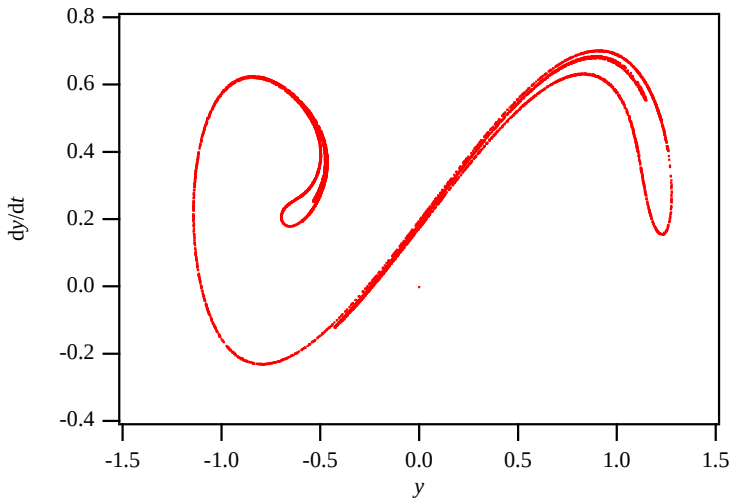
UMEÅ UNIVERSITY
Department of Physics
Tord Oscarsson
Claude Dion
Kjell Rönnmark

January 2007

Lecture notes on

COMPUTER SIMULATION TECHNIQUES



Contents

1 Introduction

During the last 50 years the computational capacity of computers has grown more or less exponentially. The original orbit calculations performed on the first computer, ENIAC, can be done some 6–7 orders of magnitude faster today. Furthermore, while ENIAC filled up the space of a room, todays computers easily fit on our desktop. In parallel with the development on the hardware side, scientific and technical applications have developed at the same rate in basically all fields. The fraction of scientific papers that rely on numerical computations increases steadily, and computer simulations are vital in todays industry. With little doubt, the need for numerical computations will continue to increase in the foreseeable future, as will the need for people trained in computational techniques.

To give a short, comprehensive course on computer simulation techniques seems impossible, considering wide range of applications available. Considering our respective scientific backgrounds in space plasma physics and atomic and molecular dynamics, it has been natural to focus on a particular type of simulations: dynamical simulations and many body systems. This confines the course to a manageable set of numerical techniques while covering such disperse fields as orange juice production (osmosis), drug design, gas and molecular dynamics, fabrication of IC-circuits, plasma dynamics and galaxy formation. It should be noted that the goal of the course is not only to teach the special numerical techniques that can be used on many body systems. By organizing the course in terms of length and time scales, we want to point out general problems hidden in the simulation of any dynamical system. As the course proceeds, the student is introduced to increasingly complex computer models, where each model is designed to handle different physical aspects of many body systems. It is our hope that by understanding the role played by the computer model that the students will be well prepared for their future role in science or industry.

2 Computer simulations

In order to define the concept "computer simulation", let us consider as a simple example the traffic flow in a city. People responsible for traffic planning are probably interested in questions like

- How is the traffic flow influenced by building new apartments in a certain area of the town?
- What happens to the traffic flow if an accident occurs at intersection A during rush hours?
- What happens when a street is closed for repairs?
- What happens if the right of precedence is changed, or traffic lights are put up at a certain intersection.

Undoubtedly, questions like these are of great importance for the planning of city traffic, and obviously real experiments would not be popular among the inhabitants. On the other hand, it is possible to make a computer model of the city traffic. This model includes the streets with all their intersections, precedence rules, circulation points etc., together with the cars, vans and trucks driving through the city. We can then perform our experiments on the computer model rather than in the town itself, that is, we can perform a computer simulation of the real system. This leads to the following definition of computer simulations: A computer simulation is an experiment performed on a computer model of a real system.[1]

The advantages of computer simulations are several. In the above example real experiments would not be convenient as they would affect people negatively. From an experimental point of view, computer simulations also have other advantages. Real experiments can be too expensive to perform, or simply impossible from a practical point of view. In such cases simulations may still be possible to use, and can give sufficient information to allow us to, at least partially, achieve what the experiment was aimed to do. Simulations can also be used in the planning stage in order optimize the scientific or financial outcome of an experiment. On the theoretical side, computer simulations are frequently used to handle complex non-linear systems and systems with a large number of degrees of freedom. That is, systems where traditional analytical methods fail to provide a comprehensive solution.

Due to the increasing power of computers, computer simulations have evolved as a new concept, a new tool by which we can attack scientific or engineering problems. The traditional two-way interaction between experiment and theory has been replaced by a three-way interaction involving experiment, theory and simulations. As illustrated in figure 1(a), computer simulations can be considered as a support for experiments and theory, in the sense discussed above. However, the student is urged to remember that computer simulations on their own have little value. As illustrated in figure 1(b), every simulation is based on an experimental and theoretical ground. Without the support of experiments providing the essential input, and a theoretical frame work to build the simulation on, computer simulations cannot be

[1] It should be emphasized that this definition is by no means unique, and that the student can expect to encounter modifications of this definition.

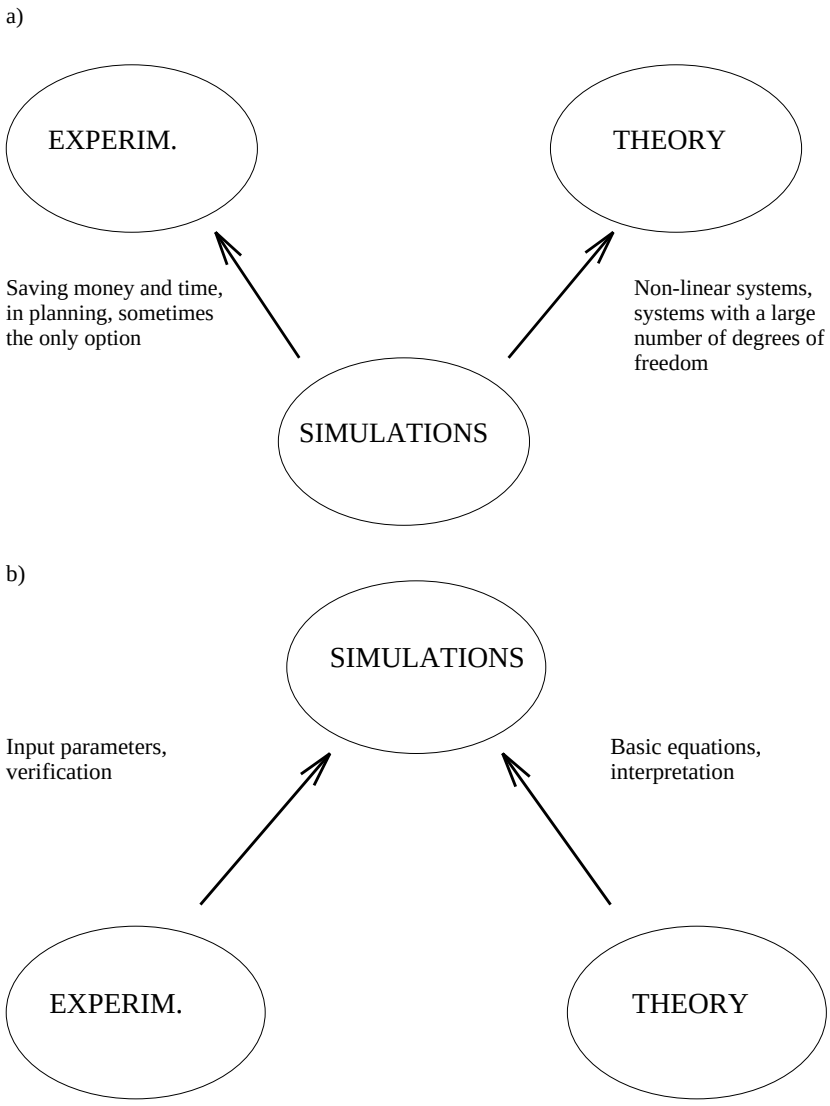


Figure 1: Schematic illustration of the interaction taking place between the three pillars of science and engineering — experiment, theory and computer simulation.

performed. Also, without the proper theoretical means for interpreting the output, and experiments to verify the results, the outcome of a computer simulation is nothing but a meaningless stream of numbers.

3 Numerical integration

To begin with, we are primarily interested in ordinary differential equations (ODEs), and here we discuss numerical methods for solving such equations. Methods for dealing with partial differential equations are discussed briefly in chapters 10 and 12.

3.1 Dynamical systems

The dynamical systems we now consider can be mathematically expressed as a system of N first-order ordinary differential equations,

$$\begin{cases} \dfrac{dx_1}{dt} &= f_1(x_1, x_2, \ldots, x_N), \\ \dfrac{dx_2}{dt} &= f_2(x_1, x_2, \ldots, x_N), \\ &\vdots \\ \dfrac{dx_N}{dt} &= f_N(x_1, x_2, \ldots, x_N), \end{cases} \tag{1}$$

evolving the state of the physical system forward in time, starting from an initial condition (usually taken at $t = 0$). This is referred to as an *initial value problem*. If time does not appear explicitly as a variable in the problem, as the one given above, the system of ODEs is said to be *autonomous*.

ODEs of higher order than 1 can always be reduced to a set of first-order equations. Take for instance Newton's equation of motion

$$m\ddot{x} = F. \tag{2}$$

By setting $x_1 = x$ and $x_2 = \dot{x}$, we get

$$\begin{cases} \dfrac{dx_1}{dt} &= x_2, \\ \dfrac{dx_2}{dt} &= \dfrac{F}{m}. \end{cases} \tag{3}$$

In a similar fashion, if the original equation is nonautonomous, it can be made autonomous by introducing an additional variable. For instance, if the force in Eq. (2) is given by $F = \sin(\omega t)$, the we can set $x_3 = \omega t$ such that the resulting system of equations is now

$$\begin{cases} \dfrac{dx_1}{dt} &= x_2, \\ \dfrac{dx_2}{dt} &= \dfrac{\sin(x_3)}{m}, \\ \dfrac{dx_3}{dt} &= \omega. \end{cases} \tag{4}$$

3.2 Finite difference equations

As discussed above, Newton's second law, in the one-dimensional case and for a single particle moving under the influence of a given force $F(x)$, can be rewritten as the system of coupled equations

$$\begin{aligned} \dot{x} &= v \\ m\dot{v} &= F(x). \end{aligned} \tag{5}$$

To solve these equations numerically we divide time into small segments Δt and replace all derivatives with finite difference equations, such that time is now a discrete, rather than a continuous, variable. For instance,

$$\dot{x} \approx \frac{x^{n+1} - x^n}{\Delta t}. \tag{6}$$

Here $x^n \equiv x(t^n)$, where the time t^n for the n:th time interval is given by

$$t^n \equiv n\Delta t. \tag{7}$$

When the time derivatives in (5) are replaced by finite differences we obtain two coupled finite difference equations. Using (6) the finite difference equations can be rewritten as

$$\begin{aligned} v^{n+1} &= v^n + \tfrac{1}{m}F(x)\Delta t \\ x^{n+1} &= x^n + v^n\Delta t. \end{aligned} \tag{8}$$

Together with the initial conditions $x(0) = x^0$ and $v(0) = v^0$, this gives a recursion relation for x and v.

3.3 Important qualities of numerical integration schemes

The integration method based on (6) is called Euler's method, and is one of the simplest numerical integration schemes. Choosing an alternative finite difference approximation to the derivatives leads to a different set of finite difference equations with other properties than the Euler scheme.

3.3.1 Consistency

The most basic requirement on a finite difference equation is that it should be consistent with the differential equation that it approximates. That is, the solution to the finite difference equation should approach the exact solution as the time step is made smaller and smaller. In the limit when $\Delta t \to 0$ the solutions should be identical. Another consistency requirement is that the finite difference equations should have the same symmetries as the original differential equation. Equation (5) is time reversible. That is, at any given time t the physical system described by (5) will return to its original position when either time evolves backwards, or when all velocities are reversed. The same should hold for the discretized set of equations.

3.3.2 Accuracy

The leading term in the difference between v^n and the approximation $(x^{n+1} - x^n)/\Delta t$ is proportional to $(\Delta t)^1$. This implies that the Euler scheme is of first order. A low-order method normally demands few operations per time step, but the time steps must be small in order to reduce discretization errors. Methods of higher order generally require more operations, but allow for a longer time step. Particle simulations are most often based on methods of order higher than one.

3.3.3 Stability

Suppose that

$$x^{n+1} = gx^n, \tag{9}$$

where the growth factor g can be complex. The magnitude of g determines how a small error in x^n is propagated to x^{n+1}. When $|g| > 1$ the error grows in each iteration, and the integration scheme is said to be unstable. On the other hand, when $|g| \leq 1$ the error decreases, or at least does not become larger, and the integration method is stable. The stability of an integration scheme depends both on the time step and on the type of problem being investigated. The Euler scheme is unconditionally unstable, that is, unstable independent of the time step, when applied to a harmonic oscillator.

3.3.4 Efficiency

The efficiency of an integration method is a measure of the number of operations required to integrate the equations of motion over a certain time. Higher order methods are used to obtain good accuracy, but due to the complex computations required, higher order methods can be rather inefficient. Simpler methods of lower order are not as accurate, but can be quite efficient for large problems. The choice of integration method is a compromise between efficiency and accuracy.

3.4 Methods for numerical integration

3.4.1 The leap-frog algorithm

The leap-frog algorithm is a second order integration method that is often used in particle simulations, especially in plasma simulations. Second-order accuracy is obtained by shifting the recursion relation for position and velocity half a time step

$$\begin{aligned} v^{n+1/2} &= v^{n-1/2} + a^n \Delta t \\ x^{n+1} &= x^n + v^{n+1/2} \Delta t. \end{aligned} \tag{10}$$

Here $a^n = (1/m)F(x^n)$. Due to the displacement, the method is sometimes also called the half-step method. Normally initial conditions are given in the form of $x(t = t^0) = x^0$, $v(t = t^0) = v^0$ for all particles in the system. The leap-frog algorithm can not be started directly

from these initial conditions. Often the algorithm includes as its first step half an Euler step back in time to obtain $v^{-1/2}$.

To investigate the stability of the leap-frog scheme we rewrite (10) as

$$x^{n+1} - 2x^n + x^{n-1} = \frac{1}{m}F(x^n)\Delta t^2 \tag{11}$$

and assume that $x^{n+1} = gx^n = g^2x^{n-1}$. When the leap-frog algorithm is applied to harmonic oscillator ($F(x) = -kx$) the growth factor is given by

$$g = 1 - \frac{1}{2}\omega^2\Delta t^2 \pm \omega\Delta t\sqrt{\frac{1}{4}\omega^2\Delta t^2 - 1} \tag{12}$$

where $\omega = \sqrt{k/m}$ is the natural frequency of the oscillator. When $\frac{1}{4}\omega^2\Delta t^2 - 1 \leq 0$ we have $|g|^2 = 1$ and the algorithm is stable. On the other hand, when $\frac{1}{4}\omega^2\Delta t^2 - 1 > 0$ the magnitude of g is greater than unity and the algorithm is unstable.

The leap-frog algorithm is reversible.

3.4.2 The Verlet algorithm

The velocity Verlet algorithm (Verlet, 1967) is a second order method where position and velocity are integrated on the same time steps

$$\left\{ \begin{array}{lcl} x^{n+1} & = & x^n + v^n\Delta t + \frac{1}{2}a^n\Delta t^2 \\ v^{n+1} & = & v^n + \frac{1}{2}(a^{n+1} + a^n)\Delta t. \end{array} \right. \tag{13}$$

Even though the recursion relations are not displaced relative to each other, the idea is the same as in the leap-frog algorithm. Positions and velocities are updated using an estimate of the velocity and acceleration at the center of the time interval,

$$\begin{array}{lcl} v^{n+1/2} & \approx & v^n + \frac{1}{2}a^n\Delta t, \\ a^{n+1/2} & \approx & \frac{1}{2}(a^{n+1} + a^n). \end{array} \tag{14}$$

Using the same type of stability analysis as for the leap-frog algorithm, it can be shown that the Verlet algorithm is stable for $\omega\Delta t \leq 2$ when applied to a harmonic oscillator. The Verlet algorithm is reversible.

Coding the velocity Verlet algorithm as it is written in Eq. (13) results in 10 floating point operations (FLOP),in addition to the calculation of a^{n+1} from x^{n+1}. This number can be greatly reduced by eliminating some multiplications by Δt. Let

$$\begin{array}{lcl} \tilde{v} & \equiv & v\Delta t, \\ \tilde{a} & \equiv & \frac{1}{2}a\Delta t^2, \end{array} \tag{15}$$

such that Eq. (13) becomes

$$\left\{ \begin{array}{lcl} x^{n+1} & = & x^n + \tilde{v}^n + \tilde{a}^n, \\ \tilde{v}^{n+1} & = & \tilde{v}^n + \tilde{a}^n + \tilde{a}^{n+1}. \end{array} \right. \tag{16}$$

Only 4 FLOP are now needed (with eventually 3 more, depending on how acceleration is calculated). The overhead is slightly increased as $\tilde{v}$ must be initially calculated and v recovered when needed (e.g., for plotting), but this is completely negligible if the number of integrations of the equations of motion is large.

3.4.3 Implicit methods and methods of higher order

The leap-frog and Verlet algorithms are both explicit in the sense that x^{n+1} and v^{n+1} can be expressed directly in terms of x^n and v^n. The Verlet algorithm uses the acceleration at the end of the time step to update the velocity, but the positions are updated before the velocities, and consequently, both x^{n+1} and a^{n+1} (again assuming the potential to depend on the position only) are known when the velocities are updated. The following algorithm is an example of an implicit algorithm,

$$\begin{aligned} x^{n+1} &= x^n + \tfrac{1}{2}(v^n + v^{n+1})\Delta t \\ v^{n+1} &= v^n + \tfrac{1}{2}(a^n + a^{n+1})\Delta t. \end{aligned} \tag{17}$$

For each time step we need to solve the system of equations (17) to obtain x^{n+1} and v^{n+1}. Since it is difficult to solve large systems of equations, implicit methods can be used only on fairly small systems or subsystems. The integration method based on (17) is just as the leap-frog and Verlet algorithm of second order. The great advantage of the implicit method lies in its stability. When applied to a harmonic oscillator the implicit scheme described here is unconditionally stable, and therefore, it is possible to use a larger time step. To allow for longer time steps while maintaining good accuracy one can use higher order methods such as Runge-Kutta or Bulirsch-Stoer. These methods can also give better stability conditions for certain types of non-linear forces. Higher order methods generally require more computations and more memory. Which method to prefer generally depends on the nature and the size of the problem.

4 Chaos

Most dynamical systems studied in physics courses show periodic motion (e.g., a mass suspended on a spring) or settle to a steady state, where motion stops (e.g., the LRC circuit). However, there exists a class of dynamical systems where motion never stops nor is periodic, but appears "unpredictable" or "random". These systems are said to be *chaotic* and, although some are complicated, their unpredictability is not due to their complexity. This can be seen by looking at the equations used to simulate them, which can be extremely simple. Chaotic systems tend to be *nonlinear*: a small perturbation (e.g., from the environment in an experimental apparatus, or due to numerical noise in the case of a computer simulation) is quickly amplified, which makes exact forecasting of their future behavior impossible.

Chaos is a broad subject, both from physical and mathematical viewpoints, and an entire course could be devoted to it. The point here is to give a quick introduction to the subject, as the numerical study of chaotic systems is quite interesting, and knowledge about chaos is useful even for the simulation of physical systems that are not chaotic.

4.1 Chaotic physical systems

A simple example of a chaotic system is a water faucet. When it is almost closed, water will drip in a regular fashion, with drops falling down at a fixed frequency. At the other extreme, when the faucet is opened wider, the water will flow in a continuous manner. In between the two, there will be an intermediate regime where chaos can be seen: it is impossible to tell exactly when the next drop will fall.

Turbulent fluid flow, for instance a waterfall, is inherently chaotic. Likewise, the mixing of two fluids, like stirring milk and coffee, can be a chaotic process. And you should already be familiar with the unpredictable behavior of the atmosphere, also known as *weather*.

Another physical system where chaos can be seen is the forced damped pendulum, shown in figure 2. This system can be modeled by a relatively simple differential equation for θ, the angle with respect to the vertical,

$$\frac{d^2\theta}{dt^2} + \nu\frac{d\theta}{dt} + \sin\theta = f\sin\omega t. \tag{18}$$

The second term on the left-hand-side accounts for the friction at the pivot point, and is parametrized by a damping coefficient ν, while the third term is the gravitational potential. On the right-hand-side appears the driving force of amplitude f and angular frequency ω. As for the faucet, in certain conditions the pendulum will oscillate in a periodic fashion, as it would in the absence of friction. But for other combinations of the parameters, its oscillations appear to be random (figure 3).

Yet another apparatus that shows chaotic behavior is illustrated in figure 4. It consists in a steel beam hanging between two magnets. At rest, the beam can be in the middle of the two magnets, or bent towards one of them. When the entire setup is shaken, the beam will oscillate, and in certain conditions this will be chaotic. Here again, the mathematical model for the apparatus is quite simple, as the differential equation for the position y of the tip of

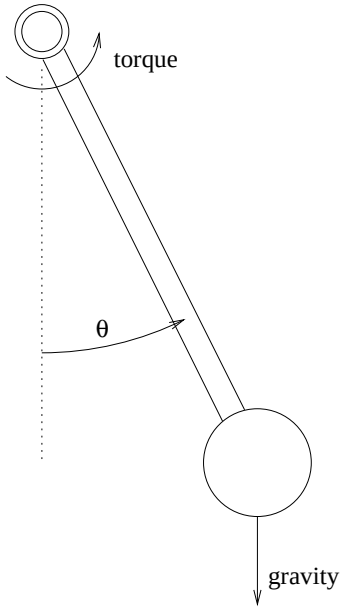


Figure 2: The forced damped pendulum.

the beam is [2]

$$\frac{d^2y}{dt^2} + \nu\frac{dy}{dt} + \Big(y^3 - y\Big) = f\cos\omega t, \tag{19}$$

with the parameters having the same meaning as those of equation (18). The force term appearing on the left-hand-side corresponds to a double-well potential (figure 5), and as such the equation could be also used to describe a ball trapped in such a potential, with periodic shaking.

4.2 Chaos and numerical simulations

4.2.1 Sensitivity to initial conditions

The last two physical systems discussed in the previous section also show a chaotic character even in cases where the motion settles in a periodic oscillation, as they exhibit a sensitivity to the initial conditions. For instance, for some values of ν, f, and ω, the steel beam will, after some transient period, end up oscillating around an equilibrium position close to the left or the right magnet (corresponding to one of the two wells of figure 5). Then, if the beam is initially slightly bent ($y(0) \neq 0$), two very close initial values of this bending can lead to an oscillation on the right in one case, and an oscillation on the left in the other case! So, even though the motion becomes strictly predictable after a certain time, it is impossible to predict beforehand where the beam will end up (unless the position of the beam could be measured with an infinite precision).

This can be seen by looking at the trajectory in phase space, that is by following the position of a system (described by a variable x) on a graph of dx/dt as a function of x, starting from

[2] The general equation $\frac{d^2x}{dt^2} + \nu\frac{dx}{dt} + \alpha x^3 + \beta x = F(t)$ is known as the *forced Duffing equation.*

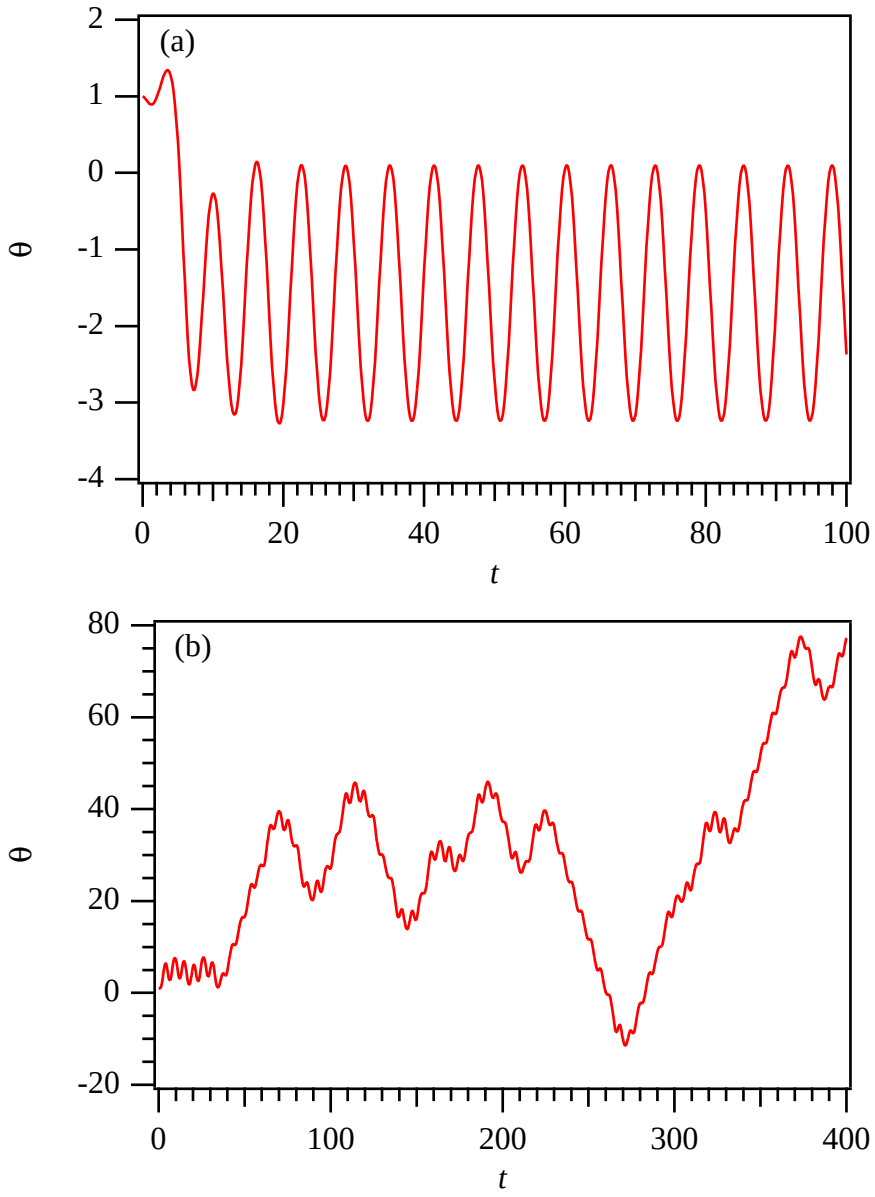


Figure 3: Time evolution of the forced damped pendulum. (a) $\nu = 0.5$, $f = 1$, $\omega = 1$, resulting in a period oscillation, (b) $\nu = 0.22$, $f = 2.7$, $\omega = 1$, showing chaotic behavior. For both, the initial conditions are $\theta(0) = 1$, $\dot{\theta} = 0$.

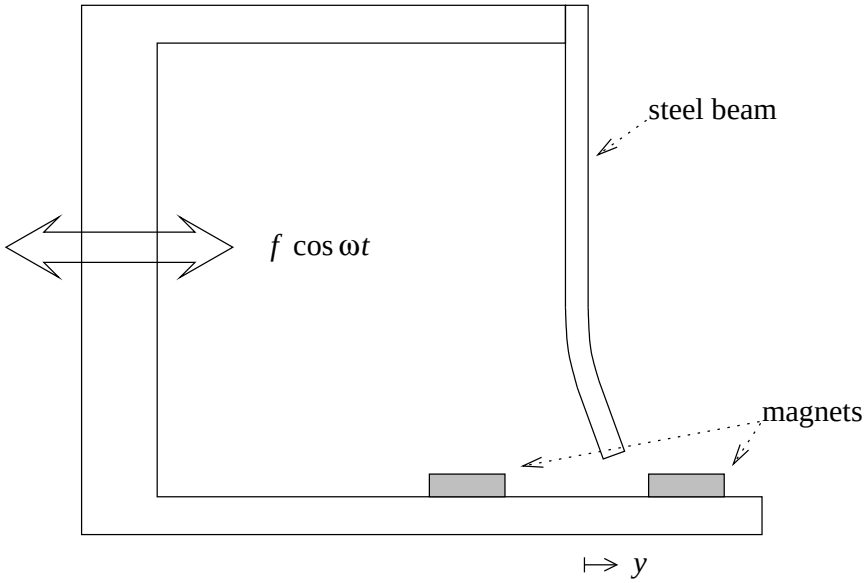


Figure 4: The forced, damped oscillating beam.

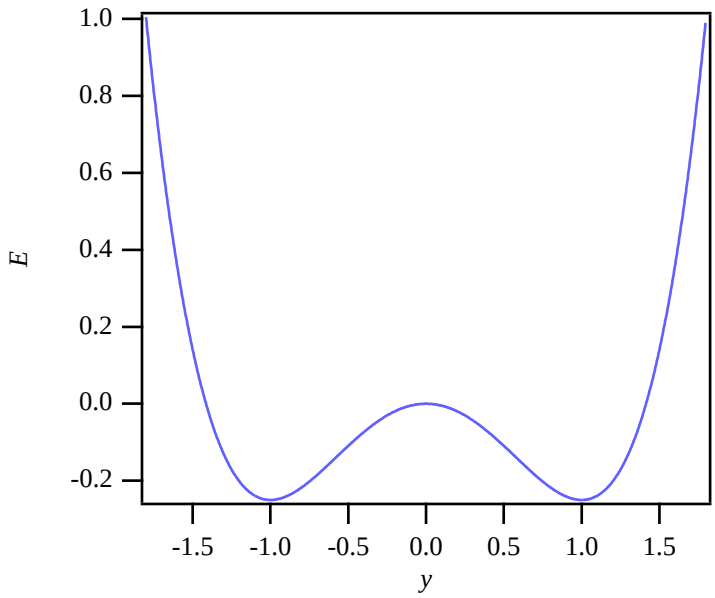


Figure 5: Double-well potential of the oscillating beam [see equation (19)].

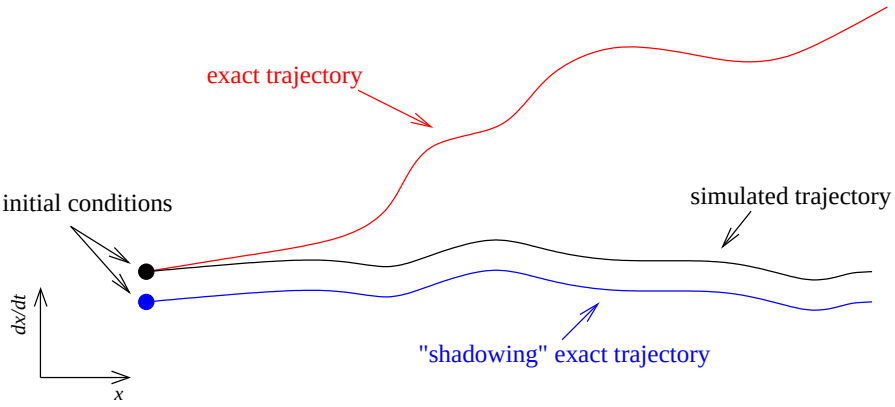


Figure 6: The simulated trajectory departs from the exact trajectory starting from the same initial condition, but stays close to another exact trajectory that "shadows" it.

the initial conditions. For non-chaotic systems, two initial points close together will lead to trajectories which are not far apart. They will eventually diverge because of inevitable numerical errors, but these can be reduced by using a smaller value of the time step of integration. In the presence of chaos, the trajectories will diverge *exponentially* fast.

4.2.2 Simulating chaotic systems

You may be asking yourself the question: *Doesn't this invalidate all simulations of nonlinear and chaotic systems?* The answer is, of course, no! It does indeed limit in some cases what can be a reasonable length for a simulation. One good example is weather. Even with the best of models, there will always be a limit due to the amount and precision of the data used to start the simulation[3]. As such, weather predictions will probably always be limited to a few days. The situation is different for *statistical* values obtained from a simulation, as will be the case in chapter 6 for molecular dynamics. With statistics, one is not interested in a particular system, but in the global behavior of a certain type of system, so deviations from an exact scenario are not a problem. It is for similar reasons that the evolution of atmospheric temperatures can be simulated for decades to come, even though we're not sure if it will be sunny next weekend.

This still leaves the question of the simulation of chaos itself. Because of discrete representation of numbers, it is certain that, starting from a precise initial condition, the numerical simulation will quickly move away from the "exact" solution, i.e., the one we would obtain if we could use an infinity of digits to represent numbers in the computer. Fortunately, there is a property called *shadowing* (mathematically proven for some systems), which states that the simulated phase space trajectory always stays close to an exact trajectory that started from slightly different initial conditions (see figure 6).

[3]The canonical, and often misused, example is the *butterfly effect*, which says that neglecting the effect of one butterfly flapping its wings can alter the simulated course of a hurricane.

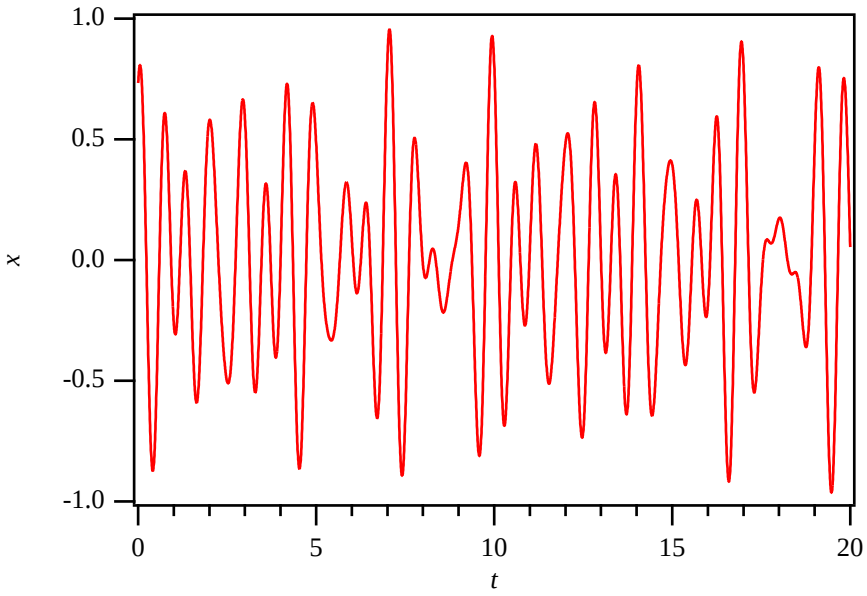


Figure 7: Quasiperiodic oscillation [see equation (20)].

4.3 Spotting chaos

When does chaos appear? There are very few *a priori* ways of knowing if a system will behave in a chaotic fashion. Moreover, for many systems, chaos only occurs for some values of the parameters characterizing it or for some particular initial conditions.

Nevertheless, there exists some minimum conditions for the appearance of chaos. For a system of first-order autonomous ODEs, the minimum number of equations (or variables) is 3. This does not mean that a system will $N \geq 3$ will necessarily be chaotic, but rather that chaos can be ruled out in systems of dimension 1 or 2. Another sign of possible chaos is when an equation is *nonlinear*, i.e., if the variable of motion appears raised to a power different than 1 or in a function (trigonometric, exponential, etc.).

It is usually much easier to detect chaos *a posteriori*, by looking at the result of the experiment or of the numerical solution. But one must be careful not to judge only by the looks of the evolution of the system. Consider for instance figure 7, which looks very irregular and appears to be random. This is not the case: the result is strictly predictable, as the generating function is

$$x = \frac{1}{3}\cos(2\pi t) + \frac{2}{5}\cos\left(2\sqrt{2}\pi t\right) + \frac{1}{4}\sin\left(2\sqrt{3}\pi t\right). \tag{20}$$

This is an example of a *quasiperiodic* system, which can be described by a few fundamental frequencies, but in ratios ω_i/ω_j that are not rational numbers.

4.3.1 Frequency spectrum

One way to be sure that a function $f(t)$ is chaotic (or random) is to look at its frequency spectrum, as obtained by a Fourier transform. A periodic or quasiperiodic signal presents only

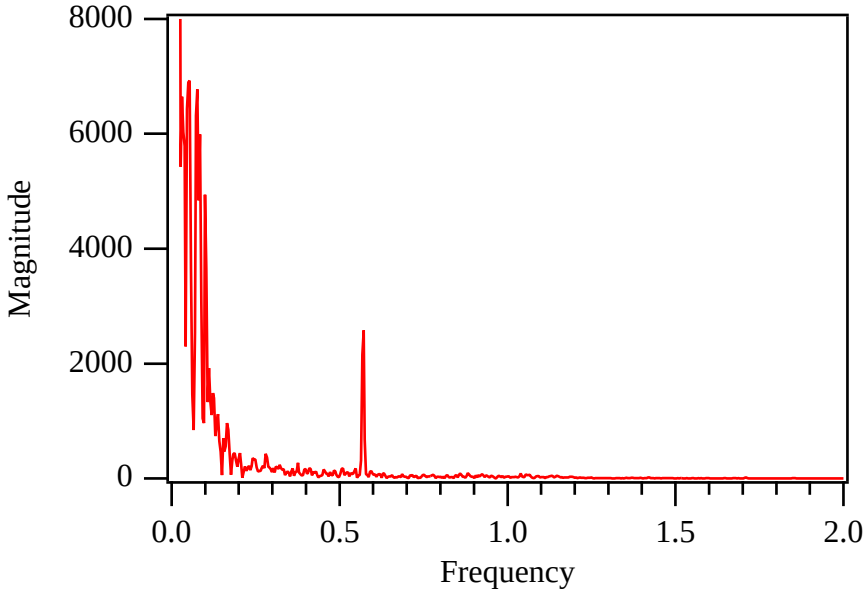


Figure 8: Fourier transform of the the function $\theta(t)$ shown in figure 3(b).

a few fundamental frequencies, whereas chaos results in a very broad frequency spectrum. An example is given in figure 8, which shows the result for the forced damped pendulum in the chaotic regime. By comparison, the result for the pendulum in periodic oscillations would show only a peak at a frequency of 1 (along with a component of zero frequency to account for the fact that the oscillation is not around $\theta = 0$).

4.3.2 Autocorrelation function

Another useful measure of the predictability of the dynamics is the autocorrelation function, defined as

$$A(\tau) = \frac{1}{T}\int_{-T}^{T} f(t)f(t+\tau)dt, \tag{21}$$

where T must be chosen large with respect to the dominant periods of the motion. If the dynamics are periodic, then so will $A(\tau)$. In the presence of chaos (or randomness), $A(\tau) \to 0$ when τ gets bigger than a certain characteristic time τ_c.

4.3.3 Poincaré map

Chaos can also be observed by looking at the trajectory in phase space of a dynamical system. One particularly interesting method consists in looking at the trajectory only at periodic intervals in time. More precisely, for a system driven at an angular frequency of ω, taking snapshots when $t \bmod 2\pi/\omega = 0$ leads to what is called a *Poincaré map*. In the absence of chaos, this map will show only a few distinct points, or closed loops. When the dynamics are chaotic, the Poincaré map will be *fractal*, which means it will have a structure that is similar at whatever scale you are looking. An example of this is given in figure 9.

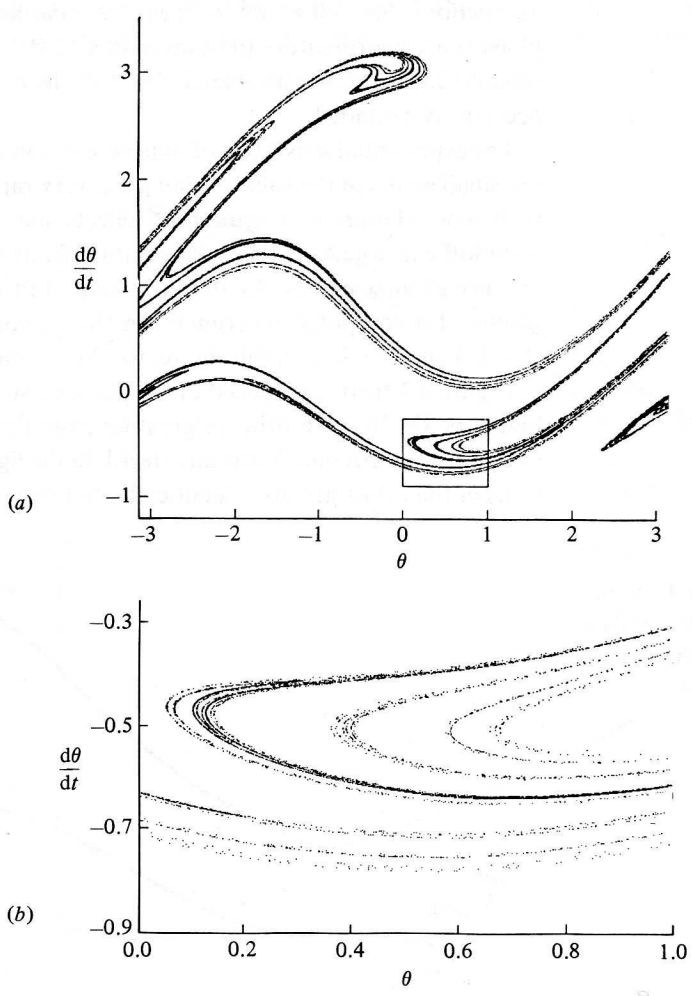


Figure 9: Poincaré map of the forced damped pendulum equation in the surface of section $\omega t \bmod 2\pi = 0$. [From Ott, 1993.]

4.4 Exercise

1. Chaotic dynamics: The oscillating beam.

5 Many body systems

The physical system that we are now interested in is a system consisting of N particles, whose coordinates are denoted by $\mathbf{x}_i,\ i = 1, \ldots, N$. We concentrate on systems where the particles interact in pairs via a potential $V(r)$ that depends only on the distance r between the two interacting particles. In such a system we can trace the individual particles by solving the equations of motion

$$m\ddot{\mathbf{x}}_i = \mathbf{F}_i \qquad 1 \leq i \leq N, \tag{22}$$

where the force on the i:th particle is given by the sum over the interaction with all other particles in the system

$$\mathbf{F}_i = \sum_{j \neq i} -\nabla V(\mathbf{x}_j - \mathbf{x}_i). \tag{23}$$

Newton's equations of motion have been studied for a long time, and have been reformulated by, e.g., Lagrange and Hamilton. Still, for large N, we can still not solve equation (22) analytically. Even the three-body problem poses difficulties. Since real physical systems often consist of a very large number of particles, it is impossible in practice to treat these systems with analytical methods only.

With the aid of computers we can study a many body system by integrating the equations of motion numerically. But also computers have limitations in the form of limited memory and limited computational capacity. In a simulation of a many body system, the position and velocity of each particle must be stored. This means that the memory required is at least (N particles)$\times$(2×3 coordinates/particle)$\times$(4 bytes/coordinate) = 24 N bytes. Today, internal memories range up to 10^9 bytes, and consequently, it is difficult to simulate systems with more than 10^7 particles. The computational capacity of todays computers lies somewhere around 10^8 – 10^9 floating point operations per second (FLOPS)[4]. Computing the force on each particle explicitly using (23) requires of the order of $10N^2$ floating point operations. Consider a simulation of 10^7 particles running over 1000 time steps. At the very best, such a simulation would take (1000 time steps)$\times$($10(10^7)^2$ flop/time step)/(10^9 flop/second) = 10^9 seconds, or 32 years.

Consequently, the available memory and the computational capacity put an upper limit on the size of the many body system that can be addressed directly with computers. In many interesting physical systems the number of particles greatly exceeds this limit. Even if computers continue to develop very quickly, we will never be able to study these physical systems in detail. Instead we must develop and work with models of the real many body system. The model must include only the limited number of particles that computers can manage, and still be able to represent an essential part of the physics in the real system. Alternatively, one can try and find completely different models, where there is no need to simulate individual particles, as is done, e.g., in fluid dynamics.

The following lectures will give you an introduction to various computational models and methods for studying many body systems. In the particle-particle (PP) method the forces are computed directly, as in (23). As already indicated, this method is not very effective.

[4]The size of the internal memory and the computational speed of the largest/fastest computer increases rapidly. Therefore, the figures given here are probably inaccurate, but this does not change the conclusions drawn in the text.

In fact, the particle-particle method is seldom used in practice, at least for large systems. However, it is needed here to give an introduction to more sophisticated methods. For short-ranged interactions it is possible to devise efficient algorithms by keeping track of all particles and their nearest neighbors. Plasma simulations are often based on the so-called particle-mesh (PM) method, which employs a spatial grid (or mesh), and compute forces by solving field equations on the grid. The particle-mesh method is most efficient when the potential is long-ranged and varies slowly in space. In addition to these methods, we will discuss computer simulation of Vlasov fluids, algorithms for solving classical fluid equations, and see how these equations can be combined with particle simulations in so-called hybrid methods.

6 Molecular dynamics

In small systems, such as a planetary system, which are closed systems consisting of relatively few particles, it is possible in practice to follow each particle in the system, and we can use a computer model that lies very close to the real system. Unfortunately, from a computational point of view, most natural systems are quite different from the planetary system. Gases, fluids and solid materials consist of a very large number of particles. Even a macroscopically small volume of gas contains some 10^{23} molecules. Obviously, it is impossible to follow the motion of every single particle in such a system. Instead, we must develop and work with a computer model that is further away from the real system, and which contains only a limited amount of information.

Molecular dynamics investigates fluids and gases using a computer model including only relatively few particles, representing only a small part of the real system. One can imagine this part being embedded in a larger volume, and the idea behind molecular dynamics is that the results obtained in this reduced model are valid for the whole system. Alternatively, computer models of varying size are used to extrapolate towards the thermodynamic limit $N \to \infty$.

Among the fluid properties suitable for molecular dynamics investigations are relations between thermodynamic quantities such as temperature, pressure, internal energy, etc. These quantities characterize systems in equilibrium, and here we are primarily interested in such systems. In statistical physics, thermodynamic quantities are formed via ensemble averages. In a computer simulation we do not have access to a complete ensemble, even though each simulation is performed within a specific ensemble. Instead, as discussed below, ergodicity[5] is assumed and time averages are used. The deterministic methods described here are mainly used for studying the microcanonical ensemble and the isobar-isenthalpic ensemble, and here we limit ourselves to the former. Other ensembles can be investigated via, e.g., Langevin, Monte Carlo or hybrid Monte Carlo methods.

The techniques discussed below are commonly used in molecular dynamics. They are based on computing forces directly from the interactions between all particles, but various methods are used to speed up the computations. Hockney and Eastwood describe such techniques as particle-particle (PP) methods.

6.1 The computer model

The computer model that we use to simulate gases consists of a model for the particle interactions, a simulation box containing a limited number of particles, and boundary conditions.

6.1.1 The Lennard-Jones potential

The interaction between neutral atoms in a gas can, at least in principle, be determined via quantum mechanics. However, in general this is very difficult, and it is often convenient to

[5] A system in which ensemble averages and time averages are equivalent is said to be ergodic. It is assumed that this is the case for thermodynamic systems.

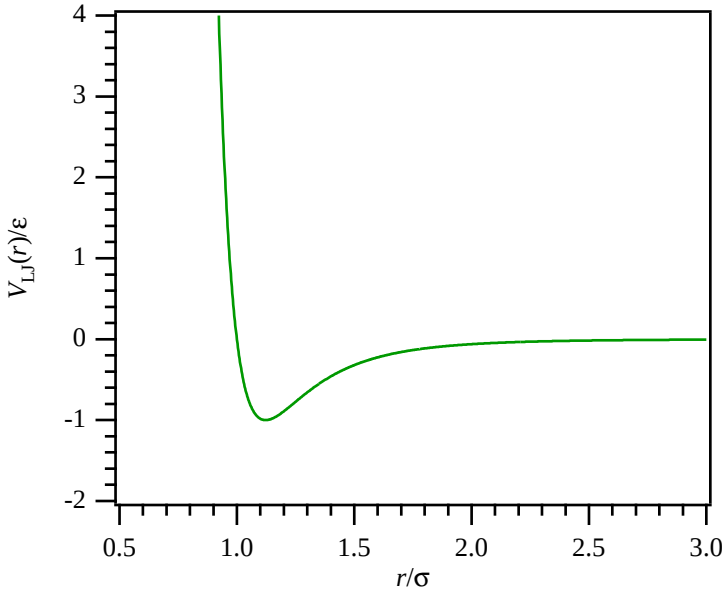


Figure 10: The Lennard-Jones potential.

use a phenomenological potential, e.g., the Lennard-Jones potential (see figure 10)

$$V_{\mathrm{LJ}}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right]. \tag{24}$$

This potential models the interaction between two atoms, and depends only on the distance between the two particles. The strength of the interaction is parametrized by an energy scale ε and a length scale σ. The value of these parameters can be determined from quantum mechanical calculations or derived empirically from experimental spectra.

When two atoms come too close to each other, their electron clouds start to overlap, and to avoid breaking the exclusion principle some of the electrons are forced to accelerate. This acceleration of the electrons causes a strong repulsive force modeled by the first term in the Lennard-Jones potential. At larger distances there is a weak attraction, the van der Waals force, which arises due to the polarization of each atom. The van der Waals force is modeled by the second term in $V_{\mathrm{LJ}}(r)$.

Notice that the potential is short-ranged in the sense that $V_{\mathrm{LJ}}(r) \approx 0$ for $r > 3\sigma$. As discussed below, to save computations the Lennard-Jones potential is often truncated at $r = r_{\mathrm{c}} \approx 3\sigma$. When the potential is truncated the force varies discontinuously at $r = r_{\mathrm{c}}$, and this will cause the total energy to fluctuate. Normally, these fluctuations are small, and will average out when computing thermodynamic quantities. However, when debugging and testing the program, the total energy is an important test parameter, and testing can be made easier by using a potential that will conserve the total energy. One way of obtaining such a potential is to truncate and shift the Lennard-Jones potential according to

$$V_{\mathrm{s}}(r) = \begin{cases} V_{\mathrm{LJ}}(r) - V_{\mathrm{LJ}}(r_{\mathrm{c}}) - (r - r_{\mathrm{c}}) \left(dV_{\mathrm{LJ}}/dr \right)_{r_{\mathrm{c}}} & r \leq r_{\mathrm{c}}, \\ 0 & r > r_{\mathrm{c}}. \end{cases} \tag{25}$$

For the shifted potential both the potential itself and the force are continuous at $r = r_c$.

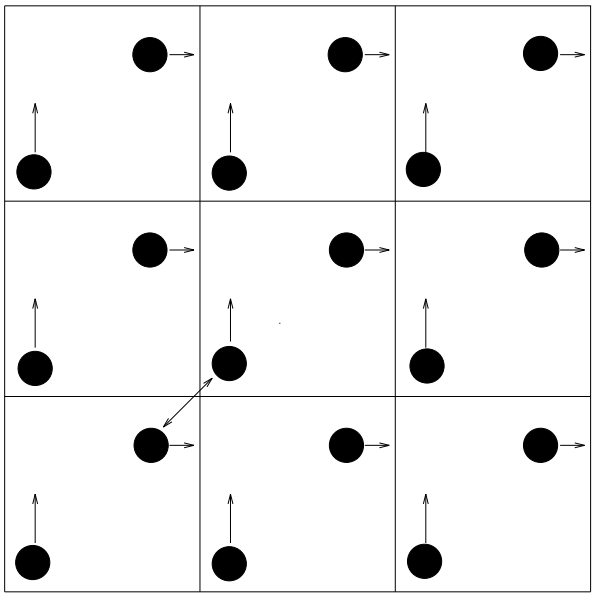


Figure 11: Illustration of periodic boundary conditions and the nearest image principle [after Gould and Tobochnik, 1988].

Some people argue that the shifted potential should be used for test purposes only, while others claim that the shifted potential should always be used and that analytical corrections for the shift in the potential can be applied after the run.

6.1.2 The simulation box and periodic boundary conditions

To simulate a gas with a given density n we can place N particles in a suitable volume V such that $n = N/V$. Once the interaction between the particles in the system has been specified, we can study the dynamics of the system by integrating the equations of motion. However, a problem arises when one of the particles hits the wall of the simulation box. If we allow the particle to leave the box, this will change the density of the gas. One can imagine letting the particle bounce off the wall back into the simulation box. However, this is not an acceptable solution. The change in particle momentum caused by the wall will cause unphysical fluctuations in the simulated gas. The fraction of gas influenced by the wall is roughly proportional to the ratio between the total wall area and the total volume. Consider a cubic simulation box of volume $V = L^3$, where L is the side of the box. For such a box, the fraction p of gas influenced by the walls is proportional to $L^2/L^3 = 1/L$. Since $N = nL^3$ we find that $p \propto N^{-1/3}$. For $N = 10^{23}$ we have $p \approx 0$ and the fluctuations in a real gas are thereby negligible. For $N = 10^7$, which is what we might achieve in a computer simulation, p is of the order of 1 %. Consequently, the effects of the walls, together with a limited number of particles in the simulation, will exaggerate fluctuations in the gas

One way of avoiding wall effects is to use periodic boundary conditions. The idea behind periodic boundary conditions is illustrated in figure 11. One of the squares in figure 11 corresponds to the simulation box. Periodic boundary conditions means that we consider a system with an infinite number of identical subsystems. When a particle leaves the simulation

box at one side, an identical particle enters the box at the opposite side. The incoming particle has exactly the same velocity as the one leaving the box.

By using periodic boundary conditions the total number of particles is conserved and wall effects disappear. At the same time the system becomes periodic with a period L equal to the dimension of the simulation box. Naturally, real gases do not possess this periodicity. However, the imposed periodicity should only affect the ability of our model to describe physical phenomena occurring at scale lengths of the order of L or larger. How well physics at shorter scale lengths is represented by the computer model depends mostly on the number of particles. Often, boxes with varying size are used to trace the effect of periodic boundary conditions.

6.1.3 The nearest image principle

Due to the periodic boundary conditions, our complete model includes identical particles, images, at $\mathbf{x}+\mathbf{m}L$, where $\mathbf{m}$ is a vector whose components take on all integer values between $-\infty$ and $+\infty$. Thus, when computing the force on a given particle, we must include the interaction with the particles in the simulation box and all their images. However, for short ranged potentials one often assumes the nearest image principle, which means that only the interaction with the nearest image is included when computing forces. That is, the distance r_{ij} between two particles at positions $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed as

$$r_{ij} = \min |\mathbf{x}_i - \mathbf{x}_j + \mathbf{m}L| \tag{26}$$

taken over all $\mathbf{m}$.

6.1.4 Choosing the time step

As discussed in chapter 4, the accuracy of finite difference schemes depends crucially on the size of the time step. If the time step is too long, the computer model may be unable to represent the physics we want to investigate.

In three dimensions the typical distance between two particles is given by

$$d = L/N^{1/3}. \tag{27}$$

To guarantee stability in a molecular dynamics simulation, the time step must be chosen such that

$$v_{\max}\Delta t \ll d, \tag{28}$$

where $v_{\max}$ is the maximum velocity in the particle system. When the time step is too big, two particles will sooner or later experience an unphysical close encounter since the particles do not feel the presence of each other until after a time Δt. This unphysical close encounter will result in a dramatic increase of the total energy of the system.

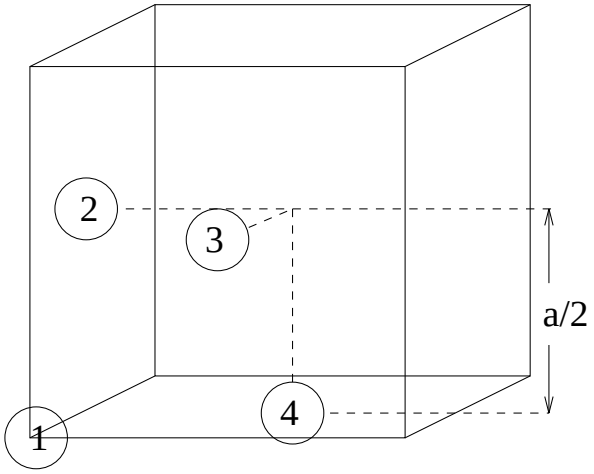


Figure 12: A face centered cubic lattice is often used when initializing Lennard-Jones fluids. The side length of the lattice cube is a and 4 particles are placed at $(0, 0, 0)$, $(0, a/2, a/2)$, $(a/2, 0, a/2)$ and $(a/2, a/2, 0)$ [after Haile, 1992].

6.2 Computer simulation of a system in equilibrium

Computer simulations aimed at investigating equilibrium quantities of gases is normally divided into three different stages:

1. Initialization
2. Approaching equilibrium
3. Production

6.2.1 Initialization

The initialization step consists of choosing coordinates and velocities for all particles. The particles should be distributed evenly in space. Random positions can be used for dilute gases, but there is always a risk that two particles are initiated very close together. This means that the total energy of the system initially can be much higher than desired, and that the initial system can be far from equilibrium (see below). Close encounters can also give rise to numerical problems. For Lennard-Jones fluids the particles are often initialized in a so-called face centered cubic (fcc) lattice (see figure 12). The simulation box is divided into a set of smaller cubes, and the particles are placed at the center of the faces of these cubes. The fcc lattice is used for Lennard-Jones fluids since, e.g., argon crystallizes into this structure. One unit cell in the fcc lattice will contain 4 particles, and therefore, a cubic simulation box will contain $4i^3$ particles, where i is an integer. Hence, one can often see Lennard-Jones simulations having 32, 108, or 256 particles.

For isolated systems the total linear momentum $\mathbf{p}$ should be conserved, and it is convenient to choose the initial velocities such that $\mathbf{p} = 0$. This is achieved by subtracting the center of

mass velocity from the initial velocities. Often the simulation is to be performed on a system at a given temperature T_1. In such cases the initial velocities can be chosen such that the initial total kinetic energy equals $\frac{3}{2}Nk_{\mathrm{B}}T_1$, (see 6.2.3) where k_{B} is the Boltzmann constant. This is achieved by, e.g., first sampling the velocities at random and then computing the corresponding temperature T_0, which most likely is not equal to the desired temperature T_1. A set of velocities corresponding to the temperature T_1 is then obtained by scaling the initial velocities according to $v \to v\sqrt{T_1/T_0}$.

6.2.2 Approaching equilibrium

An equilibrium is characterized by the fact that the thermodynamic properties of the system are constant. There are no trends in temperature, pressure, internal energy etc. This means that, while the instantaneous value of the kinetic energy will fluctuate, as will the potential energy, their average values should remain roughly constant. This simple requirement is a necessary condition for equilibrium, but it is far from sufficient. In fact, there is no way of determining exactly when an equilibrium has been attained in the simulation. That is, we have no useful sufficient condition at our disposal. Instead, one is forced to impose a set of necessary conditions, and assume that these conditions together are sufficient.

When initializing the particles in an fcc lattice as described above, one has to make sure that nothing of the lattice structure remains before starting to compute averages. One way is to monitor the order parameter $\lambda = (\lambda_x + \lambda_y + \lambda_z)/3$ introduced by Verlet, where

$$\lambda_x = \frac{1}{N}\sum_{i=1}^{N} \cos\left(\frac{4\pi x_i}{a}\right). \tag{29}$$

When using an fcc lattice, $\lambda = 1$ initially. As the system approaches equilibrium λ starts fluctuating around its equilibrium value $\lambda = 0$.

Another way of monitoring the approach towards equilibrium is to compute the particle distribution function $f(\mathbf{v})$, which is defined such that $f(\mathbf{v})d\mathbf{v}$ is the number of particles with velocities in the range $[\mathbf{v}, \mathbf{v} + d\mathbf{v}]$. From statistical physics we know that the particle distribution function is Gaussian when the system is in equilibrium. Instead of monitoring the full distribution function, one can also compute the Boltzmann H-function

$$H = \int d\mathbf{v}\, f(\mathbf{v}) \ln f(\mathbf{v}). \tag{30}$$

By inserting a Gaussian distribution function we can compute the equilibrium value of H from (30) for a given temperature. The value obtained in the simulation should approach this equilibrium value as the system approaches equilibrium.

Finding the equilibrium may seem trivial at first, but is a difficult problem. Irrespective of how the initial conditions are chosen, it is very likely that the system is not in equilibrium at the beginning of the run. Consequently, the equations of motion must be integrated over a sufficiently long time to let the system approach equilibrium. This time may be quite long, and in the worst case, a substantial amount of computer time is spent on just looking for the equilibrium. It is not straightforward to speed up the approach towards equilibrium. There is always a risk for getting trapped in metastable configurations, from which the approach

towards equilibrium may be asymptotically slow. To escape from a metastable system one can temporarily raise the total energy of the system, e.g., by scaling the velocities. One of the most useful ways of finding an equilibrium configuration is to start from an equilibrium configuration obtained in a previous run. For instance, by scaling the velocities the system can be brought to a new equilibrium at a different temperature. Often, repeated scaling is required for the operation to be successful, but the procedure can still be the most efficient way of finding the new equilibrium.

6.2.3 Production

To compute thermodynamic quantities for a system in equilibrium, ergodicity is assumed and ensemble averages are replaced by time averages. The temperature T is computed by using the equipartition theorem

$$\frac{D}{2}Nk_{\mathrm{B}}T = \left\langle \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 \right\rangle, \tag{31}$$

where $\langle\cdot\rangle$ represents time averaging, and D is the number of degrees of freedom per particle. Since velocities are readily available in the simulation, computing the temperature is straightforward.

The pressure can be computed via the virial theorem

$$PV = Nk_{\mathrm{B}}T + \frac{1}{D}\left\langle \sum_{i=1}^{N} \mathbf{x}_i \cdot \mathbf{F}_i \right\rangle \tag{32}$$

(e.g., Chandler, pp. 204–205). In our periodic computer model it is convenient to rewrite the virial in terms of the relative position of the particles according to

$$\sum_{i=1}^{N} \mathbf{x}_i \cdot \mathbf{F}_i = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij}, \tag{33}$$

where $\mathbf{F}_{ij}$ is the force acting between particles i and j.

Other thermodynamic quantities can be computed in a similar way, but here we focus on temperature and pressure.

6.3 The Box method

The Lennard-Jones potential has a range of roughly $r_{\mathrm{c}} = 3\sigma$. In practice, this means that two particles separated by a distance $r_{ij} > r_{\mathrm{c}}$ do not interact. The computer simulation can be made much more efficient by avoiding force computations for such particle pairs. As noted in the introduction, summing all interactions directly costs of the order of $10N^2$ floating point operations. By computing r_{ij} for all particle pairs and then explicitly testing if $r_{ij} > r_{\mathrm{c}}$ is much cheaper, but the box method is even more efficient.

The box method applied in a two dimensional case is illustrated in figure 13. The simulation box is divided into a number of sub-boxes or cells. The dimension of each cell equals (is is

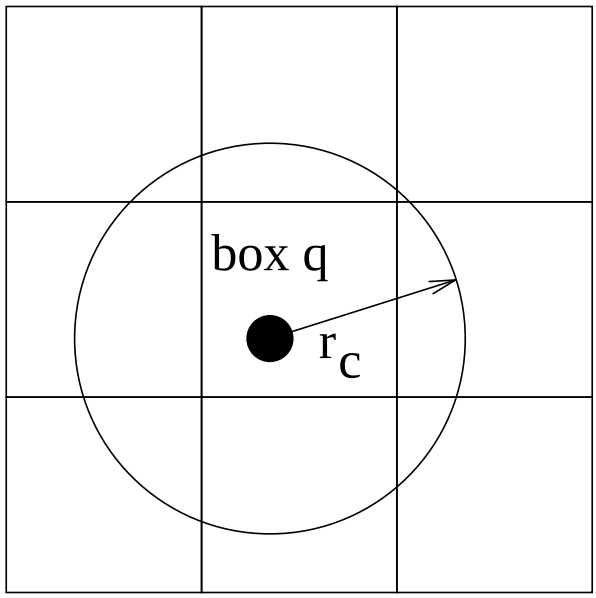


Figure 13: The box method [after Hockney and Eastwood, 1988].

greater than) the cut-off radius r_c. This means that particles in any given cell only interact with particles in the same cell or the 8 (26 in three dimensions) neighboring cells. The idea behind the box method is to maintain a list of the particles in each cell, and to compute distances and forces only between particles for which $r_{ij} < r_c$. The point is that maintaining the list is relatively cheap, and we avoid computing the distances between particles which do not interact. The box method is roughly a factor N_B faster than a direct summing of all the forces, where N_B is the number of sub-boxes.

6.4 Simulation units

The Lennard-Jones potential is a useful model for the interaction in, for instance, noble gases such as neon and argon. The Lennard-Jones parameters for argon and neon are listed in table 1 together with the atom masses.

Table 1. Lennard-Jones parameters and atom masses for argon and neon.

	ε (10^{-22}J)	σ (10^{-10}m)	m (10^{-26}kg)
Ne	4.18	3.12	3.37
Ar	16.1	3.405	6.67

Except for the difference in mass and Lennard-Jones parameters, the simulation of neon and argon is very much the same. In fact, the choice of simulation parameters simply impose a different scaling in each simulation, and in other respects the simulations are identical. As a consequence, each result obtained in a simulation of neon, can afterwards be reinterpreted as a result valid for argon. The interpretation is only a matter of choosing simulation units. To see this let

$$\varepsilon = \bar{\varepsilon} U_E, \tag{34}$$

$$\sigma = \bar{\sigma} U_{\mathrm{L}}, \tag{35}$$
$$m = \bar{m} U_{\mathrm{M}}, \tag{36}$$

where barred quantities are the parameter values in simulation units, which are denoted by U_{E} (energy), U_{L} (length), and U_{M} (mass). In practice, the barred quantities are set to suitable values that simplify the simulation (for instance, $\bar{\varepsilon} = \bar{m} = 1$), and the simulation units are then derived from equations (34)–(36) as

$$U_{\mathrm{E}} = \varepsilon / \bar{\varepsilon}, \tag{37}$$
$$U_{\mathrm{L}} = \sigma / \bar{\sigma}, \tag{38}$$
$$U_{\mathrm{M}} = m / \bar{m}. \tag{39}$$

For instance, in the simulation we may choose $\bar{m} = 1$, and for neon the simulation unit for mass then becomes

$$U_{\mathrm{M}} = 3.37 \cdot 10^{-26} \text{ kg}.$$

The length unit is obtained by fixing $\bar{\sigma}$ so that, for instance, simulating neon with $\bar{\sigma} = 1/8$ means that $U_{\mathrm{L}} = 8 \times 3.12 \cdot 10^{-10}$ m. To obtain the time unit, remember that the energy unit in the SI-system can be written as

$$1 \text{ J} = 1 \frac{\text{kg m}^2}{\text{s}^2}.$$

This relation between the energy unit and the basic units must hold also in the unit system used in the simulation, and hence,

$$U_{\mathrm{E}} = \frac{U_{\mathrm{M}} U_{\mathrm{L}}^2}{U_{\mathrm{T}}^2}. \tag{40}$$

The time unit then becomes

$$U_{\mathrm{T}} = \sqrt{\frac{U_{\mathrm{M}} U_{\mathrm{L}}^2}{U_{\mathrm{E}}}} = \sqrt{\frac{(m/\bar{m})\,(\sigma/\bar{\sigma})^2}{\varepsilon/\bar{\varepsilon}}} \tag{41}$$

The pressure unit is obtained in a similar way, and the temperature unit is determined by the simulational value of the Boltzmann constant, $\bar{k}_B$. Normally $\bar{k}_B = 1$ is used.

6.5 Exercise

2. The equation of state for an almost ideal gas

7 Particle simulations based on the field concept

The particle-particle methods described in the previous sections can be used only when the particles interact via a short-ranged potential. Whenever $V(r)$ decays slowly, the potential cannot be truncated. The forces must then be computed by summing over all the particles in the system, and the number of operations required for each time step scales as N^2. There are many examples in nature where the strength of the interaction decays slowly, including gravitational and electrostatic interactions. In both these cases $V(r)$ decays as $1/r$, but the number of particles at a distance r from a test particle scales as r^2. This means that even though the influence on the test particle from any given particle at distance r may be very small, the system as a whole influences the test particle basically over all distances. In this section we shall demonstrate how the field concept can be used to simulate a system of charged particles. To simplify the discussion we limit ourselves to the electrostatic case.

Suppose that we apply the particle-particle method on system of N electrically charged particles. In the electrostatic case the equations of motion can be written

$$\dot{\mathbf{x}}_i = \mathbf{v}_i \tag{42}$$

$$m_i\dot{\mathbf{v}}_i = \frac{1}{4\pi\epsilon_0}\sum_{j\neq i}^{N} q_i q_j \frac{(\mathbf{x}_i - \mathbf{x}_j)}{|\mathbf{x}_i - \mathbf{x}_j|^3} \tag{43}$$

where q_i is the charge of the i:th particle. Without introducing any approximations, we can write (43) with the help of an electric field as

$$m_i\dot{\mathbf{v}}_i = q_i\mathbf{E}(\mathbf{x}_i), \tag{44}$$

where

$$\mathbf{E}(\mathbf{x}) = \frac{1}{4\pi\epsilon_0}\sum_{j} q_j \frac{(\mathbf{x} - \mathbf{x}_j)}{|\mathbf{x} - \mathbf{x}_j|^3} \tag{45}$$

and where it is assumed the field of the particle itself is not included in the summation.

Suppose that the electric field has been calculated at $\mathbf{x}_i$ and $\mathbf{x}_k$, and that we now want to compute the force on a particle at $\mathbf{x}_j$ (see figure 14). We can do this by performing the summations in (43) or (45). One way of avoiding the costly summation is to estimate $\mathbf{E}(\mathbf{x}_j)$ with the help of the field at $\mathbf{x}_i$ and $\mathbf{x}_k$, e.g., by linear interpolation. To estimate the field this way would be very cheap, but results in an error. The size of this error depends on how fast the field varies in space in relation to the distances between $\mathbf{x}_i$, $\mathbf{x}_k$ and $\mathbf{x}_j$. In some cases we know beforehand that the field is slowly varying, or we may tolerate a certain error in order to speed up the computation. In such cases we may save time by computing the field exactly at a few particle positions and then use these field values to interpolate the field to the rest of the particles. However, this is not a rational way of using the field concept, since the accuracy would depend on how the particles are distributed in space.

Instead we introduce a spatial grid, and represent the field by its value $\mathbf{E}_j$ at the grid point $\mathbf{X}_j$ at the center of each cell. Here j is the index of the cell. Sometimes it is not practical to work directly with a vector field such as the electric field, and instead we use the electrostatic potential $\phi(\mathbf{x})$ satisfying

$$\mathbf{E}(\mathbf{x}) = -\nabla\phi(\mathbf{x}). \tag{46}$$

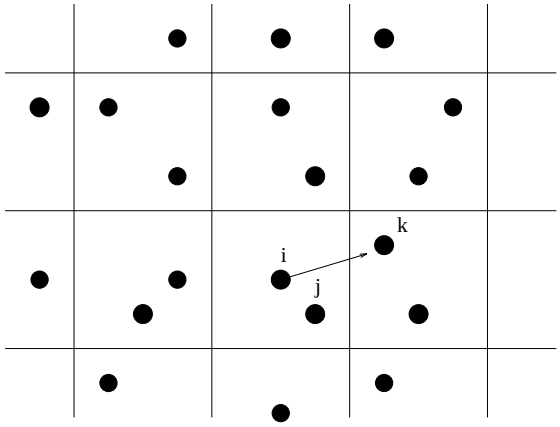


Figure 14: The use of field equations and a spatial mesh.

Once the potential is known at the grid points, the electric field can be computed through a finite difference equation approximating (46).

The electrostatic potential satisfies Poisson's equation

$$\nabla^2\phi = -\rho/\epsilon_0, \tag{47}$$

where the charge density, ρ, is defined by

$$\rho(\mathbf{x}, t) = \sum_{i=1}^{N} q_i \delta(\mathbf{x} - \mathbf{x}_i). \tag{48}$$

Once we obtain the charge density on the grid, we can solve Poisson's equation to obtain the potential, and thereby, also the electric field. However, computing the charge density is not trivial. Equation (48) cannot be used directly. The delta function in (48) must be handled numerically in such a way that all particles within a small but finite distance ς from $\mathbf{X}_j$ contribute to the charge density at $\mathbf{X}_j$. However, when this distance is small, only a small number of particles in the simulation are found within this distance from $\mathbf{X}_j$. This means that a majority of the particles in the system will not affect the charge density at the grid points, thereby having no influence on the dynamics of the system. Instead of (48) we must use a method that allows all particles to affect the charge density on the grid. We postpone a detailed discussion on how to achieve this. Here we just note that somehow we must assign each charge to one or more cells.

The use of a grided field and charge assignment is described by Hockney and Eastwood as the particle-mesh method.

Assuming that the force computation based on the discretized representation of the electric field is to be as accurate as in (43), the distance Δ between the grid points must be much smaller than the distance between the particles. That is

$$\Delta << d \approx \frac{L}{N^{1/3}}, \tag{49}$$

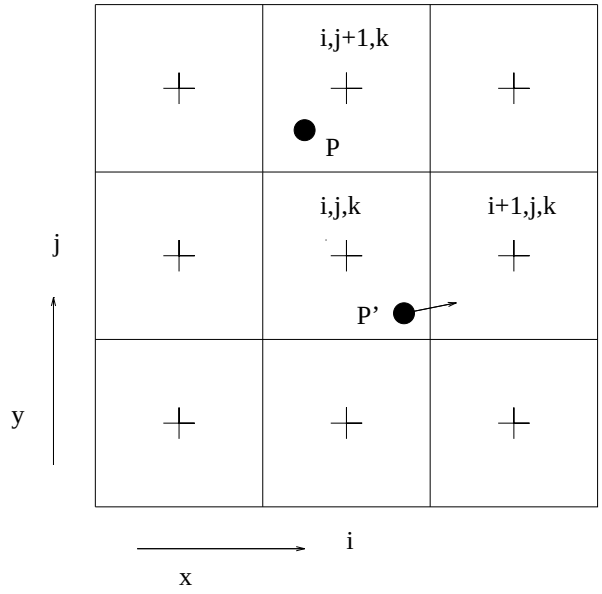


Figure 15: Illustration to why few particles per cell causes unwanted fluctuations in the solution [after Potter, 1973]

where we have assumed that the system occupies a volume L^3. This means that the grid points must be at least as many as the particles in the system, and this suggests that there is nothing to gain by introducing the grid. However, as noted earlier, we can tolerate some errors in order to speed up the computation, and consequently, we may use fewer cells whose size does not necessarily satisfy (49).

A problem arises when the number of particles is increased so that $\Delta \approx d$. This problem is illustrated in figure 15. This figure shows a part of a grid in three dimensions, where each cell is identified by three indices i, j and k. Suppose that a particle p is in cell $(i, j+1, k)$ and that another particle p' is in cell (i, j, k). Furthermore, assume that the charge density on the grid is obtained by assigning the charge of each particle to the nearest grid point. When $\Delta \approx d$ it is quite likely that each cell contains only one or at most a few particles. Therefore, the potential at p will change drastically when p' is displaced slightly so that it ends up in cell $(i+1, j, k)$ rather than in cell (i, j, k). Hence, small disturbances in particle positions lead to large fluctuations in the potential, and therefore, computer models with $\Delta \approx d$ suffer from large noise levels.

To avoid the problem of large fluctuations, the number of particles can be increased. When cells (i, j, k) and $(i+1, j, k)$ both contain, say, 1000 particles, it does not matter very much if one or a few particles ends up in the "wrong" cell. Therefore, from a numerical point of view it is more convenient to use the field concept on a grid where the cell size satisfies the opposite of (49), that is,

$$\Delta \gg d. \tag{50}$$

Clearly, since the exact position of each particle is of minor importance in such a model, the behavior of a single particle is not of as much importance as the collective behavior of the system as a whole. It remains to be seen if there exists such a system in reality, and to answer this question we are going to look more closely at the physics of a plasma.

8 Basic concepts in plasma physics

In physics, plasma is a state of matter just as solid, fluid and gas. Matter can be transformed into a gaseous state by heating of a liquid (or in some case, a solid) to its boiling temperature. When the gas is heated further the atoms or molecules are ionized, and a state is formed where the ions and the electrons are separated from each other. The thermal motion of the particles prevents recombination, and in this way the charged particles form an ionized gas controlled mainly by electromagnetic forces. Such a gas is called a plasma.

The plasma state is believed to be the dominating state in the universe. It is often said that more than 99.99% of the matter in the universe is in the state of a plasma. Interplanetary space is filled with plasma constantly radiating from the sun in the so-called solar wind. The Earth's magnetic field acts as a shield against the solar wind, and forms a cavity around the planet, the Earth's magnetosphere (see figure 16). Some of the solar wind plasma "leaks" into the magnetosphere and, together with plasma originating from the Earth's atmosphere,

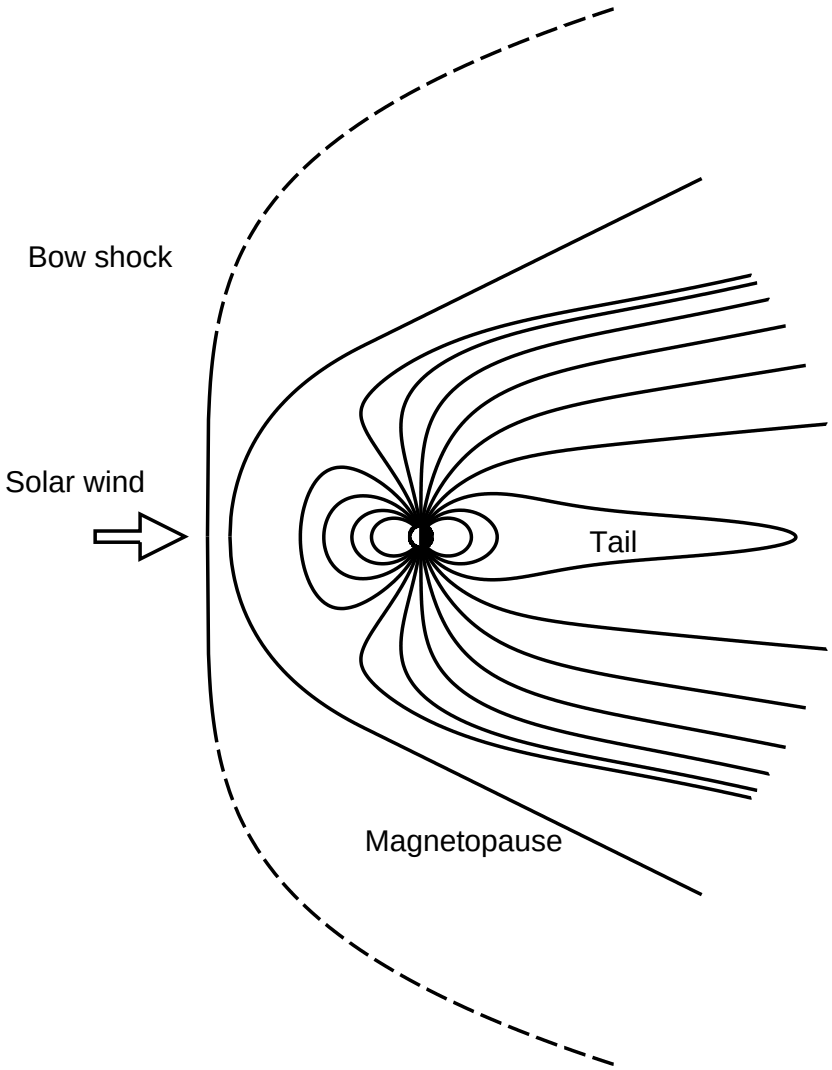


Figure 16: The Earth's magnetosphere

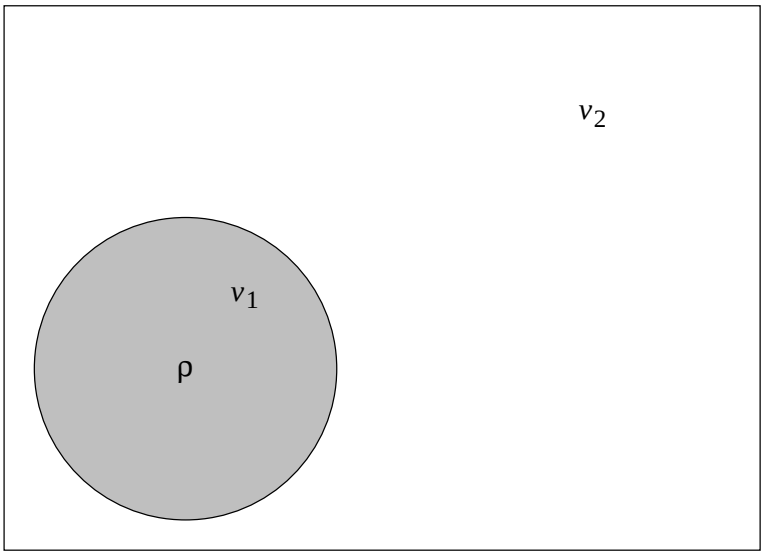


Figure 17: The Debye sphere [after Potter, 1973].

this plasma forms various particle populations in the magnetosphere. The radiation belts, or the Van Allen belts, were discovered during the first satellites missions in 1958. Closer to the Earth solar radiation ionizes the upper atmosphere, and a conducting layer, the ionosphere, is formed roughly 100 km above the surface of the Earth. The aurora (northern lights) are caused by plasma processes in the ionosphere and the magnetosphere. In our daily life we can "see" plasmas in lightning discharges, neon lights etc. Plasma phenomena are of importance for lasers, and the free electrons and holes in a semi conductor behave roughly as a plasma. Plasmas are also created in laboratories in the search for a controllable fusion reactor.

8.1 The Debye length

The particles in a plasma interact via the Coulomb force. Due to this long-ranged interaction, the motion of the plasma particles is coupled over long distances. In this way the plasma displays a "collective" behavior. At shorter distances the particles reacts as individual particles. Due to this dual properties of a plasma, a test particle in the plasma is affected by an electric field with two sources

$$\mathbf{E} = \mathbf{E}_1 + \mathbf{E}_2. \tag{51}$$

Here the field $\mathbf{E}_1$ is due to close encounters with other particles in the plasma, and $\mathbf{E}_2$ is the field generated collectively by all particles in the plasma.

Consider a plasma at thermal equilibrium at a temperature T, and suppose that we remove all electrons within a sphere of radius a (see figure 17). This sphere is then positively charged. If the radius a is small enough, the positive charge will not affect other particles much. As a measure of how much a test electron is affected by the charged sphere, we look at the change in kinetic energy of an electron "falling" from the edge of the sphere into its center. A test particle at the edge of the sphere typically has a kinetic energy $mv^2/2 \approx k_B T$. The increase of this energy caused by the positively charged sphere as the electron falls towards the center is roughly ne^2a^2/ϵ_0, where n is the plasma density and e is the electron charge. The Debye

length λ_D in a plasma is defined by

$$k_B T = ne^2 \lambda_D^2 / \epsilon_0, \tag{52}$$

or

$$\lambda_D = (\frac{\epsilon_0 k_B T}{ne^2})^{1/2}. \tag{53}$$

We see that charge density perturbations, or field perturbations, at scale lengths larger than λ_D are required to affect the test particle significantly. Provided that the number of particles N_D in the Debye sphere (a sphere with radius λ_D) is sufficiently large, the test particle cannot "see" individual particles in the sphere. Therefore, the field fluctuations affecting the test particle must correspond to $\mathbf{E}_2$ in (51). This means that collective effects in a plasma dominates at scale lengths larger than the Debye length.

Note that the change in kinetic energy was calculated under the assumption that the charge within the sphere could be represented by a continuous charge distribution $\rho = ne$. This means that we already from the start assumed that N_D was sufficiently large. When N_D is small enough, the interaction between individual particles in the sphere must be considered explicitly. The parameter N_D is an important parameter in plasmas. Indeed, an ionized gas is not considered to be a plasma unless N_D is sufficiently large. It can be shown that effects of binary collisions, basically the field $\mathbf{E}_1$ in (51), are very small in plasmas, and therefore, collective effects dominate in plasmas. In many natural plasmas, such as space plasmas, binary collisions can be neglected completely, and therefore, these plasmas are described as collisionless. A computer simulation of a collisionless plasma must contain a sufficient number of particles to ensure that the simulation results are not quenched by noise from binary collisions.

A somewhat different interpretation of the Debye length is obtained by considering the effective field from a test charge in a plasma. A test electron will repel other electrons and attract ions. Thereby a positively charged layer is created around the test electron, and this layer effectively shields off the field from the test particle. While the Coulomb potential from a test particle in vacuum varies as $1/r$, the effective potential in a plasma is given by

$$V_{\text{eff}}(r) \propto \frac{1}{r} \exp[-r/\lambda_D]. \tag{54}$$

Thus, the Coulomb potential is shielded off over a scale length λ_D. This phenomenon is described as Debye shielding.

8.2 The plasma frequency

The charged particles in a plasma respond quickly to electric fields in the plasma. When an electric disturbance appear, the particles, mainly the lighter electrons, move to screen the electric field. The inertia of the electrons causes the electrons to pass their equilibrium positions where the electric field vanishes, and in that way a new oppositely directed electric field is created. To screen off this field the electrons will react by moving in the opposite direction, again passing by their equilibrium positions, creating a new field and so on. The result is a collective oscillation by the electrons, a plasma oscillation.

To determine the frequency of the plasma oscillations, let us consider a fluid description of the plasma. It is reasonable to use a fluid description here, since we want to describe a collective oscillation, where a large number of particles are involved. To simplify the problem we assume that the frequency is high enough so that the ions will not have time to respond to the fluctuating field. Thus, we start from the following system of equations

$$m[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v}] = -e\mathbf{E} \tag{55}$$

$$\mathbf{J} = \rho \mathbf{v} \tag{56}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} \tag{57}$$

where m is the electron mass and ρ is the electron charge density. The first two equations are analogous to the equations for momentum and mass conservation used when describing classical fluids (e.g., see Tritton).

To solve the above equations we consider a small perturbation from an equilibrium. In the equilibrium the plasma is characterized by $\mathbf{v}_0, \rho_0, \ldots$, while perturbations have index 1; $\mathbf{v}_1, \rho_1, \ldots$ We assume that the equilibrium quantities satisfy equations (55) - (57), and that all products of perturbation quantities, such as $\rho_1 \mathbf{v}_1$, can be neglected. The remaining linearized equations can be rewritten by assuming that all perturbation quantities vary as $\exp[i(\mathbf{k} \cdot \mathbf{x} - \omega t)]$:

$$im\omega \mathbf{v}_1 = e\mathbf{E}_1 \tag{58}$$

$$\mathbf{J}_1 = \rho_0 \mathbf{v}_1 \tag{59}$$

$$\mathbf{J}_1 - i\epsilon_0 \omega \mathbf{E}_1 = 0. \tag{60}$$

We are looking for an electrostatic oscillation without varying magnetic field, and hence no $\mathbf{H}_1$ appears in the linearized equations. Furthermore, we assume that in the equilibrium state the plasma is at rest so that $\mathbf{v}_0 = 0$.

By expressing $\mathbf{J}_1$ in terms of $\mathbf{E}_1$ with the aid of (58) and (59), (60) can be rewritten as

$$i\epsilon_0 \omega (\frac{n_0 e^2/\epsilon_0 m}{\omega^2} - 1)\mathbf{E}_1 = 0 \tag{61}$$

where ρ_0 has been replaced by $-n_0 e$. Non-trivial solutions to equation (61) exist only when

$$D = \frac{n_0 e^2/\epsilon_0 m}{\omega^2} - 1 = 0. \tag{62}$$

That is, when

$$\omega = \omega_p \tag{63}$$

where the plasma frequency, ω_p, is defined by

$$\omega_p^2 = \frac{n_0 e^2}{\epsilon_0 m}. \tag{64}$$

8.3 The dispersion relation and instabilities

In the previous section we found that plasma particles can oscillate in a simple harmonic oscillation. The frequency of this oscillation depends on the electron mass and charge and the plasma density, but not on the wave vector **k**, that is, the wave length of the oscillation. Actually, this result is not entirely correct, but rather the result of the approximations used to derive (62). The momentum equation (equation 55) for a classical fluid at a finite temperature normally includes a pressure gradient term (e.g., Tritton). Neglecting this term corresponds to neglecting the thermal motion of the plasma particles, often called the cold plasma approximation. Cold plasmas do not exist in nature, since the thermal motion is required to avoid recombination. Still, the cold plasma approximation is used frequently as it leads to a simple description of several important plasma phenomena.

When the thermal motion of the particles is included, we obtain an equation similar to (62), but a term including the temperature and the wave vector appears. The frequency is determined as a function of **k** through the dispersion relation

$$D(\omega, \mathbf{k}) = 0, \tag{65}$$

and the result for plasma oscillations is given by

$$\omega^2 = \omega_p^2 + \frac{3}{2}k^2 v_{th}^2 \tag{66}$$

where $v_{th}^2 = 2k_B T/m$ and $k = |\mathbf{k}|$.

The frequency in (66) is real in this case, but in general the solution of the dispersion relation is complex

$$\omega = \omega_R + i\gamma. \tag{67}$$

This is true also for the plasma oscillations, and ω in equation (66) should really be replaced by ω_R. When the imaginary part of ω is negative, the waves are damped and will eventually fade away. On the other hand, when $\gamma > 0$ the waves will grow exponentially. In a plasma without external sources, the energy in the electric field can be transfered only to/from the particles in the plasma. For a plasma in thermal equilibrium, γ is always negative, that is, any electric field perturbation will be damped by the plasma particles and the field energy will be transfered into kinetic energy of the particles. A plasma that is not in thermal equilibrium contains free energy that can be transformed into electromagnetic field energy. A plasma where small fluctuations in the fields can grow at the expense of kinetic energy of the particles is described as unstable.

An ordinary gas develops towards equilibrium via collisions among the particles in the gas, that is, via interactions over distances of the order of the typical distance between the gas particles. In a collisionless plasma the particles are redistributed in a process where electric fields are generated spontaneously in the plasma. Hence, in a sense, plasma instabilities take on the role of collisions in an ordinary gas. However, note that the wave length of the generated electric fields is much longer than the typical inter-particle distance.

8.4 Electromagnetic waves, inhomogeneities and non-linear effects

Even though the discussion so far has been focused on electrostatic oscillations, waves with an oscillating magnetic field are quite common in plasmas. A purely electrostatic wave satisfies $\mathbf{k} \times \mathbf{E} = 0$, while an electromagnetic wave with purely transverse field components has $\mathbf{k} \cdot \mathbf{E} = 0$. A plasma can sustain oscillations in the full range between these two extremes. Here we focus on purely electrostatic waves since these are more accessible to computer simulations.

Without explicitly saying so, we have assumed that the equilibrium quantities are homogeneous in space and time, that is, $\nabla n_0 = 0, \partial n_0/\partial t = \ldots = 0$. In many ways an inhomogeneous plasma behaves differently from a homogeneous one, and gradients play a significant role in, e.g., laboratory plasmas. In space physics the homogeneous approximation is often useful, as it leads to a simplified description of the plasma.

Our discussion so far has been based on linearized equations, which are valid only as long as the amplitude of the perturbations is small. A perturbation in an unstable plasma cannot grow indefinitely, since there is only a finite source of free energy. Eventually, nonlinear effects will reduce the growth rate and stabilize the plasma. Another nonlinear effect is wave coupling. While linear waves can propagate independent of each other, large amplitude waves couple and transfer energy between themselves. Computer simulations have played a significant role in the understanding of nonlinear effects in plasmas.

9 Particle in cell simulation of plasmas

The Debye length defines the range of scale lengths for which collective effects dominate in the plasma. The Debye length also introduces a natural coupling to the computer model described in section 7. As already mentioned, in a spatial grid with a large number of particles per cell, the individual particles to some extent loose their identity. In the same way the individual plasma particles loose their identity in collective phenomena acting over scale lengths longer than the Debye length. Consequently, it seems logical to simulate a plasma using the particle-mesh method, and to choose the cell size to be of the order of the Debye length.

9.1 The computational cycle

The computational cycle of the particle-mesh method when applied to a plasma is described in figure 18. The index j is used to denote quantities evaluated at the grid points. The computational cycle consists of four steps, where the first step corresponds to **accel** and **move** (see section 5). This step, which is represented by the box at the top of figure 18, computes the acceleration and updates particle positions and velocities. The rightmost box in figure 18 represents a step where the charge density is evaluated at the grid points (note that ρ has index j), using the charge of each particle and the current particle positions. The following step integrates Poisson's equation to obtain the potential ϕ_j, and computes the electric field on the grid via a finite difference scheme. Finally, the force on each particle is computed from the electric field at neighboring grid points.

Plasma simulations often employ the leap-frog algorithm. This algorithm is described in section 4, and hence, the first step will not be considered here. The three latter steps are described in more detail below.

9.1.1 Solving the field equations

The field equations to be solved are

$$\mathbf{E} = -\nabla\phi \tag{68}$$

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0}. \tag{69}$$

These equations can be rewritten by using Fourier transforms:

$$\mathbf{E}(\mathbf{k}) = -i\mathbf{k}\phi(\mathbf{k}) \tag{70}$$

$$\phi(\mathbf{k}) = \frac{\rho(\mathbf{k})}{\epsilon_0 k^2} \tag{71}$$

where the Fourier transform $\mathbf{E}(\mathbf{k})$ is defined by

$$\mathbf{E}(\mathbf{k}) = \int d\mathbf{x}\mathbf{E}(\mathbf{x})\exp[-i\mathbf{k}\cdot\mathbf{x}] \tag{72}$$

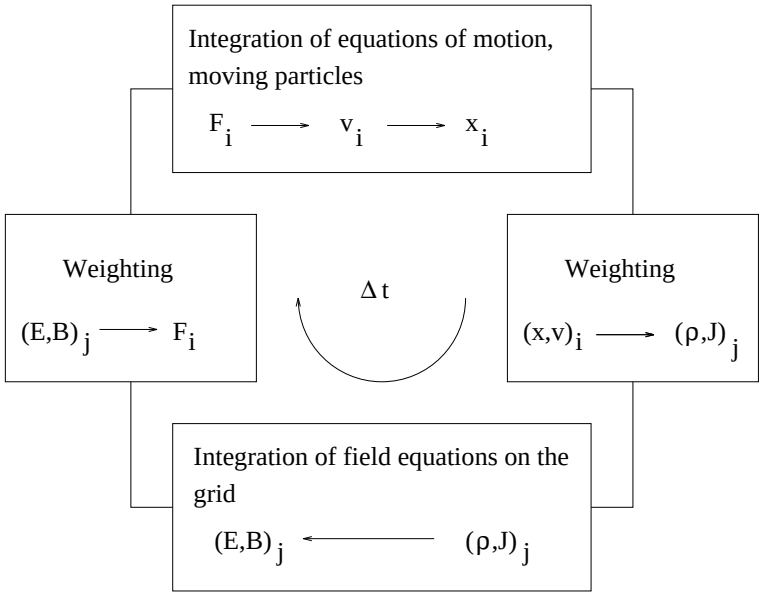


Figure 18: The computational cycle in the particle-mesh method [after Birdsall and Langdon, 1985].

and similarly for ϕ and ρ. The inverse transform is defined by

$$\mathbf{E}(\mathbf{x}) = \int \frac{d\mathbf{k}}{(2\pi)^3} \mathbf{E}(\mathbf{k}) \exp[i\mathbf{k} \cdot \mathbf{x}]. \tag{73}$$

In our case, where we use a discrete representation of the fields, we cannot use the continuous transform, but instead we expand the fields in a finite discrete Fourier transform (Fourier series). For simplicity, consider a one dimensional system of length L. For this case the finite discrete Fourier transform of the electric field can be written as

$$E(k) = \Delta x \sum_{j=0}^{N_G-1} E(X_j) \exp(-ikX_j), \tag{74}$$

where N_G is the number of grid points, and $X_j = j\Delta x$ are the coordinates for the grid points. The distance between neighboring grid points is denoted by Δx to emphasize that the system is one dimensional. A great advantage of the discrete Fourier transforms, from a computational point of view, is that they can be computed very efficiently with the aid of Fast Fourier Transform (FFT) techniques (see, e.g., Numerical Recipes).

The inverse discrete transform is defined as

$$E(X_j) = \frac{1}{L} \sum_{\ell=-N_G/2}^{N_G/2-1} E(k_\ell) \exp(ik_\ell X_j) \tag{75}$$

where $k_\ell = \ell(2\pi/L)$. Note that $|\mathbf{k}|$ take on $N_G/2$ distinct values in the range

$$k_{\min} \leq |k| \leq k_{\max}, \tag{76}$$

where $k_{\min} = 2\pi/L$ and $k_{\max} = (N_G/2)2\pi/L = \pi/\Delta x$. In terms of wave lengths these values correspond to a maximum and minimum wave length of L and $2\Delta x$, respectively.

In the one dimensional case we can write the finite difference approximation to (68) and (69) as

$$E_j = \frac{\phi_{j-1} - \phi_{j+1}}{2\Delta x} \tag{77}$$

$$-\frac{\rho_j}{\epsilon_0} = \frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{\Delta x^2}. \tag{78}$$

These equations can be rewritten with the aid of finite Fourier transforms as

$$E(k) = -i\kappa(k)\phi(k) \tag{79}$$

$$\phi(k) = \frac{\rho(k)}{\epsilon_0 K(k)^2} \tag{80}$$

where

$$\kappa(k) = k\left[\frac{\sin(k\Delta x)}{k\Delta x}\right] \tag{81}$$

and

$$K(k)^2 = k^2\left[\frac{\sin(\frac{1}{2}k\Delta x)}{\frac{1}{2}k\Delta x}\right]^2. \tag{82}$$

Notice that the factors κ and K^2 approach k and k^2, respectively, as $\Delta x \to 0$.

To solve the field equations on the grid, we transform the charge density ρ_j to obtain $\rho(k_\ell)$. Then, from (80), we obtain the transformed potential $\phi(k_\ell)$, which gives the electric field through the inverse transform of (79) or through the inverse transform of (80) together with the finite difference approximation in (77).

By using finite difference approximations, waves with wave lengths $\lambda < 2\Delta x$ will be interpreted erroneously as waves with wave lengths in the range $[2\Delta x, L]$. The same type of alias problem arises when frequency spectra are computed from a discrete time series. The aliasing phenomenon is described in more detail below. An additional consequence of the spatial grid is that the total energy is not conserved. This fact can be traced to the difference between $k, \kappa,$ and K.

9.1.2 Particle and field weighting

As discussed in section 7, the charge density on the grid must be determined by weighting the charge of each particle between neighboring grid points. In the simplest scheme each charge is assigned to the nearest grid point (NGP). The charge density on the grid is obtained simply as $\rho_j = n(j)q/\Delta x$, where $n(j)$ is the number of particles in cell j having the charge q. When the plasma contains particles with different charges, the charge density is obtained by summing over all particle species. In this way all particles contribute to the density on the grid. The contribution $n_j(x_i)$ of the i:th particle is shown in figure 19 as a function of particle position. The contribution produced by the particle is 0 as long as the particle is outside cell j, and jumps to 1 as soon as the particle crosses the cell boundary. As the particle leaves the cell, the density drops to 0 again.

The curve in 19(b) can be interpreted as the effective particle shape. The grid experience a particle of finite size and with a rectangular shape. It should be emphasized that this is

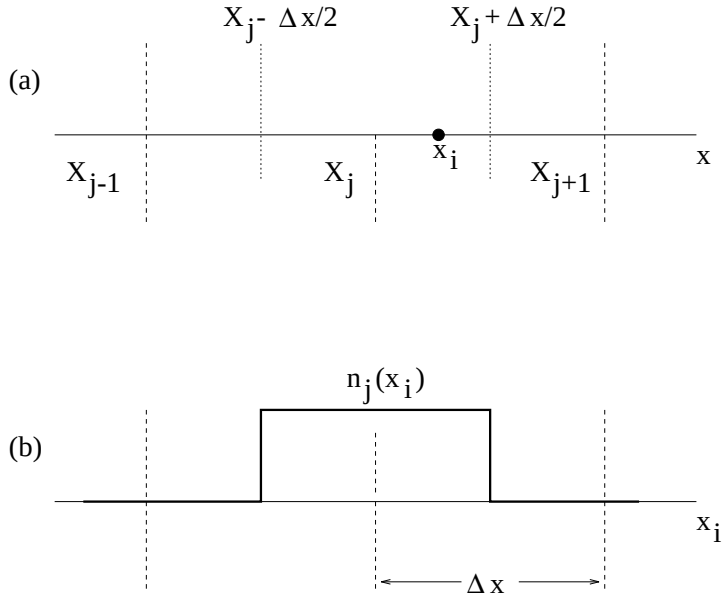


Figure 19: Nearest grid point (zeroth order) weighting. (a) The charges of all particles in the jth cell $[X_j - \Delta x/2, X_j + \Delta x/2]$ are assigned to X_j. (b) The density contribution from the ith particle to X_j as function of the particle position [after Birdsall and Langdon, 1985].

a true consequence of the grid and the charge weighting. As a consequence, we no longer simulate a system of point charges, but a system consisting of finite-sized particles. Thereby, the physics we observe in the simulation is the physics of a plasma consisting of finite-sized particles. This allows for a somewhat different interpretation of the simulation particles. One particle in the simulation does not necessarily correspond to one particle in the real plasma. As mentioned earlier, it is natural to choose the grid size of the order of the Debye length. As the "particle" in figure 19(b) has the same size as the cell, we can think of the simulation particle as a super particle representing the charges in the Debye sphere.

Note that two particles in the same cell do not interact with each other, since their charges are assigned to the same grid point. Consequently, finite-sized particles can pass through each other without colliding.

With NGP weighting the density $n(j)$ changes discontinuously when a particle pass a cell border. This causes fluctuations in the potential and the electric field. To reduce these fluctuations the charge of each particle can be assigned to more than one grid point. Figure 20 illustrates particle weighting between the two nearest grid points. In this scheme we may regard the super particle as a charge cloud of dimension Δx, where NGP weighting is applied to each part of the cloud (CIC - "cloud in cell"). The particle can also be treated as a point particle whose charge is distributed linearly with respect to the distance to the two nearest grid points (PIC - "particle in cell"). In both these cases the particle will contribute to the two nearest grid points (here j and $j+1$) according to

$$n(j) = \left[\frac{X_{j+1} - x_i}{\Delta x}\right] \tag{83}$$

$$n(j+1) = \left[\frac{x_i - X_j}{\Delta x}\right]. \tag{84}$$

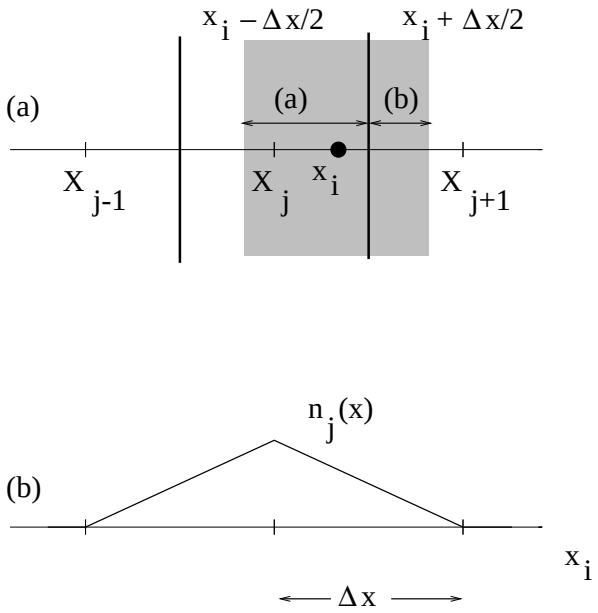


Figure 20: Cloud in cell (first order) weighting. (a) The weighting is done as if the particle is a charge cloud of size Δx (shaded). The part of the cloud that is in cell j is assigned to X_j (part a), while the part in cell $j+1$ is assigned to X_{j+1} (part b). (b) The density contribution from the ith particle to X_j as function of the particle position [after Birdsall and Langdon, 1985].

By weighting between two grid points, the grid experience a particle with a triangular shape of dimension $2\Delta x$, and the contribution to $n(j)$ varies slowly compared to the NGP case. Therefore, the CIC scheme produces less noise and is more suitable for plasma simulation than the NGP method.

The difference between NGP and CIC weighting can be described by the shape factor $S(x)$. In the two cases the form of the shape factor is given by the curves showing $n_j(x_i)$ in figures 19 and 20. In addition, the shape factor is normalized so that

$$\int_{-\infty}^{\infty} dx S(x) = 1. \tag{85}$$

Besides the discontinuities the shape factor is constant in the NGP scheme, that is, $S(x) \propto x^0$. In CIC, on the other hand, $S(x) \propto x^1$. Thus, the two weighting schemes are of zeroth and first order, respectively. These two weighting schemes are not the only ones. Higher order weighting, such as quadratic or Gaussian weighting, can be used to reduce noise even further. The simulation program described in chapter 10 employs NGP and CIC.

The force on the individual particle is obtained by weighting the electric field at the grid points to the location of the particle. It is important to use the same weighting scheme in both the particle weighting and the field weighting, in order to avoid self acceleration, that is, that the particle accelerates itself. Consequently, in CIC the electric field at $x = x_i$ should be calculated as

$$E(x_i) = \left[\frac{X_{j+1} - x_i}{\Delta x}\right] E_j + \left[\frac{x_i - X_j}{\Delta x}\right] E_{j+1} \tag{86}$$

9.2 The computer model and simulation techniques

In molecular dynamics the interaction between the particles in the system is approximated by a phenomenological potential. The short range of the potential makes it possible to truncate $V(r)$ for $r > r_c$, thereby neglecting the interaction at longer distances. The particles in a plasma interact through the Coulomb potential, whose long range makes the collective behavior of the plasma more important than binary collisions. This would justify truncating the potential for $r < r_c$ rather than $r > r_c$, where r_c in the plasma case would correspond to the Debye length. The Coulomb potential is not truncated explicitly in the particle-mesh method. Instead a grid is introduced and the charges are weighted between the grid points. The particle weighting makes the grid experience particles with a finite size, and this damps field fluctuations at wave lengths shorter than $\Delta x \approx \lambda_D$. The interaction between the plasma particles is not modeled by a phenomenological potential, but implicitly via the grid and the weighting scheme.

When the number of particles in the Debye sphere is large enough, it is possible to separate length scales, and to identify the scale length region where collective effects dominate. In addition, when $N_D >> 1$ the collision frequency is low. To make a plasma simulation meaningful, these conditions must be satisfied also in the simulated plasma. However, this can be achieved without having $N_D >> 1$. In the NGP case, the simulation particles can pass through each other without interacting. In principle, the same happens in the CIC case. Consequently, the effects of collisions are reduced by the finite size of the particles, independent of the number of particles in the Debye sphere. This means that plasma can be simulated

successfully with relatively few particles. Having too few particles in the simulation still leads to a higher noise level, but this is not necessarily important. The aim is not to remove collision effects all together, but to keep the noise level sufficiently low so that interesting physics can be observed in the simulation. For instance, a certain collision frequency ν is tolerable, as long as the simulation runs over a time less than ν^{-1}. The number of particles per cell needed in a certain simulation depends partly on the physical phenomenon to be investigated, but in general each cell should contain at least a few particles.

Even though the particle-mesh method has several qualities that makes it suitable for plasma simulation, it is still just a model of the real system. In the model the Coulomb interaction between point particles is replaced by a grided field and finite-sized particles. It is reasonable to say that the particle-mesh method lies even further from the real system than the models used in molecular dynamics. Therefore, it is important to investigate in more detail what kind of physics the particle-mesh method can represent.

9.2.1 The physics of a finite-sized particle plasma

By discretizing in space we introduce two types of unphysical effects; a discrete electric field, and, through the particle weighting, finite-sized particles. The purpose of this section is to investigate the physics of a plasma of finite-sized particles without including discretization effects.

The charge density in a system of point particles is defined through (48). To obtain the charge density in system of finite-sized particles, we replace the delta function in (48) by the shape factor

$$\rho_c(\mathbf{x}) = \sum_{i=1}^{N} q_i S(\mathbf{x} - \mathbf{x}_i) \tag{87}$$

where the index c indicates that ρ_c is the charge density for a system of cloud of charges. By using the charge density for point particles we can write

$$\rho_c(\mathbf{x}) = \int d\mathbf{x}' S(\mathbf{x}' - \mathbf{x})\rho(\mathbf{x}'). \tag{88}$$

The Fourier transform of (88) gives

$$\rho_c(\mathbf{k}) = S(\mathbf{k})\rho(\mathbf{k}) \tag{89}$$

where the transform of $S(\mathbf{x})$ is defined in accordance with (72). The transformed shape factor for NGP (one dimensional) is given by

$$S_0(k) = \frac{\sin \frac{1}{2}k\Delta x}{\frac{1}{2}k\Delta x} \tag{90}$$

and for CIC by

$$S_1(k) = \left[S_0(k)\right]^2. \tag{91}$$

It is instructive to compare the Coulomb force on a finite-sized particles (a charge cloud) with that on a point particle. In accordance with (88) the Coulomb force on a charge cloud is given by

$$\mathbf{F}_c(\mathbf{x}, \mathbf{v}) = \int d\mathbf{x}' S(\mathbf{x}' - \mathbf{x})\mathbf{F}(\mathbf{x}', \mathbf{v}) \tag{92}$$

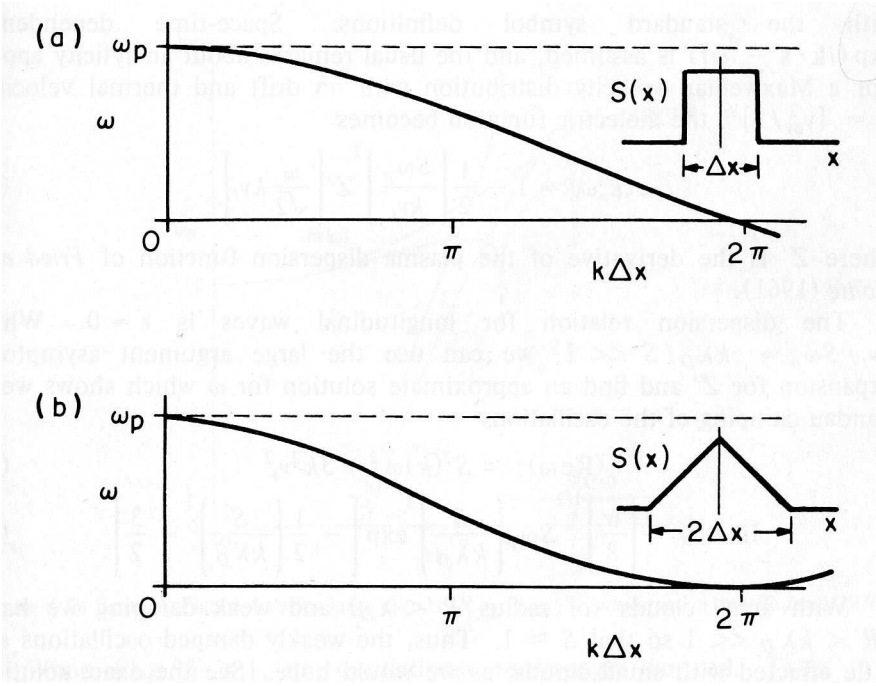


Figure 21: Dispersion in a cold finite-sized particle plasma: (a) Zeroth order shape factor; (b) First order shape factor [from Birdsall and Langdon, 1985].

where $\mathbf{x}$ denotes the coordinates of the center of the charge cloud, and $\mathbf{F} = q[\mathbf{E} + \mathbf{v} \times \mathbf{B}]$ is the force on a point particle. Fourier transforming (92) gives

$$\mathbf{F}_c(\mathbf{k}, \mathbf{v}) = S(\mathbf{k})\mathbf{F}(\mathbf{k}, \mathbf{v}). \tag{93}$$

For a particle of dimension R, $S(\mathbf{k})$ decays quickly for $|\mathbf{k}| > R^{-1}$. This means that the short wave length components of the Coulomb force are filtered away by the finite size of the particle.

In both these cases the cloud quantity is obtained from the corresponding point particle quantity by replacing the charge q by $qS(\mathbf{k})$. This means that many results for a finite-sized particle plasma can be obtained directly from ordinary plasma theory by multiplying the charge by the transformed shape factor. For instance, plasma oscillations in a cloud plasma becomes dispersive even in the cold case, since the plasma frequency is proportional to e. The frequency solving the linear dispersion relation

$$\omega = S(k)\omega_p \tag{94}$$

is shown in figure 21 for NGP (a) and CIC (b). Note that discretization effects are not included here. Such effects are discussed in the next section.

The dispersion relation for other types of plasma waves are changed in a similar way. This means that both propagation and stability properties of a cloud plasma are different than those of a point particle plasma. Figure 22 shows the solution of the dispersion relation for a warm plasma. The size of the particles is denoted by R, and in (a) we have $R/\lambda_D = 0.1$ (small clouds) while in (b) $R/\lambda_D = 2$ (large clouds). In both cases the results obtained for the cloud plasma agree with those obtained for a real plasma at wave lengths larger than R.

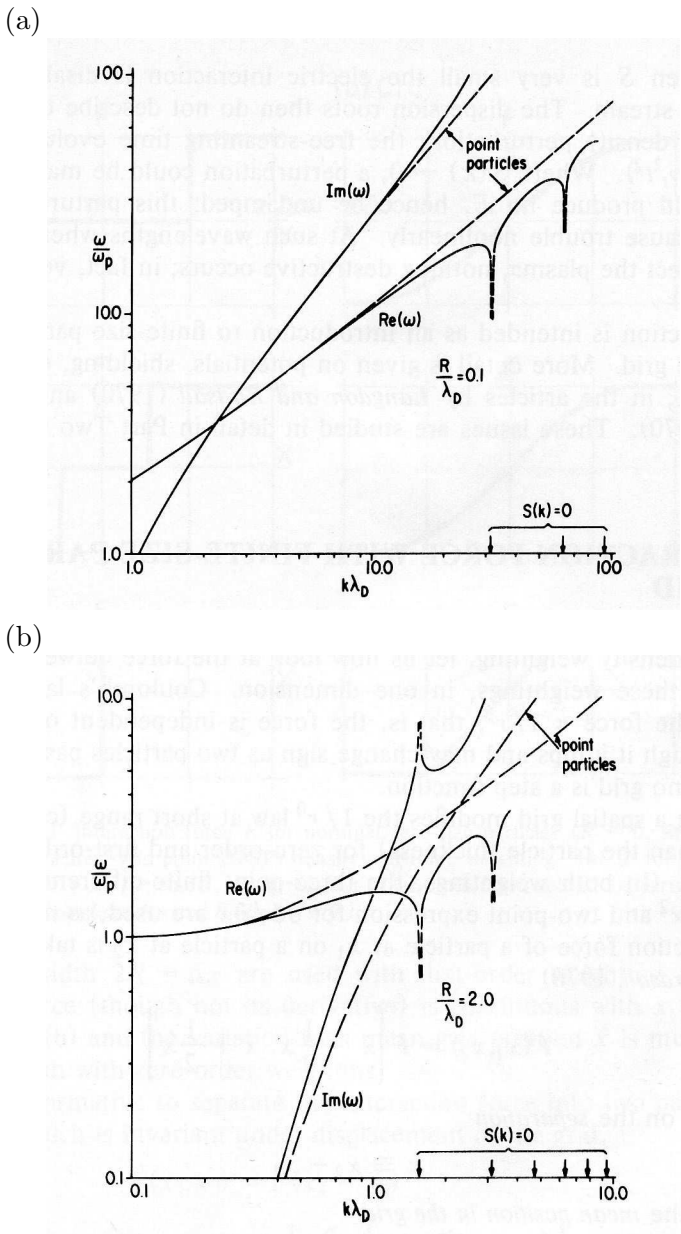


Figure 22: The solution to the dispersion relation for a warm finite-sized particle plasma. The particle half-size is denoted by R. (a) Small clouds $R/\lambda_D = 0.1$. (b) Large clouds, $R/\lambda_D = 2$. [From Birdsall and Langdon, 1985.]

Note that for some wave lengths $S(k) = 0$, and the frequency goes to zero. This means that density fluctuations at these wave lengths are not coupled to an electric field, and therefore, they are not damped.

9.2.2 Discretization effects

The charge density in a plasma consisting of charge clouds is defined by (87). Perturbations in ρ_c are coupled to the electric field, which in turn, controls the motion of the finite-sized particles. By studying how the discretization affects the charge density, we can study how the grid affects the physics in the simulated plasma. For simplicity, the following discussion is confined to one spatial dimension.

The charge density $\rho_c(x)$ is a continuous function of x, but in the simulation the representation of ρ_c is discrete. Let us denote by ρ_d the charge density registered on the grid. Obviously

$$\rho_d(X_j) = \rho_c(X_j) \quad j = 0, \ldots N_G - 1 \tag{95}$$

where as before $N_G =$ the number of grid points. We will investigate the relationship between these two quantities in Fourier space. Since ρ_c is continuous, containing an infinite number of points in the interval $0 \leq x \leq L$, and ρ_d is discretized, the two quantities are not represented in the same way in Fourier space. For the latter we use the finite transforms defined by (74) and (75), where a finite number of points in space are transformed into a finite number of Fourier coefficients, and vice versa, via

$$\rho_d(X_j) = \frac{1}{L} \sum_{m=-N_G/2}^{N_G/2-1} \rho_d(k_m) \exp(ik_m X_j) \tag{96}$$

and

$$\rho_d(k_m) = \Delta x \sum_{j=0}^{N_G-1} \rho_d(X_j) \exp(-ik_m X_j) \tag{97}$$

where

$$k_m = m\frac{2\pi}{L} \quad m = -\frac{N_G}{2}, \ldots, \frac{N_G}{2} - 1. \tag{98}$$

The continuous representation ρ_c is expanded in an infinite Fourier series according to

$$\rho_c(x) = \frac{1}{L} \sum_{n=-\infty}^{\infty} \rho_c(k_n) \exp[ik_n x] \tag{99}$$

where

$$k_n = n\frac{2\pi}{L} \quad n = 0, \pm 1, \ldots, \pm\infty \tag{100}$$

and the Fourier coefficients are given by

$$\rho_c(k_n) = \int_0^L \rho_c(x) \exp[-ik_n x]\, dx. \tag{101}$$

Compare (96) and (99) and note the difference in the Fourier representations of ρ_d and ρ_c. While the former is expanded in a finite number of Fourier modes with wave lengths in the

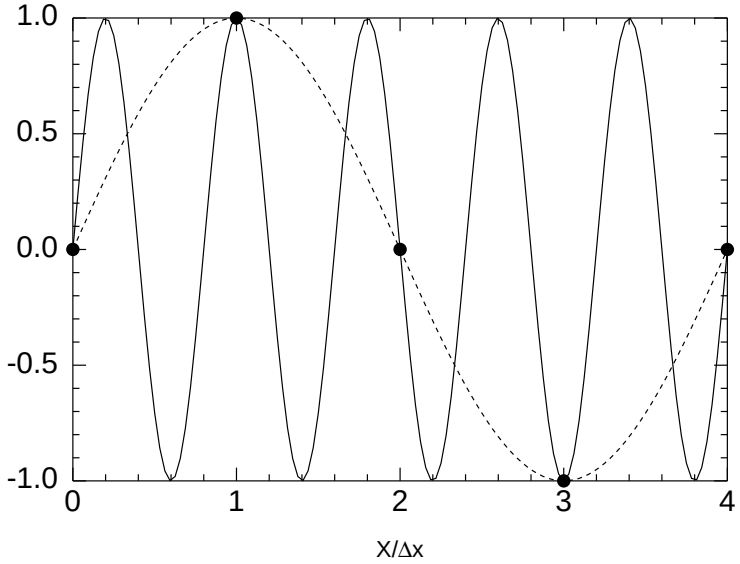


Figure 23: Illustration of the limitations of a discrete representation of a continuous function. Two sinusoidal waves with different wave lengths are recorded in exactly the same way on the discrete points $x/\Delta x = 0, \ldots, 4$.

interval $[2\Delta x, L]$, the latter is represented by an infinite number of Fourier modes with wave lengths in the interval $[0, L]$. Obviously, the infinite series contains more information about the spatial variations of the charge density than the finite series. The effect of discretization is demonstrated in figure 23. The charge density ρ_c varies as a sinusoidal wave in space, having a wave length in the interval $[0, 2\Delta x]$. Due to the finite resolution on the grid, the wave is recorded on the grid just as if the wave length is L.

To demonstrate how the grid couples field fluctuations in different wave length regimes, we write the coefficients in the finite Fourier series as

$$\begin{aligned}
\rho_d(k_m) &= \Delta x \sum_{j=0}^{N_g-1} \rho_c(X_j) \exp[-ik_m X_j] \\
&= \Delta x \sum_{j=0}^{N_G-1} \left\{ \frac{1}{L} \sum_{n=-\infty}^{\infty} \rho_c(k_n) \exp[ik_n X_j] \right\} \exp -ik_m X_j \\
&= \frac{1}{N_G} \sum_{n=-\infty}^{\infty} \rho_c(k_n) \sum_{j=0}^{N_G-1} \exp[i(k_n - k_m)X_j]. \qquad (102)
\end{aligned}$$

We can simplify the first summation by replacing n by $m' + pN_G$, where $-\infty < p < \infty$ and $-N_G/2 \leq m' < N_G/2 - 1$. We replace k_n by $k_{m'} + pk_g$, where $k_g = N_G(2\pi/L) = 2\pi/\Delta x$, and the sum over n by sums over m' and p:

$$\sum_{n=-\infty}^{\infty} \{k_n\} \rightarrow \sum_{p=-\infty}^{\infty} \sum_{m'=-N_G/2}^{N_G/2-1} \{k_{m'} + pk_g\}\,. \qquad (103)$$

Equation (102) can be rewritten as

$$\rho_d(k_m) = \frac{1}{N_G} \sum_{p=-\infty}^{\infty} \sum_{m'=-N_G/2}^{N_G/2-1} \rho_c(k_{m'} + pk_g) \sum_{j=0}^{N_G-1} \exp[i(k_{m'} - k_m)X_j], \tag{104}$$

where we have used the fact that

$$\exp[ipk_g X_j] = \exp[ip\frac{2\pi}{\Delta x} j\Delta x] = 1 \tag{105}$$

The indices m and m' now run over the same interval $[-N_G/2, N_G/2 - 1]$, and the sum over j can be summarized as

$$\sum_{j=0}^{N_G-1} \exp[i(k_{m'} - k_m)X_j] = N_G \delta_{m',m} \tag{106}$$

where $\delta_{m',m}$ is the Kronecker delta and where we again have employed equation (105). Equation (106) can be checked by replacing X_j by $j\Delta x$. The sum becomes a geometric series, which can be written in a closed form. Using (106) we can write the Fourier coefficients of the finite series as

$$\rho_d(k_m) = \sum_{p=-\infty}^{\infty} \rho_c(k_m + pk_g). \tag{107}$$

This equation expresses more exactly what we have already concluded. Fluctuations in the charge density ρ_c at wave lengths shorter than $2\Delta x$ are interpreted by the grid as fluctuations in the wave length interval $[2\Delta x, L]$.

As mentioned earlier, density fluctuations at wave lengths shorter than the particle size can be undamped, due to the fact that the transformed shape factor goes to zero at certain wave lengths. This means that such perturbations should be easily generated in the simulated plasma, and due to the aliasing described above, this could lead to a high noise level in the simulation. However, the alias effects are reduced by the finite size of the particle. To see the filtering effect of the finite particle size, we introduce $n_c(x)$ as the density of cloud centra. The charge density can be rewritten as

$$\rho_c(x) = q \int dx' S(x - x') n_c(x') \tag{108}$$

or, by using Fourier transforms, as $\rho_c(k) = qS(k)n_c(k)$. We can express $\rho_d(k_m)$ in terms of the cloud density as

$$\rho_d(k_m) = q \sum_{p=-\infty}^{\infty} S(k_m + pk_g) n_c(k_m + pk_g). \tag{109}$$

How much the density variations occurring at shorter wave lengths affects the simulation depends on how fast the Fourier transform of the shape factor decays. When the size of the particle is of the order of the Debye length, $S(k)$ decays rapidly for wave lengths shorter than λ_D.

In section 9.2.1 we noted that the finite size of the particles leads to dispersive plasma oscillations, even in the cold case. The dispersion relation shown in figure 21 does not include

discretization effects. The correct result for finite-sized particles interacting through a grid is obtained by summing over all wave length regions as in (109). Without going into details, the correct dispersion relation for plasma oscillations in the CIC case, including discretization effects, is given by

$$\omega = \omega_p \cos(\frac{1}{2} k\Delta x). \tag{110}$$

This result can be compared with the result obtained without grid effects

$$\omega = \omega_p \left[\frac{\sin \frac{1}{2} k\Delta x}{\frac{1}{2} k\Delta x} \right]^2 \tag{111}$$

9.3 Higher dimensions and electromagnetic simulations

The electrostatic simulations are particularly simple, as the fields are readily derived from Poisson's equation and from the current particle positions. In an electromagnetic simulation, where all of Maxwell's equations are used, the electric and magnetic fields are linked through the current in the system. The fields at a given time are not simply related to the current particle positions and their velocities. Instead the fields must be integrated in time in parallel with the particle coordinates. This makes the electromagnetic codes more complicated than the electrostatic ones.

In one dimensional simulations the particles can be interpreted as representing infinite two dimensional charged planes. Such a model can be useful for studying one dimensional plasma waves where the particle motion within a plane can be neglected or averaged away. One dimensional simulations have a limited application area, and most simulation codes in use today can handle more than one dimension. It is not unusual that a code can handle more velocity dimensions than spatial dimensions. Such simulations can be described as, for instance, $2\frac{1}{2}$ dimensional, to indicate that the simulation includes two space and three velocity dimensions. The particle weighting is done as in the one dimensional case, but with area (2d) or volume (3d) weighting. The number of particles required in the simulation increases quickly with the number of dimensions. A course estimate is a factor of 4 per velocity dimension and a factor of 64 per spatial dimension.

10 Electrostatic one-dimensional simulation (ES1)

In this section we present a computer program, *es1*, for simulation of electrostatic, one-dimensional plasma phenomena. The particular version that we will be using is a C-code written for the X-windows systems. Accordingly, the program is called *xes1*. The program is started by the command

`xes1` *filename*

where *filename* should be replaced by the name of an input file setting up all the computer variables needed in the code.

Parameters such as the system length and the plasma frequency provide a link between the computer simulation model and the physical system. Often the results are scalable, in the sense that a given simulation can be interpreted as a simulation of a set of physical systems related by a simple scaling. The scaling can be viewed as a scaling of the units used in the simulation. Since a completed simulation can be re-interpreted *a posteriori* by scaling the units, we should choose parameter values in the simulation that simplify computations as far as possible. Hence, the variables described below often do not have physical values in the simulation, but instead suitable values close to unity, to avoid over or under flow problems in the arithmetics of the computer.

We start by defining the most important variables used in *es1*, and then go through the system of units used in the code. Finally, we supply listings of the most important parts of the program.

10.1 Important computer variables

Notice that the code includes a large number of parameters, and that only the most important ones are listed below.

NSP Number of particle species (int). The particle species can differ in charge, temperature, density etc.

L System length (float). The units for length in the simulation is often chosen so that the system length is a multiple of π, as will be discussed in the section on simulation units.

DT Time step (float). The time stepping algorithm is explicit, and the time step must be small enough to ensure numerical stability.

NT Length of history plots (int). The diagnostics produced by the program includes history plots for, for instance, the total energy versus time. The length of these plots is set by the variable NT.

MMAX Number of Fourier modes in history plots (int). The diagnostics includes plots of energy in a particular Fourier mode as a function of time. To save memory, not all Fourier modes are stored, but only modes 1-MMAX.

NG Number of grid points (int). The system is divided into NG cells, with corresponding NG grid points at the center of each cell. It should be noticed that the first grid point is located at $x = 0$. This means that the first cell is only half a cell, and that grid point 0 and NG are identical points. This to make the system periodic, with a period L.

IW Weighting flag (int). Particle and field weighting is either NGP (IW = 1), CIC (IW = 2) or energy conserving (IW = 3).

EPSI $1/\epsilon_0$ (float). Normally set to one, and then one of the parameters defining the set of simulation units.

In addition, there are variables defined for each particle species.

N Number of particles (int).

WP Plasma frequency (float). Normally one of the variables defining the simulation units. Setting WP defines the density.

QM The charge to mass ratio (float).

VT1 Thermal velocity (float). Used during initialization to set up particles with a thermal velocity distribution.

V0 Drift velocity (float). Used during initialization to impose an initial particle drift.

MODE, X1, V1 The particles are set up with an initial perturbation determined by the three parameters MODE, X1, and V1. The perturbation is needed to achieve a dynamical situation in, for instance, a cold plasma where, otherwise, the particles would remain at rest and nothing would happen. The perturbation is sinusoidial in density or velocity. The density perturbation is achieved by displacing each particle a small distance from the equilibrium (no fields) position. The wavelength of the perturbation is set by the MODE flag, and the amplitudes by X1 (density) and V1 (velocity).

The above variables are all set in the input file supplied to the program on the command line. The input file used for setting up plasma oscillations in a cold plasma is given below.

File coldplas.inp:

```
COLD PLASMA OSCILLATIONS

Oscillations can be observed in a cold electron plasma in a neutralizing
nonmobile ion background.  The oscillations are initiated by perturbing
the density of the electrons in the fundamental mode.

nsp-----------l---------dt----------nt------mmax-----l/a
1        6.283185307    0.02         150     14         0
ng----iw----epsi-------a1------a2-------E0-------w0
32     2     1.00       0.00    0.00        0       0

SPECIES 1: Cold Electron Plasma

n----nv2---nlg---mode
64     0     1    1
wp----wc-----qm-----vt1----vt2---v0
1.00  0.00   -1.00  0.00  0.00  0.00
x1----v1---thetax--theta
0.001  0.0   0.00    0.00
```

End coldplas.inp

To move the particles, also the particle mass, M, and charge, Q, must be given, but these can be derived from the above parameters by noting from (64) that

$$\omega_p^2 = \frac{nq^2}{\epsilon_0 m}. \tag{112}$$

As the same relation must hold also in simulation units, we know that

$$\mathrm{WP} \cdot \mathrm{WP} = \mathrm{EPSI} \cdot \frac{\mathrm{N}}{\mathrm{L}} \cdot \mathrm{QM} \cdot \mathrm{Q}, \tag{113}$$

and the charge, Q, is obtained as

$$\mathrm{Q} = \frac{\mathrm{WP} \cdot \mathrm{WP}}{\mathrm{EPSI} \cdot \frac{\mathrm{N}}{\mathrm{L}} \cdot \mathrm{QM}}. \tag{114}$$

The mass, M, is then obtained as

$$\mathrm{M} = \mathrm{Q}/\mathrm{QM}. \tag{115}$$

10.2 Simulation units

The units in *es1* are set by defining the variables L (system length), WP (plasma frequency), EPSI ($1/\epsilon_0$) and QM (charge to mass ratio). Notice that, for instance, the charge unit can not be derived simply from (114). Even if we are simulating an electron plasma, the particle charge in the simulation does not correspond to one unit charge. Remember that the particles

in the simulation are super particles representing the charges in the Debye sphere, and we do not know beforehand how many real particles each super particle represents.

As in previous sections, we let U denote simulation units, so that U_t is the time unit, and so on. The length unit, U_ℓ, is defined by setting the physical length of our system. For instance, if we simulate a system with a length $\ell = 1000$ m and in the simulation we set $\mathrm{L} = 2\pi$, then this simply means that the length unit used in the simulation is

$$U_\ell = \ell/L = (1000\,\mathrm{m})/2\pi = 159\,\mathrm{m}.$$

Similarly, if we simulate a plasma with a plasma frequency of 178 kHz, corresponding to a density of 10 particles/cm^3, and set $\mathrm{WP} = 1$, then the time unit is immediately determined by

$$\omega_p = \mathrm{WP}\, U_t^{-1},$$

or

$$U_t = \mathrm{WP}/\omega_p = 1/(178 \cdot 10^3\ \mathrm{Hz}) = 5.62\mu\mathrm{s}.$$

The units U_q for charge and U_m for mass are derived in a similar way, but using also the parameters QM and EPSI. To be explicit, assume that the above plasma is an electron plasma. The charge to mass ratio then becomes

$$\frac{e}{m_e} = \mathrm{QM}\frac{U_q}{U_m}, \tag{116}$$

where e and m_e are the electron charge and mass expressed in SI-units. To find a second equation relating U_q and U_m we turn to the dielectric constant ϵ_0 and the parameter EPSI. In SI-units

$$\epsilon_0 = 8.854 \cdot 10^{-12}\frac{\mathrm{C}^2\mathrm{s}^2}{\mathrm{kg\,m}^3}$$

and, remembering that EPSI in the simulation represents $1/\epsilon_0$, we find that

$$\epsilon_0 = \mathrm{EPSI}^{-1}\frac{U_q^2 U_t^2}{U_m U_\ell^3}. \tag{117}$$

Solving (116) for U_q and inserting into (117) finally leads to

$$U_m = \epsilon_0\,\mathrm{EPSI}\,U_t^{-2}U_\ell^3\left(\frac{e}{m_e}\right)^{-2}. \tag{118}$$

In the above example, with $\mathrm{EPSI} = 1$, the mass unit becomes

$$U_m = 3.63 \cdot 10^{-17}\ \mathrm{kg}.$$

The charge unit is obtained from (116), and for the given example we find

$$U_q = \frac{1}{\mathrm{QM}}\left(\frac{e}{m_e}\right)U_m = 6.42 \cdot 10^{-6}\ \mathrm{C}.$$

Other units such as those for velocity, electrostatic potential, and electric field can be derived from the four units of charge, mass, time and length, as in the SI unit system.

11 Classical fluids

The Vlasov model is used to describe collisionless many body systems. Instead of tracing each particle we assume that the number of particles is large enough to allow us to describe the system using a phase-space density $f(\mathbf{x}, \mathbf{v})$. The variables $\mathbf{x}$ and $\mathbf{v}$ are both independent variables, and to simulate a Vlasov gas we discretize phase space and solve the Vlasov equation. Since phase space is six-dimensional the number of grid points increases rapidly with increasing system size. For instance, even for a moderate size of 32 grid points in each dimension, the total number of grid points is $(32)^6 \approx 10^9$, which is intractable from a numerical point of view. To be able to handle larger length scales, the Vlasov model must be abandoned, and the system modeled as a classical fluid. While the Vlasov model handles the system as a fluid in phase-space, a classical fluid is a fluid in configuration (ordinary) space. Below we describe the fluid equations and simple numerical schemes for solving them. The text summarizes selected parts of chapters 6 and 8 in the book by A.L. Garcia (see the references section).

11.1 The fluid equations

The classical fluid (simply fluid hereafter) model is used to describe phenomena acting over scale lengths much larger than the mean free path, that is, the distance a particle travels, on average, between collisions with other particles in the system. Furthermore, it is assumed that the state of the fluid can be characterized by a few physical quantities such as density, momentum, internal energy etc., which are all conserved along the fluid flow. This conservation is expressed by the fluid equations

$$\frac{\partial a}{\partial t} = -\nabla \cdot \mathbf{F}(a) \tag{119}$$

Note that here $\mathbf{F}$ denotes a flux. No forces will appear explicitly in this section, and hence, we take the liberty of using this, otherwise, somewhat confusing notation. In equation (119) a denotes one of the conserved quantities

$$a = \begin{cases} \text{mass density} \\ \text{momentum density} \\ \text{energy density} \end{cases} \tag{120}$$

and the flux one of

$$\mathbf{F}(a) = \begin{cases} \text{mass flux} \\ \text{momentum flux} \\ \text{energy flux} \end{cases} \tag{121}$$

The full set of fluid equations are called the Navier-Stokes equations, which also involve pressure terms as well as the temperature. The Navier-Stokes equations are all coupled, and an additional equation justified by the particular physical situation is needed to close the system. For instance, when the equations are used to describe an ideal gas, the ideal gas relation between pressure, temperature and density holds. Other assumptions such as incompressibility (density = constant) or adiabacity are frequently used. Here we focus on

the simplest cases, and consider the mass conservation equation

$$\frac{\partial \rho}{\partial t} = -\frac{\partial(\rho v)}{\partial x}. \tag{122}$$

This is the continuity equation in one dimension, and $\rho(x,t)$ is the mass density and the mass flux is simply the product of ρ and the fluid velocity $v(x,t)$.

To further simplify the problem, consider the case when the fluid velocity is constant $v(x,t) = c$. In this case, reverting to the a notation, the continuity equation becomes

$$\frac{\partial a}{\partial t} = -c\frac{\partial a}{\partial x}. \tag{123}$$

This is the advective equation, which is the simplest of all flux-conservation equations. The equation is also called the linear convection equation. The reason for this lies in its solution. Given the initial condition $a(x, t = 0) = a_0(x)$, the solution is given by $a = a_0(x - ct)$. That is the initial pulse is simply translated, or convected, without change in shape or amplitude.

11.2 The FTCS method

As we know the analytical solution, no numerical computations are really required to solve the advective equation. However, its simplicity makes is useful for introducing numerical schemes. Consider a simple numerical approach, where the time derivative is replaced by a forward difference approximation, while for the spatial derivative we use a centered difference scheme. That is, forward in time, centered in space (FTCS). Using the notation $a(x_j, t^n) = a_j^n$, the discretized advective equation becomes

$$\frac{a_j^{n+1} - a_j^n}{\Delta t} = -c\frac{a_{j+1}^n - a_{j-1}^n}{2\Delta x} \tag{124}$$

or

$$a_j^{n+1} = a_j^n - \frac{c\Delta t}{2\Delta x}(a_{j+1}^n - a_{j-1}^n). \tag{125}$$

Together with suitable boundary and initial conditions this gives a recursion relation for a. However, it turns out that this scheme is unstable for all Δt and Δx.

To see that (125) is unconditionally unstable we can perform a stability analysis similar to the one described in section 4.3.1. Assume a solution of the form

$$a(x,t) = z(t)e^{ikx} \tag{126}$$

where z is complex. For such a solution we see that

$$a_{j+1}^n = a_j^n e^{ik\Delta x}. \tag{127}$$

As in 4.3.1 we assume that

$$a_j^{n+1} = g a_j^n. \tag{128}$$

That is, we assume that $z(t + \Delta t) = gz(t)$, where g is the growth factor. Using (126)–(128) in (125) gives

$$g = 1 - i\frac{c\Delta t}{\Delta x}\sin(k\Delta x), \tag{129}$$

and the magnitude of g

$$|g| = \sqrt{1 + \left(\frac{c\Delta t}{\Delta x}\right)^2 \sin^2(k\Delta x)} \tag{130}$$

is always greater than unity. This means that the FTCS scheme is unstable irrespective of Δt and Δx.

The reason why the FTCS scheme is unstable can be traced to the fact that the finite difference approximation of the spatial derivative at x_j does not involve the value of a at that point. At the same time, the finite difference approximation of the time derivative only includes a at x_j. Hence, the time and spatial derivatives are not coupled, and this leads to instability. A simple way to remedy this problem is discussed below.

11.3 The Lax method

To obtain a coupling between the finite difference approximations of the temporal and spatial derivatives, we can replace a_j^n on the right-hand side of (125) by the average of the two neighboring values. This is Lax method, which becomes

$$a_j^{n+1} = \frac{1}{2}(a_{j+1}^n + a_{j-1}^n) - \frac{c\Delta t}{2\Delta x}(a_{j+1}^n - a_{j-1}^n). \tag{131}$$

By conducting a stability analysis similar to the one presented above, it can be shown that the growth factor for the Lax scheme is given by

$$|g| = \sqrt{\cos^2(k\Delta x) + (\frac{c\Delta t}{\Delta x})^2 \sin^2(k\Delta x)}. \tag{132}$$

For this case $|g| \leq 1$ as long as $c\Delta t/\Delta x \leq 1$, or

$$c\Delta t \leq \Delta x. \tag{133}$$

This is the Courant-Friedrichs-Levy (CFL) stability condition, which tells us that the quantity represented by a should not be convected across more than one spatial cell in one time step.

11.4 The Lax-Wendroff method

Lax method is of first order. To obtain a higher order method consider the Taylor expansion

$$a(x, t + \Delta t) = a(x, t) + \Delta t \left(\frac{\partial a}{\partial t}\right) + \frac{1}{2}(\Delta t)^2 \left(\frac{\partial^2 a}{\partial t^2}\right) + \text{higher order terms} \tag{134}$$

By using the general flux-conservation equation in one dimension

$$\frac{\partial a}{\partial t} = -\frac{\partial F(a)}{\partial x} \tag{135}$$

we can replace the first derivative in (134) by $-\partial F/\partial x$. An expression for the second derivative is obtained by differentiating (135) with respect to time

$$\frac{\partial^2 a}{\partial t^2} = -\frac{\partial}{\partial t}\frac{\partial}{\partial x}F = -\frac{\partial}{\partial x}\frac{\partial F}{\partial t}. \tag{136}$$

Here $\partial F/\partial t$ can be rewritten as

$$\frac{\partial F}{\partial t} = \frac{dF}{da}\frac{\partial a}{\partial t} = -\frac{dF}{da}\frac{\partial F}{\partial x}, \tag{137}$$

where we again have used (135). By replacing the first and second derivatives in (134), and by retaining terms up to second order we obtain

$$a(x, t+\Delta t) \approx a(x,t) - \Delta t\left(\frac{\partial F}{\partial x}\right) + \frac{1}{2}(\Delta t)^2\left(\frac{\partial}{\partial x}\frac{dF}{da}\frac{\partial F}{\partial x}\right) \tag{138}$$

By discretizing this equation we obtain the Lax-Wendroff scheme

$$\begin{aligned} a_j^{n+1} &= a_j^n - \frac{\Delta t}{2\Delta x}(F_{j+1}^n - F_{j-1}^n) + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 \cdot \\ &\quad \left[\left\{\frac{dF}{da}\right\}_{j+1/2}^n (F_{j+1}^n - F_j^n) - \left\{\frac{dF}{da}\right\}_{j-1/2}^n (F_j^n - F_{j-1}^n)\right]. \end{aligned} \tag{139}$$

Here $F_j^n = F(a(x_j, t^n))$, and the last term in (138) is discretized as

$$\frac{\partial}{\partial x}\left\{\frac{dF}{da}\frac{\partial F}{\partial x}\right\}_j \approx \frac{1}{\Delta x}\left(\left\{\frac{dF}{da}\frac{\partial F}{\partial x}\right\}_{j+1/2} - \left\{\frac{dF}{da}\frac{\partial F}{\partial x}\right\}_{j-1/2}\right). \tag{140}$$

The value of dF/da at $x_{j+1/2}$ is not available, but is approximated by the average of the derivatives at the points x_{j+1} and x_j, and similarly for the value of dF/da at $x_{j-1/2}$. For the simple case of the advective equation, where $F = ca$, $\partial F/\partial a = c$, and $\partial F/\partial x = c\partial a/\partial x$, the Lax-Wendroff scheme becomes

$$a_j^{n+1} = a_j^n - \frac{c\Delta t}{2\Delta x}(a_{j+1}^n - a_{j-1}^n) + \frac{1}{2}\left(\frac{c\Delta t}{\Delta x}\right)^2 (a_{j+1}^n - 2a_j^n + a_{j-1}^n) \tag{141}$$

The Lax-Wendroff method is more accurate than the lower order Lax scheme, but is subjected to the same stability condition, that is, the CFL condition. Both the Lax and the Lax-Wendroff method introduce numerical diffusion, and it is this diffusion that stabilizes the solution. In this sense the diffusion is a desirable property. However, for the Lax scheme the diffusion has the frustrating effect that, for a constant cell size Δx, the error increases rapidly as the time step is made smaller and smaller. The best solution is obtained for a time step at the limit of the CFL condition. For smaller time steps the numerical diffusion is strong enough to damp out the solution. The same type of damping occurs also in the Lax-Wendroff scheme, but here the damping is less severe. From this we can conclude that the time step and the cell size should be chosen close to the CFL condition, in order to minimize numerical diffusion problems.

12 Implicit numerical integration

Implicit methods were briefly mentioned in section 4.3.3, where it was emphasized that the stability of implicit methods may allow us to use a much larger time step. This is particularly important if the set of equations we intend to solve is stiff. Roughly speaking, stiffness means that the equations describe not only the (comparatively slow) processes that we are interested in but also some oscillations at much higher frequencies. These high frequency oscillations have small amplitude in the real solution, but may become unstable and blow up if an unsuitable numerical method is used and the time step is too large.

As an example we will here consider the electromagnetic field, described by Ampere's law

$$\partial_t \mathbf{E} = c^2 \nabla \times \mathbf{B} - \varepsilon_0^{-1} \mathbf{j} \tag{142}$$

and Faraday's law

$$\partial_t \mathbf{B} = -\nabla \times \mathbf{E} \tag{143}$$

To simplify the problem we will reduce it to one dimension by assuming $\mathbf{E} = E(t,z)\hat{\mathbf{x}}$, $\mathbf{j} = j(t,z)\hat{\mathbf{x}}$, and $\mathbf{B} = B(t,z)\hat{\mathbf{y}}$. When there are no variations in the x and y directions we have

$$\begin{array}{rcl} \partial_t E &=& -c^2 \partial_z B - \varepsilon_0^{-1} j \\ \partial_t B &=& -\partial_z E \end{array} \tag{144}$$

We will assume that the current is given, and that we are interested in the E and B fields associated with the current density j. As a simple example we may consider

$$j(t,z) = j_0 \sin(\Omega t - kz) \tag{145}$$

and to be specific assume $\Omega \sim 1/\text{s}$ and $k \sim 1/\text{m}$. In this case it is easy to find the solution

$$\begin{array}{rcl} E &=& \frac{\Omega}{\varepsilon_0(\Omega^2 - k^2c^2)}\, j_0 \cos(\Omega t - kz) \\ B &=& \frac{k}{\varepsilon_0(\Omega^2 - k^2c^2)}\, j_0 \cos(\Omega t - kz) \end{array} \tag{146}$$

To find this solution numerically with a reasonable resolution we would like to use a time step $\Delta t \sim 0.1$ s and a grid spacing $\Delta z \sim 0.1$ m. However, if we use an explicit method to integrate equations (144) we will be forced to use a much shorter time step. The reason is that in addition to the desired solution (146), equations (144) also describe electromagnetic waves, which exist also in vacuum where $j = 0$. The frequency of these waves is $\omega = kc$, and since the velocity of light c is so high ω may be much larger than Ω. When $k \sim 1/\text{m}$ we find $\omega \approx 3 \cdot 10^8/\text{s}$, and since stability of an explicit algorithm requires $\omega \Delta t < 1/2$ (see section 4.3.1) we would be forced to use $\Delta t \sim 10^{-9}$ s. This is at least 10^7 times shorter than the Δt we need to resolve the variations of interest, and an explicit numerical integration would be very inefficient. The stiffness of this problem is a result of the large ratio between the fastest time scale in the problem ($\sim c/\Delta z$) and the time scale of interest ($\sim \Omega$). An important advantage of implicit algorithms is that they allow us to solve stiff problems using a reasonably large Δt.

12.1 Local linearization of the equations

When deriving our implicit algorithm we will use a more general notation, employing equations (144) as an example to illustrate the meaning of the general formulas. We start by

considering a set of differential equations of the form

$$\partial_t \mathbf{F} = \mathbf{R}(\mathbf{F}, t) \tag{147}$$

where $\mathbf{F} = \mathbf{F}(t, z)$ is a vector containing the field components. In our example, $\mathbf{F} = (E, B)^T$, and the dimension of $\mathbf{F}$ is always equal to the number of scalar differential equations that describe the system at hand. Discretizing equation (147) in time symmetrically around $t + \Delta t/2$ we find

$$\mathbf{F}^+ - \mathbf{F} = \Delta t\, \mathbf{R}(\mathbf{F}(t + \Delta t/2), t + \Delta t/2) \tag{148}$$

where we introduced the shorthand notation $\mathbf{F}^+ = \mathbf{F}(t + \Delta t)$ and $\mathbf{F} = \mathbf{F}(t)$. On the right hand side of this equation, $\mathbf{R}$ in general contains differential operators and may be nonlinear in $\mathbf{F}(t, z)$. To locally linearize it around $\mathbf{F}(t + \Delta t/2)$ we use a Taylor expansion

$$\begin{aligned} \mathbf{R}(\mathbf{F}(t + \Delta t/2), t + \Delta t/2) &\approx \mathbf{R}(\mathbf{F}, t + \Delta t/2) + \tfrac{1}{2}\left[\partial_{\mathbf{F}} \mathbf{R}(\mathbf{F}(t), t)\right] \cdot (\mathbf{F}^+ - \mathbf{F}) \\ &= \tfrac{1}{2}\mathbf{A} \cdot (\mathbf{F}^+ + \mathbf{F}) + \mathbf{G}(\mathbf{F}, t + \Delta t/2) \end{aligned} \tag{149}$$

where the operator $\mathbf{A}$ is defined as the Jacobian matrix

$$\mathbf{A} = \partial_{\mathbf{F}} \mathbf{R}(\mathbf{F}, t) \tag{150}$$

and the vector $\mathbf{G}$ is

$$\mathbf{G} = \mathbf{R}(\mathbf{F}, t + \Delta t/2) - \mathbf{A} \cdot \mathbf{F} \tag{151}$$

Notice that since $\partial_{\mathbf{F}} \mathbf{R}$ in (149) is multiplied by $\mathbf{F}^+ - \mathbf{F} \propto \Delta t$, we can to first order in Δt evaluate $\mathbf{A}$ at t, but the $\mathbf{R}$ in $\mathbf{G}$ must be evaluated at $t + \Delta t/2$.

In our example we have $\mathbf{R} = (-c^2 \partial_z B - \varepsilon_0^{-1} j, -\partial_z E)^T$, which is linear in $\mathbf{F}$. We then find

$$\mathbf{A} = (\partial_E, \partial_B)\mathbf{R} = \begin{pmatrix} 0 & -c^2 \partial_z \\ -\partial_z & 0 \end{pmatrix} \tag{152}$$

and the vector $\mathbf{G}$ is

$$\mathbf{G} = (-\varepsilon_0^{-1} j(t + \Delta t/2),\ 0)^T \tag{153}$$

The locally linearized version of the general equation (147) can by using (148) and (149) be written

$$\left(I - \frac{\Delta t}{2}\mathbf{A}\right) \cdot \mathbf{F}^+ = \left(I + \frac{\Delta t}{2}\mathbf{A}\right) \cdot \mathbf{F} + \Delta t\, \mathbf{G}(\mathbf{F}, t + \Delta t/2) \tag{154}$$

12.2 Space discretization

We now continue to discretize the equations in space by introducing a uniform grid $z_n = n\Delta z$, $0 \le n < N$. Notice that n here is used as an index on the spatial grid, in order to be consistent with the application to auroral electron acceleration. The fields are then represented by a vector F of length N, where each element $\mathbf{F}_n$ is itself a vector. (In our example, $\mathbf{F}_n = (E(z_n), B(z_n))^T$.)

The derivative of a scalar field such as E is approximated by a centered finite difference

$$\partial_z E|_{z_n} \approx \frac{E(z_{n+1}) - E(z_{n-1})}{2\Delta z} \tag{155}$$

Considering $E = (E(z_0), E(z_1), \ldots, E(z_{N-1}))^T$ as a vector, the derivative ∂_z of the scalar function $E(z)$ may be represented by a $N \times N$ matrix operator

$$\partial_z E(z) \to \frac{1}{2\Delta z} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \ldots \\ -1 & 0 & 1 & 0 & 0 & \ldots \\ 0 & -1 & 0 & 1 & 0 & \ldots \\ 0 & 0 & -1 & 0 & 1 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} E_0 \\ E_1 \\ E_2 \\ E_3 \\ \ldots \\ E_{N-1} \end{bmatrix} \tag{156}$$

Since the operator $\mathbf{A}$ defined by equation (152) contains ∂_z its action on F is described by an equation related to (156), but with the scalars E_n replaced by the vectors $\mathbf{F}_n$ and the elements of the $N \times N$ matrix replaced by small 2×2 matrices. Comparing with (152) we see that the number 1 is replaced by

$$\mathbf{U} = \begin{pmatrix} 0 & -c^2 \\ -1 & 0 \end{pmatrix} \tag{157}$$

and -1 by

$$\mathbf{L} = -\mathbf{U} = \begin{pmatrix} 0 & c^2 \\ 1 & 0 \end{pmatrix} \tag{158}$$

Writing out the components of $A \cdot F$, where A is an $N \times N$ matrix we find

$$A \cdot F = \frac{1}{2\Delta z} \begin{bmatrix} 0 & \mathbf{U} & 0 & 0 & 0 & \ldots \\ \mathbf{L} & 0 & \mathbf{U} & 0 & 0 & \ldots \\ 0 & \mathbf{L} & 0 & \mathbf{U} & 0 & \ldots \\ 0 & 0 & \mathbf{L} & 0 & \mathbf{U} & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & 0 & 0 & 0 & \mathbf{L} & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{F}_0 \\ \mathbf{F}_1 \\ \mathbf{F}_2 \\ \mathbf{F}_3 \\ \ldots \\ \mathbf{F}_{N-1} \end{bmatrix} \tag{159}$$

If we in the same way write out the components of the matrix on the left hand side of equation (154) after discretization in space we find

$$(I - \frac{\Delta t}{2} A) \cdot F^+ = \begin{bmatrix} \mathbf{D} & -\frac{\Delta t}{4\Delta z}\mathbf{U} & 0 & 0 & 0 & \ldots \\ -\frac{\Delta t}{4\Delta z}\mathbf{L} & \mathbf{D} & -\frac{\Delta t}{4\Delta z}\mathbf{U} & 0 & 0 & \ldots \\ 0 & -\frac{\Delta t}{4\Delta z}\mathbf{L} & \mathbf{D} & -\frac{\Delta t}{4\Delta z}\mathbf{U} & 0 & \ldots \\ 0 & 0 & -\frac{\Delta t}{4\Delta z}\mathbf{L} & \mathbf{D} & -\frac{\Delta t}{4\Delta z}\mathbf{U} & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & 0 & 0 & 0 & -\frac{\Delta t}{4\Delta z}\mathbf{L} & \mathbf{D} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{F}_0^+ \\ \mathbf{F}_1^+ \\ \mathbf{F}_2^+ \\ \mathbf{F}_3^+ \\ \ldots \\ \mathbf{F}_{N-1}^+ \end{bmatrix} \tag{160}$$

In our simple example $\mathbf{D} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, but this is not always the case. Furthermore, the matrices $\mathbf{L}$, $\mathbf{D}$, and $\mathbf{U}$ need in general not be the same on each row of the large matrix. If

they are different we may use an index to denote the $\mathbf{D}_n$, $\mathbf{L}_n$, and $\mathbf{U}_n$ matrices on the n-th row of the large $N \times N$ matrix. A typical row in the set of equations (154) then has the form

$$\begin{aligned} -\frac{\Delta t}{4\Delta z}\mathbf{L}_n \cdot \mathbf{F}^+_{n-1} + \mathbf{D}_n \cdot \mathbf{F}^+_n - \frac{\Delta t}{4\Delta z}\mathbf{U}_n \cdot \mathbf{F}^+_{n+1} &= \\ \frac{\Delta t}{4\Delta z}\mathbf{L}_n \cdot \mathbf{F}_{n-1} + \mathbf{D}_n \cdot \mathbf{F}_n + \frac{\Delta t}{4\Delta z}\mathbf{U}_n \cdot \mathbf{F}_{n+1} &+ \Delta t\, \mathbf{G}_n(t + \Delta t/2) \end{aligned} \tag{161}$$

At each time step we must determine $\mathbf{F}^+$ by solving a set of N coupled equations such as (161). It is then important that $(I - \frac{\Delta t}{2}A)$ is represented by a tridiagonal matrix. The computer time required to solve a set of N equations generally increases as N^2, but a tridiagonal system can be solved in a time $\propto N$. The solution of tridiagonal systems with scalar coefficients is discussed in most books on numerical methods, such as *Numerical Recipes*. Similar methods can be used for block-tridiagonal systems like ours, but divisions must of course be evaluated as multiplication by the inverse of matrices such as $\mathbf{L}$. However, these matrices are small and can quickly be inverted. The number of operations per time step is then not too much greater than in an explicit algorithm, and an implicit algorithm may be orders of magnitude faster if we can use a much larger Δt.

In an explicit algorithm the update from $\mathbf{F}_n$ to $\mathbf{F}^+_n$ is local in the sense that it depends only on the fields at a few nearby points, such as $\mathbf{F}_{n-1}$, $\mathbf{F}_n$, and $\mathbf{F}_{n+1}$. A consequence of this is that information about changes in $\mathbf{F}(z)$ does not propagate faster than a few times $\Delta z/\Delta t$. In a way, this explains why an explicit algorithm cannot describe the electromagnetic waves in our example unless $\Delta z/\Delta t > c$. The characteristic feature of an implicit method is that the updates are global. At each time step we solve a set of coupled equations, which ensures that the spatial variation of $\mathbf{F}^+$ satisfies the differential equations. Information about changes in $\mathbf{F}(z)$ is distributed to the entire system at each time step, and the presence of high-velocity waves does not cause problems.

12.3 Convergence

Although an implicit algorithm may be stable for arbitrarily large time step Δt, there are in practice limits on the time steps that can be used. One limit is set by the convergence of the algorithm. To illustrate the concept of convergence we consider the model equation

$$\partial_t v + \gamma v = \frac{F(t)}{m} \tag{162}$$

which describes the motion of a particle subject to friction and a prescribed driving force $F(t)$. If we for simplicity assume that $F(t) = 0$ for $t < 0$ and $F(t) = ma$ where a is a constant for $t > 0$, the solution of equation (162) is

$$v(t) = \frac{a}{\gamma}\left(1 - e^{-\gamma t}\right) \tag{163}$$

Discretizing equation (162) in the same way as (148) we find

$$(1 + \gamma\Delta t/2)v^+ = (1 - \gamma\Delta t/2)v + a\Delta t \tag{164}$$

Starting from $t = 0$ we can by hand compute the first few steps to find $v^0 = 0$, $v^1 = a\Delta t/(1 + \gamma\Delta t/2)$,

$$v^2 = \frac{a\Delta t}{1 + \gamma\Delta t/2}\left[1 + \frac{1 - \gamma\Delta t/2}{1 + \gamma\Delta t/2}\right]$$

and by simple induction we find that

$$v^n = \frac{a\Delta t}{1 + \gamma\Delta t/2}\sum_{i=0}^{n-1}\left(\frac{1 - \gamma\Delta t/2}{1 + \gamma\Delta t/2}\right)^i \tag{165}$$

Using that the partial sum of a geometric series is

$$\sum_{i=0}^{n-1} x^i = \frac{1 - x^n}{1 - x}$$

we can rewrite the velocity after n time steps as

$$v^n = \frac{a\Delta t \quad \left[1 - \left(\frac{1-\gamma\Delta t/2}{1+\gamma\Delta t/2}\right)^n\right]}{(1 + \gamma\Delta t/2)\left[1 - \frac{1-\gamma\Delta t/2}{1+\gamma\Delta t/2}\right]} = \frac{a}{\gamma}\left[1 - \left(\frac{1 - \gamma\Delta t/2}{1 + \gamma\Delta t/2}\right)^n\right] \tag{166}$$

For sufficiently small $\Delta t \ll 2/\gamma$, the numerical solution v^n will approach the analytical solution (163), but since we use an implicit method we are mainly interested in larger Δt. We see that when $\Delta t = 2/\gamma$ the stationary value $v = a/\gamma$ will be reached in a single time step, but when $\Delta t > 2/\gamma$ the behavior of the discrete solution is more complicated. Figure 24 illustrates the case $\gamma = a = 1$ and $\Delta t = 10$, which gives $\gamma\Delta t/2 = 5$ and $(1-\gamma\Delta t/2)/(1+\gamma\Delta t/2) = -2/3$. We see that the numerical solution in the first step is too large, $v^1 = 1\frac{2}{3}\, a/\gamma$, and then v^n oscillates several times with decreasing amplitude until it converges to the proper stationary value.

From the rather poor behaviour if the solution in Figure 24 we might get the impression that it is necessary to have $\gamma\Delta t \sim 1$ also when using implicit methods. However, the slow convergence of this numerical solution is not caused primarily by the large $\gamma\Delta t$ but by the rapid increase of $F(t)$ in a time shorter than Δt. If we replace the sudden turn on of the force at $t = 0$ by a smoother driving force such as

$$F(t) = ma\left(1 - e^{-\Gamma t}\right) \tag{167}$$

the analytical solution can be written

$$v(t) = \frac{a}{\gamma}\left(1 - \frac{\gamma e^{-\Gamma t} - \Gamma e^{-\gamma t}}{\gamma - \Gamma}\right) \sim \frac{a}{\gamma}\left(1 - e^{-\Gamma t}\right) \tag{168}$$

where we in the final step assumed $\gamma \gg \Gamma$. Choosing $\Gamma = 0.1$ we can marginally reslove the variation of F with $\Delta t = 10$, since $\Gamma\Delta t = 1$. With these parameters we obtain a reasonable, though not very accurate, numerical solution with our implicit method, as shown in Figure 25. In practice we would like to have $\Gamma\Delta t \sim 0.1$, particularly if $\gamma\Delta t$ is very large.

The above example illustrates that implicit algorithms are particularly usful in problems involving very different time scales. When we are interested in slow (Γ) processes, we can use a long time step even if the system also contains very fast (γ) processes. Recall that in explicit methods, Δt is limited by the fastes time scales in the system at hand.

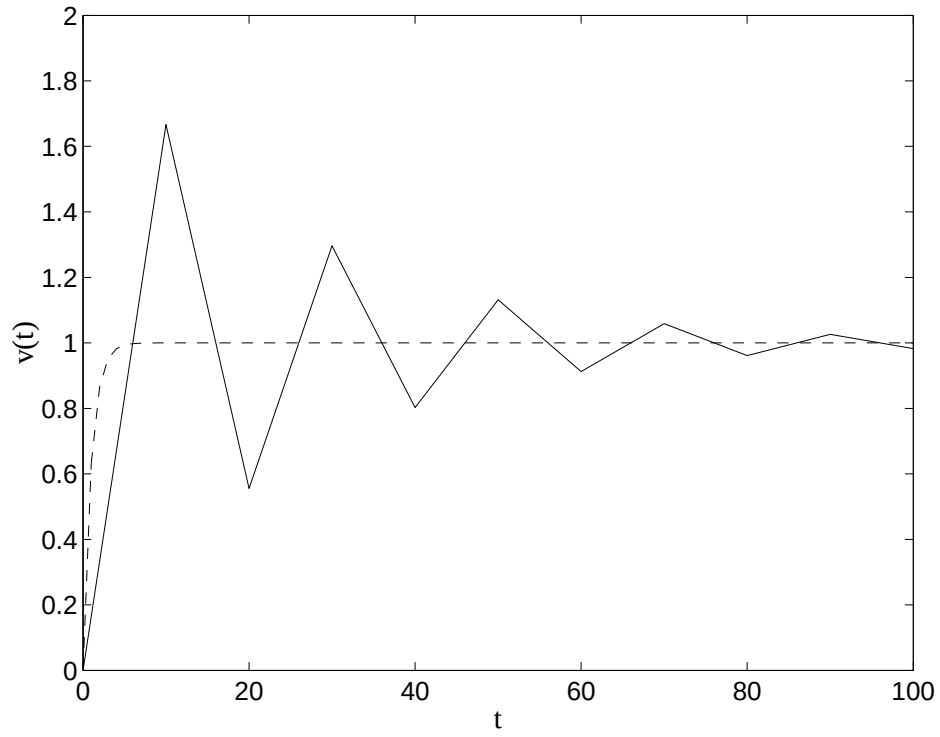


Figure 24: Illustration of how the numerical solution (full line) converges to the analytical solution (dashed) when $a = 1$, $\gamma = 1$, and $\Delta t = 10$.

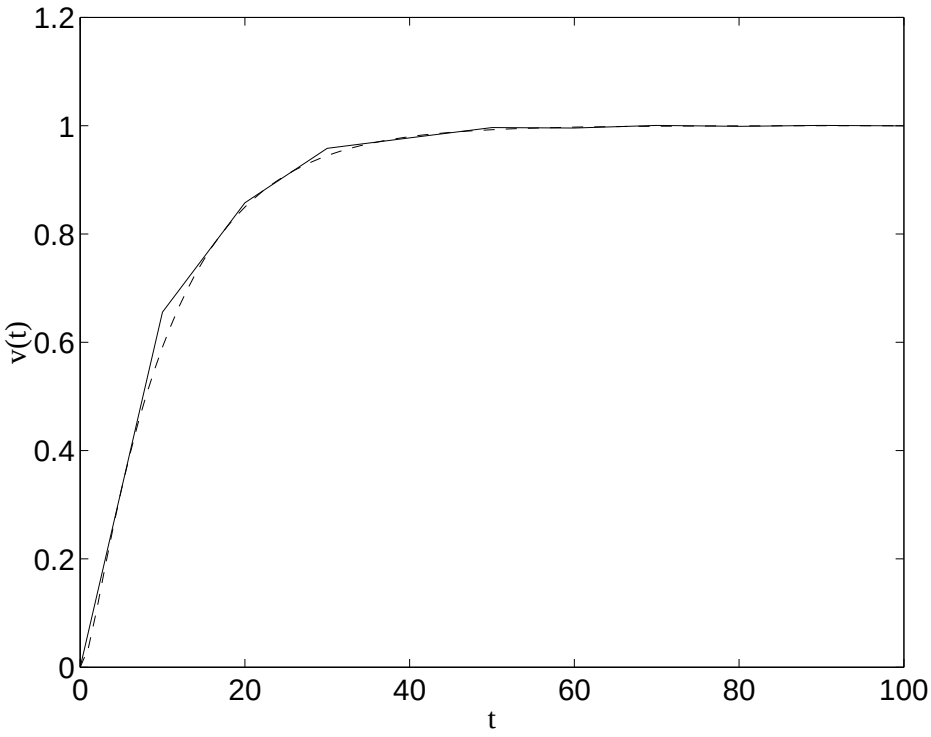


Figure 25: Illustration of how the numerical solution (full line) converges to the analytical solution (dashed) when $\Gamma = 0.1$, $a = 1$, $\gamma = 1$, and $\Delta t = 10$.

12.4 Boundary conditions

Sometimes it is possible to use slightly inaccurate or even inconsistent boundary conditions with an explicit method, since information about the boundary does not spread into the interior of the simulation region. When we use an implicit method the boundary conditions will affect the entire region at once, and it is essential that they are consistent and properly describe the desired physical situation.

Boundary conditions are specified by modifying the first and last rows of the matrix $(I-\frac{\Delta t}{2}A)$ in (160), or equivalently my modifying equation (161) for $n = 0$ and $n = N - 1$. When discussing boundary conditions it is convenient to introduce $\mathbf{F}_{-1}$ and $\mathbf{F}_N$, the values of $\mathbf{F}$ outside the boundaries. As it stands, $(I - \frac{\Delta t}{2}A)$ in (160) corresponds to $\mathbf{F}_{-1} = \mathbf{F}_N = 0$, but this is rarely what we want. Typically, the components of $\mathbf{F}$ (E and B in our example) will satisfy different boundary conditions, determined by the physical situation we intend to simulate. Let us look in detail at the equations for $\mathbf{F}_0$ found from (161)

$$\begin{aligned} -\frac{\Delta t}{4\Delta z}L_{01}B^+_{-1} + D_{00}E^+_0 - \frac{\Delta t}{4\Delta z}U_{01}B^+_1 &= R_{E0} \\ -\frac{\Delta t}{4\Delta z}L_{10}E^+_{-1} + D_{11}B^+_0 - \frac{\Delta t}{4\Delta z}U_{10}E^+_1 &= R_{B0} \end{aligned} \tag{169}$$

where the right hand side has been abbreviated in an obvious way. Notice that the subscripts on L, D, and U indicate components of the 2×2 matrices but the subscripts on E and B refer to the space index n.

As an example, assume that we want $E = 0$ and $\partial_z B = 0$ at the $z = 0$ boundary. We can ensure $E^+_0 = 0$ by specifying $E^+_{-1} = -E^+_1$, and $B^+_{-1} = B^+_1$ makes the derivative of B vanish at the boundary. Using these relations to eliminate E^+_{-1} and B^+_{-1} from equations (169) we find

$$\begin{aligned} D_{00}E^+_0 - \frac{\Delta t}{4\Delta z}(U_{01} + L_{01})B^+_1 &= R_{E0} \\ D_{11}B^+_0 - \frac{\Delta t}{4\Delta z}(U_{10} - L_{10})E^+_1 &= R_{B0} \end{aligned} \tag{170}$$

These boundary conditions can be implemented by replacing $\mathbf{U}_0$ on the first row of A by

$$\mathbf{U'}_0 = \begin{pmatrix} U_{00} & U_{01} + L_{01} \\ U_{10} - L_{10} & U_{11} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ -2 & 0 \end{pmatrix} \tag{171}$$

12.5 Two dimensional problems and factorization

A pointed out in section 12.2, it is essential for the efficiency of the implicit method that the operator $(I - \frac{\Delta t}{2}A)$ can be represented by a tridiagonal matrix. While this in general is the case in one dimensional problems, it seems at first sight as if two-dimensional implicit simulations should be almost unmanageable. Let us consider Ampere's and Faraday's laws in two dimensions by assuming $\mathbf{E_x} = E(t, x, z)\hat{\mathbf{x}}$, $\mathbf{E_z} = E(t, x, z)\hat{\mathbf{x}}$, $\mathbf{j} = j(t, x, z)\hat{\mathbf{x}}$, and $\mathbf{B} = B(t, x, z)\hat{\mathbf{y}}$. We then have the equations

$$\begin{aligned} \partial_t E_x &= -c^2\partial_z B - \varepsilon_0^{-1} j \\ \partial_t E_z &= c^2\partial_x B \\ \partial_t B &= \partial_x E_z - \partial_z E_x \end{aligned} \tag{172}$$

If we introduce a grid by $x_m = m\Delta x$, $0 \le m < M$ and $z_n = n\Delta z$, $0 \le n < N$ we find that our problem can still be written in the form (154), but the matrix representing $(I - \frac{\Delta t}{2}A)$ will no longer be tridiagonal. Hence, it seems that we would have to solve a set of $M \times N$ coupled (vector) equations, and the time required for a time step would grow in proportion to $(M \times N)^2$. This would make implicit methods almost useless in more than one dimension.

Fortunately, these difficulties can rather easily be handled, at least as long as there are no mixed spatial derivatives such as $\partial_x \partial_z$. We can then split the operator $\mathbf{A}$ into two parts as

$$\mathbf{A} = \mathbf{X} + \mathbf{Z} \tag{173}$$

where $\mathbf{X}$ contains only ∂_x and $\mathbf{Z}$ contains only ∂_z. This decomposition is not unique, and elements that contain neither ∂_x nor ∂_z should be placed so that the product $\mathbf{X} \cdot \mathbf{Z}$ becomes as small as possible. In our example (172) there is no ambiguity since each element in

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & -c^2\partial_z \\ 0 & 0 & c^2\partial_x \\ -\partial_z & \partial_x & 0 \end{pmatrix} \tag{174}$$

contains a derivative and the decomposition must be done as

$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & c^2\partial_x \\ 0 & \partial_x & 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{Z} = \begin{pmatrix} 0 & 0 & -c^2\partial_z \\ 0 & 0 & 0 \\ -\partial_z & 0 & 0 \end{pmatrix} \tag{175}$$

After this separation of ∂_x and ∂_z we find

$$I - \frac{\Delta t}{2}(\mathbf{X} + \mathbf{Z}) = (I - \frac{\Delta t}{2}\mathbf{X}) \cdot (I - \frac{\Delta t}{2}\mathbf{Z}) - \frac{\Delta t^2}{4}\mathbf{X} \cdot \mathbf{Z} \tag{176}$$

and by using this, equation (154) may be rewritten as

$$\begin{aligned} \left(I - \frac{\Delta t}{2}\mathbf{X}\right) \cdot \left(I - \frac{\Delta t}{2}\mathbf{Z}\right) \cdot \mathbf{F}^+ \; &= \\ \left(I + \frac{\Delta t}{2}\mathbf{X}\right) \cdot \left(I + \frac{\Delta t}{2}\mathbf{Z}\right) \cdot \mathbf{F} \quad &+ \quad \frac{\Delta t^2}{4}\mathbf{X} \cdot \mathbf{Z} \cdot (\mathbf{F}^+ - \mathbf{F}) + \Delta t\, \mathbf{G}(\mathbf{F}, t + \Delta t/2) \end{aligned} \tag{177}$$

Since $(\mathbf{F}^+ - \mathbf{F}) \propto \Delta t$, the term $\frac{\Delta t^2}{4}\mathbf{X} \cdot \mathbf{Z} \cdot (\mathbf{F}^+ - \mathbf{F})$ is of third order in Δt and can be neglected. If we now introduce $\mathbf{F}^* = \left(I - \frac{\Delta t}{2}\mathbf{Z}\right) \cdot \mathbf{F}^+$ we can split (177) into two equations as

$$\left(I - \frac{\Delta t}{2}\mathbf{X}\right) \cdot \mathbf{F}^* = \left(I + \frac{\Delta t}{2}\mathbf{X}\right) \cdot \left(I + \frac{\Delta t}{2}\mathbf{Z}\right) \cdot \mathbf{F} + \Delta t\, \mathbf{G}(\mathbf{F}, t + \Delta t/2) \tag{178}$$

$$\left(I - \frac{\Delta t}{2}\mathbf{Z}\right) \cdot \mathbf{F}^+ = \mathbf{F}^* \tag{179}$$

These equations can now be discretized on our two-dimensional grid. Equation (178) can for each z_n be treated in the same way as (154). Since $(I - \frac{\Delta t}{2}\mathbf{X})$ contains only ∂_x it corresponds to a tridiagonal matrix. Equation (178) then represents N independent tridiagonal systems, each of order M, and can be solved for $\mathbf{F}^*$ in a time proportional to $M \times N$. Having found $\mathbf{F}^*$ we turn to (179) which represents M tridiagonal systems of order N, and can be solved for $\mathbf{F}^+$ in a time that also is proportional to $M \times N$. We have then completed an update from $\mathbf{F}(t)$ to $\mathbf{F}(t + \Delta t)$ via the physically meaningless intermediate value $\mathbf{F}^*$.

At first sight the algorithm outlined above may seem unnecessarily complicated. Is it really worth while to do all this algebra? Well, assuming $M \sim N \sim 100$ as a typical grid size, we see that by factoring and splitting the $(I - \frac{\Delta t}{2}\mathbf{A})$ operator we have gained almost a factor $M \times N \sim 10^4$ in speed, and a two-dimensional simulation that would have taken a year can now be done in less than an hour.

A The computer simulation program

The general structure of the computer programs used in this course is the same, whatever the application.. Therefore, to simplify the included exercises, both for the students and for the supervisor, it is recommended that the program structure given below is followed as far as possible. The algorithm is written in a pseudo language and should be easy to translate into C or FORTRAN.

A.1 The main program

The fairly simple structure of the main program can be described by the following schematic algorithm (function arguments are not specified)

```
Beginning of main program
  Parameter definitions
  Declaration of variables

  Initialization (init)

  for i=1,ITMAX
    Advance one time step (step)
    Every NSNAP:th step
      Produce snapshots (snapshots)
    Store things of importance (store)
  end for

  Post processing, diagnostics (diagnostic_plots)
End of main
```

A.2 Important parts of the main program

init
All dynamic initialization is gathered in one module, e.g., initialization of graphics, loading of initial conditions, allocation of memory (C language), etc. Programs based on the leap-frog algorithm must be initialized with half an Euler step backwards to obtain $v^{-1/2}$. Sometimes it is useful to store the initial configuration.

The main loop
In the main loop the motion of the system is followed during the number of time steps prescribed by the parameter ITMAX.

step
This module is used to update the position and velocity of each particle, using one of the integration schemes described in section 3. The routine can be split into two parts: **accel**

computing the acceleration on each particle and **move** that handles the updates. **accel** is normally written as a separate module, which can be further divided into one part handling the forces, and one part computing the resulting acceleration on each particle. Although it is often useful to structure the **step** module like this, it should be noticed that some integration schemes requires a slightly different program structure. Take for for instance the Verlet algorithm, where the velocity update is done in two steps.

Snapshots
To get an overview of how the simulation is proceeding, it is often useful to produce diagnostic plots at regular intervals. These plots can include, e.g., particle positions, the kinetic, potential and total energy of the system.

Store
This module is used to store valuable information at regular intervals, and may include some analysis. The stored information comprises information that normally is not displayed during the course of the simulation, but which is processed when the main loop or the whole simulation program is done.

diagnostic_plots
Post processing of data could be done directly after the main loop in a diagnostic_plots module. Alternatively, this module may be excluded from the main program and put in a separate analysis package.

References

Armstrong T. P., R. C. Harding, G. Knorr, and D. Montgomery, *Solution of Vlasov's equation by transform methods*, in *Methods in Computational Physics*, ed. B. Alder, S. Fernbach, and M. Rotenberg (Academic Press, London, 1970).

Birdsall C. K. and A. B. Langdon, *Plasma Physics via Computer Simulation* (McGraw-Hill, New York, 1985).

Chen F. F., *Introduction to Plasma Physics* (Plenum Press, New York, 1974).

Gockenbach M. S., *Partial Differential Equations: Analytical and Numerical Methods* (SIAM, Philadelphia, 2002).

Gould H. and J. Tobochnik, *An Introduction to Computer Simulation Methods Applications to Physical Systems*, Part 1 (Addison-Wesley, Readin, Mass., 1988).

Garcia A. L., *Numerical Methods for Physics* (Prentice Hall, Englewood Cliffs, NJ, 1994).

Haile J. M., *Molecular Dynamics Simulation: Elementary Methods* (Wiley, New York, 1992).

Heerman D. H., *Computer Simulation Methods in Theoretical Physics* (Springer-Verlag, Berlin, 1986).

Hockney R. W. and J. W. Eastwood, *Computer Simulation Using Particles* (Adam Hilger, Bristol, 1988).

Moon F. C., *Chaotic and Fractal Dynamics: An Introduction for Applied Scientists and Engineers* (Wiley, New York, 1992).

Ott E., *Chaos in Dynamical Systems* (Cambridge University Press, Cambridge, 1993).

Potter D., *Computational Physics* (Wiley, New York, 1973).

Press W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. (Cambridge University Press, Cambridge, 1992).

Tajima T., *Computational Plasma Physics: With Applications to Fusion and Astrophysics* (Addison-Wesley, Redwood City, Ca., 1989).

Tritton D. J., *Physical Fluid Dynamics*, 2nd ed. (Clarendon Press, Oxford, 1988).

Verlet L., *Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecule*, Physical Review, vol. 159, pp. 98–103 (1967).

Winske D., *Hybrid Simulation Codes with Application to Shocks and Upstream Waves*, Space Science Reviews, vol. 42, pp. 53–56 (1985).

Index