The Scientific Method and Computational Science: A happy marriage? or in need of therapy?

Bill Rider, Sandia National Labs (SAND-19-2299PE)
“Computers are incredibly fast, accurate, and stupid: humans are incredibly slow, inaccurate and brilliant; together they are powerful beyond imagination.”
— Albert Einstein
The Three Goals for this Talk

Brief introduction to Sandia and National Security Labs

- **Goal 1:** Understand the fundamental tensions of the scientific method and computational science
  - Explore some basic themes in modern computational science, by looking at its origins
- **Goal 2:** Provide some background on the issues associated with the scientific method and the crisis of reproducibility
  - Software and computed results are distinct challenges to standard science
- **Goal 3:** Discuss how verification and validation is actually the way to apply the standard scientific method to computational science
  - Verification is determining that the computer has the right model
  - Validation is comparing the model results to experiment/observation

Computational Science should seamlessly align with the classical Scientific Method.
SNL’s national security mission

- Demands risk-informed decision making; analyzing complex engineering and science phenomena
- Representative high-consequence problem areas:
  - National nuclear security: maintain safe, secure, reliable nuclear stockpile with limited tests; qualify NW (in part) with modeling and simulation
  - Energy: Reduce reliance on foreign energy, reduce energy production carbon footprint energy production
  - Climate change: Understand, mitigate, adapt to effects of global warming
  - Nuclear safety: reactor operations, underground radioactive waste storage: Yucca Mountain, WIPP
  - Security: Cyber, information, infrastructure, homeland
- Limited experimentation and/or data (safety, laws/ethics, practicality, cost/availability)
Exascale Applications Respond to DOE/NNSA Missions in Discovery, Design, and National Security

Scientific Discovery
- Mesoscale materials and chemical sciences
- Improved climate models with reduced uncertainty

Engineering Design
- Nuclear power reactors
- Advanced energy technologies
- Resilient power grid

National Security
- Stockpile stewardship
- Real-time cybersecurity and incident response
- Advanced manufacturing

Grey Bold Text indicates planned or existing exascale application projects
Stockpile Stewardship Challenges

Nuclear Stockpile
- Safety
- Surety
- Reliability
- Robustness

Thermonuclear burn
- \( p, D, T, He^3, He^4 \)
- \( \Delta \tau_{\text{Burn}} \sim 10^{-12} \text{ sec} \)
- \( \Delta \tau_{ee} < 10^{-15} \text{ sec} \)

Atomic Physics

Radiation
- (Photons)

Debye screening

Coulomb Collisions

Quantum interference and diffraction

Hydrodynamics

Non-Proliferation and Nuclear Counter Terrorism

Spontaneous and stimulated emission

Burning Plasma

Weapons Science
“People don’t want to buy a quarter-inch drill. They want a quarter-inch hole.”

— Clayton M. Christensen
The basics of the scientific method

- Ask really good questions about what makes the universe tick.
- **Experiment** or Observe the real world and measure what happens. These measurements are invariably imprecise.
- **Model** and theory the processes in the universe. These models are invariably complex and not generally amenable to exact solution.

The Scientific Method as an Ongoing Process
“It doesn't matter how beautiful your theory is ... If it doesn't agree with experiment, it's wrong.”
— Richard Feynman
All models are wrong, but some are useful.
— George Box
The basics of computational science

- At the center of computational science are **computers**.
- How do we use computers to do science (all the stuff on the previous slides)
- How to use computers more generally for the good of society
- A big part is solving complex models of the universe
- This includes collecting and analyzing data
Computation as a pillar of scientific discovery and engineering design

predictions

- Theory, experiment, and computation partner to:
  - Predict, analyze scenarios
  - Generate ideas, identify gaps
  - Test or suggest theories
  - Assess risk, determine suitability
  - Optimal design, rapid virtual prototyping
  - Explore in untestable regimes

This premise is worth deeper thought and consideration

Reflect: is computation intrinsically different than what came before computers?
“Experiment is the sole source of truth. It alone can teach us something new; it alone can give us certainty.”

— Henri Poincaré
Arguments that Computation and Data are new pillars of science abound

- "Computational thinking" has been proposed as paradigm shift, a fundamentally different approach to science
  - Wolfram among others has chimed in support
  - My colleague Kolda has also thrown support
- Rhett Alain wrote an article in Wired refuting the idea
- Data science is now a proposed fourth pillar of science
- Is science broken? Or in need of revision?
Thomas Kuhn and the structure of scientific revolutions

- The origin of the (now) ubiquitous term “paradigm” shift.
- Discusses fundamental changes in science as the change in conceptual viewpoints.
  - Examples: quantum physics, Galileo
  - Closely related to the concept of disruptive innovation in business
- This contrasted the view that science was a steady march forward with the slow buildup of knowledge over time.
Freeman Dyson has suggested that there are two types of revolutions

- Conceptual – the type Kuhn wrote about
  - Think quantum physics
- Tool-based – based on changes in how we look at the world/universe
  - Think the Hubble space telescope
- Computational science is a little of both
  - Von Neumann conceptualized computational science before any “real” computers existed
  - Computers as tools allow or open new doors – Computational Thinking
  - Fundamentally a computer is a tool for extending human ability
  - Do we need new concepts today or just better tools?
Where are we today in this revolution – conceptual or tool-based?

- ASCI – advanced scientific computing initiative replacing nuclear testing with stockpile stewardship including modeling & simulation.

- Is this a conceptual change? Hmmmm?
  - Are we still invested in engaging with this at a conceptual level? Not so clear, to not so much.
  - We don’t know if it has worked.

- Is it tool-based? Yes
  -Exciting experiments and lots of data
  -The focus on high-end computing is predicated on the belief that the concept is correct.

- Is it really revolutionary? Maybe
What does computational science represent?

Is it a stunning new "third way" to conduct science augmenting theory and experiment?
Is data science a fourth?

or

Is it a stunning new set of tools to augment the standard scientific method?
“A generation which ignores history has no past — and no future.”
— Robert A. Heinlein
Lessons from the beginnings of computational science
What is CFD?
Colorful fluid dynamics
Key points

- The origin of CFD is murky and poorly known or understood.
  - Scientists are terrible historians (especially mathematicians)
  - The history available online is incomplete and/or incorrect (wikipedia)
- The people are essential to how things develop
  - Their personal views and biases are key
- Computational Science was a revolutionary idea
  - CFD is an archetype of computational science
Conceptual approach to computational simulation through physics-based modeling
Wikipedia is a bit dicey

Computational fluid dynamics, usually abbreviated as CFD, is a branch of fluid mechanics that uses numerical methods and algorithms to solve and analyze problems that involve fluid flows... With high-speed supercomputers, better solutions can be achieved. Ongoing research yields software that improves the accuracy and speed of complex simulation scenarios...
Here is its history, not wrong, but certainly not right either..

One of the earliest type of calculations resembling modern CFD are those by Lewis Fry Richardson, in the sense that these calculations used finite differences and divided the physical space in cells. Although they failed dramatically, these calculations, together with Richardson's book "Weather prediction by numerical process",[2] set the basis for modern CFD and numerical meteorology. In fact, early CFD calculations during the 1940s using ENIAC used methods close to those in Richardson's 1922 book.[3]

This account misses almost everything that should be here!

The computer power available paced development of three-dimensional methods. Probably the first work using computers to model fluid flow, as governed by the Navier-Stokes equations, was performed at Los Alamos National Lab, in the T3 group.[4][5] This group was led by Francis H. Harlow, who is widely considered as one of the pioneers of CFD. From 1957 to late 1960s, this group developed a variety of numerical methods to simulate transient two-dimensional fluid flows, such as Particle-in-cell method (Harlow, 1957),[6] Fluid-in-cell method (Gentry, Martin and Daly, 1966),[7] Vorticity stream function method (Jake Fromm, 1963),[8] and Marker-and-cell method (Harlow and Welch, 1965).[9] Fromm's vorticity-stream-function method for 2D, transient, incompressible flow was the first treatment of strongly contorting incompressible flows in the world.

The next part of the history on panel methods and aero engineering is closer to the mark, but I know much less about that.
A Presentation by Bram Van Leer in 2010 and part of my inspiration for the talk.

HISTORY OF CFD: PART II

Top level: Jay Boris, Vladimir Kolgan, Bram van Leer, Antony Jameson
Ground level: Richard Courant, Kurt Friedrichs, Hans Lewy, Robert MacCormack, Phillip Roe, John von Neumann, Stanley Osher, Amiram Harten, Peter Lax, Sergei Godunov
CFD was developed by many great minds

John Von Neumann

Robert Richtmyer

Peter Lax

Lord Rayleigh & G. I. Taylor

Courant, Friedrichs, Lewy – 1928 paper

Bethe and Feynman – the first calculations using Von Neumann’s method at Los Alamos in 1944

Teller, Metropolis, Ulam – Monte Carlo Methods and the H-Bomb

Harlow – the name CFD and Los Alamos often conjures

Landshoff & Rosenbluth

Lord Rayleigh & G. I. Taylor

Courant, Friedrichs, Lewy – 1928 paper

Bethe and Feynman – the first calculations using Von Neumann’s method at Los Alamos in 1944

Teller, Metropolis, Ulam – Monte Carlo Methods and the H-Bomb

Harlow – the name CFD and Los Alamos often conjures

Landshoff & Rosenbluth
The first “CFD” calculations

- The first hydrodynamic calculation was described in a Los Alamos report (LA-94) on June 20, 1944 – lead author Hans Bethe
  - Feynmann was the calculational lead and marked the transition from human computers to IBM machines (done in April/May ‘44).
  - They used two methods to compute shocks, but only one of them worked well (the shock fitting by Peierls). The other finite difference method produced severe post-shock “wiggles” explained as thermal excitation.

- The first calculations were 1-D and Lagrangian, shocks were tracked (no viscosity, finite differences failed completely till 1948).

- Von Neumann developed a “simple” finite difference method at Aberdeen and published his report on March 20, 1944.
A Method for the Numerical Calculation of Hydrodynamic Shocks

J. Von Neumann and R. D. Richtmyer
Institute for Advanced Study, Princeton, New Jersey
(Received September 26, 1949)

The equations of hydrodynamics are modified by the inclusion of additional terms which greatly simplify the procedures needed for stepwise numerical solution of the equations in problems involving shocks. The quantitative influence of these terms can be made as small as one wishes by choice of a sufficiently fine mesh for the numerical integrations. A set of difference equations suitable for the numerical work is given, and the condition that must be satisfied to insure their stability is derived.

I. INTRODUCTION

In the investigation of phenomena arising in the flow of a compressible fluid, it is frequently desirable to solve the equations of fluid motion by stepwise numerical procedures, but the work is usually severely complicated by the presence of shocks. The shocks manifest themselves mathematically as surfaces on which density, fluid velocity, temperature, entropy and the like have discontinuities; and clearly the partial differential equations governing the motion require boundary conditions connecting the values of these quantities on the two sides of each such surface. The necessary boundary (but preferably somewhat larger than) the spacing of the points of the network. Then the differential equations (more accurately, the corresponding difference equations) may be used for the entire calculation, just as though there were no shocks at all. In the numerical results obtained, the shocks are immediately evident as near-discontinuities that move through the fluid with very nearly the correct speed and across which pressure, temperature, etc. have very nearly the correct jumps.

It will be seen that for the assumed form of dissipation (and, indeed, for many others as well), the Rankine-Hugoniot equations are satisfied, provided the thick-
LA-671, The first description of artificial viscosity written by Richtmyer (only!)

Classified till 8/26/93. In the period right after WWII all Lab reports were intrinsically treated as classified.

The projects Richtmyer was working on in 1947 and 1948 were key to the development of the method. The application was too complex for shock fitting.
Richtmyer published a second report five months later in 1948 (March to August) reporting on numerical experiments.

\[
\frac{\partial u}{\partial t} + \frac{\partial}{\partial m}(p+q) = 0 \rightarrow \frac{\partial u}{\partial t} + \frac{\partial}{\partial m}\left(p + \mu \frac{\partial u}{\partial x}\right) = 0
\]

\[
T \Delta S = -\frac{1}{6} G \frac{1}{c^2} \left(\frac{\Delta V}{V}\right)^3 \rightarrow
\]

\[
T \Delta S = \mu \left(\frac{\partial u}{\partial x}\right)^2 \rightarrow \mu \propto (\Delta x)^2 \left|\frac{\partial u}{\partial x}\right|
\]

He uses both the term “fictitious” and “mock” to describe the term, but not “artificial”. All of these are unfortunate in their connotation.
The beginning of weather/climate/turbulence modeling is connected to all of this too, through Von Neumann.

**Staggered Grid**

\[
\begin{align*}
  k & \quad \phi, \psi, u, v \\
  k + \frac{1}{2} & \quad \rho, \omega \\
  \delta t & = 30 \text{ minutes}
\end{align*}
\]

First calculation
16x16x(3) mesh
\[\Delta x = 300 \text{ km}\]
48 time steps
\[\Delta t = 30 \text{ minutes}\]
Lax’s contributions have received a great honor - the 2005 Abel Prize

- Some of the work he was honored for started at Los Alamos and continued while at NYU’s Courant Institute.
  - The work on conservation laws begins in the wake of knowing shock capturing is a workable concept via Von Neumann-Richtmyer’s viscosity.
  - Lax’s efforts form much of the theoretical foundation for CFD today.
  - Basic theory for the analytical and numerical solution of hyperbolic conservation laws.
Computational Science has been powered by technology advances for decades...

The LLNL Plot

CFD becomes possible here

Follows Moore’s Law (approx.)
“I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.” — Abraham Maslow
“In the twilight of Moore’s Law, the transitions to multicore processors, GPU computing, and HaaS cloud computing are not separate trends, but aspects of a single trend – mainstream computers from desktops to ‘smartphones’ are being permanently transformed into heterogeneous supercomputer clusters. Henceforth, a single compute-intensive application will need to harness different kinds of cores, in immense numbers, to get its job done.

The free lunch is over. Now welcome to the hardware jungle.” — Herb Sutter 2011
Approximately a Cray 2 via linpack
“Any sufficiently advanced technology is indistinguishable from magic.”
– Arthur C. Clarke
"In the field of numerical algorithms, however, the improvement can be quantified. Here is just one example, provided by Professor Martin Grötschel of Konrad-Zuse-Zentrum für Informationstechnik Berlin. Grötschel, an expert in optimization, observes that a benchmark production planning model solved using linear programming would have taken 82 years to solve in 1988, using the computers and the linear programming algorithms of the day. Fifteen years later – in 2003 – this same model could be solved in roughly 1 minute, an improvement by a factor of roughly 43 million. Of this, a factor of roughly 1,000 was due to increased processor speed, whereas a factor of roughly 43,000 was due to improvements in algorithms! Grötschel also cites an algorithmic improvement of roughly 30,000 for mixed integer programming between 1991 and 2008."
Does Moore's Law Suddenly Matter Less?
feld.com | Mar 8th 2011

A post in the New York Times this morning asserted that Software Progress Beats Moore’s Law. It’s a short post, but the money quote is from Ed Lazowska at the University of Washington:

“The rate of change in hardware captured by Moore’s Law, experts agree, is an extraordinary achievement. “But the ingenuity that computer scientists have put into algorithms have yielded performance improvements that make even the exponential gains of Moore’s Law look trivial,” said Edward Lazowska, a professor at the University of Washington.

The rapid pace of software progress, Mr. Lazowska added, is harder to measure in algorithms performing nonnumerical tasks. But he points to the progress of recent years in artificial intelligence fields like language understanding, speech recognition and computer vision as evidence that the story of the algorithm’s ascent holds true well beyond more easily quantified benchmark tests.”
From the DoE Scales Report, 2004 (Shadid & Plimpton)

Algorithmic Speed-up: PDE Problem

- Dense LU: $O(N^3)$
- Banded LU: $O(N^2)$
- Banded PCG: $O(N^{3/2})$
- Multilevel PCG: $O(N)$

$N_{\text{linpack}}$: $1.5 \times 10^3$ to $1 \times 10^6$

Year

Comparing performance improvements between hardware and algorithms.

From the DoE Scales Report, 2004 (1994)

The jumps in performance are actually more discrete… “quantum”

We are overdue for a breakthrough, but what will it be? sublinear? A nonlinear method for a linear problem, or maybe multigrid is it?
“The fundamental law of computer science: As machines become more powerful, the efficiency of algorithms grows more important, not less.”
– Nick Trefethen
Welcome to the project!
Here's the codebase.
Existing technology often defines quality and correctness. ASC codes are good examples.

It is essential to understand quality from this perspective if progress is to be made.

A legacy code’s solutions and associated practices are the starting definition of “good.”
“... There is increasing concern that in modern research, false findings may be the majority or even the vast majority of published research claims ... However, this should not be surprising. It can be proven that most claimed research findings are false...”

By Tim Trucano (SNL, ret.)

How am I supposed to reproduce the computational work?
How was this refereed?
(Ignition on NIF isn’t going to happen.)

By Tim Trucano (SNL, ret.)

“It’s increasingly recognized that computational science is facing a credibility crisis: it’s impossible to verify most of the computational results that are presented at conferences and in papers today ...”

By Tim Trucano (SNL, ret.)

“Reproducibility is central to the progress of science, and simulation-based research is no exception.”

By Tim Trucano (SNL, ret.)

The Opportunity – TOMS Replication

“... We hope that the general concern for advancing the quality of computational science results will be incentive enough for authors to assent to the replicated computational results process...”

By Tim Trucano (SNL, ret.)

Reproducible and replicable CFD: it’s harder than you think

Completing a full replication study of our previously published findings on bluff-body aerodynamics was harder than we thought. Despite the fact that we have good reproducible-research practices, sharing our code and data openly. Here’s what we learned from three years, four CFD codes and hundreds of runs.

Olivier Mesnard, Lorena A. Barba
Mechanical and Aerospace Engineering, George Washington University, Washington DC 20052

Our research group prides itself for having adopted Reproducible Research practices. Barba made a public pledge titled “Reproducibility PI Manifesto” (PI: Principal Investigator), which at the core is a promise to make all research materials and methods open access and discoverable: releasing code, data and analysis/visualization scripts.

In 2014, we published a study on Physics of Fluids titled “Lift and wakes of flying snakes.” It is a study that uses our in-house code for solving the equations of fluid motion in two dimensions (2D), with a solution approach called the “immersed boundary method.” The key of such a method for solving the equations is that it exchanges complexity in the mesh generation step for complexity in the application of boundary conditions. It makes possible to use a simple discretization mesh (structured Cartesian), but at the cost of an elaborate process that interpolates values of fluid velocity at the boundary points to ensure the no-slip boundary condition (that fluid sticks to a wall). The main finding of our study on wakes of flying snakes was that the 2D section with anatomically correct geometry for the snake’s body experiences lift enhancement at a given angle of attack. A previous experimental study had already shown that the lift coefficient of a snake cross section in a wind tunnel gets an extra oomph of lift at 35 degrees angle-of-attack. Our simulations showed the same feature in the plot of lift coefficient. Many detailed observations of the wake (visualized from the fluid-flow solution in terms of the vorticity field in space and time) allowed us to give an explanation of the mechanism providing extra lift.

When a computational research group produces this kind of study with an in-house code, it can take one, two or even three years to write a full research software from scratch, and complete verification and validation. Often, one gets the question: why not use a commercial CFD package? Why not use another research group’s open-source code? Beyond reasons that have to do with inventing new methods, it’s a good question. To explore using an existing CFD solver for future research, we decided to first complete a full replication of our previous results with these alternatives. Our commitment to open-source software for research is unwavering, which rules out commercial packages. Perhaps the most well known open-source fluid-flow software is OpenFOAM, so we set out to replicate our published results with this code. A more specialist open-source code is IBAMR, a project born at New York University that has continued development for a decade. And finally, our own group developed a new code, implementing the same solution method we had before, but providing parallel computing via the renowned PETSc library. We embarked on a full replication study of our previous work, using three new fluid-flow codes.

This is the story of what happened next: three years of dedicated work that encountered a dozen ways that things can go wrong, conquered one after another, to arrive finally at (approximately) the same findings and a whole new understanding of what it means to do ‘reproducible research’ in computational fluid dynamics.”

As examples, I’ll focus on one of my own papers.

- This paper was written to report algorithmic progress.
- Testing, i.e., verification became important although for different reasons.
- The volume tracking paper is highly cited because of the tests it introduced.
- The testing in other papers became a bit of a tug-of-war with the editor and reviewers.
- Both issues point to the process to determine quality of calculations.
- Releasing code was achieved in one case, but has become increasingly problematic to virtually unthinkable.
- The environment at the Lab is becoming less favorable towards (full) openness although it varies with the source of your support.
Why did we write “Reconstructing Volume Tracking”?

- We wrote the paper because the standard way of coding up a volume of fluid method was so hard to debug.
- We thought we had a better way to put the method together using computational geometry (i.e., a “toolbox”)
- Once the method was coded it needed to be tested:
  - In addition, existing methods for testing these methods were “pretty lame.”
  - We came up with some new tests borrowed from the high-resolution methods community (combining the work of several researchers
    - Dukowicz’s vortex,
    - Smolarkiewicz’s deformation field and
    - Leveque’s time reversal)
The paper’s origin actually had a lot to do with how these methods were programmed.

Horrible computer code in F77 redacted due to security and legal concerns of my current and former employers.

Notes:
1. The code has high cyclomatic complexity
2. The code is not extensible
3. The code is almost impossible to debug (see #1)
The logic goes on...

Continued redaction...

by the way there are two columns of 9 point Courier text, so it is a lot of code.
The logic goes on...

More continued redaction of code.
“What I cannot create, I do not understand.”

–Richard Feynman
Using Computational Geometry to Construct a VOF or Volume Tracking Method

An intersection is forced on this line

\[ A = \frac{1}{2} \sum_{v=1}^{n} (x_v y_{v+1} - x_{v+1} y_v) \]

\[ A = \frac{\pi}{6} \sum_{v=1}^{n} (r_v + r_{v+1})(r_v z_{v+1} - r_{v+1} z_v) \]
We presented a serious rethink of the programming approach to these methods.

“Beautiful” F77 computer code redacted due to security and legal concerns of my current and former employers.

Notes:
1. The code has low cyclomatic complexity
2. The code is extensible
3. The code is simple to debug (see #1)
We even included the code... with serious restrictions imposed by LANL

```fortran
Subroutine INTERSECT (a1, rho1, a2, rho2, xi, yi, notparallel)
Implicit None
Include "param.h"
Logical notparallel
Real a1(1:2)
Real a2(1:2)
Real rho1
Real rho2
Real xi
Real yi
Real smdet ! small number for parallel line detection
Real det ! determinant of the linear system
smdet = Max (eps, smallvof * Abs(a1(1) * a2(2)),
               & smallvof * Abs(a2(1) * a1(2)))
c.... first compute the determinant of the linear system
det = a1(1) * a2(2) - a2(1) * a1(2)
c.... if the determinant is approximately zero, the linear system is not solvable and we have parallel (approximately) lines.
If (Abs(det) .gt. smdet) Then
c...... nominal (nonparallel) case
   xi = (rho1 * a2(2) - rho2 * a1(2)) / det
   yi = (rho2 * a1(1) - rho1 * a2(1)) / det
   notparallel = .true.
Else
c...... set the flag to show that parallel lines have been found
   notparallel = .false.
End If
Return
End
```

As a condition of making the code available, I had to strip out most of the comments and formatting. This is just computational geometry!

This is just 1996, not the post-2001 World either!

I fought making the code this ugly to no avail.
The code that took three viewgraphs to express can be shown on one slide

“Beautiful” F77 computer code redacted due to security and legal concerns of my current and former employers.

Notes:
0. The code doesn’t take up the whole slide either
1. The code has low cyclomatic complexity
2. The code is extensible
3. The code is simple to debug (see #1)
Why did this paper get cited so much?

Test Problems

Velocity Field

\[ u = -\frac{\partial \Psi}{\partial y}, \quad v = \frac{\partial \Psi}{\partial x} \]

\[ \Psi = \frac{1}{\pi} \sin^2(\pi x) \cos^2(\pi y) \]

\[ \Psi = \frac{1}{4\pi} \sin(4\pi(x + \frac{1}{2})) \]

\[ \times \cos(4\pi(y + \frac{1}{2})) \times \cos(\pi t/T) \]

Zalesak's disc

Too Easy!

For Debugging

J. Dukowicz produced the earliest example I found.

From P. Smolarkiewicz

From R. Leveque

Solid Body Rotation
Single Vortex: Front Tracking Solutions

32x32 grid

128x128 grid

solutions by Damir Juric
“What’s measured improves”
– Peter Drucker
We need to connect modeling & simulation with experimental design

- Science is about **understanding** and **explanation** – prediction is a quest to assist these ends
- **Validation** depends on **experiment** and measurement.
- The conduct of **experiments & computations should be conducted together** and the importance should be properly identified and focused upon – prediction & discovery.
- The assessment of modeling quality needs to consider the quality of the measurement.
  - **Bad measurements mean poor constraints for modeling.**
  - **Bad modeling should be identified by good experiments**

A conceptual picture of V&V within the context of science

Code = Theory
Simulation = Analysis
Verification and validation are essential to the quality of simulations.

- **Verification** \(\approx\) Solving the equations **correctly**
  - Mathematics/Computer Science issue
  - Applies to both codes and calculations
- **Validation** \(\approx\) Solving the **correct equations**
  - Physics/Engineering (i.e., modeling) issue
  - Applies to both codes and calculations
- **Calibration** \(\approx\) Adjusting ("tuning") parameters
  - Parameters chosen for a specific class of problems
- **Benchmarking** \(\approx\) Comparing with other codes
  - "There is no democracy in physics."

There is a simple connection!

- Verification and Validation are the **structured application of the scientific method** to computational science. It is a means of synthesis!

- Verification is determining that an intended model is being computed properly *(theory is computed right)*

- Validation is the structured comparison of experiments or observations with the computed model results *(computed results are reflecting reality)*

- Together these thread together computational work with the classical scientific method.
This shows how V&V is viewed by Modeling and Simulation “customers”

Using the “FORCE” of simulation, I now understand the universe!

Witness the power of a fully armed and operational V&V program to call your understanding into doubt!
“V&V takes the fun out of computational simulation”
– Tim Trucano
Experimental results must have error bounds.

Measurements without error bounds are (virtually) meaningless. **Corollary: calculations without error are too!**

"If there are no error bars, assume they’re given by the limits of the plot."

The Default Uncertainty is Always ZERO

- Actual UQ is more than what we call “UQ.”
- Uncertainty is “doubt”
- We have model form (users), numerical, model parameters, experimental uncertainty
- In some cases we don’t know what these are for a variety of reasons (e.g., a single experiment and hence no variability)
- The accepted habit is that an unknown uncertainty is assigned the very smallest value possible! ZERO
- This is critically damaging to the conduct of science
- Uncertainties must be estimated or bounded – especially the irreducible ones.
ONE DOES NOT SIMPLY

HAVE ZERO UNCERTAINTY
The default uncertainty is always ZERO!

- One of the key things to recognize is the community wide practice of not assessing key uncertainties in modeling and simulation and the implicit assessment of that uncertainty as exactly zero.
- This practice is widespread and pernicious.
- As a result doing any work increases uncertainty instead of decreasing it.
- This is a massive barrier to progress.
- If someone asserts a zero uncertainty, the truth is they don’t know what it is, or afraid to be truthful.
What’s the bottom line?

- Computational science & computers are a stunning new set of tools to augment the standard scientific method. **The scientific method is fine as is.**
- Computers (of all sizes), programs, algorithms, methods, data, communication, analysis are all indispensable tools to conduct scientific investigation. **Computational thinking is key.**
- The origins of computational science is intertwined with solving complex models for applied scientific purposes. **History is key.**
- Improvements in the tools are focused on big iron although algorithms have shown greater payoff.
- **Science should be highly reproducible** and the complexity and transience of computational tools makes this a huge challenge.
- **V&V is the scientific method made operational for modeling & simulation work.**
“The scientific method’s central motivation is the ubiquity of error - the awareness that mistakes and self-delusion can creep in absolutely anywhere and that the scientist’s effort is primarily expended in recognizing and rooting out error.” David Donoho et al. (2009)
“An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.”

– David Donoho