Introduction

Computer Experiment

- **Computational science**: An area of scientific investigation, where computers play a central role.
- **Scientific computing**: An area in computer science to support computational sciences by innovative use of computer systems; it involves the development of numerical algorithms, software tools, scientific visualization, etc.
- **Computational-science approach**

1. **Mathematical model** is developed for the **physical phenomenon** of interest.
   
   Example: Newton’s second law of motion for interacting particles. Three laws of motion were published by Isaac Newton in *Mathematical Principles of Natural Philosophy* (1686).

   \[
   m \frac{d^2 \vec{r}_i(t)}{dt^2} = \vec{F}_i(t) \quad (i = 1, \ldots, N)
   \]  

2. The equations of the mathematical model are cast into a **discrete algebraic form**, which is amenable to numerical solution.
   
   Example: Time discretization of the Newton’s second law of motion.

   \[
   m \frac{\vec{r}_i(t + \Delta t) - 2\vec{r}_i(t) + \vec{r}_i(t - \Delta t)}{\Delta t^2} = \vec{F}_i(t) \quad (i = 1, \ldots, N; t = 0, \Delta t, 2\Delta t, \ldots)
   \]

3. **Numerical algorithms** are used to convert the algebraic equation system into a simulation program.

   **Example 1**: Verlet algorithm.
   
   - compute \( F_i(t) \) as a function of \( x_i(t) \)
   - \( x_i(t+Dt) = 2x_i(t) - x_i(t-Dt) + F_i(t)Dt^2/m \)
   - \( v_i(t) = (x_i(t+Dt) - x_i(t-Dt))/2Dt \)

   **Example 2**: velocity-Verlet algorithm.
   
   - compute \( F_i(t) \) as a function of \( x_i(t) \)
   - \( v_i(t+Dt/2) = v_i(t) + F_i(t)Dt/2m \)
   - \( x_i(t+Dt) = x_i(t) + v_i(t+Dt/2)Dt \)
   - compute \( F_i(t+Dt) \) as a function of \( x_i(t+Dt) \)
   - \( v_i(t+Dt) = v_i(t+Dt/2) + F_i(t+Dt)Dt/2m \)
4. **Computer experiments** are performed to follow the time evolution of the model physical system.

---

**AXIOMS, OR LAWS OF MOTION**

**LAW I**

Every body continues in its state of rest, or of uniform motion in a right line, unless it is compelled to change that state by forces impressed upon it.

**LAW II**

The change of motion is proportional to the motive force impressed, and is made in the direction of the right line in which that force is impressed.

If any force generates a motion, a double force will generate double the motion, a triple force triple the motion, whether that force be impressed altogether and at once, or gradually and successively. And this motion (being always directed the same way with the generating force), if the body moved before, is added to, or subtracted from, the former motion, according as they directly coincide with or are directly contrary to each other; or obliquely joined, when they are oblique, so as to produce a new motion compounded from the determination of both.

**LAW III**

To every action there is always opposed an equal reaction; or the mutual actions of two bodies upon each other are always equal, and directed to contrary parts.

Whatever dam or presses another is as much drawn or pressed by that other. If you press a stone with your finger, the finger is also pressed by the stone.

---

**Figure.** First few pages of Newton’s *Mathematical Principles of Natural Philosophy* (1686).
- **Type of mathematical models**

<table>
<thead>
<tr>
<th></th>
<th>Particle model (ordinary differential equations)</th>
<th>Continuum model (partial differential equations)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>molecular dynamics</td>
<td>computational fluid dynamics, continuum mechanics</td>
</tr>
<tr>
<td>Stochastic</td>
<td>Monte Carlo particle simulation</td>
<td>quantum Monte Carlo</td>
</tr>
</tbody>
</table>

- **Particle vs. continuum models**: Mathematical models are either the particle type or the continuum type. Particle models trace the motion of many interacting particles, an example being Newton’s second law of motion. Particle-type laws are typically formulated as coupled ordinary differential equations.

Continuum models deal with functions extending over the space. For example, the dynamics of a quantum particle is described by a parabolic partial differential equation called the Schrödinger equation,

$$i\hbar \frac{\partial}{\partial t} \psi(\vec{r}, t) = \left( -\frac{1}{2m} \left( \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \right) + v(\vec{r}) \right) \psi(\vec{r}, t), \quad (2)$$

where $i = \sqrt{-1}$, $\hbar = 1.05 \times 10^{-34}$ J•s is the Planck constant, and $\psi(\vec{r}, t)$ is a complex-valued wave function. The square, $|\psi(\vec{r}, t)|^2$, of the wave function is the probability to find the particle at position $\vec{r}$ at time $t$.

![Molecular dynamics simulation](image)

**Figure.** (Left) A molecular-dynamics simulation consists of a collection of atoms, which exert forces to each other, depending on their mutual interactions and relative positions. (Right) An example of continuum data—uniaxial stress field in a cracked thin plate made of gallium-arsenide material.

- **Molecular dynamics (MD)**: Follows Newton’s second law of motion for interacting particles.

- **Deterministic vs. stochastic simulations**: Computer simulations are either deterministic or stochastic. Deterministic simulations usually deal with mathematical initial value problems, i.e., differential equations such as Eqs. (1) and (2) are integrated forward in time starting with some initial configuration.

Stochastic simulations use random numbers to: 1) provide approximate solutions to large-scale problems where deterministic solutions are intractable (e.g., statistical mechanics in physics); or 2) simulate stochastic natural phenomena (e.g., stock price).

- **Monte Carlo (MC) method**: A computational method that utilizes random numbers.

  Example: Stock price.
Figure. (Top) Typical fluctuation in stock price. (Bottom) Stock portfolio trading at Quantlab Financial LLC (courtesy of Dr. Andrey Omeltchenko, CACS graduate).

Application of Molecular Dynamics

- Drug design
- Materials design
- Robotics
- Computer graphics
- Games

---

1One example of the use of particle systems for computer animation is found in Twister (1996), in which particle systems are used to simulate a tornado.
History of Particle Simulations

1944  John von Neumann was attracted to the ENIAC project (the world’s first operational electronic, general-purpose computer); he wrote a memo proposing a stored-program computer. He believed: “Our present analytical methods seem unsuitable for the solution of the important problems arising in connection with nonlinear partial differential equations. The really efficient high-speed computing devices may provide us with those heuristic hints which are needed in all parts of mathematics for genuine progress.”

1953  The first MC simulation of a liquid by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller performed on the MANIAC computer at Los Alamos National Laboratory.

1955  Enrico Fermi, John Pasta, and Stanislaw Ulam studied the dynamics of a one-dimensional array of particles coupled by anharmonic springs on the MANIAC.

1956  Dynamics of hard spheres (billiards) studied by Alder and Wainwright at the Lawrence Livermore National Laboratory.

1960  Radiation damage in crystalline Cu studied with short-range repulsion and uniform attraction toward the center by George Vineyard’s group at the Brookhaven National Laboratory.

1964  The first MD simulation of a liquid (864 argon atoms) using interatomic potentials by Aneesur Rahman at the Argonne National Laboratory using a CDC 3600 computer.

Current state-of-the-art MD simulations consist of over trillion atoms.\textsuperscript{3,4} For example, an MD program developed at the Collaboratory for Advanced Computing and Simulations (CACS) at the University of Southern California (USC) has simulated material consisting of 4.95 trillion atoms using 786,432 IBM BlueGene/Q processors. For a harder problem—atomistic simulation in which interatomic forces are computed quantum mechanically, CACS scientists have performed 39.8 trillion electronic degrees-of-freedom calculation.
History of Supercomputers

The world’s first general-purpose electronic computer was ENIAC (see the figure in the right) built by Presper Eckert and John Mauchly at the University of Pennsylvania during World War II. The computer was intended to be used for calculating ballistic trajectories—a kind of particle dynamics. (Eckert was only 22 years old when he and Mauchly started the project with $half-million support from Army.) However, rewiring this computer to solve a new problem required days of work by a number of operators.

The era of vector supercomputers started in 1976 when Seymour Cray built Cray-1 (see the figure in the left). Vector processing is a type of parallelism, which speeds up computation.

In late 80’s, massively parallel computers such as the Thinking Machines CM-2 (right) became the central technology for supercomputing. The Cray T3E computer, for example, is a collection of 450MHz Digital Alpha processors connected to form three-dimensional grids.

Another important development is the invention of the microprocessor—a computer on a single semiconductor chip. The first microprocessor, Intel 4004 introduced in 1971, contained 2300 transistors and was 0.3 by 0.4 cm in size. In comparison, Pentium 4 contains 125 million transistors. Intel had a 16-bit processor two years before its competitors’ more elegant architectures, such as Motorola 68000, and this head start led to the selection of 8086 as the CPU for the IBM PC in 1981. (Andy Grove, the CEO of Intel, was the ’97 TIME Man of the Year.)

Merge of PC and supercomputer technologies: The later trend in computer technology was that the PC and supercomputer technologies are merging. For example, we acquired a Linux cluster consisting of 512 dual Intel Xeon 1.8 GHz nodes (i.e., 1,024 processors) connected by Myricom’s Myrinet interconnect at Louisiana State University (LSU) in 2002. The performance of the $2.6 million cluster, SuperMike, was rated as 2.21 teraflops ($10^{12}$ floating-point operations per second), according to the standard High Performance Linpack (HPL) benchmark (http://www.netlib.org/benchmark/hpl), and SuperMike was ranked as the 11th fastest supercomputer in the world in August 2002 (http://www.top500.org).

Figure. Dramatic increase of the number of transistors in a microprocessor.

Figure. The 1,024-processor Xeon cluster, SuperMike, at LSU.

Figure. The 1,024-processor Xeon cluster, SuperMike, at LSU.
The current world’s fastest supercomputer is the Tianhe-2 computer at the National Super Computer Center in Guangzhou, China. Tianhe-2 consists of 3.12 million processors, and its theoretical peak performance is 54.9 petaflops (1 petaflops = $10^{15}$ floating-point operations per second; see the figure). The actual measured performance of Tianhe-2 for the Linpack benchmark program is 33.9 petaflops.

You can also build PC clusters by yourself. The CACS at the USC has a 4,096-processor PC cluster. The Center for High Performance Computing and Communications (HPCC) at the USC has a 10,920-processor Linux cluster with 0.53 petaflops Linpack performance, which you will use in this class.

Beginning of the many-core parallel computing era: Computer industry is facing a historical shift, in which Moore’s law due to ever increasing clock speeds has been subsumed by increasing numbers of “cores” per microchip. Intel has already demonstrated an 80-core microchip that executed trillion operations per second with mere 62 W of power (http://en.wikipedia.org/wiki/Teraflops_Research_Chip), and the number of cores per microchip is expected to double at each generation, reaching thousands in 10 years. In addition to such many-core central processing units (CPUs), advanced graphics processing units (GPUs) in desktop computers will add extra multi-trillion operations with minimal (~ $ hundred) cost. The many-core revolution will mark the end of the free-ride era (i.e., legacy software will run faster on newer chips), resulting in a dichotomy—subsiding speed-up of conventional software and exponential speed-up of scalable parallel-computing applications. To develop a many-core CPU/GPU computing framework applied to broad applications, we have constructed a test bed consisting of a cluster of 9 Playstation3 consoles (each Playstation3 box containing one power processing and 8 streaming processing units as well as a GPU) at CACS (see the figure).

Enabling Technologies

Parallel Computing

Parallel computing technology has extended the scope of computer simulations in terms of simulated system size. In order to perform parallel computer simulations efficiently, however, algorithms developed for serial computers must often be modified. We will learn parallel MD algorithms in this lecture.

Parallel computing requires decomposing the computation into subtasks and mapping them to multiple processors. For MD simulations, the divide-and-conquer strategy based on spatial decomposition is commonly used. The total volume of the system is divided into $P$ subsystems of equal volume, and each subsystem is assigned to a node in an array of $P$ processors (see the figure below). In this lecture, you will learn message passing interface (MPI) programming on multiple computers and thread (OpenMP) programming on multicore processors. In addition, a brief overview will be given on the programming on Cell Broadband Engine on PlayStation3 consoles, GPUs using compute unified device architecture (CUDA), and prototype multicore processors such as 64-core Godson-T.
Visualization

Impact of these grand-challenge simulations on parallel computers cannot be fully realized without major breakthroughs in scientific visualization. The current practice of sequentially processing visualization data is highly ineffective for large-scale applications that produce terabytes of data. The only viable solution is to integrate visualization into simulation so that they are both performed concurrently on multiple parallel machines and then to examine the results in real time in three-dimensional immersive and interactive virtual environments.

Virtual environment such ImmersaDesk (see the figure below) enables direct interaction with large datasets in three dimensions. Visual feedback allows us to zoom in on part of a simulation, change parameters, and thus guide the progress of a simulation in real time. Furthermore, with virtual environment the observer is inside the simulation instead of getting a glimpse of the data from the outside. These immersive and interactive features provide invaluable insight into the simulations. In this lecture, we will learn OpenGL programming as a foundation of scientific visualization, as well as virtual-reality programming using CAVE Library. In addition to an ImmersaDesk, the CACS has an 8' × 14' tiled display, which is driven by a 26-processor Linux cluster (see the figure below).

CACS scientists have demonstrated real-time, interactive visualization of a billion-particle dataset in an immersive virtual reality environment, using advanced computer-science techniques such as: 1) multilevel view-frustum culling; 2) probabilistic occlusion culling; and 3) parallel/distributed preprocessing of visualization data on a remote PC cluster.\textsuperscript{14,15}
Data Management

A serious technological gap exists between the growth in processor power and that of input/output (I/O) speed. Typically, a billion-atom MD simulation produces 0.1 terabytes (TB) of data (to store atom types, coordinates, velocities, and stresses) per frame, which amounts to 10 TB per day if the simulation runs for 1,000 steps and data are saved after every 10 steps. The I/O (including data transfer to remote archival storage devices) has thus become the bottleneck in large-scale simulations. Efficient schemes to manage these massive data are crucial. In this lecture, we will learn data compression approaches for the I/O problem.16

Visualization of collective motion of many atoms is a difficult task because of the high dimensionality (3N dimensions for N atoms) of the space in which the collective motion occurs. A challenge is to extract topological defects, such as dislocations, and their activities from massive data with large thermal noises, especially at high temperatures. This will require nontrivial knowledge discovery or data-mining processes from very large noisy data sets.17 We will learn the use of data-mining algorithms using graph data structures,18,19 which is at the heart of recent federal and industrial initiatives on “Big Data”.20

Distributed Computing

Often a single parallel supercomputer does not provide sufficient computing power for grand challenge problems. In such a case, “metacomputing” uses multiple supercomputers connected by wide-area, high-speed networks. The “Grid” of geographically distributed petaflops computers and immersive/interactive virtual reality environments connected via high-speed networks can revolutionize/democratize science, by enabling hybrid simulations that integrate multiple expertise distributed globally.21 Such collaborative Grid computing will form the core of the new initiative on “Revolutionizing Science and Engineering through Cyberinfrastructure”.22

![Figure](image-url)

**Figure.** (Left panel) Multiscale MD/QM simulation of the reaction of water at a crack tip in silicon (top), on a Grid of distributed PC clusters in Japan and the US (bottom). (Right panel) Time chart (bottom) of an adaptive QM/MD simulation performed on globally distributed parallel supercomputers in the US (USC, PSC, NCSA) and Japan (AIST, Univ. of Tokyo, TITEch). The blue line denotes the execution of QM simulation, the red line shows the failure in the initialization phase, and the light blue line the failure in the simulation phase.

Such a multidisciplinary application is emerging at the forefront of computational sciences. The multiscale simulation embeds accurate quantum mechanical (QM) calculations to handle chemical reactions within a molecular dynamics (MD) simulation to describe large-scale atomistic processes. CACS scientists have performed a preliminary MD/QM simulation on a Grid of distributed PC clusters in Japan and the US (see the figure below), in collaboration with Japanese scientists.23 More recently, we have performed a larger Grid MD/QM simulation on a Grid consisting of 6 supercomputing centers in the US (USC and two NSF TeraGrid nodes at the Pittsburgh Supercomputing Center and the National
Center for Supercomputing Applications) and Japan (National Institute of Advanced Industrial Science and Technology, University of Tokyo, and Tokyo Institute of Technology). The simulation was sustained autonomously on ~700 processors for 2 weeks, involving in total of 150,000 CPU-hours, where the number of processors changed dynamically on demand and computations were migrated automatically according to both reservations and unexpected faults. A recent trend in distributed computing is “cloud”, which has gained popularity in commerce and will be covered briefly in the class.

References


