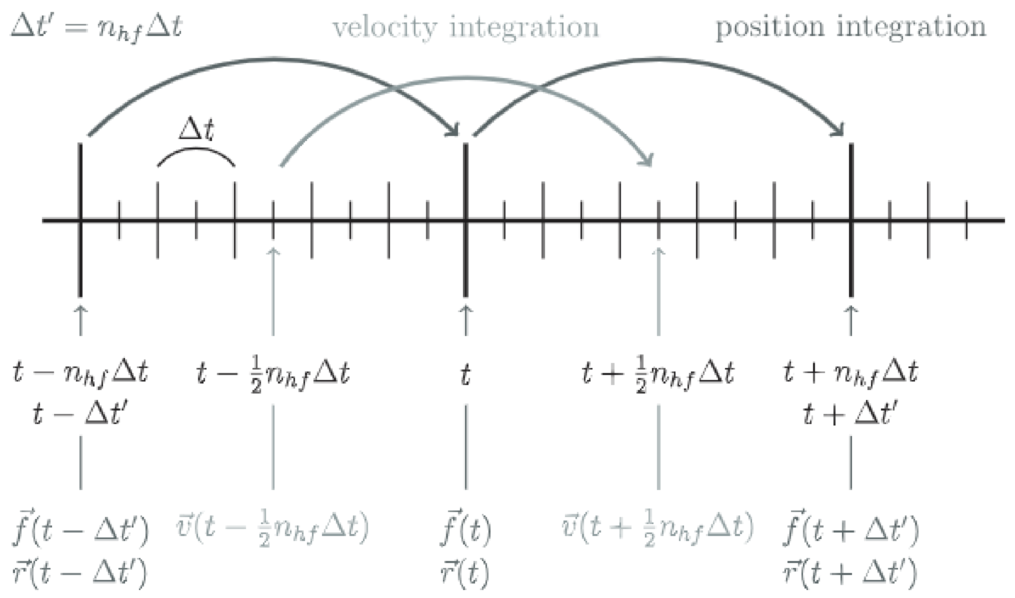
Integration Algorithm



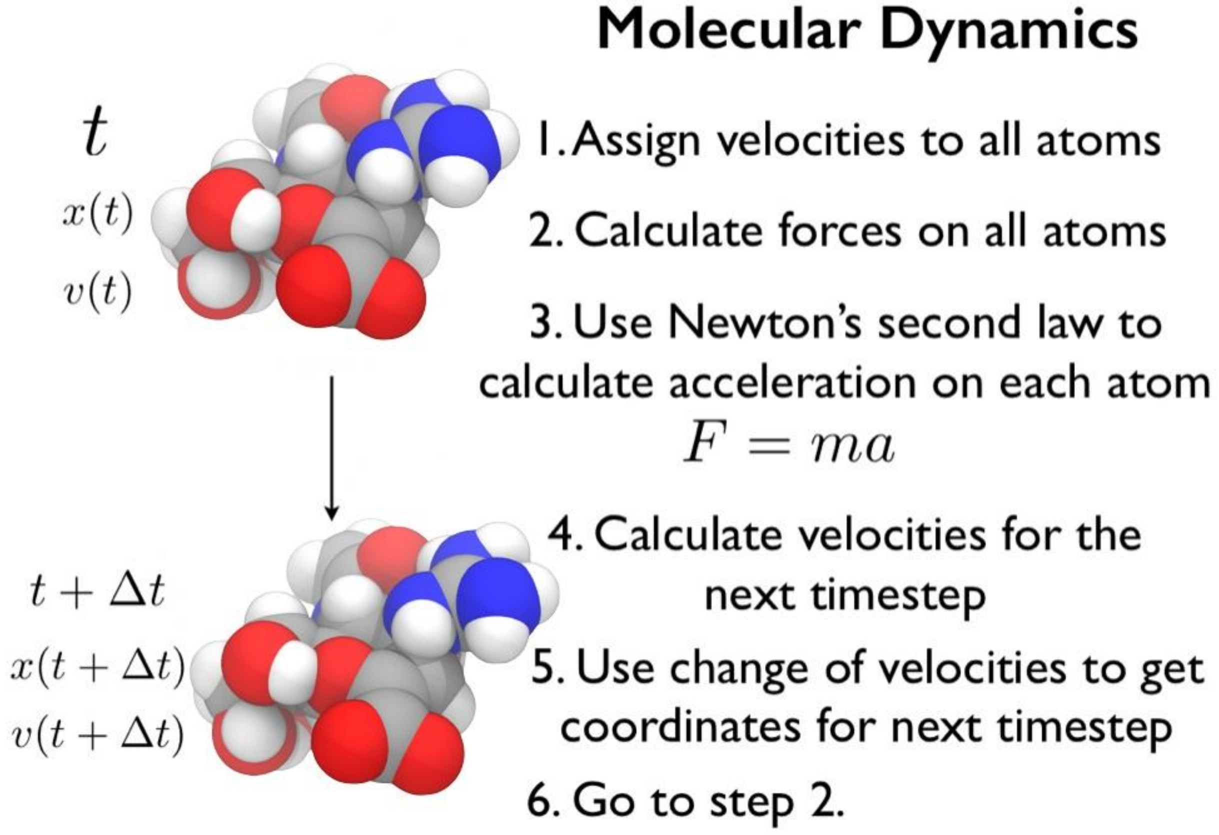
2024 Winter Seminar

January 12th, 2024

Sangmin Lee

Content

Yes we solve F=ma in classical MD simulation



Molecular Dynamics

1. Assign velocities to all atoms
2. Calculate forces on all atoms
3. Use Newton's second law to calculate acceleration on each atom

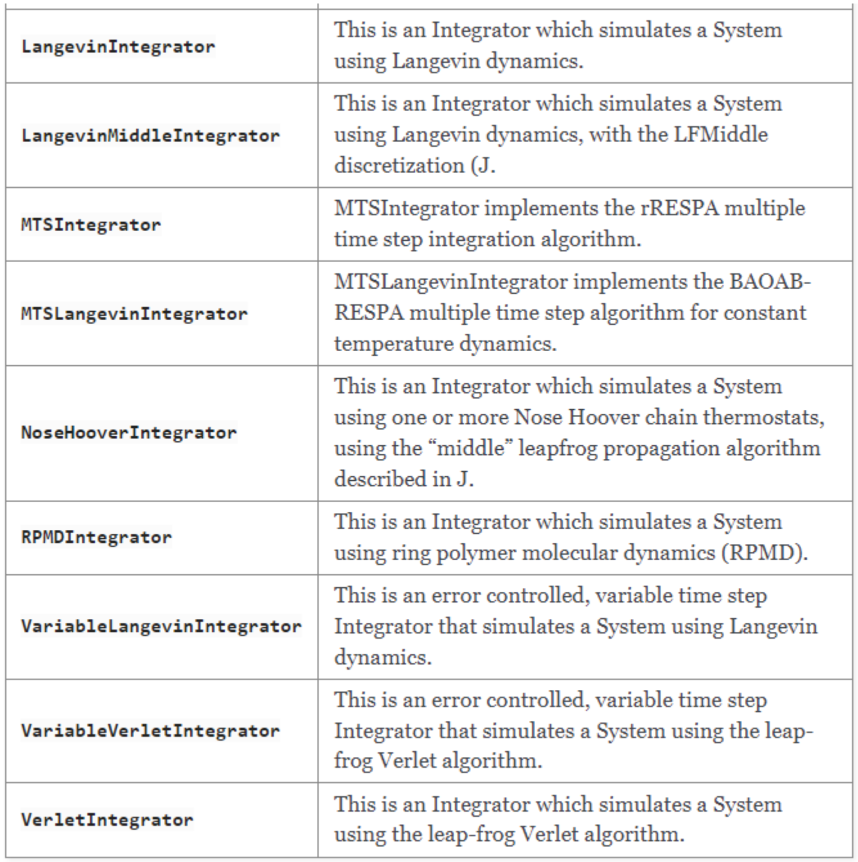
$$F = ma$$

4. Calculate velocities for the next timestep
5. Use change of velocities to get coordinates for next timestep
6. Go to step 2.

But there are much more things that we should understand here!

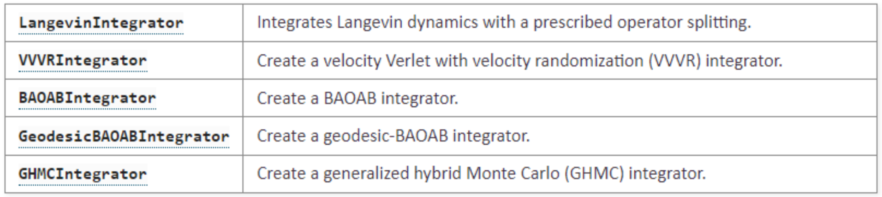
The list of integrator implemented in OpenMM

In OpenMM,

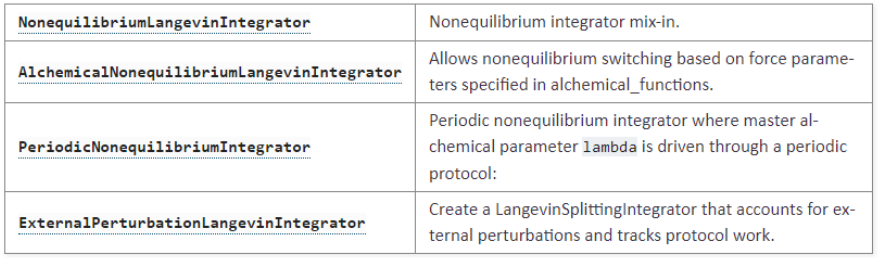


`LangevinIntegrator`	This is an Integrator which simulates a System using Langevin dynamics.
`LangevinMiddleIntegrator`	This is an Integrator which simulates a System using Langevin dynamics, with the LFMiddle discretization (J.
`MTSIntegrator`	MTSIntegrator implements the rRESPA multiple time step integration algorithm.
`MTSLangevinIntegrator`	MTSLangevinIntegrator implements the BAOAB-RESPA multiple time step algorithm for constant temperature dynamics.
`NoseHooverIntegrator`	This is an Integrator which simulates a System using one or more Nose Hoover chain thermostats, using the "middle" leapfrog propagation algorithm described in J.
`RPMDIntegrator`	This is an Integrator which simulates a System using ring polymer molecular dynamics (RPMD).
`VariableLangevinIntegrator`	This is an error controlled, variable time step Integrator that simulates a System using Langevin dynamics.
`VariableVerletIntegrator`	This is an error controlled, variable time step Integrator that simulates a System using the leap-frog Verlet algorithm.
`VerletIntegrator`	This is an Integrator which simulates a System using the leap-frog Verlet algorithm.

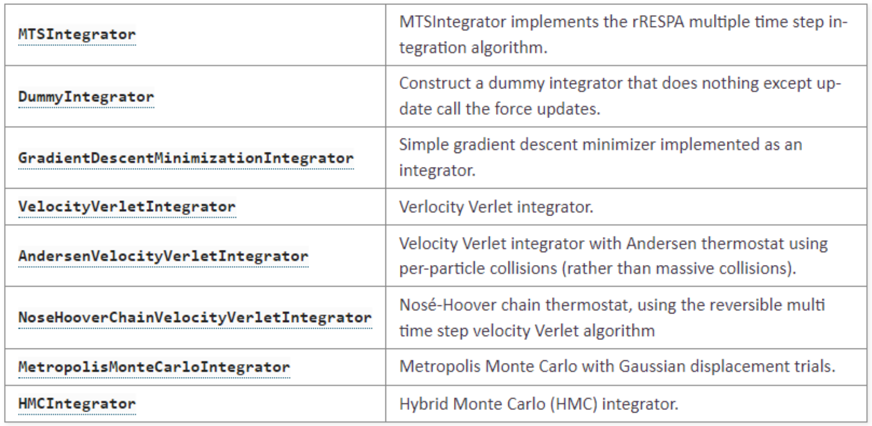
In OpenMMtools (https://openmmtools.readthedocs.io/en/stable/integrators.html),



`LangevinIntegrator`	Integrates Langevin dynamics with a prescribed operator splitting.
`VVVRIntegrator`	Create a velocity Verlet with velocity randomization (VVVR) integrator.
`BAOABIntegrator`	Create a BAOAB integrator.
`GeodesicBAOABIntegrator`	Create a geodesic-BAOAB integrator.
`GHMCIntegrator`	Create a generalized hybrid Monte Carlo (GHMC) integrator.



`NonequilibriumLangevinIntegrator`	Nonequilibrium integrator mix-in.
`AlchemicalNonequilibriumLangevinIntegrator`	Allows nonequilibrium switching based on force parameters specified in alchemical_functions.
`PeriodicNonequilibriumIntegrator`	Periodic nonequilibrium integrator where master alchemical parameter `lambda` is driven through a periodic protocol:
`ExternalPerturbationLangevinIntegrator`	Create a LangevinSplittingIntegrator that accounts for external perturbations and tracks protocol work.



`MTSIntegrator`	MTSIntegrator implements the rRESPA multiple time step integration algorithm.
`DummyIntegrator`	Construct a dummy integrator that does nothing except update call the force updates.
`GradientDescentMinimizationIntegrator`	Simple gradient descent minimizer implemented as an integrator.
`VelocityVerletIntegrator`	Verlocity Verlet integrator.
`AndersenVelocityVerletIntegrator`	Velocity Verlet integrator with Andersen thermostat using per-particle collisions (rather than massive collisions).
`NoseHooverChainVelocityVerletIntegrator`	Nosé-Hoover chain thermostat, using the reversible multi time step velocity Verlet algorithm
`MetropolisMonteCarloIntegrator`	Metropolis Monte Carlo with Gaussian displacement trials.
`HMCIntegrator`	Hybrid Monte Carlo (HMC) integrator.

Basics of MD simulation

In classical mechanics, the equation of motion is integrated to generate the trajectory.

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{r}}_i}\right) - \frac{\partial \mathcal{L}}{\partial \mathbf{r}_i} = 0. \qquad \dot{q}_\alpha = \frac{\partial \mathcal{H}}{\partial p_\alpha}, \quad \dot{p}_\alpha = -\frac{\partial \mathcal{H}}{\partial q_\alpha}.$$

Energy is conserved in classical mechanics!

By solving F=ma, we sample the **microcanonical** ensemble (constant E) for ensemble averages.

$$\langle a \rangle = \frac{\int \mathrm{dx}\, a(\mathrm{x})\delta(\mathcal{H}(\mathrm{x}) - E)}{\int \mathrm{dx}\, \delta(\mathcal{H}(\mathrm{x}) - E)} = \lim_{\mathcal{T}\to\infty} \frac{1}{\mathcal{T}} \int_0^{\mathcal{T}} \mathrm{d}t\, a(\mathrm{x}_t) \equiv \bar{a}.$$

We need to sample the x_t in microcanonical ensemble, which we call it **trajectory**.(.dcd)

In here, we introduce the time discretization parameter dt, known as the **time step.**

Starting with the initial cond x_0, x_dt, x_2dt, x_3dt are generated by applying the integrator iteratively.

$$A = \langle a \rangle = \frac{1}{M} \sum_{n=1}^{M} a(\mathrm{x}_{n\Delta t}) \equiv \bar{a}.$$

Simple integration schemes

A discretization of the equations of motion can be obtained by Taylor expansion:

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\ \mathbf{v}_i(t) + \frac{\Delta t^2}{2m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{r}}_i(t) + \mathcal{O}(\Delta t^4)$$
$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\Delta t}{m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^2}{2}\ \ddot{\mathbf{v}}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{v}}_i(t) + \mathcal{O}(\Delta t^4).$$

By the **Euler algorithm**, the trajectory is calculated according to:

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\ \mathbf{v}_i(t) + \frac{\Delta t^2}{2m_i}\mathbf{f}_i(t) + \mathcal{O}(\Delta t^3)$$
$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\Delta t}{m_i}\ \mathbf{f}_i(t) + \mathcal{O}(\Delta t^2)$$

WE DO NOT USE THIS, since this is not stable (neither time-reversible nor **symplectic**)

Simple integration schemes

A discretization of the equations of motion can be obtained by Taylor expansion:

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\ \mathbf{v}_i(t) + \frac{\Delta t^2}{2m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{r}}_i(t) + \mathcal{O}(\Delta t^4)$$
$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\Delta t}{m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^2}{2}\ \ddot{\mathbf{v}}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{v}}_i(t) + \mathcal{O}(\Delta t^4).$$

By the **Verlet algorithm** (1967), the trajectory is calculated according to:

$$\mathbf{r}_i(t-\Delta t) = \mathbf{r}_i(t) - \Delta t\ \mathbf{v}_i(t) + \frac{\Delta t^2}{2m_i}\ \mathbf{f}_i(t) - \frac{\Delta t^3}{3!}\ \dddot{\mathbf{r}}_i(t) + \mathcal{O}(\Delta t^4).$$

The updating equation for the positions and velocities are:

$$\mathbf{r}_i(t+\Delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\Delta t) + \frac{\Delta t^2}{m_i}\mathbf{f}_i(t) + \mathcal{O}(\Delta t^4),$$
$$\mathbf{v}_i(t) = \frac{\mathbf{r}_i(t+\Delta t) - \mathbf{r}_i(t-\Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^3).$$

Ulf D. Schiller, *An overview of integration schemes for molecular dynamics simulations,* **2008**
Phys. Rev. ***1967***, 159, 98.

Simple integration schemes

A discretization of the equations of motion can be obtained by Taylor expansion:

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\ \mathbf{v}_i(t) + \frac{\Delta t^2}{2m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{r}}_i(t) + \mathcal{O}(\Delta t^4)$$
$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\Delta t}{m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^2}{2}\ \ddot{\mathbf{v}}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{v}}_i(t) + \mathcal{O}(\Delta t^4).$$

By the **Leap-frog algorithm**, the trajectory is calculated according to:

$$\mathbf{v}_i(t+\frac{\Delta t}{2}) = \mathbf{v}_i(t-\frac{\Delta t}{2}) + \frac{\Delta t}{m_i}\ \mathbf{f}_i(t),$$
$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\ \mathbf{v}_i(t+\frac{\Delta t}{2}).$$

The velocities are updated at half time steps and ‘leap’ ahead the positions as:

$$\mathbf{v}_i(t) = \frac{\mathbf{v}_i(t-\frac{\Delta t}{2}) + \mathbf{v}_i(\Delta t+\frac{\Delta t}{2})}{2}.$$

Simple integration schemes

A discretization of the equations of motion can be obtained by Taylor expansion:

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\ \mathbf{v}_i(t) + \frac{\Delta t^2}{2m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{r}}_i(t) + \mathcal{O}(\Delta t^4)$$
$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{\Delta t}{m_i}\ \mathbf{f}_i(t) + \frac{\Delta t^2}{2}\ \ddot{\mathbf{v}}_i(t) + \frac{\Delta t^3}{3!}\ \dddot{\mathbf{v}}_i(t) + \mathcal{O}(\Delta t^4).$$

By the **Velocity-Verlet algorithm** (1982), the trajectory is calculated according to:

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t\ \mathbf{v}_i(t) + \frac{\Delta t^2}{m_i}\ \mathbf{f}_i(t) + \mathcal{O}(\Delta t^3),$$
$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}(t) + \frac{\Delta t}{2m_i}\left(\mathbf{f}_i(t) + \mathbf{f}_i(t+\Delta t)\right) + \mathcal{O}(\Delta t^3).$$

This is very stable and has become the perhaps most widely used integration algorithm.

The Velocity-Verlet scheme is a **symplectic** integrator.

Ulf D. Schiller, *An overview of integration schemes for molecular dynamics simulations,* **2008**
J. Chem. Phys. ***1982***, 76, 637-649.

What properties should integration algorithms satisfy?

1. Stability

- Energy conservation
- Symplecticity
- Time reversibility

2. Efficiency

- Maximum permissible time step
- Constraint algorithm
- Hydrogen mass repartitioning
- Multi-step integration

3. Accuracy

- Configurational sampling
- Dynamical properties

What properties should integration algorithms satisfy?

1. Stability

- Energy conservation
- Symplecticity
- Time reversibility

2. Efficiency

- Maximum permissible time step
- Constraint algorithm
- Hydrogen mass repartitioning
- Multi-step integration

3. Accuracy

- Configurational sampling
- Dynamical properties

Classical time evolution operator and numerical integrators

It would be nice if we can derive the Verlet integration algorithm directly from the classical mechanics such as Hamilton's equation. Then '**symplecticity**' (the phase-space volume preserving property) is guaranteed.

For the time evolution of any function a(x) of the phase space vector,

$$\frac{\mathrm{d}a}{\mathrm{d}t} = \sum_{\alpha=1}^{3N} \left[\frac{\partial a}{\partial q_\alpha} \dot{q}_\alpha + \frac{\partial a}{\partial p_\alpha} \dot{p}_\alpha \right].$$

$$\dot{q}_\alpha = \frac{\partial \mathcal{H}}{\partial p_\alpha}, \qquad \dot{p}_\alpha = -\frac{\partial \mathcal{H}}{\partial q_\alpha}$$

$$\frac{\mathrm{d}a}{\mathrm{d}t} = \sum_{\alpha=1}^{3N} \left[\frac{\partial a}{\partial q_\alpha} \frac{\partial \mathcal{H}}{\partial p_\alpha} - \frac{\partial a}{\partial p_\alpha} \frac{\partial \mathcal{H}}{\partial q_\alpha} \right]$$

$$= \{a, \mathcal{H}\}.$$

Mark T. Tuckerman, *Statistical mechanics: Theory and Molecular Simulation 2thed*, **2023**

Classical time evolution operator and numerical integrators

Let's define the Liouville operator L as: $iLa = \{a, \mathcal{H}\}$

$$iL = \sum_{\alpha=1}^{3N} \left[\frac{\partial \mathcal{H}}{\partial p_\alpha} \frac{\partial}{\partial q_\alpha} - \frac{\partial \mathcal{H}}{\partial q_\alpha} \frac{\partial}{\partial p_\alpha} \right]$$

$$\mathrm{d}a/\mathrm{d}t = iLa$$

$$a(\mathrm{x}_t) = \mathrm{e}^{iLt} a(\mathrm{x}_0).$$

$$\mathrm{x}_t = \mathrm{e}^{iLt} \mathrm{x}_0.$$

We call the operator exp(iLt) as the classical propagator.

From now on, let's split the Liouville operator into two:

$$iL = iL_1 + iL_2,$$

$$\mathcal{H} = \frac{p^2}{2m} + U(x).$$

$$iL_1 = \sum_{\alpha=1}^{3N} \frac{\partial \mathcal{H}}{\partial p_\alpha} \frac{\partial}{\partial q_\alpha}$$

$$iL_1 = \frac{p}{m} \frac{\partial}{\partial x}, \qquad iL_2 = F(x) \frac{\partial}{\partial p},$$

$$iL_2 = -\sum_{\alpha=1}^{3N} \frac{\partial \mathcal{H}}{\partial q_\alpha} \frac{\partial}{\partial p_\alpha}.$$

Classical time evolution operator and numerical integrators

Let's define the Liouville operator L as: $iLa = \{a, \mathcal{H}\}$

$$iL = \sum_{\alpha=1}^{3N} \left[\frac{\partial \mathcal{H}}{\partial p_\alpha} \frac{\partial}{\partial q_\alpha} - \frac{\partial \mathcal{H}}{\partial q_\alpha} \frac{\partial}{\partial p_\alpha} \right]$$

$$\mathrm{d}a/\mathrm{d}t = iLa$$

$$a(\mathrm{x}_t) = \mathrm{e}^{iLt} a(\mathrm{x}_0).$$

$$\mathrm{x}_t = \mathrm{e}^{iLt} \mathrm{x}_0.$$

We call the operator exp(iLt) as the classical propagator.

From now on, let's split the Liouville operator into two:

$$iL = iL_1 + iL_2,$$

$$\mathcal{H} = \frac{p^2}{2m} + U(x).$$

$$iL_1 = \sum_{\alpha=1}^{3N} \frac{\partial \mathcal{H}}{\partial p_\alpha} \frac{\partial}{\partial q_\alpha}$$

$$iL_2 = -\sum_{\alpha=1}^{3N} \frac{\partial \mathcal{H}}{\partial q_\alpha} \frac{\partial}{\partial p_\alpha}.$$

$$iL_1 = \frac{p}{m} \frac{\partial}{\partial x}, \qquad iL_2 = F(x) \frac{\partial}{\partial p},$$

Mark T. Tuckerman, *Statistical mechanics: Theory and Molecular Simulation 2thed*, **2023**

Classical time evolution operator and numerical integrators

By the Trotter theorem, $\mathrm{e}^{A+B} = \lim_{P\to\infty} \left[\mathrm{e}^{B/2P}\mathrm{e}^{A/P}\mathrm{e}^{B/2P}\right]^P$

$$\mathrm{e}^{iLt} \approx \left[\mathrm{e}^{iL_2\Delta t/2}\mathrm{e}^{iL_1\Delta t}\mathrm{e}^{iL_2\Delta t/2}\right]^P + \mathcal{O}\left(P\Delta t^3\right)$$

For the Hamiltonian $\mathcal{H} = p^2/2m + U(x)$

$$\exp(iL\Delta t) \approx \exp\left(\frac{\Delta t}{2}F(x)\frac{\partial}{\partial p}\right)\exp\left(\Delta t\frac{p}{m}\frac{\partial}{\partial x}\right)\exp\left(\frac{\Delta t}{2}F(x)\frac{\partial}{\partial p}\right)$$

$$\begin{pmatrix} x(\Delta t) \\ p(\Delta t) \end{pmatrix} \approx \exp\left(\frac{\Delta t}{2}F(x)\frac{\partial}{\partial p}\right)\exp\left(\Delta t\frac{p}{m}\frac{\partial}{\partial x}\right)\exp\left(\frac{\Delta t}{2}F(x)\frac{\partial}{\partial p}\right)\begin{pmatrix} x \\ p \end{pmatrix}.$$

Classical time evolution operator and numerical integrators

$$\begin{pmatrix} x(\Delta t) \\ p(\Delta t) \end{pmatrix} \approx \exp\left(\frac{\Delta t}{2} F(x) \frac{\partial}{\partial p}\right) \exp\left(\Delta t \frac{p}{m} \frac{\partial}{\partial x}\right) \exp\left(\frac{\Delta t}{2} F(x) \frac{\partial}{\partial p}\right) \begin{pmatrix} x \\ p \end{pmatrix}.$$

$$\exp\left(\frac{\Delta t}{2} F(x) \frac{\partial}{\partial p}\right) \begin{pmatrix} x \\ p \end{pmatrix} = \begin{pmatrix} x \\ \\ p + \frac{\Delta t}{2} F(x) \end{pmatrix}$$

$$\exp\left(\Delta t \frac{p}{m} \frac{\partial}{\partial x}\right) \begin{pmatrix} x \\ \\ p + \frac{\Delta t}{2} F(x) \end{pmatrix} = \begin{pmatrix} x + \Delta t \frac{p}{m} \\ \\ p + \frac{\Delta t}{2} F\left(x + \Delta t \frac{p}{m}\right) \end{pmatrix}$$

$$\exp\left(\frac{\Delta t}{2} F(x) \frac{\partial}{\partial p}\right) \begin{pmatrix} x + \Delta t \frac{p}{m} \\ \\ p + \frac{\Delta t}{2} F\left(x + \Delta t \frac{p}{m}\right) \end{pmatrix}$$

$$= \begin{pmatrix} x + \frac{\Delta t}{m}\left(p + \frac{\Delta t}{2} F(x)\right) \\ \\ p + \frac{\Delta t}{2} F(x) + \frac{\Delta t}{2} F\left(x + \frac{\Delta t}{m}\left(p + \frac{\Delta t}{2} F(x)\right)\right) \end{pmatrix}.$$

Classical time evolution operator and numerical integrators

$$\exp\left(\frac{\Delta t}{2}F(x)\frac{\partial}{\partial p}\right)\begin{pmatrix} x+\Delta t\frac{p}{m} \\ p+\frac{\Delta t}{2}F\left(x+\Delta t\frac{p}{m}\right)\end{pmatrix}$$

$$=\begin{pmatrix} x+\frac{\Delta t}{m}\left(p+\frac{\Delta t}{2}F(x)\right) \\ p+\frac{\Delta t}{2}F(x)+\frac{\Delta t}{2}F\left(x+\frac{\Delta t}{m}\left(p+\frac{\Delta t}{2}F(x)\right)\right)\end{pmatrix}.$$

$$x(\Delta t)=x(0)+\Delta t v(0)+\frac{\Delta t^2}{2m}F(x(0))$$

$$v(\Delta t)=v(0)+\frac{\Delta t}{2m}\left[F(x(0))+F(x(\Delta t))\right]$$

This is just the update step in the Velocity Verlet algorithm!

Symplecticity is now guaranteed!

Classical time evolution operator and numerical integrators

For the N-particle systems: $\mathcal{H} = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + U(\mathbf{r}_1, ..., \mathbf{r}_N),$

$$iL = \sum_{i=1}^{N} \frac{\mathbf{p}_i}{m_i} \cdot \frac{\partial}{\partial \mathbf{r}_i} + \sum_{i=1}^{N} \mathbf{F}_i \cdot \frac{\partial}{\partial \mathbf{p}_i}.$$

Again, we can split the Liouville operator into two: $iL = iL_1 + iL_2,$

$$p(\Delta t/2) = p(0) + \frac{\Delta t}{2} F(x(0))$$

$$x(\Delta t) = x(0) + \frac{\Delta t}{m} p(\Delta t/2)$$

$$p(\Delta t) = p(\Delta t/2) + \frac{\Delta t}{2} F(x(\Delta t)).$$

$\Rightarrow$

$p = p + 0.5 * \Delta t * F$

$x = x + \Delta t * p/m$

Recalculate the force

$p = p + 0.5 * \Delta t * F.$

Symplecticity

Q. Why symplecticity is good to be guaranteed?

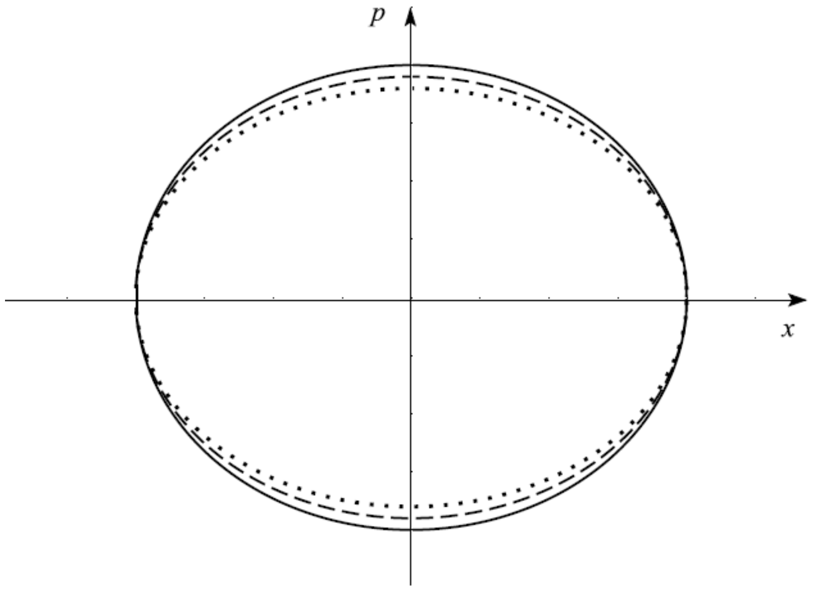
A. Sympletic integrator has the important property that there exists a shadow Hamiltonian that remains close to the true Hamiltonian and is exactly **conserved** by the algorithm.

Example: 1D Harmonic Oscillator

$$x(\Delta t) = x(0) + \Delta t \frac{p(0)}{m} - \frac{1}{2}\Delta t^2 \omega^2 x(0)$$

$$\mathcal{H}(x,p) = \frac{p^2}{2m} + \frac{1}{2} m\omega^2 x^2.$$

$$p(\Delta t) = p(0) - \frac{m\omega^2 \Delta t}{2}\left[x(0) + x(\Delta t)\right],$$



$$\tilde{\mathcal{H}}(x,p;\Delta t) = \frac{p^2}{2m(1-\omega^2\Delta t^2/4)} + \frac{1}{2} m\omega^2 x^2$$

What properties should integration algorithms satisfy?

1. Stability

- Energy conservation
- Symplecticity
- Time reversibility

2. Efficiency

- Maximum permissible time step
- Constraint algorithm
- Hydrogen mass repartitioning
- Multi-step integration

3. Accuracy

- Configurational sampling
- Dynamical properties

The maximum permissible time step

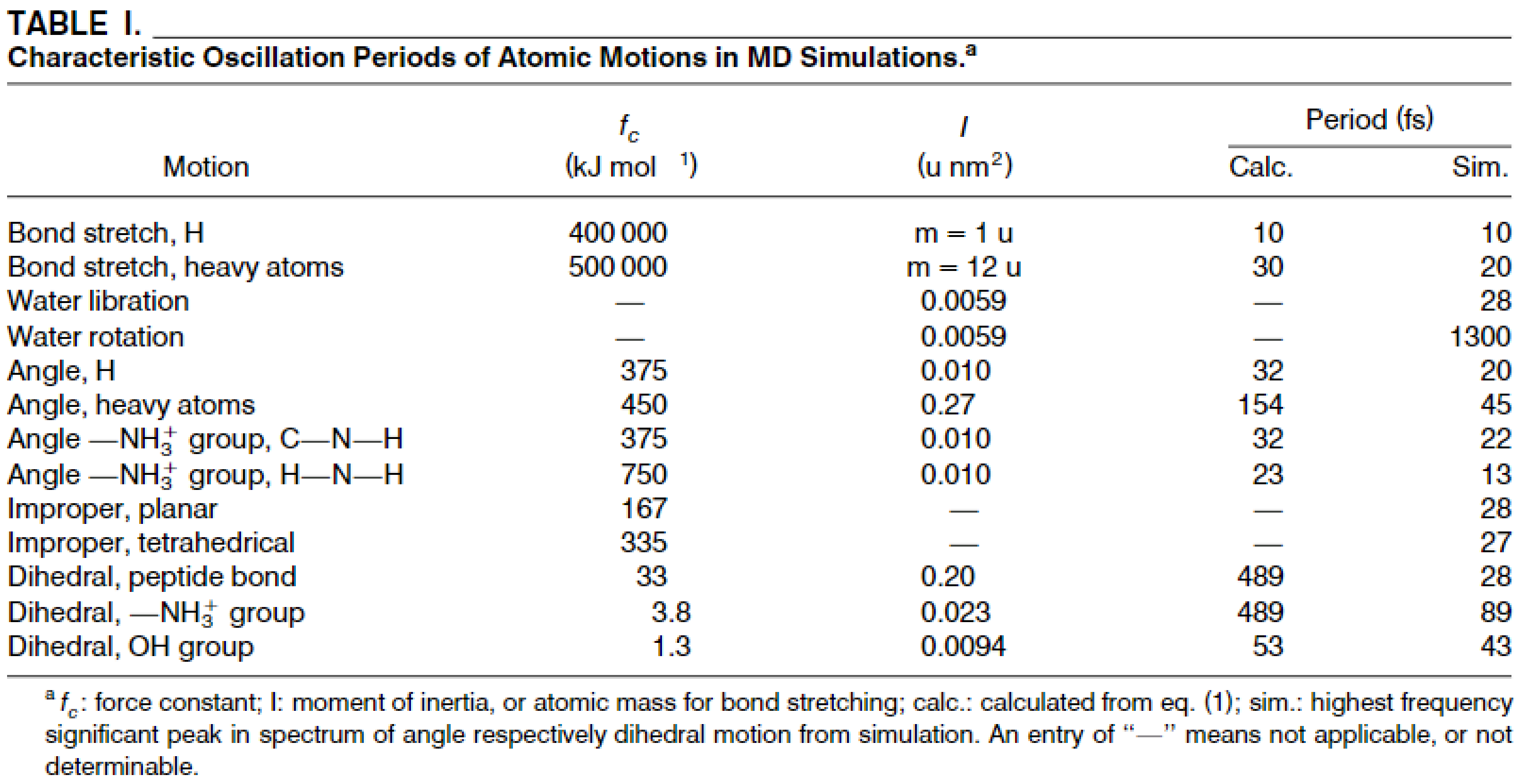


TABLE I.
Characteristic Oscillation Periods of Atomic Motions in MD Simulations.[a]

Motion	f_c (kJ mol ^{1})	I (u nm^2)	Period (fs)	
			Calc.	Sim.
Bond stretch, H	400 000	m = 1 u	10	10
Bond stretch, heavy atoms	500 000	m = 12 u	30	20
Water libration	—	0.0059	—	28
Water rotation	—	0.0059	—	1300
Angle, H	375	0.010	32	20
Angle, heavy atoms	450	0.27	154	45
Angle —NH_3^+ group, C—N—H	375	0.010	32	22
Angle —NH_3^+ group, H—N—H	750	0.010	23	13
Improper, planar	167	—	—	28
Improper, tetrahedrical	335	—	—	27
Dihedral, peptide bond	33	0.20	489	28
Dihedral, —NH_3^+ group	3.8	0.023	489	89
Dihedral, OH group	1.3	0.0094	53	43

[a] f_c: force constant; I: moment of inertia, or atomic mass for bond stretching; calc.: calculated from eq. (1); sim.: highest frequency significant peak in spectrum of angle respectively dihedral motion from simulation. An entry of "—" means not applicable, or not determinable.

The maximum time step in simulation is limited by the fastest motions which invariably involve **hydrogen** atoms, which is usually set to be **1 fs** for usual simulation.

Some simulation techniques are further required to increase the maximum time step, but how?

The idea of constraint

Rigid distance (or angle, proper angle, improper angle) **constraints** are used to increase the integration step size from **1 fs to 2 fs.**

The idea is to use **the method of Lagrange multipliers**.

$$\sigma_i(\{\mathbf{r}_k\}) = |\mathbf{r}_m - \mathbf{r}_n| - d_i$$

We apply a constraint force to atoms m and n, which produces a combined displacements δ along the constraint algorithm such that σ = 0 at the end of the time step.

This requires **iteratively** solving a system of nonlinear equations (i.e. Newton iteration)

$$\delta^{N+1} = \delta^{N} - \mathbf{J}^{-1}\boldsymbol{\sigma}^{N} \qquad \mathbf{J}_{ij} = \frac{\partial \sigma_i}{\partial \delta_j}$$

There are many constraint algorithms in MD simulation package which differentiates with the way to construct the jacobian matrix J and the way to perform the iteration.

List of algorithms: SHAKE, M-SHAKE, SHAPE, RATTLE, SETTLE, LINCS, P-LINCS, CCMA....

Constraint algorithm implemented in MD simulation package

In OpenMM,

SETTLE for water molecules only.

SHAKE for isolated clusters of one heavy atom with up to three hydrogens bonded to it.

CCMA for anything not handled by one of the above algorithms.

(LINCS is not implemented in OpenMM)

$$\delta^{N+1} = \delta^{N} - \mathbf{J}^{-1}\boldsymbol{\sigma}^{N} \qquad \mathbf{J}_{ij} = \frac{\partial \sigma_i}{\partial \delta_j}$$

Remark.

M-SHAKE constructs the jacobian matrix and then invert it, which is quite stable but high-cost.

LINCS circumvents the invertion of jacobian matrix by representing J^(-1) as a power series, but for strongly connected system, this series converges very slowly or even fail to converge.

SHAKE approximates J^(-1) using its upper triangle form for improved convergence at very little extra cost. This method is very popular but hard to be implemented efficiently on parallel architectures.

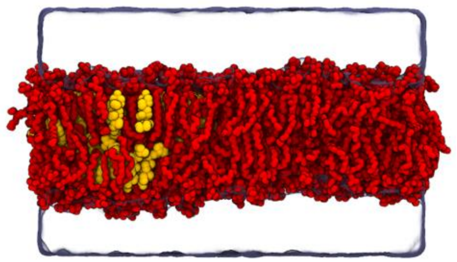
SETTLE uses an analytical solution for rigid water molecules rather than solving it iteratively. But only applicable to water molecules.

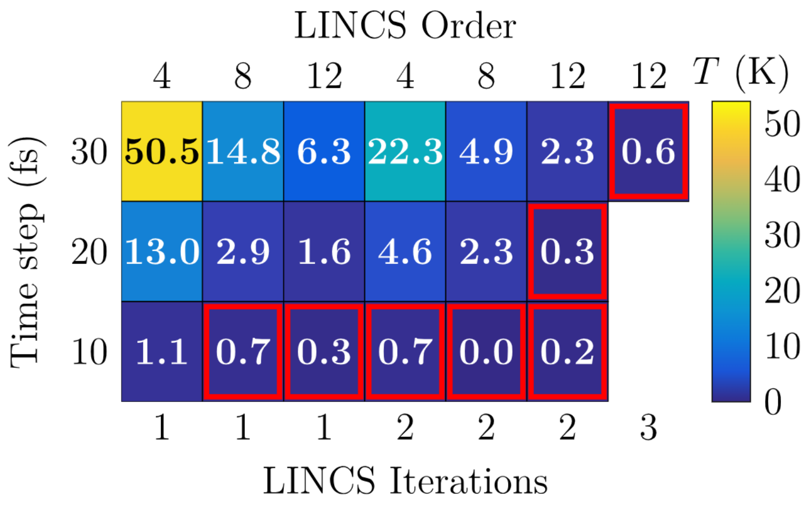
CCMA approximates J^(-1) with a different matrix K^(-1) that is easier to calculate on parallel architectures.

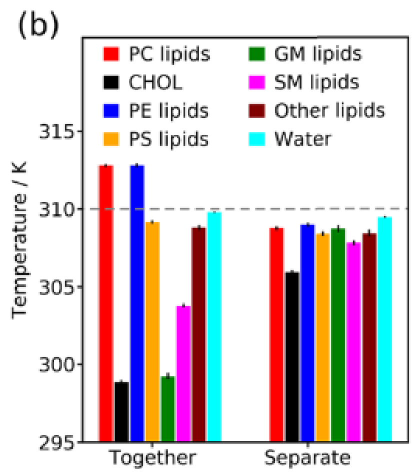
Convergence issues on constraint algorithm

Cholesterol in MARTINI force field is a notorious molecule due to highly coupled constraints.

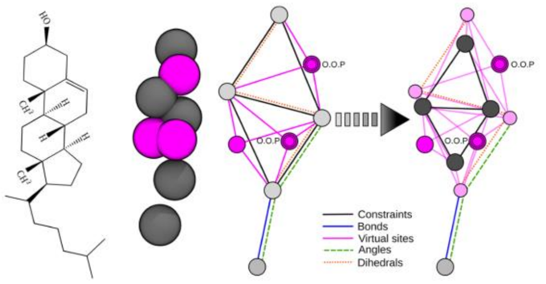
Applying **LINCS algorithm** with small LINCS Order and Iterations fails to be converged, which causes a negative energy drift and subsequently to cooling.

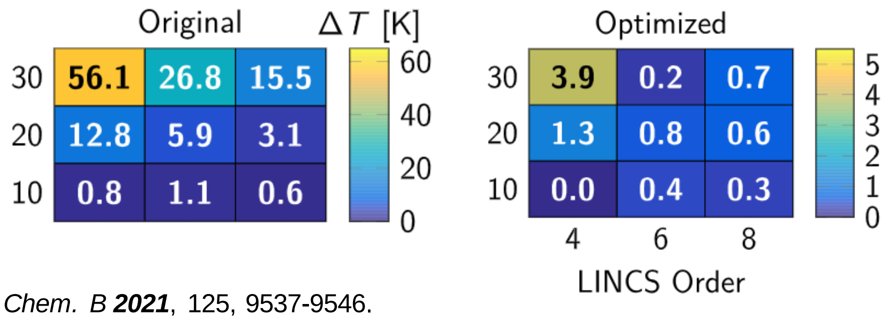






Such issue can be circumvented by additional incorporating the virtual sites in cholesterol.



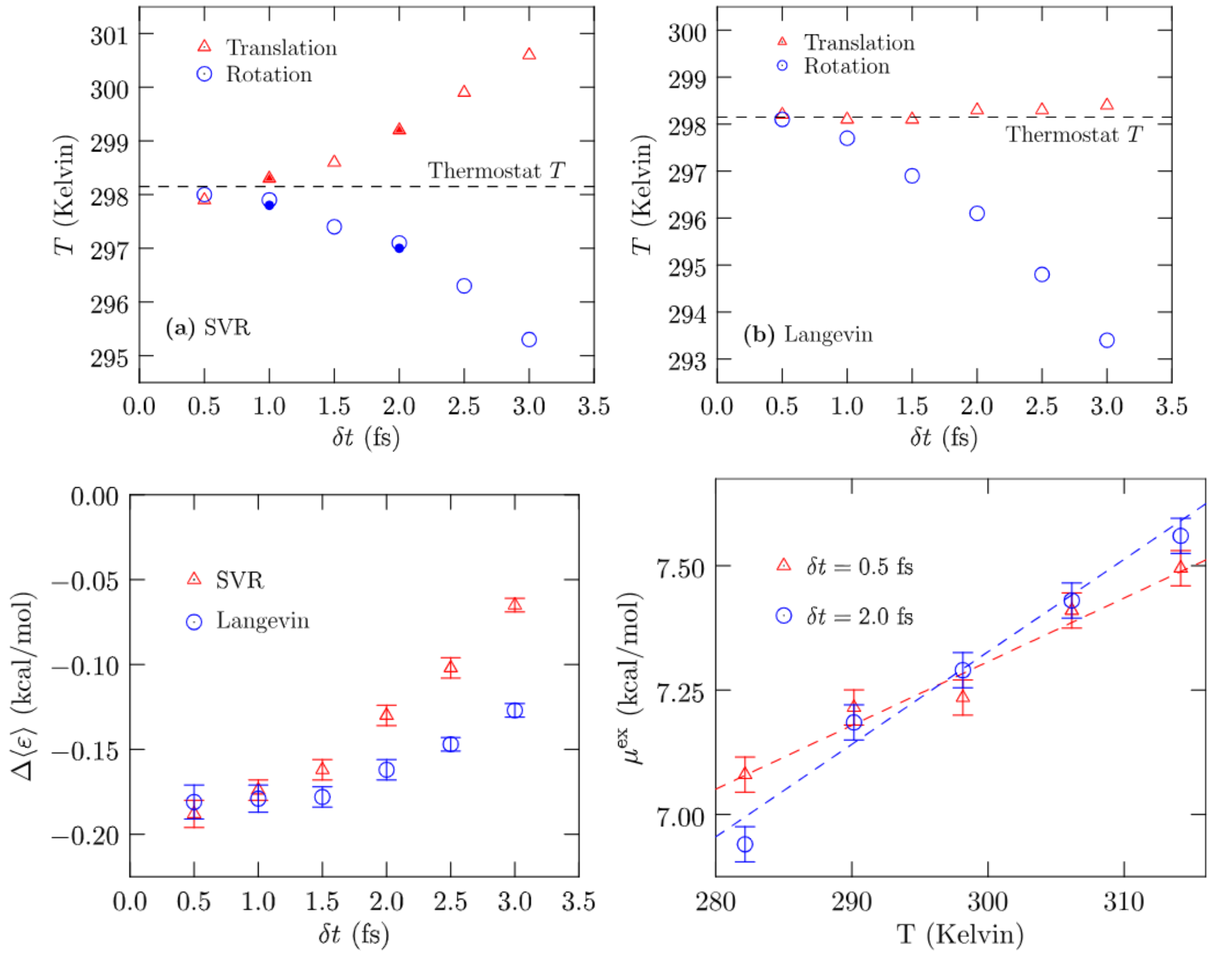


J. Phys. Chem. B ***2021***, 125, 9537-9546.
J. Chem. Theory Comput. ***2023,*** 19, 1592-1601.

Shot time step is sometimes inevitable

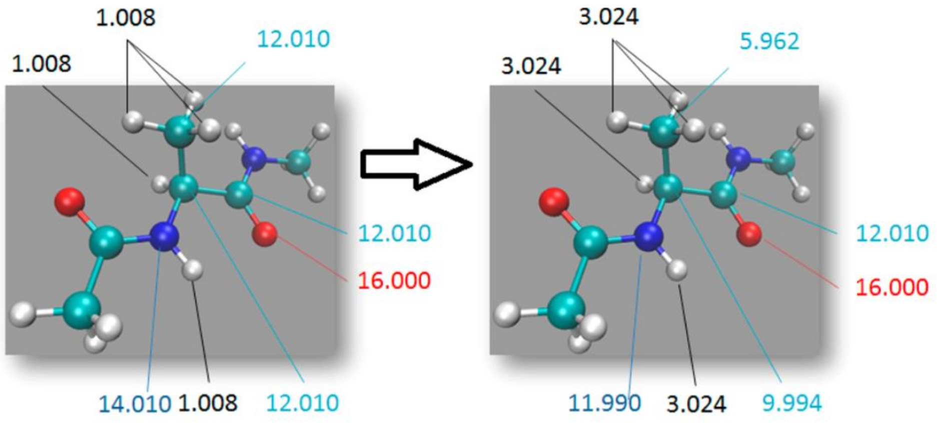
dt=2fs is often used in simulating rigid models of water with constraint (SHAKE, SETTLE)

"Fast internal vibrations are usually **decoupled** from rotational and translational motions?"



Hydrogen Mass Repartitioning (HMR)

The idea is to increase the mass of the hydrogen atoms while decreasing the mass of the oxygen atoms to increase the integration step size from **2 fs to 4 fs.**



(a) (b)

TABLE IV.
Summary of Maximum Time Steps (Δt_{max}).

Topology type	Δt_{max} (fs) A[a]	Δt_{max} (fs) B[b]
Normal 1 u	3	3
Normal 4 u	6	4
Dummy 1 u	8	7
Dummy 4 u	8	7

TABLE II.
Atomic Masses in Water.[a]

Mass (u) H	Mass (u) O	I (u nm^2)	η (10 ^{4} kg m ^{1} s ^{1})	D (10 ^{9} m^2 s ^{1})	$\tau_{H\ bond}$ (ps)	Drift E$_{tot}$ (kJ mol ^{1} ps ^{1})	Δt_{max} (fs)
1	16	0.0059	4.3	4.08	0.67	1.04	6.6
2	14	0.0104	4.7	3.89	0.74	0.86	8.9
3	12	0.0133	4.9	3.79	0.89	0.42	10.0
4	10	0.0148	4.9	3.34	0.79	0.36	10.3
5	8	0.0148	5.1	3.50	0.84	0.47	10.4
6	6	0.0133	5.3	3.35	0.84	0.59	8.6
7	4	0.0104	5.2	3.34	0.88	0.43	7.5
8	2	0.0059	5.1	3.60	0.95	0.61	5.6
Real H_2O		—	8.0	2.3	0.59	—	—
Real D_2O		—	—	2.0	—	—	—

[a] I: corresponding smallest moments of inertia; resulting dynamical properties: η: viscosity; D: diffusion constant. Values of H_2O and D_2O from Lide et al.[33] and hydrogen-bond lifetime ($\tau_{H\ bond}$) value of H_2O from Montrose[31]; RMS drift of the total energy over 12 runs at a time step of 4 fs; maximum time step (Δt_{max}) at a maximum order of 10 of the drift as a function of time step.

J. Comp. Chem., ***1999***, 20, 8, 786-798.
J. Chem. Theory Comput. ***2015***, 11, 1864-1874.

Multi-Timestep Integrator (MTS)

$$U(\mathbf{r}_1, ..., \mathbf{r}_N) = \sum_{\text{bonds}} \frac{1}{2} K_{\text{bond}} (r - r_0)^2 + \sum_{\text{bends}} \frac{1}{2} K_{\text{bend}} (\theta - \theta_0)^2$$
$$+ \sum_{\text{tors}} \sum_{n=0}^{6} A_n \left[1 + \cos(C_n \phi + \delta_n)\right]$$
$$+ \sum_{i,j \in nb} \left\{ 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6} \right] + \frac{q_i q_j}{r_{ij}} \right\}.$$

Intramolecular potentials (fast force) have large and rapidly varying components while **nonbonded potentials (slow force)** have slowly varying components due to long-range nature.

But we should choose integrator time step dt for the fast force.

Why don't we use a integrator capable of separating the time scales for a gain in computational efficiency, such as allowing the slow forces to be recomputed less frequently than the fast forces?

$$\dot{x} = \frac{p}{m}$$
$$\dot{p} = F_{\text{fast}}(x) + F_{\text{slow}}(x).$$

Multi-Timestep Integrator (MTS)

$$\dot{x} = \frac{p}{m}$$
$$\dot{p} = F_{\text{fast}}(x) + F_{\text{slow}}(x).$$

Liouville operator for this system is given by

$$iL = \frac{p}{m}\frac{\partial}{\partial x} + \left[F_{\text{fast}}(x) + F_{\text{slow}}(x)\right]\frac{\partial}{\partial p}$$

This separation (kinetic/force) leads to the standard velocity Verlet algorithm:

$$iL = iL_1 + iL_2$$

$$iL_1 = \frac{p}{m}\frac{\partial}{\partial x}$$

$$iL_2 = \left[F_{\text{fast}}(x) + F_{\text{slow}}(x)\right]\frac{\partial}{\partial p}.$$

Multi-Timestep Integrator (MTS)

$$\dot{x} = \frac{p}{m}$$
$$\dot{p} = F_{\text{fast}}(x) + F_{\text{slow}}(x).$$

Liouville operator for this system is given by

$$iL = \frac{p}{m}\frac{\partial}{\partial x} + \left[F_{\text{fast}}(x) + F_{\text{slow}}(x)\right]\frac{\partial}{\partial p}$$

This separation (fast/slow) leads to the **reference system propagator (RESPA) algorithm**:

$$iL = iL_{\text{fast}} + iL_{\text{slow}}$$

$$iL_{\text{fast}} = \frac{p}{m}\frac{\partial}{\partial x} + F_{\text{fast}}(x)\frac{\partial}{\partial p}$$
$$iL_{\text{slow}} = F_{\text{slow}}(x)\frac{\partial}{\partial p}.$$

$$\exp(iL\Delta t) = \exp\left(iL_{\text{slow}}\frac{\Delta t}{2}\right)\exp(iL_{\text{fast}}\Delta t)\exp\left(iL_{\text{slow}}\frac{\Delta t}{2}\right).$$

Mark T. Tuckerman, *Statistical mechanics: Theory and Molecular Simulation 2thed*, **2023**

Multi-Timestep Integrator (MTS)

$$\exp(iL\Delta t) = \exp\left(iL_{\mathrm{slow}}\frac{\Delta t}{2}\right)\exp(iL_{\mathrm{fast}}\Delta t)\exp\left(iL_{\mathrm{slow}}\frac{\Delta t}{2}\right).$$

In here, Δt is the time step of the slow force.

Introducing the time step of the fast force δt = Δt/n,

$$\exp(iL_{\mathrm{fast}}\Delta t) = \left[\exp\left(\frac{\delta t}{2}F_{\mathrm{fast}}\frac{\partial}{\partial p}\right)\exp\left(\delta t\frac{p}{m}\frac{\partial}{\partial x}\right)\exp\left(\frac{\delta t}{2}F_{\mathrm{fast}}\frac{\partial}{\partial p}\right)\right]^n.$$

$$\begin{aligned}\exp(iL\Delta t) = {} & \exp\left(\frac{\Delta t}{2}F_{\mathrm{slow}}\frac{\partial}{\partial p}\right)\\ & \times\left[\exp\left(\frac{\delta t}{2}F_{\mathrm{fast}}\frac{\partial}{\partial p}\right)\exp\left(\delta t\frac{p}{m}\frac{\partial}{\partial x}\right)\exp\left(\frac{\delta t}{2}F_{\mathrm{fast}}\frac{\partial}{\partial p}\right)\right]^n\\ & \times\exp\left(\frac{\Delta t}{2}F_{\mathrm{slow}}\frac{\partial}{\partial p}\right).\end{aligned}$$

Multi-Timestep Integrator (MTS)

$$\exp(iL\Delta t) = \exp\left(iL_{\text{slow}}\frac{\Delta t}{2}\right)\exp(iL_{\text{fast}}\Delta t)\exp\left(iL_{\text{slow}}\frac{\Delta t}{2}\right).$$

In here, Δt is the time step of the slow force.

Introducing the time step of the fast force δt = Δt/n,

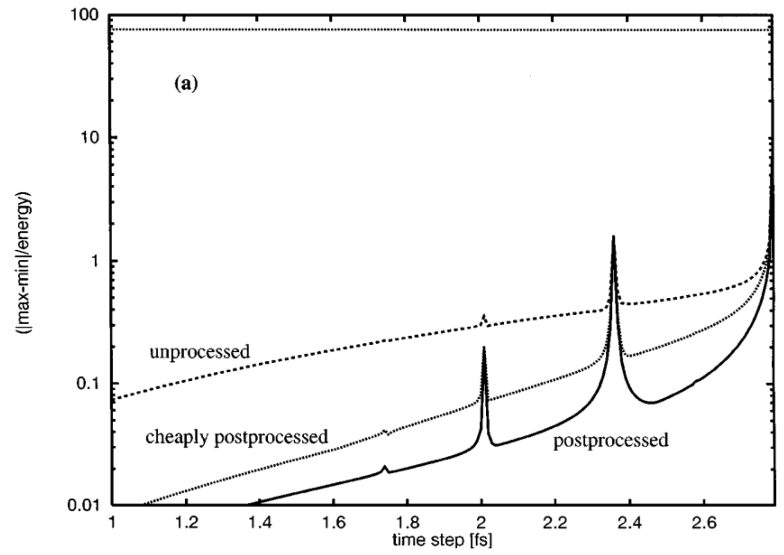
```
$p = p + 0.5 * \Delta t * F_{\text{slow}}$
for i = 1 to n
    $p = p + 0.5 * \delta t * F_{\text{fast}}$
    $x = x + \delta t * p/m$
    Recalculate fast force
    $p = p + 0.5 * \delta t * F_{\text{fast}}$
endfor
Recalculate slow force
$p = p + 0.5 * \Delta t * F_{\text{slow}}$.
```

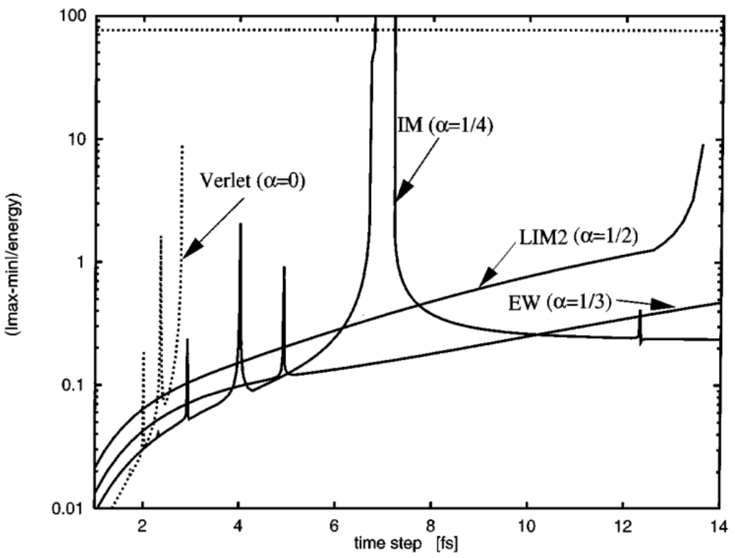
Resonance Instability

There is a limit on the size of the time step for the slow forces by that of the fast forces

So called the **resonant time step** is set by the fast frequency w.

The higher the value of w, the smaller time step should be chosen, which limits the computational savings afforded by the algorithm independent of how slow the slow force is.





What properties should integration algorithms satisfy?

1. Stability

- Energy conservation
- Symplecticity
- Time reversibility

2. Efficiency

- Maximum permissible time step
- Constraint algorithm
- Hydrogen mass repartitioning
- Multi-step integration

3. Accuracy

- Configurational sampling
- Dynamical properties

Operator splitting order is related to the configurational sampling

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix}}_{\mathsf{R}} + \underbrace{\begin{bmatrix} 0 \\ -M^{-1}\nabla U(\mathbf{x}) \end{bmatrix}}_{\mathsf{V}} + \underbrace{\begin{bmatrix} 0 \\ -\gamma\,\mathbf{v} + \sqrt{2\gamma}\,(\beta M)^{-1/2}\dot{\mathbf{W}} \end{bmatrix}}_{\mathsf{O}}$$

$$\mathsf{R} \ : \ e^{\mathcal{L}_{\mathsf{R}}\tau} \quad : \quad \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix}\tau$$

$$\mathsf{V} \ : \ e^{\mathcal{L}_{\mathsf{V}}\tau} \quad : \quad \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ -M^{-1}\nabla U(\mathbf{x}) \end{bmatrix}\tau$$

$$\mathsf{O} \ : \ e^{\mathcal{L}_{\mathsf{O}}\tau} \quad : \quad \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ (a(\tau)-1)\mathbf{v} + \sqrt{1-a(\tau)^2}\,(\beta M)^{-1/2}\,\xi \end{bmatrix}$$

Once a splitting is defined, the propagator exp(Ldt) can be approximated as a Trotter factorization.

$$\begin{aligned} & e^{[A+B+C+D]\Delta t} \\ &= e^{A\Delta t/2} e^{[B+C+D]\Delta t} e^{A\Delta t/2} + \mathcal{O}(\Delta t^3) \\ &= e^{A\Delta t/2} e^{B\Delta t/2} e^{[C+D]\Delta t} e^{B\Delta t/2} e^{A\Delta t/2} + \mathcal{O}(\Delta t^3) \\ &= e^{A\Delta t/2} e^{B\Delta t/2} e^{C\Delta t/2} e^{D\Delta t} e^{C\Delta t/2} e^{B\Delta t/2} e^{A\Delta t/2} + \mathcal{O}(\Delta t^3) \end{aligned}$$

Operator splitting order is related to the configurational sampling

ORVRO

$$e^{[\mathcal{L}_o+\mathcal{L}_v+\mathcal{L}_r+\mathcal{L}_h]\Delta t} \simeq e^{\mathcal{L}_o\Delta t/2} e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_v\Delta t} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_o\Delta t/2}$$

RVOVR (ABOBA)

$$e^{[\mathcal{L}_o+\mathcal{L}_v+\mathcal{L}_r+\mathcal{L}_h]\Delta t} \simeq e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_v\Delta t/2} e^{\mathcal{L}_o\Delta t} e^{\mathcal{L}_v\Delta t/2} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_r\Delta t/2}$$

VRORV (BAOAB)

$$e^{[\mathcal{L}_o+\mathcal{L}_v+\mathcal{L}_r+\mathcal{L}_h]\Delta t} \simeq e^{\mathcal{L}_v\Delta t/2} e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_o\Delta t} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_v\Delta t/2}$$

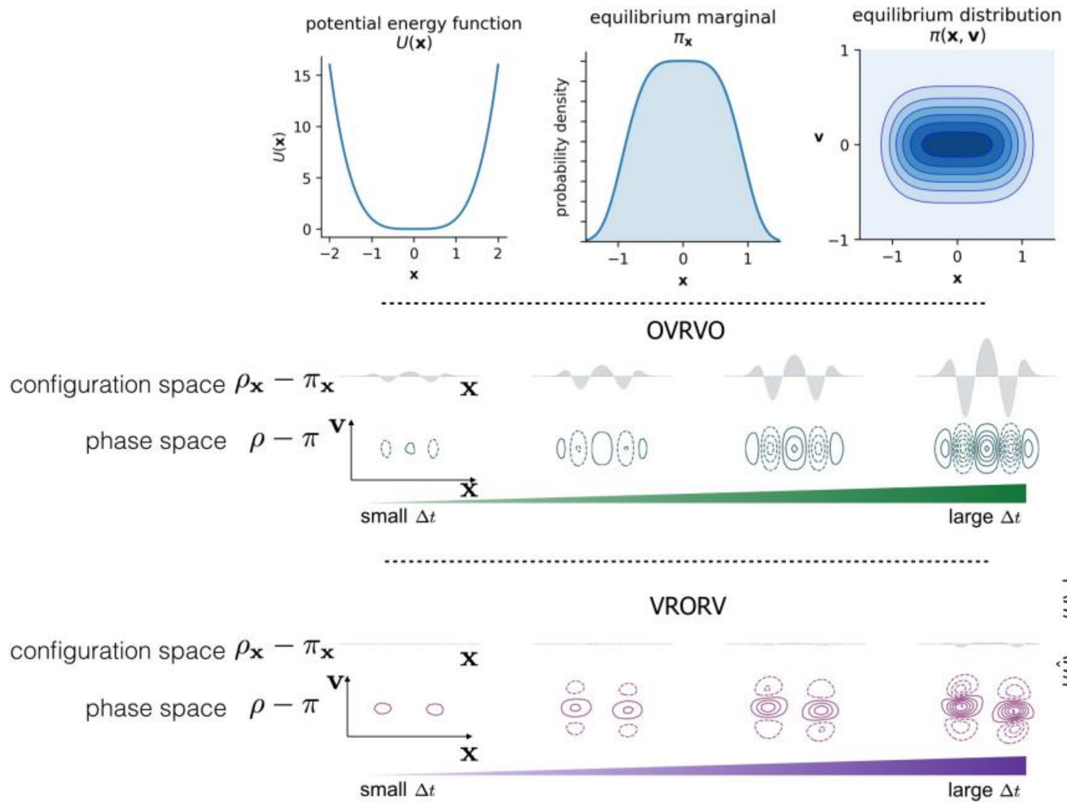
ROVOR

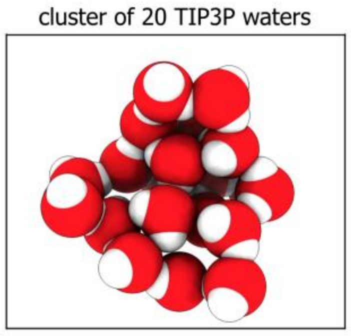
$$e^{[\mathcal{L}_o+\mathcal{L}_v+\mathcal{L}_r+\mathcal{L}_h]\Delta t} \simeq e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_o\Delta t/2} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_v\Delta t} e^{\mathcal{L}_h\Delta t/2} e^{\mathcal{L}_o\Delta t/2} e^{\mathcal{L}_r\Delta t/2}$$

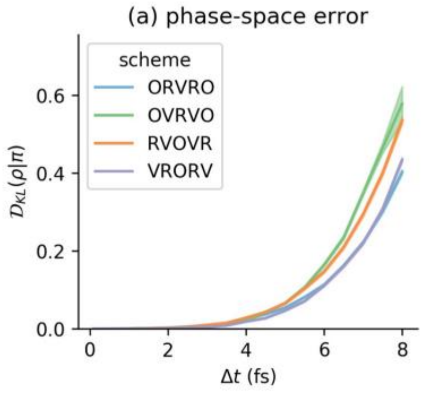
VOROV

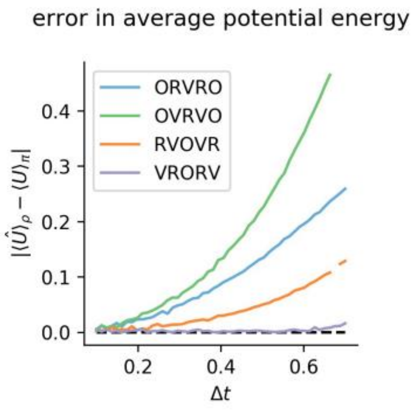
$$e^{[\mathcal{L}_o+\mathcal{L}_v+\mathcal{L}_r+\mathcal{L}_h]\Delta t} \simeq e^{\mathcal{L}_v\Delta t/2} e^{\mathcal{L}_o\Delta t/2} e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_h\Delta t} e^{\mathcal{L}_r\Delta t/2} e^{\mathcal{L}_o\Delta t/2} e^{\mathcal{L}_v\Delta t/2}$$

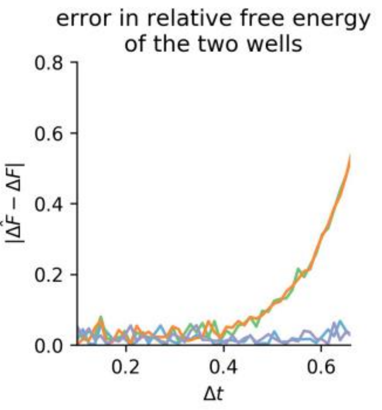
Operator splitting order is related to the configurational sampling


potential energy function $U(\mathbf{x})$
equilibrium marginal $\pi_{\mathbf{x}}$
equilibrium distribution $\pi(\mathbf{x}, \mathbf{v})$
$U(\mathbf{x})$
probability density
$\mathbf{v}$
$\mathbf{x}$
OVRVO
configuration space $\rho_{\mathbf{x}} - \pi_{\mathbf{x}}$
phase space $\rho - \pi$
small Δt
large Δt
VRORV
configuration space $\rho_{\mathbf{x}} - \pi_{\mathbf{x}}$
phase space $\rho - \pi$
small Δt
large Δt









Operator splitting order is related to the dynamic properties

Table 1. Definition of Dynamical Properties[a]

external force	quantity	expression			continuous-limit value
zero	mean-squared displacement	$\langle r^2(n)\rangle$	or	$\langle r^2(n+1/2)\rangle$	$2/(\beta m\gamma)\ n\Delta t$
	mean-squared velocity	$\langle v^2(n)\rangle$	or	$\langle v^2(n+1/2)\rangle$	$1/(\beta m)$
	velocity autocorrelation	$\langle v(n)v(n+\Delta n)\rangle$	or	$\langle v(n+1/2)\ v(n+1/2+\Delta n)\rangle$	$1/(\beta m)\ e^{-\gamma\Delta n\Delta t}$
uniform, f	terminal drift	$\langle r(n+1)-r(n)\rangle\ /\ \Delta t$	=	$\langle r(n+1/2)-r(n-1/2)\rangle\ /\ \Delta t$	$f/(m\gamma)$
linear, $-kr$	mean-squared displacement	$\langle r^2(n)\rangle$	or	$\langle r^2(n+1/2)\rangle$	$1/(\beta k)$
	mean-squared velocity	$\langle v^2(n)\rangle$	or	$\langle v^2(n+1/2)\rangle$	$1/(\beta m)$
	virial	$m\langle v^2(n)\rangle - k\langle r^2(n)\rangle$	or	$m\langle v^2(n+1/2)\rangle - k\langle r^2(n+1/2)\rangle$	0

Table 2. Comparison of Properties for Different Splittings[a]

desideratum	OVRVO	ORVRO	RVOVR	VRORV	VOROV	ROVOR
			All Six Splittings Perform Identically			
form is time-reversal symmetric	yes	yes	yes	yes	yes	yes
splits heat, work, and shadow work	yes	yes	yes	yes	yes	yes
easily incorporates constraints	yes	yes	yes	yes	yes	yes
force evaluations per time step	one	one	one	one	one	one
zero-force MSV	exact	exact	exact	exact	exact	exact
zero-force VAC	exact	exact	exact	exact	exact	exact
zero-force MSD	exact	exact	exact	exact	exact	exact
linear-force virial	$O(\Delta t^2)$	$O(\Delta t^2)$	$O(\Delta t^2)$	$O(\Delta t^2)$	$O(\Delta t^2)$	$O(\Delta t^2)$
			Splittings Differ in Performance			
uniform-force terminal drift	exact	exact	exact	exact	$O(\Delta t^2)$	$O(\Delta t^2)$
linear-force MSD	$O(\Delta t^2)$ at n exact at $n+1/2$	$O(\Delta t^2)$ at n exact at $n+1/2$	exact at n	exact at n	$O(\Delta t^2)$	$O(\Delta t^2)$
linear-force MSV	exact at n	exact at n	$O(\Delta t^2)$ at n exact at $n+1/2$	$O(\Delta t^2)$ at n exact at $n+1/2$	$O(\Delta t^4)$ at n	$O(\Delta t^2)$ at n $O(\Delta t^4)$ at $n+1/2$

Takeaways

■ Liouville operator splitting is important to ensure the stability of the integration algorithm, and to provide the direct translation of algorithms.

■ Efficiency of the integration algorithm can be gained by increasing the maximum permissible time step.

■ Several strategies are the constraint algorithm, hydrogen mass repartitioning, and multi-step integration algorithms.

■ Changing the splitting orders of the integration procedure can impact on the configurational sampling and the dynamic properties of the system.

Q&A