PCFFT IMPLEMENTATION NOTES

A PREPRINT

January 26, 2026

1 Computing the spreading grid

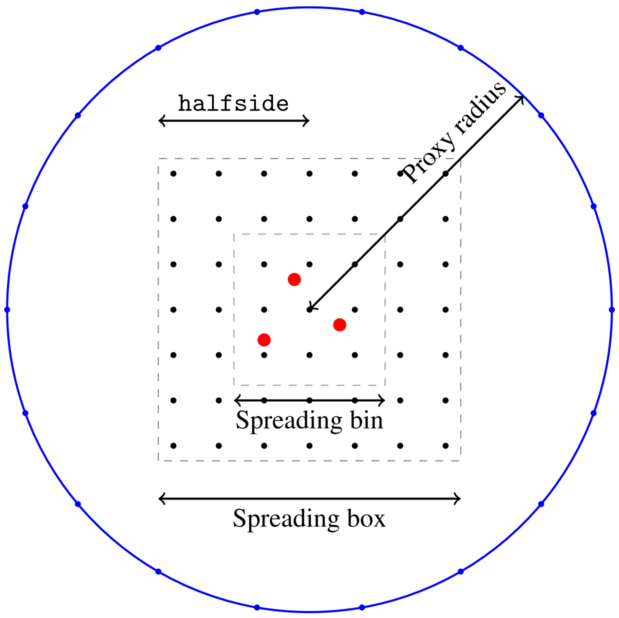


Figure 1: Schematic of the spreading geometry in 2D: sources (red), the regular discretization (black dots), and a proxy ring (blue).

1.1 Computing the spreading box size

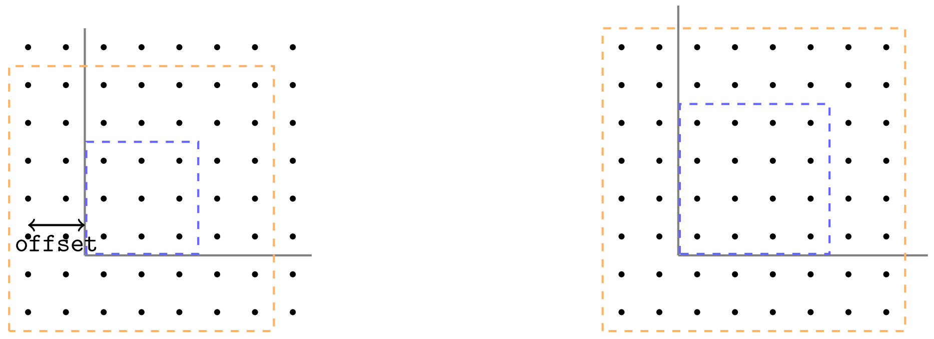
The spreading box size is computed by `spread_halfside()`. This is meant to approximately control the number of source points in a particular spreading box.

1.2 Computing the regular grid spacing

This is performed in `dx_nproxy()`. We want to find parameters `dx` and `nproxy`. `dx` is the grid spacing of the regular discretization of the spreading box, which starts at $-\texttt{halfside} + \frac{\texttt{dx}}{2}$ and ends at $\texttt{halfside} - \frac{\texttt{dx}}{2}$. `nproxy` is the number of proxy points placed on a proxy ring (or sphere) outside the spreading box.

Notes on the geometry used:

- We put source points in a bin with sidelength $\texttt{c\_bwidth} \times \texttt{halfside}$.



(a) $\texttt{nbinpts} = 3, \texttt{nspread} = 7$ **(b)** $\texttt{nbinpts} = 4, \texttt{nspread} = 8$

Figure 2: Schematic of the padding of the regular grid around the source points. The black lines form the bottom left corner of the bounding rectangle, the blue dashed line is the first spreading bin, and the orange dashed line is its associated spreading box. The regular grid points (black dots) start slightly below the bounding rectangle. There are $\texttt{pad} = \lceil(\texttt{nspread} - \texttt{nbinpts})/2\rceil$ extra grid points hanging below and to the left of the bounding rectangle to ensure proper padding for spreading.

- We place a proxy ring of radius $\sqrt{\texttt{d}} \times \texttt{halfside} \times \texttt{crad}$ where `d` is the dimension (2 or 3) and `crad` is a constant (default 2.0).
- If we consider breaking the spreading box into `nspread` cells, the grid points are placed at the center of each cell, so $\texttt{dx} = \frac{2\times\texttt{halfside}}{\texttt{nspread}}$.

Notes on the algorithm used to compute `dx` and `nproxy`:

- Generate random sources in the spreading bin with side length `halfside`. Generate target points on a ring/sphere of radius $1.1\times$ the proxy radius.
- Increase `nspread` and `nproxy` until the error tolerance is met.
- Decrease `nspread` until the error tolerance is no longer met.
- Spreading bin is $\texttt{dx} \times \texttt{nbinpts} = \texttt{dx} \times \lfloor\texttt{nspread}/2\rfloor$.

1.3 Constructing the regular grid

This is performed in `get_grid()`, which calls `spread_halfside()` and `dx_nproxy()`. Here are the basic steps:

- Compute `halfside` using `spread_halfside()`.
- Compute `dx`, `nspread`, and `nproxy` using `dx_nproxy()`.
- Construct a bounding rectangle around the source points, which, along with the side length of the spreading bin, determines `ngrid`, the number of regular grid points in each dimension.
- Add a padding of grid points on each side so that the spreading bins fit properly into the corners of the bounding rectangle. $\texttt{pad} = \lceil(\texttt{nspread} - \texttt{nbinpts})/2\rceil$.
- Start the regular grid points a bit below the bottom corner of the bounding rectangle. This offset is $\texttt{offset} = \texttt{pad} * \texttt{dx} - \texttt{dx}/2$ See Figure 2 for an illustration of this padding.

1.4 Computing the spreading matrix

The spreading matrix $\boldsymbol{A}$ maps from weights on the source points to equivalent weights on the regular grid. This matrix is constructed in `get_spread()`.

1.5 Computing the addsub matrix

Let's look at an N-body sum that we are trying to approximate. Suppose there are target points $\{x_i\}_{i=1,\dots}$ and source points $\{y_i\}_{j=1,\dots}$.

$$u_i = \sum_{j=1}^{N} K(x_i - y_j)\mu_j \tag{1}$$

$$= \sum_{j\in\text{Near}(x_i)} K(x_i - y_j)\mu_j + \sum_{j\notin\text{Near}(x_i)} K(x_i - y_j)\mu_j. \tag{2}$$

Ideally, we'd want to use the spreading matrix computed in the previous step to approximate the far interactions as

$$\sum_{j\notin\text{Near}(x_i)} K(x_i - y_j)\mu_j \approx \sum_k K(x_i - z_k)\boldsymbol{A}_{k,:}\boldsymbol{\mu},$$

where z_k are the grid points.

Let $\tilde{\boldsymbol{K}}$ be the kernel matrix from regular grid points to themselves. We would then compute

$$\sum_{j\notin\text{Near}(x_i)} K(x_i - y_j)\mu_j = \boldsymbol{A}^T\tilde{\boldsymbol{K}}\boldsymbol{A}\boldsymbol{\mu}, \tag{3}$$

However, the right hand sum contains all of the interactions, not just far interactions. So we need to compute a matrix $\boldsymbol{B}$ which approximates:

$$\begin{aligned}\boldsymbol{B}_{i,:}\boldsymbol{\mu} &= \sum_{j\in\text{Near}(x_i)} K(x_i - y_j)\boldsymbol{\mu}_j - \sum_k K(x_i - z_k)\boldsymbol{A}_{k,\text{Near}(i)}\boldsymbol{\mu}_{\text{Near}(i)} \\ &= \boldsymbol{A}^{(\text{add})}_{i,:}\boldsymbol{\mu} - \boldsymbol{A}^{(\text{sub})}_{i,:}\boldsymbol{\mu}\end{aligned}$$

where $\boldsymbol{A}^{(\text{add})}$ maps from source points to target points, and $\boldsymbol{A}^{(\text{sub})}_{i,j}$ is equal to $(\boldsymbol{A}^T\tilde{\boldsymbol{K}}\boldsymbol{A})_{i,j}$ if $j \in \text{Near}(x_i)$ and zero otherwise. Then we can evaluate (1) as:

$$\begin{aligned}u_i &= \sum_{j\in\text{Near}(x_i)} K(x_i - y_j)\mu_j + \sum_{j\notin\text{Near}(x_i)} K(x_i - y_j)\mu_j \\ &= \boldsymbol{A}^{(\text{add})}_{i,:}\boldsymbol{\mu} - \boldsymbol{A}^{(\text{sub})}_{i,:}\boldsymbol{\mu} + \boldsymbol{A}^T\tilde{\boldsymbol{K}}\boldsymbol{A}\boldsymbol{\mu}\end{aligned}$$

where $\tilde{\boldsymbol{K}}$ is the kernel matrix from regular grid points to themselves. Steps for computing this matrix:

- For each bin i, return a list of each bin j where the proxy circle of i intersects with the proxy circle of j.
- Compute the exact near-field interactions $\sum_{j\in\text{Near}(x_i)} K(x_i - y_j)\mu_j$.
- Compute the contribution of the near-field interactions to $\sum_k K(x_i - z_k)\boldsymbol{A}_{k,\text{Near}(i)}\boldsymbol{\mu}$.

1.6 Derivatives

If we want to dipoles instead of point charges, we change the spreading matrix

$$\sum_{j\notin\text{Near}(x_i)} n_j \cdot \nabla_y K(x_i - y_j)\mu_j \approx \sum_k K(x_i - z_k)\boldsymbol{A}^d_{k,:}\boldsymbol{\mu},$$

We would then evaluate the bulk of the potential as

$$u = \boldsymbol{A}^T\tilde{\boldsymbol{K}}\boldsymbol{A}^d\boldsymbol{\mu}, \tag{4}$$

If we want the derivate of the potential due to point charges, we would instead evaluate

$$\partial_n u = (\boldsymbol{A}^d)^T\tilde{\boldsymbol{K}}\boldsymbol{A}\boldsymbol{\mu}, \tag{5}$$