Numerical Methodology

The preceding lecture focused on the description of a system comprising $N$ particles, the intermolecular forces, the governing equations describing the dynamics of $N$ particles, and the boundary conditions. In this lecture we will discuss some of the numerical methods used to solve the dynamics equations. The computation of the trajectories of $N$ particles involves the solution of a system of $6N$ coupled, first-order ordinary differential equations. A variety of numerical methods are available to solve these equations. We will discuss some of these methods and other related issues relevant to MD simulations. These methods may be broadly classified into four groups:

1. Runge-Kutta (R-K) Methods
2. Leap Frog Methods
3. Multi-Step and Predictor Corrector Methods
4. Gear Predictor-Corrector Method

9.1 Runge-Kutta (R-K) Methods

R-K methods of various orders have been commonly used to solve ordinary differential equations (ODEs) involving initial value problems. In order to understand the fundamental concepts, let us start with the basic Euler method used to solve a first-order ODE:

$$\frac{dx}{dt} = \dot{x} = f(x,t) \quad \text{with} \quad x(t=0)=x_0 \quad (9.1)$$

over an interval $x=a$ to $x=b$.

As discussed in Lecture 8, the particle dynamics in a one-dimensional formulation is governed by two ODEs of the type given in Eq. (9.1). The integration of Eq. (9.1) yields
Here $h=\Delta t$ represents the temporal step size and $x$ the particle position. Various R-K methods differ essentially in the approximation used to evaluate the above integral. For the Euler method, $f(x,t)$ is assumed to be constant and equal to its value $f(x_i,t)$ over the interval $dt$. This is equivalent to writing the Taylor series for $x(t+h)$ as

$$x_{i+1} = x_i + h \dot{x}(t) + O(h^2) \quad (9.3)$$

The method is first-order accurate with the local truncation error being proportional to $h^2$, or $O(h^2)$. It can easily be shown that while the local truncation error of $O(h^2)$, the global truncation error, which is accumulated over the entire computational interval $(b-a)$ is $O(h)$.

The Euler method can also be considered as the first-order R-K method, since various R-K methods differ in how the slope is evaluated over the interval $t$ and $t+h$. Each R-K method evaluates the slope at different points in the interval $(h)$, and computes a weighted average that is used in Eq. (9.2). For example, a second-order R-K method yields

$$x_{i+1} = x_i + \frac{1}{2} h (k_1 + k_2) \quad (9.4)$$

with $k_1 = f(x_i,t)$ and $k_2 = f(x_i + k_1, t + h)$.

This method has a local truncation error (l te) of $O(h^3)$, and requires two computations of the function $f(x,t)$ for each time step. Similarly a fourth-order R-K method, which is commonly used for solving a system of ODEs, has a lte of $O(h^5)$ and requires four computations of the function $f(x,t)$ for each time step. Note that the particle velocity is calculated using the same approach, with $f(x,t)$ now representing the force (or acceleration in terms of the non-dimensional variables). Further details can be found in Ref. 7.
R-K methods are self-starting, easy to program, and the step size can be automatically controlled to achieve the desired accuracy. However, these methods are less desirable for MD simulations due to two considerations. One is that higher-order R-K methods do not have any significant advantage over the low-order R-K methods. This is because the accuracy of trajectory calculations, due to the strong repulsive forces at short distances, puts an upper limit on the temporal step size \( h \). The second consideration is the number of evaluations of particle force or acceleration per time step. For example, a fourth-order R-K method involves four evaluations per time step. Consequently, when MD simulations involve a long time period, the R-K methods would require excessive amount of computational time. Then other methods such as predictor-corrector methods are preferred.

### 9.1.1 Various Types of Errors

Here we will discuss two major types of errors; truncation error and round-off error (roe). As indicated above, the Euler method, Eq. (9.3), has a local truncation error given by

\[
\text{lte} = \frac{h^2}{2} \ddot{x}(t) \quad (9.4.1)
\]

A common terminology to quantify this error is to state that the local truncation error is of the order of \( h^2 \) or \( O(h^2) \). Also, it can easily be shown that the total truncation error accumulated over a finite number of steps is \( O(h) \). The corresponding local and total truncation errors for the second-order R-K method are \( O(h^3) \) and \( O(h^2) \). Thus the truncation error is determined by the time step \( h = \Delta t \) and the accuracy of the numerical algorithm used. The roe on the other hand depends on the number of significant figures used in the computation, the order by which the computations are performed, and the approximations used for calculating square roots, exponentials, logarithms etc. Since roe depends on the total number of computations or time steps in the simulation, it also depends on \( \Delta t \). The global error is the sum of truncation error and round-off error, and, thus, clearly depends on \( \Delta t \). Generally speaking as \( \Delta t \) is decreased, the te decreases, while the roe increases. Thus the choice of \( \Delta t \) should be such that it corresponds to the minimum total error, as shown in Fig. 9.1, taken from Haile [1].
Fig. 9.1 Global error plotted versus the non-dimensional time step.

For this figure, the average global error is computed in terms of the non-dimensional energy of the system and plotted versus the non-dimensional time step. The average global error is defined as

$$< ge > = \sqrt{\frac{1}{M} \sum_{k=1}^{M} \left[ E^*(0) - E^*(k\Delta^*) \right]^2} \quad (9.4.2)$$

where $E^*$ is the non-dimensional energy defined as $E^* = E/e$, $\Delta^* = \Delta t/(\sigma m e)^{1/2}$ is the non-dimensional time, and $M$ the number of time steps. The total error in energy is computed for a one-dimensional
collision between two Lennard-Jones atoms using the Gear's predictor-corrector method, which is discussed later in this lecture. Results are shown for simulations performed using both the single- and double-precision. As indicated in the figure, for $\Delta t^*5x10^{-3}$, the te makes the largest contribution to the global error, and thus, the single- and double-precision computations yield the same global error, i.e. the roe is of no consequence. However, for $\Delta t^*<10^{-3}$ global errors with single precision are dominated by round-off, but in double-precision roe remain unimportant for $\Delta t^*$ as small as $10^{-5}$. There are many ways to monitor the total error in the simulation. As indicated in the above example, a common procedure in MD simulations is to calculate the total error by monitoring the total energy as a function of time. For additional discussion on various types of errors, the reader is referred to Refs. [1, 7]. Note all the references pertaining to the material on MD simulations are listed in numerical order in Lecture 7.

9.2 Leap Frog Method

Writing the Taylor series for $x(t+h)$ as

$$x_{i+1} = x(t + h) = x_i + h\left(\dot{x}(t) + \frac{h}{2} \ddot{x}(t)\right) + O(h^3) \quad (9.5)$$

Here $\dot{x}(t)$ and $\ddot{x}(t)$ represent the particle velocity and acceleration respectively. The above expression can also be written as

$$x_{i+1} = x_i + h \dot{x}(t + h/2) + O(h^3) \quad (9.6)$$

where $\dot{x}(t + h/2)$ is expressed as

$$\dot{x}(t + h/2) = \dot{x}(t) + \frac{h}{2} \ddot{x}(t) + O(h^2) \quad (9.7)$$

The new particle position is calculated using Eqs. (9.6) and (9.7), while the particle velocity is calculated from

$$\dot{x}(t + h/2) = \dot{x}(t - h/2) + h \ddot{x}(t) + O(h^2) \quad (9.8)$$
Here the particle acceleration is obtained from the net force on the particle using the approach described in Lecture 8. Note that the particle position and velocity are being calculated at different times, which does not represent any problem. In describing initial conditions, the distinction between $\dot{x}(0)$ and $\dot{x}(h/2)$ is also ignored. Another method which has been used more commonly compared to the leap frog method is the Verlet algorithm.

**9.2.1 Verlet Algorithm**

Writing the Taylor series for $x(t-h)$ and combining it with Eq. (9.5) yields the Verlet formula

$$x(t + h) = 2x(t) - x(t - h) + h^2 \ddot{x}(t) + O(h^4) \quad (9.9)$$

which is third-order accurate, and requires only one force evaluation per time step. The velocity is then obtained using

$$\dot{x}(t) = \frac{x(t + h) - x(t - h)}{2h} + O(h^2) \quad (9.10)$$

Higher-order algorithms are available to compute velocity, but generally are not used. The Verlet algorithm is a two-step method, and not self-starting since it requires $x(t-h)$. A special procedure is needed to obtain $x$ at $t=-h$. The main advantages of this method are its efficiency, simplicity, relatively higher-order accuracy, and good stability characteristics.

**9.3 Multi-Step and Predictor-Corrector Methods**

These methods generally involve two steps for integration over one time step ($h=\Delta t$); the predictor step in which the particle position and velocity are predicted, followed by the corrector step in which the predicted values are corrected in order to improve the accuracy and/or the stability of the algorithm. Either of these steps can employ a self-starting method, such as Runge-Kutta methods, or a multi-step method, although the latter type has been used more commonly. In this section, we will first discuss the multi-step methods, followed by a discussion on the predictor-corrector algorithm. We will end the
section by discussing a specific predictor-corrector methodology that is commonly used in MD simulations. The starting point for deriving formulas for multi-step methods is the Taylor series for \( x(t+h) \)

\[
x_{i+1} = x(t+h) = x_i + h \dot{x}(t) + \frac{1}{2} h^2 \ddot{x}(t) + O(h^3) \quad (9.11)
\]

Writing \( \ddot{x}(t) \) as

\[
\ddot{x}(t) = \frac{\dot{x}(t) - \dot{x}(t-h)}{h} + O(h)
\]

we obtain a formula for a second-order multi-step method

\[
x(t+h) = x_i + \frac{h}{2} (3\dot{x}(t) - \dot{x}(t-h)) + O(h^3) \quad (9.12)
\]

Since this formula requires information at two time steps, \( t \) and \( t-h \), the algorithm is not self-starting. Similarly, we can derive higher-order formulas by considering higher-order terms in the Taylor series, and using finite-difference approximations for these terms. For example, considering the third-order term in Eq. (9.11), and using approximation for \( \dddot{x}(t) \) and \( x^{(3)}(t) \) as

\[
\dddot{x}(t) = \frac{3 \dot{x}(t) - 4 \dot{x}(t-h) + \dot{x}(t-2h)}{2h} + O(h^2) \quad (9.13)
\]

\[
x^{(3)}(t) = \frac{\dddot{x}(t) - 2 \dddot{x}(t-h) + \dddot{x}(t-2h)}{h^2} + O(h)
\]

yields a third-order multi-step algorithm, i.e.,

\[
x(t+h) = x_i + \frac{h}{12} (23 \dot{x}(t) - 16 \dot{x}(t-h) + 5 \dot{x}(t-2h)) + O(h^4) \quad (9.14)
\]

Note the superscript (3) with \( x \) is used to represent the third derivative of \( x \). Again the algorithm is not self-starting, as it requires information
at three time steps. Multi-step algorithms with fourth- and higher-order accuracy can be obtained following a similar procedure. Similar formulas can be used for computing the particle velocity at the new step. The major advantage of multi-step methods is their efficiency since only one function evaluation is needed at each time step, although the storage requirement is increased. The above formulas provide explicit expressions for the unknown variable, and, therefore, are known as explicit methods. Another class of multi-step methods involves implicit expressions for the unknown. To derive such expressions, we write the Taylor series for \( x(t) \) as

\[
x(t) = x(t) + h \dot{x}(t) + \frac{1}{2} h^2 \ddot{x}(t) + \frac{1}{6} h^3 x^{(3)}(t) + O(h^4)
\]

and then rewrite it in terms of \( x(t+h) \)

\[
x(t+h) = x_i + h \dot{x}(t+h) - \frac{1}{2} h^2 \ddot{x}(t+h) + \frac{1}{6} h^3 x^{(3)}(t+h) + O(h^4) \quad (9.15)
\]

Using the following finite-difference approximations for \( \ddot{x}(t+h) \) and \( x^{(3)}(t+h) \)

\[
\ddot{x}(t+h) = \frac{3 \ddot{x}(t+h) - 4 \ddot{x}(t) + \ddot{x}(t-h)}{2h} + O(h^2) \quad (9.16)
\]

\[
x^{(3)}(t+h) = \frac{\dot{x}(t+h) - 2 \dot{x}(t) + \dot{x}(t-h)}{h^2} + O(h) \]

we obtain a third-order implicit formula for \( x(t+h) \)

\[
x(t+h) = x_i + \frac{h}{12} \left( 5 \ddot{x}(t+h) + 8 \ddot{x}(t) - \ddot{x}(t-h) \right) + O(h^4) \quad (9.17)
\]

Similarly we can obtain higher-order implicit formulas.
9.3.1 Predictor-Corrector Methods

The predictor-corrector methods consist of two steps; the predictor step and the correction step. The predictor step generally employs an explicit formula, while the correction step employs an implicit formula. For example, we may use the third-order explicit formula, Eq. (9.14), in the predictor step to calculate an intermediate value of particle position, i.e.,

\[ x^*(t+h) = x_i + \frac{h}{12} \left( 23\dot{x}(t) - 16\dot{x}(t-h) + 5\dot{x}(t-2h) \right) + O(h^4) \]  \hspace{1cm} (9.14)

A similar formula is used to calculate the intermediate value of particle velocity. These values are then corrected by using a third-order implicit formula, i.e., Eq. (9.17) as

\[ x(t+h) = x_i + \frac{h}{12} \left( 5\dot{x}^*(t+h) + 8\dot{x}(t) - \ddot{x}(t-h) \right) + O(h^4) \]  \hspace{1cm} (9.17)

where \( x^*(t+h) \) represents the intermediate particle velocity calculated in the predictor step. Again a similar formula is used to calculate the particle velocity in the corrector step. The particle acceleration, \( \ddot{x}(t+h) \), required in the corrector step, is calculated using the particle position, \( x^*(t+h) \), which has been obtained in the predictor step. Since many different explicit and implicit multi-step formulas can be used, a variety of predictor-corrector algorithms are available in the literature [7]. Here we will discuss a fifth-order predictor-corrector algorithm due to Gear [8], which has been commonly used in MD simulations.

9.4 Gear Predictor-Corrector Method

The algorithm consists of the following three steps.

Predictor Step

Taylor series is used to predict \( x(t+h) \) and higher derivatives
The higher order derivatives needed in the above equations are taken from the previous simulation. For starting the simulation, the second-order derivative is obtained by calculating the force and then using the Newton’s second law, while higher order derivatives at \( t=0 \) are assumed to be zero.

**Evaluation Step**

This step involves the determination of forces (or accelerations) using the intermolecular potential discussed earlier. The total force on particle \( i \) can be written as

\[
F_i = - \sum_{j \neq i} \frac{\partial u(r_{ij})}{\partial r_{ij}} \hat{r}_{ij} \tag{9.24}
\]

For example, using the Lennard-Jones potential, the force \( F_i \) can be written as

\[
F_i = 24 \frac{\varepsilon}{\sigma} \sum_{j \neq i} \left[ 2 \left( \frac{\sigma}{r_{ij}} \right)^{13} - \left( \frac{\sigma}{r_{ij}} \right)^7 \right] \hat{r}_{ij} \tag{9.25}
\]
where \( \hat{r}_{ij} \) represents the unit vector in the \( r_{ij} \) direction. The force \( F_i \) at the advanced time is computed using the particle positions calculated in the predictor step. Note that in the discussion of numerical methods, the particle position has been represented interchangeably by \( x \) and \( r \). The computation of force is the most time-consuming part of the simulation, since the sum in Eq. (9.24) involves forces due to all the particles in the simulation. Several strategies are used to reduce the computational effort involved in this step. One is to employ the Newton's third law for forces on particles \( i \) and \( j \), i.e.,

\[
F_{ij} = -F_{ji} \quad (9.26)
\]

which reduces the computational effort by a factor of 2. Another strategy is to use the truncated potential and thus the truncated force. Equation (9.25) is then modified for the truncated force as

\[
\begin{align*}
F_i &= 24 \frac{\epsilon}{\alpha} \sum_{j<i} \left[ 2 \left( \frac{\alpha}{r_{ij}} \right)^{13} - \left( \frac{\alpha}{r_{ij}} \right)^7 \right] \hat{r}_{ij} \quad \text{for } r_{ij} \leq r_c \quad (9.27a) \\
F_i &= 0 \quad \text{for } r_{ij} > r_c 
\end{align*}
\]

As discussed earlier, this leads to a significant reduction in the computational effort. The effect of using the truncated potential and the periodic boundary conditions (PBC) on \( F_{ji} \) in the context of the truncated potential and force has also been discussed earlier. Their effect on the simulation and the predicted properties will be discussed later.

**Correction Step**

An error or a correction term is now obtained using the acceleration computed from Eq. (9.27) and that from Eq. (9.20). The error term can be written as
\[ \Delta \ddot{r}_i = \left( \ddot{r}_i(t + h) - \ddot{r}_i^P(t + h) \right) \quad \text{(9.28)} \]

or \[ \Delta R2 = \frac{h^2}{2} \Delta \ddot{r}_i \]

The superscript P is used to indicate the accelerations computed from Eq. (9.20). Again it should be noted that there is no distinction between x and r, as both are used to represent the particle position. This correction term is used to correct the predicted values as

\[ r_i = r_i^p + \alpha_0 \Delta R2 \quad \text{(9.29)} \]
\[ \dot{r}_i = \dot{r}_i^p h + \alpha_1 \Delta R2 \quad \text{(9.30)} \]
\[ \ddot{r}_i = \frac{\dot{r}_i^p}{2} h^2 + \alpha_2 \Delta R2 \quad \text{(9.31)} \]
\[ r_i^{(3)} = r_i^{(3)p} \frac{h^3}{6} + \alpha_3 \Delta R2 \quad \text{(9.32)} \]
\[ r_i^{(4)} = r_i^{(4)p} \frac{h^4}{24} + \alpha_4 \Delta R2 \quad \text{(9.33)} \]
\[ r_i^{(5)} = r_i^{(5)p} \frac{h^5}{120} + \alpha_5 \Delta R2 \quad \text{(9.34)} \]

The values of the parameters \( \alpha_i \) depend upon the accuracy of the formulas used in the predictor and corrector steps. For a third-order predictor-corrector method, the values of the \( \alpha_i \) are \( \alpha_0=1/6, \alpha_1=5/6, \alpha_2=1, \alpha_3=1/3 \). For the fifth-order Gear method discussed here, the corresponding values are: \( \alpha_0=3/16, \alpha_1=325/360, \alpha_2=1, \alpha_3=11/18, \alpha_4=1/6, \) and \( \alpha_5=1/60 \). These values are obtained by performing a linear stability analysis [8]. Equations (9.29) and (9.30) provide the particle position and velocity at the new time step. Equation (9.30) is used to compute the force at the new time step, while Eqs. (9.32), (9.33), and (9.34) provide the higher derivatives which are then used
in Eqs. (9.18)-(9.23), and the computations are continued for the required number of time steps. It is important to note that the particle positions obtained using Eq. (9.29) are further modified using the periodic boundary conditions.