So far, considered generic methods applicable to all differential equations

It may be useful to consider methods for specific restricted classes of equations. Two reasons:

1. Restricting to a particular class may simplify design of methods with some desirable property.

Example: Numerov's method for linear 2^{nd} order ODEs without 1^{st} derivative. A simple 4^{th} order method. Most useful for boundary value problems, and we'll consider it when we talk about them.

2. A particular class of equations may have some important properties, and it is desirable to preserve them even for a finite time step.

Newton's 2nd law equations for a conservative system

$$m_i \ddot{x} = F_i(\{x_j\}), \quad i = 1,...,M$$ *M*=3*N* for *N* particles in 3D.

$$F_i = -\frac{\partial U}{\partial x_i}$$ No dependence of *F* on the velocities or the time.

Hamiltonian $$H = \sum_{i=1}^{M} \frac{p_i^2}{2m_i} + U(\{x_i\}) \qquad \dot{x}_i = \frac{\partial H}{\partial p_i} = \frac{p_i}{m_i} \qquad \dot{p}_i = -\frac{\partial H}{\partial x_i} = -\frac{\partial U}{\partial x_i}$$

Important properties: conservation laws (energy, momentum, angular momentum), phase volume preservation (Liouville theorem).

For the harmonic oscillator:

Euler method: the amplitude grows with time and diverges. The phase volume diverges.

RK4: the amplitude decreases with time (much more slowly), eventually approaches 0. The phase volume shrinks to 0.

For the trapezoidal method, constant amplitude, but it is implicit.

If we can design a method conserving the phase volume even for a finite step, the trajectory should not diverge nor collapse.

Generally, expect such a method to conserve energy much more accurately.

Liouville theorem plays a huge role in stat. mech. The distribution function $\rho(\{p\},\{q\})$ is constant along the trajectories of the system in phase space. If we want to simulate a system and obtain some thermodynamic properties (molecular dynamics), need to preserve the phase volume.

For simplicity, consider the 1D case and put m=1. p=v

$$\dot{v} = F(x) \qquad \dot{x} = v$$

Euler method: for $\dot{\vec{y}} = \vec{f}(\vec{y})$, $\vec{y}_{n+1} = \vec{y}_n + \vec{f}(\vec{y})h$

In our case, $\vec{y} = (v, x)$, $\vec{f} = (F(x), v)$

$$v_{n+1} = v_n + F(x_n)h, \quad x_{n+1} = x_n + v_n h$$

Transformation $(v_n, x_n) \to (v_{n+1}, x_{n+1})$

Jacobian $$\begin{vmatrix} \frac{\partial v_{n+1}}{\partial v_n} & \frac{\partial v_{n+1}}{\partial x_n} \\ \frac{\partial x_{n+1}}{\partial v_n} & \frac{\partial x_{n+1}}{\partial x_n} \end{vmatrix} = \begin{vmatrix} 1 & F'(x_n)h \\ h & 1 \end{vmatrix} = 1 - F'(x_n)h^2 \neq 1$$

Preserved to 1st order, but no exact preservation. E.g., for the harmonic oscillator, F(x)=-kx (Hooke's law), and the volume grows exponentially.

Euler: $v_{n+1}=v_n+F(x_n)h\,,\quad x_{n+1}=x_n+v_n h$

Make a single modification: $\boxed{v_{n+1}=v_n+F(x_n)h\,,\quad x_{n+1}=x_n+\underline{v_{n+1}}\,h}$

$$x_{n+1}=x_n+v_n h+F(x_n)h^2$$

The difference between the old and the new x_{n+1} is only $O(h^2)$, so it is still a legitimate 1st order method.

Transformation $(v_n\,,x_n)\to(v_{n+1}\,,x_{n+1})$. Matrix T. Jacobian

$$|\mathrm{T}|=\begin{vmatrix}\dfrac{\partial v_{n+1}}{\partial v_n} & \dfrac{\partial v_{n+1}}{\partial x_n}\\ \dfrac{\partial x_{n+1}}{\partial v_n} & \dfrac{\partial x_{n+1}}{\partial x_n}\end{vmatrix}=\begin{vmatrix}1 & F'(x_n)h\\ h & 1+F'(x_n)h^2\end{vmatrix}=1+F'(x_n)h^2-F'(x_n)h^2=1$$

The volume is preserved **exactly**.

Semi-implicit Euler or Euler-Cromer method. Formally, implicit, because one component of the vector (v_{n+1},x_{n+1}) does appear on the right-hand side. But, of course, in practice everything can be calculated explicitly.

In fact, the Euler-Cromer method satisfies a more general property of Hamiltonian systems called symplecticity. In 1D equivalent to phase volume preservation, in higher dimensions phase volume preservation follows from it.

$$T^T \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} T = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$$

For the derivation of this property, see, e.g., Frenkel & Smit, Understanding molecular simulation, 2002, Sec. A.3.2.

For this reason, the Euler-Cromer algorithm is also called symplectic Euler. Belongs to the class of symplectic algorithms.

Euler-Cromer: $v_{n+1} = v_n + F(x_n)h, \quad x_{n+1} = x_n + v_{n+1}h$

Apply it backwards, going from n+1 to n.

$$v_n = v_{n+1} - F(x_{n+1})h, \quad x_n = x_{n+1} - v_n h$$

$$x_{n+1} = x_n + v_n h, \quad v_{n+1} = v_n + F(x_{n+1})h$$

This way get another 1st order symplectic algorithm.

$$v_{n+1}=v_n+F(x_n)h\,,\quad x_{n+1}=x_n+v_{n+1}h \qquad \text{(algorithm 1)}$$

$$x_{n+1}=x_n+v_nh\,,\quad v_{n+1}=v_n+F(x_{n+1})h \qquad \text{(algorithm 2)}$$

Let us alternate between the two algorithms.

.....

$$v_{n+1}=v_n+F(x_n)h$$

$$x_{n+1}=x_n+v_{n+1}h$$

$$x_{n+2}=x_{n+1}+v_{n+1}h \qquad \Rightarrow \quad x_{n+2}=x_n+2\,v_{n+1}h$$

$$v_{n+2}=v_{n+1}+F(x_{n+2})h$$

$$v_{n+3}=v_{n+2}+F(x_{n+2})h \qquad \Rightarrow \quad v_{n+3}=v_{n+1}+2\,F(x_{n+2})h$$

.....

New scheme:

$$v_{n+1}=v_{n-1}+2\,F(x_n)h$$

$$x_{n+2}=x_n+2\,v_{n+1}h$$

or, after 2h→h

$$v_{n+1/2}=v_{n-1/2}+F(x_n)h$$

$$x_{n+1}=x_n+v_{n+1/2}h$$

Leapfrog algorithm. Staggered grid. Obviously, symplectic as well.

$$x_n = x_{n-1} + v_{n-1/2} h \;\Rightarrow\; x_n - x_{n-1} = v_{n-1/2} h$$

$$v_{n+1/2} = v_{n-1/2} + F(x_n) h \;\Rightarrow\; v_{n+1/2} - v_{n-1/2} = F(x_n) h$$

$$x_{n+1} = x_n + v_{n+1/2} h \;\Rightarrow\; x_{n+1} - x_n = v_{n+1/2} h$$

Subtract the 1st line from the 3rd:

$$x_{n+1} - 2x_n + x_{n-1} = (v_{n+1/2} - v_{n-1/2}) h = F(x_n) h^2$$

$x_{n+1} = 2x_n - x_{n-1} + F(x_n) h^2$ Verlet algorithm. A multipoint scheme

Equivalent to leapfrog, likewise symplectic.

Note that $\frac{x_{n+1} - 2x_n + x_{n-1}}{h^2} = \ddot{x} + O(h^2)$. So $x_{n+1} = 2x_n - x_{n-1} + F(x_n) h^2 + O(h^4)$

It is tempting to say that the global error is $O(h^3)$, but for multipoint schemes such considerations do not work.

$$E_{n+1} = 2E_n - E_{n-1} + Ch^4 \;\Rightarrow\; E_{n+1} - E_n = E_n - E_{n-1} + Ch^4 \;\Rightarrow\; E_{n+1} - E_n \approx Ch^4 n$$

$E_N \sim h^4 N^2 \sim h^2$ The method is quadratic, which is still better than

Euler-Cromer. Also, time-reversal invariant (Euler-Cromer is not).

$$x_{n+1} = 2x_n - x_{n-1} + F(x_n)h^2$$

Does not involve velocities. If they are needed, just use centred difference:

$$v_n = \frac{x_{n+1} - x_{n-1}}{2h}$$

Not self-starting: initially, need x_0 and x_1, rather than x_0 and v_0.

$$x_1 = x_0 + v_0 h + F(x_0)h^2/2 + O(h^3)$$

For leapfrog $\left(v_{n+1/2} = v_{n-1/2} + F(x_n)h\,,\quad x_{n+1} = x_n + v_{n+1/2}h\right)$

the velocities and the positions are not given at the same time.
Complicates both starting and calculating, e.g., total energy.

It is possible to derive yet another equivalent algorithm.

.....

$$v_{n+1} = v_n + F(x_n)h$$

$$x_{n+1} = x_n + v_{n+1}h$$

$$x_{n+2} = x_{n+1} + v_{n+1}h$$

$$v_{n+2} = v_{n+1} + F(x_{n+2})h$$

$$v_{n+3} = v_{n+2} + F(x_{n+2})h$$

.....

For leapfrog, combined these pairs

$$x_{n+2} = x_{n+1} + v_{n+1}h = x_n + 2v_{n+1}h = x_n + 2v_n h + 2F(x_n)h^2$$

$$v_{n+2} = v_{n+1} + F(x_{n+2})h = v_n + [F(x_n) + F(x_{n+2})]h$$

After 2h→h

$$x_{n+1} = x_n + v_n h + F(x_n)h^2/2 \qquad v_{n+1} = v_n + [F(x_n) + F(x_{n+1})]h/2$$

Velocity Verlet. Generally preferable to the other two: self-starting, x and v calculated at the same time. Also, smaller round-off error than original Verlet.

Other ways to derive the Verlet algorithm. Variational method.

Least action principle: the trajectory of the system between given points in configuration space corresponds to the extremum of the action

$$S=\int_{t_0}^{t} L\left(x(t'),\dot{x}(t')\right)dt'=\int_{t_0}^{t}\frac{\dot{x}(t')^2}{2}dt'-\int_{t_0}^{t}U\left(x(t')\right)dt'\approx$$

$$\sum_{i=0}^{N-1}\frac{(x_{i+1}-x_i)^2}{2h}-\sum_{i=0}^{N-1}\frac{1}{2}\left[U(x_i)+U(x_{i+1})\right]h$$

Midpoint rule for the derivatives, trapezoidal rule for U.

$$\frac{\partial S}{\partial x_i}=\frac{1}{h}\left[-(x_{i+1}-x_i)+(x_i-x_{i-1})\right]-\frac{dU(x_i)}{dx_i}h=0 \;\Rightarrow\; -x_{i+1}+2x_i-x_{i-1}+h^2F(x_i)=0$$

Yet another approach is based on expanding the Liouville operator of the evolution of the system: $(p,x)=e^{iLt}(p_0,x_0)$. $L=L_x+L_p$.

$$iL_x=\dot{x}\frac{d}{dx},\quad iL_p=\dot{p}\frac{d}{dp} \qquad\qquad e^{iL_xt+iL_pt}\approx e^{iL_pt/2}e^{iL_xt}e^{iL_pt/2}$$

Frenkel and Smit, Understanding Molecular Simulation, 2002, Sec. 4.3.3.

Higher-order methods can be derived. Particularly useful in accelerator physics.

Verlet algorithm first used for molecular dynamics simulations by Verlet in 1967. Størmer (also spelled Störmer) used a similar method in 1907, so it is sometimes called Størmer-Verlet. Also, used by Delambre (1792). Interestingly, it was also used by Newton to prove Kepler's second law (reproduced by Feynman in Character of Physical Law).

http://www.youtube.com/watch?v=kd0xTfdt6qw (around 18:00)

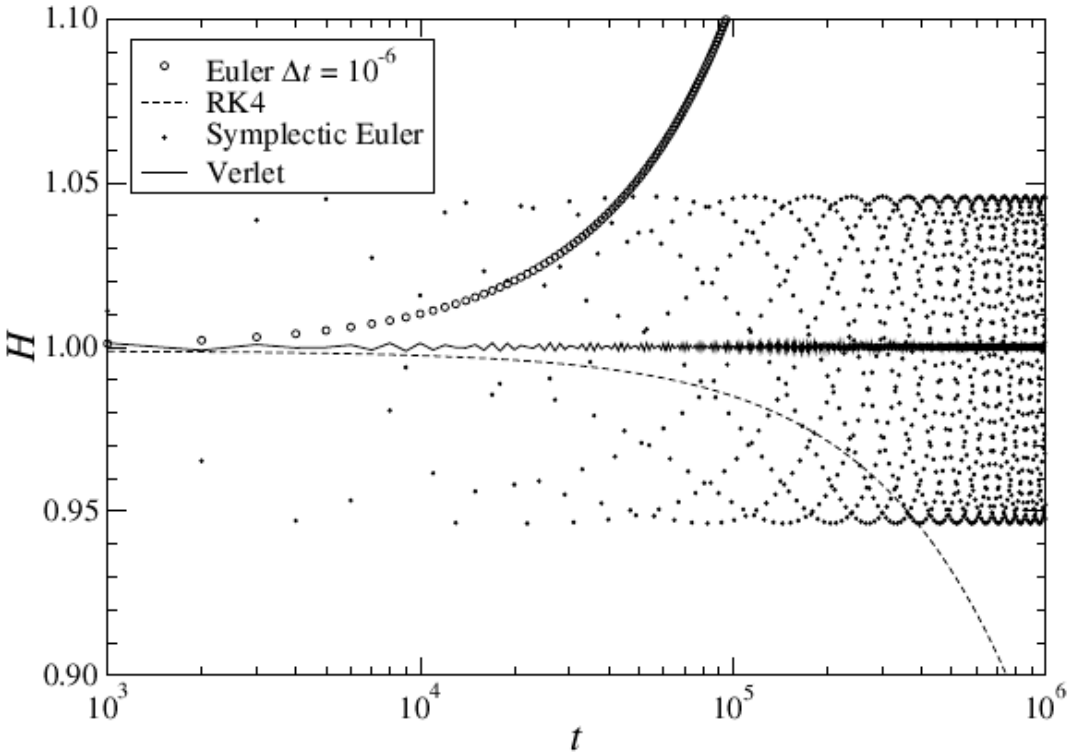
$$x_{n+1} = 2x_n - x_{n-1} + F(x_n)h^2$$

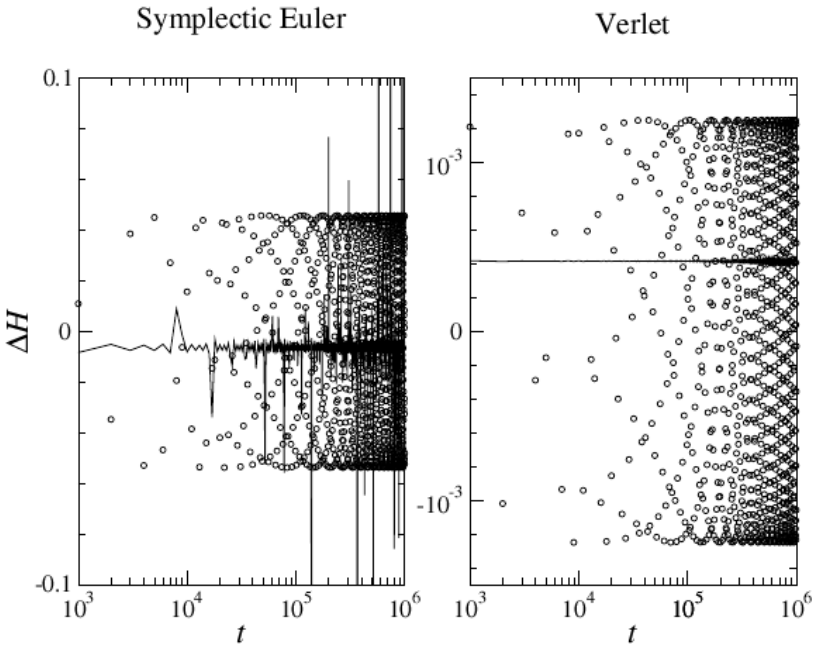
Energy conservation

Verlet algorithm does not conserve energy. However, the energy remains bounded for a very long time, with only an extremely slow drift. For an analytic Hamiltonian,

$$|H(\vec{p}_n, \vec{x}_n) - H(\vec{p}_{0,} \vec{x}_0)| \leq Ch^2 + C_0 e^{-c/h} t, \quad t < e^{c/h}$$

Aside from the slow drift, H oscillates, but there is a “shadow Hamiltonian” H' that remains constant.





http://www.che.ufl.edu/ladd/publications/kmz_09.pdf

Molecular dynamics

A widely used method for studying classical many-particle systems. Can study both equilibrium properties (e.g., equation of state of a non-ideal gas) and dynamics (e.g., motion of a charged polymer in an electric field).

The particles can represent atoms or molecules, but they can also correspond to larger units (coarse-graining).

Take a system of interacting particles, simulate their motion by solving the equations of motion numerically, collect statistics. Usually needs to be simulated over a sufficiently long time both to equilibrate and to collect enough statistics, so using a symplectic method like Verlet algorithm is essential.

Interaction forces

$$U = U\left(\{\vec{x}_i\}\right)$$

$$U\left(\{\vec{x}_i\}\right) = \sum_{i=1}^{N} U^{(1)}\left(\vec{x}_i\right) + \sum_{i=1}^{N}\sum_{j=1,\, j\neq i}^{N} U^{(2)}\left(\vec{x}_i, \vec{x}_j\right) + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1, i\neq j\neq k}^{N} U^{(3)}\left(\vec{x}_i, \vec{x}_j, \vec{x}_k\right) + \ldots$$

1-, 2-, 3-, ...,-body forces.

If all particles interact with all particles, then the computational cost of evaluating the n-body term is ~N^n. But often only close particles interact sufficiently strongly.

Where do we get the forces?

1) Theoretically (e.g., Coulomb interaction);

2) Computationally, using a quantum mechanical method (e.g., DFT);

3) Parametrize and fit to experimental data;

4) Just choose something reasonable (and easy computationally).

1-body: a charged particle in an external electric field

2-body: Coulomb interaction between charged particles (~1/r), dipole-dipole interaction (~$1/r^3$), etc.

Even for neutral particles with all multipole moments zero, there is always so-called dispersion interaction. Quantum fluctuations of the dipole moment of one atom induce dipole moments ~$1/r^3$ in neighbouring atoms. Then the potential is ~$(1/r^3)(1/r^3)$~$1/r^6$

Repulsion at short distances, when electron clouds overlap.

Lennard-Jones (LJ) potential $$U(r) = 4U_0\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

No justification for power 12.

If attraction is weak, use shifted LJ with a cutoff (Weeks-Chandler-Andersen)

Bond-stretching interaction for covalently bonded atoms: harmonic, anharmonic springs. Finitely extensible nonlinear elastic (FENE) interaction -

$$U(r) = -\frac{1}{2}kr_0^2 \ln\left(1 - \frac{r^2}{r_0^2}\right)$$

3-body: bond-bending (angular) forces

4-body: dihedral (or torsion) forces

Simulations often limited to small system sizes. Severe finite size effects if the system has open boundaries or is enclosed in a container. A better choice: periodic boundary conditions (PBC). It is assumed that the system is repeated periodically in all directions. In principle, it means that every particle should interact with an infinite number of particles in all “images”. In practice many interactions (not electrostatic) are sufficiently short-range that minimum image convention can be imposed: only particles in the same cell and their images in neighbouring cells are taken into account.

Truncate the interactions and shift so the potential is continuous. Still, derivatives are discontinuous. A reason to prefer relatively low-order schemes: higher-order schemes assume that there are many continuous derivatives. Higher accuracy not guaranteed.

Even with the minimum image convention, the number of pairs scales as N^2. Ideally, only consider particles within the cutoff distance of the interaction.

Use cell lists, or neighbour lists, or both.

Cell lists: divide the simulation box into cells of size greater than or equal to the cutoff distance. Store lists of particles in each cell. For a given particle, only its interactions with particles in the same cell and neighbouring cells are considered. Update as necessary.

Neighbour list: for each particle, create a list of particles within distance *d* somewhat larger than the cutoff distance. Update when a particle moves by distance larger than d minus the cutoff distance.

Can be combined: use cell lists to create neighbour lists.

Coulomb interaction

Long range – cannot be cut off. Even minimum image convention is undesirable. Generally, the Coulomb sum $\Sigma q_i/r_{ij}$ is only conditionally convergent.

3 classes of methods: Ewald summation [$O(N^{3/2})$], particle-mesh (aka particle-in-cell) [O(NlogN)] and fast multipole [O(N)]. Despite scaling, particle-mesh is often faster than fast multipole and easier to implement.

Quantities that depend explicitly on the coordinates and velocities can be found easily. E.g., the energy.

Pressure can be obtained from the virial theorem: in d dimensions with 2-body interactions only,

$$2\langle K\rangle = -\sum_i \langle \vec{F}_i \dot{\vec{r}}_i \rangle \;\Rightarrow\; P = nk_B T + \frac{1}{dV}\left\langle \sum_{i<j} \vec{F}(\vec{r}_{ij})\cdot\vec{r}_{ij} \right\rangle$$

Temperature can be calculated from equipartition (*f* d.o.f.):

$$k_B T = \frac{2\langle K\rangle}{f}$$

Quantities like the entropy and the free energy cannot be obtained directly. Thermodynamic integration.

$$P = -\left(\frac{\partial F}{\partial V}\right)_{N,T} \;\Rightarrow\; F(T,V_1) = F(T,V_0) - \int_{V_0}^{V_1} P(T,V)\,dV$$

$$U_\lambda = U_A + \lambda(U_B - U_A) \;\Rightarrow\; F_B - F_A = \int_0^1 \lambda \left\langle \frac{\partial U}{\partial \lambda} \right\rangle_\lambda$$

Various ensembles

MD simulations are done at (approximately) constant energy, and therefore correspond to the microcanonical ensemble. Experimental systems are often coupled to a heat bath (or a thermostat) and correspond to the canonical ensemble. The difference is particularly significant for small systems.

To simulate the canonical ensemble, need to couple to some sort of “thermostat”. But need this to be computationally efficient. One possibility is coupling to noise (Langevin dynamics). But there are also deterministic algorithms. Nosé-Hoover thermostat. Introduces one additional degree of freedom that rescales time.

Can also simulate at constant pressure by making the volume a dynamic variable.