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

$$\begin{aligned} x_0(t+\Delta t) &= x_0(t) + x_1(t) + x_2(t) + x_3(t) \\ x_1(t+\Delta t) &= x_1(t) + 2x_2(t) + 3x_3(t) \\ x_2(t+\Delta t) &= x_2(t) + 3x_3(t) \\ x_3(t+\Delta t) &= x_3(t). \end{aligned} \tag{E.1.1}$$

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:

$$x_n^{\text{corrected}} = x_n^{\text{predicted}} + C_n \Delta x_2, \tag{E.1.2}$$

where the C_n are constants are 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

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

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. The equations of motion are given by equations (6.1.24)–(6.1.27). Since 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). In a standard constant-N,V,E simulation, the velocity Verlet algorithm is of the following form:

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

When we use this scheme for the equations of motion (6.1.24)–(6.1.26), we obtain

$$r_i(t+\Delta t) = r_i(t) + v(t)\Delta t + [f_i(t)/m_i - \xi(t)v_i(t)]\frac{\Delta t^2}{2} \tag{E.2.1}$$

$$\begin{aligned} v_i(t+\Delta t) = {}& v_i(t) + [f_i(t+\Delta t)/m_i - \xi(t+\Delta t)v_i(t+\Delta t) \\ & + f_i(t)/m_i - \xi(t)v_i(t)]\frac{\Delta t}{2} \end{aligned} \tag{E.2.2}$$

$$\ln s(t+\Delta t) = \ln s(t) + \xi(t)\Delta t + \left(\sum_i m_i v_i^2(t) - gT\right)\frac{\Delta t^2}{2Q} \tag{E.2.3}$$

$$\begin{aligned} \xi(t+\Delta t) = {}& \xi(t) + \left\{\left[\sum_i m_i v_i^2(t+\Delta t) - gT\right]\right. \\ & \left. + \left[\sum_i m_i v_i^2(t) - gT\right]\right\}\frac{\Delta t}{2Q}. \end{aligned} \tag{E.2.4}$$

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':

$$v_i(t+\Delta t) = v'_i + [f_i(t+\Delta t)/m_i - \xi(t+\Delta t)v_i(t+\Delta t)]\frac{\Delta t}{2} \tag{E.2.5}$$

$$\xi(t+\Delta t) = \xi' + \left[\sum_i m_i v_i^2(t+\Delta t) - gT\right]\frac{\Delta t}{2Q}. \tag{E.2.6}$$

In both equations, $v_i(t+\Delta t)$ and $\xi(t+\Delta t)$ appear on the right- and left-hand sides, therefore, these equations cannot be integrated exactly.[1] For this

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

Algorithm 37 (Equations of Motion: Nosé-Hoover Thermostat A)

```
subroutine integrate1(f,en,temp)      Integrate equations of motion:
                                      with Nosé-Hoover thermostat
v2=0                                  first step velocity Verlet alg.
do i=1,npart
   fi=f(i)-xi*v(i)                    Nose Hoover force
   x(i)=x(i)+dt*(v(i)+dt*fi/2)        update positions current time
   v2=v2+v(i)**2
   v(i)=v(i)+delt*fi/2                first update velocity
enddo
fs=(v2-g*temp)/q
s=s+dt*(xi+dt*fs/2)                   update s
xi=xi+dt*fs/2                         update ξ
return
end
```

Comment to this algorithm:

1. *This subroutine performs the first step of the velocity Verlet algorithm:*

$$\begin{aligned}
r_i(t+\Delta t) &= r_i(t) + v(t)\Delta t + [f_i(t)/m_i - \xi(t)v_i(t)]\frac{\Delta t^2}{2}\\
v_i(t') &= v_i(t) + [f_i(t)/m_i - \xi(t)v_i(t)]\frac{\Delta t}{2}\\
s(t+\Delta t) &= s(t) + \xi(t)\Delta t + \left[\sum_i m_i v_i^2(t) - gT\right]\frac{\Delta t^2}{2Q}\\
\xi(t') &= \xi(t) + \left[\sum_i m_i v_i^2(t) - gT\right]\frac{\Delta t}{2Q}.
\end{aligned}$$

reason the Nosé-Hoover method is usually implemented using a predictor-corrector scheme. In fact, in the case of velocity-dependent forces, most of the advantages of Verlet-style algorithms disappear. However, as we show in Algorithm 37, a velocity Verlet algorithm can still be used.

However, it is relatively straightforward to solve equations (E.2.5) and (E.2.6) numerically. In this way we can still use the velocity Verlet algorithm. The equations that we have to solve are of the form

$$h_i(v_i,\xi) = v_i' + [f_i/m_i - \xi v_i]\frac{\Delta t}{2} - v_i = 0$$

$$h_{N+1}(v_i,\xi) = \xi' + \left[\sum_i m_i v_i^2 - gT\right]\frac{\Delta t}{2Q} - \xi = 0.$$

In this equation, we have dropped the argument $(t+\Delta t)$ for all ξ and the v_i. One possible approach to solve this equation is to use the Newton-Raphson scheme [30]; that is, to perform a Taylor expansion of h_i to lowest order:

$$h_i(x+\delta x) = h_i(x) + \sum_{j=0}^{N}\frac{\partial h_i(x)}{\partial x_j}\delta x_j.$$

In what follows, we define $x_i = v_i$ for $i = 1,\cdots,N$ and $x_i = \xi$ for $i = N+1$. At every iteration we must solve, for each i,

$$\sum_{j=1}^{N+1}\frac{\partial h_i(x)}{\partial x_j}\delta x_j = -h_i(x). \tag{E.2.7}$$

In the most general case, this would involve the inversion of an $N\times N$ matrix. However, in this case, the matrix has such a simple form that equation (E.2.7) can be solved analytically. The partial derivatives are

$$\frac{\partial h_i(x)}{\partial x_j} = \begin{cases} a = -1 & i = N+1, j = N+1\\ b_j = m_j v_j \Delta t/Q & i = N+1, j \neq N+1\\ c_i = -v_i\Delta t/2 & i \neq N+1, j = N+1\\ d = -\xi\Delta t/2 - 1 & i \neq N+1, j = i\\ 0 & \text{elsewhere.}\end{cases}$$

Substitution into equation (E.2.7), gives the following equations for $i = 0$ and $i > 0$, respectively:

$$\begin{array}{lll} c_i\delta x_{N+1} & +d\delta x_i & = -h_i\\ a\delta x_{N+1} & +\sum_{j=1}^{N} b_j\delta x_j & = -h_{N+1}.\end{array}$$

These equations have the following solution:

$$\delta x_{N+1} = \frac{h_{N+1}d - \sum_{i=1}^{N} h_i b_i}{-ad + \sum_{i=1}^{N} b_i c_i} \tag{E.2.8}$$

$$\delta x_i = \frac{1}{d}(h_i - c_i\delta x_{N+1}). \tag{E.2.9}$$

With these equations one can make a very efficient implementation of the Newton-Raphson scheme [30]. In Algorithms 37 and 38, an example is given of the implementation. Note that, compared to Algorithm 15, we have separated the integration into two separate routines, one for each step in the

Algorithm 38 (Equations of Motion: Nosé-Hoover B)

```
  subroutine integrate2(f,en,temp)     integrate equations of motion
  v2=0                                 Nosé-Hoover thermostat
  do i=1,npart                         2nd step velocity Verlet alg.
     vn(i)=v(i)
     v2=v2+vn(i)**2
  enddo
  xin=xi
  ready=.false.                        start Newton-Raphson loop
  do while (.not.ready)
    xio=xin                            store previous value
    delxi=0
    do i=1,npart                       solve equation (E.2.7)
      vo(i)=vn(i)
      h(i)=vo(i)-v(i)
+         -(f(i)-xio*vo(i))*dt/2
      bi=vo(i)*dt/q
      delxi=delxi-h(i)*bi
    enddo
    d=-xio*dt/2-1
    h(0)=xio-xin
+       +d*(-v2-g*temp)*dt/(2*q)
    cibi=-v2*dt**2/(2*q)
    delxi=(delxi+h(0)*d)/(d-cibi)      from equation (E.2.8)
    xin=xio+delxi                      new guess for ξ
    v2=0
    do i=1,npart
     ci=-vo(i)*dt/2                    new guess v_i
     vn(i)=vo(i)+(h(i)-ci*delxi)/d     from equation (E.2.9)
     v2=v2+vn(i)**2
    enddo
    ready=.true.                       test for convergence
    i=0
    do while (i.le.npart.and.ready)
      i=i+1
      if (i.le.npart) then
       if (abs((vn(i)-vu(i))/vn(i))
+          .gt.err) ready=.false.
      else
        if (abs((xin-xio)/xin)
+          .gt.err) ready=.false.
      endif
      ...(continue)....
```

```
      ...(continue)....
    enddo
  enddo
  do i=1,npart
     v(i)=vn(i)                        converged velocity
  enddo
  xi=xin
  ham=en+v2/2+(xi**2*q)/2+g*temp*s     conserved quantity
  return
  end
```

Comments to this algorithm:

1. *This subroutine performs the second step of the velocity Verlet algorithm:*

$$\begin{aligned} v_i(t+\Delta t) &= v_i(t') + [f_i(t+\Delta t)/m_i - \xi(t+\Delta t)v_i(t+\Delta t)]\frac{\Delta t}{2} \\ \xi(t+\Delta t) &= \xi(t') + \left[\sum_i m_i v_i^2(t+\Delta t) - gT\right]\frac{\Delta t}{2Q}. \end{aligned}$$

   The Newton-Raphson scheme is used to solve these equations numerically.

2. *The term* `ham`*, defined in equation (6.1.28), is a quantity that needs to be conserved and therefore is a useful check of the algorithm.*
3. *The term* `err` *is the convergence criteria.*

velocity Verlet algorithm. For the Lennard-Jones fluid (see, Case Study 11), we find that in approximately three iterations the results have converged to an accuracy of 1 in 10^{10}.

For the harmonic oscillator discussed in section 6.1.3, the iterative scheme just described is not required. Substitution of equation (E.2.6) into equation (E.2.5), yields the following cubic equation in v:

$$a_3 v^3(t+\Delta t) + a_1 v(t+\Delta t) + a_0 = 0$$

with

$$\begin{aligned} a_3 &= \frac{\Delta t^2}{4Q} \\ a_1 &= [\xi(t') - gT]\frac{\Delta t^2}{4Q} - 1 \end{aligned}$$