



OpenMM Theory Guide

Preview Release 4

August 19, 2009

Website: SimTK.org/home/openmm

Copyright and Permission Notice

Portions copyright (c) 2009 Stanford University and Peter Eastman.

Permission is hereby granted, free of charge, to any person obtaining a copy of this document (the "Document"), to deal in the Document without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Document, and to permit persons to whom the Document is furnished to do so, subject to the following conditions:

This copyright and permission notice shall be included in all copies or substantial portions of the Document.

THE DOCUMENT IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS, CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE DOCUMENT OR THE USE OR OTHER DEALINGS IN THE DOCUMENT.

Acknowledgments

OpenMM software and all related activities are funded by the [Simbios](#) National Center for Biomedical Computing through the National Institutes of Health Roadmap for Medical Research, Grant U54 GM072970. Information on the National Centers can be found at <http://nihroadmap.nih.gov/bioinformatics>.

Table of Contents

1	INTRODUCTION	1
1.1	Overview	1
1.2	Units	1
2	FORCES	3
2.1	HarmonicBondForce	3
2.2	HarmonicAngleForce	3
2.3	PeriodicTorsionForce	4
2.4	RBTorsionForce	4
2.5	NonbondedForce	4
2.5.1	Lennard-Jones Interaction	5
2.5.2	Coulomb Interaction Without Cutoff	6
2.5.3	Coulomb Interaction With Cutoff	6
2.5.4	Coulomb Interaction With Ewald Summation	7
2.6	GBSAOBCForce	8
2.6.1	Generalized Born Term	8
2.6.2	Surface Area Term	9
2.7	AndersenThermostat	10
2.8	CMMotionRemover	10
3	INTEGRATORS	11
3.1	VerletIntegrator	11
3.2	LangevinIntegrator	11
3.3	BrownianIntegrator	12
3.4	VariableVerletIntegrator	12
3.5	VariableLangevinIntegrator	14

1 Introduction

1.1 Overview

This guide describes the mathematical theory behind OpenMM. For each computational class, it describes what computations the class performs and how it should be used. This serves two purposes. If you are using OpenMM within an application, this guide teaches you how to use it correctly. If you are implementing the OpenMM API for a new Platform, it teaches you how to correctly implement the required kernels.

On the other hand, many details are intentionally left unspecified. Any behavior that is not specified either in this guide or in the API documentation is left up to the Platform, and may be implemented in different ways by different Platforms. For example, an Integrator is required to produce a trajectory that satisfies constraints to within the user specified tolerance, but the algorithm used to enforce those constraints is left up to the Platform. Similarly, this guide provides the functional form of each Force, but does not specify what level of numerical precision it must be calculated to.

This is an essential feature of the design of OpenMM, because it allows the API to be implemented efficiently on a wide variety of hardware and software platforms, using whatever methods are most appropriate for each platform. On the other hand, it means that a single program may produce meaningfully different results depending on which Platform it uses. For example, different constraint algorithms may have different regions of convergence, and thus a time step that is stable on one platform may be unstable on a different one. It is essential that you validate your simulation methodology on each Platform you intend to use, and do not assume that good results on one Platform will guarantee good results on another Platform when using identical parameters.

1.2 Units

There are several different sets of units widely used in molecular simulations. For example, energies may be measured in kcal/mol or kJ/mol, distances may be in Angstroms or nm, and angles may be in degrees or radians. OpenMM uses the following units everywhere.

Quantity	Units
distance	nm
time	ps
mass	atomic mass units
charge	proton charge
temperature	Kelvin
angle	radians
energy	kJ/mol

These units have the important feature that they form an internally consistent set. For example, a force always has the same units (kJ/mol/nm) whether it is calculated as the gradient of an energy or as the product of a mass and an acceleration. This is not true in some other widely used unit systems, such as those that express energy in kcal/mol.

The header file `Units.h` contains predefined constants for converting between the OpenMM units and some other common units. For example, if your application expresses distances in Angstroms, you should multiply them by `OpenMM::NmPerAngstrom` before passing them to OpenMM, and positions calculated by OpenMM should be multiplied by `OpenMM::AngstromsPerNm` before passing them back to your application.

2 Forces

2.1 HarmonicBondForce

Each harmonic bond is represented by an energy term of the form

$$E = \frac{1}{2} k (x - x_0)^2$$

where x is the distance between the two particles, x_0 is the equilibrium distance, and k is the force constant. This produces a force of magnitude $k(x-x_0)$.

Be aware that some force fields define their harmonic bond parameters in a slightly different way: $E = k'(x-x_0)^2$, leading to a force of magnitude $2k'(x-x_0)$. Comparing these two forms, you can see that $k = 2k'$. Be sure to check which form a particular force field uses, and if necessary multiply the force constant by 2.

2.2 HarmonicAngleForce

Each harmonic angle is represented by an energy term of the form

$$E = \frac{1}{2} k (\theta - \theta_0)^2$$

where θ is the angle formed by the three particles, θ_0 is the equilibrium angle, and k is the force constant.

As with HarmonicBondForce, be aware that some force fields define their harmonic angle parameters as $E = k'(\theta - \theta_0)^2$. Be sure to check which form a particular force field uses, and if necessary multiply the force constant by 2.

2.3 PeriodicTorsionForce

Each torsion is represented by an energy term of the form

$$E = k(1 + \cos(n\theta - \theta_0))$$

where θ is the dihedral angle formed by the four particles, θ_0 is the equilibrium angle, n is the periodicity, and k is the force constant.

2.4 RBTorsionForce

Each torsion is represented by an energy term of the form

$$E = \sum_{i=0}^5 C_i (\cos \phi)^i$$

where ϕ is the dihedral angle formed by the four particles and C_0 through C_5 are constant coefficients.

For reason of convention, PeriodicTorsionForce and RBTorsionForce define the torsion angle differently. θ is zero when the first and last particles are on the *same* side of the bond formed by the middle two particles (the *cis* configuration), whereas ϕ is zero when they are on *opposite* sides (the *trans* configuration). This means that $\theta = \phi - \pi$.

2.5 NonbondedForce

2.5.1 Lennard-Jones Interaction

The Lennard-Jones interaction between each pair of particles is represented by an energy term of the form

$$E = 4\varepsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right)$$

where r is the distance between the two particles, σ is the distance at which the energy equals zero, and ε sets the strength of the interaction. If the NonbondedMethod in use is anything other than NoCutoff and r is greater than the cutoff distance, the energy and force are both set to zero. Because the interaction decreases very quickly with distance, the cutoff usually has little effect on the accuracy of simulations.

When an exception has been added for a pair of particles, σ and ε are the parameters specified by the exception. Otherwise they are determined from the parameters of the individual particles using the Lorentz-Bertelot combining rule:

$$\sigma = \frac{\sigma_1 + \sigma_2}{2}$$

$$\varepsilon = \sqrt{\varepsilon_1 \varepsilon_2}$$

The Lennard-Jones interaction is often parameterized in two other equivalent ways. One is

$$E = \varepsilon \left(\left(\frac{r_{min}}{r} \right)^{12} - 2 \left(\frac{r_{min}}{r} \right)^6 \right)$$

where r_{min} is the distance at which the energy is minimum. It is related to σ by

$$\sigma = \frac{r_{min}}{2^{1/6}}$$

In turn, r_{min} is related to the van der Waals radius by $r_{min} = 2 r_{vdw}$.

Another common form is

$$E = \frac{A}{r^{12}} - \frac{B}{r^6}$$

The coefficients A and B are related to σ and ε by

$$\sigma = \left(\frac{A}{B} \right)^{1/6}$$

$$\varepsilon = \frac{B^2}{4A}$$

2.5.2 Coulomb Interaction Without Cutoff

The form of the Coulomb interaction between each pair of particles depends on the NonbondedMethod in use. For NoCutoff, it is given by

$$E = \frac{1}{4\pi\varepsilon_0} \frac{q_1 q_2}{r}$$

where q_1 and q_2 are the charges of the two particles, and r is the distance between them.

2.5.3 Coulomb Interaction With Cutoff

For CutoffNonPeriodic or CutoffPeriodic, it is modified using the reaction field approximation. This is derived by assuming everything beyond the cutoff distance is a solvent with a uniform dielectric constant.¹

$$E = \frac{q_1 q_2}{4\pi\varepsilon_0} \left(\frac{1}{r} + k_{rf} r^2 - c_{rf} \right)$$

$$k_{rf} = \left(\frac{1}{r_{cutoff}^3} \right) \left(\frac{\varepsilon_{solvent} - 1}{2\varepsilon_{solvent} + 1} \right)$$

$$c_{rf} = \left(\frac{1}{r_{cutoff}} \right) \left(\frac{3\epsilon_{solvent}}{2\epsilon_{solvent} + 1} \right)$$

where r_{cutoff} is the cutoff distance and $\epsilon_{solvent}$ is the dielectric constant of the solvent. In the limit $\epsilon_{solvent} \gg 1$, this causes the force to go to zero at the cutoff.

2.5.4 Coulomb Interaction With Ewald Summation

For Ewald, the total Coulomb energy is the sum of three terms: the *direct space sum*, the *reciprocal space sum*, and the *self-energy term*.²

$$E = E_{dir} + E_{rec} + E_{self}$$

$$E_{dir} = \frac{1}{2} \sum_{i,j} \sum_{\mathbf{n}} q_i q_j \frac{\text{erfc}(\alpha r_{ij,\mathbf{n}})}{r_{ij,\mathbf{n}}}$$

$$E_{rec} = \frac{1}{2\pi V} \sum_{i,j} q_i q_j \sum_{\mathbf{k} \neq 0} \frac{\exp(-(\pi \mathbf{k} / \alpha)^2 + 2\pi i \mathbf{k} \cdot (\mathbf{r}_i - \mathbf{r}_j))}{\mathbf{k}^2}$$

$$E_{self} = -\frac{\alpha}{\sqrt{\pi}} \sum_i q_i^2$$

In the above expressions, the indices i and j run over all particles, $\mathbf{n} = (n_1, n_2, n_3)$ runs over all copies of the periodic cell, and $\mathbf{k} = (k_1, k_2, k_3)$ runs over all integer wave vectors from $(-k_{max}, -k_{max}, -k_{max})$ to $(k_{max}, k_{max}, k_{max})$ excluding $(0, 0, 0)$. \mathbf{r}_i is the position of particle i , while r_{ij} is the distance between particles i and j . V is the volume of the periodic cell, and α is an internal parameter.

In the direct space sum, all pairs that are further apart than the cutoff distance are ignored. Because the cutoff is required to be less than half the width of the periodic cell, the number of terms in this sum is never greater than the square of the number of particles.

The error made by applying the direct space cutoff depends on the magnitude of $\text{erfc}(\alpha r_{cutoff})$. Similarly, the error made in the reciprocal space sum by ignoring wave numbers beyond k_{max}

depends on the magnitude of $\exp(-(\pi k_{max}/\alpha)^2)$. By changing α , one can decrease the error in either term while increasing the error in the other one.

Instead of having the user specify α and k_{max} , NonbondedForce instead asks the user to choose an error tolerance δ . It then calculates the other parameters as

$$\alpha = \sqrt{-\log(\delta)/r_{cutoff}}$$

$$k_{max} = \frac{-d}{\pi r_{cutoff}} \log(\delta)$$

where d is the width of the periodic box. (If the box is not square, k_{max} will have a different value along each axis.)

This means that the accuracy of the calculation is determined by δ . r_{cutoff} does not affect the accuracy of the result, but does affect the speed of the calculation by changing the relative costs of the direct space and reciprocal space sums. You therefore should test different cutoffs to find the value that gives best performance; this will in general vary both with the size of the system and with the Platform being used for the calculation. When the optimal cutoff is used for every simulation, the overall cost of evaluating the nonbonded forces scales as $O(N^{3/2})$ in the number of particles.

2.6 GBSAOBCForce

2.6.1 Generalized Born Term

GBSAOBCForce consists of two energy terms: a Generalized Born Approximation term to represent the electrostatic interaction between the solute and solvent, and a surface area term to represent the free energy cost of solvating a neutral molecule. The Generalized Born energy is given by³

$$E = -\frac{1}{2} \left(\frac{1}{\epsilon_{solute}} - \frac{1}{\epsilon_{solvent}} \right) \sum_{i,j} \frac{q_i q_j}{f^{GB}(d_{ij}, R_i, R_j)}$$

where the indices i and j run over all particles, ϵ_{solute} and $\epsilon_{solvent}$ are the dielectric constants of the solute and solvent respectively, q_i is the charge of particle i , and d_{ij} is the distance between particles i and j . $f^{GB}(d_{ij}, R_i, R_j)$ is defined as

$$f^{GB}(d_{ij}, R_i, R_j) = \left[d_{ij}^2 + R_i R_j \exp\left(\frac{-d_{ij}}{4R_i R_j}\right) \right]^{1/2}$$

R_i is the Born radius of particle i , which calculated as

$$R_i = \frac{1}{\rho_i^{-1} - \rho_i^{-1} \tanh(\alpha \Psi_i - \beta \Psi_i^2 + \gamma \Psi_i^3)}$$

where α , β , and γ are the GB^{OBCII} parameters $\alpha = 1$, $\beta = 0.8$, and $\gamma = 4.85$. ρ_i is the adjusted atomic radius of particle i , which is calculated from the atomic radius r_i as $\rho_i = r_i - 0.009$ nm. Ψ_i is calculated as an integral over the van der Waals spheres of all particles outside particle i :

$$\Psi_i = \frac{\rho_i}{4\pi} \int_{VDW} \theta(|\mathbf{r}| - \rho_i) \frac{1}{|\mathbf{r}|^4} d^3\mathbf{r}$$

where $\theta(r)$ is a step function that excludes the interior of particle i from the integral.

2.6.2 Surface Area Term

The surface area term is given by^{4, 5}

$$E = 4\pi \cdot 2.26 \sum_i (r_i + r_{solvent})^2 \left(\frac{r_i}{R_i} \right)^6$$

where r_i is the atomic radius of particle i , R_i is its Born radius, and $r_{solvent}$ is the solvent radius, which is taken to be 0.14 nm.

2.7 AndersenThermostat

AndersenThermostat couples the system to a heat bath by randomly selecting a subset of particles at the start of each time step, then setting their velocities to new values chosen from a Boltzmann distribution. This represents the effect of random collisions between particles in the system and particles in the heat bath.⁶

The probability that a given particle will experience a collision in a given time step is

$$P = 1 - e^{-f\Delta t}$$

where f is the collision frequency and Δt is the step size. Each component of its velocity is then set to

$$\mathbf{v}_i = \sqrt{\frac{k_B T}{m}} R$$

where T is the thermostat temperature, m is the particle mass, and R is a random number chosen from a normal distribution with mean of zero and variance of one.

2.8 CMMotionRemover

CMMotionRemover prevents the system from drifting in space by periodically removing all center of mass motion. At the start of every n 'th time step (where n is set by the user), it calculates the total center of mass velocity of the system:

$$\mathbf{v}_{CM} = \frac{\sum_i m_i \mathbf{v}_i}{\sum_i m_i}$$

where m_i and \mathbf{v}_i are the mass and velocity of particle i . It then subtracts \mathbf{v}_{CM} from the velocity of every particle.

3 Integrators

3.1 VerletIntegrator

VerletIntegrator implements the leap-frog Verlet integration method. The positions and velocities stored in the context are offset from each other by half a time step. In each step, they are updated as follows:

$$\begin{aligned}\mathbf{v}_i(t + \Delta t / 2) &= \mathbf{v}_i(t - \Delta t / 2) + \mathbf{f}_i(t) \Delta t / m_i \\ \mathbf{r}_i(t + \Delta t) &= \mathbf{r}_i(t) + \mathbf{v}_i(t + \Delta t / 2) \Delta t\end{aligned}$$

where \mathbf{v}_i is the velocity of particle i , \mathbf{r}_i is its position, \mathbf{f}_i is the force acting on it, m_i is its mass, and Δt is the time step.

Because the positions are always half a time step later than the velocities, care must be used when calculating the energy of the system. In particular, the potential energy and kinetic energy in a State correspond to different times, and you cannot simply add them to get the total energy of the system. Instead, it is better to retrieve States after two successive time steps, calculate the on-step velocities as

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

then use those velocities to calculate the kinetic energy at time t .

3.2 LangevinIntegrator

LangevinIntegrator simulates a system in contact with a heat bath by integrating the Langevin equation of motion:

$$m_i \frac{d\mathbf{v}_i}{dt} = \mathbf{f}_i - \gamma m_i \mathbf{v}_i + \mathbf{R}_i$$

where \mathbf{v}_i is the velocity of particle i , \mathbf{f}_i is the force acting on it, m_i is its mass, γ is the friction coefficient, and \mathbf{R}_i is an uncorrelated random force whose components are chosen from a normal distribution with mean zero and variance $2m_i\gamma k_B T$, where T is the temperature of the heat bath.

The integration is done using a leap-frog method similar to VerletIntegrator.⁷ The same comments about the offset between positions and velocities apply to this integrator as to that one.

3.3 BrownianIntegrator

BrownianIntegrator simulates a system in contact with a heat bath by integrating the Brownian equation of motion:

$$\frac{d\mathbf{r}_i}{dt} = \frac{1}{\gamma} \mathbf{f}_i + \mathbf{R}_i$$

where \mathbf{r}_i is the position of particle i , \mathbf{f}_i is the force acting on it, γ is the friction coefficient, and \mathbf{R}_i is an uncorrelated random force whose components are chosen from a normal distribution with mean zero and variance $2k_B T/\gamma$, where T is the temperature of the heat bath.

The Brownian equation of motion is derived from the Langevin equation of motion in the limit of large γ . In that case, the velocity of a particle is determined entirely by the instantaneous force acting on it, and kinetic energy ceases to have much meaning, since it disappears as soon as the applied force is removed.

3.4 VariableVerletIntegrator

This is very similar to VerletIntegrator, but instead of using the same step size for every time step, it continuously adjusts the step size to keep the integration error below a user specified tolerance. It compares the positions generated by Verlet integration with those that would be generated by an explicit Euler integrator, and takes the difference between them as an estimate of the integration error:

$$error = (\Delta t)^2 \sum_i \frac{|\mathbf{f}_i|}{m_i}$$

where \mathbf{f}_i is the force acting on particle i and m_i is its mass. (In practice, the error made by the Euler integrator is usually larger than that made by the Verlet integrator, so this tends to overestimate the true error. Even so, it can provide a useful mechanism for step size control.)

It then selects the value of Δt that makes the error exactly equal the specified error tolerance:

$$\Delta t = \sqrt{\frac{\delta}{\sum_i \frac{|\mathbf{f}_i|}{m_i}}}$$

where δ is the error tolerance. This is the largest step that may be taken consistent with the user specified accuracy requirement.

(Note that the integrator may sometimes choose to use a smaller value for Δt than given above. For example, it might restrict how much the step size can grow from one step to the next, or keep the step size constant rather than increasing it by a very small amount. This behavior is not specified and may vary between Platforms. It is required, however, that Δt never be larger than the value given above.)

A variable time step integrator is generally superior to a fixed time step one in both stability and efficiency. It can take larger steps on average, but will automatically reduce the step size to preserve accuracy and avoid instability when unusually large forces occur. Conversely, when each uses the same step size on average, the variable time step one will usually be

more accurate since the time steps are concentrated in the most difficult areas of the trajectory.

Unlike a fixed step size Verlet integrator, variable step size Verlet is not symplectic. This means that for a given average step size, it will not conserve energy as precisely over long time periods, even though each local region of the trajectory is more accurate. For this reason, it is most appropriate when precise energy conservation is not important, such as when simulating a system at constant temperature. For constant energy simulations that must maintain the energy accurately over long time periods, the fixed step size Verlet may be more appropriate.

3.5 VariableLangevinIntegrator

This is similar to LangevinIntegrator, but it continuously adjusts the step size using the same method as VariableVerletIntegrator. It is usually preferred over the fixed step size Langevin integrator for the reasons given above. Furthermore, because Langevin dynamics involves a random force, it can never be symplectic and therefore the fixed step size Verlet integrator's advantages do not apply to the Langevin integrator.

4 References

1. Tironi, I. G.; Sperb, R.; Smith, P. E.; van Gunsteren, W. F., A generalized reaction field method for molecular dynamics simulations. *Journal of Chemical Physics* **1995**, 102, (13), 5451-5459.
2. Toukmaji, A. Y.; Board Jr, J. A., Ewald summation techniques in perspective: a survey. *Computer Physics Communications* **1996**, 95, 73-92.
3. Onufriev, A.; Bashford, D.; Case, D. A., Exploring protein native states and large-scale conformational changes with a modified generalized born model. *Proteins* **2004**, 55, (22), 383-394.
4. Schaefer, M.; Bartels, C.; Karplus, M., Solution conformations and thermodynamics of structured peptides: molecular dynamics simulation with an implicit solvation model. *Journal of Molecular Biology* **1998**, 284, (3), 835-848.
5. Ponder, J., Personal communication. This expression differs slightly from that given by Schaefer et al. This form was found to give a better correlation with surface area.
6. Andersen, H. C., Molecular dynamics simulations at constant pressure and/or temperature. *Journal of Chemical Physics* **1980**, 72, (4), 2384-2393.
7. van Gunsteren, W. F.; Berendsen, H. J. C., A Leap-Frog Algorithm for Stochastic Dynamics. *Molecular Simulation* **1988**, 1, 173-185.