5 The particle-mesh technique

Direct summation of particle-particle forces scales with $\mathcal{O}(N^2)$. An important approach to accelerate the force calculation for an N-body system lies in the use of an auxiliary mesh, which allows very efficient summation in the reciprocal space.

The slowly converging direct sum is split into two terms, a direct summation over short-range interactions and a summation in reciprocal space of long-range interactions.

$$V = V^{\text{short}} + V^{\text{long}} \tag{5.1}$$

Among the methods to efficiently compute V^{long}, we will first concentrate on the so-called particle-mesh (PM) technique (**??**) that was originally pioneered in plasma physics (**?**). In the last Chapter, we will introduce the particle-mesh Ewald technique, a method that originates from Ewald summation (**?**) and is often superior to PM in case of periodic boundary conditions. PM techniques, with or without Ewald, are widely used and have been applied in simulations of molecules, galaxies or, among the first applications, of plasmas. Both procedures in general involve four steps:

1. Construction of a density field ρ on a suitable mesh.

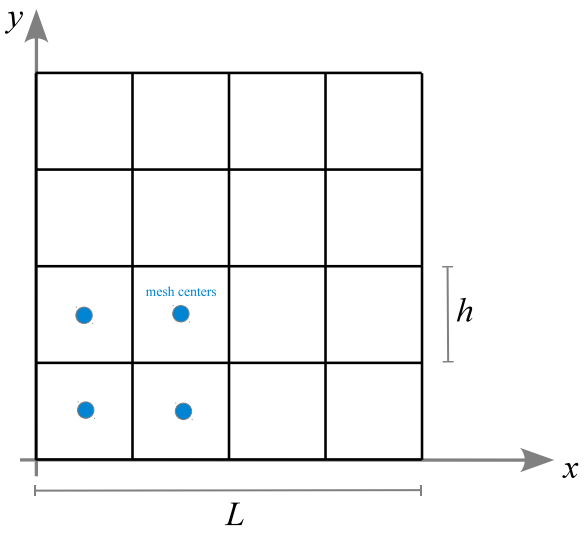
2. Computation of the potential on the mesh by solving the Poisson equation.

3. Calculation of the force field from the potential.

4. Calculation of the forces at the original particle positions.

We shall now discuss these four steps in turn.

5.1 Mass/charge assignment



We want to put N particles with mass m_i and coordinates $\mathbf{x}_i$ $(i = 1, 2, \ldots, N)$ onto a mesh with uniform spacing $h = L/N_\mathrm{g}$. For simplicity, we will assume a cubical calculational domain with extension L and a number of N_g grid cells per dimension. Let $\{\mathbf{x_p}\}$ denote the set of discrete cell-centers, with $\mathbf{p} = (p_x, p_y, p_z)$ being a suitable integer index $(0 \leq p_{x,y,z} < N_g)$. Note that one may equally well identify the $\{\mathbf{x_p}\}$ with the lower left corner of a mesh cell, if this is more practical.

We associate a shape function $S(\mathbf{x})$ with each particle, normalized according to

$$\int S(\mathbf{x})\,\mathrm{d}\mathbf{x} = 1. \tag{5.2}$$

To each mesh-cell, we then assign the fraction $W_\mathbf{p}(\mathbf{x}_i)$ of particle i's mass that falls into the cell indexed by $\mathbf{p}$. This is given by the overlap of the mesh cell with the shape function, namely:

$$W_\mathbf{p}(\mathbf{x}_i) = \int_{\mathbf{x_p}-\frac{\mathbf{h}}{2}}^{\mathbf{x_p}+\frac{\mathbf{h}}{2}} S(\mathbf{x}_i - \mathbf{x})\,\mathrm{d}\mathbf{x} \tag{5.3}$$

The integration extends here over the cubical cell $\mathbf{p}$. By introducing the top-hat function

$$\Pi(\mathbf{x}) = \begin{cases} 1 & \text{for } |\mathbf{x}| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

we can extend the integration boundaries to all space and write instead:

$$W_\mathbf{p}(\mathbf{x}_i) = \int \Pi\left(\frac{\mathbf{x} - \mathbf{x_p}}{h}\right) S(\mathbf{x}_i - \mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{5.5}$$

Note that this also shows that the assignment function W is a convolution of Π with S. The full density in grid cell $\mathbf{p}$ is then given by

$$\rho_{\mathbf{p}} = \frac{1}{h^3} \sum_{i=1}^{N} m_i W_{\mathbf{p}}(\mathbf{x}_i). \tag{5.6}$$

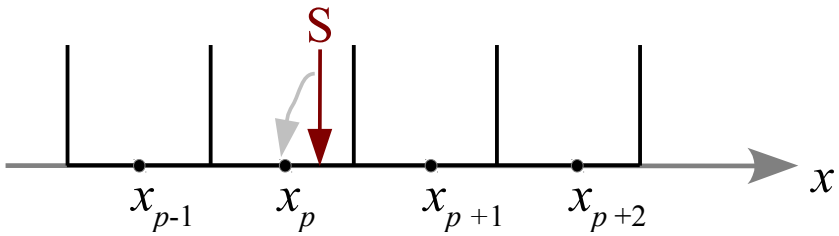
These general formula evidently depend on the specific choice one makes for the shape function $S(\mathbf{x})$. We discuss below a few of the most commonly employed low-order assignment schemes.

5.1.1 Nearest grid point (NGP) assignment

The simplest possible choice for S is a Dirac δ-function. One then gets:

$$W_{\mathbf{p}}(\mathbf{x}_i) = \int \Pi\left(\frac{\mathbf{x} - \mathbf{x_p}}{h}\right) \delta(\mathbf{x}_i - \mathbf{x})\, \mathrm{d}\mathbf{x} = \Pi\left(\frac{\mathbf{x}_i - \mathbf{x_p}}{h}\right). \tag{5.7}$$

In other words, this means that $W_{\mathbf{p}}$ is either 1 (if the coordinate of particle i lies inside the cell), or otherwise it is zero. Consequently, the mass of particle i is fully assigned to exactly one cell – the nearest grid point. The sketch below illustrates this further.



5.1.2 Clouds-in-cell (CIC) assignment

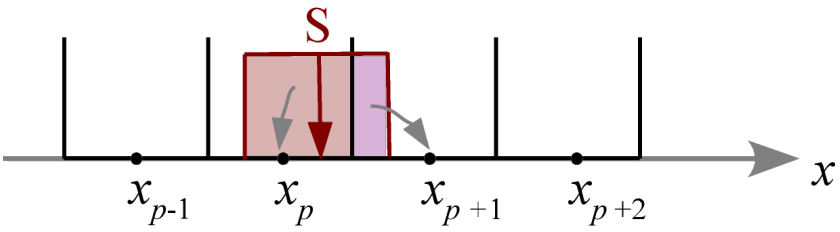
Here one adopts as shape function

$$S(\mathbf{x}) = \frac{1}{h^3} \Pi\left(\frac{\mathbf{x}}{h}\right), \tag{5.8}$$

which is the same cubical 'cloud' shape as that of individual mesh cells. The assignment function is

$$W_{\mathbf{p}}(\mathbf{x}_i) = \int \Pi\left(\frac{\mathbf{x} - \mathbf{x_p}}{h}\right) \frac{1}{h^3} \Pi\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right) \mathrm{d}\mathbf{x}, \tag{5.9}$$

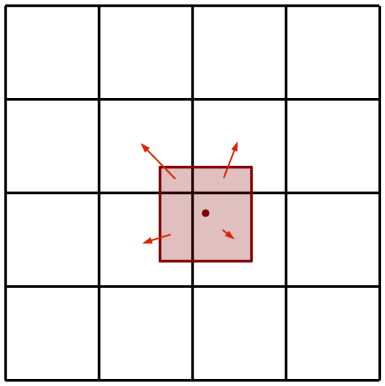
which only has a non-zero (and then constant) integrand if the cubes centred on $\mathbf{x}_i$ and $\mathbf{x_p}$ overlap. How can this overlap be calculated? The 1D sketch below can help to make this clear.



Recall that for one of the dimensions we have $x_p = (p_x+1/2)h$ for $p \in \{0, 1, 2, \ldots, N-1\}$. for a given particle coordinate x_i we may first calculate a 'floating point index' by inverting this relation, yielding $p_f = x_i/h - 1/2$. The index of the left cell of the two cells with some overlap is then given by $p = \lfloor p_f \rfloor$, where the brackets denote the integer floor, i.e. the largest integer not larger than p_f. We may then further define $p^\star \equiv p_f - p$, which is a number between 0 and 1. From the sketch, we see that the length of the overlap of the particle's cloud with the cell p is $h - hp^\star$, hence the assignment function at cell p takes on the value $W_p = 1 - m^\star$ for this location of the particle, whereas the assignment function for the neighboring cell $p + 1$ will take on the value $W_{p+1} = m^\star$.

These considerations readily generalize to 2D and 3D. For example, in 2D, we first assign to the y_i-coordinate of point i a 'floating point index' $q_f = y_i/h - 1/2$. We can then use this to compute a cell index as the integer floor $q = \lfloor q_f \rfloor$, and a fractional contribution $q^\star = q_f - q$. We then obtain the following weights for the assignment of a particle's mass to the four cells its 'cloud' touches in 2D (as sketched):

$$W_{p,q} = (1 - p^\star)(1 - q^\star) \tag{5.10}$$
$$W_{p+1,q} = p^\star(1 - q^\star) \tag{5.11}$$
$$W_{p,q+1} = (1 - p^\star)q^\star \tag{5.12}$$
$$W_{p+1,q+1} = p^\star q^\star \tag{5.13}$$



In the corresponding 3D case, each particle contributes to the weight functions of 8 cells, or in other words, it is spread over 8 cells.

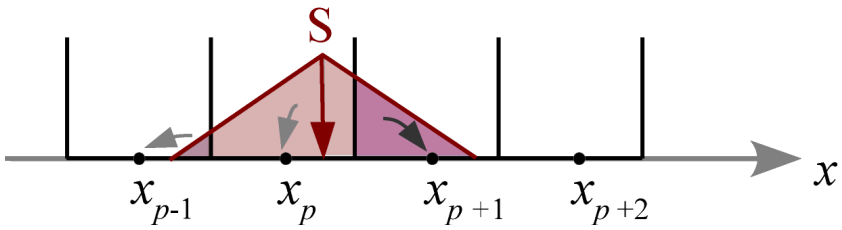
5.1.3 Triangular shaped clouds (TSC) assignment

One can construct a systematic sequence of ever higher-order shape functions by adding more convolutions with the top-hat kernel. For example, the next higher order (in 3D) is given by

$$W_{\mathbf{p}}(\mathbf{x}_i) = \int \Pi\left(\frac{\mathbf{x}-\mathbf{x_p}}{h}\right) \frac{1}{h^3}\Pi\left(\frac{\mathbf{x}_i-\mathbf{x}-\mathbf{x}'}{h}\right) \frac{1}{h^3}\Pi\left(\frac{\mathbf{x}'}{h}\right) \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}' \quad (5.14)$$

$$= \frac{1}{h^6}\int \Pi\left(\frac{\mathbf{x}-\mathbf{x_p}}{h}\right) \Pi\left(\frac{\mathbf{x}_i-\mathbf{x}}{h}\right) \Pi\left(\frac{\mathbf{x}'-\mathbf{x}}{h}\right) \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{x}'. \quad (5.15)$$

This still has a simple geometric interpretation. If one pictures the kernel shape as a triangle with total base length $2h$, then the fraction assigned to a certain cell is given by the area of overlap of this triangle with the cell of interest. The triangle will now in general touch 3 cells per dimension, making an evaluation correspondingly more expensive. In 3D, 27 cells are touched for every particle.



What's then the advantage of using TSC over CIC, if any? Or should one stick with NGP? The assignment schemes differ in the smoothness and differentiability of the reconstructed density field. In particular, for NGP, the assigned density and hence the resulting force jump discontinuously when a particle crosses a cell boundary. The resulting force law will then at best be piece-wise constant.

In contrast, the CIC scheme produces a force that is piece-wise linear and continuous, but has a first derivative that jumps. Here the information where a particle is inside a certain cell is not completely lost, unlike in NGP.

Finally, TSC is yet smoother, and also the first derivative of the force is continuous. Which of these schemes is the preferred choice is ultimately problem-dependent. In most cases, CIC and TSC are quite good options, providing sufficient accuracy with still reasonably small (and hence computationally cheap) assignment kernels. The latter get invariably bigger and bigger for higher-order assignment schemes, which not only is computationally ever more costly but also invokes additional communication overhead in certain parallelization schemes.

5.2 Solving for the gravitational potential

Once the density field is obtained, one would like to solve Poisson's equation

$$\nabla^2\Phi = 4\pi G\,\rho \quad (5.16)$$

Table 5.1: Commonly used shape functions.

Name	Cloud shape $S(x)$	# of cells used	assignment function shape
NGP	$\delta(x)$	1^d	Π
CIC	$\frac{1}{h^d}\Pi\left(\frac{\mathbf{x}}{h}\right)$	2^d	$\Pi \star \Pi$
TSC	$\frac{1}{h^d}\Pi\left(\frac{\mathbf{x}}{h}\right) \star \frac{1}{h^d}\Pi\left(\frac{\mathbf{x}}{h}\right)$	3^d	$\Pi \star \Pi \star \Pi$

and obtain the gravitational potential discretized on the same mesh. There are primarily two methods that are in widespread use for this:

First, there are Fourier-transform based methods which exploit the fact that the potential can be viewed as a convolution of a Green's function with the density field. In Fourier-space, one can then exploit the convolution theorem and cast the computationally expensive convolution into a cheap algebraic multiplication. Due to the importance of this approach, we will discuss it extensively in the next chapter.

Second, there are so-called iterative solvers for Poisson's equation which yield a solution directly in real-space. Simple versions of such iterative solvers use Jacobi or Gauss-Seidel iteration, more complicated ones employ a sophisticated multi-grid approach to speed up convergence. We shall discuss these methods in chapter 7.

5.3 Calculation of the forces

Let's assume for the moment that we already obtained the gravitational potential Φ on the mesh, with one of the methods mentioned above. We would then like to get the acceleration field from

$$\mathbf{a} = -\nabla\Phi. \tag{5.17}$$

One can achieve this by calculating a numerical derivative of the potential by *finite differencing*. For example, the simplest estimate of the force in the x-direction would be

$$a_x(i,j,k) = -\frac{\Phi(i+1,j,k) - \Phi(i-1,j,k)}{2h}, \tag{5.18}$$

where $\mathbf{p} = (i,j,k)$ is a cell index. The truncation error of this expression is $\mathcal{O}(h^2)$, hence the estimate of the derivative is second-order accurate.

Alternatively, one can use *larger stencils* to obtain a more accurate finite difference approximation of the derivative, at greater computational cost. For example, the 4-point expression

$$a_x(i,j,k) = -\frac{1}{2h}\left\{\frac{4}{3}\left[\Phi(i+1,j,k) - \Phi(i-1,j,k)\right] - \frac{1}{6}\left[\Phi(i+2,j,k) - \Phi(i-2,j,k)\right]\right\} \tag{5.19}$$

can be used, which has a truncation error of $\mathcal{O}(h^4)$ (which can be simply proven through Taylor expansion).

For the y- and z-dimensions, corresponding formulae where j or k are varied and the other cell coordinates are held fixed can be used. Whether a second- or fourth-order discretization formula is used depends again on the question which compromise between accuracy and speed one considers best for a given problem. In many collisionless systems, the residual truncation error of a second-order finite difference approximation of the force will be negligible compared to other errors inherent in the simulation scheme, hence the second-order formula would be sufficient in this case.

5.4 Interpolating from the mesh to the particles

Once we have the force field on a mesh, we are not yet fully done. We actually desire the forces at the particle coordinates of the N-body system, not at the coordinates of the mesh cells of our auxiliary computational grid. We are hence left with the problem of interpolating the forces from the mesh to the particle coordinates.

Recall that we defined the density field in terms of mass assignment functions, of the form

$$\rho_{\mathbf{p}} = \frac{1}{h^3} \sum_i W_{\mathbf{p}}(\mathbf{x}_i) = \frac{1}{h^3} \sum_i W(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}}). \tag{5.20}$$

Here we introduced in the last expression an alternative notation for the weight assignment function.

Assume that we have computed the acceleration field on the grid, $\{\mathbf{a}_{\mathbf{p}}\}$. It turns out to be very important to *use the same* assignment kernel as used in the density construction also for the force interpolation, i.e. the force at coordinate $\mathbf{x}$ for a mass m needs to be computed as

$$\mathbf{F}(\mathbf{x}) = m \sum_{\mathbf{p}} \mathbf{a}_{\mathbf{p}} W(\mathbf{x} - \mathbf{x}_{\mathbf{p}}), \tag{5.21}$$

where W denotes the assignment function used for computing the density field on the mesh. This requirement results from the desire to have a vanishing *self-force*, as well as pairwise antisymmetric forces between every particle pair. The self-force is the force that a particle would feel if just it alone would be present in the system. If numerically this force would be calculated with a non-zero value, the particle would accelerate all by itself, violating momentum conservation. Likewise, for two particles, we would like to have that the forces they mutually exert on each other are equal in magnitude and opposite in direction to each other, such that momentum conservation is manifest.

We now prove that using the same kernels for the mass assignment and force interpolation protects against these numerical artefacts. We start by noting that the acceleration field at a mesh point $\mathbf{p}$ is a linear response to the mass at another mesh point $\mathbf{p}'$ (this can be trivially seen when Fourier techniques are used to solve

the Poisson equation). We can hence express the field as

$$\mathbf{a}_{\mathbf{p}} = \sum_{\mathbf{p}'} \mathbf{d}(\mathbf{p}, \mathbf{p}')\, h^3 \rho_{\mathbf{p}'} \tag{5.22}$$

with a Green's function $\mathbf{d}(\mathbf{p}, \mathbf{p}')$. This vector-valued Green's function for the force is antisymmetric, i.e. it changes sign when the two points in the argument are swapped. Note that $h^3\rho_{\mathbf{p}'}$ is simply the mass contained in mesh cell $\mathbf{p}'$.

We can now calculate the self-force resulting from the density assignment and interpolation steps:

$$\begin{aligned}
\mathbf{F}_{\text{self}}(\mathbf{x}_i) &= m_i \sum_{\mathbf{p}} W(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}}) \mathbf{a}_{\mathbf{p}} && (5.23)\\
&= m_i \sum_{\mathbf{p}} W(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}}) \sum_{\mathbf{p}'} \mathbf{d}(\mathbf{p}, \mathbf{p}') h^3 \rho_{\mathbf{p}'} && (5.24)\\
&= m_i \sum_{\mathbf{p}} W(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}}) \sum_{\mathbf{p}'} \mathbf{d}(\mathbf{p}, \mathbf{p}') m_i W(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}'}) && (5.25)\\
&= m_i^2 \sum_{\mathbf{p},\mathbf{p}'} \mathbf{d}(\mathbf{p}, \mathbf{p}') W(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}}) W(\mathbf{x}_i - \mathbf{x}_{\mathbf{p}'}) && (5.26)\\
&= 0. && (5.27)
\end{aligned}$$

Here we have started out with the interpolation from the mesh-based acceleration field, and then inserted the expansion of the latter as as convolution over the density field of the mesh. Finally, we put in the density contribution created by the particle i at a mesh cell $\mathbf{p}'$. We then see that the double sum vanishes because of the antisymmetry of $\mathbf{d}$ and the symmetry of the kernel product under exchange of $\mathbf{p}$ and $\mathbf{p}'$. Note that this however only works because the kernels used for force interpolation and density assignment are actually equal – it would have not worked out if they would be different, which brings us back to the point emphasized above.

Now let's turn to the force antisymmetry. The force exerted on a particle 1 of mass m_1 at location $\mathbf{x}_1$ due to a particle 2 of mass m_2 at location $\mathbf{x}_2$ is given by

$$\begin{aligned}
\mathbf{F}_{12} &= m_1 \mathbf{a}(\mathbf{x}_1) = m_1 \sum_{\mathbf{p}} W(\mathbf{x}_1 - \mathbf{x}_{\mathbf{p}}) \mathbf{a}_{\mathbf{p}} && (5.28)\\
&= m_1 \sum_{\mathbf{p}} W(\mathbf{x}_1 - \mathbf{x}_{\mathbf{p}}) \sum_{\mathbf{p}'} \mathbf{d}(\mathbf{p}, \mathbf{p}') h^3 \rho_{\mathbf{p}'} && (5.29)\\
&= m_1 \sum_{\mathbf{p}} W(\mathbf{x}_1 - \mathbf{x}_{\mathbf{p}}) \sum_{\mathbf{p}'} \mathbf{d}(\mathbf{p}, \mathbf{p}') m_2 W(\mathbf{x}_2 - \mathbf{x}_{\mathbf{p}'}) && (5.30)\\
&= m_1 m_2 \sum_{\mathbf{p},\mathbf{p}'} \mathbf{d}(\mathbf{p}, \mathbf{p}') W(\mathbf{x}_1 - \mathbf{x}_{\mathbf{p}}) W(\mathbf{x}_2 - \mathbf{x}_{\mathbf{p}'}) && (5.31)
\end{aligned}$$

Likewise we obtain for the force experienced by particle 2 due to particle 1:

$$\mathbf{F}_{21} = m_1 m_2 \sum_{\mathbf{p}',\mathbf{p}} \mathbf{d}(\mathbf{p}, \mathbf{p}') W(\mathbf{x}_2 - \mathbf{x}_{\mathbf{p}}) W(\mathbf{x}_1 - \mathbf{x}_{\mathbf{p}'}) \tag{5.32}$$

We may swap the summation indices through relabeling and exploit the antisymmetry of $\mathbf{d}$, obtaining:

$$\mathbf{F}_{21} = -m_1 m_2 \sum_{\mathbf{p}',\mathbf{p}} \mathbf{d}(\mathbf{p},\mathbf{p}') W(\mathbf{x}_1 - \mathbf{x}_{\mathbf{p}}) W(\mathbf{x}_2 - \mathbf{x}_{\mathbf{p}'}) \tag{5.33}$$

Hence we have $\mathbf{F}_{12} + \mathbf{F}_{21} = 0$, independent of where the points are located on the mesh.