Computation as a Tool for Discovery in Physics

Branislav K. Nikolić

Department of Physics & Astronomy, University of Delaware, Newark, DE 19716, U.S.A.

PHYS 460/660: Computational Methods of Physics

http://wiki.physics.udel.edu/phys660

Challenges for Computational Simulations

- **The performance challenge:**

  producing high-performance computers

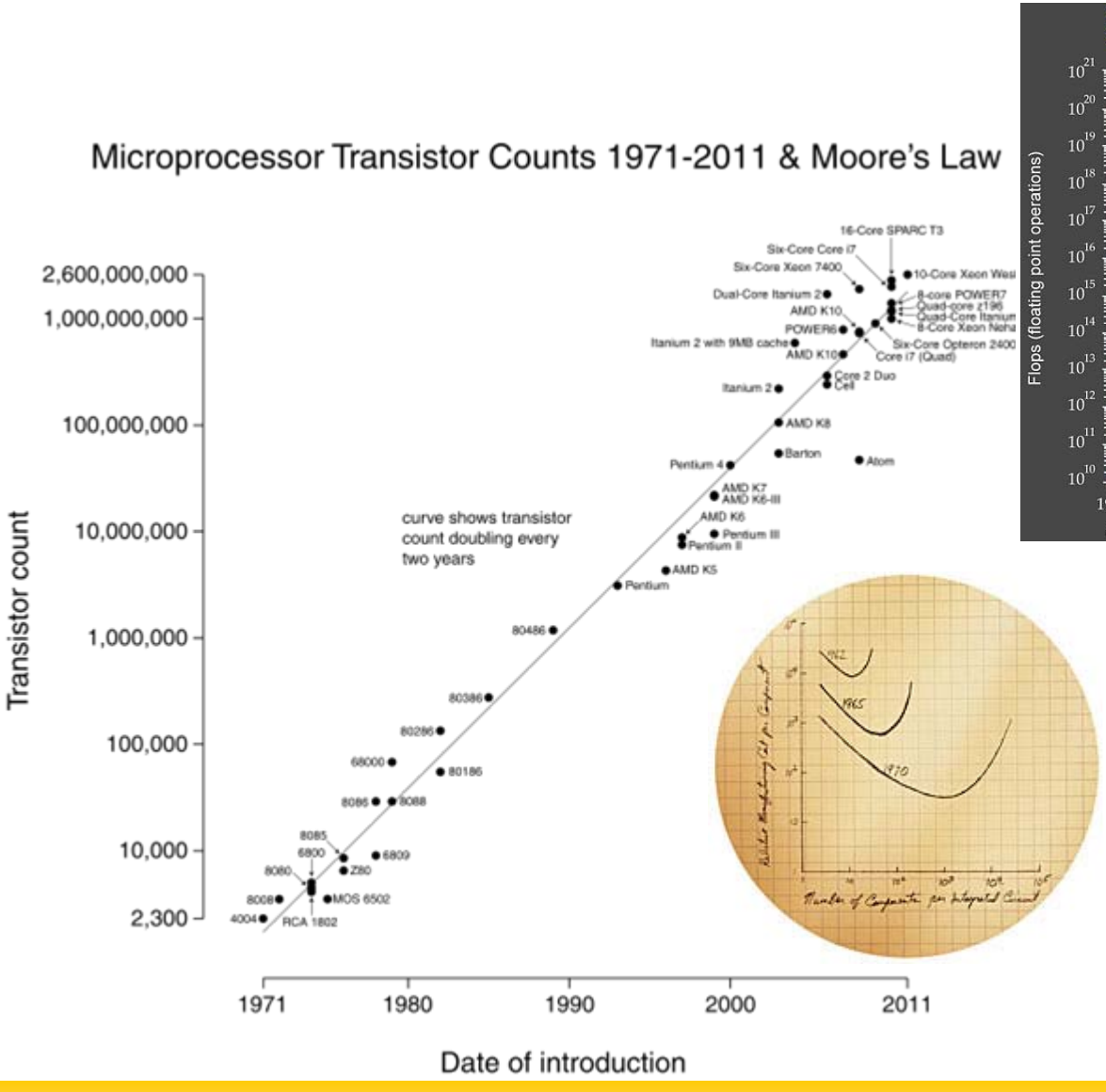
- **The programming challenge:**

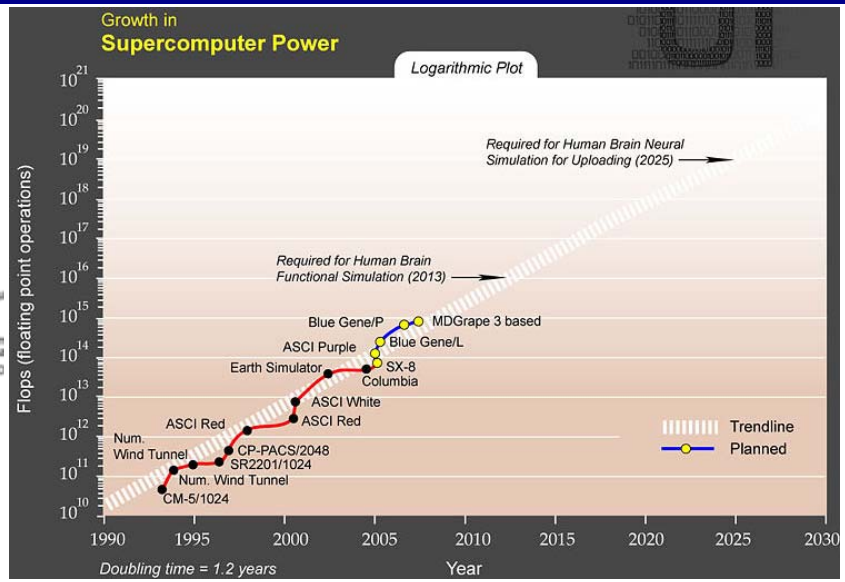
  programming for complex computers

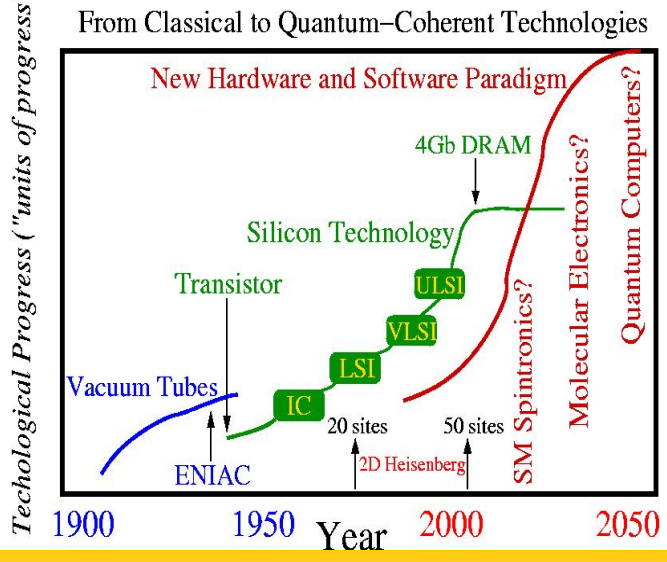
- **The prediction challenge:**

developing truly predictive complex application codes

Progress in Information Technologies vs. Progress in Scientific Computation





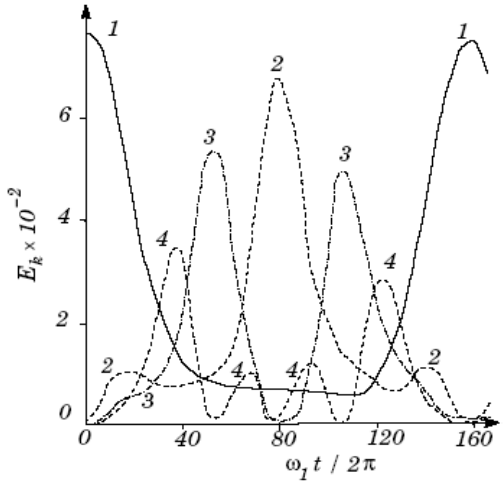


Early Days of Computational Physics: MRT and FPU Simulations

Metropolis, Rosenbluth, Teller (1953):
Monte Carlo simulation of hard disks
("**Metropolis algorithm**").

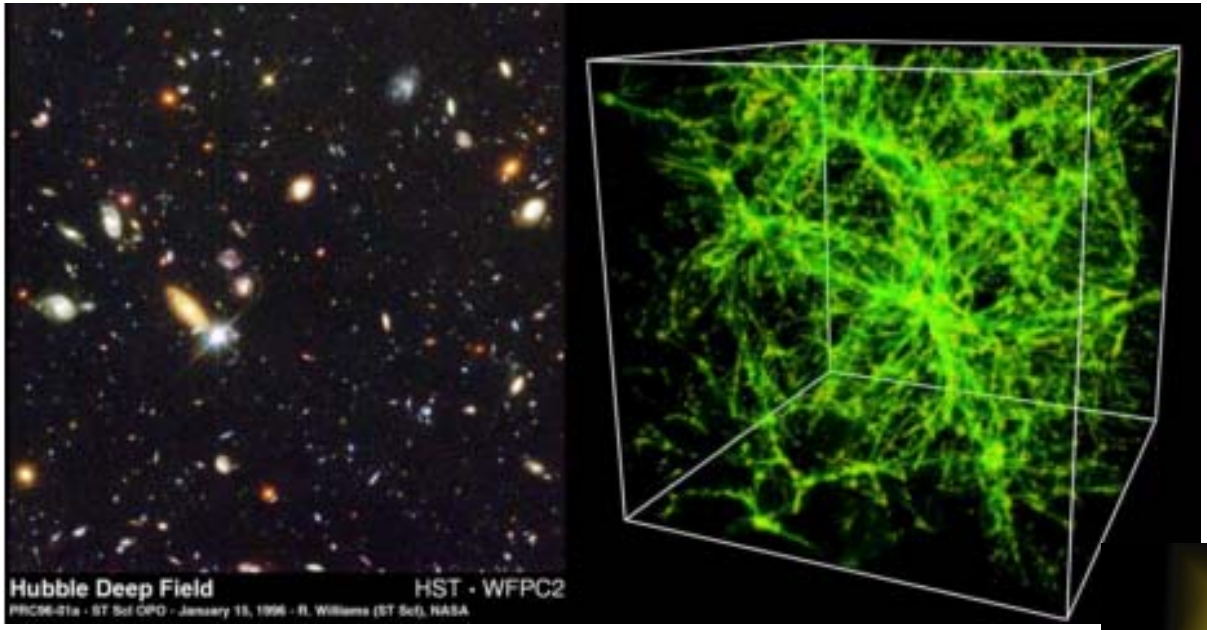
Fermi, Pasta, Ulam (1954): First computer experiment

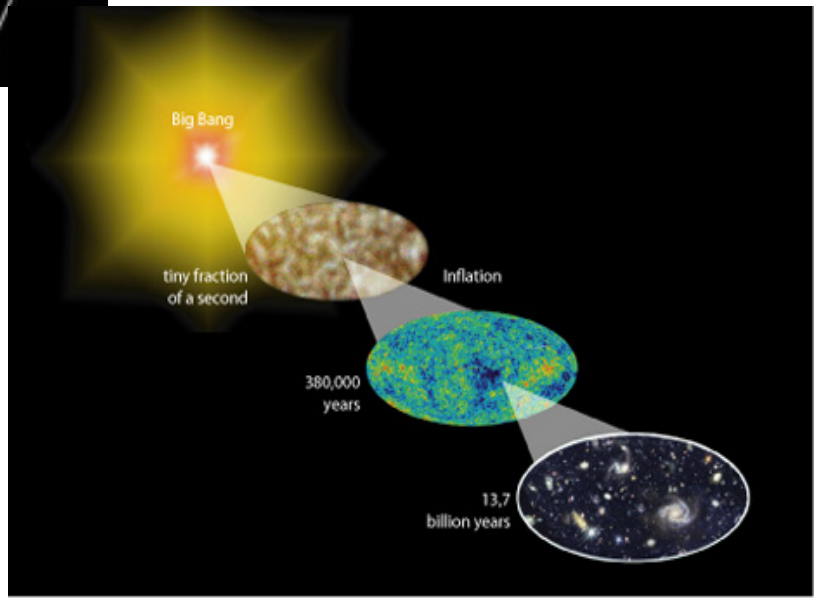
1-D anharmonic chain: $V = \sum_i \left[\left(q_{i+1} - q_i \right)^2 + \alpha \left(q_{i+1} - q_i \right)^3 \right]$



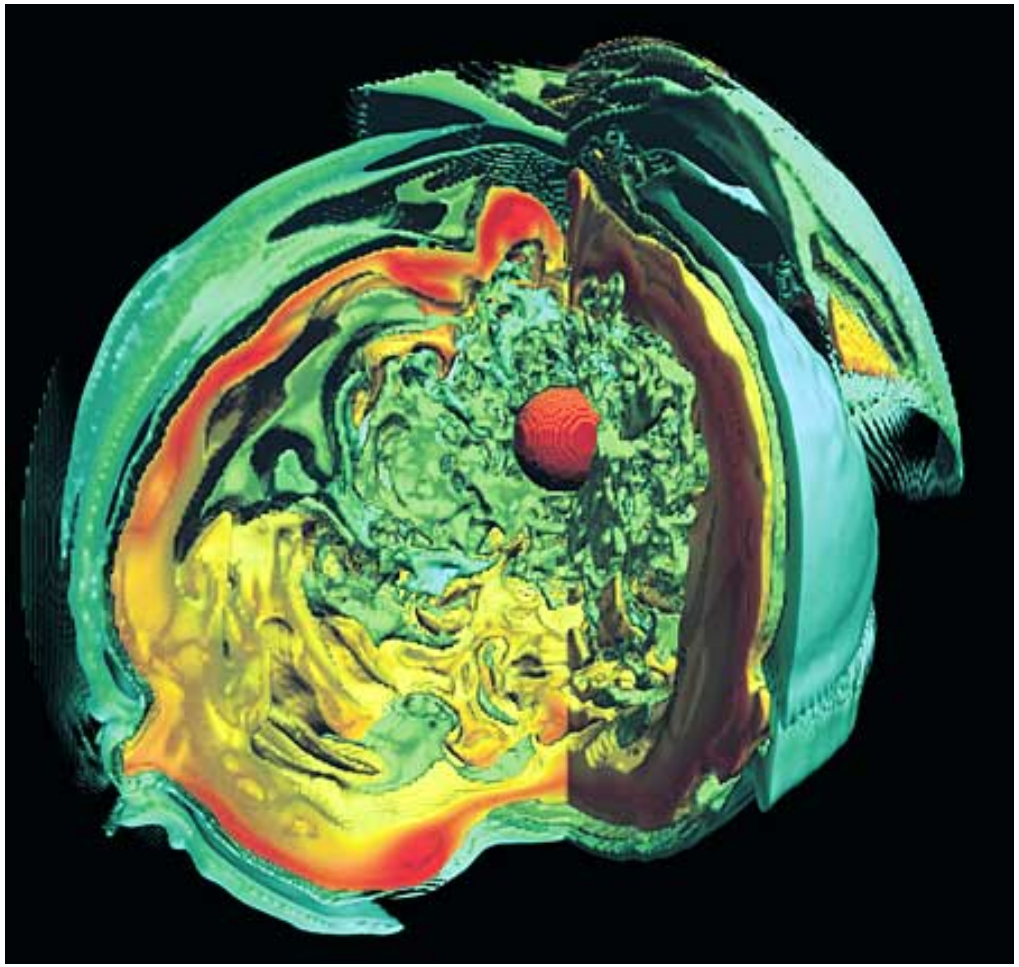
"Let us say here that the results of our computations were, from the beginning, surprising us. Instead of a continuous flow of energy from the first mode to the higher modes, all of the problems show an entirely different behavior. … Instead of a gradual increase of all the higher modes, the energy is exchanged, essentially, among only a certain few. It is, therefore, very hard to observe the rate of "thermalization" or mixing in our problem, and this was the initial purpose of the calculation."

Computation in Cosmology



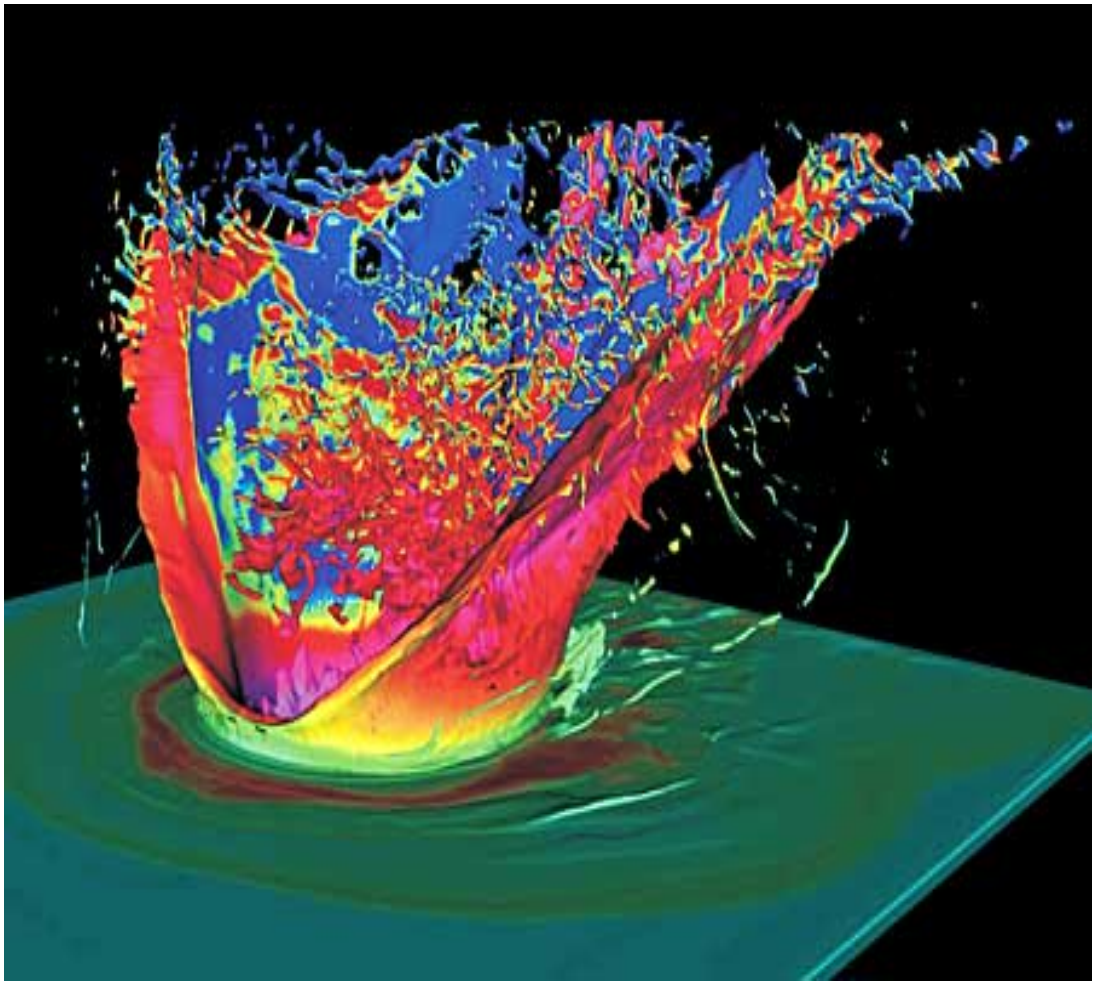


Computation in Astrophysics



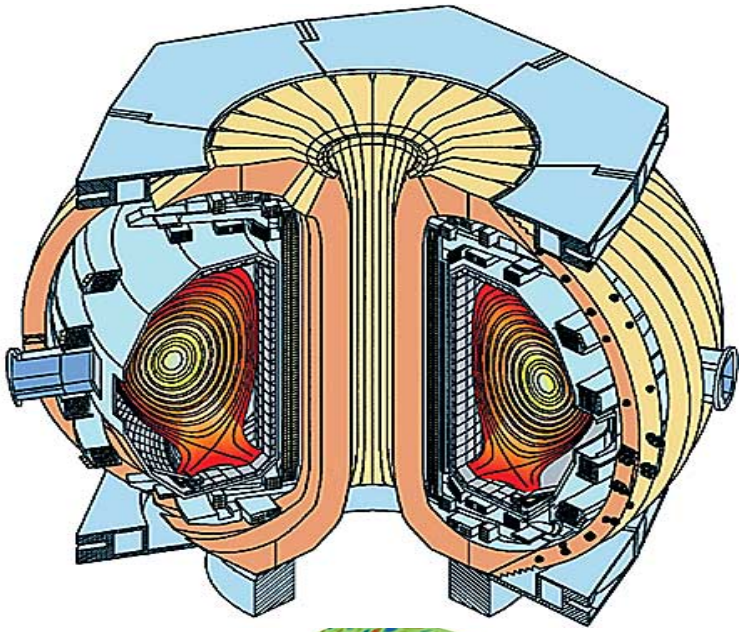
For a star **10 times heavier than the Sun**, about to explode as a **type II supernova**, the collapse of the star's iron core creates a hot fluid of nucleons. This model simulation shows the instability and turbulent flow of the hot nucleon fluid a fraction of a second after core collapse. Colors show the fluid's entropy: Red indicates the highest-entropy, hottest fluid; cooler, lower-entropy fluid is shown green. The outer surface of the nucleon fluid is just inside the expanding shock front that formed when the collapsing core rebounded off the superdense proto-neutron star (the small blue sphere). At the instant shown, the front is stalled at a radius of about 150 km. Reenergized by fluid instability, neutrino flux, and other mechanisms, the shock front will reach the star's surface in a few hours, thus creating the visible supernova. This image is based on a **three-dimensional simulation performed** at Oak Ridge National Laboratory's Center for Computational Sciences.

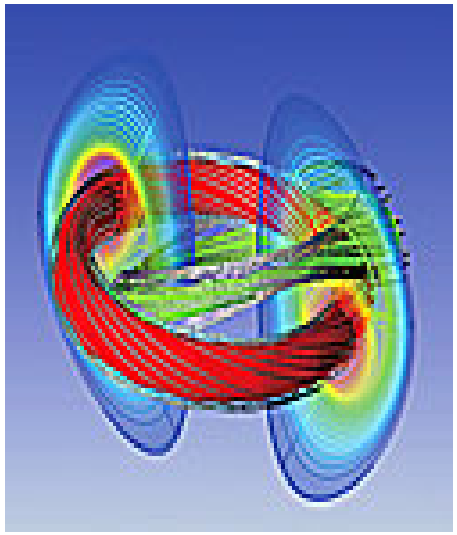
Computation in Geophysics

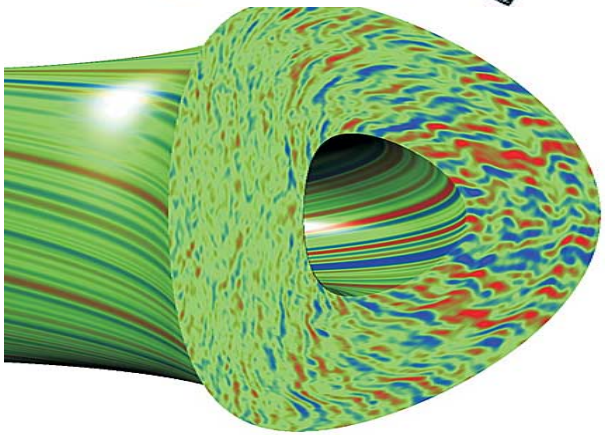


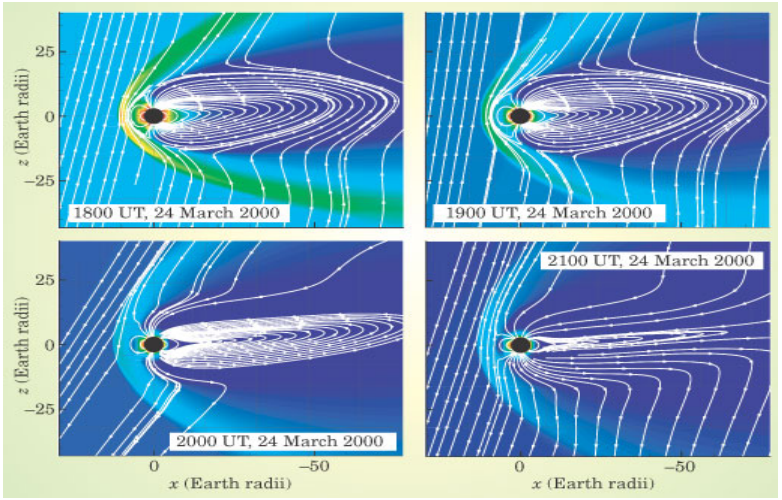
Simulated impact of the 10-km- diameter asteroid that struck the Yucatan peninsula **65 million years ago** and presumably triggered the **worldwide extinction of the dinosaurs**. Shown here 42 seconds after impact, the expanding column of debris from the asteroid and crater is about 100 km high. Colors indicate temperature: The hottest material (red) is at about 6000 K, and the coolest (blue) has returned to ambient temperature. The simulation uses the SAGE code developed at Los Alamos National Laboratory.

Computation in Plasma Physics

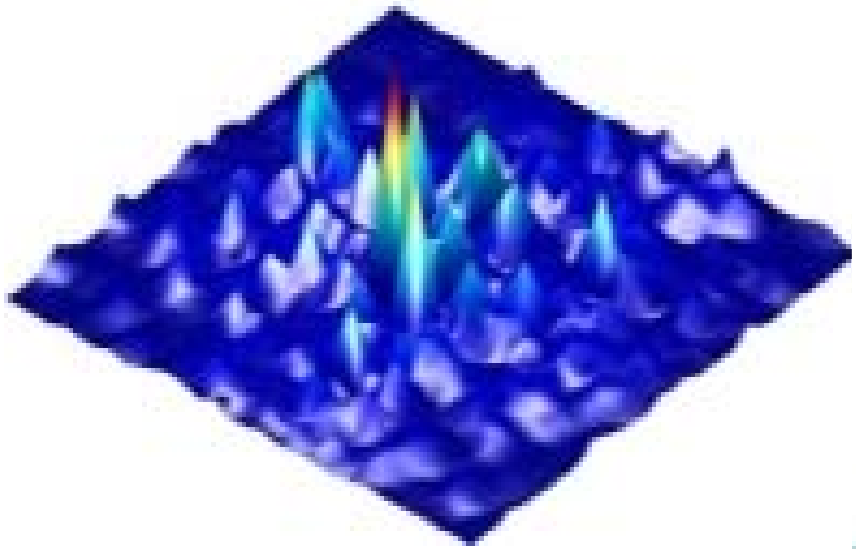


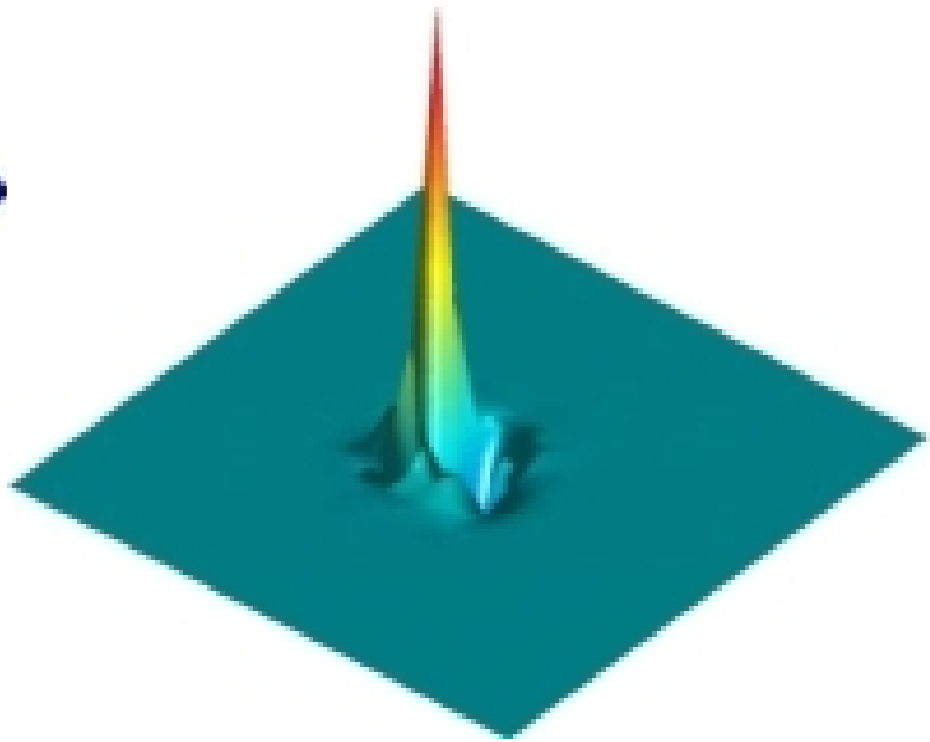


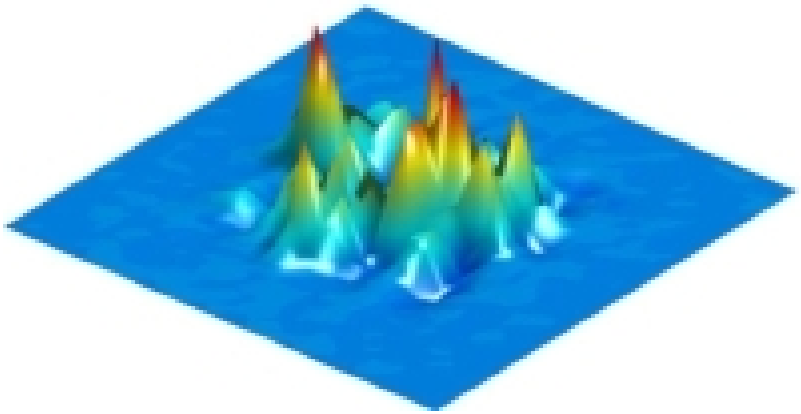




Computation in Atomic and Molecular Physics

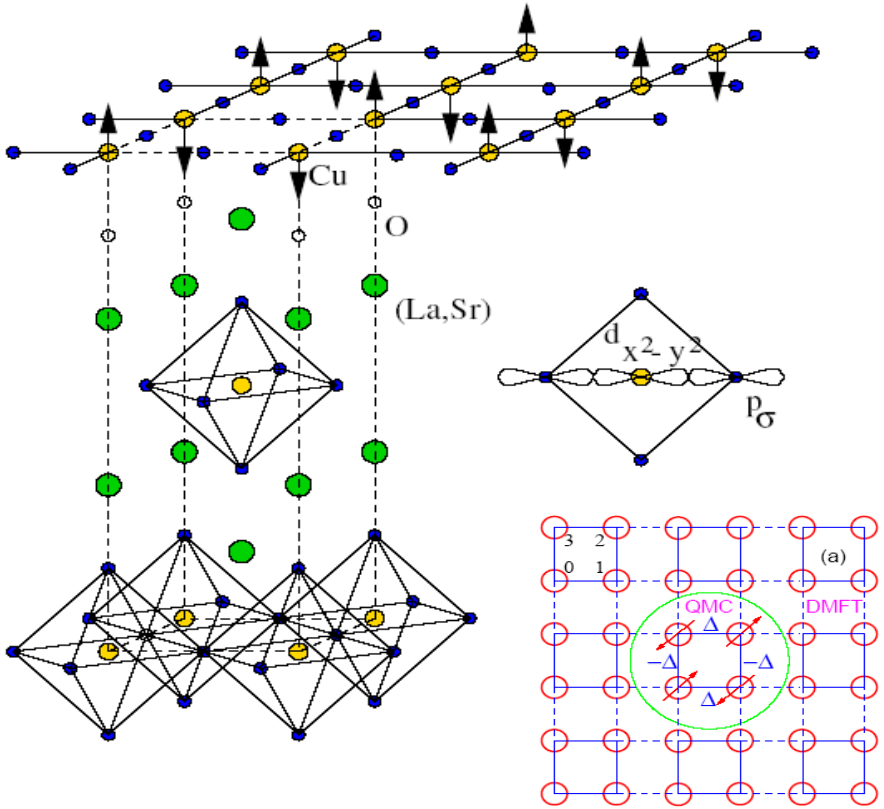






Computation in Condensed Matter Physics

"The general theory of quantum mechanics is now almost complete. The underlying physical laws necessary for the mathematical theory of a **large part of physics and the whole of chemistry** are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble." —P. A. M. Dirac (1929)



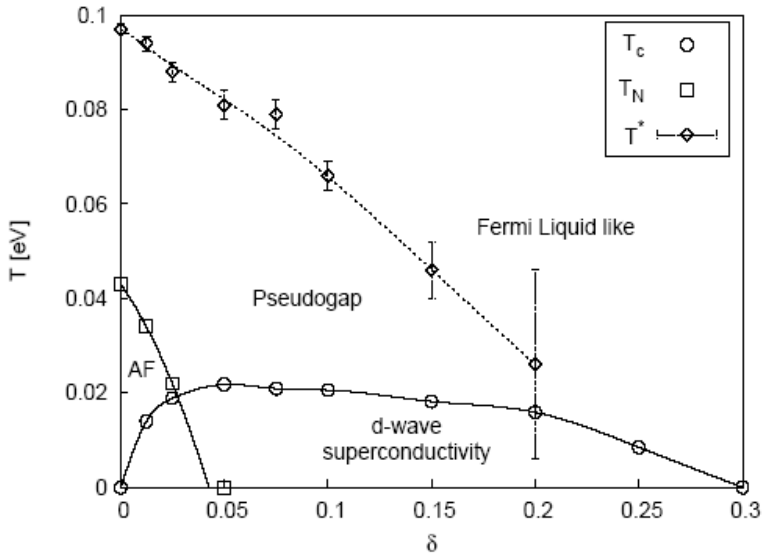
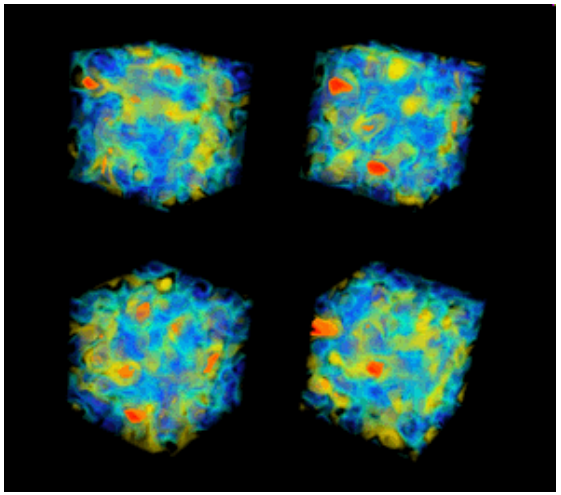
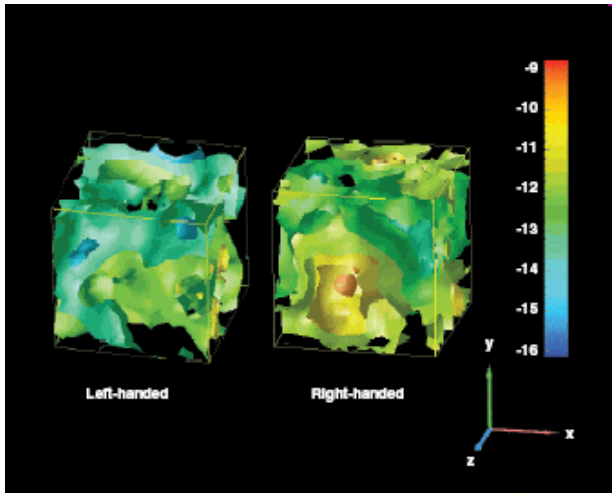
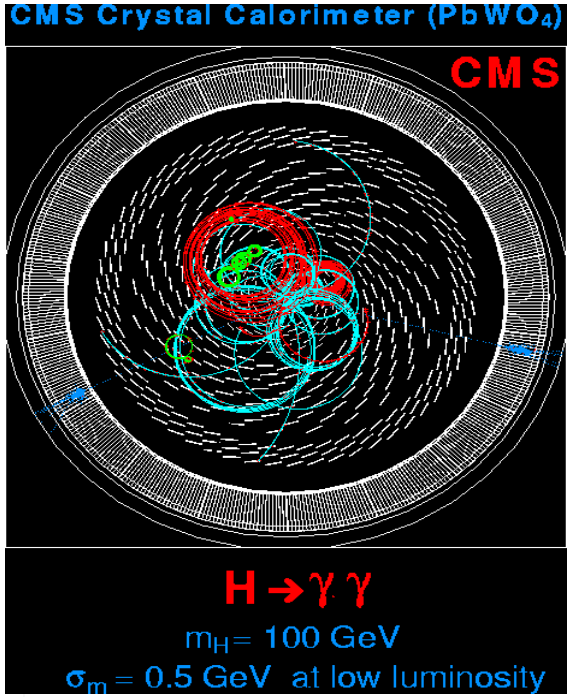


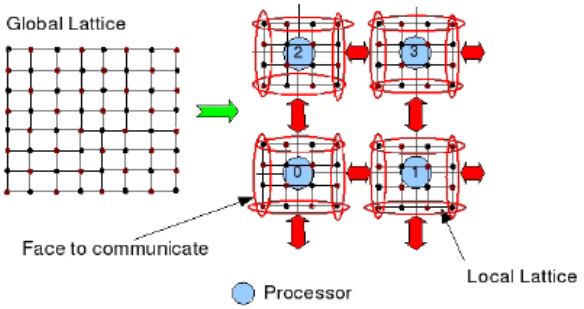
Figure 41 Temperature-doping phase diagram of the 2D Hubbard model when $U = 8t$ calculated with DCA/QMC for a 4-site cluster, $N_c = 4$. The error bars on T^* result from the difficulty in locating the maximum in the uniform spin susceptibility. Regions of antiferromagnetism, d-wave superconducting and pseudogap behavior are seen.

Computation in High-Energy Physics









: A QCDOC mother board which holds 64 nodes as a 2^6 hypercube.

Computation vs. Lab Experiments vs. Theory

Computer Simulation	Laboratory Experiment
model	sample
program	physical apparatus
testing of program	calibration
computation	measurement
data analysis	data analysis

Why do Computer Simulations?

- Simulations are the only general method for "solving" many-body problems.
- Experiment is limited and expensive. Simulations can complement the experiment.
- Simulations can be easy even for complex systems.
- They scale up with the computer power.

Two Routes in Computer Simulations of Physical Processes

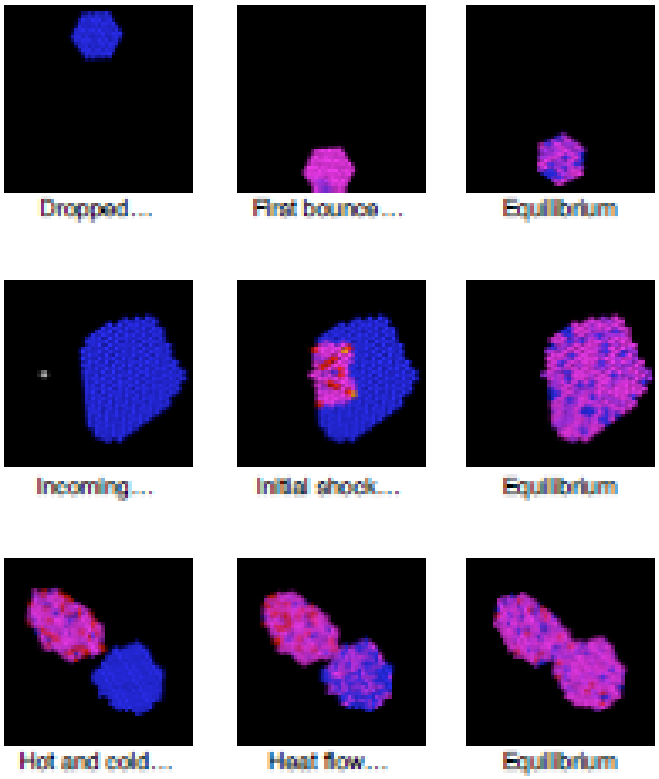
- Newton, Maxwell, Boltzmann and Schrödinger gave us the model. All we must do is numerically solve the mathematical problem and determine the properties → first principles or ab initio methods.

- Give us the phenomena and invent a model to mimic the problem. The semi-empirical approach. But one cannot reliably extrapolate the model away from the empirical data.

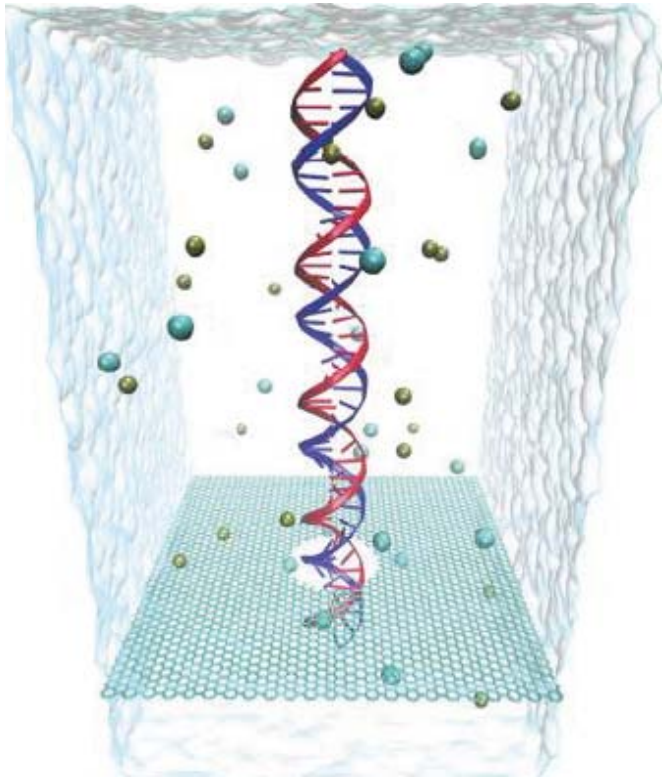
Example: Molecular Dynamics

1. Pick particles, masses and potential.
2. Initialize positions and momentum. (boundary conditions in space and time)
3. Solve $\vec{F} = m\vec{a}$ to determine $\vec{r}(t), \vec{v}(t)$.
4. Compute properties along the trajectory.
5. Estimate errors.
6. Try to use the simulation to answer physical questions.

In teaching
(Statistical mechanics illustrations by Daniel Schroeder)



In research
(DNA translocation through nanopores)



Know the Computational Complexity of Your Problem Before You Put it on the Computer

- **Computational Complexity:** The branch of computer science which classifies problems according to the computational resources required to solve them.

- Time complexity: $T(x) = \max_{|x|=n} t(x)$

CPU clock independent unit of time: Time it takes to perform an elementary operations such as adding two integer numbers.

$$T(n) = \Theta(g(n)), \quad c_1 g(n) \le T(n) \le c_2 g(n), \forall n \ge n_0$$

- Example: $A_{n\times n} \cdot B_{n\times n} \Rightarrow \begin{cases} T(n) = \Theta(n^3) & \text{textbook} \\ T(n) = \Theta(n^{2.376}) & \text{research} \end{cases}$

- Tractable versus Intractable: $\Theta(n^k)$ vs. $\Theta(2^n)$ or $\Theta(n!)$

Example: Hubbard model on 16 x 16 with 220 electrons $\Rightarrow \dim(H) = 10^{150}$

Computational Complexity References

LIMITS OF COMPUTATION

Computational Complexity for Physicists

The theory of computational complexity has some interesting links to physics, in particular to quantum computing and statistical mechanics. This article contains an informal introduction to this theory and its links to physics.

Compared to the traditionally close relationship between physics and mathematics, an exchange of ideas and methods between physics and computer science barely exists. However, the few interactions that have gone beyond Fortran programming and the quest for faster computers have been successful and have provided surprising insights in both fields. This is particularly true for the mutual exchange between statistical mechanics and the theory of *computational complexity*. Here, I discuss this exchange in a manner directed at physicists with little or no knowledge of the theory.

The measure of complexity

The branch of theoretical computer science known as computational complexity is concerned with classifying problems according to the computational resources required to solve them (for additional information about this field, see the "Related Works" sidebar). What can be measured (or computed) is the time that a particular algorithm uses to solve the problem. This time, in turn, depends on the algorithm's implementation as well as the computer on which the program is running.

The theory of computational complexity provides us with a notion of complexity that is largely independent of implementation details and the computer at hand. Its precise definition requires a considerable formalism, however. This is not surprising because it is related to a highly nontrivial question that touches the foundation of mathematics: *What do we mean when we say a problem is solvable?* Thinking about this question leads to Gödel's incompleteness theorem, Turing machines, and the Church-Turing thesis on computable functions.

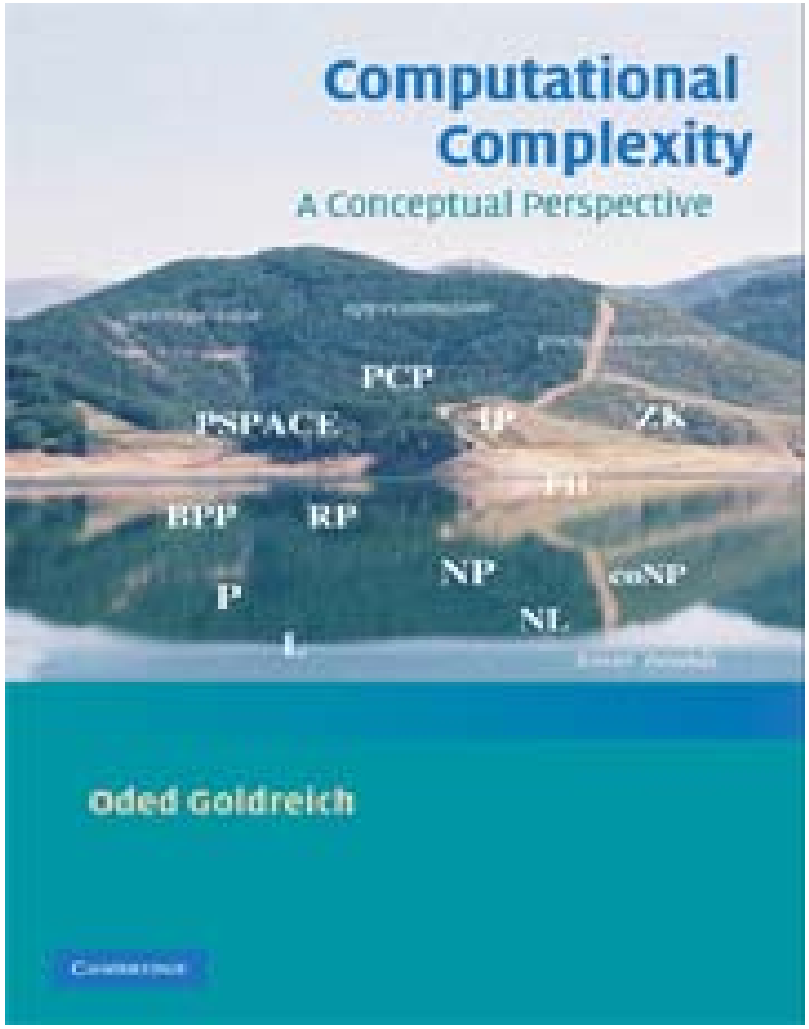
Here we adopt a more informal, pragmatic viewpoint. A problem is solvable if a computer program written in your favorite programming language can solve it. Your program's running time or *time complexity* must then be defined with some care to serve as a meaningful measure of the problem's complexity.

Time complexity

In general, running time depends on a problem's size and on the specific input data—the *instance*. Sorting 1,000 numbers takes longer than sorting 10 numbers. Some sorting algorithms run faster if the input data is partially sorted already. To minimize the dependency on the specific instance, we consider the *worst-case* time complexity $T(n)$:

1521-9615/02/$17.00 © 2002 IEEE

Stephan Mertens
Otto-von-Guericke University, Magdeburg



Code Verification Techniques

VERIFICATION: Does the code solves the chosen model correctly?

The few existing studies of error levels in scientific computer codes indicate that the defect rate is about **seven faults per 1000 lines of FORTRAN code.**

- Comparing code results to a related problem with an exact answer.
- Establishing that the convergence rate of the truncation error with changing grid spacing is consistent with expectations.
- Comparing calculated with expected results for a problem specially manufactured to test the code.
- Monitoring conserved quantities and parameters, preservation of symmetry properties, and other easily predictable outcomes.
- Benchmarking—that is, comparing results with those from existing codes that can calculate similar problems.

Code Validation Techniques

VALIDATION: Does the model itself captures the essential physical phenomena with adequate fidelity?

- Passive observations of physical events—for example, weather or supernovae.
- Controlled experiments designed to investigate specific physics or engineering principles—for example, nuclear reactions or spectroscopy.
- Experiments designed to certify the performance of a physical component or system—for example, full-scale wind tunnels.
- Experiments specifically designed to validate code calculations—for example, laser-fusion facilities.