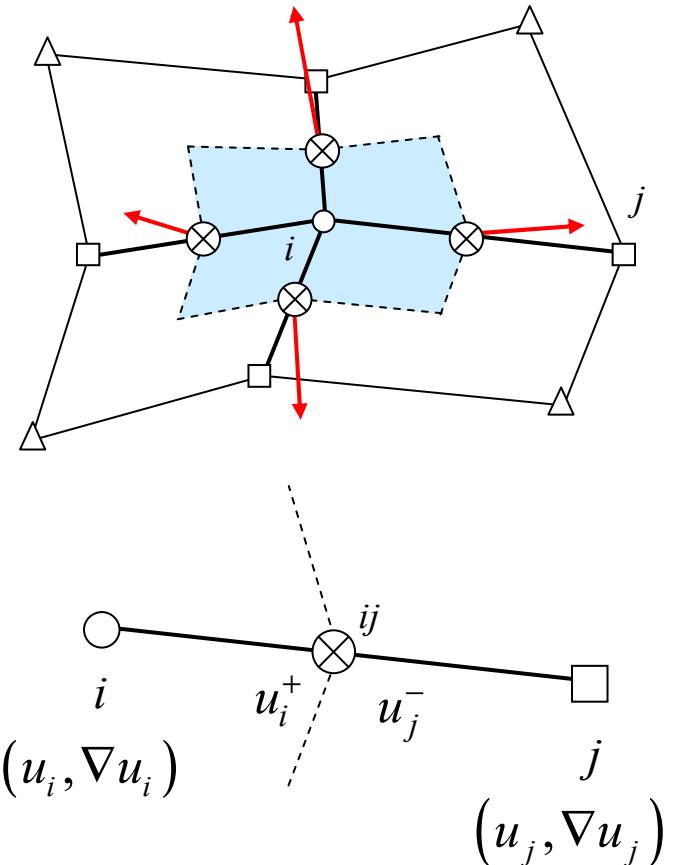
- Gradients are pre-assembled and stored on nodes
- Limiters are computed on edges

$$\nabla u_i = \frac{1}{\Delta V_i} \sum_{j \in NE(i)} \frac{1}{2} (u_i + u_j) \bullet \mathbf{n}_{ij} |S_{ij}|$$

$$(u_x)_i = \sum_{j \in NE(i)} W_{ij}^x (u_j - u_i)$$

$$(u_y)_i = \sum_{j \in NE(i)} W_{ij}^y (u_j - u_i)$$

$$(u_z)_i = \sum_{j \in NE(i)} W_{ij}^z (u_j - u_i)$$



$$\Delta_i^+ = \Delta_j^- = u_j - u_i$$

$$\Delta_i^- = 2 \nabla u_i \bullet \Delta \mathbf{r}_{ij} - (u_j - u_i)$$

$$\Delta_j^+ = 2 \nabla u_j \bullet \Delta \mathbf{r}_{ij} - (u_j - u_i)$$

$$\Phi(a, b) = \frac{ab + |ab| + \varepsilon}{a^2 + b^2 + \varepsilon}$$



Edge-Based Matrix Assembly: Residual



- Conservation Law form of Governing Equations

$$\int_V \frac{\partial \mathbf{W}}{\partial t} dV + \oint_S (\mathbf{F}^c + \mathbf{F}^v) \bullet \mathbf{n} dS = \int_V \mathbf{H} dV$$

- Semi-Discrete Equations (Residuals)

$$\mathbf{G}_i = \frac{\partial \mathbf{W}_i}{\partial t} \Delta V_i + \sum_{j \in NE(i)} \left[\mathbf{F}_{ij}^c - \mathbf{F}_{ij}^v \right] \left| \mathbf{S}_{ij} \right| + \mathbf{H}_i = 0, \quad \forall i \in \Omega(V) \quad \mathbf{W}_i \Delta V_i \approx \int_{V_i} \mathbf{W} dV$$

- Roe's Approximate Riemann flux function

$$\mathbf{F}_{ij}^c = \frac{1}{2} \left[\mathbf{F}(\mathbf{W}_i^+, \mathbf{n}_{ij}) + \mathbf{F}(\mathbf{W}_j^-, \mathbf{n}_{ij}) - \left| A(\mathbf{W}_i^+, \mathbf{W}_j^-, \mathbf{n}_{ij}) \right| (\mathbf{W}_j^- - \mathbf{W}_i^+) \right]$$

- Viscous Fluxes require construction of gradients at edge midpoints

- Premo blends node gradients with the node difference,
(Rie and Chow, O'Rourke)

$$\mathbf{F}_{ij}^v = \mathbf{F}(\mathbf{U}_i, \nabla \mathbf{U}_i, \mathbf{U}_j, \nabla \mathbf{U}_j, \mathbf{n}_{ij}) \quad \nabla u_{ij} \approx \frac{(\nabla u_i + \nabla u_j)}{2} + \left(u_j - u_i - \frac{(\nabla u_i + \nabla u_j)}{2} \bullet \Delta \mathbf{r}_{ij} \right) \frac{\Delta \mathbf{r}_{ij}}{|\Delta \mathbf{r}_{ij}|^2}$$



Boundary Condition Enforcement



- **Weak** (Add flux from boundary surface)
 - Performed during other flux calculations
 - Similar to interior flux contributions

$$\frac{\partial \mathbf{W}_b}{\partial t} \Delta V_b + \sum_{j \in NE(b)} [\mathbf{F}_{ij}^c - \mathbf{F}_{ij}^v] \bullet \mathbf{n}_{ij} |\mathbf{S}_{ij}| + \mathbf{H}_b + \mathbf{F}_b \bullet \mathbf{n}_b |\mathbf{S}_b| = 0, \quad \forall i \in \Omega(\Gamma)$$

- **UR – constant**
- **UR – U(Ub)**

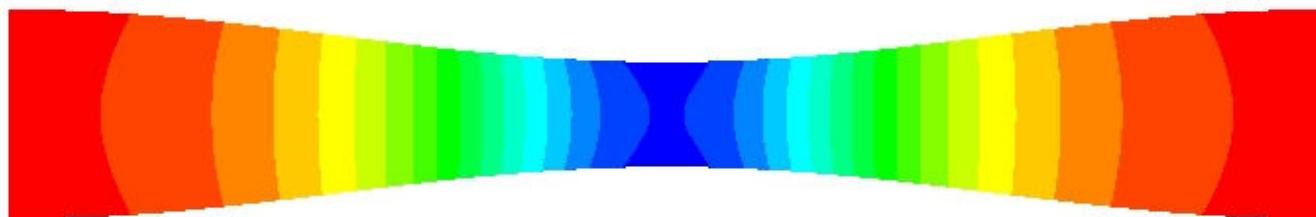
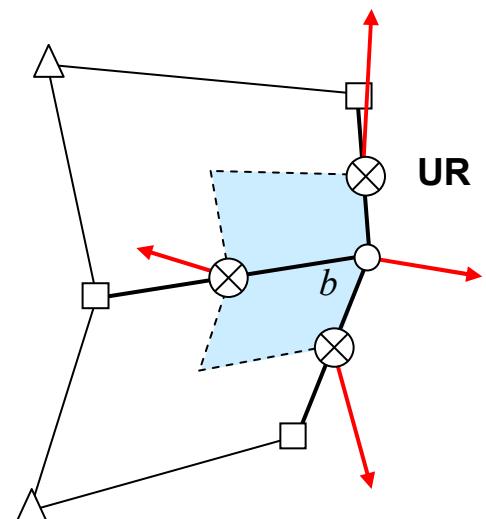
- **Residual** (Set the residual (or ΔU) to obtain U)
 - Performed after all other flux calculations
 - Replace flux balance with modified residual equation

$$\delta \mathbf{W} = \mathbf{R}$$

$$\delta \mathbf{U} = \mathbf{M} \delta \mathbf{W}$$

$$\delta \mathbf{U}^* = \mathbf{U}_b - \mathbf{U}$$

$$\delta \mathbf{W}^* = \mathbf{M}^{-1} \delta \mathbf{U}^*$$





Solution Strategies

- Requirements
 - Time accurate
 - Minimize time to solution for steady-state problems
 - Robust for broad range of compressible flow problems containing a wide range of time scales
- Implementations
 - Four-stage low-storage Runge-Kutta scheme (2nd order accurate)
$$\frac{\partial \mathbf{W}_i}{\partial t} = R(\mathbf{W})$$
$$\mathbf{W}^{n+1} = \mathbf{W}^n + \alpha \Delta t R(\mathbf{W}^n)$$
 - Newton-Krylov method

$$G(\mathbf{W}) = \frac{\partial \mathbf{W}}{\partial t} - R(\mathbf{W})$$

$$G(\mathbf{W}^*) = 0$$

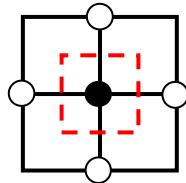
$$\mathbf{J} = \frac{\partial G}{\partial W}$$

$$\mathbf{J}(\mathbf{W}^{n+1,k}) \Delta \mathbf{W}^{k+1} = -G(\mathbf{W}^{n+1,k})$$

$$\mathbf{W}^{n+1,k+1} = \mathbf{W}^{n+1,k} + \Delta \mathbf{W}^{k+1}$$

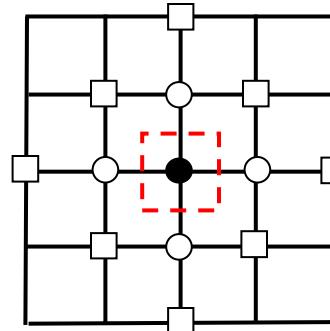


Matrix Graphs for NCFVM



Edge matrix graph for first-order Jacobian and first- and second-order residual

2D Quad – 5*DOF
3D Hex – 7*DOF



Distance-2 matrix graph for second-order Jacobian

D2 assembly requires;
1) Looping over edges,
2) Looping over edge nodes,
3) Storing information on edges,
4) Looping over edges sharing nodes
Not currently available in the framework!

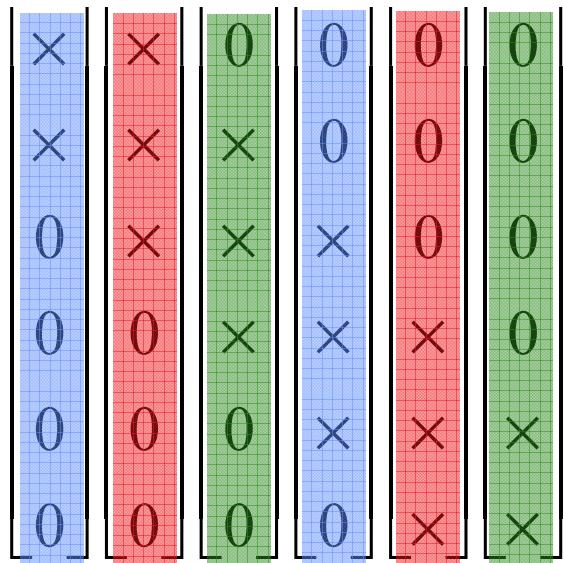
2D Quad – 13*DOF
3D Hex – 25*DOF

In both cases, the number of columns is greater for tetrahedral meshes



Example of coloring capability

Example: Tridiagonal matrix

$$J_{ij} = \frac{\partial R_i}{\partial x_j} \approx \frac{R_i(x + \epsilon e_j) - R_i(x)}{\epsilon}$$


$$C_1 = \{1, 4\},$$

$$C_2 = \{2, 5\},$$

$$C_3 = \{3, 6\}$$

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{c}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Currently uses Greedy Algorithm



Edge-Based Matrix Assembly



- Discrete Jacobian based only on edge-node data (first-order)

$$\frac{\partial \mathbf{G}_i}{\partial \mathbf{W}_i} = \mathbf{I} \frac{\Delta V_i}{\Delta t} + \sum_{j \in NE(i)} \frac{1}{2} \left(\frac{\partial \mathbf{F}^c(\mathbf{W}_i, \mathbf{n}_{ij})}{\partial \mathbf{W}_i} + |\mathbf{A}^c(\mathbf{W}_i, \mathbf{W}_j, \mathbf{n}_{ij})| + \frac{\partial |\mathbf{A}^c(\mathbf{W}_i, \mathbf{W}_j, \mathbf{n}_{ij})|}{\partial \mathbf{W}_i} (\mathbf{W}_j - \mathbf{W}_i) - \frac{\partial \mathbf{F}^v}{\partial \mathbf{W}_i} \right) |\mathbf{S}_{ij}|$$

$$\frac{\partial \mathbf{G}_i}{\partial \mathbf{W}_j} = \sum_{j \in NE(i)} \frac{1}{2} \left(\frac{\partial \mathbf{F}^c(\mathbf{W}_j, \mathbf{n}_{ij})}{\partial \mathbf{W}_j} - |\mathbf{A}^c(\mathbf{W}_i, \mathbf{W}_j, \mathbf{n}_{ij})| - \frac{\partial |\mathbf{A}^c(\mathbf{W}_i, \mathbf{W}_j, \mathbf{n}_{ij})|}{\partial \mathbf{W}_j} (\mathbf{W}_j - \mathbf{W}_i) - \frac{\partial \mathbf{F}^v}{\partial \mathbf{W}_j} \right) |\mathbf{S}_{ij}|$$

$$\frac{\partial \mathbf{G}_b}{\partial \mathbf{W}_b} = \mathbf{I} \frac{\Delta V_b}{\Delta t} + \sum_{j \in NE(i)} \frac{1}{2} \left(\frac{\partial \mathbf{F}^c(\mathbf{W}_b, \mathbf{n}_b)}{\partial \mathbf{W}_b} + \frac{\partial \mathbf{F}^c(\mathbf{W}_R, \mathbf{n}_b)}{\partial \mathbf{W}_R} \frac{\partial \mathbf{W}_R}{\partial \mathbf{W}_b} \right. \\ \left. - |\mathbf{A}^c(\mathbf{W}_b, \mathbf{W}_R, \mathbf{n}_{ij})| \left(\frac{\partial \mathbf{W}_R}{\partial \mathbf{W}_b} - \mathbf{I} \right) + \frac{\partial}{\partial \mathbf{W}_b} |\mathbf{A}^c(\mathbf{W}_b, \mathbf{W}_R, \mathbf{n}_{ij})| (\mathbf{W}_R - \mathbf{W}_b) \right) |\mathbf{S}_b|$$

- Viscous Jacobian considers only the node differences

$$\frac{\partial \mathbf{F}_i^v}{\partial \mathbf{W}_j} \Rightarrow \quad \nabla u_{ij} \approx (u_j - u_i) \frac{\Delta \mathbf{r}_{ij}}{|\Delta \mathbf{r}_{ij}|^2}$$

- Viscosity, conductivity and the velocity in the energy dissipation are approximated by the average of the two node values



Operator Evaluation Cost

Operator	Hex mesh (1 elem)	Tet mesh (3D)
RHS	1	1
MF	1	1
AD	7.34	4.78
ROEFD	5.54	3.29
ROE	5.30	2.93
FDC	61/32.98	156/81.24

Operator	Euler (Hex mesh 1 elem)	NS (Hex mesh 1 elem)	SARAN S (Hex mesh 1 elem)
RHS	1	1	1
MF	1	1	1
AD	7.34	7.56	8.17
FDC	61/32.98	61/34.94	67/61.56



Solver Overview

Strategy	Jacobian, \tilde{J}	Residual, \tilde{F}
Transient	$\frac{1}{\Delta t} \mathbf{I} + \frac{\partial \mathbf{R}}{\partial \mathbf{W}}$	$\frac{\mathbf{W} - \mathbf{W}_{t-\Delta t}}{\Delta t} + \mathbf{R}$
Pseudo Transient	$\frac{1}{\Delta t} \mathbf{I} + \frac{\partial \mathbf{R}}{\partial \mathbf{W}}$	\mathbf{R}

$$\Delta t_{n+1} = \Delta t_n \frac{\| R_p^{\text{old}} \|_1}{\| R_p^{\text{new}} \|_1}$$

$$\Delta t_n = f_{CFL} \Delta t_{n-1}$$

NOX



At each step, solve:

$$\mathcal{J}\Delta \mathbf{W} = -\mathcal{G}^0$$



Operator	Form
Matrix-Free (MF)	$\mathcal{J}\mathbf{M}^{-1}\mathbf{p} \approx \frac{\mathcal{G}(\mathbf{W} + \varepsilon \mathbf{M}^{-1}\mathbf{p}) - \mathcal{G}(\mathbf{W})}{\varepsilon}$
Finite-Diff. w/ Coloring (FDC)	$J_{y(m)} \approx \frac{\mathcal{G}_i^0(\mathbf{W} + \delta \mathbf{c}_m) - \mathcal{G}_i^0(\mathbf{W})}{\delta}$
Analytic	AD, ROEFD, ROE, SD

Jac Only

Jac and/or Prec.

Jac and/or Prec.

For each NonLinear iteration, solve:

$$\mathcal{J}\mathbf{M}^{-1}\mathbf{y} = -\mathcal{G}^0$$

$$\mathbf{M}\Delta \mathbf{W} = \mathbf{y}$$

GMRES with Right-Preconditioning:



Inviscid Supersonic Flow



Flow Conditions:

M=2

Alpha=0

Mesh:

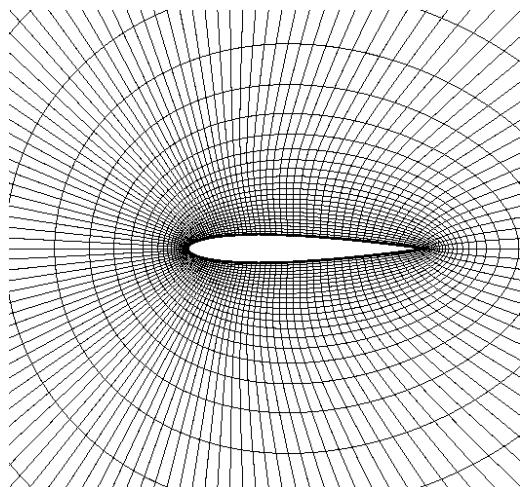
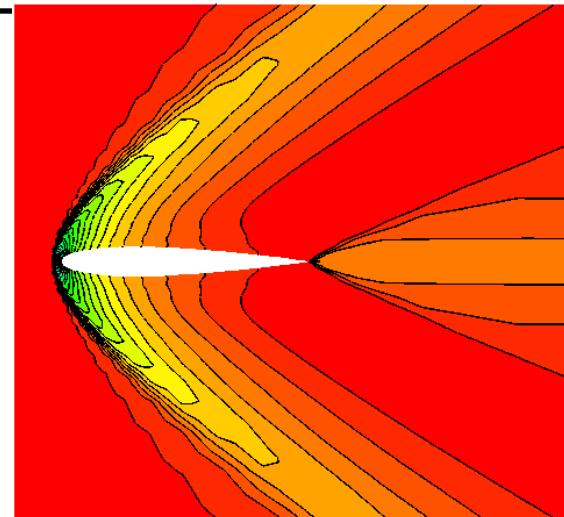
NACA0012 – O-Grid

129x33x2

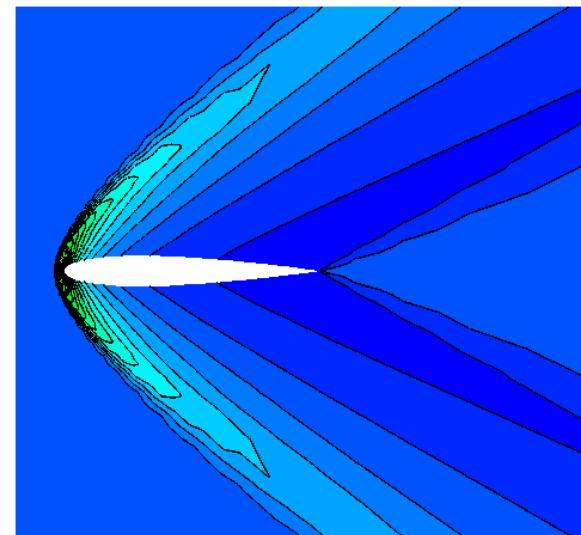
Boundaries 25 chords away from airfoil

Solver Settings:

Eta=10-1



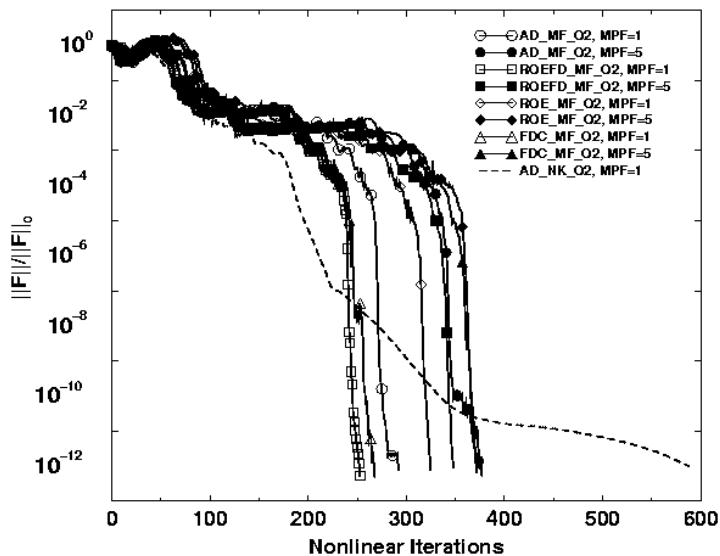
Mesh courtesy of
M. Berggren





Operator Evaluation Cost

	MPF	CFL	Adapt	Cpu	NLits	LIts
AD_MF_O2	1	1-5000	1.05	1260	293	1400
AD_MF_O2	5	1-5000	1.2	989	378	1633
ROEFD_MF_O2	1	1-5000	1.05	1082	254	1262
ROEFD_MF_O2	5	1-5000	1.2	886	349	1337
ROE_MF_O2	1	1-5000	1.05	1299	326	1453
ROE_MF_O2	5	1-5000	1.2	1000	373	1592
FDC_MF_O2	1	1-5000	1.05	2874	269	1909
FDC_MF_O2	5	1-5000	1.2	1526	376	1563
AD_NK_O2	1	1-50	1.05	1578	589	1918



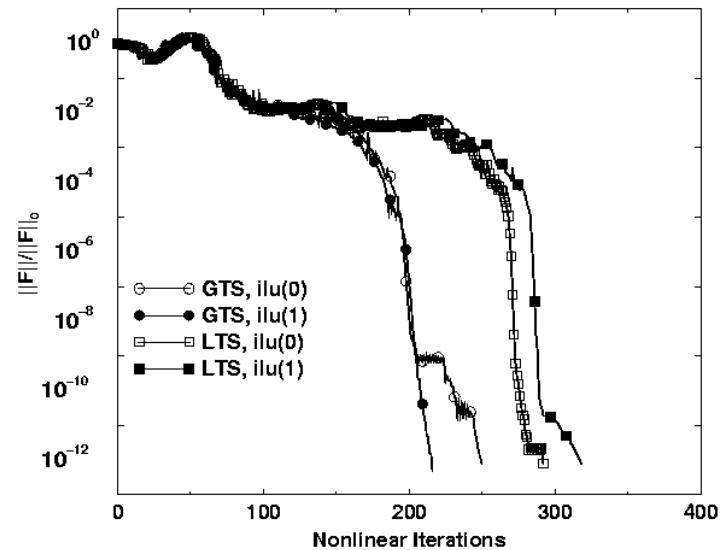
LTS
IIu(0), noRCM



Global vs. Local Timestepping



	RCM	CFL	Adapt	Cpu	NLits	LIts
GTS, ilu(0)	no	1-107	1.05	1271	251	1781
GTS, ilu(1)	yes	1-107	1.05	4281	217	1136
LTS, ilu(0)	no	1-5000	1.05	1260	293	1400
LTS, ilu(1)	yes	1-5000	1.05	6048	319	1240

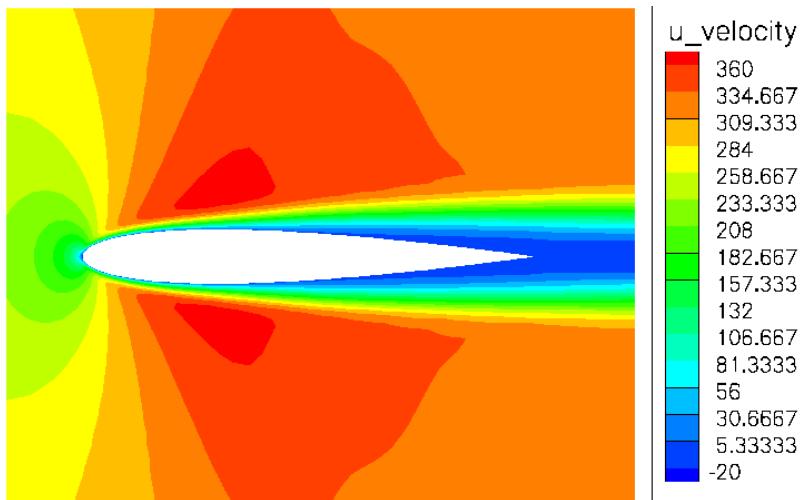


MPF=1

$$CFL^{n+1} = CFL^n f \frac{\|R_\rho^{n-1}\|_1}{\|R_\rho^n\|_1}$$



NACA0012 – Laminar



Flow Conditions:

M=0.85

Re=2000

Alpha=0

Mesh

Cmesh 193x65x2

Boundaries 10-25 chords

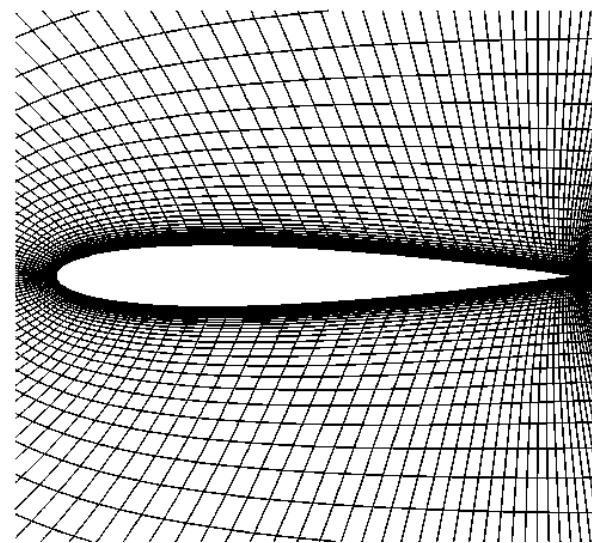
Away from airfoil

Solver settings:

CFL=10-1E7

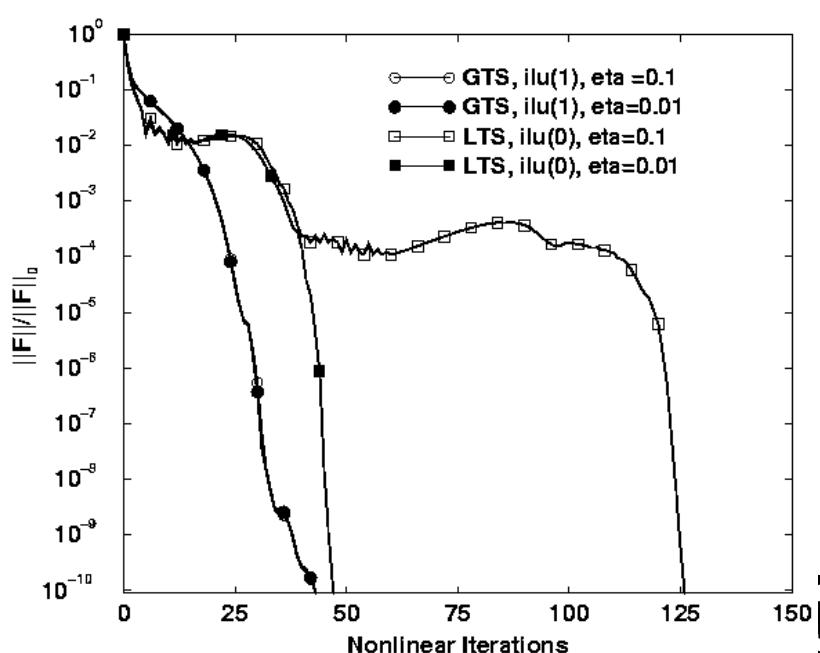
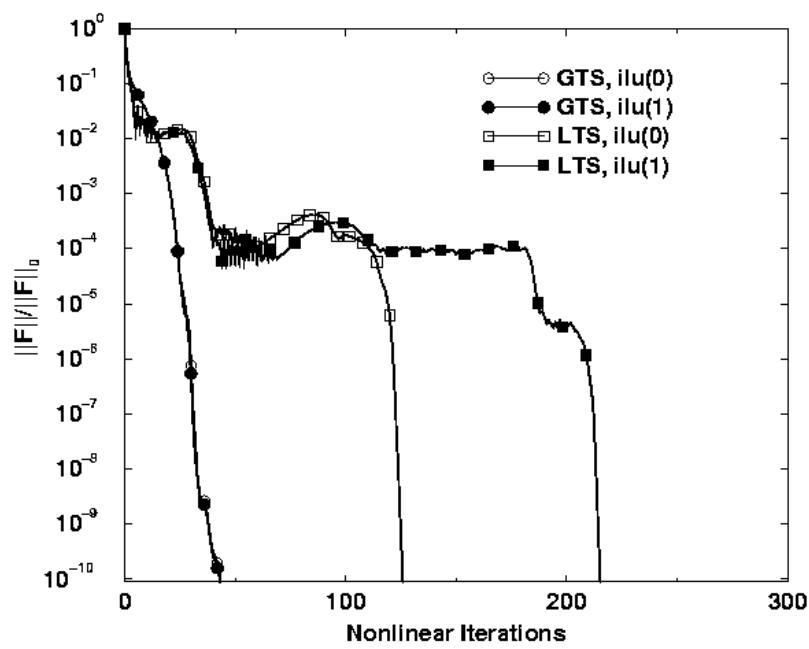
MPF=1

Adapt factor=1.1





	Ilu/RCM	Timestep	Eta	cpu	NLIts	LIts
11	0/no RCM	GTS	10-1	12211	45	7120
9	1/RCM	GTS	10-1	5410	45	2031
12	0/no RCM	LTS	10-1	6624	127	3808
10	1/ RCM	LTS	10-1	13413	217	1634
16	1/RCM	GTS	10-2	6216	44	2768
15	0/no RCM	LTS	10-2	6560	49	3741





Mesh Sequencing



Flow Conditions:

M=0.729

Alpha=2.31

Mesh:

C-grids: 137x25x2, 273x49x2, 545x97x2

Domain boundaries 10 chord lengths
away from airfoil

Solver settings:

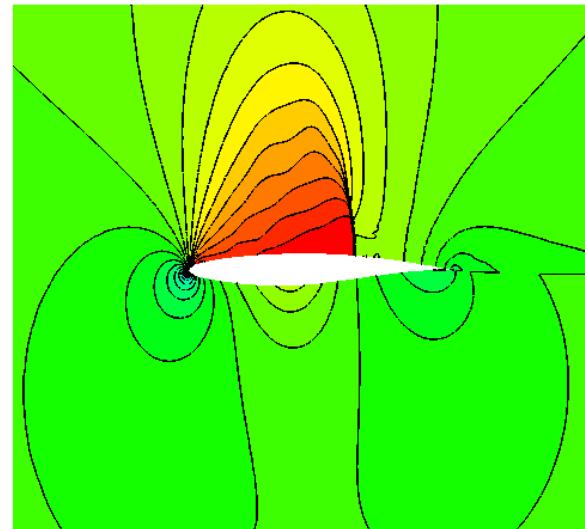
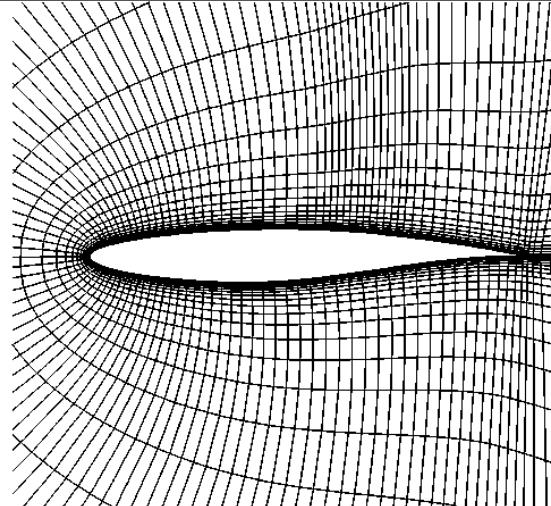
ilu(1)/RCM

Eta=10-1

MPF=5

LTS

Adaptive factor=1.2



Meshes courtesy of
M. Berggren

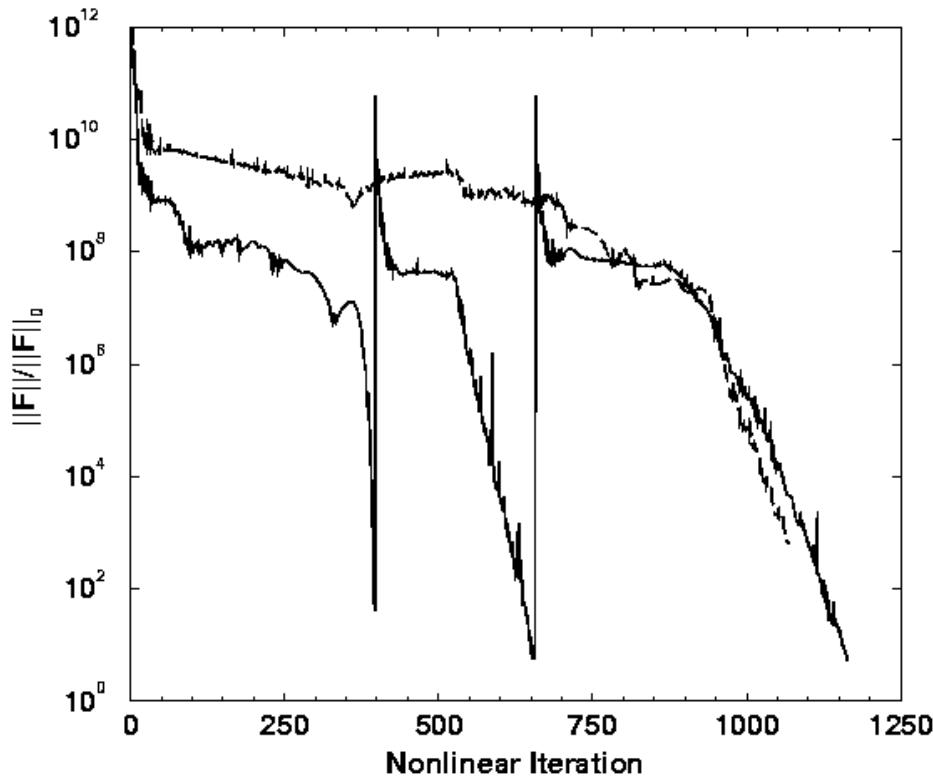
Mach No. Contours



Convergence History With Mesh Sequencing



	CFL	Mesh	DOF	NLIts	LIts	CPU
Coarse	1-1000	137x25x2	34,250	397	1715	1075
Medium	10-500	273x49x2	133,770	260	2671	4966
Fine	10-500	545x97x2	528,650	506	5089	31,746
						37,787
Fine	10-500	545x97x2	528,650	1068	5089	55,641





SARANS Convergence History



	MPF	Eta	Timestep	Adapt	CFL	Nu_hat	Cpu	NLits	Lits
1	1	10-3	LTS	1.05	1-100	2.7e-5	112,591	434	11279
2	5	10-1	LTS	1.025	10-500	2.7e-5	38,983	1085	9392
3	5	10-2	LTS	1.2	1-500	2.5e-5	51,028	809	16893
4	10	10-1	LTS	1.2	1-500	2.5e-5	32619	928	9467
5	5	10-1	GTS	1.2	1-107	2.5e-5	----	---	---

Flow conditions:

M=0.729

Re=6.5E6

Alpha=2.31

Mesh:

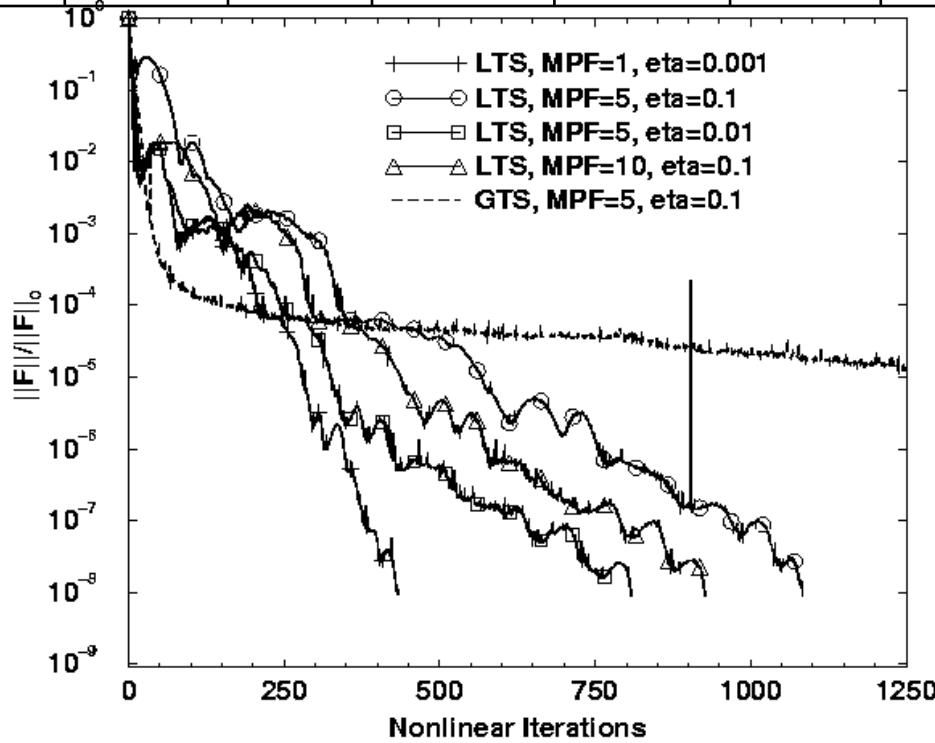
C-grid, 369x65x2

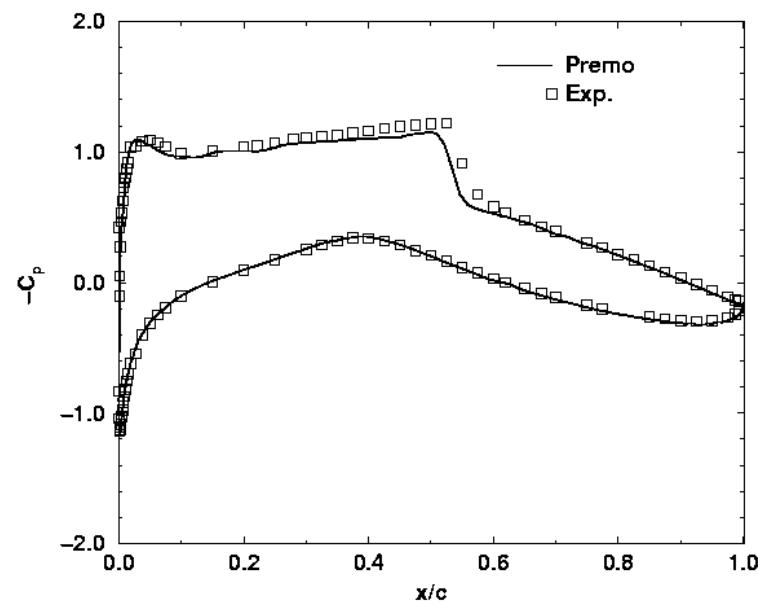
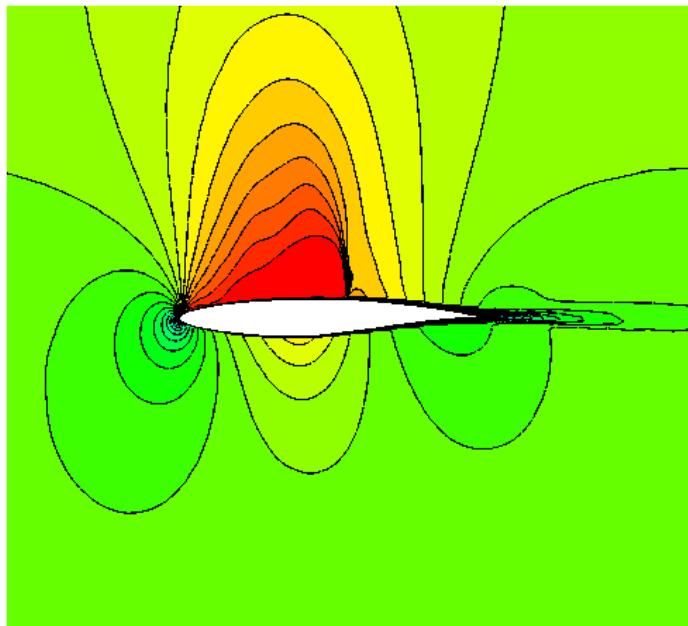
Boundaries ~20 chords

Away from airfoil

Solver settings:

Ilu(1)/RCM



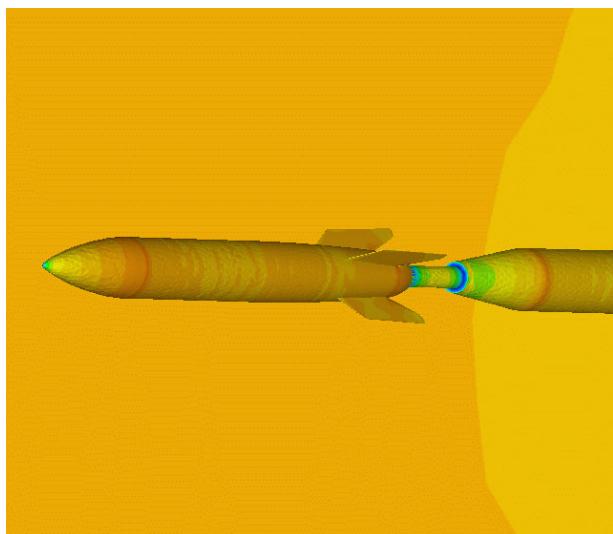
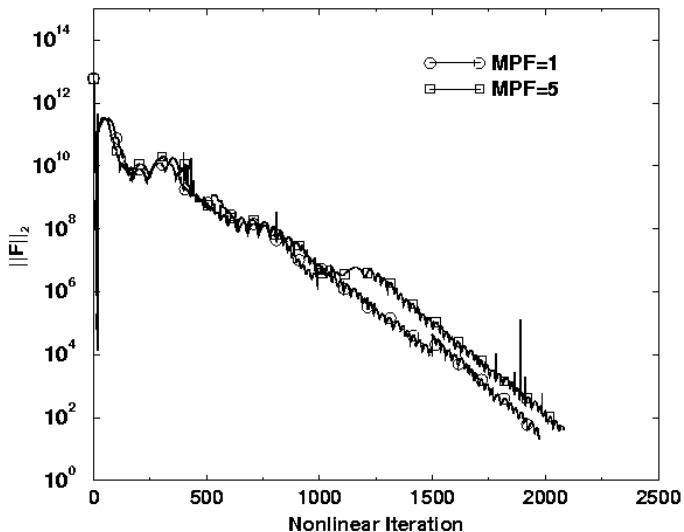


Cook, P.H., M.A. McDonald, M.C.P. Firmin, "Aerofoil RAE 2822 - Pressure Distributions, and Boundary Layer and Wake Measurements," *Experimental Data Base for ComputerProgram Assessment*, AGARD Report AR 138,1979.

<http://www.grc.nasa.gov/WWW/wind/valid/raetaf/raetaf01/raetaf01.html>

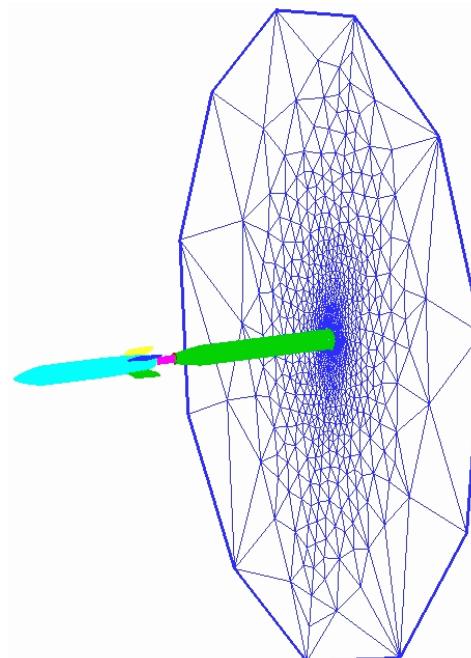


Unstructured Mesh Parallel Solver



MPF	CPU (hrs.)
1	40
5	24

Flow Conditions:
 $M=0.8$, $\alpha=0$
Mesh: 174,383 nodes
916,009 Tet elements
Solver settings:
 $CFL_{max}=500$
 $Ilu(1)/RCM$, overlap=1
 $Eta=10^{-1}$
Discretization: AD, LS, vanAlbada



Mesh courtesy of J.L. Payne



Summary



- For unstructured mesh CFD solvers, MFNK nonlinear solver can be constructed that builds on the explicit solver
 - Minimum requirement is a Residual function
 - Same connectivity list to form matrix graph
 - Offers flexibility in constructing preconditioning/Jacobian operators
- Obtained steady-state solutions to a variety of flow problems using a MFNK method employing different preconditioner operators
 - Inviscid nozzle flow
 - Subsonic, transonic and supersonic inviscid flows
 - Laminar flows
 - RANS
 - Hypersonic Laminar blunt body flow
- Demonstrated Improved performance
 - Implicit solver greatly reduces time-to-solution for steady-state problems compared to explicit
 - AD was found to be superior to approximate and finite-difference methods
 - Modified FDC is competitive and much easier to implement



The End

