Exascale Opportunities for Computational Aerodynamics

Dimitri Mavriplis
University of Wyoming

Petaflops Opportunities for the NASA Fundamental Aeronautics Program

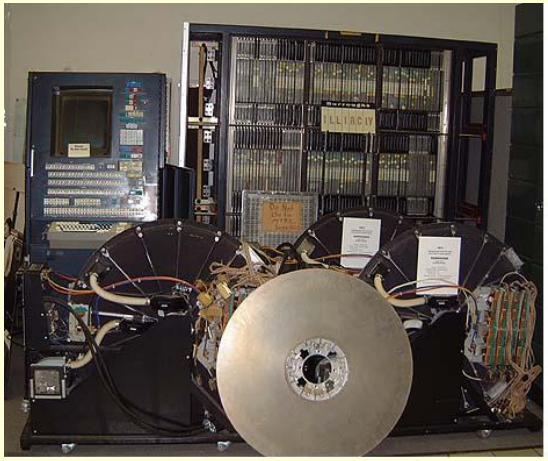
Dimitri Mavriplis (University of Wyoming)
David Darmofal (MIT)
David Keyes (Columbia University)
Mark Turner (University of Cincinnati)

AIAA 2007-4048

Overview **(AIAA-2007-4048)**

- Two principal intertwined themes
  - 1: NASA simulation capability risks becoming commoditized
    - Rapid advance of parallelism (> 1M cores)
    - Fundamental improvements in algorithms and development tools not keeping pace
    - Hardware and software complexity outstripping our ability to simulate (J. Alonso)
    - Clear vision of enabling possibilities is required
      - What would you do with 1000 times more computational power ?
  - 2: HPC Resurgent at National Level : Competitiveness
    - Aerospace industry is at the heart of national competitiveness
    - NASA is at the heart of aerospace industry
    - **Aeronautics seldom mentioned in national HPC reports**

ARMD's Historic HPC Leadership
(Code R)



- ILLIAC IV (1976)
- National Aerodynamic Simulator (1980's)
- 1992 HPCCP Budget:
  - $596M (Total)
    - $93M Department of Energy (DOE)
    - $71M NASA
      - Earth and Space Sciences (ESS)
      - Computational Aerosciences (CAS)

- Computational Aerosciences (CAS) Objectives (1992):
  - "…*integrated, multi-disciplinary simulations and design optimization of aerospace vehicles throughout their mission profiles*"
  - "… *develop algorithm and architectural testbeds … scalable to sustained teraflops performance*"

Algorithm Development Opportunities

- Modest investment in cross-cutting algorithmic work would complement mission driven work and ensure continual long-term progress (including NASA expertise for determining successful future technologies)
  - Scalable non-linear solvers
  - Higher-order and adaptive methods for unstructured meshes
  - Optimization (especially for unsteady problems)
  - Reduced-order modeling
  - Uncertainty quantification
  - Geometry management
- Current simulation capabilities (NASA/DOE/others) rests on algorithmic developments, many funded by NASA
- Revolutionary Computational Aerosciences Program

From Petascale to Exascale

- Petascale is here
  - National HPC centers > 1Pflop

- Exascale is coming
  - Up to 1B threads
  - Deep memory hiearchies
  - Heterogeneous architectures
  - Power considerations dominant
  - Petascale at the mid-range

IESP
INTERNATIONAL EXASCALE SOFTWARE PROJECT

From Petascale to Exascale

- Petascale is here
  - National HPC centers > 1Pflop

- Exascale is coming
  - Up to 1B threads
  - Deep memory hiearchies
  - Heterogeneous architectures
  - Power considerations dominant
  - Petascale at the mid-range
    - Terascale on your phone ?

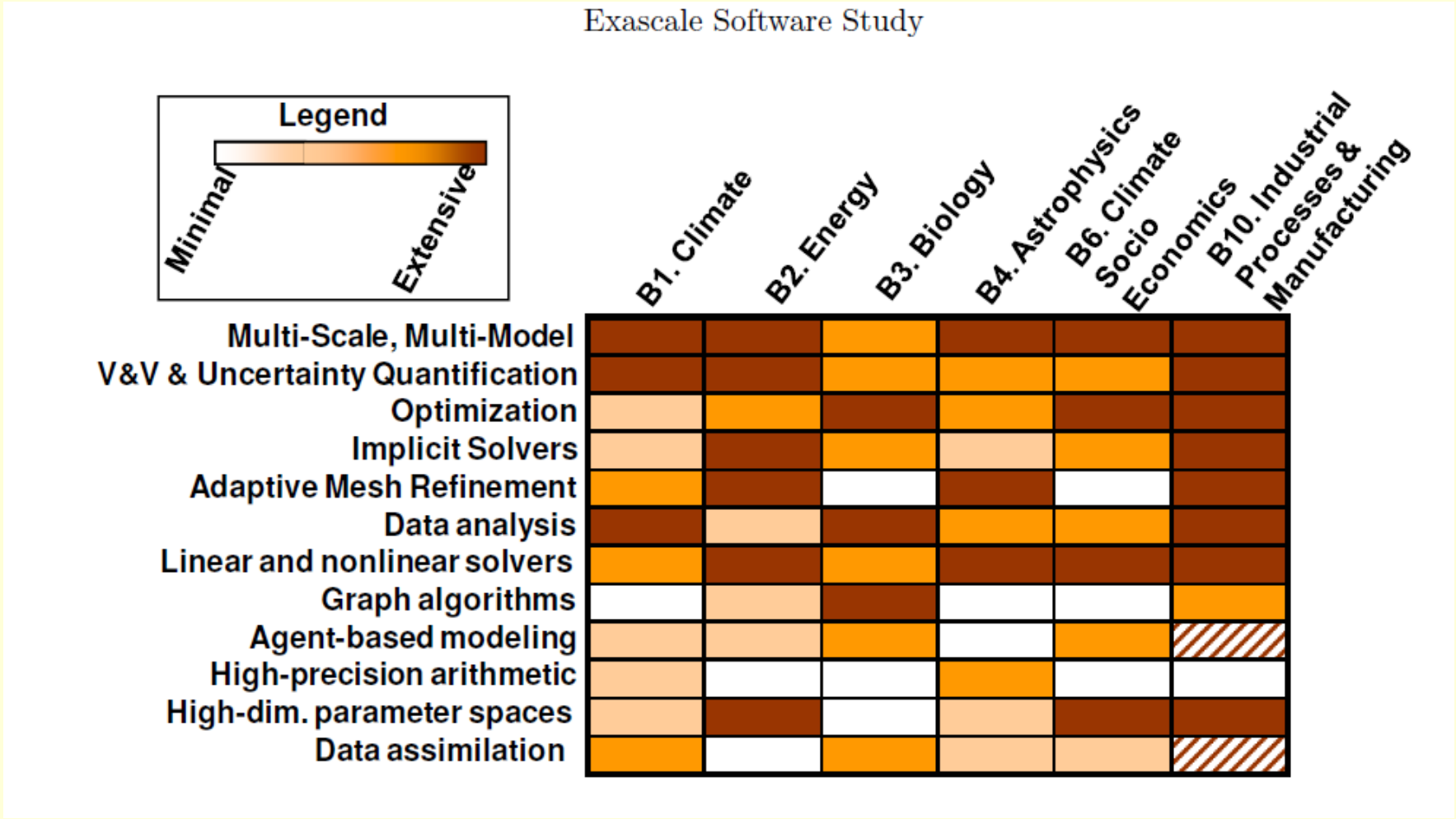
IESP
INTERNATIONAL EXASCALE SOFTWARE PROJECT

Getting to Exascale

- Strong scaling of current simulations
  - Running same problem faster
  - **Highly unlikely**
- Weak scaling of current simulations
  - Increasing problem size with hardware capability
    - eg Climate simulation: Insatiable resolution requirements
  - Algorithmic consequences
    - Implicit time stepping will be required to maintain suitable real time climate simulation rates
      - 5 years of simulation per wall clock day

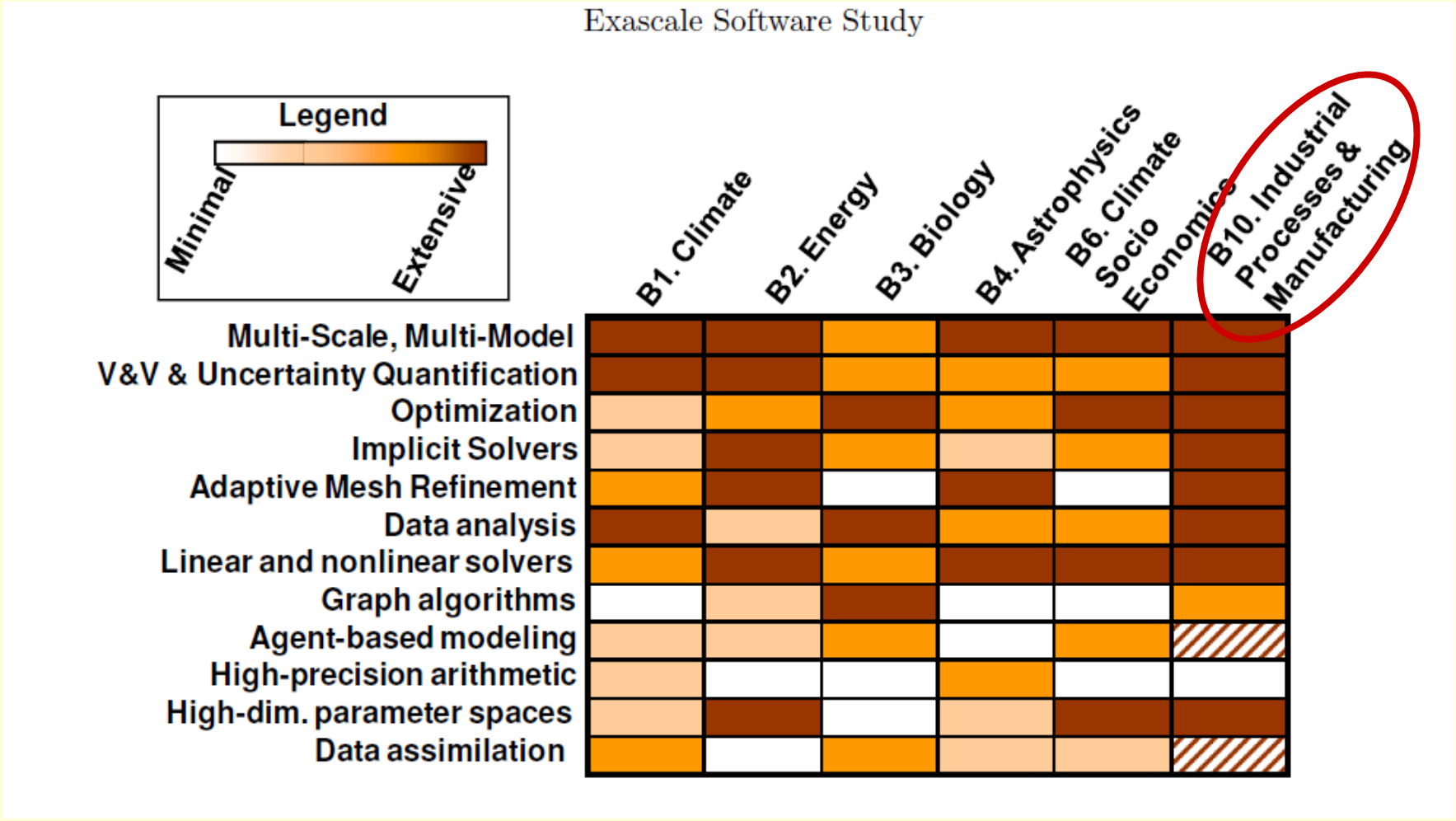
Aeronautics/Aerospace HPC

- Aerospace is **engineering** based discipline
- HPC advocacy has increasingly been taken up by the **science** community
  - Numerical simulation is now the third pillar of scientific discovery on an equal footing alongside theory and experiment
  - Increased investment in HPC will enable new scientific discoveries
- Engineering is not discovery based
  - Arguable more difficult to reach exascale
    - e.g Gradient-based optimization is inherently sequential



- From: DARPA/IPTO/AFRL Exascale Computing Study (2008)

http://users.ece.gatech.edu/~mrichard/ExascaleComputingStudyReports/ECS_reports.htm

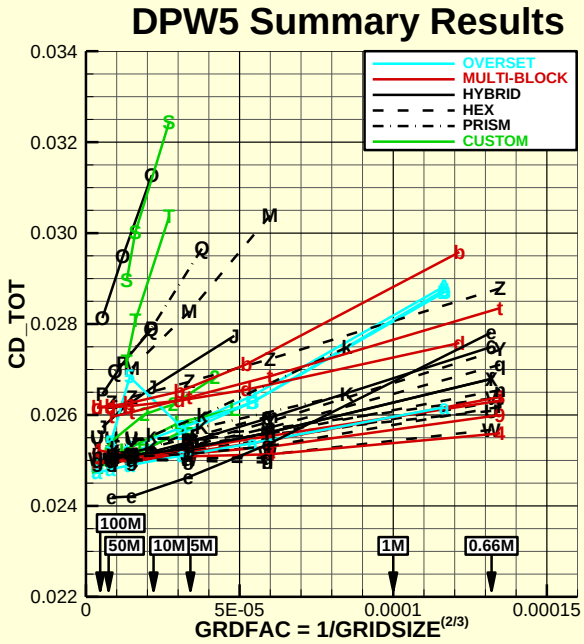


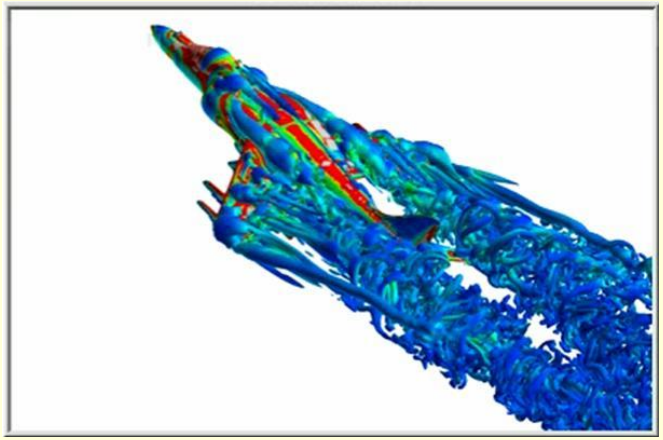
- From: DARPA/IPTO/AFRL Exascale Computing Study (2008)

http://users.ece.gatech.edu/~mrichard/ExascaleComputingStudyReports/ECS_reports.htm

Reaching Aeronautics Exascale

- Weak Scaling
  - Still only beginning to understand resolution requirements
  - Need dramatically more spatial resolution to increase fidelity
  - Most high-fidelity simulations have many time scales
  - Learning more about true resolution requirements as formal error estimation becomes part of CFD process
  - Towards LES/DNS of full aircraft or propulsion systems
    - Estimates by Spalart et al. (1997)

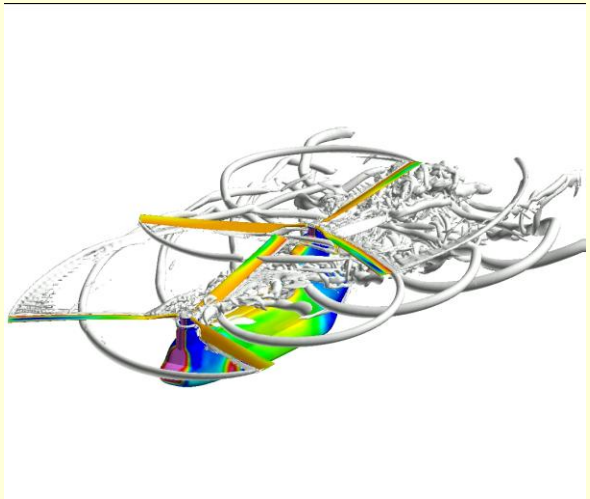




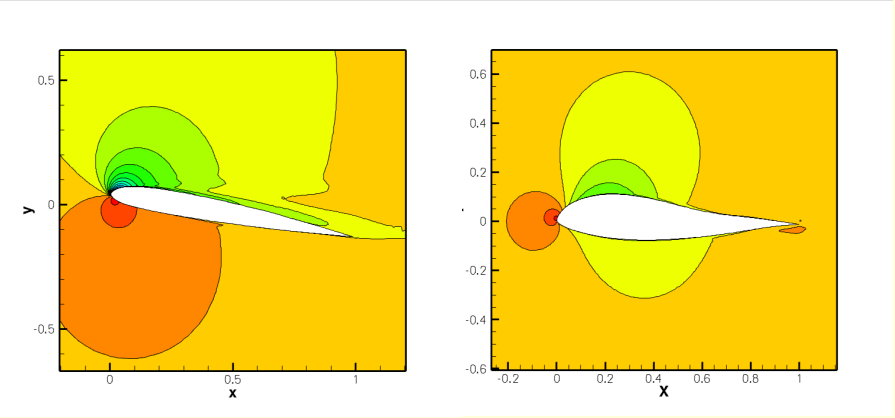
Aeronautics Exascale

- Many problems do not require ever-increasing spatial resolution
- 10M or 100M grid points “good enough” for engineering decisions
- Long time integration of stiff implicit systems makes for expensive simulations
- Gradient-based optimization is sequential in nature and becomes expensive (especially time-dependent optimization)

Overflow/RCAS CH-47 simulation
(Dimanlig/Bhagwat – AFDD, Boeing, ART)



Airfoil optimization for dynamic stall
(Mani and Mavriplis 2012)



Base Optimized

Aeronautics Exascale

- Problems with limited opportunities for spatial parallelism will need to seek other avenues for concurrency
  - Parameter space
    - Embarrassingly parallel
  - Time parallelism
    - Time spectral
    - Space-time methods
  - Alternate optimization approaches
    - Hessian construction for Newton Optimization

Time-Spectral Formulation

$$\frac{\partial}{\partial t}(V\mathbf{U}) + \mathbf{R}(\mathbf{U}, \vec{\mathbf{x}}(t), \vec{\mathbf{n}}(t)) + \mathbf{S}(\mathbf{U}, \vec{\mathbf{n}}(t)) = 0$$

- Discrete Fourier and Fourier inverse transform

$$\hat{\mathbf{U}}_k = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{U}^n e^{-ik\frac{2\pi}{T}n\Delta t} \qquad \mathbf{U}^n = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{\mathbf{U}}_k e^{ik\frac{2\pi}{T}n\Delta t}$$

- Time Derivative

$$\frac{\partial}{\partial t}(\mathbf{U}^n) = \frac{2\pi}{T}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} ik\hat{\mathbf{U}}_k e^{ik\frac{2\pi}{T}n\Delta t} = \sum_{j=0}^{N-1} d_n^j \mathbf{U}^j$$

$$d_n^j = \begin{cases} \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}\cot\left(\frac{\pi(n-j)}{N}\right) & n \neq j \\ 0 & n = j \end{cases}$$

for an even number of N

Time Spectral Formulation

$$d_n^j = \begin{cases} \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}\,cosec\left(\frac{\pi(n-j)}{N}\right) & n \neq j \\ 0 & n = j \end{cases}$$

for an odd number of N

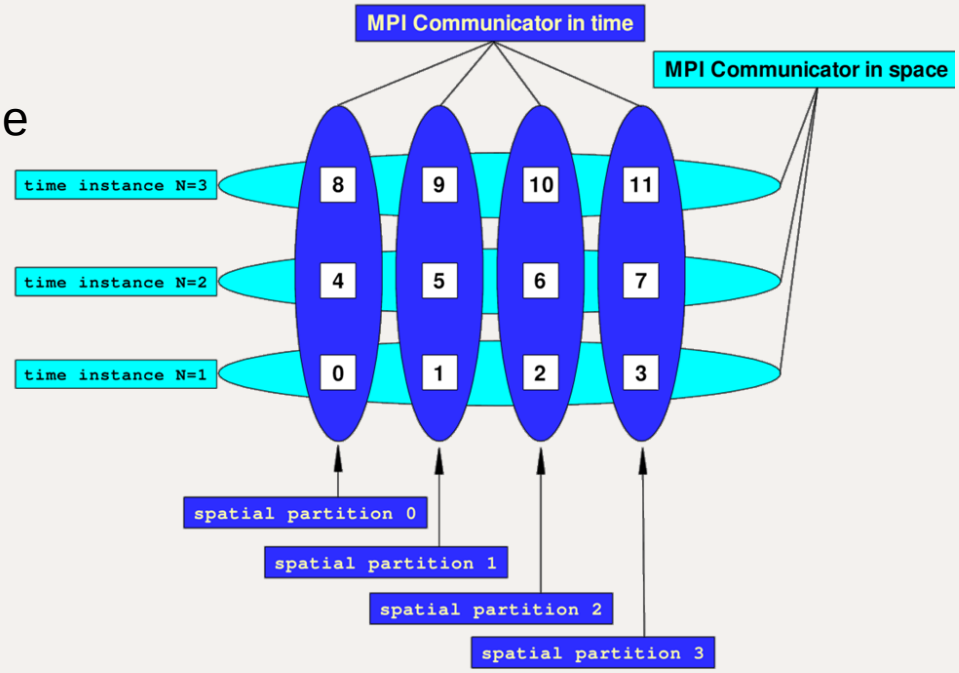
- Discrete equations

$$\sum_{j=0}^{N-1} d_n^j V^j \mathbf{U}^j + \mathbf{R}(\mathbf{U}^n, \bar{\mathbf{x}}^n, \vec{\mathbf{n}}^n) + \mathbf{S}(\mathbf{U}^n, \vec{\mathbf{n}}^n) = 0 \qquad n = 0,1,2,\ldots,\ N-1$$

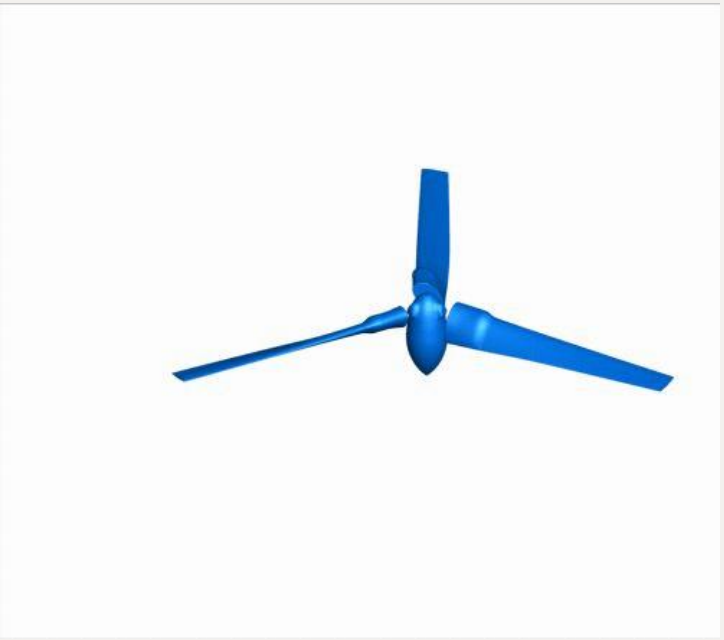
- Time-spectral method may be implemented without any modifications to an existing spatial discretization, requiring only the addition of the temporal discretization coupling term
- All N time instances coupled and solved simultaneously
- Extensions possible for quasi-periodic problems

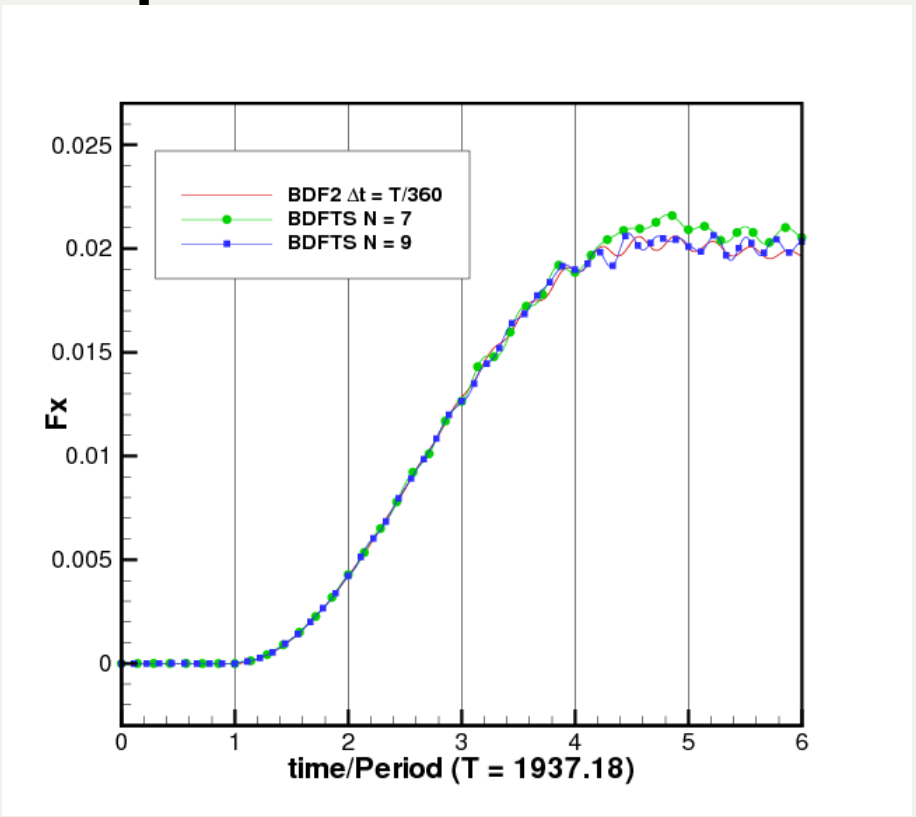
Formulation

- Parallel Implementation
  - Parallelism in time and space. Two types of inter-processor communication: communication between spatial partitions and communication between all of the time instances
  - For multicore and/or multiprocessor hardware nodes within a distributed memory parallel machine, the optimal strategy consists of placing all time instances of a particular spatial partition on the same node



Parallel Time Spectral Simulation



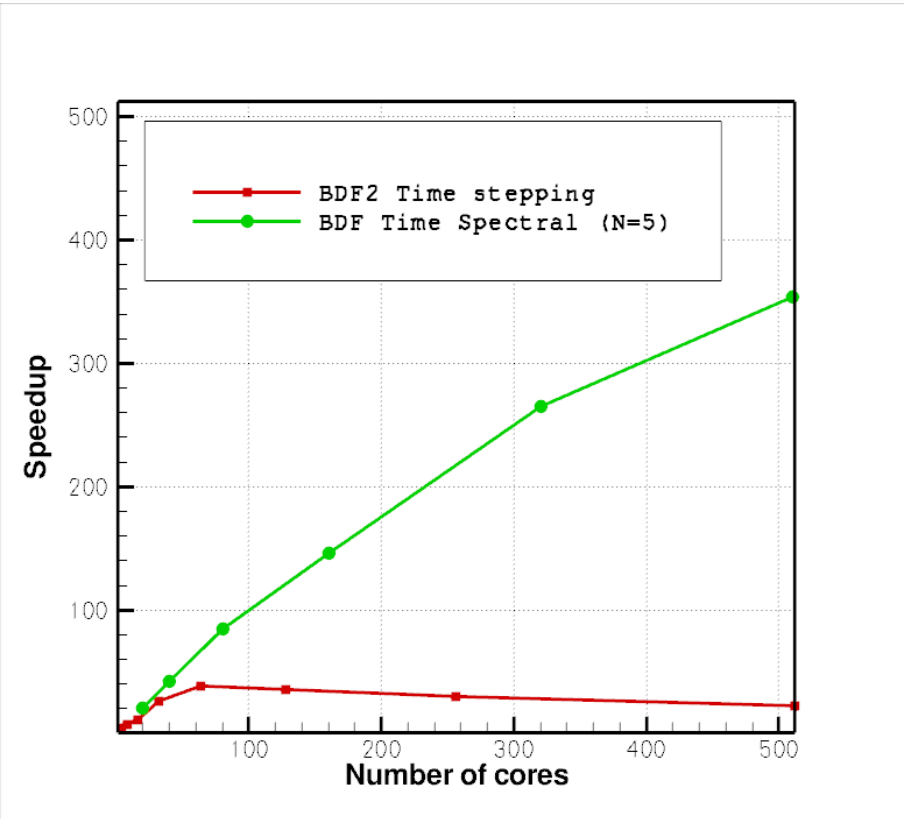


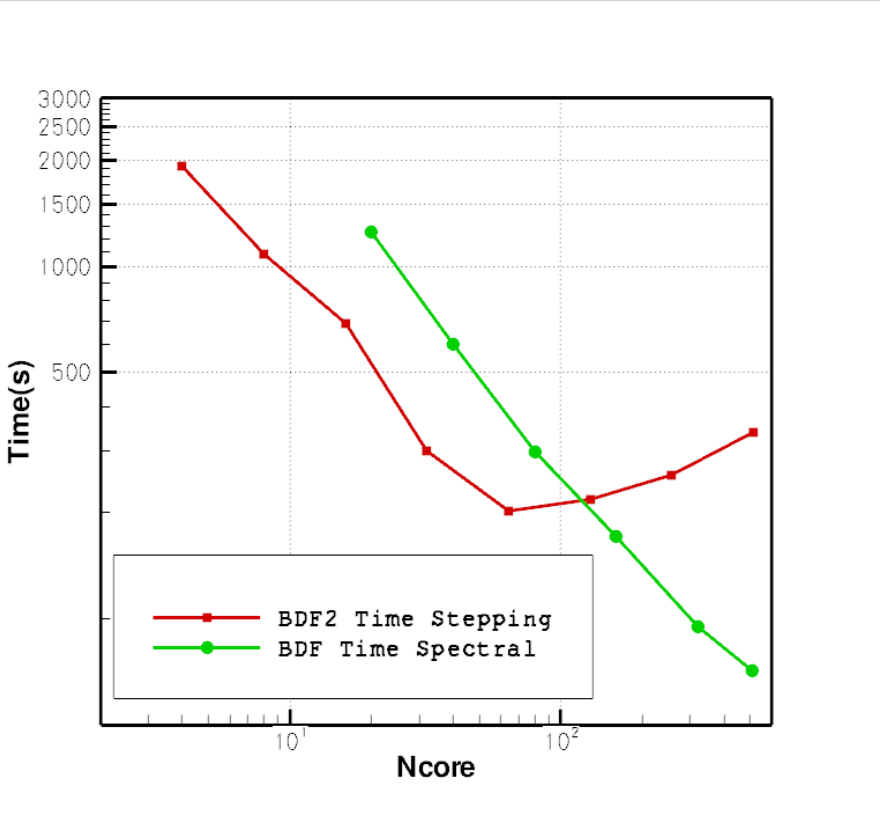
- BDF2: 50 multigrid cycles per time step, 360 time steps per revolution, 6 revolutions
  8 processes, 8 spatial partitions: 24.1137 X 50 X 360 X 6 = 2,604,028 s

- BDFTS: N = 7, 300 multigrid cycles per revolution, 6 revolutions
  56 processes, 8 spatial partitions: 31.167 X 300 X 6 = 56,101.3 s

- BDFTS: N = 9, 300 multigrid cycles per revolution, 6 revolutions
  72 processes, 8 spatial partitions: 32.935 X 300 X 6 = 59,282.5 s

Time Spectral Scalability





- Coarse 500,000 pt mesh with limited spatial parallelism
- N=5 time spectral simulation employs 5 times more cores

Second-Order Sensitivity Methods

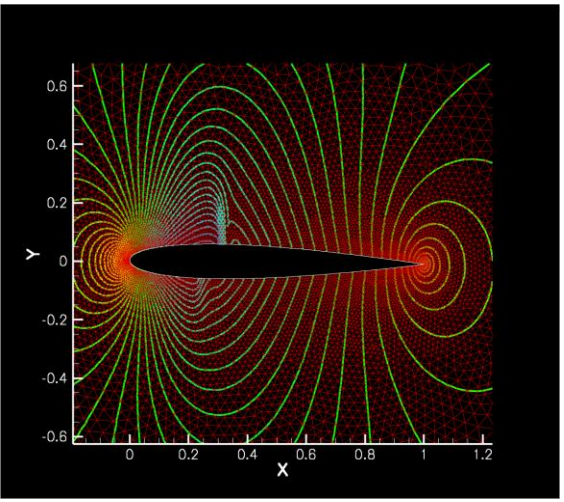
- Adjoint is efficient approach for calculating first-order sensitivities (first derivatives)
- Second-order (Hessian) information can be useful for enhanced capabilities:
  - Optimization
    - Hessian corresponds to Jacobian of optimization problem (Newton optimization)
      - Unsteady optimization seems to be hard to converge
    - Optimization for stability derivatives
    - Optimization under uncertainty
  - Uncertainty quantification
    - Method of moments (Mean of inputs $\neq$ input of means)
    - Inexpensive Monte-Carlo (using quadratic extrapolation)

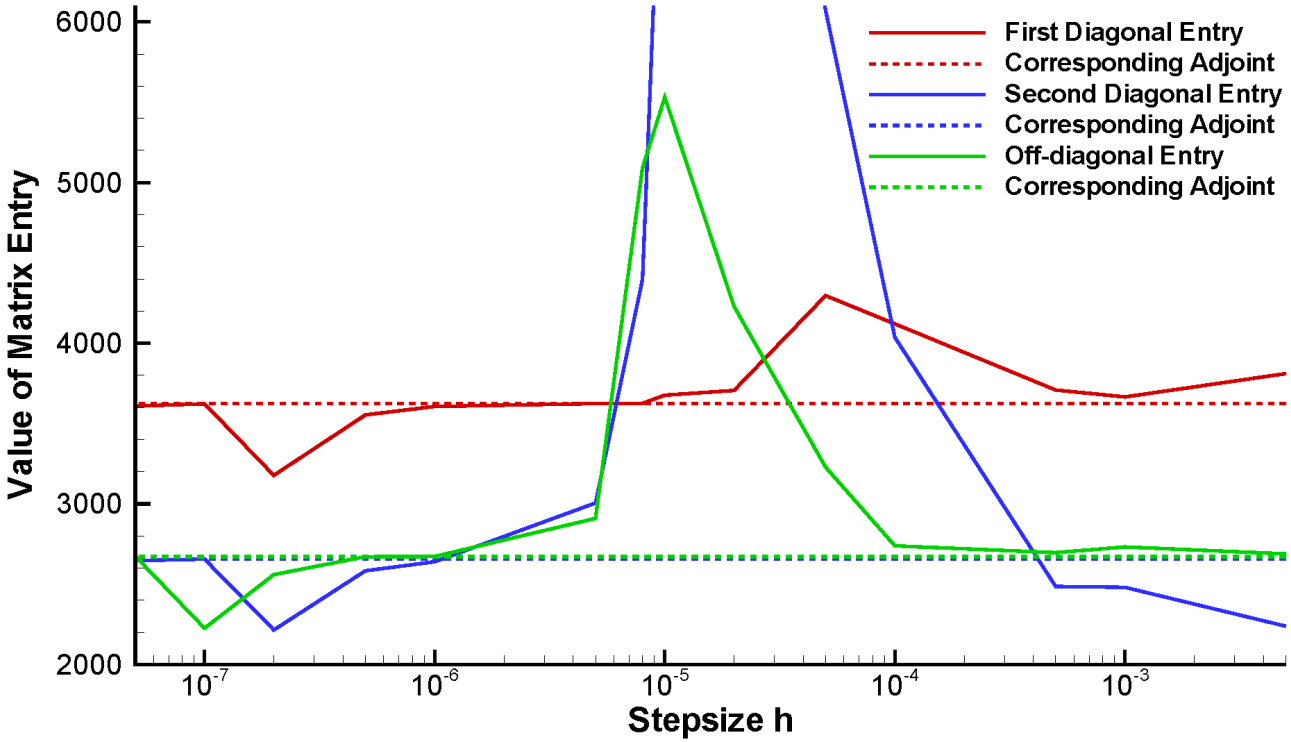
Forward-Reverse Hessian Construction

$$\frac{\partial^2 L}{\partial D_i \partial D_j}$$

- Hessian for N inputs is a NxN matrix
- Complete Hessian matrix can be computed with:
  - One tangent/forward problem for each input
  - One adjoint problem
  - Inner products involving local second derivatives computed with automatic differentiation
- Overall cost is N+1 solves for NxN Hessian matrix
  - Lower than double finite-difference: $O(N^2)$
  - All N+1 sensitivity runs may be performed in parallel

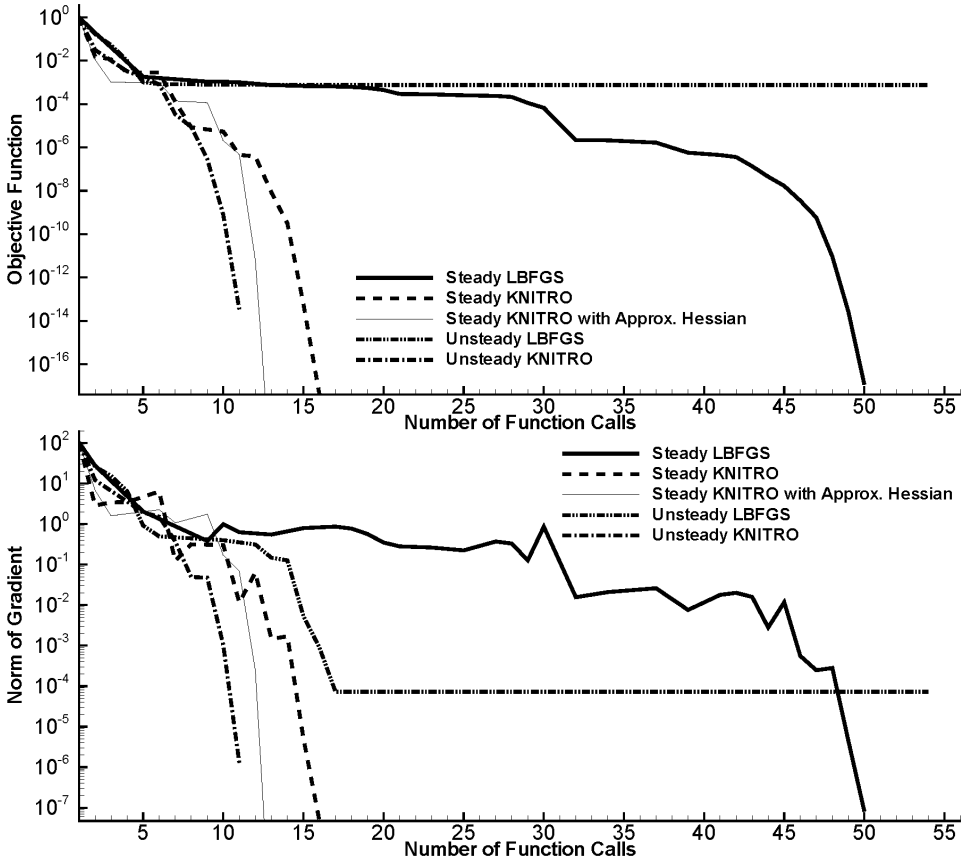
Hessian Implementation





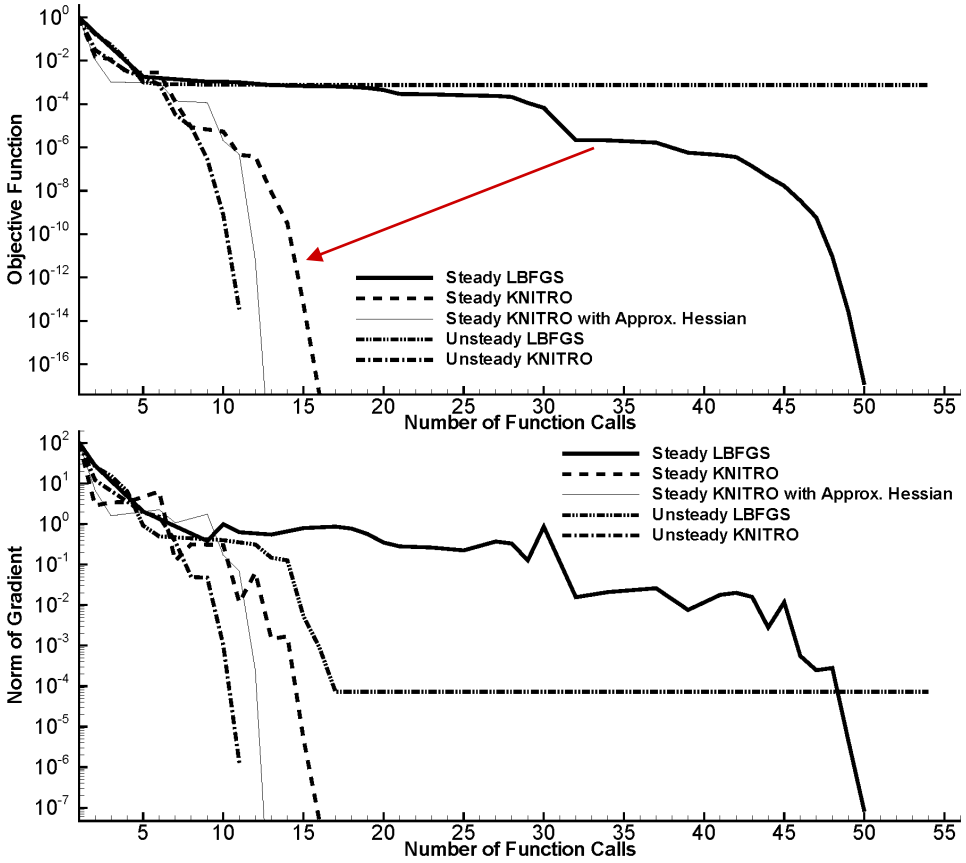
- Implemented for steady and unsteady 2D airfoil problems
- Validated against double finite difference for Hicks-Henne bump function design variables

Newton Optimization with Hessian



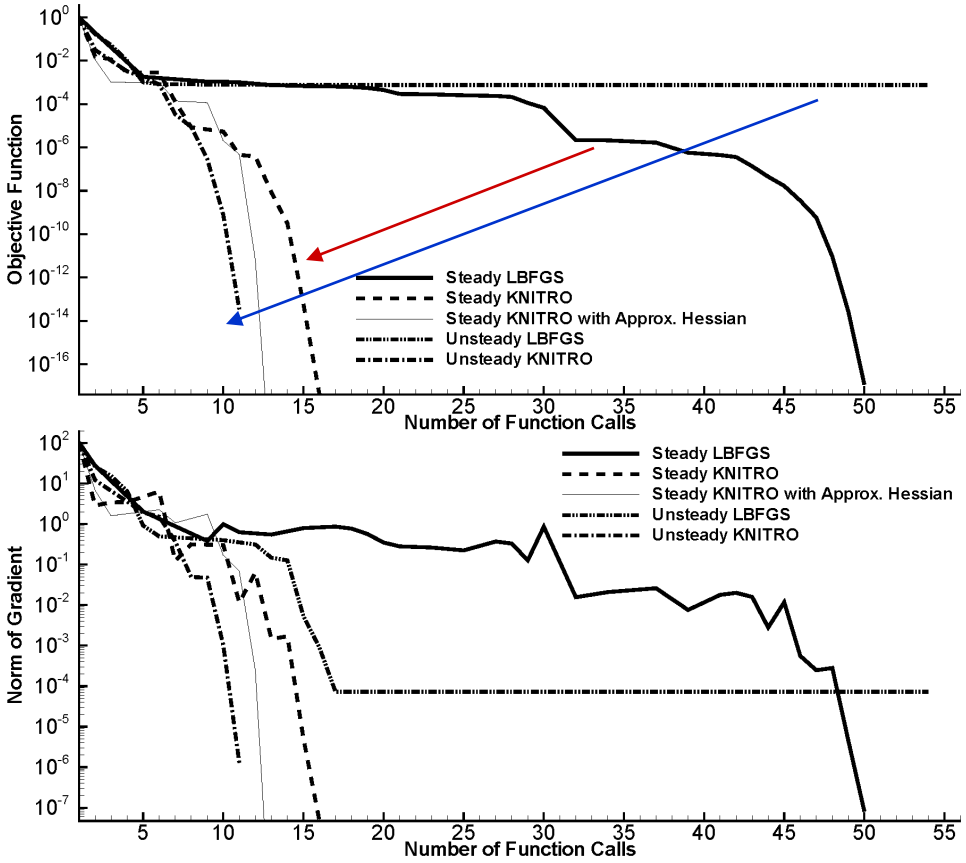
- LBFGS is “best” gradient-based optimizer
  - Constructs approximate Hessian based on previous design iterations
- KNITRO is Newton optimizer
  - Requires Hessian as input
- Superior performance in terms of number of function calls
  - Added cost of Hessian recovered (2 to 6 design variables)

Newton Optimization with Hessian



- LBFGS is “best” gradient-based optimizer
  - Constructs approximate Hessian based on previous design iterations
- KNITRO is Newton optimizer
  - Requires Hessian as input
- Superior performance in terms of number of function calls
  - Added cost of Hessian recovered (2 to 6 design variables)

Newton Optimization with Hessian



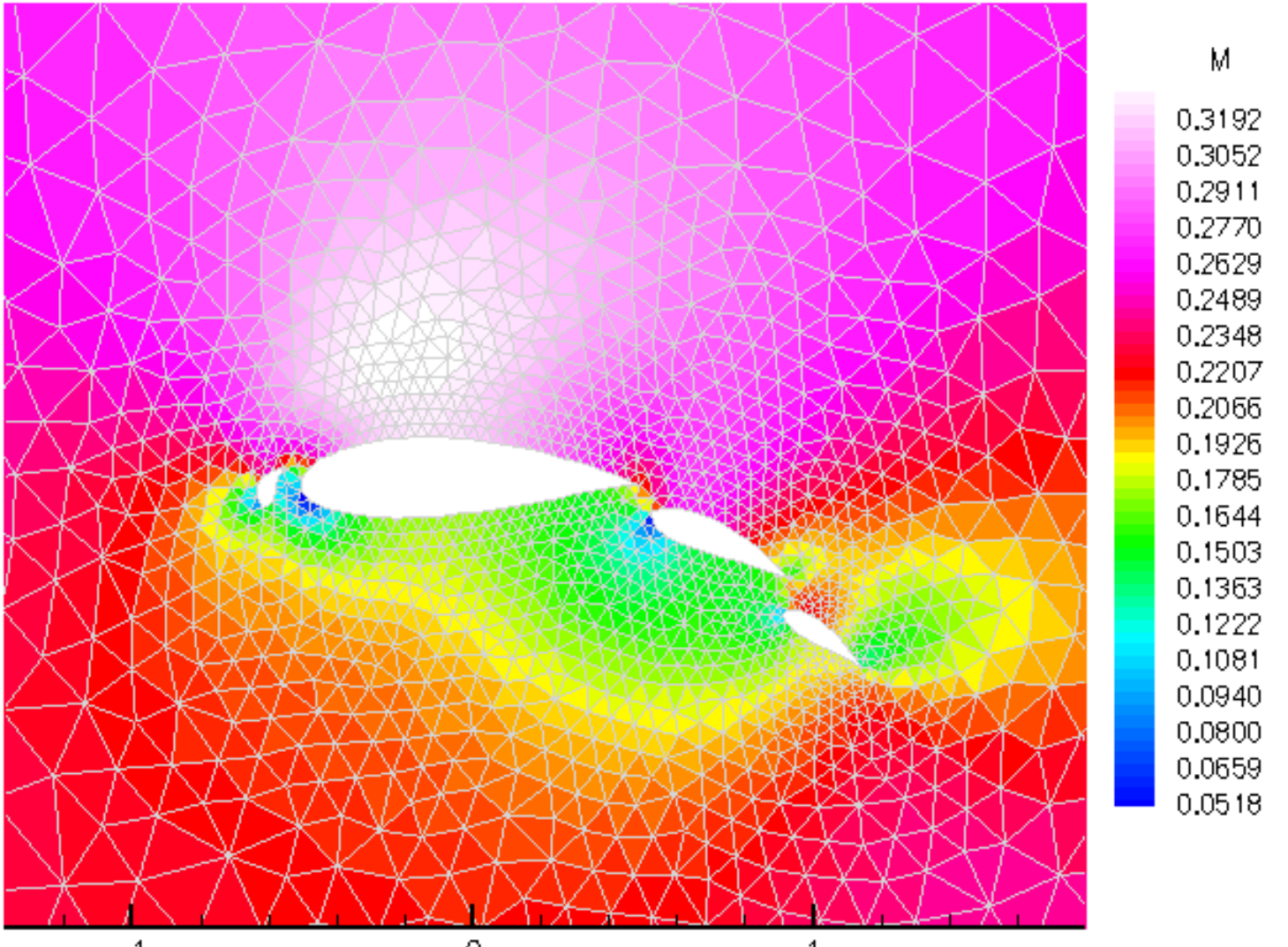
- LBFGS is “best” gradient-based optimizer
  - Constructs approximate Hessian based on previous design iterations
- KNITRO is Newton optimizer
  - Requires Hessian as input
- Superior performance in terms of number of function calls
  - Added cost of Hessian recovered (2 to 6 design variables)

High Order Methods

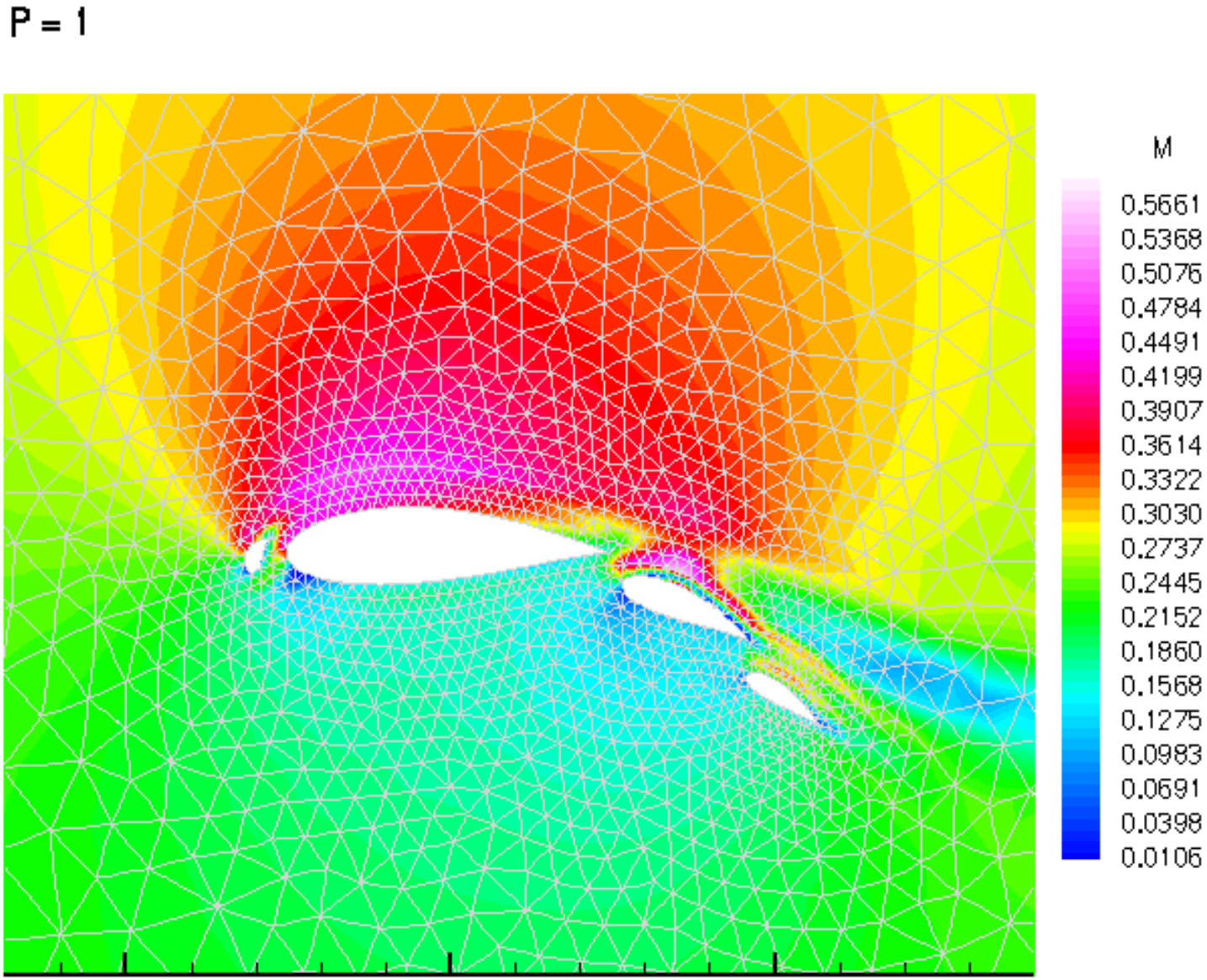
- Higher order methods such as Discontinuous Galerkin best suited to meet high accuracy requirements
  - Asymptotic properties
- HOMs reduce grid generation requirements
- HOMs reduce grid handling infrastructure
  - Dynamic load balancing
- Compact data representation (data compression)
  - Smaller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

- HOMs scale very well on massively parallel architectures using h-p multigrid solver

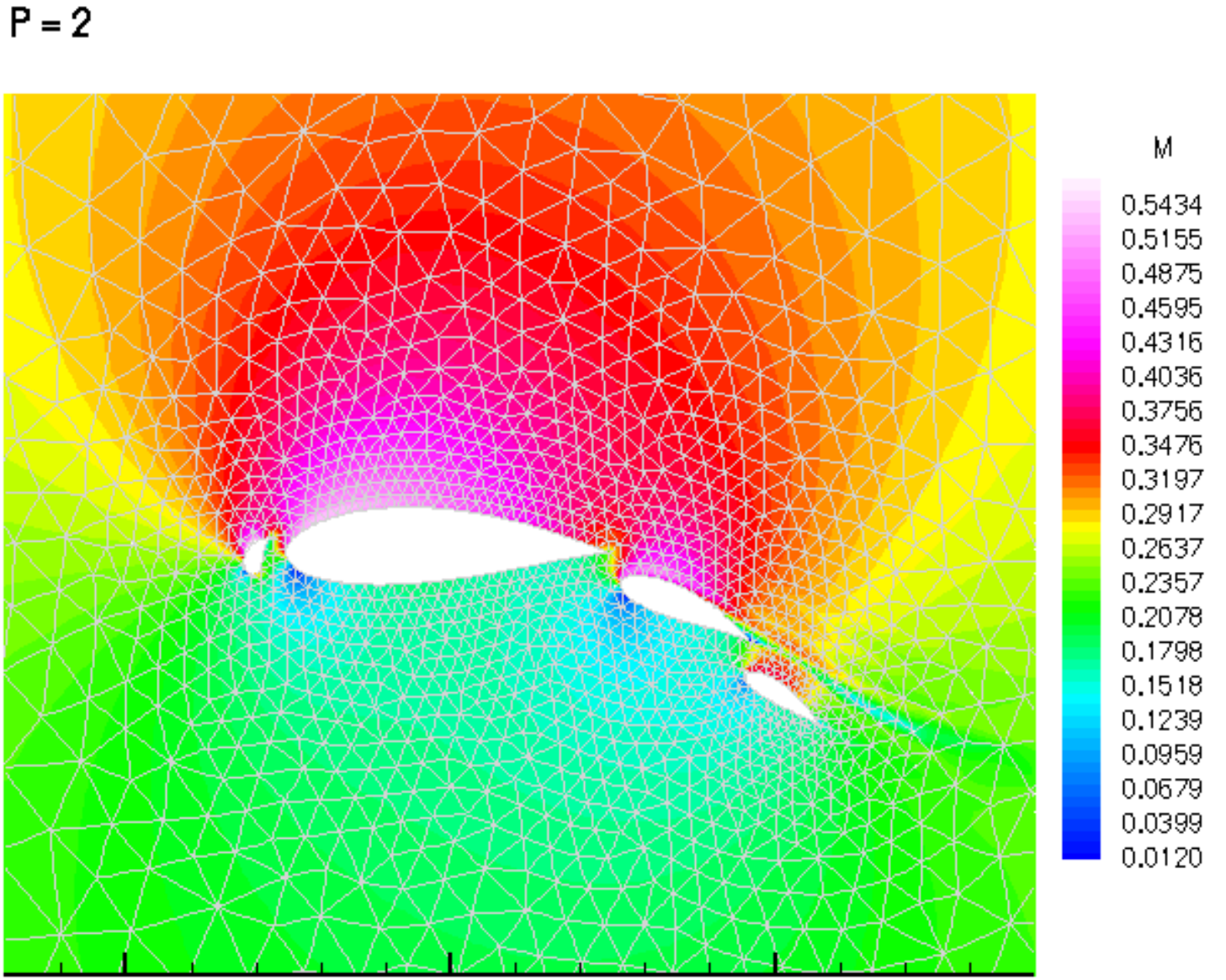
4-Element Airfoil (Euler Solution)



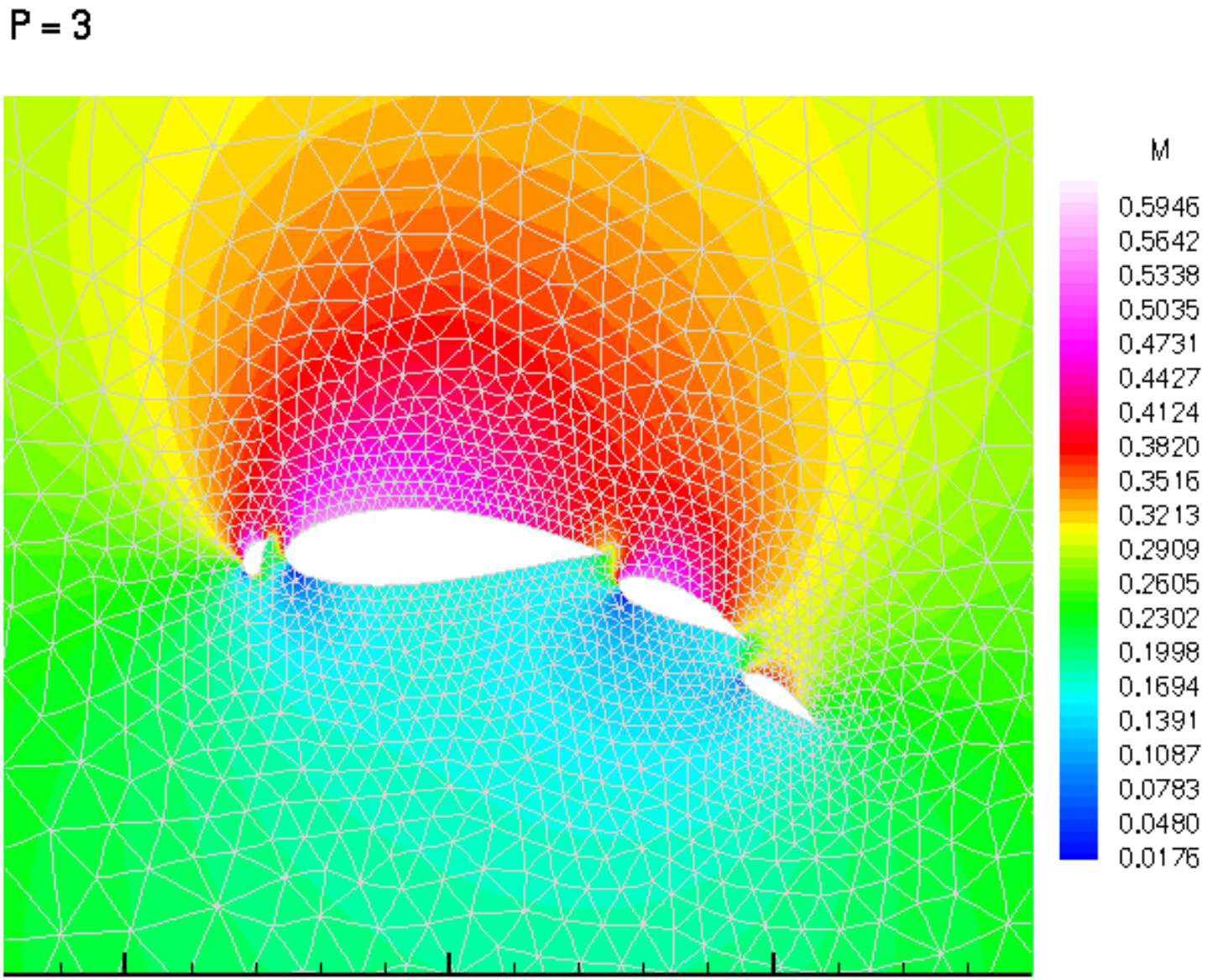
4-Element Airfoil (Euler Solution)



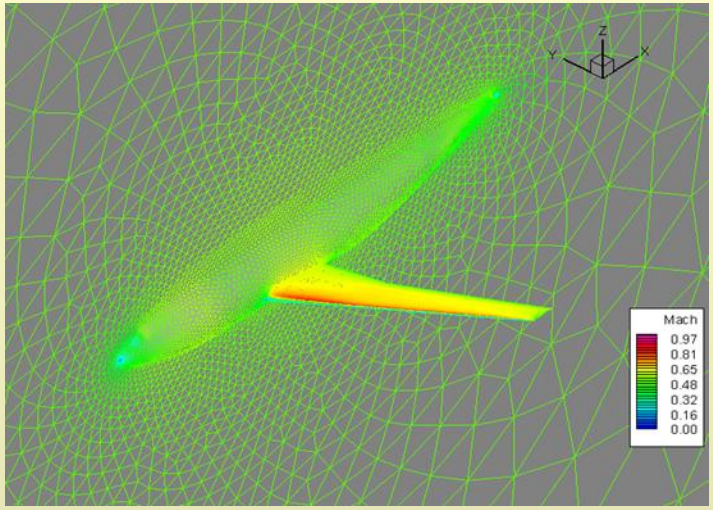
4-Element Airfoil (Euler Solution)



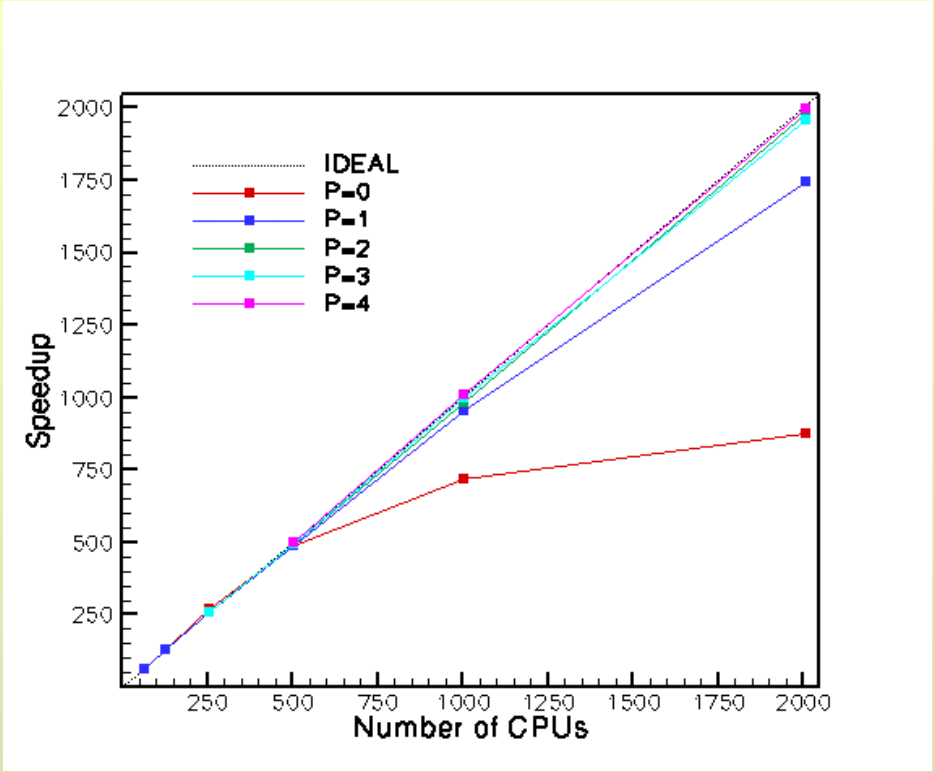
4-Element Airfoil (Euler Solution)



Parallel Performance of High-Order DG Methods

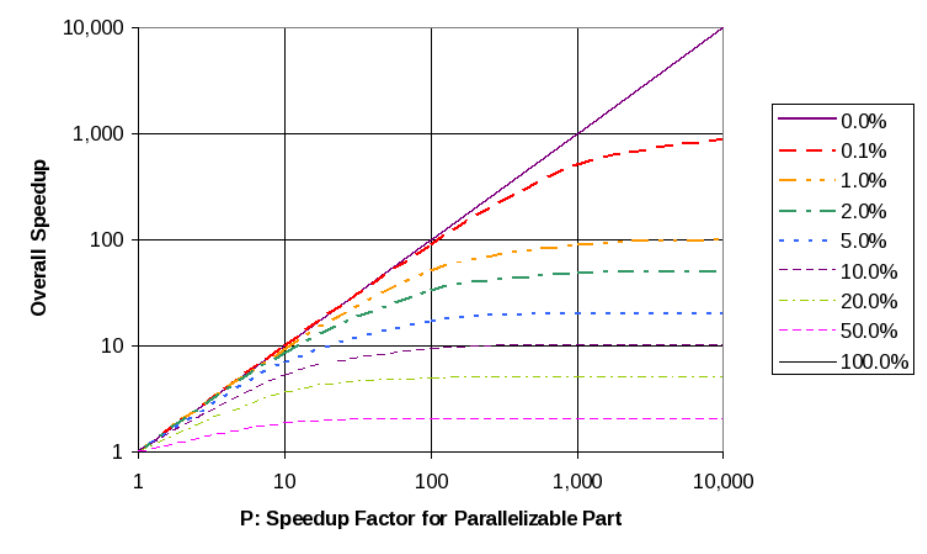


2.5M point mesh



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

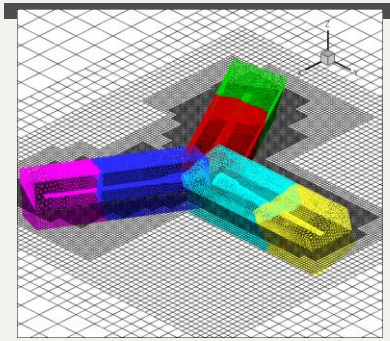
Sequential Bottlenecks

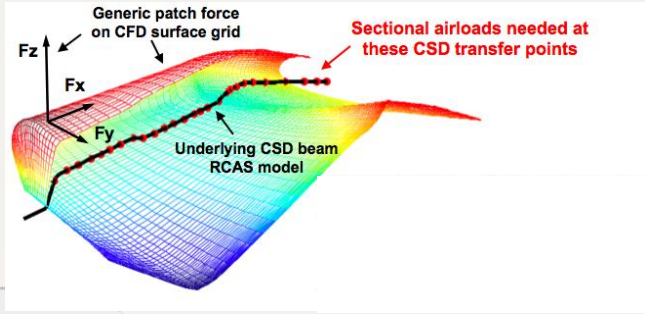


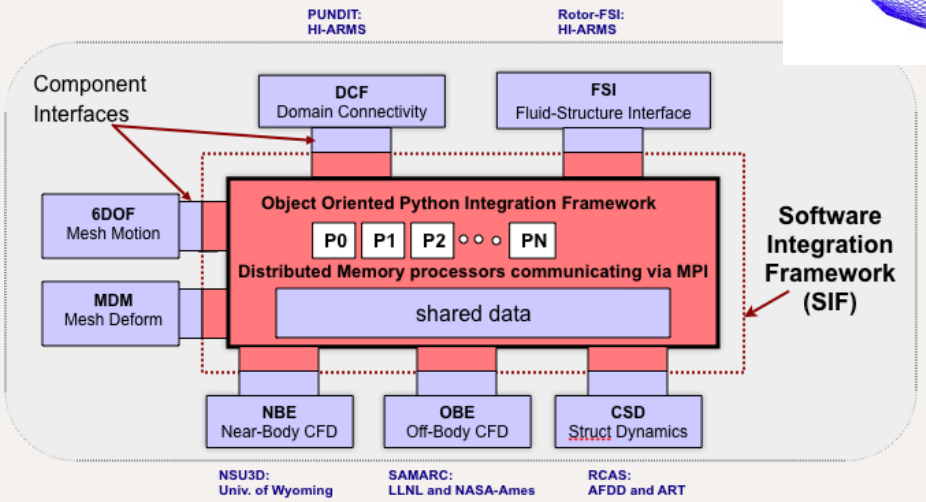
- Severe consequences of Amdahl's law at exascale
- HOMs reduce grid related bottlenecks
- Multidisciplinary software is complex and must be designed to avoid any sequential portions

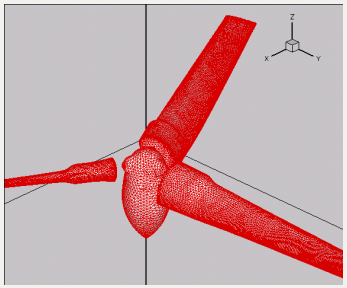
"MDO codes will never scale past 128 cpus" (1992)

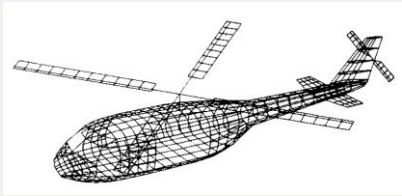
HELIOS Multidisciplinary Rotorcraft Simulation Software

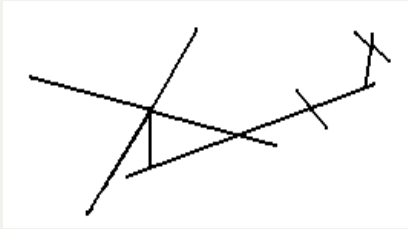


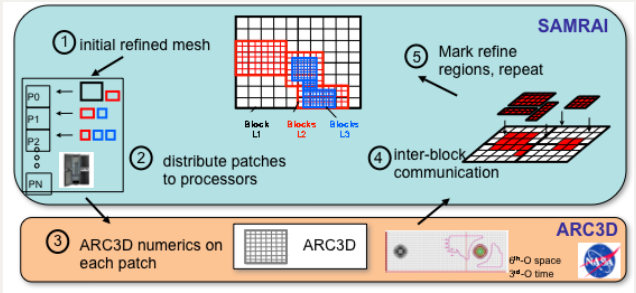




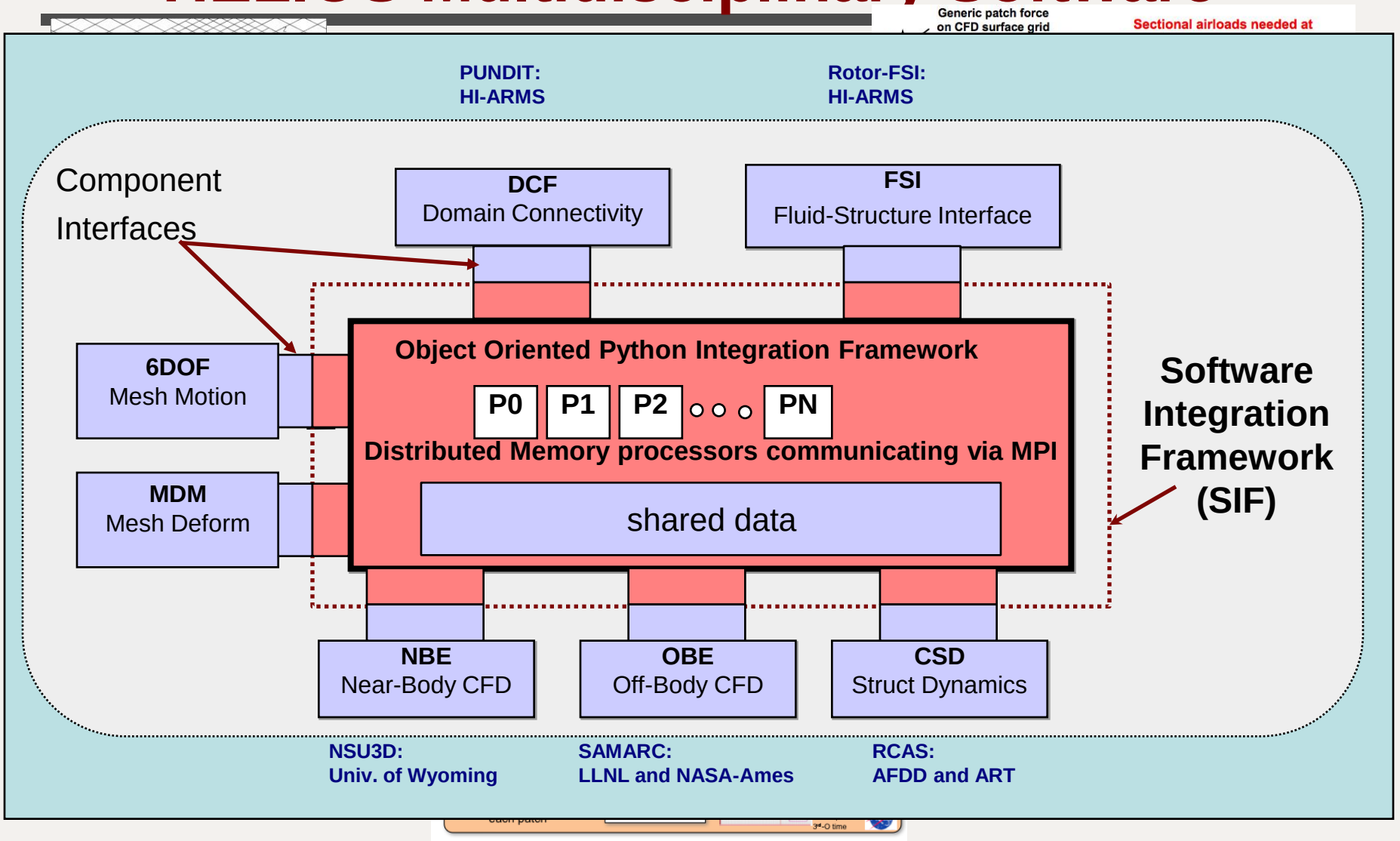






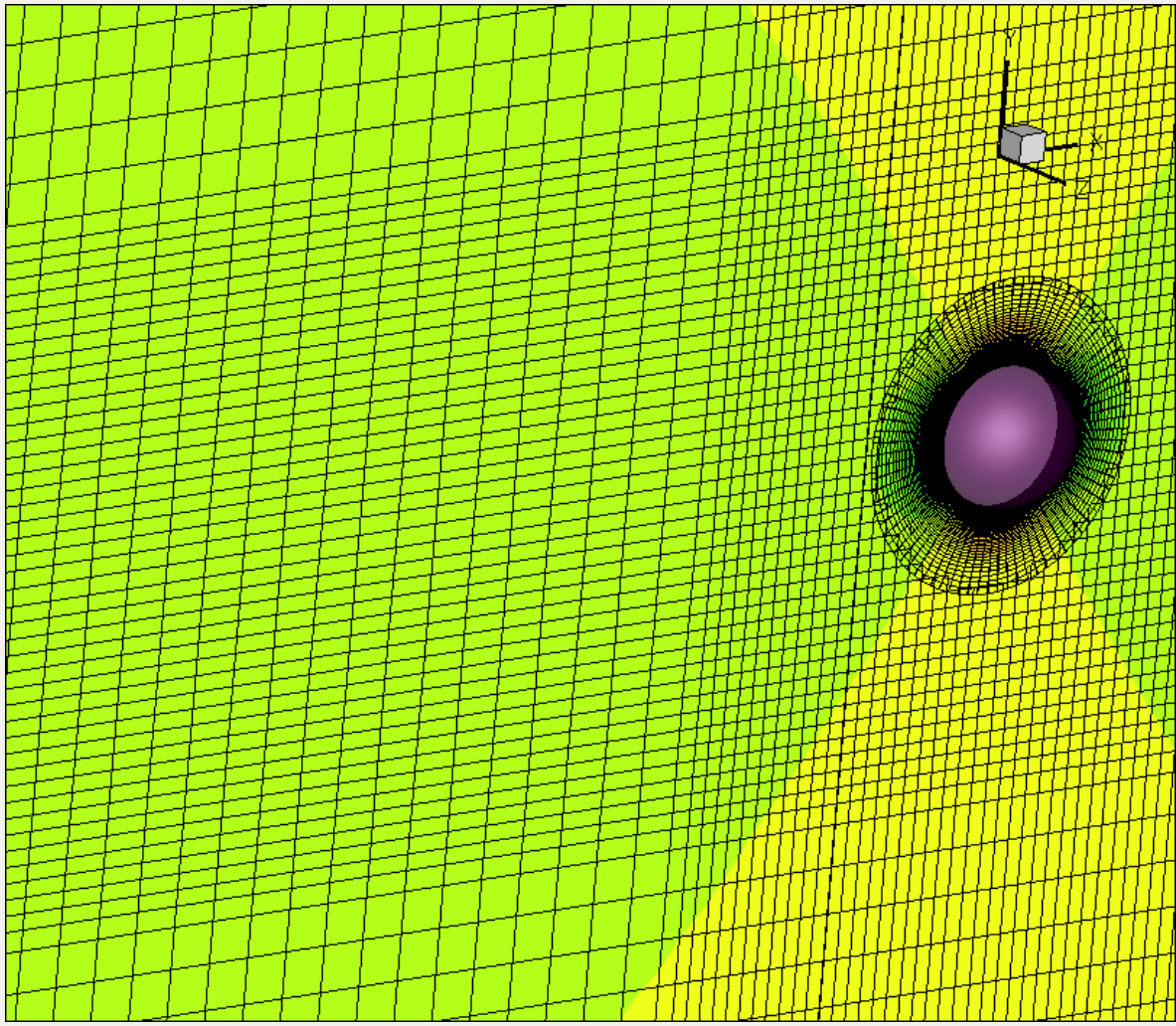


HELIOS Multidisciplinary Software



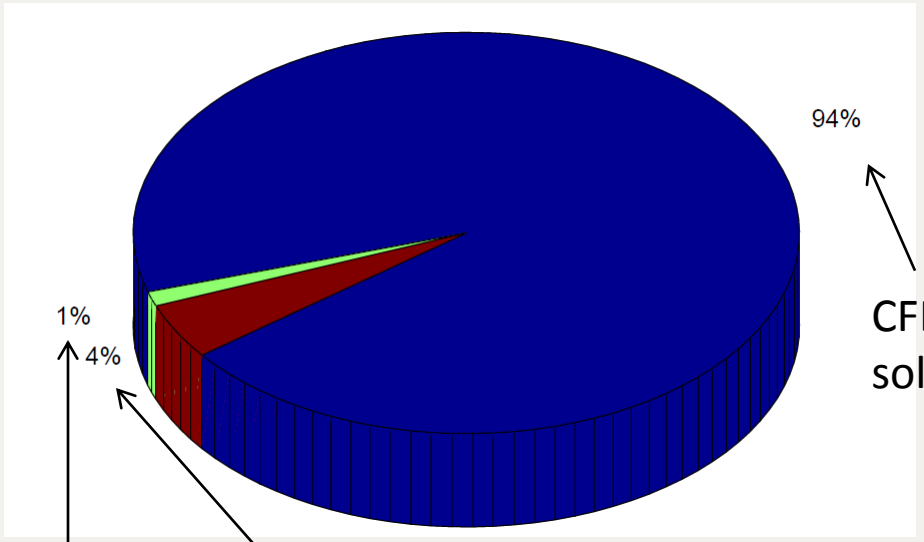
Software Complexity for Heterogeneous Architectures (GPUs)

- Overlapping mesh
- Overlap/Interpolation patterns recomputed at each time step
- CFD performed on GPU
- Mesh assembly performed on GPU



GPU vs CPU Performance(3D)
(0.93 million points – moving sphere)
(Chandar, Sitaraman and Mavriplis, 2012)

CPU



CFD solver

preprocessing

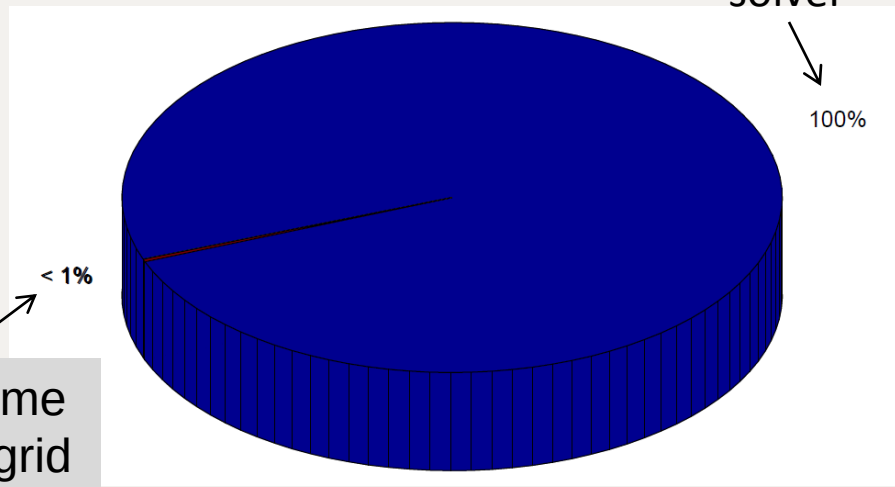
Domain connectivity

Total about 5% time

Overall speedup :
- 9x faster for the solver
- 100x faster for overset assembly
- overall **10x** faster

GPU

CFD solver

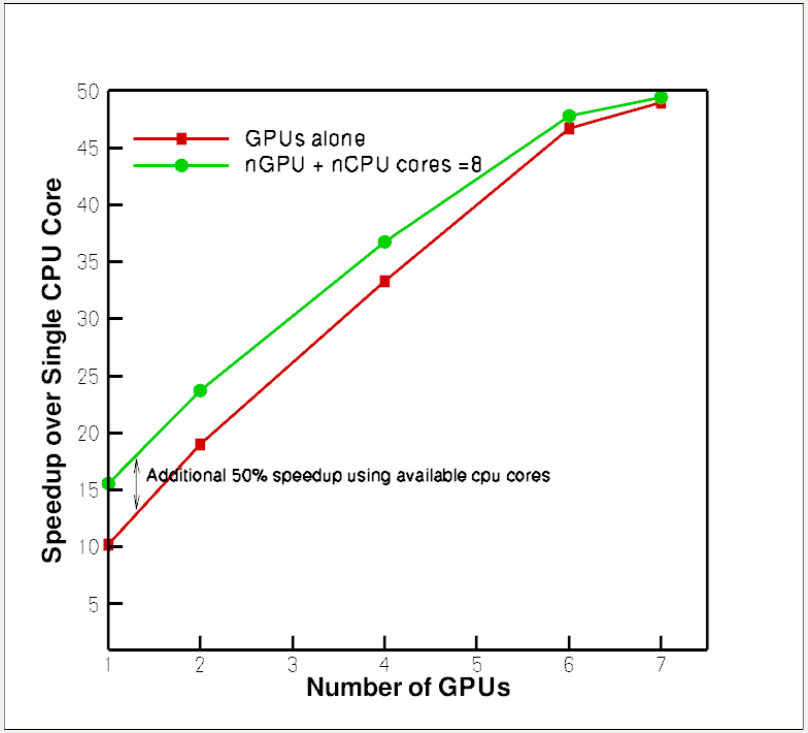


Negligible time for overset grid assembly (0.2%)

Masking Heterogeneity with C++ Expression Templates

Flow solver implemented in C++ using expression templates

- Low level CUDA code written at template level
- Select CPU or GPU version at compile time
- Same source code CPU or GPU capable (or both simultaneously)



- Additional speedup using all available cores
- Hedge against future architecture trends
- Not applicable to legacy codes !!
- Afraid to commit to $O(10^6)$ line code to highly custom hardware

Software Complexity for HPC

"MPI will suffice for a few stunts.

- MPI + OpenMP per socket isn't much better"

"God forbid, but is CUDA the model for the future..."

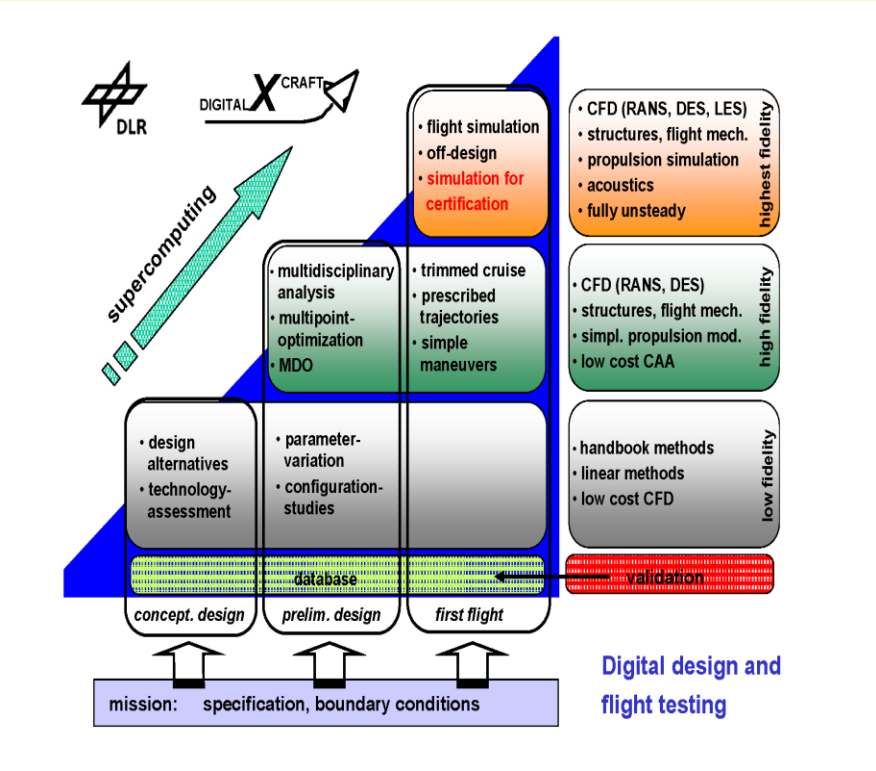
- Challenge: Hardware is hierarchical, but MPI, OpenMP and UPC are "flat"
- Algorithm design must adapt
  - Traditionally HPC hardware comes first, algorithms never catch up.
    - 1st applications on new hardware are generally parametric studies
    - Good development environment and efficient codes at about 3-4 yrs
    - Software cycle (10-15yrs) is longer than hardware cycle (~4 yrs)
    - Immediate need for research into algorithms that respect hierarchical nature of foreseeable hardware

'Bad News:

Even after 20 years, we're still not done porting codes to parallel systems"

Conclusions

- Exascale will enable revolutionary capabilities in aerospace analysis, design, understanding, capabilities
  - **Decadal Survey of Civil Aeronautics (NAE):** ***"…****an important benefit of advances in physics-based analysis tools is the new technology and systems frontiers they open"*
- Achieving exascale for aeronautical /aerospace applications will be very challenging



Conclusions

- Past NASA funding is responsible for many of the HPC advances in use today
  - ICASE in the 1990's

Full Aircraft Simulation on 512 processors of Intel Delta; Sussman, Saltz, Das, Gupta, Mavriplis,Ponnusamy. ICASE 1992

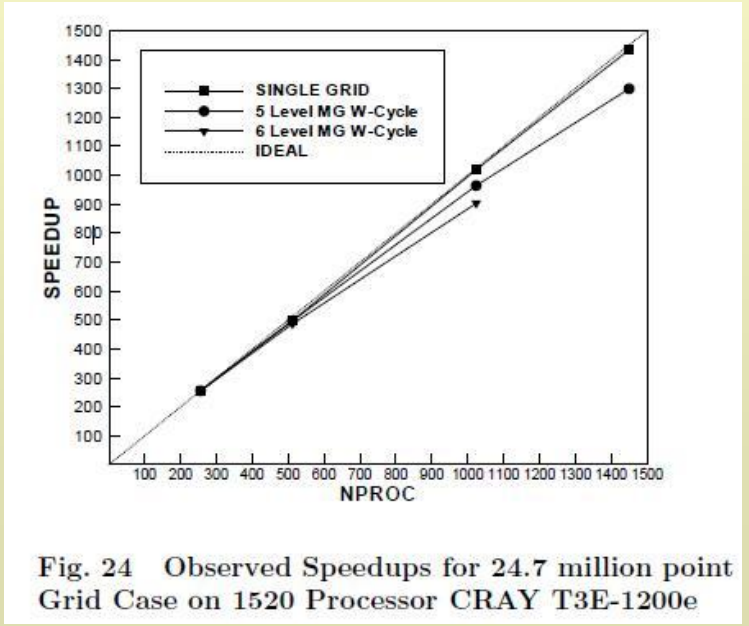


Fig. 24 Observed Speedups for 24.7 million point Grid Case on 1520 Processor CRAY T3E-1200e

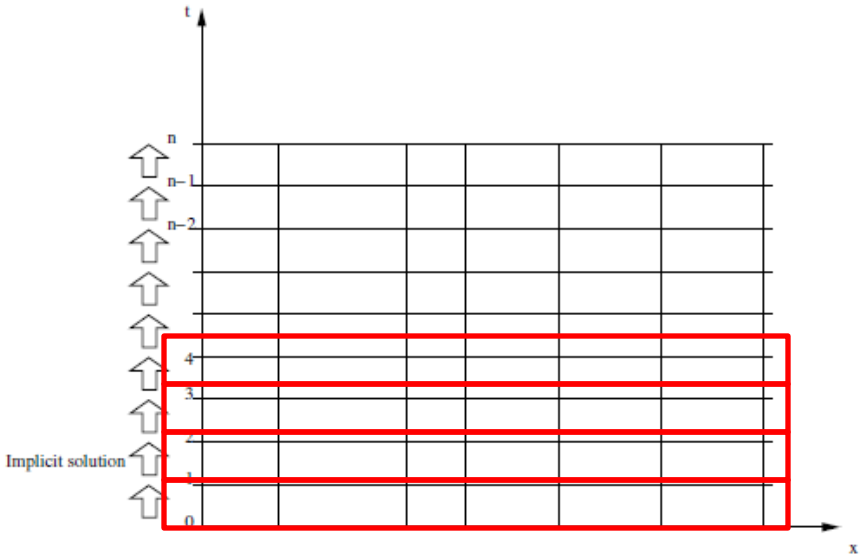
Mavriplis and Pirzadeh AIAA-1999-0537

Conclusions

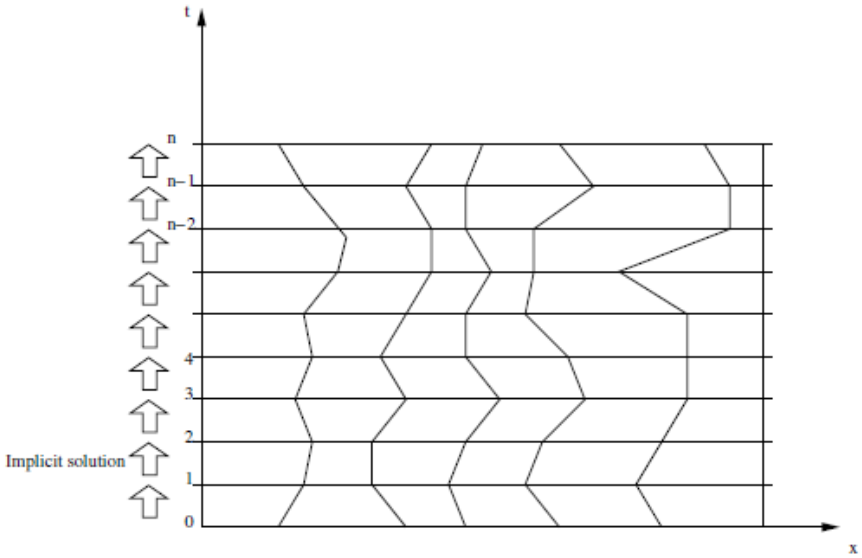
- ICASE in the 1990's
  - High performance fortran (HPF)
  - PARTI: encapsulating parallelism prior to MPI standard
  - Provided access to emerging architectures
  - Ardent, Stardent, Intel Hypercube, Intel Touchstone Delta, Connection Machine
  - Housed one of the first "Beowulf" clusters
- Are we meeting the challenge today ?
- At the very least, the aerospace community should participate forcefully in national exascale initiatives.

Adaptive Implicit Space-Time Methods

Traditional approach to solving unsteady problems (An illustration in 1D space):



(a) Spatially uniform time stepping without mesh deformation



(b) Spatially uniform time stepping with spatial mesh deformation

Define residual operator R at each time-step:

$$\mathbf{R}^n(\mathbf{U}^n, \mathbf{U}^{n-1}, \mathbf{x}^n, \mathbf{x}^{n-1}) = 0$$

Linearize and solve using Newton iterations:

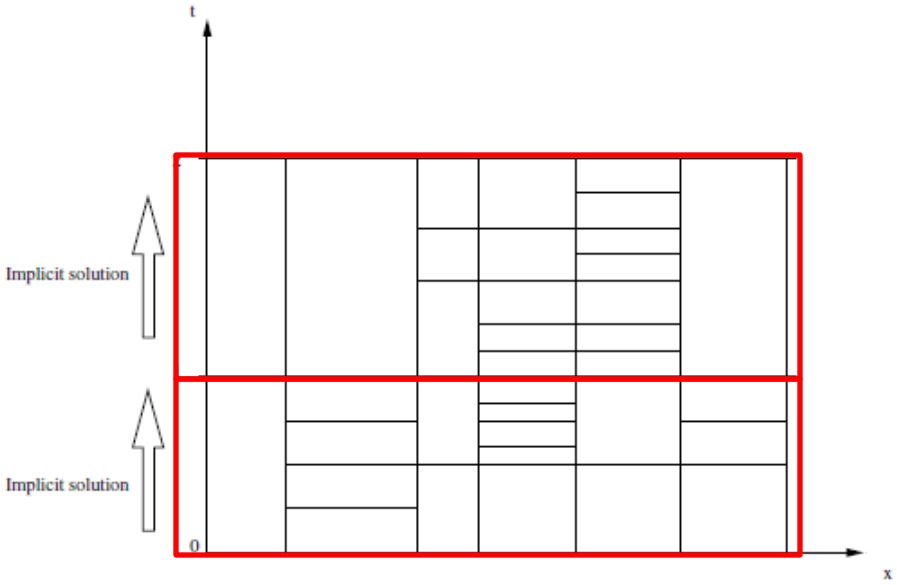
$$\left[\frac{\partial \mathbf{R}(\mathbf{U}^k, \mathbf{x})}{\partial \mathbf{U}^k}\right] \delta\mathbf{U}^k = -\mathbf{R}(\mathbf{U}^k, \mathbf{x})$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k + \delta\mathbf{U}^k$$

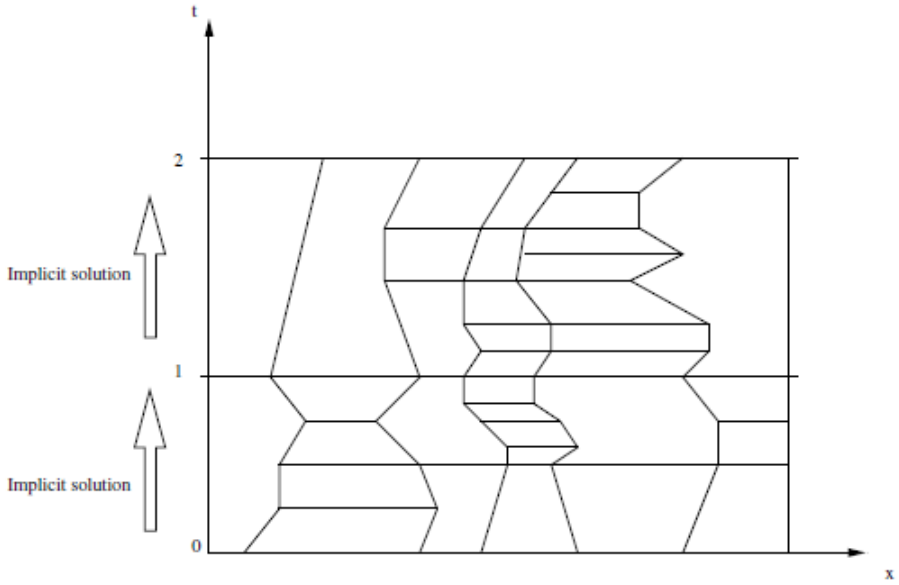
$$\delta\mathbf{U}^k \to 0, \mathbf{U}^{k+1} = \mathbf{U}^n$$

Spatially Non-Uniform Time-Stepping

The space-time slab-based unsteady solution process (An illustration in 1D space):



(a) Spatially nonuniform time stepping without mesh deformation



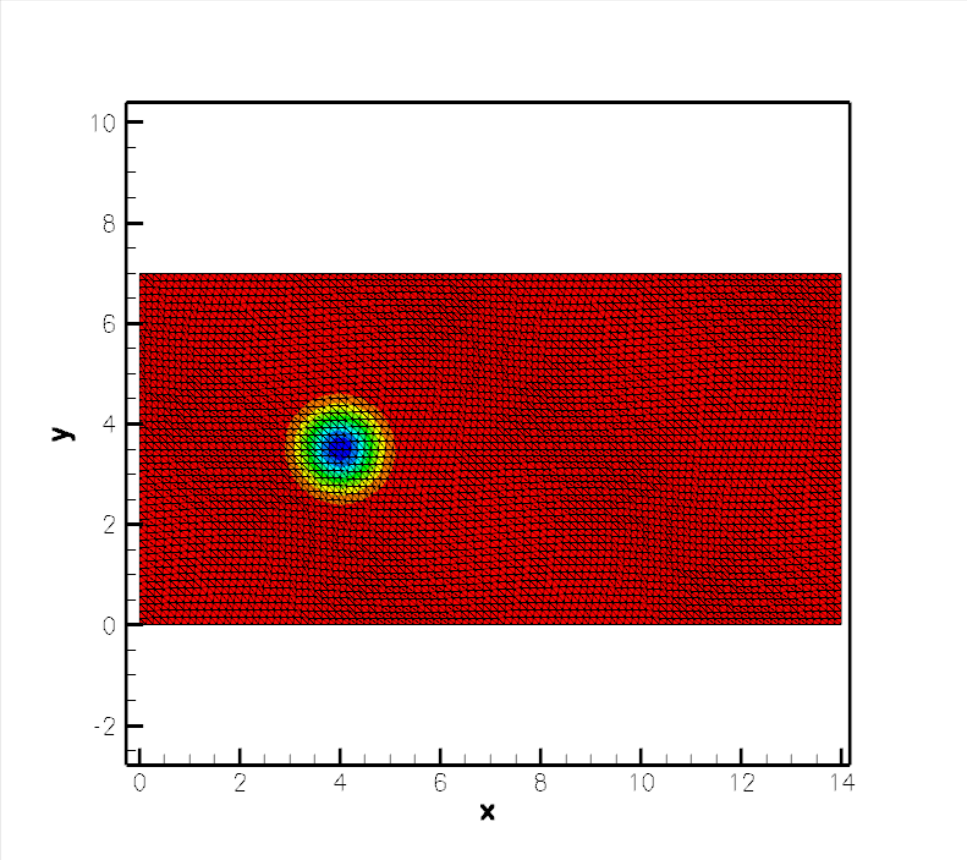
(b) Spatially nonuniform time stepping with spatial mesh deformation

Define residual operator R over whole slab and solve for all U within slab (which are unknown) in one-shot using Newton iterations.

Results

Isentropic Vortex Convection

Vortex seeded at [3.5,3.5] -> allowed to convect with free-stream
Prescribed mesh deformation through whole domain -> no mesh solution required
Time domain from [0,15]



Case Description

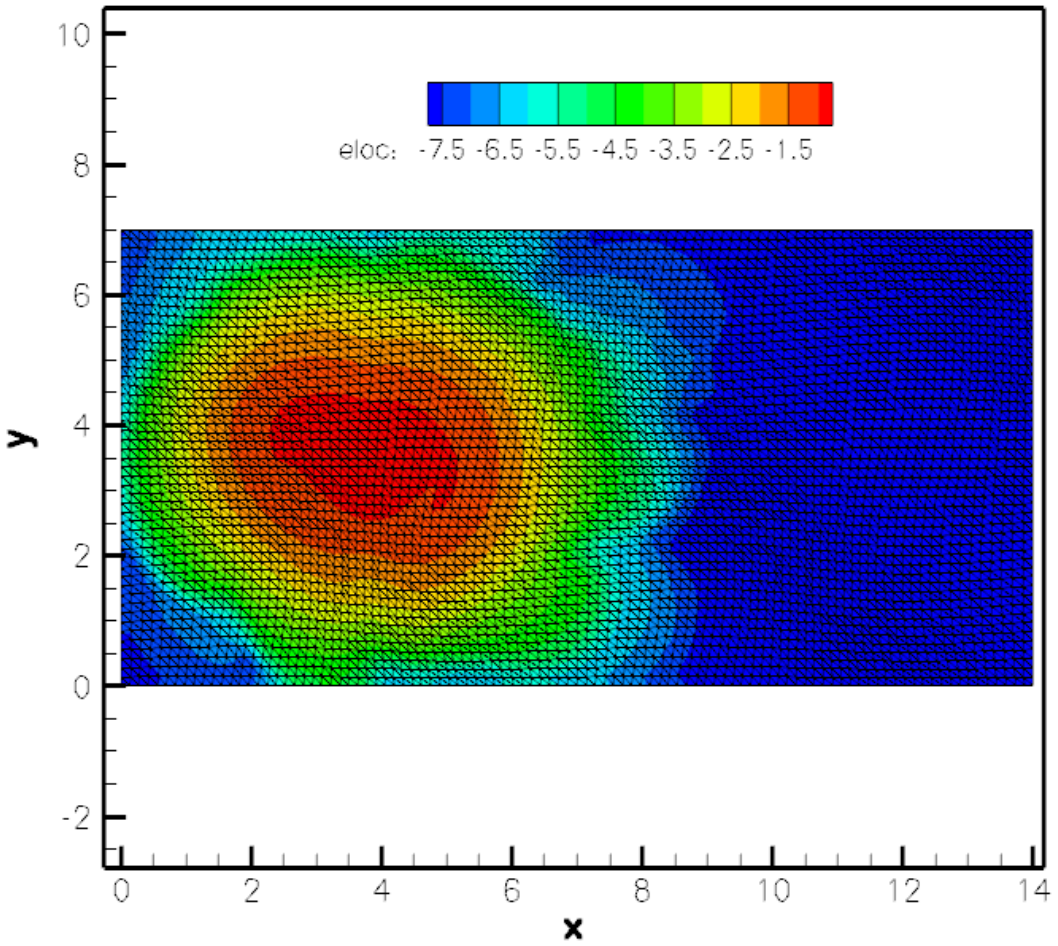
Local error indicator used for adapting time-step for each element:

$$e_{local} = \left\| \left[\frac{d\mathbf{U}}{dt}\right]_{BDF2} - \left[\frac{d\mathbf{U}}{dt}\right]_{BDF1} \right\|_2$$

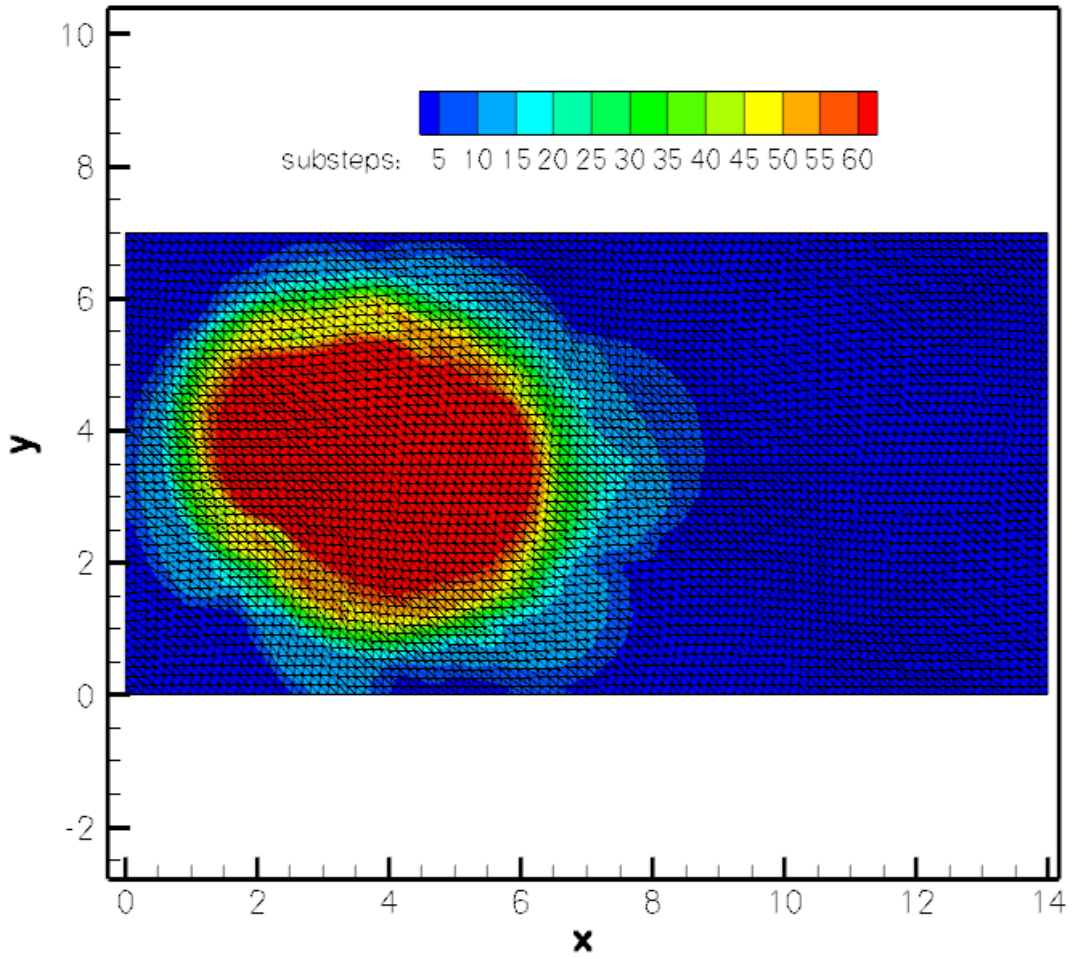
Strictly true for static meshes. Used here as an approximation.

- Starting solution for non-uniform case -> 15 slabs -> thickness 1 each.
- Starts with each spatial element in each slab taking 2 time-steps of 0.5.
- Uniform case starts with 30 time-steps of 0.5 for all spatial elements.
- Uniform case doubles number of time-steps at each adaptation cycle.
- Non-uniform case adapts based on local-error indicator.
- Comparisons made against reference solution with 30,720 time-steps.

Local Error Animation (final adaptation)



Sub-Steps Animation (final adaptation)



Density Comparison

Centerline density comparison at top of each slab against uniform case