Chapter 4

Molecular Dynamics

Literature

- D.C. RAPAPORT, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, 1995.

It is still an important field of research to describe macroscopically observable properties of matter on the basis of microscopical kinematics and dynamics of molecules. On the other hand, the simultaneous motion of a large number of interacting bodies cannot be described analytically and, thus, statistical mechanics is required to make some simplifying assumptions in order to arrive at practical solutions. Nevertheless, it is hard to make estimates on the influence those simplifications might have on the solutions acquired. From this the necessity of numerical simulations becomes obvious. There are essentially two methods to determine physical quantities over a restricted set of states, namely Molecular Dynamics (MD) and Monte Carlo (MC).

Molecular dynamics is a widely used method to study classical systems. But quantum systems can also be studied by MD and the number of applications is steadily increasing. In quantum problems one is interested in the motion of atoms in molecules and solids. Due to their much lighter mass, the electron dynamics is much faster than that of the nuclei and it is in most cases a good approximation, to treat the dynamics of the nuclei classically, *i.e.:* by NEWTON's equation of motion

$$\ddot{x}_{i\alpha}(t) = \frac{1}{m_i}\, F_{i\alpha}[\mathbf{r}(t), t] = f_{i\alpha}[\mathbf{r}(t), t], \tag{4.1}$$

where $x_{i\alpha}$ is the α-th coordinate of particle i. The vector $\mathbf{r}$ stands for the collection of coordinates $\{x_{i\alpha}\}$. Similarly, we introduce the vector $\mathbf{f}$ for the set $\{f_{i\alpha}\}$, and all this results in the equation of motion:

$$\ddot{\mathbf{r}}(t) = \mathbf{f}[\mathbf{r}(t), t]. \tag{4.2}$$

The force acting on the nuclei originates from the direct Coulomb interaction between the nuclei plus the indirect contribution stemming from electrons. The latter is either approximated by parameterized two-particle potentials (e.g. LENNARD-JONES) or it is determined quantum mechanically from the electronic ground state (CAR-PARRINELLO). MD is widely used for studying many-particle systems. It consists of integrating the equations of motion numerically. It can, therefore, be viewed as a simulation of the system as it develops over a period of time. The dynamics coincides with the actual trajectories, as compared to Monte-Carlo dynamics. The great advantage of MD is that it provides both, thermodynamic averages and dynamical properties even far from equilibrium.

For finite systems boundary conditions are important. The vast majority of MD simulations is performed for periodic boundary conditions (pbc), which means that the finite system is surrounded by identical systems with exactly the same configuration in phase-space. Forces act across the boundary of neighboring replicas. Another common situation are open boundary conditions (obc). The angular momentum is not conserved if pbc are imposed.

The time average is restricted to finite times for obvious reasons. For liquid argon, which is a widely studied system in MD since it can be described reliably by simple LENNARD-JONES pair forces, the typical time step used in the numerical integration of the equations of motion is about 10^{-14} seconds, which means that a total simulation time of about 10^{-8} seconds can be covered; this corresponds to 10^6 time steps. The correlation time has to be much smaller than this in order to yield reasonable results. In addition, the influence of the finite system size will be noticeable after some time. One would expect to observe differences as soon as the particles have travelled on average more than half the linear system size. In practice such effects show up at much longer time scales, which are of the order of the recurrence time.

The numerical integration algorithm is not infinitely accurate. There is always a tradeoff between accuracy and speed. As a matter of fact, the MD trajectory will gradually deviate from the true trajectory the system would follow in reality.

4.1 MD at constant energy

If the forces acting on the particles depend on the their mutual relative position, then energy and total momentum are conserved. Trivially, particle number and volume are conserved, as well. The time averages correspond to microcanonical or (NVE) ensembles. Here we describe the microcanonical ensemble in some detail. The rough structure of the algorithm is

- Initialization,
- Start simulation and let the system reach equilibrium,
- Continue simulation and store (measure) results.

Initialization:

The number of particles and the finite size volume are specified. The temperature is usually of greater interest than the total energy and is therefore specified as an input parameter. We will discuss below how to fix the temperature in a MD simulation.

The particles are assigned positions and momenta. Typically the positions are chosen on a regular grid or at random, while the momenta (velocities) $\mathbf{p}_i$ are generated according to the BOLTZMANN distribution

$$p(v_\alpha) \propto e^{-mv_\alpha^2/(2k_BT)}.$$

A vanishing total momentum is achieved by subtracting the mean momentum $\overline{\mathbf{p}_i}$ from all particle momenta.

Measurement

Of particular interest are growth processes, phase transitions, molecular vibrations, reactions etc. The thermodynamic (ensemble) average is obtained via averaging over time. For an observable O the expectation value reads

$$\langle O \rangle = \lim_{T\to\infty} \frac{1}{T}\int_0^T dt\, O(t).$$

For instance, the inner energy $U = \langle E \rangle$. The MD simulations are performed for constant energy first. For a microcanonical ensemble, constant temperature simulations are required, which can be achieved by rescaling the kinetic energy from time to time. The thermodynamic average reads

$$\langle O \rangle = \frac{\sum_x e^{-\beta E(x)} O(x)}{\sum_x e^{-\beta E(x)}}.$$

The virial theorem (see Appendix B)

$$\frac{3}{2}PV = \frac{1}{2}\left\langle \sum_{j=1}^{N} \mathbf{p}_j \cdot \frac{\partial H}{\partial \mathbf{p}_j} \right\rangle - \frac{1}{2}\left\langle \sum_{j>k=1}^{N} \mathbf{r}_{jk} \cdot \frac{\partial \Phi(\mathbf{r}_{jk})}{\partial \mathbf{r}_{jk}} \right\rangle$$

can be exploited in order to determine the pressure. It holds for a stationary motion. Obviously, the first term is the kinetic energy, which in a microcanonical ensemble is given by $\frac{3}{2}k_B TN$, hence

$$\frac{3}{2}PV = \frac{3}{2}k_B TN - \frac{1}{2}\left\langle \sum_{j>k=1}^{N} \mathbf{r}_{jk} \cdot \frac{\partial\Phi(\mathbf{r}_{jk})}{\partial\mathbf{r}_{jk}} \right\rangle$$

$$\beta P/N = 1 - \frac{\beta}{3N}\left\langle \sum_{j>k=1}^{N} \mathbf{r}_{jk} \cdot \frac{\partial\Phi(\mathbf{r}_{jk})}{\partial\mathbf{r}_{jk}} \right\rangle,$$

with $\beta = 1/(k_B T)$ and k_B is BOLTZMANN's constant. The right hand site can easily be measured during the simulation.

4.1.1 Verlet algorithm

We will concentrate on the most widely used algorithm, which is both, simple and yet reliable, the Verlet algorithm.

$$\mathbf{r}(\tau) = \mathbf{r}(0) + \tau\,\dot{\mathbf{r}}(0) + \frac{\tau^2}{2}\ddot{\mathbf{r}}(0) + \frac{\tau^3}{6}\dddot{\mathbf{r}}(0) + \mathcal{O}(\tau^4)$$
$$\mathbf{r}(-\tau) = \mathbf{r}(0) - \tau\,\dot{\mathbf{r}}(0) + \frac{\tau^2}{2}\ddot{\mathbf{r}}(0) - \frac{\tau^3}{6}\dddot{\mathbf{r}}(0) + \mathcal{O}(\tau^4)$$

$$\mathbf{r}(\tau) + \mathbf{r}(-\tau) = 2\mathbf{r}(0) + \tau^2\,\mathbf{f}[\mathbf{r}(0), 0] + \mathcal{O}(\tau^4).$$

The Verlet algorithm uses the relation

$$\mathbf{r}(\tau) = 2\mathbf{r}(0) - \mathbf{r}(-\tau) + \tau^2\,\mathbf{f}[\mathbf{r}(0), 0]. \tag{4.3}$$

Since, $\mathbf{r}(-\tau)$ is not known, one employs the simpler discretization

$$\mathbf{r}(\tau) = \mathbf{r}(0) + \tau\,\mathbf{v}(0) + \frac{\tau^2}{2}\,\mathbf{f}[\mathbf{r}(0), 0]. \tag{4.4}$$

for the first time step. The corresponding discretization error is $\mathcal{O}(\tau^3)$, which occurs, however, only once, while the $\mathcal{O}(\tau^4)$ error is encountered in each time step and also adds up to $N\mathcal{O}(\tau^4) = \mathcal{O}(\tau^3)$.

VERLET ALGORITHM

$$\mathbf{r}(t+\tau) = 2\mathbf{r}(t) - \mathbf{r}(t-\tau) + \tau^2\,\mathbf{f}[\mathbf{r}(t), t], \qquad t = \tau, 2\tau, \ldots$$
$$\mathbf{r}(\tau) = \mathbf{r}(0) + \tau\,\mathbf{v}(0) + \frac{\tau^2}{2}\,\mathbf{f}[\mathbf{r}(0), 0].$$

There is another alternative which is more robust against rounding errors. For the velocities

$$\frac{\mathbf{v}(\tau/2) - \mathbf{v}(-\tau/2)}{\tau} = \dot{\mathbf{v}}(0) + \mathcal{O}(\tau^2)$$

we obtain

$$\mathbf{v}(\tau/2) = \mathbf{v}(-\tau/2) + \tau\, \mathbf{f}[\mathbf{r}(0), 0] + \mathcal{O}(\tau^3).$$

We use, likewise, for the positions

$$\frac{\mathbf{r}(\tau/2 + \tau/2) - \mathbf{r}(\tau/2 - \tau/2)}{\tau} = \dot{\mathbf{r}}(\tau/2) + \mathcal{O}(\tau^2)$$

which leads to

$$\mathbf{r}(\tau) = \mathbf{r}(0) + \tau\, \mathbf{v}(\tau/2) + \mathcal{O}(\tau^3).$$

This approach is called the leap-frog algorithm due to the way, the time axis is 'touched'. Space-coordinates are computed at $0, \tau, 2\tau, \ldots$ and velocities are provided at intermediate times $\tau/2, 3\tau/2, \ldots$.

LEAP-FROG ALGORITHM I

$$\begin{aligned}
\mathbf{r}(t+\tau) &= \mathbf{r}(t) + \tau \cdot \mathbf{v}(t+\tau/2) && t = 0, \tau, \ldots \\
\mathbf{v}(t+\tau/2) &= \mathbf{v}(t-\tau/2) + \tau \cdot \mathbf{f}[\mathbf{r}(t), t] && t = \tau, 2\tau, \ldots \\
\mathbf{v}(\tau/2) &= \mathbf{v}(0) + \frac{\tau}{2} \cdot \mathbf{f}[\mathbf{r}(0), 0].
\end{aligned}$$

Next we consider yet another modification, which we call leap-frog II.

$$\begin{aligned}
\mathbf{r}(t+\tau) &= \mathbf{r}(t) + \tau \cdot \mathbf{v}(t) + \frac{\tau^2}{2}\ddot{\mathbf{r}}(t) + \mathcal{O}(\tau^3) \\
&= \mathbf{r}(t) + \tau \cdot \mathbf{v}(t) + \frac{\tau^2}{2}\mathbf{f}[\mathbf{r}(t), t] + \mathcal{O}(\tau^3)
\end{aligned}$$

and

$$\mathbf{v}(t+\tau) = \mathbf{v}(t) + \tau \ddot{\mathbf{r}}(t+\tau/2) + \mathcal{O}(\tau^3).$$

This approach would require the knowledge of $\ddot{\mathbf{r}}(t+\tau/2)$, which can be approximated by

$$\frac{\ddot{\mathbf{r}}(t+\tau) + \ddot{\mathbf{r}}(t)}{2} = \ddot{\mathbf{r}}(t+\tau/2) + \mathcal{O}(\tau^2)\ .$$

Hence, we end up with

LEAP-FROG ALGORITHM II

$$
\begin{aligned}
\mathbf{r}(t+\tau) &= \mathbf{r}(t) + \tau \cdot \mathbf{v}(t) + \frac{\tau^2}{2}\mathbf{f}[\mathbf{r}(t), t] \\
\mathbf{v}(t+\tau) &= \mathbf{v}(t) + \tau\frac{1}{2}\left\{\mathbf{f}[\mathbf{r}(t), t] + \mathbf{f}[\mathbf{r}(t+\tau), t+\tau]\right\}.
\end{aligned}
$$

This form is completely equivalent to the other two schemes as long as all computational steps are performed with infinite accuracy, but it is less susceptible to numerical errors.

4.1.2 Example I: Harmonic Oscillator

Here we consider the harmonic oscillator since it allows to asses the stability of the algorithms analytically. The equation of motion for the 1D harmonic oscillator reads

$$\ddot{x} = -\omega^2\, x.$$

For $t > 0$ the Verlet algorithm reads

$$x(t+\tau) - 2x(t) + x(t-\tau) = -\tau^2\omega^2\, x(t), \tag{4.5}$$

For $t = 0$ we have $\dot{v} = f$ and, therefore, $v(\tau) = v(0) + \tau f[x(0)]$. Along with the initial condition $v(0) = 0$ we have

$$x(\tau) = x(0) - \frac{\tau^2}{2}\omega^2 x(0). \tag{4.6}$$

Eqs. (4.5) can be expressed in matrix form if we use the assignment $x_n = x(n\,\tau)$ for $n \in \mathbb{N}_0$:

$$
\underbrace{\begin{pmatrix}
1 & -2 & 1 & & \\
 & 1 & -2 & 1 & \\
 & & 1 & -2 & \ddots \\
 & & & \ddots & \ddots
\end{pmatrix}}_{:=\boldsymbol{M}} \mathbf{x} = -\tau^2\omega^2\mathbf{x}.
$$

Zero matrix elements are supressed. Thus, the eigenvalue equation $\boldsymbol{M}\mathbf{x} = \lambda\mathbf{x}$ has to solved for the given eigenvalue $\lambda = -(\tau\omega)^2$. Since $x(t) = e^{i\omega t}$ is the exact solution for the harmonic oscillator, we try the ansatz

$$x_n = e^{i\alpha\, n}. \tag{4.7}$$

The general condition, for $\tau > 0$, reads

$$e^{i\alpha(n+1)} - 2e^{i\alpha n} + e^{i\alpha(n-1)} = -(\tau\omega)^2 \, e^{i\alpha n}$$

or rather

$$e^{i\alpha} - 2 + e^{-i\alpha} = -(\tau\omega)^2.$$

Hence,

$$1 - (\tau\omega)^2/2 = \cos(\alpha). \tag{4.8}$$

This implicit equation for α has real solutions only for

$$-1 \leq (\tau\omega)^2/2 - 1 \leq 1$$

or rather

$$0 \leq \tau\omega \leq 2.$$

In other words, the discretization has to obey

$$\tau \leq \frac{2}{\omega}, \tag{4.9}$$

otherwise, α becomes complex and the solution (4.7) obtains exponentially increasing components, as we see in Fig. 4.3. Equation (4.8) has always two roots, namely $\pm\alpha^*$. Hence, the general solution, obeying the initial condition reads

$$x(n\tau) = Ae^{i\alpha n} + Be^{-i\alpha n} = a\cos(\alpha n + \varphi),$$

with parameters which are fixed by the initial conditions $x(t = 0) = x(0)$ and $v(0) = 0$, *i.e.:* $x(0) = a\cos(\varphi)$. We still have to satisfy the first equation, Eq. (4.6)

$$\begin{aligned} a\cos(\alpha + \varphi) &= a\cos(\varphi) - \frac{(\tau\omega)^2}{2} a\cos(\varphi) \\ a\cos(\alpha + \varphi) &= a\cos(\varphi)\left[1 - \frac{(\tau\omega)^2}{2}\right]. \end{aligned}$$

Along with Eq. (4.8) we have

$$\begin{aligned} \cos(\alpha + \varphi) &= \cos(\varphi)\cos(\alpha) \\ \cos(\alpha)\cos(\varphi) - \sin(\alpha)\sin(\varphi) &= \cos(\varphi)\cos(\alpha) \\ \sin(\varphi) &= 0. \end{aligned}$$

The solution, therefore, reads

$$x_n = x_0 \cos(\alpha n). \tag{4.10}$$

In the original representation this yields (continuum limit) $x(t) = x(0)\cos(\frac{\alpha}{\tau}t)$, which corresponds to the exact solution for $\frac{\alpha}{\tau} = \omega$. For $\tau\omega \ll 1$ Eq. (4.8) yields

$$
\begin{aligned}
1 - (\tau\omega)^2/2 \simeq 1 - \frac{\alpha^2}{2} \\
\alpha \simeq \tau\omega,
\end{aligned}
$$

and the numerical solutions approaches the correct solution for $\tau \to 0$. Obviously, there is a tradeoff between the accuracy and the total 'real' time, that can be simulated by a fixed number of iterations.

The figures 4.1, 4.2 and 4.3 show the time dependence of $x(t)$ for different parameters $\tau\omega$. We see that the simulation is stable over many periods, if $\tau\omega \ll 1$, while it goes off course when $\tau\omega$ approaches 2. As anticipated, the trajectory diverges for $\tau\omega > 2$.

We estimate the number of periods until the discretization error becomes significant. The result of the Verlet algorithm oscillates with frequency α/τ. According to Eq. (4.10) the position at time $t = n\tau$ is given by $x^v(t) = x_0\cos(\alpha n)$. A typical time t^* after which the result goes off course is when $x^v(t^*) = -x(t^*)$, *i.e.:* when the phase error is π:

$$
\begin{aligned}
|\omega t^* - \alpha t^*/\tau| = \pi \\
t^* = \frac{\pi}{|\omega - \alpha/\tau|}.
\end{aligned}
$$

The number N^* of periods $T = 2\pi/\omega$ corresponding to t^* is:

$$
N^* = \frac{t^*}{T} = \frac{\omega\tau}{2\,|\omega\tau - \alpha|}.
$$

The leading order Taylor expansion of α yields

$$
|\omega\tau - \alpha| = \frac{(\omega\tau)^3}{24},
$$

end hence

$$
N^* = \frac{12}{(\omega\tau)^2}. \tag{4.11}
$$

Alternatively, if a certain number N^* of stable periods is required, the condition for the discretization reads

$$
\omega\tau = 2\sqrt{\frac{3}{N^*}}\,,
$$

or rather

$$
M^* = \frac{T}{\tau} = \pi\sqrt{\frac{N^*}{3}},
$$

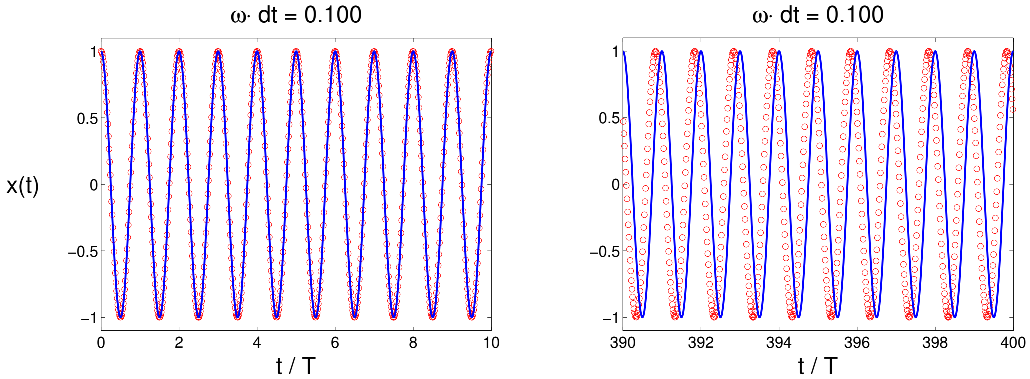

Figure 4.1: Simulation of the harmonic oscillator with $\omega\tau = 0.1$. According to Eq. (4.11) it takes $N^* = 1200$ cycles for the Verlet algorithm to go off course.

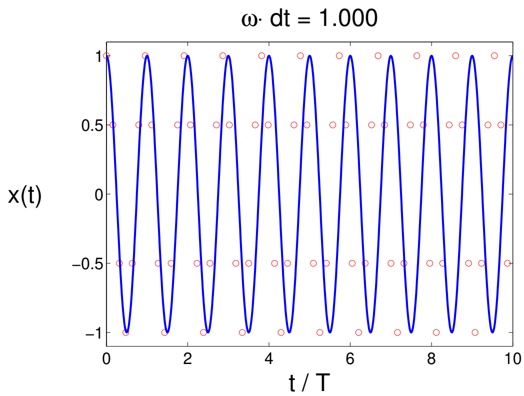

Figure 4.2: Simulation of the harmonic oscillator with the foolish choice $\omega\tau = 1$. The result of formula (4.11), $N^* = 12$, is corroborated by the simulation.

which gives the number of partitions of one period. E.g. for $N^* = 1000$ stable periods we need $M^* = 57$ time steps τ for each period.

An important issue for practical implementations is the efficient computation of the forces. Details are given in the textbook mentioned at the beginning of this chapter.

The time needed to reach equilibrium depends on the initial condition and on the details of the forces. To check whether equilibrium has been reached, it is advisable to monitor global physical properties such as kinetic energy, pressure, etc.

To perform simulations for a predefined temperature T rather than the total energy, the velocities of all particles are rescaled during the equilibration phase, *i.e.:*

$$\mathbf{v}_i(t) \to \lambda \mathbf{v}_i(t) \ , \forall i = 1, \ldots, N,$$

with

$$\lambda = \sqrt{\frac{(N-1)3/2k_B T}{1/2 \sum_i m \mathbf{v}_i^2}}.$$

The factor $(N-1)$ originates from the fact that the total momentum is conserved if no external forces are applied and therefore the number of *independent* velocities, entering the kinetic energy, are reduced by one for each spatial direction.

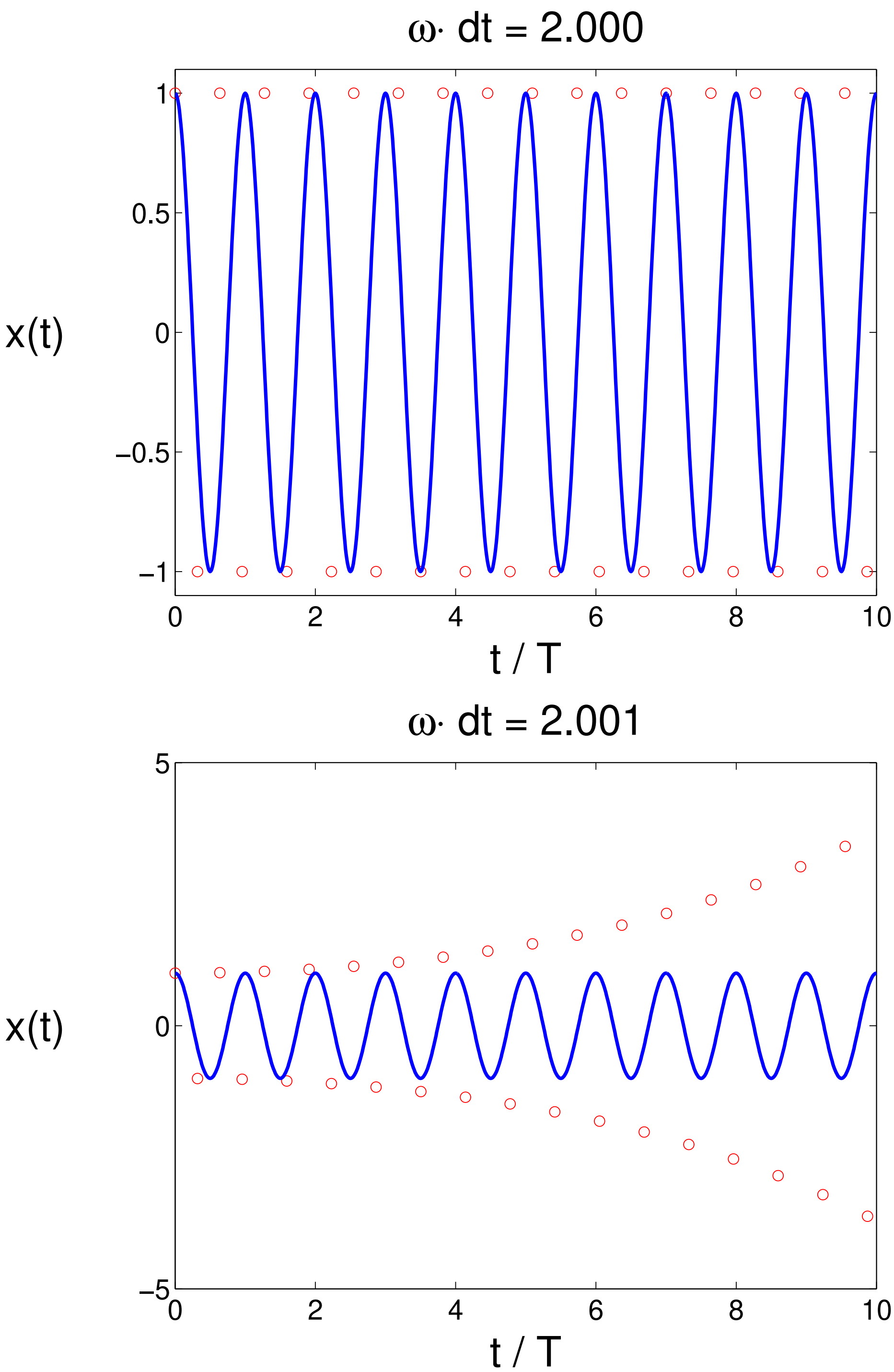

Figure 4.3: For illustration purposes the simulations for $\omega\tau = 2.$ and $\omega\tau = 2.001$ are also shown. Obviously, the phase becomes complex for $\omega\tau > 2$ leading to an exponential increase of the amplitude.