17th AIAA Computational Fluid Dynamics Conference
June 6–9, 2005 / Toronto, Ontario
Numerical Methods and Code Optimization II session 44-CFD-10

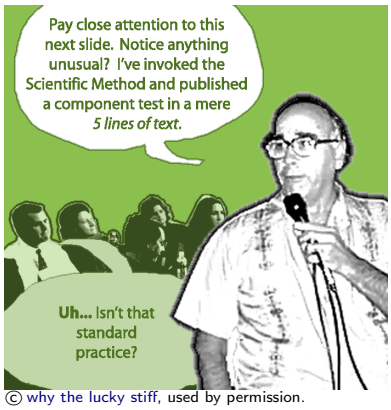
AIAA Paper 2005-4873

Computational Simulations and the Scientific Method

by Bil Kleb and Bill Wood

As scientific simulation software becomes more complicated, the scientific-software implementor's need for component tests from new model developers becomes more crucial. The community's ability to follow the basic premise of the Scientific Method requires independently repeatable experiments, and model innovators are in the best position to create these test fixtures. Scientific software developers also need to quickly judge the value of the new model, i.e., its cost-to-benefit ratio in terms of gains provided by the new model and implementation risks such as cost, time, and quality.

This paper asks two questions. The first is whether other scientific software developers would find published component tests useful, and the second is whether model innovators think publishing test fixtures is a feasible approach.



1 Introduction

This paper is the second installment of the discussion begun by AIAA Paper 2004-2627, "CFD: A Castle in the Sand?", which argues that software unit-testing practices are essential for advancing computational simulation capabilities as models become more complex.[1] That paper called for model and algorithm innovators to publish succinct test fixtures so that subsequent implementors can independently verify they have correctly translated the new innovation to source code, i.e., so the Scientific Method's notion of independently-verifiable experiments can be used. This installment provides an alternative presentation of those ideas in light of copious feedback generated by the first paper.

As growth in computational power facilitates higher-fidelity computational simulation techniques, the number and variety of building-block components also increases. While this increased complexity is forcing a change from the cottage industry of one person/one code to team software development to address increasing software system size,[2] the community is not yet routinely publishing independently verifiable tests for new models or algorithms to address the code-verification complexity. The survey results of table 1 show that only 22% of new models published are accompanied by tests suitable for independently verifying the new model.

To sustain our growing numerical simulation capability, we need to become competent software developers;[3] and one measure of software development competence is sound software testing practices. For example, before inserting a new

[1] Bil Kleb and Bill Wood, *CFD: A Castle in the Sand?*, AIAA Paper 2004-2627, 2004.

[2] L. Cambier and M. Gazaix, *elsA: An Efficient Object-Oriented Solution to CFD Complexity*, AIAA Paper 2002-0108, 2002; N. Kroll, et al., *MEGAFLOW—A Numerical Flow Simulation System*, ICAS 98-2.7.4, 1998; and Natalia Alexandrov, et al., *Team Software Development for Aerothermodynamic and Aerodynamic Analysis and Design*, NASA/TM 2003-212421, 2003.

Table 1: Survey of new component publishing that includes the percent of articles introducing new models that contain tests. Upticks indicate articles with component tests, downticks indicate articles lacking component tests, and dots indicate articles that did not appear to introduce a new model.

journal	vol(#)	articles	%
JCP	192(2)		0
	192(1)		23
	191(2)		27
IJNMF	43(10–11)		0
	43(9)		20
	43(8)		67
			22

[3] James J. Quirk, "Computational Science: Same Old Silence, Same Old Mistakes, Something More is Needed", in *Adaptive Mesh Refinement – Theory and Applications*, edited by Tomaz Plewa, et al., Springer-Verlag, 2004, pp. 1–26.

component into a system, software developers will perform a set of component-level tests. Based on feedback from the first paper,[4] everyone agrees with the need for component-level testing in the computational simulation community but there is disagreement about how to implement it.

While each development group could independently derive component-level tests for each model they choose to implement, this duplication is unnecessary and would not likely catch the special cases that the original innovator would likely know intimately. Besides, the Hatton studies of scientific codes underscores the difficulty in achieving consistent implementations: 1 fault per 170 lines.[5]

This paper calls for institutionalizing component-level testing in the computational simulation community and offers one possible route toward implementation. The paper begins by exploring the current practice, recalls basic tenets of the Scientific Method, proposes a course of action, gives a couple brief examples, and finishes with some concluding remarks.

[4] Bil Kleb and Bill Wood, *CFD: A Castle in the Sand?*, AIAA Paper 2004-2627, 2004.

[5] Les Hatton, "The T Experiments: Errors in Scientific Software", *IEEE Computational Science and Engineering*, 4(2), 1997, pp. 27–38; and Les Hatton and Andy Roberts, "How Accurate is Scientific Software?", *IEEE Transactions on Software Engineering*, 20(10), 1994, pp. 785–797.

2 Current Practice

For the sake of discussion, consider the components of a Computational Fluid Dynamics (CFD) code. While developing such a code, a team will pull components, such as flux functions, boundary conditions, turbulence models, transition models, gas chemistry models, data structures, and so on—each from a different original publication. For example, consider 24 components that comprise the FUN3D flow solver[6] listed in table 2. Now, consider the potential interactions between

[6] fun3d.larc.nasa.gov

Table 2: Components in the FUN3D flow solver. Data provided by Eric Nielsen of NASA.

Turbulence model	Transition model	Boundary conditions	Flux limiter
Flux reconstruction	Time relaxation	Convergence acceleration	Flux functions
Entropy fix	Transport properties	Data structures	Gas chemistry
Time integration	Preconditioners	Flux Jacobians	Governing equations
Multiprocessing	Domain decomposition	Preprocessing	Postprocessing
Grid sequencing	Grid adaptation	Grid movement	Load balancing

these components as indicated by the lines in figure 1. While arguments can be made about whether all components necessarily influence all the other components (as drawn), even the most ardent detractor has to concede that this system is nevertheless a complicated set of interrelated components.

As the number of components increases, the potential interactions grow as $n^2/2$, where n is the number of components. The task of finding an error in a system of interrelated components is daunting, but this task becomes untenable if the components have not been independently verified. Rational

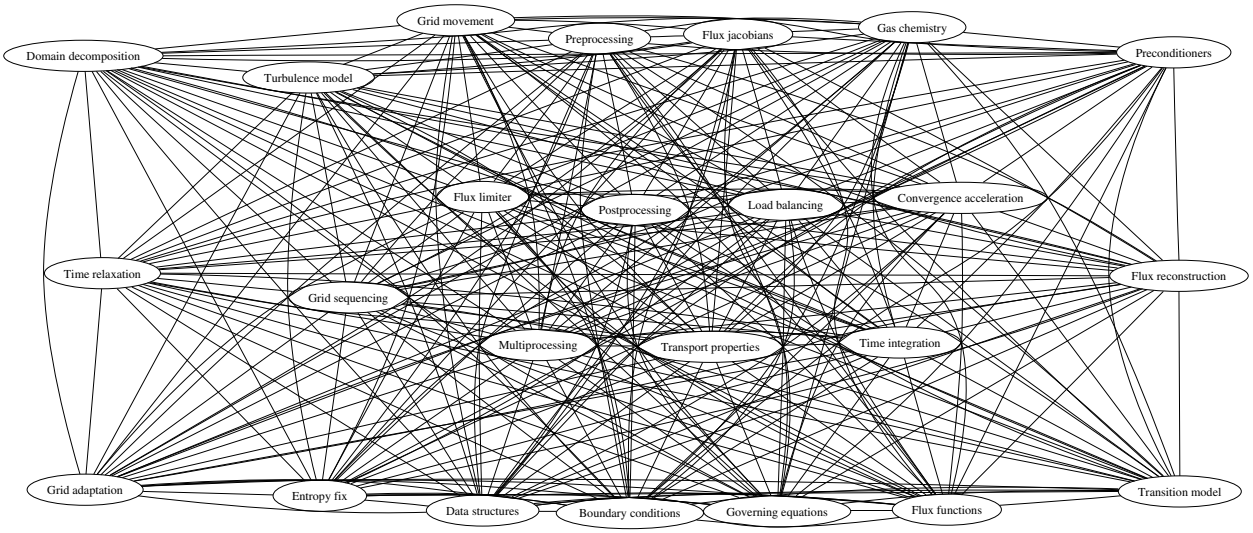


Figure 1: Potential component interactions in the FUN3D flow solver.

verification of this complicated system must proceed in two steps: (1) verification of components and (2) verification of their interactions.

The current computational verification and validation literature recommends verification on the system level by using the Method of Manufactured Solutions (MMS).[7] While this is a necessary step in every code-verification process, it has not yet been widely practiced due to implementation overhead[8] and because if this system-level test fails, the debugging task could be in any of n components *in addition to* the roughly $n^2/2$ component interactions. Therefore, before attempting MMS on a system of components, each component should be independently verified.

Consider for example, the dilemma created by the debut publication of the popular Spalart-Allmaras turbulence model.[9] The document contains a mathematical description of the model and then shows comparisons with experimental boundary layer profiles that require a complete CFD code system. This scenario is sketched in figure 2, in which New Component is the mathematical description of the new turbulence model and the author's code are indicated by Component Code A and System Code A. The boundary layer profile output appears at the bottom.

The dilemma is that no isolated tests of the turbulence model itself, either mathematical or numerical, are presented. So, when another CFD code development team (path B) elects to install this new model in their system, a comparison with boundary layer profiles does not assure the model was implemented in the same way as the original because the other code components are completely different. The specific effects

[7] Christopher J. Roy, "Review of Code and Solution Verification Procedures for Computational Simulation", *Journal of Computational Physics*, 205(1), 2005, pp. 131–156.

[8] MMS typically requires the addition of arbitrary boundary conditions and source functions. In addition, selection of the appropriate basis function remains an art, and so far, only smooth-valued solutions have been manufactured.

[9] P. R. Spalart and S. R. Allmaras, *A One-Equation Turbulence Model for Aerodynamic Flows*, AIAA Paper 92–0439, 1992.

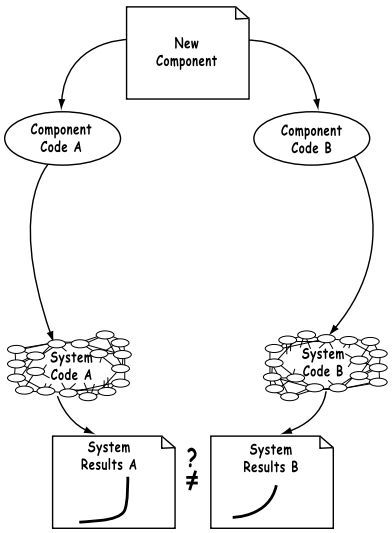


Figure 2: Current method of translating the "paper" model to numerical results.

of the turbulence model become lost in the large computational simulation infrastructure, and there is no credible means to determine that both codes are using precisely the same model. As a consequence of this implementation risk and the lack of test data, the implementor is unable to quickly determine the value of the new model.

3 The Scientific Method

In a computational context, component-based verification testing is the engine behind the Scientific Method that Roger Bacon first described in the thirteenth century: a repeating cycle of *observation*, *hypothesis*, *experimentation*, and the need for *independent verification*.[10]

[10] Roger Bacon, *Opii: Majus, Minus, and Tertium*, c.1267.

Popularized by Francis Bacon and Galileo Galilei, the Scientific Method has since become a means of differentiating science from pseudoscience. The Scientific Method is fueled by the idea that hypotheses and models must be presented to the community *along with* the description of experiments that support the hypothesis. The experiments that accompany a hypothesis must be documented to the extent that others can repeat the experiment—Roger Bacon's independent verification.

This last notion, that others should be able to repeat an

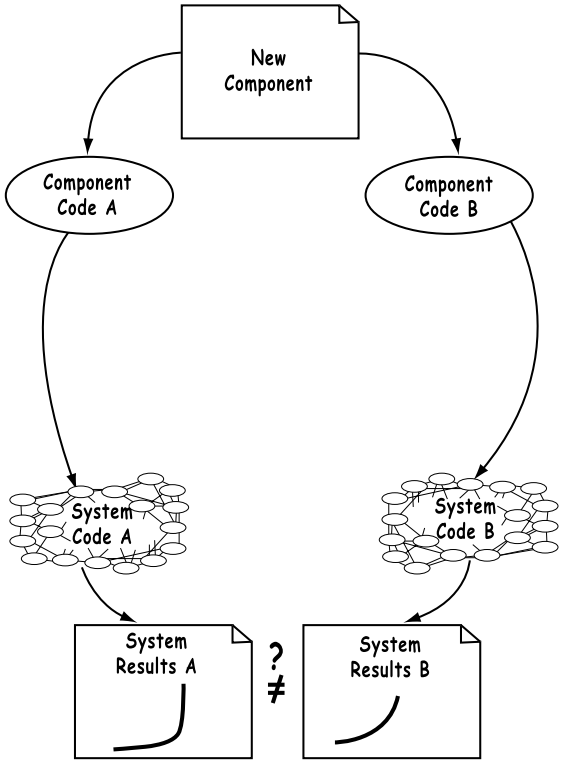


Figure 2: Current method (repeated for convenience).

experiment is currently not widely practiced by computer simulation software community. In part, this is due to the large body of software required by a modern simulation capability. While electronic documentation methods such as Quirk's Amrita system[11] can go a long way toward solving this issue, the fact remains that our experiments must be small enough and isolated enough to be independently repeatable and widely applicable. Ultimately, an implementor should be able to come to the same conclusion as another implementor based solely on the numerical evidence.[12]

[11] www.amrita-cfd.org

[12] Michael Hemsch's restatement of Shewhart's 1st Law of Data Presentation—for the original, see page 58 of W. A. Shewhart, *Economic Control of Quality of Manufactured Product*, D. Van Nostrand Company, 1931.

4 *Proposed Practice*

How can the computational simulation community leverage the Scientific Method?—by having innovators publish a set of tests when they present a new model or algorithm so implementors can gage the innovation's value and reliably make the transition from the mathematics to the numerics. This notion is depicted by the pages labeled Component Verification in figure 3, where model innovators publish component test fixtures so that developer B can verify the numerical implementation of the mathematical model or algorithm in isolation before inserting it into her simulation system. The tests, or numerical experiments, should consist

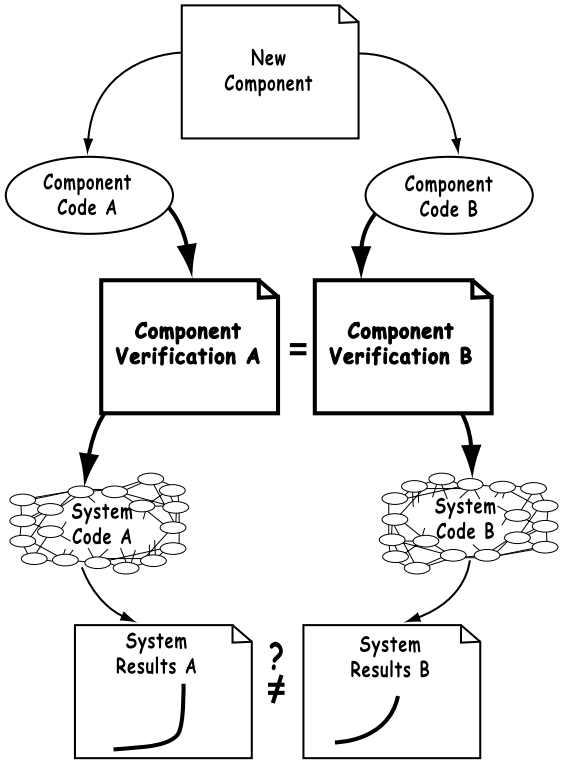


Figure 3: Proposed method of translating the "paper" model to numerical results.

of simple input/output combinations that document the behavior of the model. In particular, boundary cases and any other special cases should be documented. For example, the temperature domain of Sutherland's viscosity law or the nonrealizable initial states for a linearized Riemann solver flux function.

Wherever possible, tests should be written at the mathematical level,[13] but some actual discrete values should also be presented. The latter is particularly advantageous if the experiments are designed to expose boundary areas that are sensitive to divided differences, nonlinear limiters, or truncation error. (Examples are given in section 5.)

All subsequent developers that implement the model and publish their results would be required to document which of the original verification experiments they conducted. Over time, the popular techniques could have a suite of tests formally sanctioned by a governing body such as the AIAA so that any implementation would have to pass the standard tests to be considered verified. Otherwise, the community is left to suffer the fate of unquantified uncertainties as described by Youden's two seminal works.[14]

[13] These tests could also be published in terms of Method of Manufactured Solutions at the component level.

[14] W. J. Youden, "Systematic Errors in Physical Constants", *Physics Today*, 1961, pp. 32–43; and W. J. Youden, "Enduring Values", *Technometrics*, 14(1), 1972, pp. 1–11.

5 Examples

The last paper contains examples for the CIR Scheme,[15] and the Van Albada limiter function. In this paper, two simple components are discussed: the Sutherland viscosity law and the modified Newtonian law.

[15] The one-dimensional version of Roe's Flux Difference Splitting flux function.

Table 3 shows a succinct component test fixture for Sutherland's viscosity law, which gives the viscosity of air as a function of temperature. The mathematical form is presented along with boundary points and a value from the middle of the domain. While this example is trivial, it demonstrates the very localized level at which components should be considered.

Table 3: Sutherland's viscosity law component test fixture.

input	output
T (K)	μ (kg/s-m)
$200 \leq T \leq 3000$	$K^* \frac{T^{1.4}}{T+110.4}$
199	error
200	1.329×10^5
2000	6.179×10^5
3000	7.702×10^5
3001	error

* where $K = 1.458 \times 10^{-6}$.

For example, the flux function example presented in the previous paper was attacked on the grounds that it would be impossible to cover all the discretization settings in which it would be applied, e.g., finite volume, finite difference, or finite element. These considerations are an indication that the component is being defined at too high a level.

Another component examined is the Modified Newtonian law, which gives pressure coefficient as a function of surface inclination according to,

$$C_p = C_{p_{\max}} \sin^2 \theta \tag{1}$$

where θ is the surface inclination angle in degrees, i.e., the angle between the incoming flow and the surface normal vector. The stagnation pressure coefficient is governed by

shock relations,

$$C_{p_{\max}} = \frac{2}{\gamma M_\infty^2} \left\{ \left[\frac{(\gamma+1)^2 M_\infty^2}{4\gamma M_\infty^2 - 2(\gamma-1)} \right]^{\gamma/(\gamma-1)} \left[\frac{1-\gamma+2\gamma M_\infty^2}{\gamma+1} \right] - 1 \right\}$$

where M_∞ is the freestream Mach number and γ is the ratio of specific heats.

A sample component test fixture for this law is shown in table 4. Again, it begins by defining the valid input domains with pure math. Next, certain limiting cases are provided along with a sampling of interior points. Finally, boundary cases are shown and suggested error messages are given.

Other examples of component-based testing are available for an advection-diffusion solver that was written during an exploration of using test-driven development for scientific simulation codes.[16]

Table 4: Modified Newtonian Law component test fixture.

input			output
θ (deg)	M_∞	γ	C_p
$0 \le \theta \le 90$	$5 \le M_\infty$	$1 \le \gamma \le 3$	Eq. 1
0	100	1	2.000
45	100	1	0.500
0	100	1.4	1.839
0	5	1.4	1.809
0	4.9	$\forall$	error
91	$\forall$	$\forall$	error
-91	$\forall$	$\forall$	error

Where $\forall$ indicates "for all valid values".

6 *Concluding Remarks*

To sustain our growing numerical simulation capabilities, we need to become competent software developers by increasing our use of component testing practices. The implementation path offered here is to have model innovators publish simple, component-level verification test fixtures so that implementors can verify their implementation according to the basic premise of the Scientific Method—independently-verifiable experiments.

Based on feedback from the first paper in this series,[17] most readers agree that component-level testing should be standard practice in the computational simulation software development community. However, two questions remain:

> Do scientific software developers want published component tests?
>
> Is the proposed solution palatable by model innovators?

If the answer to either is "no," then how should we proceed?

Unless feedback on this paper dictates otherwise, the next installment of this series will present a more detailed example by using Test-Driven Development,[18] a promising extension of agile programming methodologies.

[16] Bill Wood and Bil Kleb, "Exploring eXtreme Programming for Scientific Research", *IEEE Software*, 20(3), 2003, pp. 30–36

[17] Bil Kleb and Bill Wood, *CFD: A Castle in the Sand?*, AIAA Paper 2004-2627, 2004.

[18] See for example, c2.com/cgi/wiki?TestDrivenDevelopment, last accessed June 1st, 2005.

Acknowledgments

All the folks mentioned in the first paper[19] plus all the folks that responded directly during the Portland conference plus all the independently received comments/suggestions plus the three anonymous journal reviewers for AIAA's *Journal of Aerospace Computing, Information, and Communication*, and David Coppit, Professor of Computer Science at The College of William and Mary.

[19] Bil Kleb and Bill Wood, *CFD: A Castle in the Sand?*, AIAA Paper 2004-2627, 2004.

About the Authors

Bil Kleb, a lifetime senior AIAA member, has worked in the area of computational aerothermodynamics at NASA's Langley Research Center for the past 15 years. Since 1999, Bil has been active in the agile software development community[20] and has given several invited lectures on team software development for scientific software. For the past two years, Bil has been a steward of the FUN3D software development team[21] and has been serving on various agile software development conference committees since 2002.

For those that measure by certificates and degrees, Bil has a B.S. and M.S. in Aeronautical and Astronautical Engineering from Purdue University, an M.B.A. from The College of William and Mary, a Ph.D. of Aerospace Engineering from the University of Michigan, and a commercial pilot certificate with instrument rating.

Email: Bil.Kleb@NASA.Gov

[20] For agile software development's succinct, but surprisingly powerful manifesto, see agilemanifesto.org.

[21] fun3d.larc.nasa.gov

Bill Wood has worked in the field of CFD in the Aerothermodynamics Branch at NASA's Langley Research Center since 1991, earning a Ph.D. in Aerospace Engineering from Virginia Tech in 2001. He served on the program committee for the software development conference XP/Agile Universe from 2002 to 2004 and is now serving on the Agile 2005 conference committee.

Email: Bill.Wood@NASA.Gov

Colophon

This document was typeset in Computer Modern font with the LaTeX typesetting system using the `handout` option of the AIAA package,[22] version 3.8. The `handout` class option used here strives toward the layout style espoused by the visual information design expert Edward Tufte.[23] Also employed were the `color`, `rcsinfo`, `bibentry`, `varioref`, `wrapfig`, `threeparttable`, `booktabs`, `wrapfig`, `hyperref`, and `nohyperref` packages.

[22] www.ctan.org

[23] Edward R. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, 1983; Edward R. Tufte, *Envisioning Information*, Graphics Press, 1990; and Edward R. Tufte, *Visual Explanations: Images and Quantities, Evidence and Narrative*, Graphics Press, 1997.