LECTURE 1 23.3.2006

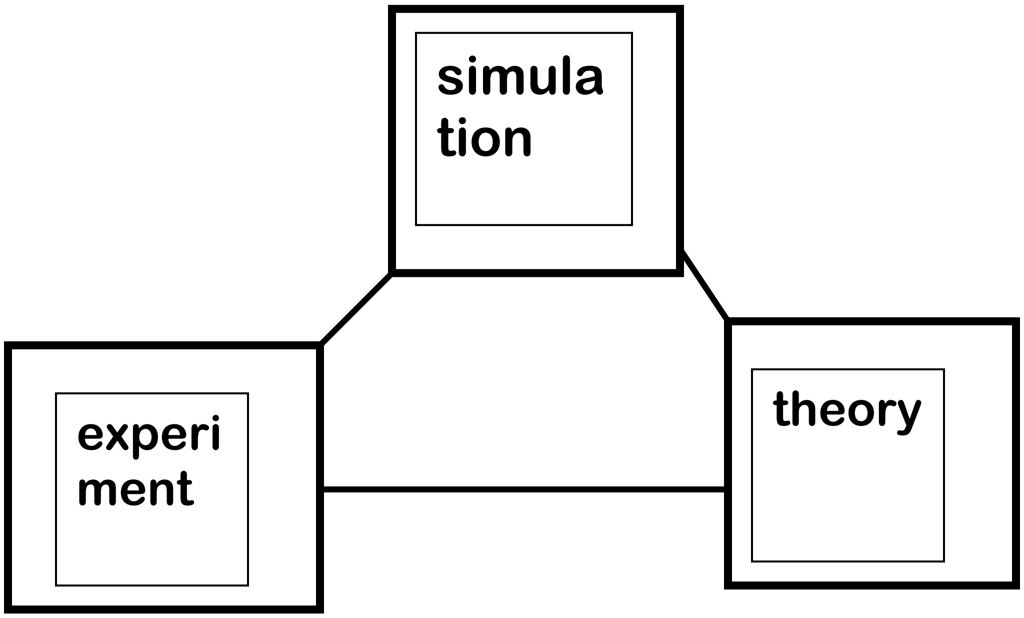
A BRIEF HISTORY OF ATOMISTIC SIMULATIONS

- 1953: The first simulation of a liquid (Metropolis et al, Los Alamos)
  → Metropolis Monte Carlo algorithm, J Chem Phys 21, 1087 (1953)
- 1957: Monte Carlo simulation of a "Lennard-Jones liquid" , Wood and Parker, J Chem Phys 27, 720 (1957)
- 1957: The first dynamical simulation by solving Newton equations of motion for hard spheres, Alder, Wainwright J Chem Phys 27, 1208 (1957).
- 1964: Dynamical simulation of argon liquid (Lennard-Jones potential), Rahman, Phys. Rev. 136A, 405 (1964) → Molecular dynamic simulation method, MD

MD (and the related force-field molecular mechanics) is a very versatile tool to study structural and dynamical properties of:

- Large molecules, molecular systems, proteins, drug design…
- Condensed matter in solid and liquid states
- Quark-gluon plasma, nuclear reactions, motion of planets…
- Any system where you can reasonably know (or derive) basic interactions between the constituents of your system
- Mesoscopic (coarse-grained) dynamics: system consists of quasiparticles

MD is a standard tool of computational science, which is now an established *third* tool to observe & understand Nature!



FUNDAMENTALS (see excellent discussion by Haile, Chapter 2)

- **Molecular dynamics = calculating trajectories of N particles in phase space $\{r^N , p^N \}$ over observation period of $\mathcal{T}$. The dynamics may be described by Newtonian, Hamiltonian or Lagrangean equations of motion. Interest may be in (a) evaluating time averages (hopefully also ensemble averages) or (b) studying the dynamical behavior of the system**
- **Example of visualization of 2-dimensional phase space: ODHO, Haile Fig. 2.2.**
- **Classification of dynamical systems:**
    1. **Recurrent trajectories vs. nonrecurrent trajectories (most comets)**
    2. **Recurrent: Hamiltonian vs. non-Hamiltonian (e.g. dissipative systems, friction).**
    3. **Hamiltonian: Integrable vs. non-integrable.**
    4. **Integrable: periodic vs. quasiperiodic**
    5. **Non-integrable: nonergodic, ergodic, mixing, K-flow, B-flow**
- **Example of periodic vs. quasiperiodic systems: Haile Fig. 2.4 2-dim HO with different initial conditions (quaisperiodic: Lissajous pendulum)**
- **Ergodicity: over a sufficiently long time, the phase space trajectory passes all the points (system samples "all possible" configurations). Real dynamic systems are not ergodic, but time averages over "sufficiently long" phase space trajectories equal ensemble averages (calculated e.g. by Monte Carlo method) $\leftarrow\rightarrow$ ergodicity hypothesis**
- **Deterministic Newtonian dynamics can create chaotic trajectories, see Haile Fig. 2.7 & 2.8.**
- **"Measured" value of quantity A from MD trajectory:**

$$A_m = t^{-1} \int_{t0}^{t0+t} A(r^N(\mathcal{T}) , p^N(\mathcal{T}))d\mathcal{T}$$

- **Time average**

$$< A > = \lim (t\rightarrow\infty)\ t^{-1} \int_{t0}^{t0+t} A(r^N(\mathcal{T}) , p^N(\mathcal{T}))d\mathcal{T}$$

- **A statistically good MD simulation allows one to assume**

$$A_m = < A >$$

- **Boundary conditions determine the system: from 0 periodic directions (cluster) to 3 (bulk matter)**

- **Conservation principles of isolated systems (microcanonical ensemble):**
    1. **Mass**
    2. **Total energy**
    3. **Linear momentum** $\mathcal{P} = \sum_i p_i (t) = \sum_i p_i (t) = \text{const}$
    4. **Angular momentum** $L = \sum_i r_i (t) \times p_i (t) = \text{const}$

PROJECT: BUILD A WORKING MD PROGRAM FOR LENNARD-JONES MATERIAL

- **C, Fortran or MATLAB O.K.**
- **Modify the Fortran sample (a working code shown in Lecture 1, http://www.openmp.org/drupal/samples/md.html) or build from scratch using bits and pieces shown in lectures**
- **Minimum requirement: code works properly and you have finished at least 2 exercises that will be given beginning of April.**
- **Documentation of the project and solutions to the exercises in the "learning diary"**

STRUCTURE OF THE CODE : See the sample code from OpenMP.org website

PROGRAMMING HINTS:

- **Modular code**
- **Comment program parameters, input & output variables, subroutines and main units (do-loops, if-then-else blocks)**
- **Test the code piece by piece**