In Case Study 4, this method was used to compute the standard deviation of the energy in a Molecular Dynamics simulation. In Figure 4.4 a typical plot of this estimate of the variance as a function of block size is shown. For small values of M, the number of blocking operations, the data are correlated and as a consequence the variance will increase if we perform the blocking operation. For very high values of M we have only a few samples, and as a result, the statistical error in our estimate of $\sigma^2(A)$ will be large. The plateau in between gives us the value of $\sigma^2(A)$ we are interested in.

Appendix E

Integration Schemes

E.1 Higher-Order Schemes

The basic idea behind the predictor-corrector algorithms is to use information about the position and its first n derivatives at time t to arrive at a prediction for the position and its first n derivatives at time $t + \Delta t$. We then compute the forces (and thereby the accelerations) at the predicted positions. And then we find that these accelerations are *not* equal to the values that we had predicted. So we adjust our predictions for the accelerations to match the facts. But we do more than that. On the basis of the observed discrepancy between the predicted and observed accelerations, we also try to improve our estimate of the positions and the remaining $n-1$ derivatives. This is the "corrector" part of the predictor-corrector algorithm. The precise "recipe" used in applying this correction is a compromise between accuracy and stability. Here, we shall simply show a specific example of a predictor-corrector algorithm, without attempting to justify the form of the corrector part.

Consider the Taylor expansion of the coordinate of a given particle at time $t + \Delta t$:

$$r(t+\Delta t) = r(t) + \Delta t \frac{\partial r}{\partial t} + \frac{\Delta t^2}{2!}\frac{\partial^2 r}{\partial t^2} + \frac{\Delta t^3}{3!}\frac{\partial^3 r}{\partial t^3} + \cdots .$$

Using the notation

$$\begin{aligned} x_0(t) &\equiv r(t) \\ x_1(t) &\equiv \Delta t \frac{\partial r}{\partial t} \\ x_2(t) &\equiv \frac{\Delta t^2}{2!}\frac{\partial^2 r}{\partial t^2} \\ x_3(t) &\equiv \frac{\Delta t^3}{3!}\frac{\partial^3 r}{\partial t^3}, \end{aligned}$$

we can write the following *predictions* for $x_0(t+\Delta t)$ through $x_3(t+\Delta t)$:

$$\begin{aligned} x_0(t+\Delta t) &= x_0(t) + x_1(t) + x_2(t) + x_3(t) \\ x_1(t+\Delta t) &= x_1(t) + 2x_2(t) + 3x_3(t) \\ x_2(t+\Delta t) &= x_2(t) + 3x_3(t) \\ x_3(t+\Delta t) &= x_3(t). \end{aligned}$$

Now that we have $x_0(t+\Delta t)$, we can compute the forces at the predicted position, and thus compute the corrected value for $x_2(t+\Delta t)$. We denote the difference between $x_2^{\text{corrected}}$ and $x_2^{\text{predicted}}$ by Δx_2:

$$\Delta x_2 \equiv x_2^{\text{corrected}} - x_2^{\text{predicted}}.$$

We now estimate "corrected" values for x_0 through x_3, as follows:

$$x_n^{\text{corrected}} = x_n^{\text{predicted}} + C_n \Delta x_2, \qquad \text{(E.1.1)}$$

where the C_n are constants fixed for a given order algorithm. As indicated, the values for C_n are such that they yield an optimal compromise between the accuracy and the stability of the algorithm. For instance, for a fifth-order predictor-corrector algorithm (i.e., one that uses x_0 through x_4), the values for C_n are

$$\begin{aligned} C_0 &= \frac{19}{120} \\ C_1 &= \frac{3}{4} \\ C_2 &= 1 \qquad \text{(of course)} \\ C_3 &= \frac{1}{2} \\ C_4 &= \frac{1}{12}. \end{aligned}$$

One may iterate the predictor and corrector steps to self-consistency. However, there is little point in doing so because (1) every iteration requires a force calculation. One would be better off spending the same computer time to run with a *shorter* time step and only one iteration because (2) even if we iterate the predictor-corrector algorithm to convergence, we still do not get the *exact* trajectory: the error is still of order Δt^n for an nth-order algorithm. This is why we gain more accuracy by going to a shorter time step than by iterating to convergence at a fixed value of Δt.

E.2 Nosé-Hoover Algorithms

As discussed in section 6.1.2, it is advantageous to implement the Nosé thermostat using the formulation of Hoover, equations (6.1.24)–(6.1.27). Since the velocity also appears on the right-hand side of equation (6.1.25), this scheme cannot be implemented directly into the velocity Verlet algorithm (see also section 4.3). To see this, consider a standard constant-N,V,E simulation, for which the velocity Verlet algorithm is of the form

$$\begin{aligned} r(t+\Delta t) &= r(t) + v(t)\Delta t + f(t)\Delta t^2/(2m) \\ v(t+\Delta t) &= v(t) + \frac{f(t+\Delta t) + f(t)}{2m}\Delta t. \end{aligned}$$

When we use this scheme for the Nosé-Hoover equations of motion, we obtain for the positions and velocities

$$\begin{aligned} r_i(t+\Delta t) &= r_i(t) + v(t)\Delta t + [f_i(t)/m_i - \xi(t)v_i(t)]\,\Delta t^2/2 \qquad \text{(E.2.1)} \\ v_i(t+\Delta t) &= v_i(t) + [f_i(t+\Delta t)/m_i - \xi(t+\Delta t)v_i(t+\Delta t) \\ &\qquad + f_i(t)/m_i - \xi(t)v_i(t)]\,\Delta t/2. \qquad \text{(E.2.2)} \end{aligned}$$

The first step of the velocity Verlet algorithm can be carried out without difficulty. In the second step, we first update the velocity, using the old "forces" to the intermediate value $v(t+\Delta t/2) \equiv v'$. And then we must use the new "forces" to update v':

$$v_i(t+\Delta t) = v'_i + [f_i(t+\Delta t)/m_i - \xi(t+\Delta t)v_i(t+\Delta t)]\,\Delta t/2. \qquad \text{(E.2.3)}$$

In these equations $v_i(t+\Delta t)$ appears on the right- and left-hand sides; therefore, these equations cannot be integrated exactly.[1] For this reason the Nosé-Hoover method is usually implemented using a predictor-corrector scheme or solved iteratively [138]. This has a disadvantage that the solution is no longer time reversible. Martyna *et al.* [85] have developed a set of explicit reversible integrators using the Liouville approach (see section 4.3.3) for this type of extended systems.

[1] For the harmonic oscillator it is possible to find an analytic solution (see Case Study 12).

E.2.1 Canonical Ensemble

For M chains, the Nosé-Hoover equations of motion are given by (see also section 6.1.3)

$$\begin{aligned}
\dot{\mathbf{r}}_i &= \mathbf{p}_i/m_i \\
\dot{\mathbf{p}}_i &= \mathbf{F}_i - \frac{p_{\xi_1}}{Q_1}\mathbf{p}_i \\
\dot{\xi}_k &= \frac{p_{\xi_k}}{Q_k} \qquad\qquad k = 1,\ldots,M \\
\dot{p}_{\xi_1} &= \left(\sum_i p_i^2/m_i - Lk_BT\right) - \frac{p_{\xi_2}}{Q_2}p_{\xi_1} \\
\dot{p}_{\xi_k} &= \left[\frac{p_{\xi_{k-1}}^2}{Q_{k-1}} - k_BT\right] - \frac{p_{\xi_{k+1}}}{Q_{k+1}}p_{\xi_k} \\
\dot{p}_{\xi_M} &= \left[\frac{p_{\xi_{M-1}}^2}{Q_{M-1}} - k_BT\right].
\end{aligned}$$

The Liouville operator for the equation of motions is defined as (see section 4.3.3)

$$iL \equiv \dot{\eta}\frac{\partial}{\partial\eta}$$

with $\eta = (\mathbf{r}^N, \mathbf{p}^N, \xi^M, p_\xi^M)$. Using the equations of motion, $\mathbf{p}_i = m_i\mathbf{v}_i$, and $p_{\xi_k} = Q_k v_{\xi_k}$, we obtain as Liouville operator for the Nosé-Hoover chains

$$\begin{aligned}
iL_{NHC} = &\sum_{i=1}^{N}\mathbf{v}_i\cdot\nabla_{\mathbf{r}_i} + \sum_{i=1}^{N}\left[\frac{\mathbf{F}_i(\mathbf{r}_i)}{m_i}\right]\cdot\nabla_{\mathbf{v}_i} - \sum_{i=1}^{N}v_{\xi_1}\mathbf{v}_i\cdot\nabla_{\mathbf{v}_i} + \sum_{k=1}^{M}v_{\xi_k}\frac{\partial}{\partial\xi_k} \\
&+ \sum_{k=1}^{M-1}(G_k - v_{\xi_k}v_{\xi_{k+1}})\frac{\partial}{\partial v_{\xi_k}} + G_M\frac{\partial}{\partial v_{\xi_M}}
\end{aligned}$$

with

$$\begin{aligned}
G_1 &= \frac{1}{Q_1}\left(\sum_{i=1}^{N}m_iv_i^2 - Lk_BT\right) \\
G_k &= \frac{1}{Q_k}\left(Q_{k-1}v_{\xi_{k-1}}^2 - k_BT\right).
\end{aligned}$$

As explained in section 4.3.3, the Liouville equation combined with the Trotter formula is a powerful technique for deriving a time-reversible algorithm for solving the equations of motion numerically. Here we will use this technique to derive such a scheme for the Nosé-Hoover thermostats. We use a simplified version; a more complete description can be found in ref. [85].

We have to make an intelligent separation of the Liouville operator. The first step is to separate the part of the Liouville operator that only involves the positions (iL_r) and the velocities (iL_v) from the parts that involve the Nosé-Hoover thermostats (iL_C):

$$iL_{NHC} = iL_r + iL_v + iL_C$$

with

$$\begin{aligned}
iL_r &= \sum_{i=1}^{N}\mathbf{v}_i\cdot\nabla_{\mathbf{r}_i} \\
iL_v &= \sum_{i=1}^{N}\frac{\mathbf{F}_i(\mathbf{r}_i)}{m_i}\cdot\nabla_{\mathbf{v}_i} \\
iL_C &= \sum_{k=1}^{M}v_{\xi_k}\frac{\partial}{\partial\xi_k} - \sum_{i=1}^{N}v_{\xi_1}\mathbf{v}_i\cdot\nabla_{\mathbf{v}_i} \\
&\quad + \sum_{k=1}^{M-1}(G_k - v_{\xi_k}v_{\xi_{k+1}})\frac{\partial}{\partial v_{\xi_k}} + G_M\frac{\partial}{\partial v_{\xi_M}}.
\end{aligned}$$

There are several ways to factorize iL_{NHC} using the Trotter formula; we follow the one used by Martyna *et al.* [85]:

$$e^{(iL\Delta t)} = e^{(iL_C\Delta t/2)}e^{(iL_v\Delta t/2)}e^{(iL_r\Delta t)}e^{(iL_v\Delta t/2)}e^{(iL_C\Delta t/2)} + \mathcal{O}\left(\Delta t^3\right). \quad \text{(E.2.4)}$$

The Nosé-Hoover chain part L_C has to be further factorized. Here, we will do this for a chain of length $M = 2$; the more general case is discussed in ref. [85]. The Nosé-Hoover part of the Liouville operator for this chain length can be separated into five terms:

$$iL_C = iL_\xi + iL_{C_v} + iL_{G1} + iL_{v\xi_1} + iL_{G2},$$

where the terms are defined as

$$\begin{aligned}
iL_\xi &\equiv \sum_{k=1}^{2}v_{\xi_k}\frac{\partial}{\partial\xi_k} \\
iL_{C_v} &\equiv -\sum_{i=1}^{N}v_{\xi_1}\mathbf{v}_i\cdot\nabla_{\mathbf{v}_i} \\
iL_{G1} &\equiv G_1\frac{\partial}{\partial v_{\xi_1}} \\
iL_{v\xi_1} &\equiv -(v_{\xi_1}v_{\xi_2})\frac{\partial}{\partial v_{\xi_1}} \\
iL_{G2} &\equiv G_2\frac{\partial}{\partial v_{\xi_2}}.
\end{aligned}$$

The factorization for the Trotter equation that we use is[2]

$$\begin{aligned} e^{(iL_C\Delta t/2)} &= e^{(iL_{G2}\Delta t/4)}e^{(iL_{v\xi_1}\Delta t/4+iL_{G1}\Delta t/4)} \\ &\quad \times e^{(iL_\xi\Delta t/2)}e^{(iL_{Cv}\Delta t/2)}e^{(iL_{G1}\Delta t/4+iL_{v\xi_1}\Delta t/4)}e^{(iL_{G2}\Delta t/4)} \\ &= e^{(iL_{G_2}\Delta t/4)}\left[e^{(iL_{v\xi_1}\Delta t/8)}e^{(iL_{G1}\Delta t/4)}e^{(iL_{v\xi_1}\Delta t/8)}\right] \\ &\quad \times e^{(iL_\xi\Delta t/2)}e^{(iL_{Cv}\Delta t/2)} \\ &\quad \times \left[e^{(iL_{v\xi_1}\Delta t/8)}e^{(iL_{G1}\Delta t/4)}e^{(iL_{v\xi_1}\Delta t/8)}\right]e^{(iL_{G2}\Delta t/4)}. \end{aligned} \tag{E.2.5}$$

Our numerical algorithm is now fully defined by equations (E.2.4) and (E.2.5). This seemingly complicated set of equations is actually relatively easy to implement in a simulation.

To see how the implementation works, we need to know how each operator works on our coordinates $\eta = (\mathbf{r}^N, \mathbf{v}^N, \xi_1, v_{\xi_1}, \xi_2, v_{\xi_2})$. If we start at $t = 0$ with initial condition η, the position at time $t = \Delta t$ follows from

$$e^{iL_{NHC}\Delta t} f[\mathbf{r}^N, \mathbf{v}^N, \xi_1, v_{\xi_1}, \xi_2, v_{\xi_2}].$$

Because of the Trotter expansion, we can apply each term in iL_{NHC}, sequentially. For example, if we let the first term of the Liouville operator, iL_{G2}, act on the initial state η,

$$\begin{aligned} &\exp\left(\frac{\Delta t}{4}G_2\frac{\partial}{\partial v_{\xi_2}}\right) f\left[\mathbf{r}^N, \mathbf{p}^N, \xi_1, v_{\xi_1}, \xi_2, v_{\xi_2}\right] \\ &= \sum_{n=0}^{\infty}\frac{(G_2\Delta t/4)^n}{n!}\frac{\partial^n}{\partial v_{\xi_2}^n} f\left[\mathbf{r}^N, \mathbf{p}^N, \xi_1, v_{\xi_1}, \xi_2, v_{\xi_2}\right] \\ &= f\left[\mathbf{r}^N, \mathbf{p}^N, \xi_1, v_{\xi_1}, \xi_2, v_{\xi_2} + G_2\Delta t/4\right]. \end{aligned}$$

This shows that the effect of iL_{G2} is to shift v_{ξ_2} without affecting the other coordinates. This gives as transformation rule for this operator:

$$e^{(iL_{G2}\Delta t/4)}: \quad v_{\xi_2} \to v_{\xi_2} + G_2\Delta t/4. \tag{E.2.6}$$

The operators ($iL_{v\xi_1}$ and iL_{Cv}) are of the form $\exp(ax\partial/\partial x)$; such operators give a scaling of the x coordinate:

$$\begin{aligned} \exp\left(ax\frac{\partial}{\partial x}\right) f(x) &= \exp\left(a\frac{\partial}{\partial \ln(x)}\right) f\{\exp[\ln(x)]\} \\ &= f\{\exp[\ln(x) + a]\} = f[x\exp(a)] \end{aligned}$$

[2]The second factorization, indicated by $[\cdots]$, is used to avoid a hyperbolic sine function, which has a possible singularity. See ref. [85] for details.

If we apply this result[3] to $iL_{v\xi_1}$, we obtain for this operator

$$\begin{aligned} &\exp\left(-\frac{\Delta t}{8}v_{\xi_2}v_{\xi_1}\frac{\partial}{\partial v_{\xi_1}}\right) f\left[\mathbf{r}^N, \mathbf{p}^N, \xi_1, v_{\xi_1}, \xi_2, v_{\xi_2}\right] \\ &= f\left[\mathbf{r}^N, \mathbf{p}^N, \xi_1, \exp\left(-\frac{\Delta t}{8}v_{\xi_2}\right)v_{\xi_1}, \xi_2, v_{\xi_2}\right], \end{aligned}$$

giving the transformation rule

$$e^{(iL_{v\xi_1}\Delta t/8)}: \quad v_{\xi_1} \to \exp\left[-v_{\xi_2}\Delta t/8\right]v_{\xi_1}. \tag{E.2.7}$$

In a similar way we can derive for the other terms

$$e^{(iL_{G1}\Delta t/4)}: \quad v_{\xi_1} \to v_{\xi_1} + G_1\Delta t/4 \tag{E.2.8}$$

$$e^{(iL_\xi\Delta t/2)}: \quad \xi_1 \to \xi_1 - v_{\xi_1}\Delta t/2 \tag{E.2.9}$$

$$\xi_2 \to \xi_2 - v_{\xi_2}\Delta t/2 \tag{E.2.10}$$

$$e^{(iL_{Cv}\Delta t/2)}: \quad v_i \to \exp\left[-v_{\xi_1}\Delta t/2\right]v_i. \tag{E.2.11}$$

Finally, the transformation rules that are associated to iL_v and iL_r are similar to the velocity Verlet algorithm, i.e.,

$$e^{(iL_v\Delta t/2)}: \quad \mathbf{v}_i \to \mathbf{v}_i + \mathbf{F}_i\Delta t/(2m) \tag{E.2.12}$$

$$e^{(iL_r\Delta t)}: \quad \mathbf{r}_i \to \mathbf{r}_i + \mathbf{v}_i\Delta t. \tag{E.2.13}$$

With these transformation rules (E.2.6)–(E.2.13) we can write down our numerical algorithm by subsequently applying the transformation rules according to the order defined by equations (E.2.4) and (E.2.5). If we start with initial coordinate $\eta(0) = (\mathbf{r}^N, \mathbf{v}^N, \xi_1, v_{\xi_1}, \xi_2, v_{\xi_2})$, we have to apply first $e^{iL_C}\eta$. Since this operator is further factorized according to equation (E.2.5) the first step in our algorithm is to apply $e^{(iL_{G2}\Delta t/4)}$. According to transformation rule (E.2.6) applying this operator on η gives as new state

$$v_{\xi_2}(\Delta t/4) = v_{\xi_2} + G_2\Delta t/4.$$

The output of this rule is the new state on which we apply the next operator in equation (E.2.5), $iL_{v\xi_1}$, with transformation rule (E.2.9):

$$v_{\xi_1}(\Delta t/8) = \exp\left[-v_{\xi_2}(\Delta t/4)\Delta t/8\right]v_{\xi_1}.$$

[3]This can be generalized, giving the identity

$$\begin{aligned} \exp\left(a\frac{\partial}{\partial g(x)}\right) f(x) &= \exp\left(a\frac{\partial}{\partial g(x)}\right) f\left\{g^{-1}[g(x)])\right\} \\ &= \exp\left(a\frac{\partial}{\partial y}\right) f\left[g^{-1}(y)\right] \\ &= f\left\{g^{-1}[y + a]\right\} = f\left\{g^{-1}[g(x) + a]\right\}. \end{aligned}$$

Algorithm 30 (Equations of Motion: Nosé-Hoover)

```
subroutine integrate     integrate equations of motion
                         Nosé-Hoover thermostat
call chain(uk)
call pos_vel(uk)
call chain(uk)
return
end
```

Comments to this algorithm:

1. *This subroutine solves the equations of motion for a single time step* Δt *using the Trotter equations (E.2.4) and (E.2.5).*
2. *In the subroutine* `chain` *we apply* $e^{iL_C \Delta t/4}$ *to the current state (see Algorithm 31).*
3. *In the subroutine* `pos_vel` *we apply* $e^{(iL_r+iL_p)\Delta t}$ *to the current state (see Algorithm 32).*
4. `uk` *is the total kinetic energy.*

The next step is to apply iL_{G1}, followed by again $iL_{v_{\xi 1}}$, etc. In this way we continue to apply all operators on the output of the previous step.

Applying the Nosé-Hoover part of the Liouville operator changes ξ_k, v_{ξ_k}, and $\mathbf{v}_i$. The other two Liouville operators change $\mathbf{v}_i$ and $\mathbf{r}_i$. This makes it convenient to separate the algorithm into two parts in which the positions and velocities of the particles and the Nosé-Hoover chains are considered separately. An example of a possible implementation is shown in Algorithm 30.

E.2.2 The Isothermal-Isobaric Ensemble

Similar to the canonical ensemble we can derive a time-reversible integration scheme for simulation in the NPT ensemble. The equations of motions are given by expressions (6.2.1)–(6.2.8):

$$\begin{aligned}
\dot{\mathbf{r}}_i &= \frac{\mathbf{p}_i}{m_i} + \frac{p_\epsilon}{W}\mathbf{r}_i \\
\dot{\mathbf{p}}_i &= \mathbf{F}_i - \left(1 + \frac{d}{dN}\right)\frac{p_\epsilon}{W}\mathbf{p}_i - \frac{p_{\xi_1}}{Q_1}\mathbf{p}_i \\
\dot{V} &= dV p_\epsilon / W
\end{aligned}$$

Algorithm 31 (Propagating the chain)

```
subroutine chain(uk)              apply equation (E.2.5)
                                  to the current position

G2=(Q1*vxi1*vxi1-T)
vxi2=vxi2+G2*delt4                Update v_ξ2 using equation (E.2.6)
vxi1=vxi1*exp(-vxi2*delt8)        Update v_ξ1 using equation (E.2.7)
G1=(2*uk-L*T)/Q1
vxi1=vxi1+G1*delt4                Update v_ξ1 using equation (E.2.8)
vxi1=vxi1*exp(-vxi2*delt8)        Update v_ξ1 using equation (E.2.7)
xi1=xi1+vxi1*delt2                Update ξ1 using equation (E.2.9)
xi2=xi2+vxi2*delt2                Update ξ2 using equation (E.2.10)
s=exp(-vxi1*delt2)                Scale factor in equation (E.2.11)
do i=1,npart
  v(i)=s*v(i)                     update v_i using equation (E.2.11)
enddo
uk=uk*s*s                         update kinetic energy
vxi1=vxi1*exp(-vxi2*delt8)        Update v_ξ1 using equation (E.2.7)
G1=(2*uk-L*T)/Q1
vxi1=vxi1+G1*delt4                Update v_ξ1 using equation (E.2.8)
vxi1=vxi1*exp(-vxi2*delt8)        Update v_ξ1 using equation (E.2.7)
G2=(Q1*vxi1*vxi1-T)/Q2
vxi2=vxi2+G2*delt4                Update v_ξ2 using equation (E.2.6)
return
end
```

Comments to this algorithm:

1. *In this subroutine* `T` *is the imposed temperature,* `delt`$= \Delta t$, `delt2`$= \Delta t/2$, `delt4`$= \Delta t/4$, *and* `delt8`$= \Delta t/8$.
2. `uk` *is the total kinetic energy.*

Algorithm 32 (Propagating the Positions and Velocities)

```
subroutine pos_vel(uk)          apply equation (E.2.4)
                                to the current position
uk=0
do i=1,npart
  x(i)=x(i)+v(i)*delt2          update x_i using equation (E.2.13)
enddo
call force                      calculate the force
do i=1,npart
  v(i)=v(i)+f(i)*delt/m         update v_i using equation (E.2.12)
  x(i)=x(i)+v(i)*delt2          update x_i using equation (E.2.13)
  uk=uk+m*v(i)*v(i)/2           update kinetic energy
enddo
return
end
```

Comments to this algorithm:

1. *In this subroutine* `delt` $= \Delta t$ *and* `delt2` $= \Delta t/2$.
2. *The subroutine* `force` *calculates the force on the particles.*

$$
\begin{aligned}
\dot{p}_\epsilon &= dV(P_{int} - P_{ext}) + \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{m_i} - \frac{p_{\xi_1}}{Q_1}p_\epsilon \\
\dot{\xi}_k &= \frac{p_{\xi_k}}{Q_k} \qquad \text{for } k = 1,\ldots,M \\
\dot{p}_{\xi_1} &= \sum_{i=1}^{N}\frac{\mathbf{p}_i^2}{m_i} + \frac{p_\epsilon^2}{W} - (dN+1)k_BT - \frac{p_{\xi_2}}{Q_2}p_{\xi_1} \\
\dot{p}_{\xi_k} &= \frac{p_{\xi_{k-1}}^2}{Q_{k-1}} - k_BT - \frac{p_{\xi_{k+1}}}{Q_{k+1}}p_{\xi_k} \qquad \text{for } k = 2,\ldots,M-1 \\
\dot{p}_{\xi_M} &= \frac{p_{\xi_{M-1}}^2}{Q_{M-1}} - k_BT.
\end{aligned}
$$

To derive a time-reversible numerical integration scheme to solve the equations of motion we use again the Liouville approach.

A state is characterized by the variables $\eta = (\mathbf{r}^N, \mathbf{p}^N, \epsilon, p_\epsilon, \xi^M, p_\xi^M)$. The Liouville operator is defined by

$$
iL_{NPT} \equiv \dot{\eta}\frac{\partial}{\partial\eta}.
$$

For a chain of length $M = 2$, using $\mathbf{p}_i = m_i\mathbf{v}_i (\neq m_i\dot{\mathbf{r}}_i)$, $p_{\xi_k} = Q_k v_{\xi_k}$, $\epsilon = (\ln V)/d$, and $p_\eta = W v_\eta$, the Liouville operator for these equations of motion can be written as

$$
iL_{NPT} = iL_r + iL_v + iL_{CP},
$$

in which we define the operators

$$
\begin{aligned}
iL_r &= \sum_{i=1}^{N}(\mathbf{v}_i + v_\epsilon\mathbf{r}_i)\cdot\nabla_{\mathbf{r}_i} + v_\epsilon\frac{\partial}{\partial\epsilon} \\
iL_v &= \sum_{i=1}^{N}\left[\frac{\mathbf{F}_i(\mathbf{r})}{m_i}\right]\cdot\nabla_{\mathbf{v}_i} \\
iL_{CP} &= -\sum_{i=1}^{N}v_{\xi_1}\mathbf{v}_i\cdot\nabla_{\mathbf{v}_i} + \sum_{k=1}^{M}v_{\xi_k}\frac{\partial}{\partial\xi_k} + \sum_{k=1}^{M-1}(G_k - v_{\xi_k}v_{\xi_{k+1}})\frac{\partial}{\partial v_{\xi_k}} \\
&\quad + G_M\frac{\partial}{\partial v_{\xi_M}} - \left(1+\frac{1}{N}\right)\sum_{i=1}^{N}v_\epsilon\mathbf{v}_i\cdot\nabla_{\mathbf{v}_i} + (G_\epsilon - v_\epsilon v_{\xi_1})\frac{\partial}{\partial v_\epsilon}
\end{aligned}
$$

with

$$
\begin{aligned}
G_1 &= \frac{1}{Q}\left[\sum_{i=1}^{N}m_iv_i^2 + Wv_\epsilon^2 - (N_f+1)k_BT\right] \\
G_k &= \frac{1}{Q_k}\left(Q_{k-1}v_{\xi_{k-1}}^2 - k_BT\right) \\
G_\epsilon &= \frac{1}{W}\left[\left(1+\frac{1}{N}\right)\sum_{i=1}^{N}m_iv_i^2 + \sum_{i=1}^{N}\mathbf{r}_i\cdot\mathbf{F}_i(\mathbf{r})m_i - dV\frac{\partial U(\mathbf{r},V)}{\partial V} - dP_{ext}V\right]
\end{aligned}
$$

An appropriate Trotter equation for the equations of motion is [85]

$$
e^{(iL_{NPT}\Delta t)} = e^{(iL_{CP}\Delta t/2)}e^{(iL_v\Delta t/2)}e^{(iL_r\Delta t)}e^{(iL_v\Delta t/2)}e^{(iL_{CP}\Delta t/2)} + \mathcal{O}\left(\Delta t^3\right). \tag{E.2.14}
$$

The operator iL_{CP} has to be further factorized:

$$
iL_{CP} = iL_\xi + iL_{C_v} + iL_{G\epsilon} + iL_{v\epsilon} + iL_{G1} + iL_{v\xi_1} + iL_{G2},
$$

where the terms are defined as

$$
\begin{aligned}
iL_\xi &\equiv \sum_{k=1}^{2} v_{\xi_k} \frac{\partial}{\partial \xi_k} \\
iL_{C_v} &\equiv -\sum_{i=1}^{N} \left[v_{\xi_1} + \left(1 + \frac{d}{dN}\right) \right] \mathbf{v}_i \cdot \nabla_{\mathbf{v}_i} \\
iL_{G\epsilon} &\equiv G_\epsilon \frac{\partial}{\partial v_\epsilon} \\
iL_{v\epsilon} &\equiv -(v_{\xi_1} v_\epsilon) \frac{\partial}{\partial v_\epsilon} \\
iL_{G1} &\equiv G_1 \frac{\partial}{\partial v_{\xi_1}} \\
iL_{v\xi_1} &\equiv -(v_{\xi_1} v_{\xi_2}) \frac{\partial}{\partial v_{\xi_1}} \\
iL_{G2} &\equiv G_2 \frac{\partial}{\partial v_{\xi_2}}.
\end{aligned}
$$

The Trotter expansion of the term iL_C is

$$
\begin{aligned}
e^{(iL_{CP}\Delta t/2)} &= e^{(iL_{G2}\Delta t/4 + iL_{v\xi_1}\Delta t/4)} e^{(iL_{G1}\Delta t/4)} e^{(iL_{G\epsilon}\Delta t/4 + iL_{v\epsilon}\Delta t/4)} \\
&\quad \times e^{(iL_\xi \Delta t/2)} e^{(iL_{Cv}\Delta t/2)} \\
&\quad \times e^{(iL_{v\epsilon}\Delta t/4 + iL_{G\epsilon}\Delta t/4)} e^{(iL_{G1}\Delta t/4 + iL_{v\xi_1}\Delta t/4)} e^{(iL_{G2}\Delta t/4)} \\
&= e^{(iL_{G_2}\Delta t/4)} \left[e^{(iL_{v\xi_1}\Delta t/8)} e^{(iL_{G1}\Delta t/4)} e^{(iL_{v\xi_1}\Delta t/8)} \right] \\
&\quad \times \left[e^{(iL_{v\epsilon}\Delta t/8)} e^{(iL_{G\epsilon}\Delta t/4)} e^{(iL_{v\epsilon}\Delta t/8)} \right] \\
&\quad \times e^{(iL_\xi \Delta t/2)} e^{(iL_{Cv}\Delta t/2)} \left[e^{(iL_{v\epsilon}\Delta t/8)} e^{(iL_{G\epsilon}\Delta t/4)} e^{(iL_{v\epsilon}\Delta t/8)} \right] \\
&\quad \times \left[e^{(iL_{v\xi_1}\Delta t/8)} e^{(iL_{G1}\Delta t/4)} e^{(iL_{v\xi_1}\Delta t/8)} \right] e^{(iL_{G2}\Delta t/4)}. \quad \text{(E.2.15)}
\end{aligned}
$$

Similar to the NVT version the transformation rules of the various operators can be derived and translated into an algorithm. Such an algorithm is presented in ref. [85].

Appendix F

Saving CPU Time

The energy or force calculation is the most time-consuming part of almost al Molecular Dynamics and Monte Carlo simulations. If we consider a mode system with pairwise additive interactions (as is done in many molecula simulations), we have to consider the contribution to the force on particle i by all its neighbors. If we do not truncate the interactions, this implies that for a system of N particles, we must evaluate $N(N-1)/2$ pair interactions And even if we do truncate the potential, we still would have to compute al $N(N-1)/2$ pair distances to describe which pairs can interact. This implie: that, if we use no tricks, the time needed for the evaluation of the energy scales as N^2. There exist efficient techniques for speeding up the evalua tion of both short-range and long-range interactions in such a way that th computing time scales as $N^{3/2}$, rather than N^2. The techniques for the long range interactions were discussed in Chapter 12.1; here, we discuss some o the techniques used for the short-range interactions. These techniques are:

1. Verlet list
2. Cell (or linked) list
3. Combination of Verlet and cell lists

F.1 Verlet List

If we simulate a large system and use a cutoff that is smaller than the simu lation box, many particles do not contribute to the energy of a particle i. I is advantageous therefore to exclude the particles that do not interact from the expensive energy calculation. Verlet [13] developed a bookkeeping tech nique, commonly referred to as the Verlet list or neighbor list, which is il lustrated in Figure F.1. In this method a second cutoff radius $r_v > r_c$ is in