

Progress towards a High-Accuracy 4D Computational Aeroacoustics (CAA) Prediction Tool

Ray Hixon

Acoustics Branch
NASA Glenn Research Center
and
Mechanical, Industrial, and Manufacturing Engineering (MIME) Department
University of Toledo

Duane.Hixon@utoledo.edu

Acoustics Technical Working Group
NASA Glenn Research Center

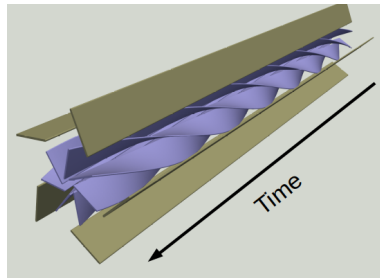
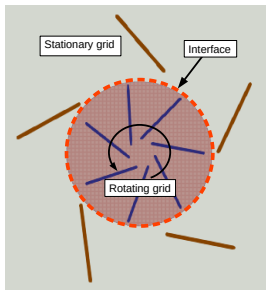
November 1, 2023

Acknowledgements

- This work is supported by the NASA Advanced Air Transport Technology (AATT) Project.
- Dr. Edmane Envira of the NASA Glenn Research Center is the technical monitor for this work.

The Story So Far...

- The goal: High-accuracy predictions of unsteady flow and noise...
- The idea: Solving the problem in 4D space-time provides new efficiencies for unsteady flow prediction.



A revolutionary CAA prediction tool is being developed at NASA

The Advantages of Space-Time

- Some of the practical advantages of the space-time approach are:
 - Can use a *lot* of existing technology:
 - Robust and efficient CAA differencing schemes.
 - Fast CFD steady-flow convergence techniques.
 - Smooth elliptic grid generation methods.
 - Gives a pathway to quantum computing...
 - This also sidesteps a *lot* of existing problems with time marching:
 - How to resolve a range of time scales?
 - How to transfer data efficiently between moving and stationary grids?
 - How to achieve large-scale parallel efficiency?

The space-time method could solve a lot of problems!

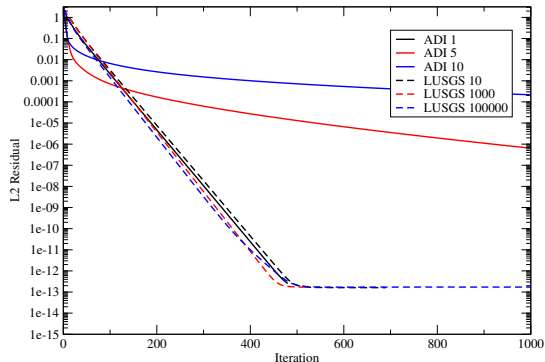
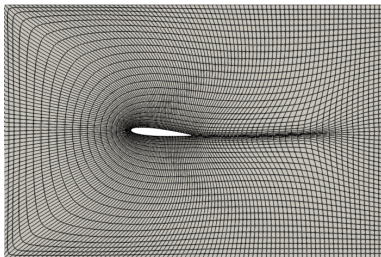
The Challenges of Space-Time

- There are a *lot* of questions that must be answered! For example:
 - Structured grid generation:
 - Do the existing 3D CFD convergence methods work for 4D elliptic equations? (January 2023)
 - Need **automatic structured** grid generation for arbitrary configurations **in 4D!** (2D work in progress (Gabe Gutierrez))
 - Need **automatic grid adaption** to the flow solution **in 4D!**
 - Flow solver:
 - Do the existing implicit CFD convergence methods work for high-accuracy differencing methods? (1D testing complete (Zaid Sabri): March 2023; 4D work in progress)
 - How to maintain accuracy on complex grids? (August 2023)
 - What are the 4D flow boundary conditions?

There are a lot of possible showstoppers on the way!

Grid Generation in 4D (January 2023)

- A 4D single-block C-grid was generated about a pitching NACA 0012 infinite wing.
- Both the ADI and LU-SGS methods successfully converged the grid.



The grid generation equations were converged to machine precision!

Implicit Convergence Methods (Sabri)

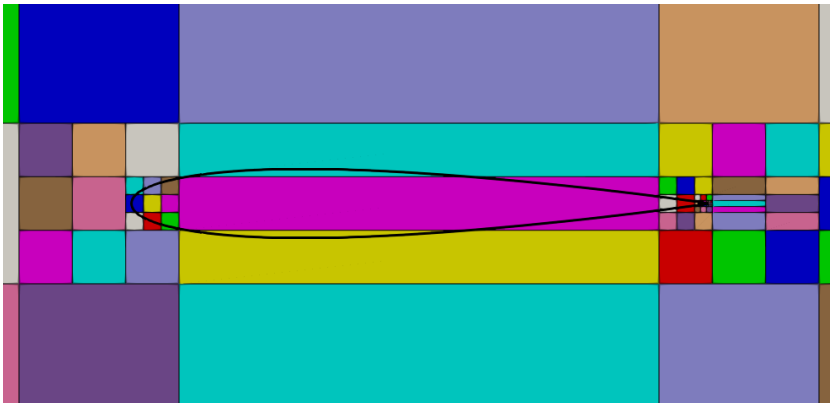
- The objective is to leverage and extend previous 3D approaches for the 4D flow solver:
 - Use high-order differencing schemes for efficiency and solution accuracy.
 - Use low-order small-stencil differencing schemes in the iterative solver for fast and robust convergence.

$$\begin{bmatrix} \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & & & \\ \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & & & \\ \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & & \\ \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & -\alpha & -\kappa & -\tau & I & \tau & \kappa & \alpha & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ 0 & & & & & & & & & 0 \end{bmatrix} \xrightarrow[\text{as:}]{\text{Approximated}} \begin{bmatrix} \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & & & & & \\ \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & & & & \\ & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & & & \\ & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & & & & & \\ & & & -\beta & I & \beta & & & & \\ & & & & \ddots & \ddots & \ddots & & & \\ & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & & & & \begin{bmatrix} \end{bmatrix} & \begin{bmatrix} \end{bmatrix} \\ & & & & & & & & & \begin{bmatrix} \end{bmatrix} \\ 0 & & & & & & & & & 0 \end{bmatrix}$$

Three stable candidate methods have been verified in 1D

Automatic Topology Generation (2D)

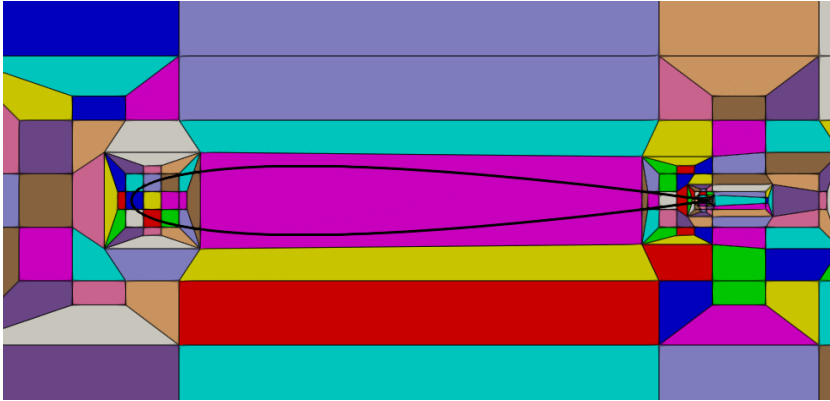
- A coarse-grain Cartesian grid method is used to define the grid blocks...



The Cartesian method 'cleanly' extends to 3D and 4D!

Automatic Topology Generation (2D)

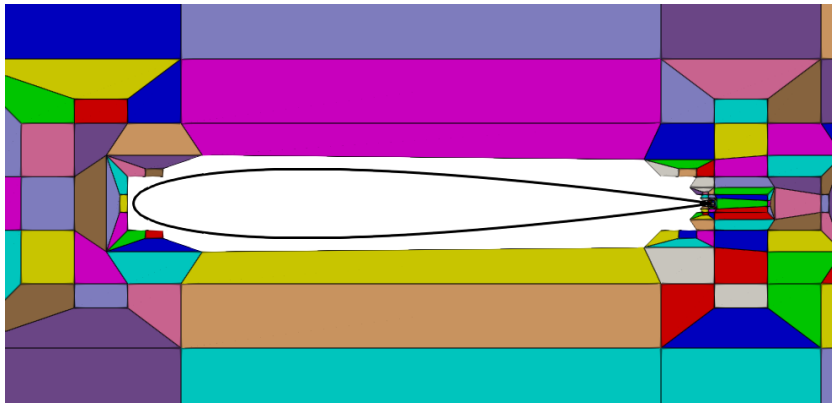
- 'Patch templates' are used to obtain a full-face-matching grid topology...



Patch templates have been defined for 3D and 4D...

Automatic Topology Generation (2D)

- The surfaces are cut out...



Sharp corners on the surface need to be radiused...

Automatic Topology Generation (2D)

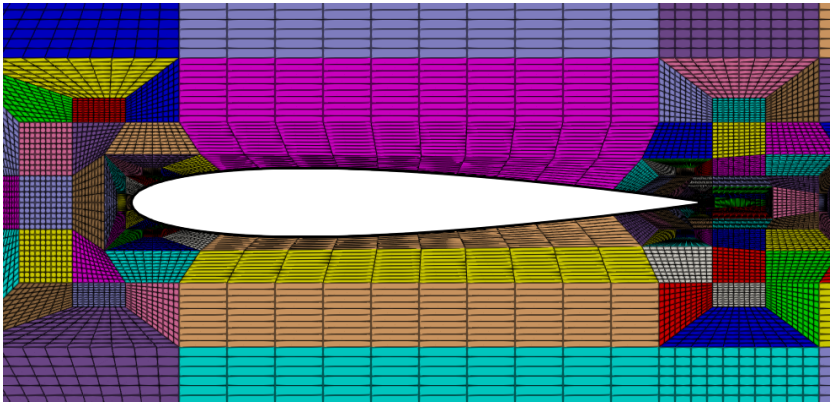
- The 'open' block faces are extruded to obtain a body-fitted topology...



The result is a body-fitted topology for arbitrary configurations!

Automatic Topology Generation (2D)

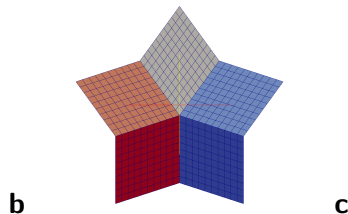
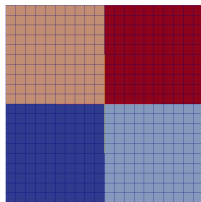
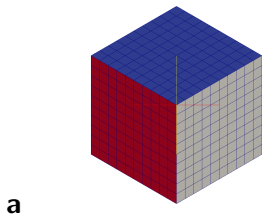
- Algebraic transfinite interpolation (TFI) is used to define the initial grid, which will be smoothed by the elliptic grid generator before use.



This gives a path to automatic 4D structured grid generation...

Multiblock Grid Singularities (2D)

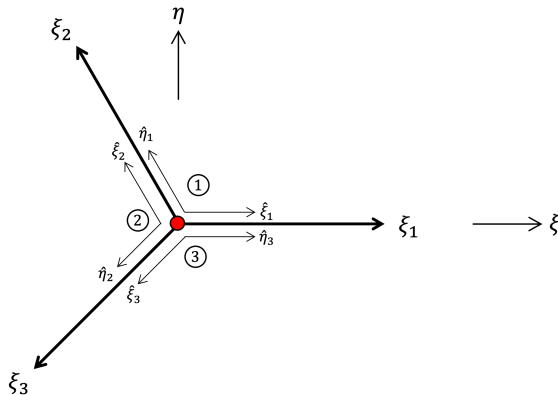
- Structured schemes drop to 1st order at grid singularities.
- These are examples of 3-way (a), 4-way (b), and 5-way (c) grid singularities at the corners of 2D blocks



There are a lot of grid singularities in the automatic topologies!

The SMART Method

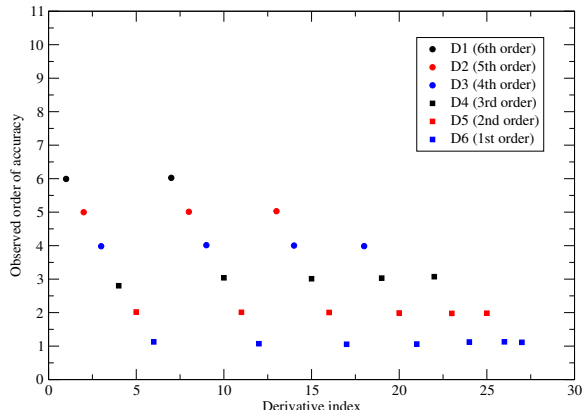
- The **S**tructured **M**apping **A**t **R**educed dimension **T**opological elements (SMART) method obtains high-order accuracy at grid singularities.



Special curvilinear mappings are defined at block boundaries...

SMART results for 6th order (2D)

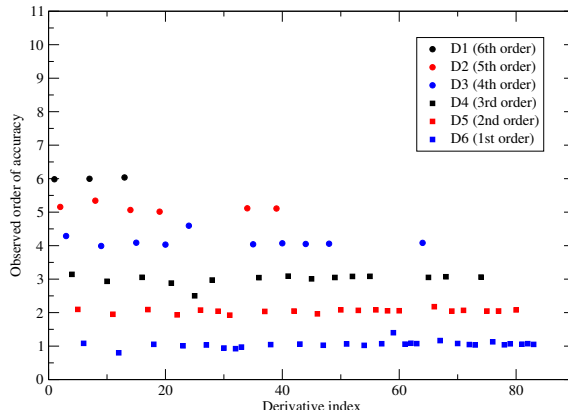
- These are SMART results for a 3-way interior point singularity in 2D:



A 2D nonsingular corner point is 4-way...

SMART results for 6th order (3D)

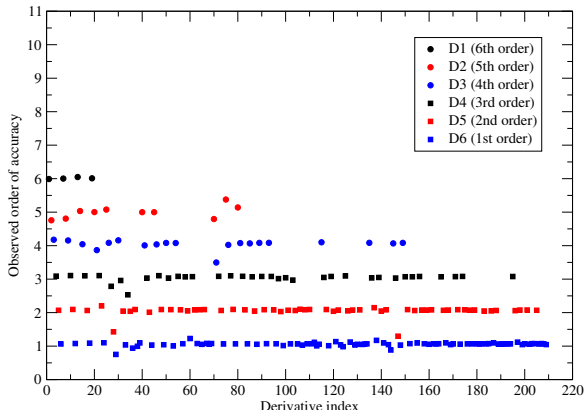
- These are SMART results for a 4-way interior point singularity in 3D:



A 3D nonsingular corner point is 8-way...

SMART results for 6th order (4D)

- These are SMART results for a 5-way interior point singularity in 4D:



A 4D nonsingular corner point is 16-way...

Coding and testing the SMART Method

- The SMART method by the numbers:
 - Original grid singularity issues found in 2003 (NASA BASS code).
 - Formulation and initial hardwired testing for SMART method: 4 months (starting August 2022).
 - Generalized SMART object coding time: 7 months (my initial estimate: two weeks!)
 - Lines of code since February 2023 (generalized SMART object and testing routines only): 103,000 lines.
 - From the word count, the SMART object is a 640 page novel in Fortran2008...

Next question: is SMART stable in an iterative solver???

Coming Attractions

- The next steps in the process:
 - Multiblock implementation and testing of 4D flow solver
 - Code parallelization
 - Multigrid and automatic grid sequencing
 - 4D automatic topology generation and improved grid generation
 - Grid adaption

All objects are coded in 2D, 3D, and 4D for use in other codes...

Questions?