SAND2022-7079C

Sandia National Laboratories

Exceptional service in the national interest

Code Verification Implications for Algebraic Equations

Aaron Krueger, Brian Freno, and Blake Lance

ASME VVUQ 2022 Symposium

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Multifidelity background

- Algebraic models can often be found as part of multifidelity modeling
  - Uses a combination of high, medium, and low fidelity models to make predictions
  - The fidelity level refers to the relative level of physics captured by the model
  - Capturing more physics costs more computational time, so a balance must be reached
- When using multiple models, multiple code verification strategies might have to be leveraged
  - Specifically, how do we do code verification for algebraic models?

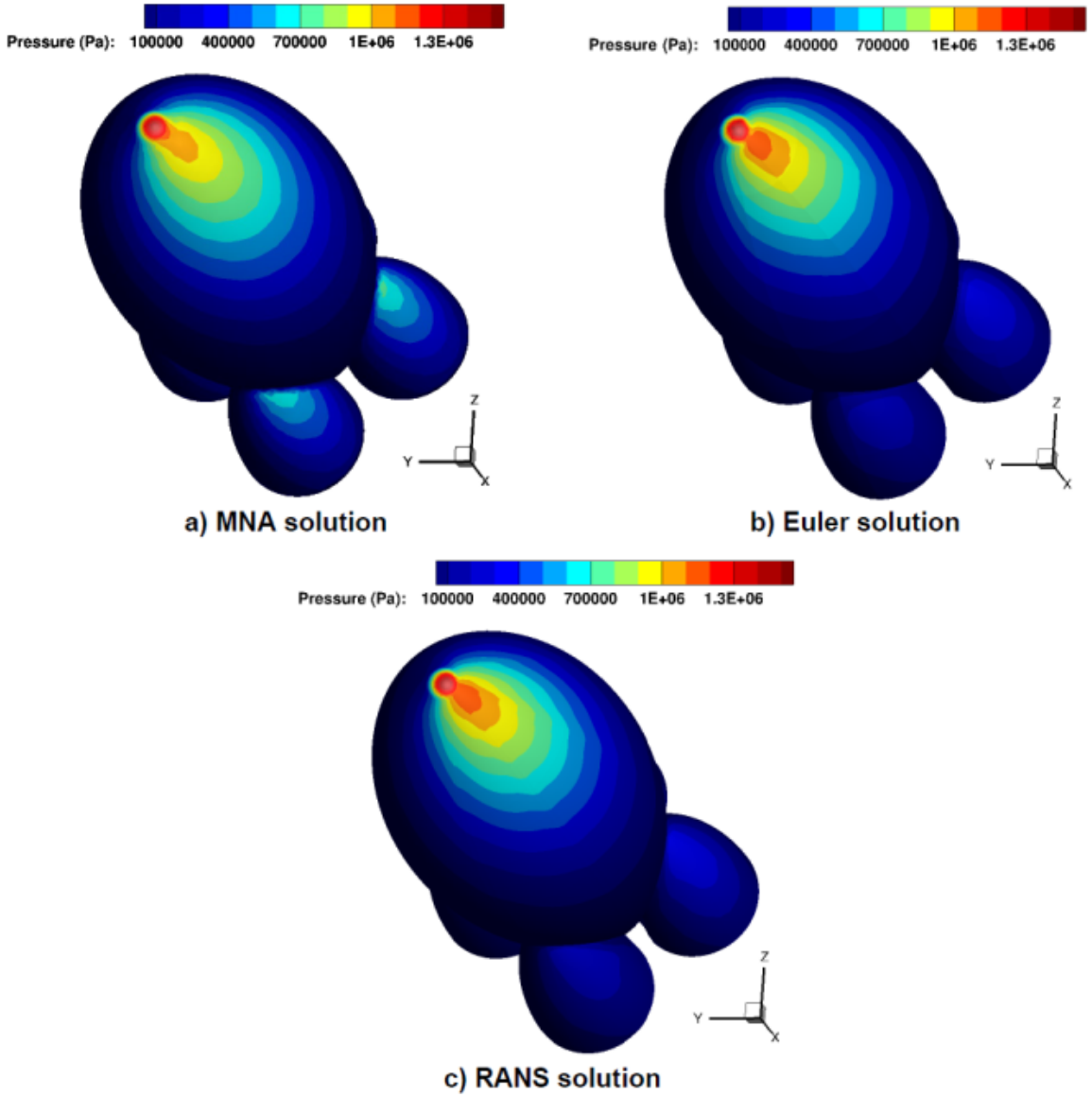


Figure 1: Example of Multifidelity Toolkit (MFTK) results for pressure[1]

[1]Wagnild, R. M., Dinzl, D. J., Bopp, M. S., Dement, D. C., Robbins, B. A., Bruner, C.W. S., Grant, M. J., Murray, J., and Harper, J. M., "Development of a Multi-fidelity Toolkit for Rapid Aerothermal Model Development," Sandia Report SAND2019-13632, Sandia National Laboratories, Oct 2019.

What is code verification?

- According to ASME V&V 20, " Code verification establishes that the code accurately solves the mathematical model incorporated in the code (i.e., that the code is free of mistakes for the simulations of interest)".
- ASME V&V 20 also says "Code verification, establishing the correctness of the code itself, can only be done by systematic discretization convergence tests and monitoring the convergence of the solutions towards a known "benchmark" solution (i.e., a standard of comparison)."
- What if a computational model doesn't have any discretization error? How do we do code verification?

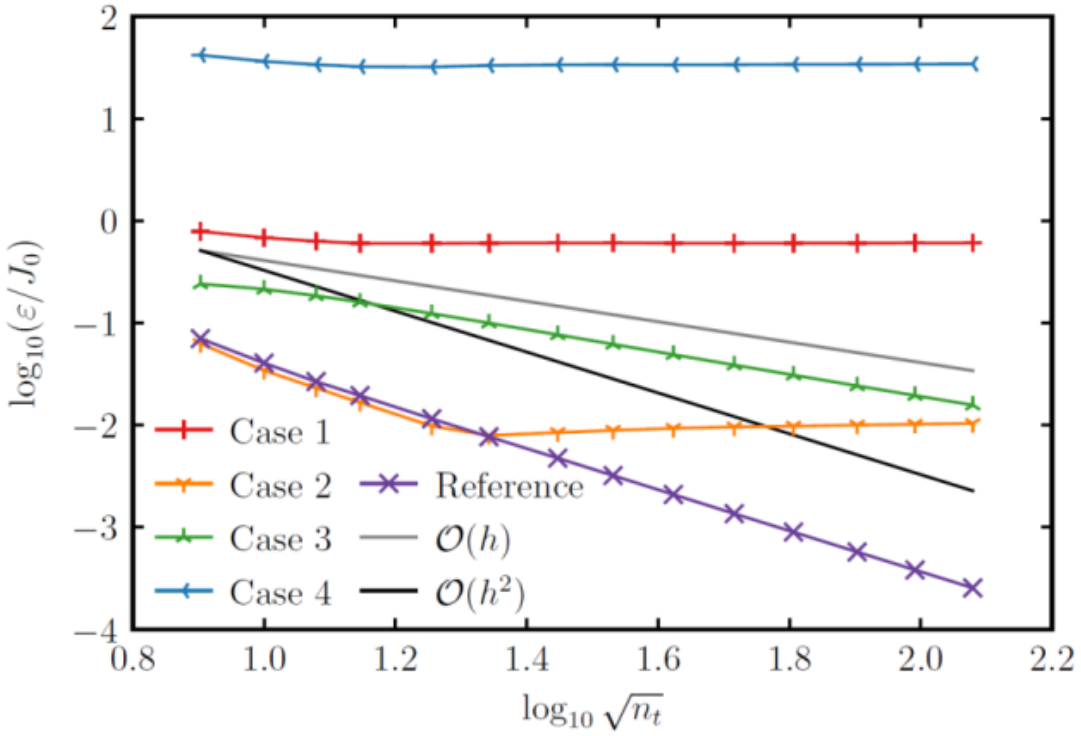
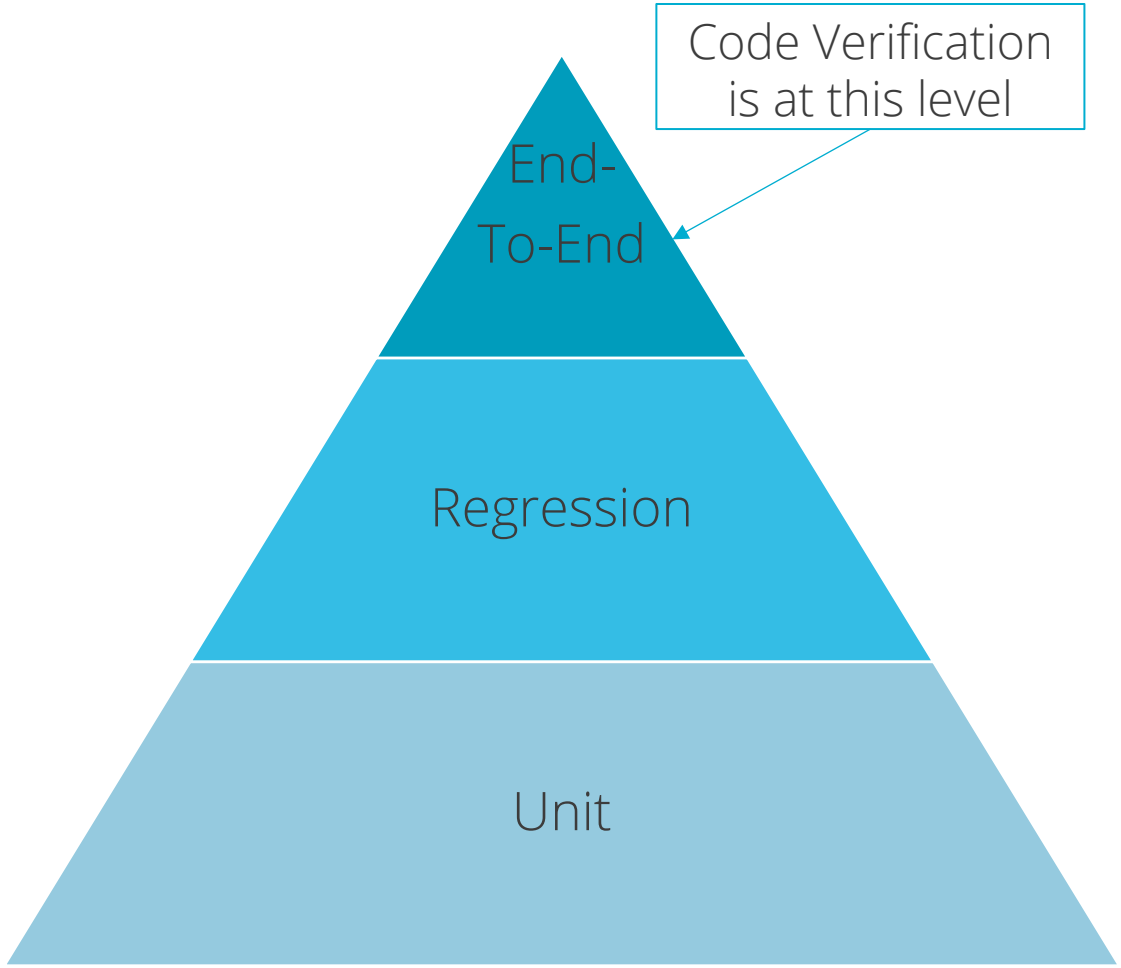


Figure 2: Convergence example with and without coding errors[2]

[2] Freno, Brian A., Carnes, Brian R., and Weirs, V. Gregory. "Code-verification techniques for hypersonic reacting flows in thermochemical nonequilibrium." *Journal of Computational Physics*, Vol. 425, Jan. 2021.

What should the testing strategy be for algebraic models?

- Is it a regression test?
  - Regression testing is “Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.”[3]
  - This is typically an automated relative test that compares today’s result with yesterday’s result, rather than comparing with some known true solution
- Is it a unit test?
  - Unit testing is “Testing of individual hardware or software units or groups of related units.” [3]
  - This is testing of individual functions rather than a large portion of the code
- We need a test that covers a large portion of the code and compares to a true solution
  - This type of testing is with the spirit of code verification, but different in which metric to measure



[3]IEEE StD 610.12-1990, “IEEE Standard Glossary of Software Engineering Terminology

Using analytic solutions for algebraic models

- Since an analytic solution is readily available for algebraic models, comparing the code solution to the analytic solution is straight forward
  - This comparison should match exactly up to round-off precision
  - Since we are comparing the solutions directly, only one mesh is required (more refinements confirm results)
  - Unlike the method of manufactured solutions (MMS), no right hand side term is needed
- To test out this strategy, we'll apply this code verification technique to a low-fidelity model in a high-speed compressible flow code
- Testing will start simple with additional complexity added later
- Complete a validation assessment before and after code verification testing

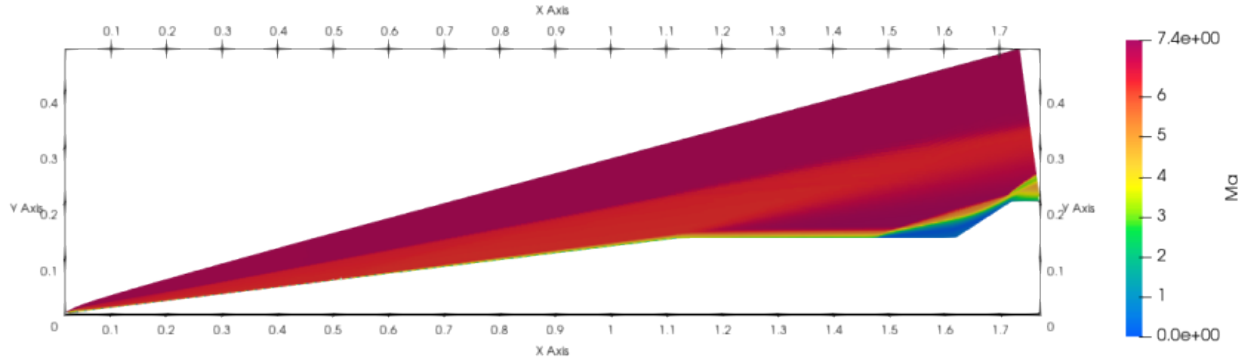


Figure 3: HIFiRE-1 wind tunnel simulation Mach number predictions for RANS-SST[4]

[4]B. W. Lance, A. M. Krueger, B. A. Freno, R. M. Wagnild, Verification and Validation Activities for the Multi-Fidelity Toolkit, Tech. Rep. SAND2022-1479, Sandia National Laboratories, Albuquerque, NM, Feb 2022.

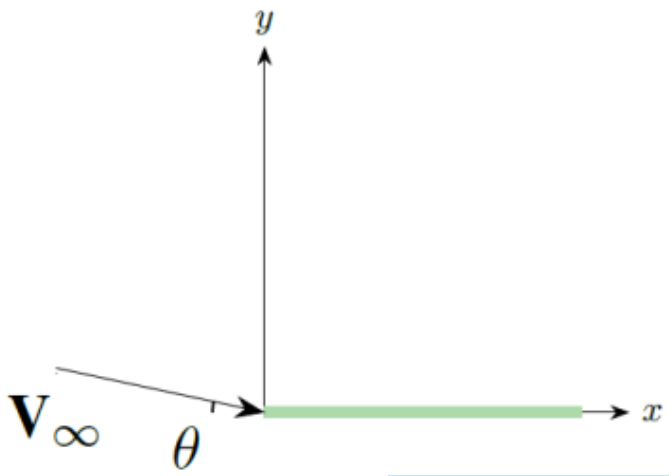
High-speed compressible flow problem set up

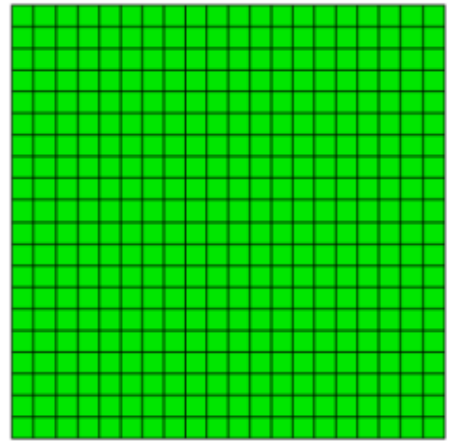
- Two separate problems in MFTK were analyzed
  - Flat plate
  - Inclined plate
- Only one mesh was required since discretization errors were not present
- Using analytical solutions, the maximum relative error is identified

$$\varepsilon_\infty = \max_i \left| \frac{QoI_{i_{exact}} - QoI_{i_{SPARC}}}{QoI_{i_{exact}}} \right|$$

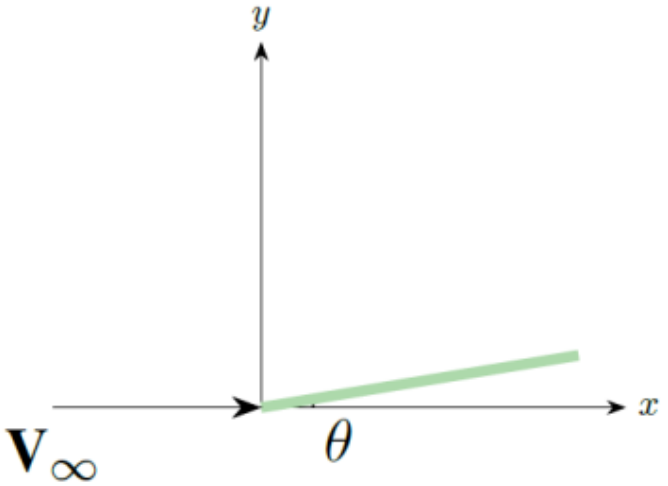
- For problems without discretization errors, ε_∞ should be on the order of round-off error (10^{-10})
- $C_p, P_e, \boldsymbol{V}_e, M_e, T_e, \rho_e, \boldsymbol{n}_v$, Dist, τ, and q_w are tested

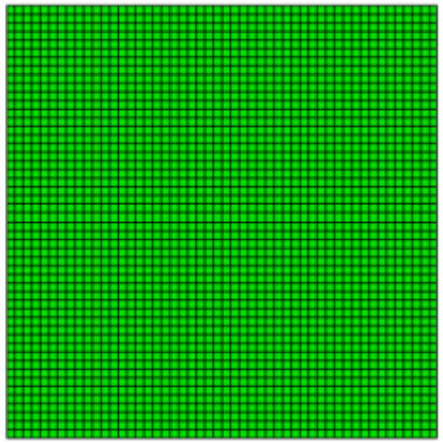
Flat Plate Problem





Inclined Plate Problem

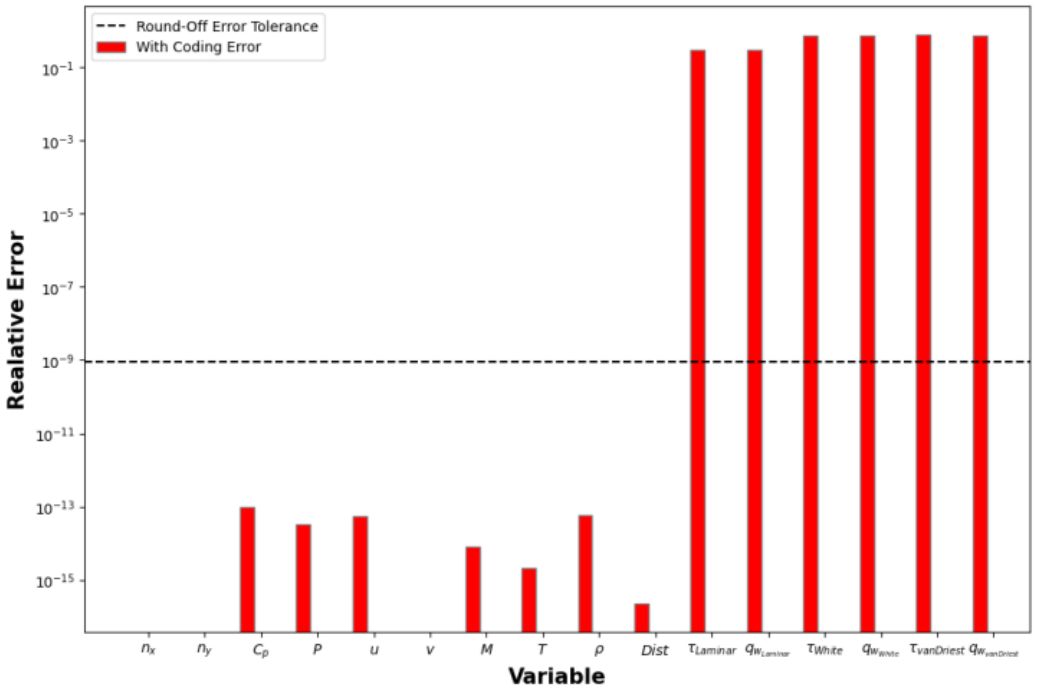




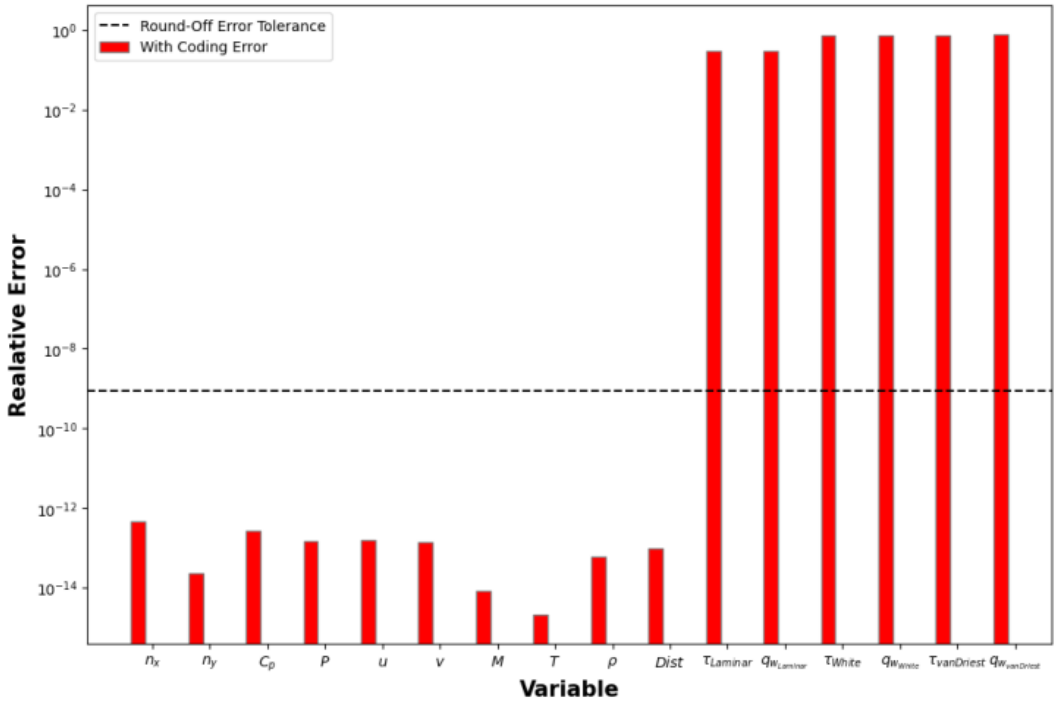
Initial code verification results

- Code verification of inviscid variables have been previously completed
- Compute ε_∞ for each QoI
  - Coding error exists if $\varepsilon_\infty > 10^{-10}$
  - Coding error does not exists if $\varepsilon_\infty < 10^{-10}$

- The shear stress and heat flux for all three viscous models have coding errors
- Debugging of these models can now start



Flat Plate Problem



Inclined Plate Problem

Code bugs found

Code Bug in Laminar Equations

- A code bug was identified in the laminar coefficient of skin friction calculation
  - Impact shear stress and heat flux
  - All other inputs into this equation were verified

$$C_f = \frac{0.664\sqrt{C^*}}{\sqrt{Re_{xe}}}$$

$$C^* = \frac{\rho^* \mu^*}{\rho_e \mu_e}$$

$$\mu^*_{bug} = C_{visc} \boldsymbol{T_w} \frac{\sqrt{\boldsymbol{T_w}}}{\boldsymbol{T_w} + S_{visc}} \qquad \mu^*_{correct} = C_{visc} \boldsymbol{T^*} \frac{\sqrt{\boldsymbol{T^*}}}{\boldsymbol{T^*} + S_{visc}}$$

Code Bug in Turbulence Equations

- A code bug was identified in the turbulent coefficient of skin friction calculation
  - Impact shear stress and heat flux
  - All other inputs into this equation were verified

$$C_f \approx \frac{0.455}{S^2 \ln^2\left(\frac{0.06}{S} Re_{xe} \frac{\mu_e}{\mu_w} \sqrt{\frac{T_e}{T_w}}\right)}$$

$$S_{bug} = \frac{\sqrt{\frac{T_{aw}}{\boldsymbol{T_w}}}}{\sin^{-1} A + \sin^{-1} B} \qquad S_{correct} = \frac{\sqrt{\frac{T_{aw}}{\boldsymbol{T_e}}}}{\sin^{-1} A + \sin^{-1} B}$$

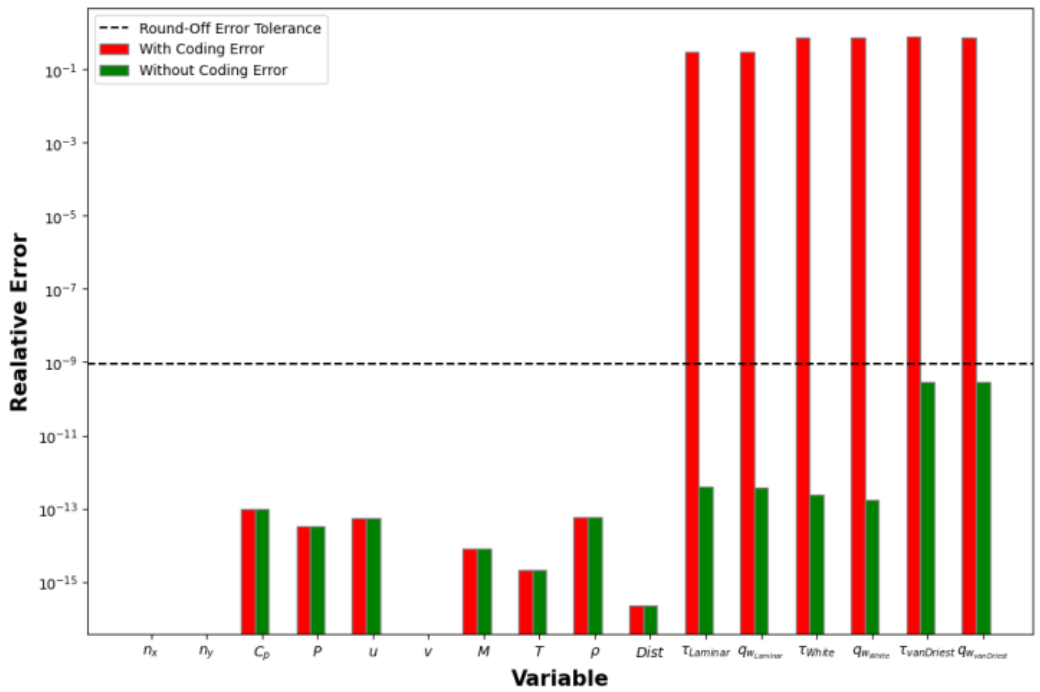
$$A = f(b) \quad \text{and} \quad B = f(b)$$

$$b_{bug} = \frac{T_{aw}}{T_w} \qquad b_{correct} = \frac{T_{aw}}{T_w} - \boldsymbol{1}$$

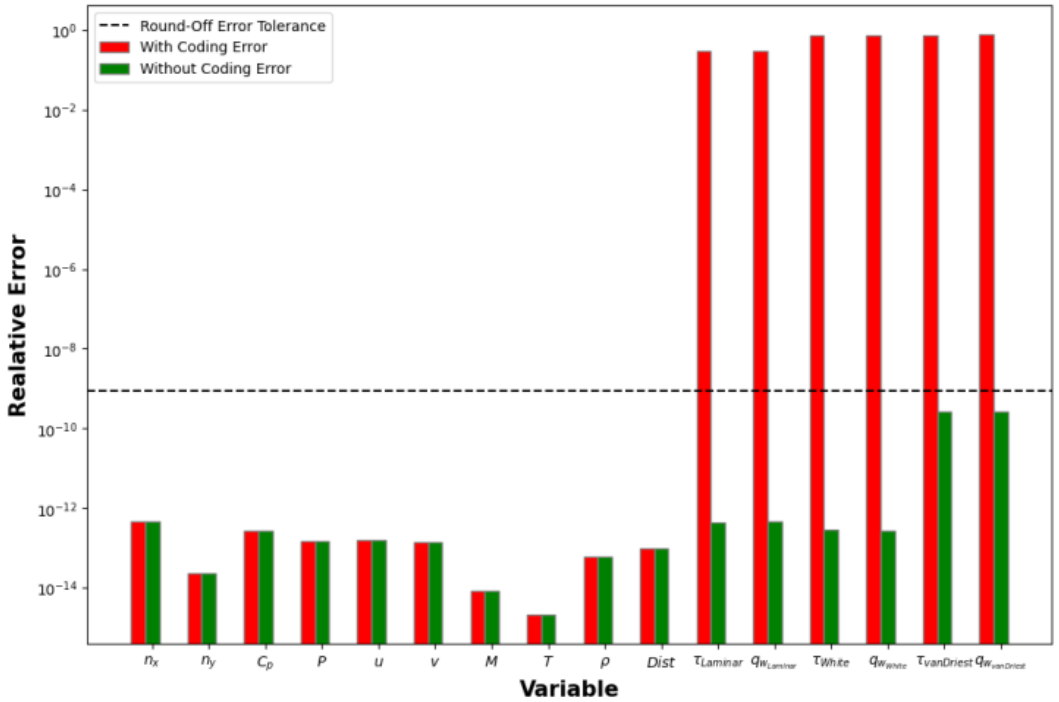
Initial code verification results

- Once the coding errors were fixed, the simulations were reran
- Computed ε_∞ for each QoI
  - Coding error exists if $\varepsilon_\infty > 10^{-10}$
  - Coding error does not exists if $\varepsilon_\infty < 10^{-10}$

- Since all variables have $\varepsilon_\infty < 10^{-10}$, no coding errors exist
- Code verification activities can continue for more complex scenarios



Flat Plate Problem



Inclined Plate Problem

Measuring the impact

- Now that the model is bug free, let's measure the impact on a validation study
- Using the HIFiRE-1 wind tunnel test data, we are able to see the impact of these coding errors on assessing model form error
- This process also highlights the importance of completing code verification before a validation study
- Both a laminar case and a turbulent case results are shown

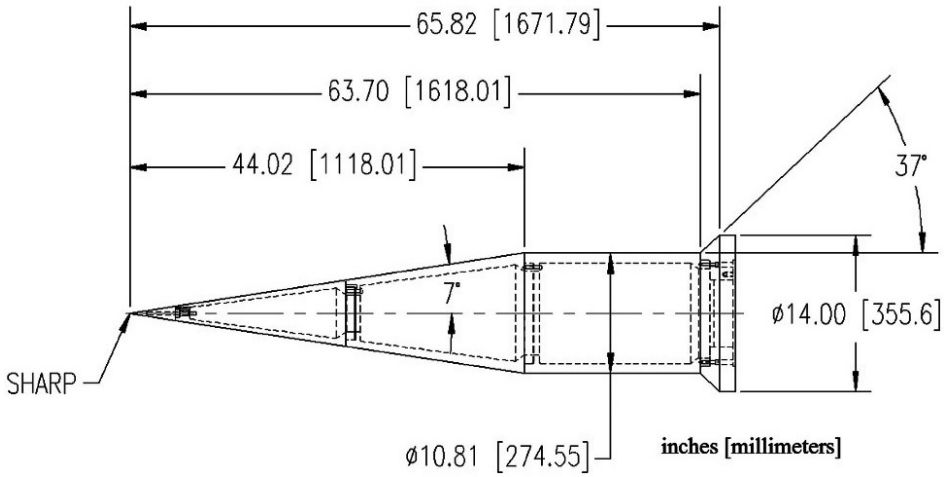
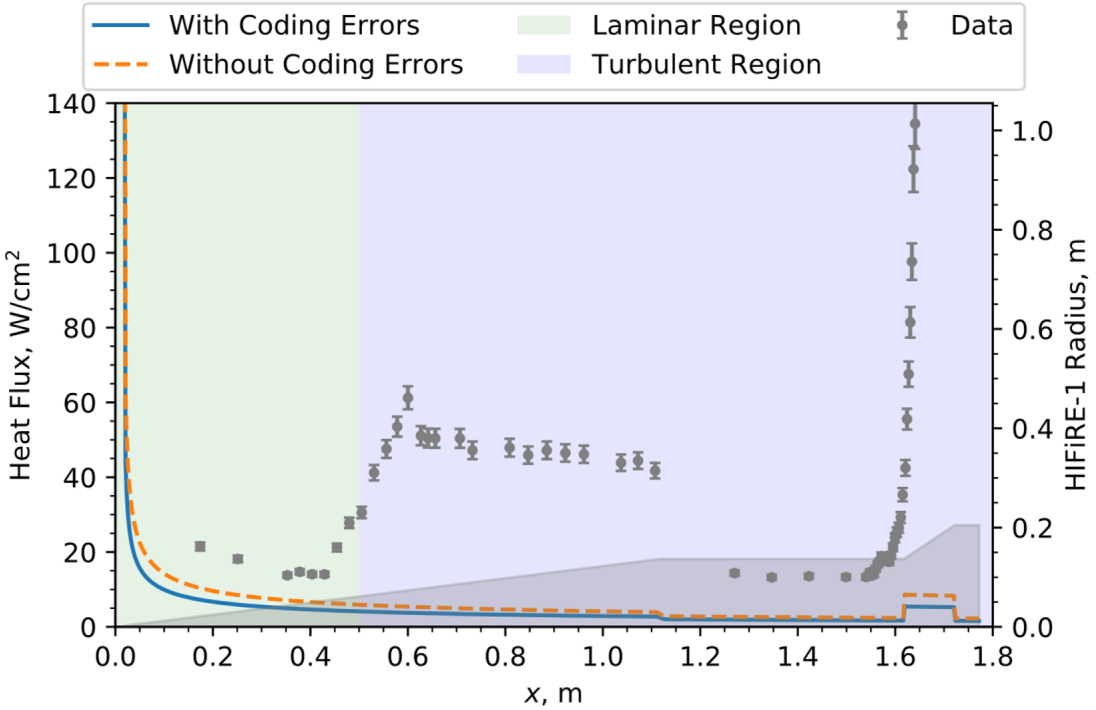


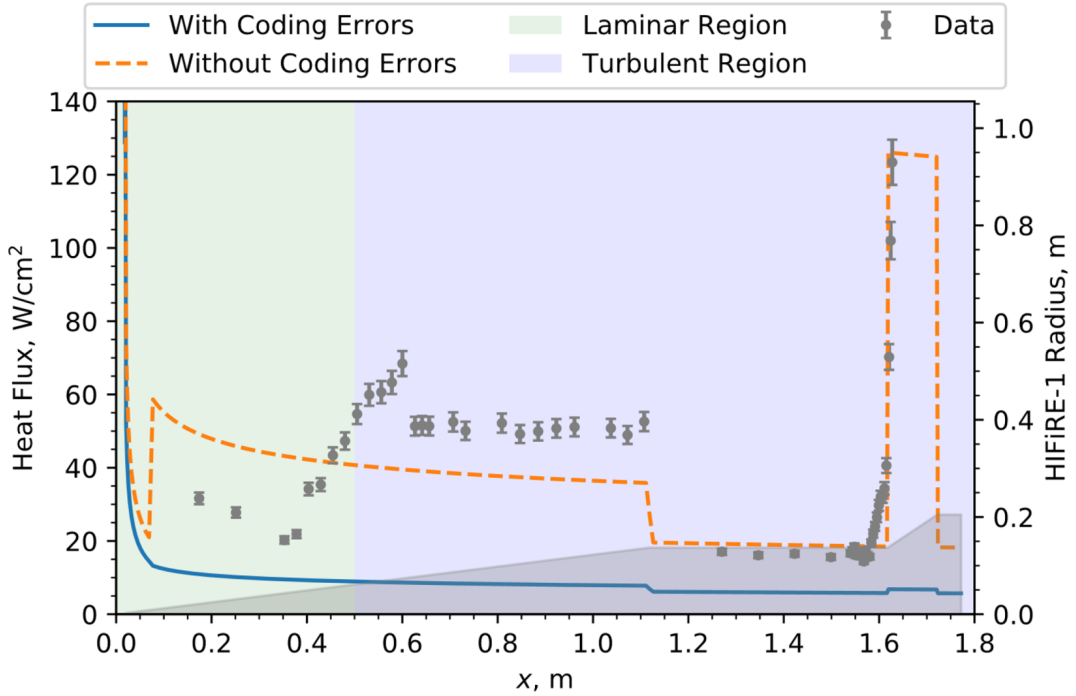
Figure 3: The HIFiRE-1 wind tunnel test geometry that shows the fore-cone on the left, the cylindrical section in the center, and the flare on the right; from Wadhams 2008. The text states that the final nosetip was changed from sharp to a radius of 2.5 mm and the flare angle was changed from 37˚ to 33 ˚.

Impact on results

- For the laminar case, small differences are seen along the streamline direction
- This would have a small impact on validation



- For the turbulent case, large differences are seen along the streamline direction, especially at the tail
- This would have a large impact on validation

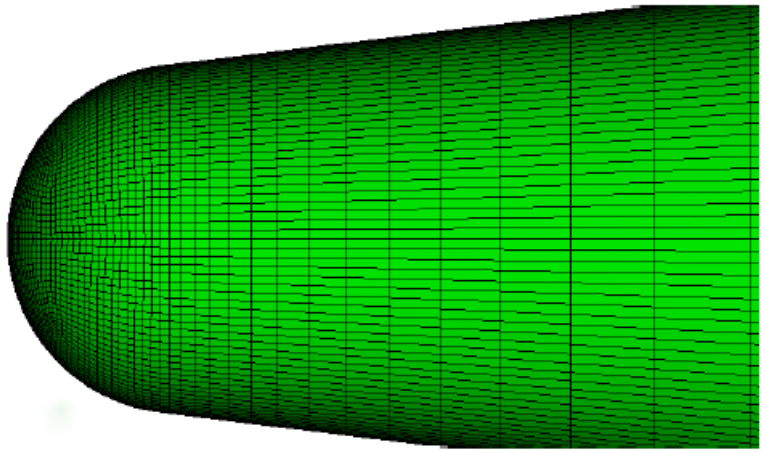


Simulation results generated by
Jared Kirsch of Sandia National Labs

Algebraic models with discretization errors

- For certain problems, numerical errors can be present
  - Geometric discretization
  - Only part of the model is algebraic
  - Iterative errors can still be present
- Break problem up into purely algebraic and discrete if possible
- Initial testing should match analytic solution to round-off for algebraic portion
- Order-of-accuracy testing should cover portions of the code that were not testing in previous testing

- Cone problem introduces geometric discretization
- Additionally, the streamline (Dist) calculation introduces discretization error



Curved Mesh

Conclusions

- We applied code verification methods in a slightly different way
- We apply this methodology to a high-speed compressible flow code
- Three coding errors were identified in the calculation of the coefficient of skin friction
- We showed the impact of the code bug on the HIFiRE-1 wind tunnel test problem
  - This highlights the impact on a validation assessment
- Future work is to continue code verification on the cone problem, which will use order-of-accuracy testing
- When selecting problems, it is valuable to isolate specific errors
  - Start simple and evolve tests to include more possible sources of errors
- Acknowledge that the “verification infrastructure” can be causing issues
  - Add unit testing to cover calculating the analytic solution