Entropy **2014**, *16*, 138-162; doi:10.3390/e16010138

OPEN ACCESS

entropy

ISSN 1099-4300

www.mdpi.com/journal/entropy

Article

Time Integrators for Molecular Dynamics

Nawaf Bou-Rabee

Department of Mathematical Sciences, Rutgers University—Camden, 311 N 5th Street, Camden, NJ 08102, USA; E-Mail: nawaf.bourabee@rutgers.edu; Tel.: +1-856-225-6093; Fax: +1-856-225-6602

Received: 19 September 2013; in revised form: 20 November 2013 / Accepted: 4 December 2013 / Published: 27 December 2013

Abstract: This paper invites the reader to learn more about time integrators for Molecular Dynamics simulation through a simple MATLAB implementation. An overview of methods is provided from an algorithmic viewpoint that emphasizes long-time stability and finite-time dynamic accuracy. The given software simulates Langevin dynamics using an explicit, second-order (weakly) accurate integrator that exactly reproduces the Boltzmann-Gibbs density. This latter feature comes from adding a Metropolis acceptance-rejection step to the integrator. The paper discusses in detail the properties of the integrator. Since these properties do not rely on a specific form of a heat or pressure bath model, the given algorithm can be used to simulate other bath models including, e.g., the widely used v-rescale thermostat.

Keywords: explicit integrators; Metropolis algorithm; ergodicity; weak accuracy

MSC Classification: 82C80 (Primary); 82C31, 65C30, 65C05 (Secondary)

---

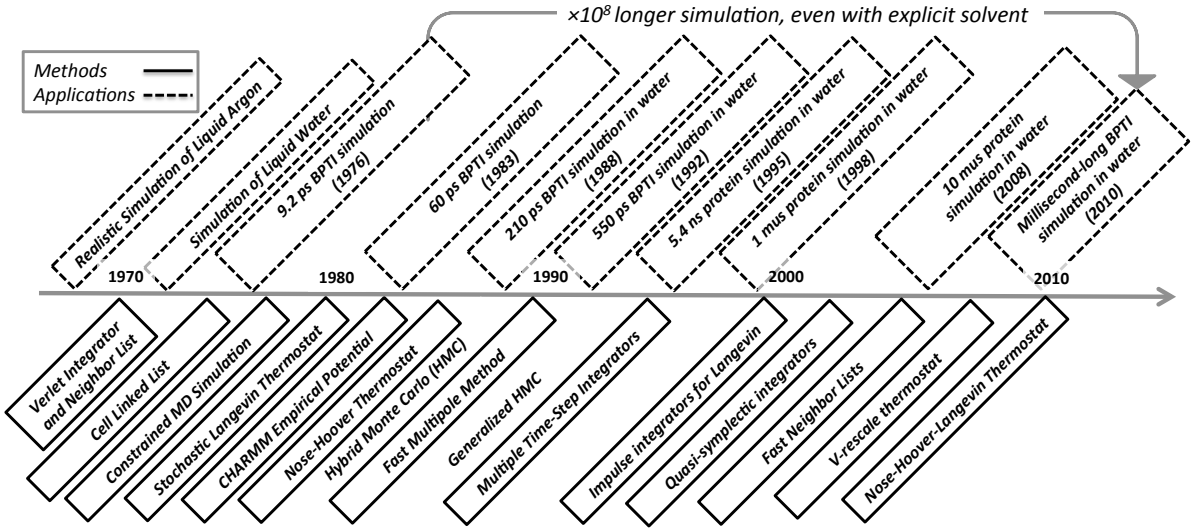
1. Introduction

Molecular Dynamics (MD) simulation refers to the time integration of Hamilton's equations often coupled to a heat or pressure bath [1–5]. From its early use in computing equilibrium dynamics of homogeneous molecular systems [6–13] and pico- to nano-scale protein dynamics [14–23], the method has evolved into a general purpose tool for simulating statistical properties of heterogeneous molecular systems [24]. Accessible time horizons have increased remarkably: the time line in Figure 1 attempts to capture this nearly billion-fold improvement in capability over the last forty or so years. To put this speedup in perspective, though, computing power has increased by about eight powers of ten over this time period as predicted by Moore's law.

To be clear, the selection of applications and methods shown in Figure 1 is not comprehensive and heavily biased towards the specific ideas and methods that inform this paper. The applications highlighted are simulations of liquid argon [6], water [11], protein dynamics without solvent [14,15] and biopolymer dynamics with solvent [25–31]. The methods include the following "upgrades" to MD simulation: Verlet integrator and neighbor lists [7], cell linked list [32], the SHAKE integrator for constraints [33], stochastic heat baths via Langevin dynamics [34,35], a library of empirical potentials [36], a deterministic heat bath via Nosé-Hoover dynamics [37,38], the fast multipole method [39], multiple time steps [40], splitting methods for Langevin dynamics [41–43], quasi-symplectic integrators [44,45], (fast) combined neighbor and cell lists [46], the v-rescale thermostat [47] and the stochastic Nosé-Hoover Langevin thermostat [48–50].

Near future applications of MD simulation include micro- to milli-scale simulations of biomolecular processes, like protein folding, ligand binding, membrane transport and biopolymer conformational changes [51–53]. In addition, atomistic MD simulations are used more sparingly in multiscale models [54–58] and rare event simulation, such as the finite temperature string method and milestoning [59–62]. Given this continuous development and generalization of MD, it is not a stretch to suppose that MD will play a transformative role in medicine, technology and education in the twenty-first century.

Figure 1. A time line of selected developments in MD simulation.



In its standard form, the method inputs a random initial condition, physical and numerical parameters and outputs a long discrete path of the molecular system. Statistical quantities, like velocity correlation or mean radius of gyration, are usually computed online, *i.e.*, as points along this trajectory are produced. MD simulation is built atop a cheap forward Euler-like integrator that requires only a single interactomic force field evaluation per step. Even though MD seems straightforward, software implementations of MD are typically optimized for performance [36,63,64], and as a side effect, make it cumbersome for non-experts to learn and modify.

Also, besides this issue, due to the interplay between stochastic Brownian and molecular forces, infinitely long trajectories of existing MD integrators do not have the right distribution. What happens

is that the Brownian force can cause the integrator to enter regions where its approximation to the molecular force is inaccurate and possibly destabilizing. In the latter case, the approximation spends a disproportionate amount of time at higher energies, and thus, the invariant measure of the approximation, if it even exists, is not correct. This phenomenon is a well-known shortcoming of explicit integrators for nonlinear diffusions [65–69].

Recently, a probabilistic approach was proposed to solve this problem, which questions the notion that Monte Carlo methods and MD have different aims: the former strictly samples probability distributions, and the latter estimates dynamics. The basic idea is to combine a standard MD integrator with a Metropolis-Hastings algorithm targeted to the Boltzmann-Gibbs distribution [70–72]. Because the scheme is a Monte Carlo method, it exactly preserves the desired distribution [71,72]. This property implies numerical stability over long-time simulations. However, the price to be paid for this stability is a loss of accuracy whenever a move is rejected and some overhead in evaluating the Metropolis acceptance-rejection step. Still, a Metropolized integrator is dynamically accurate on finite-time intervals [72,73], and so, even though a Metropolized integrator involves a Monte Carlo step, its aim and philosophy are very different from Monte Carlo methods, whose only goal is to sample a target distribution with no concern for the dynamics [71,74–82]. In principle, this approach offers a simple alternative to costly implicit integrators, but are Metropolized integrators ready for daily use in MD simulation? The answer to this question is unclear, since this approach is new and has not been tested on enough examples.

Motivated by these issues, this paper builds a software system for MD simulation with a Metropolis step built in and applies it to a homogeneous molecular system. The algorithm and its properties are introduced in a step-by-step fashion. In particular, we show that the integrator is second-order weakly accurate on finite-time intervals and converges to the Boltzmann-Gibbs distribution in the long-time limit. The software version of the algorithm is written in the latest version of MATLAB with plenty of comments, variables that are descriptively named and operations that can be easily translated into mathematical expressions [83]. Since MATLAB is widely available, this design ensures that the software will be easy-to-use and cross-platform. The following MATLAB-specific file formats will be used.

(F1) MATLAB script and function files are written in the MATLAB language and can be run from the MATLAB command line without ever compiling them.

(F2) MATLAB executable (MEX) files are written in the "C" language and compiled using the MATLAB `mex` function. The resulting executable is comparable in efficiency to a "C" code and can be called directly from the MATLAB command line. We will use MEX-files for performance-critical routines [84].

(F3) MATLAB binary (MAT) files will be used to store simulation data.

The paper is organized as follows. We begin with an overview of integrators that have been proposed in MD simulation in Section 2. We explain how to Metropolize each of these schemes to make them long-time stable in Section 3, and as an application, we use a Metropolized scheme to generate a long trajectory of a Lennard-Jones fluid in Section 4. Generalizations of corrected MD integrators to other molecular models are discussed in Section 5. The paper closes by discussing some potential pitfalls in high dimension and tricks to get the integrator to scale well in Section 6.

2. Algorithmic Introduction to Time Integrators for MD Simulation

For pedagogical reasons, we will start with Langevin dynamics of a system of N molecules. Then, we show in Section 5 how to simulate more general models of molecular systems. Denote by $m_j > 0$ and $\boldsymbol{q}_j$ the mass and position of the j-th molecule, respectively. The governing Langevin equation is given by:

$$\begin{cases} \frac{d\boldsymbol{q}_j}{dt}(t) = m_j^{-1}\boldsymbol{p}_j(t) \ , \\ d\boldsymbol{p}_j(t) = -\frac{\partial U}{\partial \boldsymbol{q}_j}(\boldsymbol{q}(t))dt - \gamma \boldsymbol{p}_j(t)dt + \sqrt{2kT\gamma m_j}d\boldsymbol{w}_j \ , \end{cases} \qquad j = 1, \cdots, N \tag{1}$$

where $\boldsymbol{q} = (\boldsymbol{q}_1, \cdots, \boldsymbol{q}_N)$ and $\boldsymbol{p} = (\boldsymbol{p}_1, \cdots, \boldsymbol{p}_N)$ denote the positions and momenta of the particles, kT is the temperature factor, and $\{\boldsymbol{w}_j\}_{j=1}^N$ are N-independent Brownian motions. The last two terms in the second equation in (1) represent the effect of a heat bath with parameter γ. In Langevin dynamics, positions are differentiable, and due to the irregularity of the Brownian force, momenta are just continuous, but not differentiable. This difference in regularity explains why the first equation in (1) is written as an ordinary differential equation (ODE) and the second equation is written as a stochastic differential equation (SDE).

The bath-free dynamics is a Hamiltonian system with the following Hamiltonian energy function:

$$H(\boldsymbol{q}, \boldsymbol{p}) = \sum_{j=1}^{N} \frac{1}{2m_j}|\boldsymbol{p}_j|^2 + U(\boldsymbol{q}) \tag{2}$$

Since the masses are constant, this Hamiltonian nicely separates into a kinetic and potential energy that are purely functions of $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively. The stationary probability density of the solution to Equation (1) is the Boltzmann-Gibbs density given by:

$$\nu(\boldsymbol{q}, \boldsymbol{p}) = Z^{-1} \exp\left(-\frac{1}{kT}H(\boldsymbol{q}, \boldsymbol{p})\right) \ , \ \ Z = \int \exp\left(-\frac{1}{kT}H(\boldsymbol{q}, \boldsymbol{p})\right) d\boldsymbol{q}d\boldsymbol{p} \tag{3}$$

Let h be a given time step size and $\boldsymbol{m} = \mathrm{diag}(m_1, \cdots, m_N)$. Let $(\boldsymbol{Q}_0, \boldsymbol{P}_0)$ denote the position and momentum of the molecular system at time $t > 0$. The simplest approximation to Equation (1) is a forward Euler discretization or Euler-Maruyama scheme [85] that computes an updated position and momentum $(\boldsymbol{Q}_1, \boldsymbol{P}_1)$ at $t + h$ using:

$$\begin{aligned} \boldsymbol{Q}_1 &= \boldsymbol{Q}_0 + h\boldsymbol{m}^{-1}\boldsymbol{P}_0 \\ \boldsymbol{P}_1 &= \boldsymbol{P}_0 - h\nabla U(\boldsymbol{Q}_0) - h\gamma \boldsymbol{P}_0 + \sqrt{h}\sqrt{2kT\gamma}\boldsymbol{m}^{1/2}\boldsymbol{\xi} \end{aligned} \qquad \text{(forward Euler)}$$

Here, $\boldsymbol{\xi} \in \mathbb{R}^n$ denotes a Gaussian random vector with mean zero and covariance $\mathbb{E}(\boldsymbol{\xi}_i\boldsymbol{\xi}_j) = \boldsymbol{\delta}_{ij}$. The problem with this approximation is that the forward Euler method is known to diverge in finite-time when the derivatives of the potential are unbounded, which is the norm in MD simulation. The precise statement and proof of divergence in a general setting can be found in [86]. By far the most computationally intensive part of the time-stepping algorithm is the evaluation of the potential force. Thus, we will restrict our discussion to schemes that, like Euler, only require a single force field evaluation per step.

An improvement to the forward Euler method is the following two-step scheme:

$$\boldsymbol{Q}_2 = (1+e^{-\gamma h})\boldsymbol{Q}_1 - e^{-\gamma h}\boldsymbol{Q}_0 + \frac{1-e^{-\gamma h}}{\gamma}\boldsymbol{m}^{-1}\left(-h\nabla U(\boldsymbol{Q}_1) + \sqrt{h}\sqrt{2kT\gamma}\boldsymbol{m}^{1/2}\boldsymbol{\xi}\right) \qquad \text{(BBK)}$$

In the limit, $\gamma \to 0$, this scheme reduces to the well-known Verlet integrator for MD simulation [7]. Just like Verlet, this integrator defines a map on pairs of molecular system configurations. Substituting the approximation, $e^{-\gamma h} \approx (1-\gamma h/2)/(1+\gamma h/2)$, into the above yields the Brünger-Brooks-Karplus (BBK) scheme, as appearing in [35]. Like the forward Euler method, this method is explicit and only requires one new force evaluation per step.

Second-order accurate schemes that generalize the Velocity Verlet integrator to Langevin dynamics were proposed in a sequence of papers [42–44,87,88]. Here, we mention two of these schemes that are both Strang splittings of Equation (1). The first was proposed by Ricci and Ciccotti [42] and consists of the following sub-steps:

$$\underbrace{\begin{pmatrix}\dot{\boldsymbol{q}}(t) = \boldsymbol{m}^{-1}\boldsymbol{p}(t)\\ d\boldsymbol{p}(t) = \boldsymbol{0}\end{pmatrix}}_{\text{exactly evolve by } 1/2 \text{ a step}} \circ \underbrace{\begin{pmatrix}\dot{\boldsymbol{q}}(t) = \boldsymbol{0}\\ d\boldsymbol{p}(t) = -\nabla U(\boldsymbol{q}(t))dt - \gamma\boldsymbol{p}(t)dt + \sqrt{2kT\gamma}\boldsymbol{m}^{1/2}d\boldsymbol{W}\end{pmatrix}}_{\text{exactly evolve by a step}} \circ \underbrace{\begin{pmatrix}\dot{\boldsymbol{q}}(t) = \boldsymbol{m}^{-1}\boldsymbol{p}(t)\\ d\boldsymbol{p}(t) = \boldsymbol{0}\end{pmatrix}}_{\text{exactly evolve by } 1/2 \text{ a step}}$$

Each step in this decomposition can be exactly solved. Clearly, the half-steps are easy to solve, since momentum is constant over each of these half-steps. The SDE appearing in the inner step can also be exactly solved, since it is linear in momentum (see Chapter 5 in [89]). This splitting is quite natural, since it treats the heat bath forces in the same way as the potential forces.

A related, but different, splitting method was proposed by Bussi and Parinello in [43] and is given by:

$$\underbrace{\begin{pmatrix}\dot{\boldsymbol{q}}(t) = \boldsymbol{0}\\ d\boldsymbol{p}(t) = -\gamma\boldsymbol{p}(t)dt + \sqrt{2kT\gamma}\boldsymbol{m}^{1/2}d\boldsymbol{W}\end{pmatrix}}_{\text{exactly evolve by } 1/2 \text{ a step}} \circ \underbrace{\begin{pmatrix}\dot{\boldsymbol{q}}(t) = \boldsymbol{m}^{-1}\boldsymbol{p}(t)\\ \dot{\boldsymbol{p}}(t) = -\nabla U(\boldsymbol{q}(t))\end{pmatrix}}_{\substack{\text{approximately evolve}\\ \text{using a step of Verlet}}} \circ \underbrace{\begin{pmatrix}\dot{\boldsymbol{q}}(t) = \boldsymbol{0}\\ d\boldsymbol{p}(t) = -\gamma\boldsymbol{p}(t)dt + \sqrt{2kT\gamma}\boldsymbol{m}^{1/2}d\boldsymbol{W}\end{pmatrix}}_{\text{exactly evolve by } 1/2 \text{ a step}}$$

Notice that this decomposition splits the Langevin dynamics into its Hamiltonian and heat bath parts, which makes it easy to analyze the structural properties of the scheme. A Velocity Verlet integrator is used to approximate the Hamiltonian dynamics. This approximation exactly preserves phase space volume and preserves energy to third-order accuracy per step. Moreover, the solution to the SDE appearing in the half-steps exactly preserves the Boltzmann-Gibbs density.

Since the Velocity Verlet integrator does not exactly preserve energy, the composition above does not exactly preserve the stationary distribution with density in Equation (3). In [90], it was shown that if the derivatives of the potential are all bounded, the Bussi and Parinello integrator possesses an invariant measure that is $\mathcal{O}(h^2)$ close to the Boltzmann-Gibbs distribution. In this same context, the leading order error term in the integrator's approximation to the invariant measure was explicitly determined [91]. Technically speaking, however, these results do not directly apply to MD simulation, since real MD simulation involves potentials whose derivatives are unbounded, e.g., Lennard-Jones forces. As a consequence of this irregularity in the force fields and discretization error, explicit schemes, like this one, may either not detect features of the potential energy properly, which leads to unnoticed, but large errors in dynamic quantities such as the mean first passage time, or may mishandle

soft- or hard-core potentials, which leads to numerical instabilities; see the numerical examples in [92]. These numerical artifacts motivate adding a Metropolis accept/refusal sub-step to the integrator. In the next section, we show how to Metropolize all of the MD integrators presented in this section. In Section 5, we explain how to generalize the Metropolis-corrected Bussi and Parinello algorithm to a larger class of diffusion processes.

3. Metropolis-Corrected MD Integrators

Here, we show how to add a Metropolis acceptance-rejection step to a BBK-type scheme and the Bussi and Parinello splitting scheme and then precisely state the properties of these integrators. We start with a detailed description of each algorithm. Both algorithms require evaluating the acceptance probability given by the usual Metropolis ratio:

$$\alpha(\boldsymbol{q},\boldsymbol{p},\boldsymbol{Q},\boldsymbol{P}) = \min\left(1, \exp\left(-\frac{1}{kT}(H(\boldsymbol{Q},\boldsymbol{P}) - H(\boldsymbol{q},\boldsymbol{p}))\right)\right) \tag{4}$$

The procedure to Metropolize the Ricci and Ciccotti scheme can be found in Section 2 of [70].

Algorithm 3.1 (First-order BBK-type integrator)**.** Given the current state $(\boldsymbol{Q}_0, \boldsymbol{P}_0)$ at time t, the algorithm proposes a new state $(\boldsymbol{Q}_1^\star, \boldsymbol{P}_1^\star)$ at time $t+h$ for some time step $h > 0$ via:

$$\begin{pmatrix}\boldsymbol{Q}_1^\star\\ \boldsymbol{P}_1^\star\end{pmatrix} = \begin{pmatrix}\boldsymbol{Q}_0 + \boldsymbol{m}^{-1}\left(h\boldsymbol{P}_0 - \frac{h^2}{2}\nabla U(\boldsymbol{Q}_0)\right)\\ \boldsymbol{P}_0 - \frac{h}{2}\left(\nabla U(\boldsymbol{Q}_0) + \nabla U(\boldsymbol{Q}_1^\star)\right)\end{pmatrix} \tag{Step 1}$$

This "proposal move" $(\boldsymbol{Q}_1^\star, \boldsymbol{P}_1^\star)$ is then accepted or rejected:

$$\begin{pmatrix}\tilde{\boldsymbol{Q}}_1\\ \tilde{\boldsymbol{P}}_1\end{pmatrix} = x\begin{pmatrix}\boldsymbol{Q}_1^\star\\ \boldsymbol{P}_1^\star\end{pmatrix} + (1-x)\begin{pmatrix}\boldsymbol{Q}_0\\ -\boldsymbol{P}_0\end{pmatrix} \tag{Step 2}$$

where x is a Bernoulli random variable with parameter $\alpha(\boldsymbol{Q}_0, \boldsymbol{P}_0, \boldsymbol{Q}_1^\star, \boldsymbol{P}_1^\star)$ given by Equation (4). The actual update of the system is taken to be:

$$\begin{pmatrix}\boldsymbol{Q}_1\\ \boldsymbol{P}_1\end{pmatrix} = \begin{pmatrix}\tilde{\boldsymbol{Q}}_1\\ \exp(-\gamma h)\tilde{\boldsymbol{P}}_1 + \sqrt{kT}\sqrt{1-\exp(-2\gamma h)}\boldsymbol{m}^{1/2}\boldsymbol{\xi}\end{pmatrix} \tag{Step 3}$$

Here, $\boldsymbol{\xi} \in \mathbb{R}^n$ denotes a Gaussian random vector with mean zero and covariance $\mathbb{E}(\boldsymbol{\xi}_i\boldsymbol{\xi}_j) = kT\boldsymbol{\delta}_{ij}$.

The momenta of the molecules gets reversed if a move is rejected in Step 2 of Algorithm 3.1. This momentum flip is necessary for the algorithm to preserve the correct stationary distribution [70,71], but results in an $O(1)$ error in dynamics. High acceptance rates are therefore needed to ensure that the time lag between successive rejections is frequently long enough for the approximation to capture the desired dynamics. Since the acceptance rate in Equation (4) is related to how well the Verlet integrator in (Step 1) preserves energy after a single step, this rejection rate is $O(h^3)$. Thus, in practice, we find

that the time step required to obtain a sufficiently high acceptance rate is often automatically fulfilled by a time step that sufficiently resolves the desired dynamics. Each step of this algorithm requires: evaluating the atomic force field once in the third equation of (Step 1), generating a Bernoulli random variable with parameter α in (Step 2) and generating an n-dimensional Gaussian vector in (Step 3). We stress that (Step 2) in Algorithm 3.1 is all that is needed to get MD integrators to exactly preserve the Boltzmann-Gibbs density in Equation (3).

Next, we show how to Metropolize the Bussi and Parinello splitting integrator.

Algorithm 3.2 (Second-order Bussi and Parinello integrator)**.** Let $\boldsymbol{\xi}, \boldsymbol{\eta} \in \mathbb{R}^n$ be two independent Gaussian random vectors with mean zero and covariance $\mathbb{E}(\boldsymbol{\xi}_i \boldsymbol{\xi}_j) = \mathbb{E}(\boldsymbol{\eta}_i \boldsymbol{\eta}_j) = \boldsymbol{\delta}_{ij}$. Given a time step size h and the current state $(\boldsymbol{Q}_0, \boldsymbol{P}_0)$ at time t, the algorithm takes a half-step of the heat bath dynamics:

$$\begin{pmatrix} \tilde{\boldsymbol{Q}}_0 \\ \tilde{\boldsymbol{P}}_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q}_0 \\ \exp(-\gamma h/2)\tilde{\boldsymbol{P}}_0 + \sqrt{kT}\sqrt{1-\exp(-\gamma h)}\boldsymbol{m}^{1/2}\boldsymbol{\xi} \end{pmatrix} \tag{Step 1}$$

Followed by a full step of Verlet to compute a proposal move $(\tilde{\boldsymbol{Q}}_1^\star, \tilde{\boldsymbol{P}}_1^\star)$:

$$\begin{pmatrix} \tilde{\boldsymbol{Q}}_1^\star \\ \tilde{\boldsymbol{P}}_1^\star \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{Q}}_0 + \boldsymbol{m}^{-1}\left(h\tilde{\boldsymbol{P}}_0 - \frac{h^2}{2}\nabla U(\tilde{\boldsymbol{Q}}_0)\right) \\ \boldsymbol{P}_0 - \frac{h}{2}\left(\nabla U(\tilde{\boldsymbol{Q}}_0) + \nabla U(\tilde{\boldsymbol{Q}}_1^\star)\right) \end{pmatrix} \tag{Step 2}$$

This proposal move $(\tilde{\boldsymbol{Q}}_1^\star, \tilde{\boldsymbol{P}}_1^\star)$ is then accepted or rejected:

$$\begin{pmatrix} \tilde{\boldsymbol{Q}}_1 \\ \tilde{\boldsymbol{P}}_1 \end{pmatrix} = x \begin{pmatrix} \tilde{\boldsymbol{Q}}_1^\star \\ \tilde{\boldsymbol{P}}_1^\star \end{pmatrix} + (1-x)\begin{pmatrix} \tilde{\boldsymbol{Q}}_0 \\ -\tilde{\boldsymbol{P}}_0 \end{pmatrix} \tag{Step 3}$$

where x is a Bernoulli random variable with parameter $\alpha(\tilde{\boldsymbol{Q}}_0, \tilde{\boldsymbol{P}}_0, \tilde{\boldsymbol{Q}}_1^\star, \tilde{\boldsymbol{P}}_1^\star)$ given by Equation (4). The actual update of the system at time $t+h$ is taken to be:

$$\begin{pmatrix} \boldsymbol{Q}_1 \\ \boldsymbol{P}_1 \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{Q}}_1 \\ \exp(-\gamma h/2)\tilde{\boldsymbol{P}}_1 + \sqrt{kT}\sqrt{1-\exp(-\gamma h)}\boldsymbol{m}^{1/2}\boldsymbol{\eta} \end{pmatrix} \tag{Step 4}$$

This algorithm requires generating two independent n-dimensional Gaussian vectors per step. Thus, it is more costly than Algorithm 3.1. However, the advantage of doing this is that the resulting Metropolis corrected algorithm is second-order weakly accurate, as the following Proposition states.

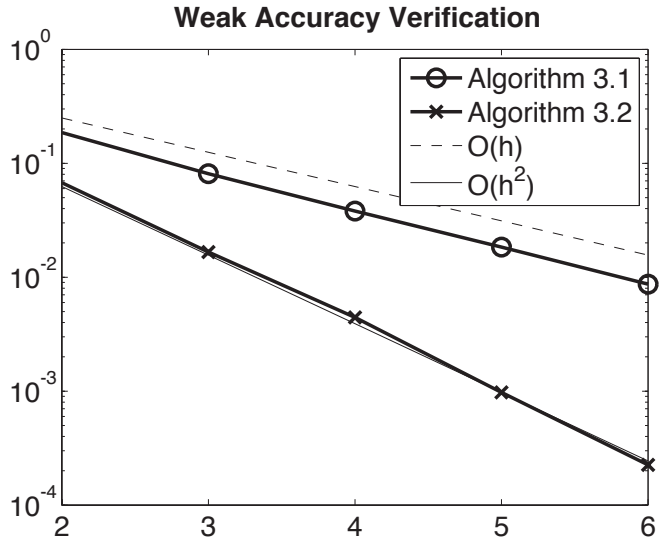
Proposition 3.3. *Let $(\boldsymbol{Q}_n, \boldsymbol{P}_n)$ represent the numerical approximation produced by Algorithm 3.2 at time nh with the same initial condition as the true solution: $(\boldsymbol{Q}_0, \boldsymbol{P}_0) = (\boldsymbol{q}(0), \boldsymbol{p}(0))$. For every time interval $T > 0$ and for suitable observables $f(\boldsymbol{q}, \boldsymbol{p})$, there exists a $C(T) > 0$, such that:*

$$|\mathbb{E}f(\boldsymbol{q}(\lfloor t/h \rfloor h), \boldsymbol{p}(\lfloor t/h \rfloor h)) - \mathbb{E}f(\boldsymbol{Q}_{\lfloor t/h \rfloor}, \boldsymbol{P}_{\lfloor t/h \rfloor})| \leq C(T)h^2 \tag{5}$$

for all $t < T$.

This accuracy concept is sufficient for computing means and correlation functions at finite-time and equilibrium correlations. Figure 2 verifies this Proposition by checking the weak accuracy of Algorithms 3.1 and 3.2 on a harmonic oscillator test problem.

Figure 2. Langevin dynamics of a harmonic oscillator.



To be specific, Figure 2 plots the weak accuracy of the Metropolis-corrected MD integrators with respect to the true solution of the Langevin dynamics of a harmonic oscillator: $\dot{q}(t) = p(t)$, $dp(t) = -q(t) - p(t) + \sqrt{2}dw(t)$, with initial condition $q(0) = 1.0$, $p(0) = 0$. The time steps tested are $h = 2^{-n}$, where n is given on the x-axis. The quantity monitored for the error is the estimate of $\mathbb{E}(q(1)^2 + p(1)^2) = 1.699445410$ computed analytically. The dashed and solid curves are the graphs of $2^{-n}(= h)$ and $2^{-2n}(= h^2)$ *versus* n, respectively.

Proof. The desired single-step error estimate can be obtained from an application of the triangle inequality:

$$|\mathbb{E}f(\boldsymbol{q}(h),\boldsymbol{p}(h))-\mathbb{E}f(\boldsymbol{Q}_1,\boldsymbol{P}_1)| \leq |\mathbb{E}f(\boldsymbol{q}(h),\boldsymbol{p}(h))-\mathbb{E}f(\hat{\boldsymbol{Q}}_1,\hat{\boldsymbol{P}}_1)|+|\mathbb{E}f(\hat{\boldsymbol{Q}}_1,\hat{\boldsymbol{P}}_1)-\mathbb{E}f(\boldsymbol{Q}_1,\boldsymbol{P}_1)| \quad (6)$$

where $(\hat{\boldsymbol{Q}}_1,\hat{\boldsymbol{P}}_1)$ denotes one step of the uncorrected Bussi and Parinello scheme with $(\hat{\boldsymbol{Q}}_0,\hat{\boldsymbol{P}}_0) = (\boldsymbol{q}(0),\boldsymbol{p}(0))$. The first term in the upper bound in Equation (6) is $O(h^3)$, since the unadjusted scheme is a Strang splitting of Equation (1). To bound the second term in Equation (6), note that:

$$\mathbb{E}f(\boldsymbol{Q}_1,\boldsymbol{P}_1) - \mathbb{E}f(\hat{\boldsymbol{Q}}_1,\hat{\boldsymbol{P}}_1) = \mathbb{E}\left\{\left(\bar{f}(\tilde{\boldsymbol{Q}}_1^\star,\tilde{\boldsymbol{P}}_1^\star) - \bar{f}(\tilde{\boldsymbol{Q}}_0,-\tilde{\boldsymbol{P}}_0)\right)\left(\alpha(\tilde{\boldsymbol{Q}}_0,\tilde{\boldsymbol{P}}_0,\tilde{\boldsymbol{Q}}_1^\star,\tilde{\boldsymbol{P}}_1^\star) - 1\right)\right\}$$

where we have introduced the auxilary function:

$$\bar{f}(\boldsymbol{q},\boldsymbol{p}) = \mathbb{E}f(\boldsymbol{q},\exp(-\gamma h/2)\boldsymbol{p} + \sqrt{kT}\sqrt{1-\exp(-\gamma h)}\boldsymbol{m}^{1/2}\boldsymbol{\eta})$$

Since the rejection rate is $\mathcal{O}(h^3)$, it follows from the above expression that the second term in the upper bound of Equation (6) is also $O(h^3)$. Standard results in numerical analysis for SDEs then imply that the algorithm converges weakly on finite-time intervals with global order two; see, for instance, [93] (Chapter 2.2). □

For completeness sake, we also provide a statement that both algorithms are ergodic.

Proposition 3.4. *Let* $(\boldsymbol{Q}_n, \boldsymbol{P}_n)$ *be the numerical approximation produced by Algorithms 3.1 or 3.2 at time* nh*. Then, for suitable observables* $f(\boldsymbol{q}, \boldsymbol{p})$*:*

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T f(\boldsymbol{Q}_{\lfloor t/h \rfloor}, \boldsymbol{P}_{\lfloor t/h \rfloor}) dt \to \int_{\mathbb{R}^{2n}} f(\boldsymbol{q}, \boldsymbol{p}) \nu(\boldsymbol{q}, \boldsymbol{p}) d\boldsymbol{q} d\boldsymbol{p} \tag{7}$$

Here, $\nu(\boldsymbol{q}, \boldsymbol{p})$ *denotes the Boltzmann-Gibbs density defined in Equation* (3)*.*

A proof of this Proposition can be found in [72].

4. Application to Lennard-Jones Fluid

Listing 1 translates Algorithm 3.2 into the MATLAB language. Intrinsically defined MATLAB functions appear in boldface. The algorithm uses MATLAB's built in random number generators to carry out Step 1, Step 3 and Step 4. In particular, the Bernoulli random variable, x, in Step 3 is generated in Line 20, and the Gaussian vectors in Step 1 and Step 4 are generated on Line 9 and Line 29, respectively. In addition to updating the positions and momenta of the system, the program also stores the previous value of the potential energy and force, so that the force and potential energy is evaluated in Line 15 just once per simulation step. This evaluation calls a MEX function, which inputs the current position of the molecular system and outputs the force field and potential energy at that position. We use a MEX function, because the atomistic force field evaluation cannot be easily vectorized and is, by far, the most computationally demanding step in MD. The `PreProcessing` script file called in Line 2 defines the physical and numerical parameters, sets the initial condition and allocates space for storing simulation data. Sample averages are updated as new points on the trajectory are produced in the `UpdateSampleAverages` script file invoked in Line 35. Finally, the outputs produced by the algorithm are handled by the `PostProcessing` script file in Line 39.

Let us consider a concrete example: a Lennard-Jones fluid that consists of N identical atoms [1–3]. The configuration space of this system is a fixed cubic box with periodic boundary conditions. The distance between the i-th and j-th particle is defined according to the minimum image convention, which states that the distance between $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$ in a cubic box of length ℓ is:

$$d_{MD}(\boldsymbol{q}_i, \boldsymbol{q}_j) \stackrel{\text{def}}{=} |(\boldsymbol{q}_i - \boldsymbol{q}_j) - \ell \lfloor (\boldsymbol{q}_i - \boldsymbol{q}_j)/\ell \rceil| \tag{8}$$

where $\lfloor \cdot \rceil$ is the nearest integer function. In terms of this distance, the total potential energy is a sum over all pairs:

$$U(\boldsymbol{q}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} U_{LJ}(d_{MD}(\boldsymbol{q}_i, \boldsymbol{q}_j)) \tag{9}$$

where $U_{LJ}(r)$ is the following truncated Lennard-Jones potential function:

$$U_{LJ}(r) = \begin{cases} f(r) - f(r_c), & r < r_c \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Listing 1. Metropolized MD Integrator: `MDintegrator.m`

```
1
2 PreProcessing;
3
4 for i = 1:Ns
5
6     %--- Step 1 --- Heat Bath Step
7
8     tQ0=Q0;
9     tP0=f1*P0+f2*randn(3*Nm,1);
10
11    %--- Step 2 --- Velocity Verlet Proposal
12
13    Ppt5=tP0+0.5*h*F0;
14    tQ1star=tQ0+h*Ppt5;
15    [tF1star,tU1star]=ForceFieldmex(tQ1star,Nm,rcut2,ell);
16    tP1star=Ppt5+0.5*h*tF1star;
17
18    %--- Step 3 --- Accept or Refuse Step
19
20    x=(rand<exp(-(0.5*tP1star'*tP1star-0.5*tP0'*tP0+tU1star-U0)/kT));
21
22    tP1=x*P1star-(1-x)*P0;
23    tQ1=x*Q1star+(1-x)*Q0;
24    F1=x*tF1star+(1-x)*F0; U1=x*tU1star+(1-x)*U0;
25
26    %--- Step 4 --- Heat Bath Step
27
28    Q1=tQ1;
29    P1=f1*tP1+f2*randn(3*Nm,1);
30
31    %--- iterate
32
33    Q0=Q1; P0=P1; F0=F1; U0=U1;
34
35    UpdateSampleAverages;
36
37 end
38
39 PostProcessing;
```

Listing 2. Metropolized MD Integrator: `PreProcessing.m`

```
%--- seed random # generator

rng(123);

%--- physical parameters

rho=0.6;                    % density
kT=0.5;                     % temperature factor
gama=0.1;                   % heat bath parameter
Nm=500;                     % # of molecules
T=2.0;                      % time span for velocity correlation
ell=(Nm/rho)^(1/3);         % length of cubic box

%--- simulation parameters

h=0.005;                    % time-step size
Ns=1e3;                     % # of steps
rcut = 2.0^(1/6);           % cutoff radius
rcut2 = rcut*rcut;

f1=exp(-0.5*gama*h); f2=sqrt((1.0-exp(-gama*h))*kT);

%--- initial condition

A=fcclattice(Nm,ell);
Q0=reshape(A, [3*Nm 1]);    % atoms on an fcc lattice
P0=zeros(3*Nm,1);           % atoms at rest

%--- initialize statistics

NA=ceil(T/h)+1;             % preallocate space for
acf=zeros(NA,1);            % online correlation computation
varacf=zeros(NA,1);
pivot=zeros(NA,3*Nm);
nacf=zeros(NA,1);

AP=zeros(Ns,1);             % vector of acceptance probabilities

[F0,U0]=ForceFieldmex(Q0,Nm,rcut2,ell);  % initial force & energy
```

Here, $f(r) = 4(1/r^{12} - 1/r^6)$ and r_c is the cutoff radius, which is bounded above by the size of the simulation box; and we have used dimensionless units to describe this system, where energy is rescaled by the depth of the Lennard-Jones potential energy and length by the point where the potential energy is zero. The error introduced by the truncation in Equation (10) is proportional to the density of the molecular system and can be made arbitrarily small by selecting the cutoff distance to be sufficiently large. A direct evaluation of the potential force, $\nabla U(\boldsymbol{q})$, scales like $O(N^2)$, and typically dominates the total computational cost. In practice, neighbor/cell lists, also called Verlet lists, are used in order to obtain a force evaluation that scales linearly with system size. Since the system we consider will have just a few hundred atoms, there is, however, little advantage to using these data structures, or using a fast force field evaluation, and thus, `ForceFieldmex` evaluates the force and energy using a sum over all particle pairs.

Table 1. Simulation parameters.

	Parameter	Description	Value
Physical Parameters	ρ	density	$\{0.6, 0.7, 0.8, 0.9, 1.0, 1.1\}$
	kT	temperature factor	0.5
	γ	heat bath parameter	0.01
	N_m	# of molecules	512
	T	time-span for autocorrelation	2
Numerical Parameters	h	time step	0.005
	N_s	# of simulation steps	10^5
	r_c	Lennard-Jones force cutoff radius	$2^{1/6}$

Listing 2 shows the `PreProcessing` script, which sets the parameters provided in Table 1 and constructs the initial condition, where the N atoms are assumed to be at rest and on the sites of a face-centered cubic lattice. The command, `rng(123)`, on *Line 3* sets the seed of the random number generator functions, `RAND` and `RANDN`. The acceptance rates at every step and the velocity autocorrelation are updated in the `UpdateSampleAverages` script shown in Listing 3. The mean acceptance rate, which is outputted in the `PostProcessing` script shown in Listing 4, must be high enough to ensure that the dynamics is accurately represented. To compute the autocorrelation of an observable over a time interval of length T, the value of that observable along the entire trajectory is not needed. In fact, it suffices to use the values of this observable along a piece of trajectory over a moving time-window $[t_i, t_i + T]$, where $t_i = i \times h$. This storage space is allocated in `PreProcessing` and is updated in `UpdateSampleAverages`. More precisely, the molecular velocities are stored in the `pivot` array from $i - N_a$ to i, where i is the index of the current position and $N_a = \lceil T/h \rceil + 1$. Notice that velocity autocorrelations are not computed until after the index, i, exceeds 10^4. This *equilibration time* removes some of the statistical bias that may arise from using a non-random initial condition. Short-time trajectories of this molecular system are plotted in Figure 3 from an initial condition where atoms are placed on the sites of a face-centered cubic lattice and at rest. The trajectory is computed using the numerical and physical parameters indicated in Table 1, with the exception of the number of steps,

which is set equal to $N_s = 1000$. Notice that at lower densities particle trajectories are more diffusive and less localized. Using the parameters provided in Table 1, we compute velocity autocorrelations for a range of density values in Figure 4. Since the heat bath parameter is set to a small value, these figures are in qualitative agreement with those obtained by simulating the molecular system with no heat bath as shown in Figure 5.2 of [3].

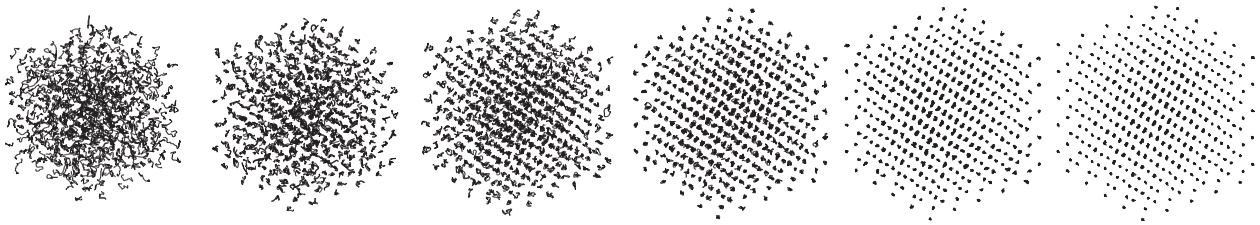
Listing 3. Metropolized MD Integrator: `UpdateSampleAverages.m`

```
%--- store acceptance probability

AP(i)=x;

%--- update correlation function

if (i>1e4)

    pp=mod(i-1,NA)+1;
    pivot(pp,:)=P0;

    for j=1:min(i,NA)
        nacf(j)=nacf(j)+1;
        mui=acf(j);
        vari=varacf(j);
        n_samples=nacf(j);
        xip1=pivot(mod(pp-j,NA)+1,:)*pivot(pp,:)'/(3.0*Nm);
        acf(j)=mui+(xip1-mui)/n_samples;
        varacf(j)=((n_samples-1)*vari+...
            (xip1-mui)*(xip1-acf(j)))/n_samples;
    end

end
```

Listing 4. Metropolized MD Integrator: `PostProcessing.m`

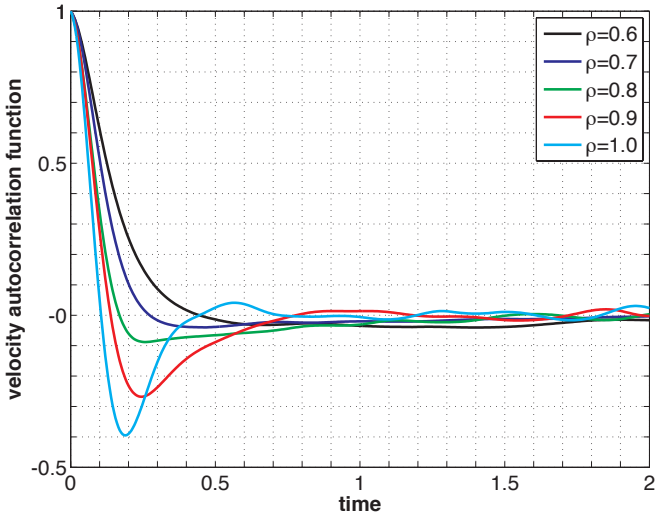
```
%--- output results

disp(['h=' num2str(h)  ',<AP>=' num2str(mean(AP))]);

figure(2); clf; hold on; tt=0:h:T;
errorbar(tt,acf,1.96*sqrt(varacf)./sqrt(nacf));

save('VelocityAutocorrelation.mat', 'tt', 'acf', 'varacf');
```

Figure 3. Atomic trajectories in a simulation box.



(**a**) $\rho = 0.6$ (**b**) $\rho = 0.7$ (**c**) $\rho = 0.8$ (**d**) $\rho = 0.9$ (**e**) $\rho = 1.0$ (**f**) $\rho = 1.1$

Figure 4. Soft-sphere velocity autocorrelation functions. A reproduction of Figure 5.2 of [3] using Langevin dynamics with heat bath parameter $\gamma = 0.01$. The remaining parameters are set equal to those provided in Table 1. The negative correlations at higher densities are consistent with what has been found in the literature [6,8].



5. General Case

Here, we show how the preceding ideas extend to other molecular systems that obey stochastic differential equations. In the process, we generalize the Metropolized Bussi and Parinello integrator (Algorithm 3.2) to a big class of diffusion processes, including the v-rescale thermostat. We begin with the underlying Hamiltonian dynamics of a molecular system.

5.1. Bath-Free Dynamics

MD is based on Hamilton's equations for a Hamiltonian $H : \mathbb{R}^{2d} \to \mathbb{R}$:

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{J}\nabla H(\boldsymbol{z}(t)) \,, \quad \boldsymbol{z}(0) \in \mathbb{R}^{2d} \tag{11}$$

where $\boldsymbol{z}(t) = (\boldsymbol{q}(t), \boldsymbol{p}(t))$ is a vector of molecular positions $\boldsymbol{q}(t) \in \mathbb{R}^d$ and momenta $\boldsymbol{p}(t) \in \mathbb{R}^d$ and $\boldsymbol{J}$ is the $2d \times 2d$ skew-symmetric matrix defined as:

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{0}_{d\times d} & \boldsymbol{I}_{d\times d} \\ -\boldsymbol{I}_{d\times d} & \boldsymbol{0}_{d\times d} \end{pmatrix} \tag{12}$$

The Hamiltonian, $H(\boldsymbol{z})$, represents the total energy of the molecular system and is typically "separable", meaning that it can be written as:

$$H(\boldsymbol{z}) = K(\boldsymbol{p}) + U(\boldsymbol{q}) \ , \quad \boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{p}) \tag{13}$$

where $K(\boldsymbol{p})$ and $U(\boldsymbol{q})$ are the kinetic and potential energy functions, respectively [94]. In MD, the kinetic energy function is a positive definite quadratic form, and the potential energy function involves "fudge factors" determined from experimental or quantum mechanical studies of pieces of the molecular system of interest [36]. The accuracy of the resulting energy function must be systematically verified by comparing MD simulation data to experimental data [95]. The flow that Equation (11) determines has the following structure:

(S1) volume-preserving (since the vector-field in Equation (11) is divergenceless); and

(S2) energy-preserving (since $\boldsymbol{J}$ is skew-symmetric and constant).

Explicit *symplectic integrators*, like the Verlet scheme, exploit these properties to obtain long-time stable schemes for Hamilton's equations [96,97].

5.2. Governing Stochastic Dynamics

In order to mimic experimental conditions, Equation (11) is often coupled to a bath that puts the system at constant temperature and/or pressure. The standard way to do this is to assume that the system with a bath is governed by a stochastic ordinary differential equation (SDE) of the type:

$$\boxed{d\boldsymbol{Y}(t) = \underbrace{\boldsymbol{A}(\boldsymbol{Y}(t))dt}_{\text{deterministic drift}} + \underbrace{(\operatorname{div} \boldsymbol{D})(\boldsymbol{Y}(t))dt + \sqrt{2kT}\boldsymbol{B}(\boldsymbol{Y}(t))d\boldsymbol{W}(t)}_{\text{heat bath}}} \tag{14}$$

Here, we have introduced the following notation.

$\boldsymbol{Y}(t) \in \mathbb{R}^n$	state of the (extended) system
$\boldsymbol{A}(\boldsymbol{x}) \in \mathbb{R}^n$	deterministic drift vector field
$\boldsymbol{B}(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$	noise-coefficient matrix
$\boldsymbol{D}(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$	diffusion matrix
$\boldsymbol{W}(t) \in \mathbb{R}^n$	n-dimensional Brownian motion
kT	temperature factor

The $n \times n$ *diffusion matrix*, $\boldsymbol{D}(\boldsymbol{x})$, is defined in terms of the noise coefficient matrix, $\boldsymbol{B}(\boldsymbol{x})$, as:

$$\boldsymbol{D}(\boldsymbol{x}) \stackrel{\text{def}}{=} kT\boldsymbol{B}(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x})^T \ , \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^n \tag{15}$$

where $\boldsymbol{B}(\boldsymbol{x})^T$ denotes the transpose of the real matrix, $\boldsymbol{B}(\boldsymbol{x})$. The diffusion matrix is symmetric and nonnegative definite. Depending on the particular bath that is used, the dimension, n, of $\boldsymbol{Y}(t)$ in Equation (14) is related to the dimension, $2d$, of $\boldsymbol{z}(t)$ in Equation (11) by the inequality: $n \geq 2d$. For example, in Nosé-Hoover Langevin dynamics, a single bath degree of freedom is added to Equation (11), so that $n = 2d + 1$, while in Langevin dynamics, the effect of the bath is modeled by added friction and Brownian forces that keep $n = 2d$. The Langevin Equation (1) can be put in the form of Equation (14) by letting $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{p})$,

$$\boldsymbol{A}(\boldsymbol{x}) = \begin{pmatrix} \boldsymbol{m}^{-1}\boldsymbol{p} \\ -\nabla U(\boldsymbol{q}) - \gamma \boldsymbol{p} \end{pmatrix} \ , \ \boldsymbol{B} = \sqrt{\gamma} \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{m}^{1/2} \end{pmatrix} \ , \text{ and } \boldsymbol{W} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_N) \tag{16}$$

where $\boldsymbol{m} = \mathrm{diag}(m_1, \cdots, m_N)$.

Equation (14) generates a stochastic process, $\boldsymbol{Y}(t)$, that is a Markov diffusion process. We assume that this diffusion process admits a *stationary* distribution $\mu(d\boldsymbol{x})$, *i.e.*, a probability distribution preserved by the dynamics [98,99]. We denote by $\nu(\boldsymbol{x})$ the density of this distribution. Even though the diffusion matrix in Equation (15) is not necessarily positive definite, one can use the Hörmander's condition to prove that the process, $\boldsymbol{Y}(t)$, is an ergodic process with a unique stationary distribution [100,101]. By the ergodic theorem, it then follows that:

$$\frac{1}{T}\int_0^T f(\boldsymbol{Y}(t))dt \to \int_{\mathbb{R}^n} f(\boldsymbol{x})\nu(\boldsymbol{x})d\boldsymbol{x} \ , \quad \text{as } T \to \infty, \quad \text{a.s.} \tag{17}$$

where $f(\boldsymbol{x})$ is a suitable test function.

The evolution of the probability density of the law of $\boldsymbol{Y}(t)$ at time t, $\rho(t, \boldsymbol{x})$, satisfies the Fokker-Planck equation:

$$-\frac{\partial \rho}{\partial t} + L\rho = 0 \tag{18}$$

where $\rho(0, \cdot)$ is the density of the initial distribution, $\boldsymbol{Y}(0) \sim \rho(0, \cdot)$, and L is defined as the following second-order partial differential operator:

$$(Lf)(\boldsymbol{x}) \stackrel{\text{def}}{=} \mathrm{div}\left(\mathrm{div}(\boldsymbol{D}(\boldsymbol{x})f(\boldsymbol{x})) - \boldsymbol{A}(\boldsymbol{x})f(\boldsymbol{x})\right) \tag{19}$$

Since $\mu(d\boldsymbol{x}) = \nu(\boldsymbol{x})d\boldsymbol{x}$ is a stationary distribution of $\boldsymbol{Y}(t)$, the probability density, $\nu(\boldsymbol{x})$, is a steady-state solution of Equation (18), *i.e.*, it satisfies:

$$(L\nu)(\boldsymbol{x}) = 0 \tag{20}$$

Define the *probability current* as the vector field:

$$\boldsymbol{j}(\boldsymbol{x}) \stackrel{\text{def}}{=} \mathrm{div}(\boldsymbol{D}(\boldsymbol{x})\nu(\boldsymbol{x})) - \boldsymbol{A}(\boldsymbol{x})\nu(\boldsymbol{x}) \tag{21}$$

The stationarity condition in Equation (20) implies that $\boldsymbol{j}(\boldsymbol{x})$ is divergenceless. In the zero-current case, the diffusion process, $\boldsymbol{Y}(t)$, is *reversible*, and the stationary density $\nu(\boldsymbol{x})$ is called the equilibrium probability density of the diffusion [102].

In this case, the operator, L, is self-adjoint, in the sense that:

$$\langle Lf, g\rangle_\nu = \langle f, Lg\rangle_\nu \qquad \text{for all suitable test functions } f, g \tag{22}$$

where $\langle \cdot, \cdot \rangle_\nu$ denotes an L^2 inner product weighted by the density, $\nu(x)$. This property implies that the diffusion is ν-symmetric [103]:

$$\nu(\boldsymbol{x}) p_t(\boldsymbol{x}, \boldsymbol{y}) = \nu(\boldsymbol{y}) p_t(\boldsymbol{y}, \boldsymbol{x}) \qquad \text{for all } t > 0 \tag{23}$$

where $p_t(\boldsymbol{x}, \boldsymbol{y})$ denotes the transition probability density of $\boldsymbol{Y}(t)$. Indeed, Equation (22) is simply an infinitesimal version of Equation (23), which is referred to as the detailed balance condition. In the self-adjoint case, the drift is uniquely determined by the diffusion matrix and the stationary density $\nu(\boldsymbol{x})$:

$$\boldsymbol{j}(\boldsymbol{x}) = \boldsymbol{0} \implies \boldsymbol{A}(\boldsymbol{x}) = \frac{1}{\nu(\boldsymbol{x})} \operatorname{div}(\boldsymbol{D}(\boldsymbol{x}) \nu(\boldsymbol{x})) \tag{24}$$

Long-time stable explicit schemes adapted to this structure have been recently developed [92].

5.3. Splitting Approach to MD Simulation

We are now in a position to explain our general approach for deriving a long-time stable scheme for Equation (14). Crucial to our approach is that in MD simulation, we usually have a formula for a function proportional to the stationary density $\nu(\boldsymbol{x})$. Following [90], we can split Equation (14) into:

$$d\boldsymbol{Y} = -\boldsymbol{D}(\boldsymbol{Y}) \nabla H_\nu(\boldsymbol{Y}) dt + \operatorname{div} \boldsymbol{D}(\boldsymbol{Y}) dt + \sqrt{2kT} B(\boldsymbol{Y}) dW \tag{25}$$

$$\dot{\boldsymbol{Y}} = \boldsymbol{A}(\boldsymbol{Y}) + \boldsymbol{D}(\boldsymbol{Y}) \nabla H_\nu(\boldsymbol{Y}) \tag{26}$$

where we have introduced $H_\nu(\boldsymbol{x}) = -(\log \nu)(\boldsymbol{x})$. An *exact splitting method* preserves $\mu(d\boldsymbol{x})$. It is formed by taking the exact solution (in law) of Equation (25) in composition with the exact flow of Equation (26). The process produced by Equation (25) is self-adjoint with respect to $\nu(\boldsymbol{x})$. Moreover, the stationarity of $\nu(\boldsymbol{x})$ implies that the flow of the ODE (26) preserves it. Since each step is preservative, their composition is, too.

In place of the exact splitting, a Metropolized explicit integrator can be used for Equation (25) [92], and a measure-preserving scheme can be designed to solve the ODE [72,104]. In [92], explicit schemes are introduced for Equation (25) that: (i) sample the exact equilibrium probability density of the SDE when this density exists (*i.e.*, whenever $\nu(\boldsymbol{x})$ is normalizable); (ii) generates a weakly accurate approximation to the solution of Equation (14) at constant kT; (iii) acquire higher order accuracy in the small noise limit, $kT \to 0$; and (iv) avoid computing the divergence of the diffusion matrix $\boldsymbol{D}(\boldsymbol{x})$. Compared to the methods in [72], the main novelty of these schemes stems from (iii) and (iv). The resulting explicit splitting method is accurate, since it is an additive splitting of Equation (14); and typically ergodic when the continuous process is ergodic [72].

This type of splitting of Equation (14) is quite natural and has been used before in MD [43,87], dissipative particle dynamics [105,106] and the simulation of inertial particles [107]. Other closely related schemes for Equation (14) include Brünger-Brooks-Karplus (BBK) [35], van Gunsteren and Berendsen (vGB) [108] and the Langevin-Impulse (LI) methods [41] and quasi-symplectic integrators [44]. However, for general MD force fields, none of these explicit integrators are long-time stable. Our framework to stabilize explicit MD integrators is the Metropolis-Hastings algorithm.

5.4. Metropolis-Hastings Algorithm

A Metropolis-Hastings method is a Monte Carlo method for producing samples from a probability distribution, given a formula for a function proportional to its density [74,75]. The algorithm consists of two sub-steps: firstly, a proposal move is generated according to a transition density, $g(\boldsymbol{x}, \boldsymbol{y})$; and secondly, this proposal move is accepted or rejected with a probability:

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = 1 \wedge \frac{g(\boldsymbol{y}, \boldsymbol{x})\nu(\boldsymbol{y})}{g(\boldsymbol{x}, \boldsymbol{y})\nu(\boldsymbol{x})} \tag{27}$$

Standard results on Metropolis-Hastings methods can be used to classify this algorithm as ergodic [100,109,110].

6. Conclusions

This paper provided an algorithmic introduction to time integrators for MD simulation. A quick overview of existing algorithms was given. When the derivatives of the potential are bounded, it is well known that these integrators work just fine: they are convergent on finite-time intervals and possess an invariant measure that is nearby the Boltzmann-Gibbs density. However, in realistic MD simulation, the derivatives of the potential are unbounded. This lack of regularity can cause numerical instabilities or artifacts in explicit integrators. The paper demonstrated how a Metropolis acceptance-rejection step can be added to explicit MD integrators to mitigate some of these problems and, in principle, obtain long-time stable and finite-time accurate schemes. A MATLAB implementation of Metropolis-corrected MD integrators was provided and used to compute the velocity autocorrelation of a sea of Lennard-Jones particles at various densities between the solid and liquid phases. The paper did not provide an in-depth review of the theory of Metropolis integrators, which can be found elsewhere [72,73].

Calculating the force field at every step dominates the overall computational cost of MD simulation. These force fields involve: bonded interactions and non-bonded Lennard-Jones and electrostatic interactions. The calculation of bonded interactions is straightforward to vectorize and scales like $O(N)$. In addition, Lennard-Jones forces rapidly decay with interatomic distance. To a good approximation, every atom interacts only with neighbors within a sufficiently large ball. By using data structures, like neighbor lists and cell linked lists, these interactions can be calculated in $O(N)$ steps, and therefore, the Lennard-Jones interactions can be calculated in $O(N)$ steps [46]. On the other hand, the electrostatic energy between particles decays, like $1/r$, where r denotes an interatomic distance, which leads to long-range interactions between atoms. Unlike Lennard-Jones interaction, this interaction cannot be cutoff without introducing large errors. In this case, one can use sophisticated techniques, like the fast multipole method, to rigorously handle such interactions in $\mathcal{O}(N)$ steps [39,58].

However, the effect of these 'mathematical tricks' for fast calculation of the force field can become muted if the time step requirement for stability or accuracy becomes more severe in high dimension. This can happen in the Metropolis integrator, if the acceptance probability in Step 2 of Algorithm 3.1 or Step 3 of Algorithm 3.2 deteriorates in high dimension. The scaling of Metropolis algorithms has been quantified for the random walk Metropolis, hybrid Monte Carlo and Metropolis-adjusted Langevin algorithm (MALA) [111–115]. Since the acceptance probability is a function of an extensive quantity,

the acceptance rate can artificially deteriorate with increasing system size, unless the time step is reduced. Because high acceptance rates are required to maintain dynamic accuracy, the dependence of the time step on system size limits the application of Metropolized schemes to large-scale systems. Fortunately, this scalability issue can often be resolved by using local, rather than global proposal moves, because the change in energy induced by a local move is typically an intensive quantity. For molecular dynamics calculations, this approach was pursued in [73]. Using dynamically consistent local moves (a so-called J-splitting [116]), it was shown that in certain situations, a scalable Metropolis integrator can be designed; however, the extent to which this strategy remedies the issue of high rejection rate in high dimension is not clear at this point and should be tested in applications.

Acknowledgments

The author wishes to acknowledge Eric Vanden-Eijnden for useful comments on an earlier version of this paper. The research that led to this paper was funded by the US National Science Foundation through Division of Mathematical Sciences (DMS) grant # DMS-1212058.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Allen, M.P.; Tildesley, D.J. *Computer Simulation of Liquids*; Clarendon Press: Oxford, UK, 1987.
2. Frenkel, D.; Smit, B. *Understanding Molecular Simulation: From Algorithms to Applications*, 2nd ed.; Academic Press: Waltham, MA, USA, 2002.
3. Rapaport, D.C. *The Art of Molecular Dynamics Simulation*; Cambridge University Press: Cambridge, UK, 2004.
4. Tuckerman, M. *Statistical Mechanics and Molecular Simulations*; Oxford University Press: Oxford, UK, 2008.
5. Schlick, T. *Molecular Modeling and Simulation: An Interdisciplinary Guide*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 21.
6. Rahman, A. Correlations in the motion of atoms in liquid argon. *Phys. Rev.* **1964**, *136*, A405.
7. Verlet, L. Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.* **1967**, *159*, 98–103.
8. Alder, B.J.; Wainwright, T.E. Velocity autocorrelations for hard spheres. *Phys. Rev. Lett.* **1967**, *18*, 988–990.
9. Alder, B.J.; Gass, D.M.; Wainwright, T.E. Studies in molecular dynamics. VIII. The transport coefficients for a hard-sphere fluid. *J. Chem. Phys.* **1970**, *53*, 3813–3826.
10. Harp, G.D.; Berne, B.J. Time-correlation functions, memory functions, and molecular dynamics. *Phys. Rev. A* **1970**, *2*, 975–996.
11. Rahman, A.; Stillinger, F.H. Molecular dynamics study of liquid water. *J. Chem. Phys.* **1971**, *55*, 3336–3359.

12. Stillinger, F.H.; Rahman, A. Improved simulation of liquid water by molecular dynamics. *J. Chem. Phys.* **1974**, *60*, 1545–1557.
13. Stillinger, F.H. Water revisited. *Science* **1980**, *209*, 451–457.
14. McCammon, A.J.; Gelin, B.R.; Karplus, M. Dynamics of folded proteins. *Nature* **1977**, *267*, 585–590.
15. Van Gunsteren, W.F.; Berendsen, H.J.C. Algorithms for macromolecular dynamics and constraint dynamics. *Mol. Phys.* **1977**, *34*, 1311–1327.
16. McCammon, J.A.; Karplus, M. Simulation of protein dynamics. *Annu. Rev. Phys. Chem.* **1980**, *31*, 29–45.
17. Van Gunsteren, W.F.; Karplus, M. Protein dynamics in solution and in a crystalline environment: A molecular dynamics study. *Biochemistry* **1982**, *21*, 2259–2274.
18. Karplus, M.; McCammon, A.J. Dynamics of proteins: Elements and function. *Annu. Rev. Biochem.* **1983**, *52*, 263–300.
19. Van Gunsteren, W.F.; Berendsen, H.J.C. Computer simulation of molecular dynamics: Methodology, applications, and perspectives in chemistry. *Angew. Chem. Int. Ed. Engl.* **1990**, *29*, 992–1023.
20. Karplus, M.; McCammon, A.J. Molecular dynamics simulations of biomolecules. *Nat. Struct. Mol. Biol.* **2002**, *9*, 646–652.
21. Case, D.A. Molecular dynamics and NMR spin relaxation in proteins. *Acc. Chem. Res.* **2002**, *35*, 325–331.
22. Adcock, A.S.; McCammon, A.J. Molecular dynamics: Survey of methods for simulating the activity of proteins. *Chem. Rev.* **2006**, *106*, 1589–1615.
23. Van Gunsteren, W.F.; Dolenc, J.; Mark, A.E. Molecular simulation as an aid to experimentalists. *Curr. Opin. Struct. Biol.* **2008**, *18*, 149–153.
24. Kapral, R.; Ciccotti, G. Molecular dynamics: An Account of its Evolution. In *Theory and Applications of Computational Chemistry: The First Forty Years*; Elsevier: New York, NY, USA, 2005; pp. 425–441.
25. Levitt, M. Molecular dynamics of native protein: I. Computer simulation of trajectories. *J. Mol. Biol.* **1983**, *168*, 595–617.
26. Levitt, M.; Sharon, R. Accurate simulation of protein dynamics in solution. *Proc. Natl. Acad. Sci. USA* **1988**, *85*, 7557–7561.
27. Daggett, V.; Levitt, M. A model of the molten globule state from molecular dynamics simulations. *Proc. Natl. Acad. Sci. USA* **1992**, *89*, 5142–5146.
28. Li, A.; Daggett, V. Investigation of the solution structure of chymotrypsin inhibitor 2 using molecular dynamics: Comparison to X-Ray crystallographic and NMR data. *Protein Eng.* **1995**, *8*, 1117–1128.
29. Duan, Y.; Kollman, P.A. Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution. *Science* **1998**, *282*, 740–744.
30. Freddolino, P.L.; Liu, F.; Gruebele, M.; Schulten, K. Ten-microsecond molecular dynamics simulation of a fast-folding WW domain. *Biophys. J.* **2008**, *94*, 75–77.

31. Shaw, D.E.; Maragakis, P.; Lindorff-Larsen, K.; Piana, S.; Dror, R.O.; Eastwood, M.P.; Bank, J.A.; Jumper, J.M.; Salmon, J.K.; Shan, Y. Atomic-level characterization of the structural dynamics of proteins. *Science* **2010**, *330*, 341–346.
32. Quentrec, B.; Brot, C. New method for searching for neighbors in molecular dynamics computations. *J. Comput. Phys.* **1973**, *13*, 430–432.
33. Ryckaert, J.P.; Ciccotti, G.; Berendsen, H.J.C. Numerical integration of the Cartesian equations of motion of a system with constraints: Molecular dynamics of n-Alkanes. *J. Comput. Phys.* **1977**, *23*, 327–341.
34. Schneider, T.; Stoll, E. Molecular dynamics study of a three-dimensional one-component model for distortive phase transitions. *Phys. Rev. B* **1978**, *17*, 1302–1322.
35. Brünger, A.; Brooks, C.L.; Karplus, M. Stochastic boundary conditions for molecular dynamics simulations of ST2 water. *Chem. Phys. Lett.* **1984**, *105*, 495–500.
36. Brooks, B.R.; Bruccoleri, R.E.; Olafson, B.D.; States, D.J.; Swaminathan, S.; Karplus, M. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.* **1983**, *4*, 187–217.
37. Nosé, S. A unified formulation for constant temperature molecular dynamics methods. *J. Chem. Phys.* **1984**, *81*, 511–519.
38. Hoover, W.G. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A* **1985**, *31*, 1695–1697.
39. Greengard, L.; Rokhlin, V. A fast algorithm for particle simulations. *J. Comput. Phys.* **1987**, *73*, 325–348.
40. Tuckerman, M.E.; Berne, B.J.; Martyna, G. Reversible multiple time scale molecular dynamics. *J. Chem. Phys.* **1992**, *97*, 1990–2001.
41. Skeel, R.D.; Izaguirre, J. An impulse integrator for Langevin dynamics. *Mol. Phys.* **2002**, *100*, 3885–3891.
42. Ricci, A.; Ciccotti, G. Algorithms for Brownian dynamics. *Mol. Phys.* **2003**, *101*, 1927–1931.
43. Bussi, G.; Parrinello, M. Accurate sampling using Langevin dynamics. *Phys. Rev. E* **2007**, *75*, 056707.
44. Milstein, G.N.; Tretyakov, M.V. Quasi-symplectic methods for Langevin-type equations. *IMA J. Num. Anal.* **2003**, *23*, 593–626.
45. Milstein, G.N.; Tretyakov, M.V. Computing ergodic limits for Langevin equations. *Physica D* **2007**, *229*, 81–95.
46. Yao, Z.; Wang, J.S.; Liu, G.R.; Cheng, M. Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. *Comput. Phys. Commun.* **2004**, *161*, 27–35.
47. Bussi, G.; Donadio, D.; Parrinello, M. Canonical sampling through velocity rescaling. *J. Chem. Phys.* **2007**, *126*, 014101.
48. Samoletov, A.A.; Chaplain, M.A.; Dettmann, C.P. Thermostats for "Slow" configurational modes. *J. Stat. Phys.* **2008**, *128*, 1321–1336.
49. Leimkuhler, B.; Noorizadeh, E.; Theil, F. A gentle stochastic thermostat for molecular dynamics. *J. Stat. Phys.* **2009**, *135*, 261–277.

50. Leimkuhler, B.; Reich, S. A Metropolis adjusted Nosé-Hoover thermostat. *Math. Model. Num. Anal.* **2009**, *43*, 743–755.
51. Scheraga, H.A.; Khalili, M.; Liwo, A. Protein-folding dynamics: Overview of molecular simulation techniques. *Annu. Rev. Phys. Chem.* **2007**, *58*, 57–83.
52. Dror, R.O.; Dirks, R.M.; Grossman, J.P.; Xu, H.; Shaw, D.E. Biomolecular simulation: A computational microscope for molecular biology. *Annu. Rev. Biophys.* **2012**, *41*, 429–452.
53. Lane, T.J.; Shukla, D.; Beauchamp, K.A.; Pande, V.S. To milliseconds and beyond: Challenges in the simulation of protein folding. *Curr. Opin. Struct. Biol.* **2013**, *23*, 58–65.
54. Nielsen, S.O.; Lopez, C.F.; Srinivas, G.; Klein, M.L. Coarse grain models and the computer simulation of soft materials. *J. Phys. Condens. Matter* **2004**, *16*, R481.
55. Tozzini, V. Coarse-grained models for proteins. *Curr. Opin. Struct. Biol.* **2005**, *15*, 144–150.
56. Clementi, C. Coarse-grained models of protein folding: Toy models or predictive tools? *Curr. Opin. Struct. Biol.* **2008**, *18*, 10–15.
57. Sherwood, P.; Brooks, B.R.; Sansom, M.S.P. Multiscale methods for macromolecular simulations. *Curr. Opin. Struct. Biol.* **2008**, *18*, 630–640.
58. Weinan, E. *Principles of Multiscale Modeling*; Cambridge University Press: Cambridge, UK, 2011.
59. Weinan, E.; Vanden-Eijnden, E. Metastability, Conformation Dynamics, and Transition Pathways in Complex Systems. In *Multiscale Modelling and Simulation*; Attinger, S., Koumoutsakos, P., Eds.; Lecture Notes in Computational Science and Engineering; Springer: Berlin/Heidelberg, Germany, 2004; pp. 35–68.
60. Vanden-Eijnden, E.; Venturoli, M. Markovian milestoning with Voronoi tessellations. *J. Chem. Phys.* **2009**, *130*, 194101.
61. Vanden-Eijnden, E.; Venturoli, M. Exact rate calculations by trajectory parallelization and twisting. *J. Chem. Phys.* **2009**, *131*, 044120.
62. Weinan, E.; Vanden-Eijnden, E. Transition-path theory and path-finding algorithms for the study of rare events. *Annu. Rev. Phys. Chem.* **2010**, *61*, 391–420.
63. Nelson, M.T.; Humphrey, W.; Gursoy, A.; Dalke, A.; Kalé, L.V.; Skeel, R.D.; Schulten, K. NAMD: A parallel, object-oriented molecular dynamics program. *Int. J. High Perform. Comput. Appl.* **1996**, *10*, 251–268.
64. Scott, W.R.P.; Hünenberger, P.H.; Tironi, I.G.; Mark, A.E.; Billeter, S.R.; Fennen, J.; Torda, A.E.; Huber, T.; Krüger, P.; van Gunsteren, W.F. The GROMOS biomolecular simulation program package. *J. Phys. Chem. A* **1999**, *103*, 3596–3607.
65. Talay, D. Stochastic Hamiltonian systems: Exponential convergence to the invariant measure, and discretization by the implicit Euler scheme. *Markov Process. Relat. Fields* **2002**, *8*, 1–36.
66. Higham, D.J.; Mao, X.; Stuart, A.M. Strong convergence of Euler-type methods for nonlinear stochastic differential equations. *IMA J. Num. Anal.* **2002**, *40*, 1041–1063.
67. Milstein, G.N.; Tretyakov, M.V. Numerical integration of stochastic differential equations with nonglobally Lipschitz coefficients. *IMA J. Num. Anal.* **2005**, *43*, 1139–1154.
68. Higham, D.J. Stochastic ordinary differential equations in applied and computational mathematics. *IMA J. Appl. Math.* **2011**, *76*, 449–474.

69. Hutzenthaler, M.; Jentzen, A.; Kloeden, P.E. Strong convergence of an explicit numerical method for SDEs with non-globally Lipschitz continuous coefficients. *Ann. Appl. Probab.* **2012**, *22*, 1611–1641.
70. Scemama, A.; Lelièvre, T.; Stoltz, G.; Cancés, E.; Caffarel, M. An efficient sampling algorithm for variational Monte Carlo. *J. Chem. Phys.* **2006**, *125*, 114105.
71. Akhmatskaya, E.; Bou-Rabee, N.; Reich, S. A comparison of generalized hybrid Monte Carlo methods with and without momentum flip. *J. Comput. Phys.* **2009**, *228*, 2256–2265.
72. Bou-Rabee, N.; Vanden-Eijnden, E. Pathwise accuracy and ergodicity of Metropolized integrators for SDEs. *Commun. Pure Appl. Math.* **2010**, *63*, 655–696.
73. Bou-Rabee, N.; Vanden-Eijnden, E. A patch that imparts unconditional stability to explicit integrators for Langevin-like equations. *J. Comput. Phys.* **2012**, *231*, 2565–2580.
74. Metropolis, N.; Rosenbluth, A.W.; Rosenbluth, M.N.; Teller, A.H.; Teller, E. Equations of state calculations by fast computing machines. *J. Chem. Phys.* **1953**, *21*, 1087–1092.
75. Hastings, W.K. Monte-Carlo methods using Markov chains and their applications. *Biometrika* **1970**, *57*, 97–109.
76. Rossky, P.J.; Doll, J.D.; Friedman, H.L. Brownian dynamics as smart Monte Carlo simulation. *J. Chem. Phys.* **1978**, *69*, 4628.
77. Duane, S.; Kennedy, A.D.; Pendleton, B.J.; Roweth, D. Hybrid Monte-Carlo. *Phys. Lett. B* **1987**, *195*, 216–222.
78. Horowitz, A.M. A generalized guided Monte-Carlo algorithm. *Phys. Lett. B* **1991**, *268*, 247–252.
79. Kennedy, A.D.; Pendleton, B. Cost of the generalized hybrid Monte Carlo algorithm for free field theory. *Nucl. Phys. B* **2001**, *607*, 456–510.
80. Liu, J.S. *Monte Carlo Strategies in Scientific Computing*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2008.
81. Akhmatskaya, E.; Reich, S. GSHMC: An efficient method for molecular simulation. *J. Comput. Phys.* **2008**, *227*, 4937–4954.
82. Lelièvre, T.; Rousset, M.; Stoltz, G. *Free Energy Computations: A Mathematical Perspective*, 1st ed.; Imperial College Press: London, UK, 2010.
83. *MATLAB*, Version 8.0.0 (R2012b); The MathWorks Inc.: Natick, MA, USA, 2012.
84. Introducing MEX-Files. Available online: http://www.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html (accessed on 19 September 2013).
85. Kloeden, P.E.; Platen, E. *Numerical Solution of Stochastic Differential Equations*; Springer: Berlin, Germany, 1992.
86. Hutzenthaler, M.; Jentzen, A.; Kloeden, P.E. Strong and weak divergence in finite time of Euler's method for stochastic differential equations with non-globally Lipschitz continuous coefficients. *Proc. R. Soc. A: Math. Phys. Eng. Sci.* **2011**, *467*, 1563–1576.
87. Vanden-Eijnden, E.; Ciccotti, G. Second-order integrators for Langevin equations with holonomic constraints. *Chem. Phys. Lett.* **2006**, *429*, 310–316.
88. Leimkuhler, B.; Matthews, C. Robust and efficient configurational molecular sampling via Langevin dynamics. *J. Chem. Phys.* **2013**, *138*, 174102.

89. Evans, L. *An Introduction to Stochastic Differential Equations*; American Mathematical Society: Providence, RI, USA, 2013.
90. Bou-Rabee, N.; Owhadi, H. Long-run accuracy of variational integrators in the stochastic context. *SIAM J. Numer. Anal.* **2010**, *48*, 278–297.
91. Leimkuhler, B.; Matthews, C.; Stoltz, G. The computation of averages from equilibrium and nonequilibrium Langevin molecular dynamics. **2013**, arXiv:1308.5814.
92. Bou-Rabee, N.; Donev, A.; Vanden-Eijnden, E. Metropolized integration schemes for self-adjoint diffusions. **2013**, arXiv:1309.5037.
93. Milstein, G.N.; Tretyakov, M.V. *Stochastic Numerics for Mathematical Physics*; Springer: Berlin, Germany, 2004.
94. Marsden, J.E; Ratiu, T.S. *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*; Springer: Berlin/Heidelberg, Germany, 1999.
95. Van Gunsteren, W.F.; Mark, A.E. Validation of molecular dynamics simulation. *J. Chem. Phys.* **1998**, *108*, 6109–6116.
96. Leimkuhler, B.; Reich, S. *Simulating Hamiltonian Dynamics*; Cambridge Monographs on Applied and Computational Mathematics; Cambridge University Press: Cambridge, UK, 2004.
97. Hairer, E.; Lubich, C.; Wanner, G. *Geometric Numerical Integration*; Springer: Berlin/Heidelberg, Germany, 2010.
98. Ikeda, N.; Watanabe, S. *Stochastic Differential Equations and Diffusion Processes*; North-Holland: Amsterdam, The Netherlands, 1989.
99. Klebaner, F.C. *Introduction to Stochastic Calculus with Applications*; Imperial College Press: London, UK, 2005.
100. Mengersen, K.L.; Tweedie, R.L. Rates of convergence of the Hastings and Metropolis algorithms. *Ann. Stat.* **1996**, *24*, 101–121.
101. Prato, G.D.; Zabczyk, J. *Ergodicity for Infinite Dimensional Systems*; Cambridge University Press: Cambridge, UK, 1996.
102. Haussman, U.G.; Pardoux, E. Time reversal for diffusions. *Ann. Probab.* **1986**, *14*, 1188–1205.
103. Kent, J. Time-reversible diffusions. *Adv. Appl. Prob.* **1978**, *10*, 819–835.
104. Ezra, G.S. Reversible measure-preserving integrators for non-Hamiltonian systems. *J. Chem. Phys.* **2006**, *125*, 034104.
105. Shardlow, T. Splitting for dissipative particle dynamics. *SIAM J. Sci. Comput.* **2003**, *24*, 1267–1282.
106. Serrano, M.; de Fabritiis, G.; Espanol, P.; Coveney, P.V. A stochastic Trotter integration scheme for dissipative particle dynamics. *Math. Comput. Simulat.* **2006**, *72*, 190–194.
107. Pavliotis, G.A.; Stuart, A.M.; Zygalakis, K.C. Calculating effective diffusivities in the limit of vanishing molecular diffusion. *J. Comput. Phys.* **2008**, *228*, 1030–1055.
108. Van Gunsteren, W.F.; Berendsen, H.J.C. Algorithms for Brownian dynamics. *Mol. Phys.* **1982**, *45*, 637–647.
109. Nummelin, E. *General Irreducible Markov Chains and Non-Negative Operators*; Cambridge University Press: New York, NY, USA, 1984.
110. Tierney, L. Markov chains for exploring posterior distributions. *Ann. Stat.* **1994**, *22*, 1701–1728.

111. Gelman, A.; Gilks, W.R.; Roberts, G.O. Weak convergence and optimal scaling of random walk Metropolis algorithms. *Ann. Appl. Probab.* **1997**, *7*, 110–120.
112. Roberts, G.O.; Rosenthal, J.S. Optimal scaling of discrete approximations to Langevin diffusions. *J. R. Stat. Soc. Ser. B* **1998**, *60*, 255–268.
113. Beskos, A.; Roberts, G.O.; Stuart, A.M. Optimal scalings for local Metropolis-Hastings chains on non-product targets in high dimensions. *Ann. Appl. Probab.* **2009**, *19*, 863–898.
114. Beskos, A.; Pillai, N.S.; Roberts, G.O.; Sanz-Serna, J.M.; Stuart, A.M. Optimal tuning of hybrid Monte-Carlo algorithm. **2010**, arXiv:1001.4460.
115. Mattingly, J.C.; Pillai, N.S.; Stuart, A.M. Diffusion limits of the random walk Metropolis algorithm in high dimensions. *Ann. Appl. Probab.* **2012**, *22*, 881–930.
116. Kang, F.; Dao-Liu, W. Dynamical Systems and Geometric Construction of Algorithms. In *Computational Mathematics in China*; Contemporary Mathmatics, Volume 163; Shi, Z.-C., Yang, C.C., Eds.; American Mathmatical Society: New York, NY, USA,1994; pp. 1–32.