Reliability of Modern-Day High-Performance Scientific Computing

(How) Can I Trust It?

Pi-Yueh Chuang
pychuang@vt.edu
Virginia Tech, VA

June 11th, 2024

Disclaimer

- No rocket science
- No equations
- No machine learning/AI
- No take-home messages but take-home questions
- **Mostly subjective opinions**

We know calculation correctness is important to any science relying on computing, but we're still sloppy on it

I know physical activity is critical to our health, but I still don't work out ...

Why?

About Me

Aerospace and mechanical engineering… BS, MS, PhD

- Had no computer-science degrees … now @CS
- Failed my freshman-level physics twice … now @INT
- Interests:
  - Numerical Methods / Numerical Partial Differential Equations
  - Computer-Aided Engineering
  - Okay-Performance Scientific Computing


I have plenty of confidence in my students. Knowing them, I for a fact can assure you this plane will never even start.

My Very 1st Lesson in Numerical Analysis/Methods

Professor: hope your code won't be the culprit of any accidents

YEAR	ACCIDENT	DEATHS	LOSS	CAUSE
1962	Mariner 1 Self-Destruction	0	>> $18 M	Math Implementation
1985	Therac-25 Radiation Overdose	> 3	N/A	Race Conditions
1991	Patriot Missile Failure	28	$1 B	Round-Off Error Accumulation
1991	Sleipner A Oil Platform Sinking	0	$700M	Inaccurate Math Approximation
1996	Ariane 5 Rocket Explosion	0	>> $370 M	Integer Overflow
2015	Airbus A400M Crash	4	>> $165 M	Software Deployment
2018	Uber Self-Driving Car Incident	1	N/A	Model Limitations

Luckily, Academia Is So Friendly

The worst scenario is paper retraction.

From Retraction Watch Database:

Reasons
Error in Analysis
Error in Methods
Results Not Reproducible

+

Journal Keywords
Numerical
Computational
Simulations
* Learning
Artificial Intelligence
Neural Networks
Analysis

=

31 Retractions Since 2014

Note: Retraction is Good.

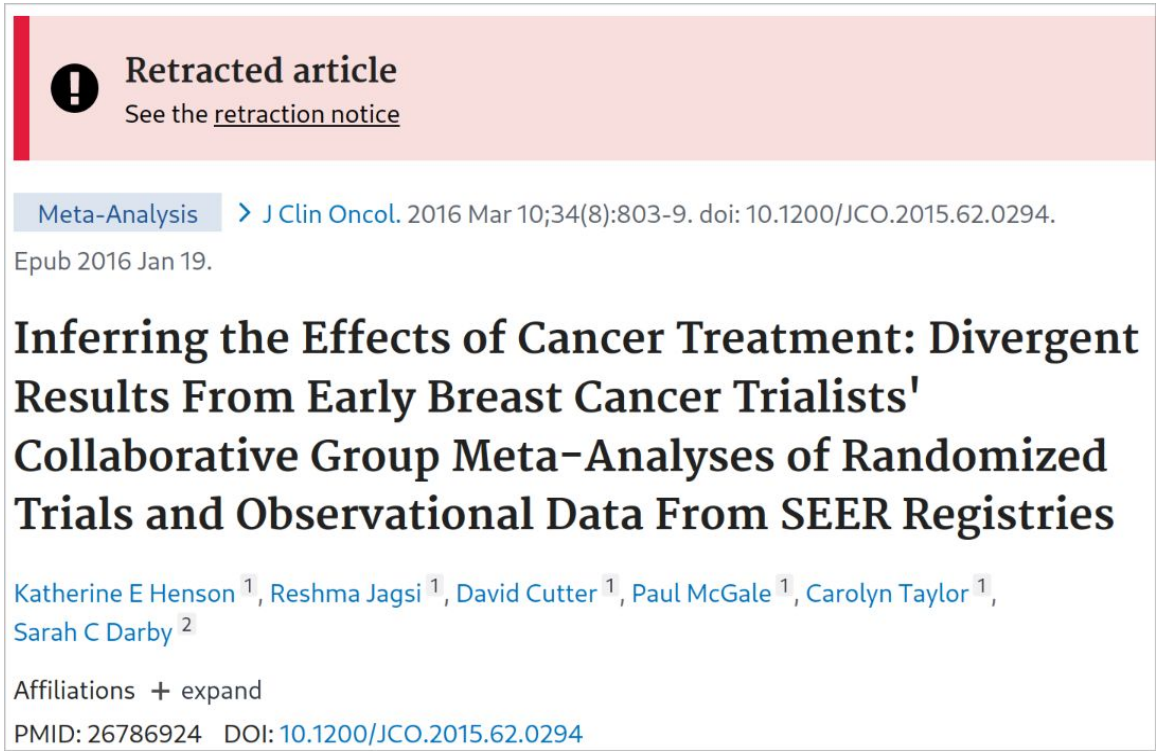
Retraction: Phase transitions, percolation, fracture of materials, and deep learning [Phys. Rev. E **102**, 011001(R) (2020)]

Serveh Kamrava, Pejman Tahmasebi, Muhammad Sahimi, and Sepehr Arbabi
Phys. Rev. E **104**, 049901 – Published 5 October 2021

> *We retract the paper because to compute part of the results additional information had been used, which, due to an oversight, was not mentioned in the paper. The computational algorithm contained an inner loop that imposed a constraint on the predictions of the algorithm at each step so as to prevent the trends in the predictions from deviating significantly from the predictions in the prior steps.*

Note: Retraction is Good.

Error in one line of code… made by a doctoral student



Retracted article
See the retraction notice

Meta-Analysis > J Clin Oncol. 2016 Mar 10;34(8):803-9. doi: 10.1200/JCO.2015.62.0294. Epub 2016 Jan 19.

Inferring the Effects of Cancer Treatment: Divergent Results From Early Breast Cancer Trialists' Collaborative Group Meta-Analyses of Randomized Trials and Observational Data From SEER Registries

Katherine E Henson [1], Reshma Jagsi [1], David Cutter [1], Paul McGale [1], Carolyn Taylor [1], Sarah C Darby [2]

Affiliations + expand

PMID: 26786924 DOI: 10.1200/JCO.2015.62.0294

Note: Retraction is Good.

Side question: does it make sense to blame "students"?

"All software has bugs." ~~ True

"It's usually not a huge deal as long as you're honest about it and you do your best to correct it." ~~ True

Can we trust any computing result in academia?

Can we still trust the 1st image of a black hole?

"Good unit tests resolve the issue." ~~ **What?????? Not really…**

YEAR	INCIDENT	Prevent by UT	REMARKS
1962	Mariner 1 Self-Destruction	**Unlikely**	Misunderstanding of math symbols
1985	Therac-25 Radiation Overdose	**Maybe**	Skipped safety checkers due to race condition
1991	Patriot Missile Failure	**Unlikely**	Round-off error accumulations after 100+ hours
1991	Sleipner A Platform Sinking	**Unlikely**	Inaccurate math approximation
1996	Ariane 5 Rocket Explosion	**Maybe**	Overflows 64bit floats → 16bit integers
2015	Airbus A400M Crash	**Unlikely**	External info being deleted during deployment
2018	Uber Self-Driving Car Incident	**Unlikely**	Limitations in the prediction model itself

Basic Quality Assurance for Scientific Computing

Stage	Defined By	Core Concept
Code Verification	AIAA/ASME	Components working correctly up to specifications
Solution Verification	AIAA/ASME	Software solving mathematics correctly
Validation	AIAA/ASME	Mathematics modeling physics correctly
Reproduction	NASEM	Same results using the same inputs/methods/code
Replication	NASEM	Same answer to the same scientific question using different approaches/data/code/methods.

† AIAA: American Institute of Aeronautics and Astronautics
† ASME: American Society of Mechanical Engineers
† NASEM: National Academies of Sciences, Engineering, and Medicine

A silly example:

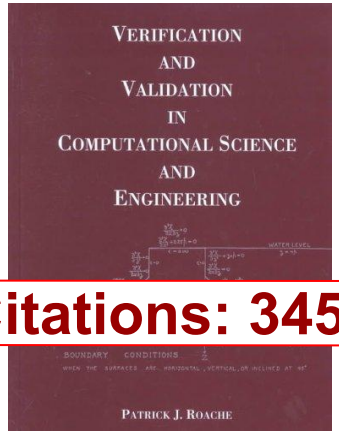
A car traveling at v_0=2 m/s suddenly starts accelerating at a=3 m/s². How long does it take to cover 100 m?

ANS: solving $0.5 \cdot a \cdot t^2 + v_0 \cdot t - 100 = 0$ with $t = (- v_0 + (v_0^2 + 4 \cdot 0.5 \cdot a \cdot 100)^{0.5}) / a$

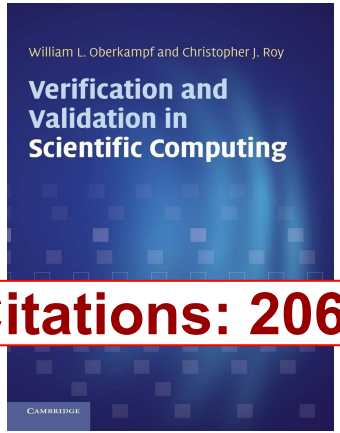
In Our Silly Example:

Code Verification	Does the code return $(- v_0 + (v_0^2 + 4 \cdot 0.5 \cdot a \cdot 100)^{0.5}) / a$?
Solution Verification	Is $(- v_0 + (v_0^2 + 4 \cdot 0.5 \cdot a \cdot 100)^{0.5}) / a$ the positive real solution of $0.5 \cdot a \cdot t^2 + v_0 \cdot t - 100 = 0$?
Validation	Does solving $0.5 \cdot a \cdot t^2 + v_0 \cdot t - 100 = 0$ really give the time needed for 100m?
Reproduction	Can we get the same needed time if we re-run the code?
Replication	Can others get the same needed time by other approaches? (e.g., numerically solving $ds/dt = v_0 + a \cdot t$)

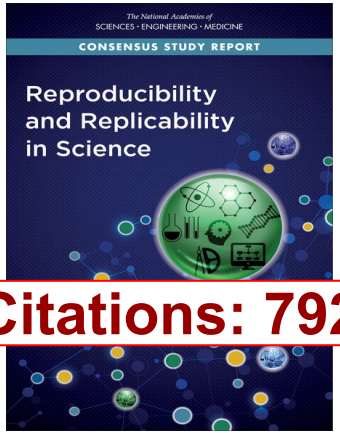
- Roache, P. J. (1998). Verification and validation in computational science and engineering
- Oberkampf, W. L., & Roy, C. J. (2010). Verification and validation in scientific computing.
- National Academies of Sciences, Engineering, and Medicine. 2019. Reproducibility and Replicability in Science.



Citations: 3459



Citations: 2060



Citations: 792

How many people in scientific computing care about the quality of scientific calculations?

We've known but neglected quality assurance for so long; is it worse in the modern days?

From the era of Fortran 77/90 + MPI to the era of Python + PyTorch/TensorFlow, what have changed?

High-Performance Scientific Computing: Not Just Coding

Given a problem we want to solve:

Paper & Pen	Code Development	End-Users
Physics Models	Implementations	Runtime System Configuration
Math Models	Optimizations	Solution Verification & Validation
Numerical Models	Code & Solution Verification	Design of Experiments
Solution Schemes	Packaging/Deployment	
Parallel Algorithms		

Past vs. Now

ALL tasks are critical to the performance/efficiency of computation

Past
More Self-Contained: All are important, so let's take everything into our own hands and control it ourselves.

Now
More Professionalized: All are important, so we must outsource each task to experts in each specific discipline.

- Don't know if we are doing it right or not
- Don't know if others are doing it right or not

Third-Party Dependencies

Past	Fewer | Slower Releasing | Shallower | More Centralized Development
Now	Tremendous | Faster Releasing | Deeper & Nested | More Community Contributions

- What's the vendor/versions of BLAS/LAPACK in your NumPy?
- What's the hardware configurations for OpenBLAS or MKL?
- What are the CUDA runtime configs of the PyTorch being used?
- What are the compilation flags of the Python packages?
- Are 64bit floats being used throughout all dependencies?

This is from SciPy... one of the most used Python packages now

How many people know that their numerical library is using another library that had a long-standing bug, which was fixed only in 2021?

Hardware	
Past	Homogeneous & Simpler
Now	Heterogeneous & Complicated

Programming Framework	
Past	Unified
Now	Diverse

- Heterogeneous & complicated hardware:
  - more code to develop
  - more code to maintain
  - more architecture-oriented coding
  - more prone to bugs
  - more V&V to do
- Diverse programming frameworks:
  - more context switches
  - less skilled with each framework
  - more prone to bugs
  - more V&V to do

Some Real-Life Cases/Stories

None of these stories could be
prevented by unit tests

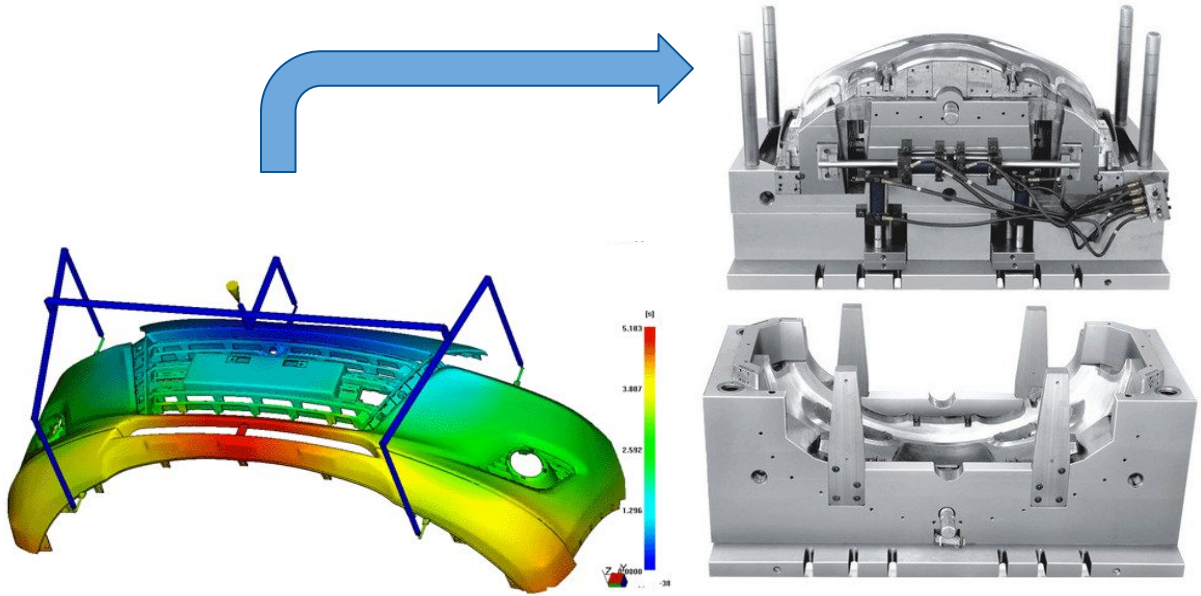
Story 1: Plastic Injection Molding (2012)

Loss

$1M

Cause

Forgot to update the version on the cluster after a local bug fix.



Story 2: Memory Leak & Out-of-Memory (2024)

Just two weeks ago, May 29th, 2024

Cause

Forgot to update the version on the cluster after a local bug fix.

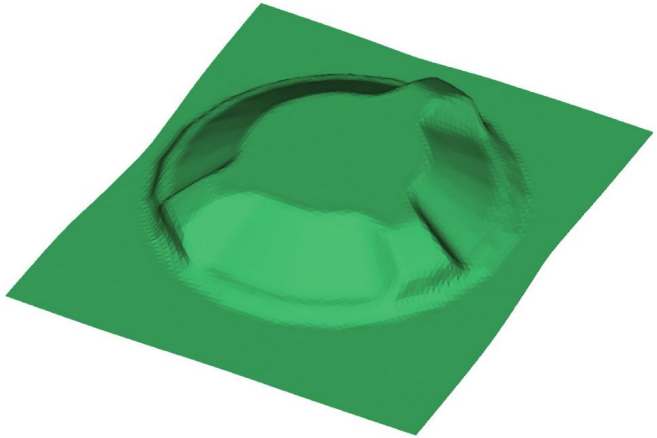
Loss

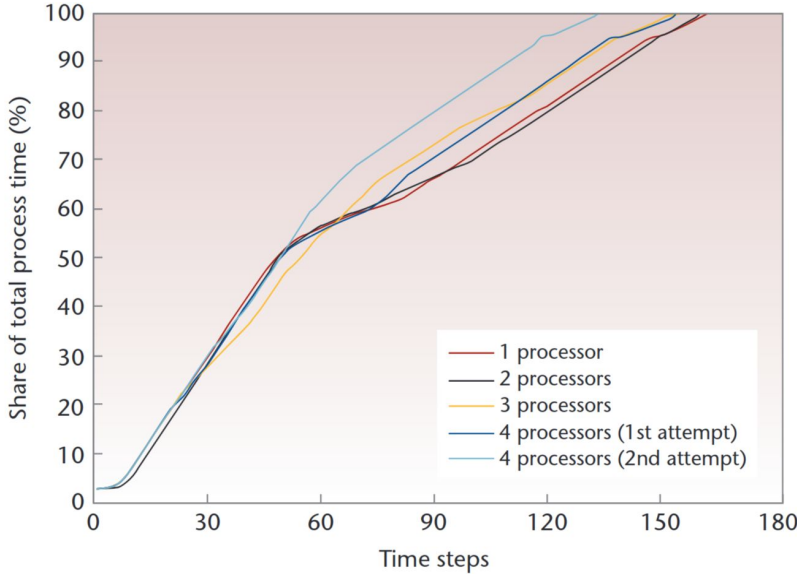
About 12k CPU hours of the allocation

Exactly the same cause of the incident 1. Nothing there to prevent the same accident after 12 years?

Story 3: Irreproducible MPI Simulations

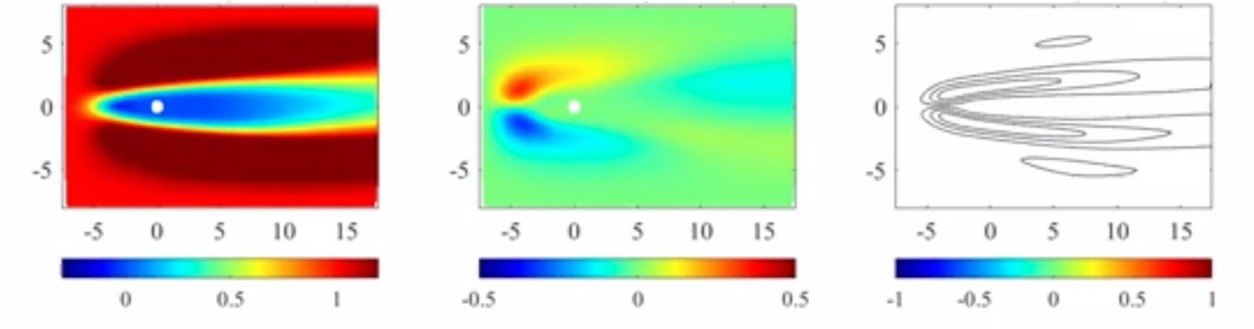
Diethelm, K. (2011). The limits of reproducibility in numerical simulation. Computing in Science & Engineering, 14(1), 64-72.





Cause: non-associativity of floating-point number in parallel reduction

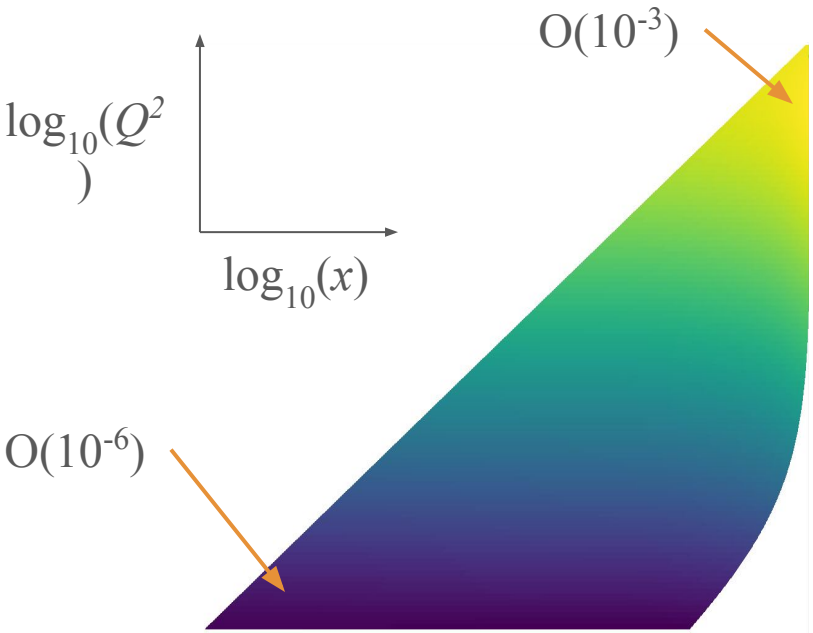
Story 5: AI-Predicted Impossible Cylinder Flow



Cause: ~~Exaggeration~~ Projection without Validation.

- We showed his "novel method" could not solve flow over a cylinder
- He was extremely angry... and showed us "working" videos and plots
- And we found apparent errors in these videos and plots

Story 6: Neutron Cross-Section—Python vs. C++ Code



Relative Difference in Neutron Cross-Section

- Prototyped in Python
- Re-implemented in C++
- Relative diff. in $[10^{-6}, 10^{-3}]$
- Engineers: "*It's a nuance due to compiler floating-point optimizations*"
- Physicists: *"This looks good."*

But there was actually a bug. What's missing?

How Should We Continue On Research w/ Distrust?

I am *scared* to fly on my students' airplane, but I still have to.

"scared"—it's all about **CONFIDENCE**

- Start talking reproducibility—GOOD
  - Easy for computational works: Docker, Singularity, Apptainer, etc
- Start talking open-science—GOOD
  - Stopped "code available upon request"

BUT

- They're showing honesty rather than correctness.
- It is the 21st century now. When can we move on to increasing confidence in calculation correctness?

Acknowledging and facing the problem is always the first step towards solving it.