# Sparse approach to accelerate Particle-In-Cell method in 3D

Clément Guillet


# THÈSE

**En vue de l'obtention du**

## DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

**Délivré par :** *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

---

**Présentée et soutenue le** *28/06/2023* **par :**

Clément GUILLET

## Approche sur grilles parcimonieuses pour accélérer la méthode Particle-In-Cell.

---

**JURY**

| | | |
|---|---|---|
| Nicolas CROUSEILLES | Directeur de Recherche INRIA | Rapporteur |
| Raphaël LOUBERE | Directeur de Recherche CNRS | Rapporteur |
| Anne BOURDON | Directrice de Recherche CNRS | Examinatrice |
| Frédériques CHARLES | Maître(sse) de Conférences | Examinatrice |
| Pierre JOLIVET | Chargé de Recherche | Examinateur |
| Jacek NARSKI | Maître de Conférences | Examinateur |
| Fabrice DELUZET | Ingénieur de Recherche | Co-directeur de thèse |
| Laurent GARRIGUES | Directeur de Recherche CNRS | Co-directeur de thèse |
| Gwenael FUBIANI | Directeur de Recherche CNRS | Invité |
| Luc GIRAUD | Directeur de Recherche INRIA | Invité |

---

**École doctorale et spécialité :**
*GEET : Ingénierie des PLASMAS*

**Unité de Recherche :**
*Institut de Mathématiques de Toulouse (IMT), Laboratoire Plasma et Conversion d'Énergie (LAPLACE)*

**Directeur(s) de Thèse :**
*Fabrice DELUZET* et *Laurent GARRIGUES*

**Rapporteurs :**
*Nicolas CROUSEILLES* et *Raphael LOUBERE*

# Remerciements

Mes premiers remerciements se tournent naturellement vers mes directeurs de thèse, Fabrice et Laurent qui ont su m'aider de la meilleure des manières possibles durant ces trois années, sur le plan scientifique mais également sur le plan humain. Merci Fabrice pour ton aide constante tant sur le fond que sur la forme de mes travaux, notamment tes (très) nombreuses relectures ; ta disponibilité ; ton écoute, tes réponses à mes questions et tes nombreux enseignements ; ainsi que ton soutien dans les moments difficiles. Merci Laurent pour ton accompagnement au cours de ces trois années ; tes conseils, remarques et longues discussions scientifiques ; ainsi que tes nombreux apprentissages sur la physique des plasmas qui m'ont beaucoup apporté. Enfin, merci à vous pour votre bonne humeur constante et votre bienveillance.

Je remercie également Nicolas Crouseilles et Raphael Loubere d'avoir accepté de rapporter ma thèse, pour leur lecture et leur commentaires précis sur ce manuscrit, ainsi qu'à la totalité du jury : Pierre Jolivet, Frédériques Charles, Anne Bourdon, etc. d'avoir accepté mon invitation et d'avoir consacré une partie de leur temps à l'examen de mon travail.

Je tiens à remercier ceux qui m'ont également guidé tout au long de ses trois années. Merci Jacek pour ton écoute, ton aide précieuse à de nombreuses reprises et ton implication dans ma thèse. Merci Gwenael pour tes nombreuses remarques constructives sur mon travail qui m'ont permis à plusieurs reprises d'avancer dans ma recherche. Merci à tous les deux pour votre sympathie et votre soutien.

Il me faut également remercier la nombreuse équipe des doctorants/post-docs de l'IMT, sans qui je me serais senti bien seul durant cette thèse. Tout d'abord, les occupants (et ex-occupants) du bureau 301, avec qui ce fut un bonheur de partager ce bureau ces dernières années. Merci Michèle pour ton inépuisable joie de vivre, tes décorations qui ont donné de la vie à notre bureau et ton aide depuis mes tout premiers jours de thèse. Merci Lucas pour ta gentillesse ; ton intérêt au quotidien pour ce que je faisais, ça compte beaucoup pour moi ; pour nos longues discussions ; et les nombreuses fois où tu m'as aidé quand je rencontrais un problème. Merci également à mes voisins de bureau (302) : Louis, Étienne, Diego, Viviana, ainsi que tous les autres doctorants et post-docs que je ne pourrais pas tous citer : Alain, Benjamin, Paola, Corentin, Perla, Axel, Anthony, Alexandre, Nicolas O., Maxime, Mingmin, Alejandro, Javier, Joachim, Nicolas E.C., Sophia et les autres. Merci à tous pour ces moments passés ensemble à l'IMT ou en dehors. Merci également à Marc pour ton aide à plusieurs reprises.

Je remercie aussi mes amis extérieurs au labo. Mes amis nantais de l'université : Emilie, Flavie, Gaël, Louis, Louise, Myriam ; de qui je suis parti bien loin en allant à Toulouse et que j'espère retrouver.

Mes amis du sw : Agustin, Carlos, Chawal, Christophe, José, Killian, Mathieu, Mayeul, Rodrigo, Téïva et les autres; avec qui j'ai pu découvrir la ville de Toulouse. Merci pour ces (très) nombreux entraînements passés ensembles au park d'Empalot où j'ai largement passé la plupart de mon temps quand je n'étais pas au labo ou chez moi.

Et mes amis de toujours, les casseurs : Arnaud, Aurore, Hannah, Jeanne, Joséphine, Manon, Mathéo, Nathan, Roxane, Tangi, Tanguy; qui ont su être là pour moi et me remonter le moral quand j'en avais besoin. Le fait de pouvoir toujours compter sur vous représente énormément pour moi.

Je pense à ma famille. Merci au maurensois : Emmanuelle, Max, Laure et Félix, pour leur présence quand j'ai pu me sentir seul dans cette région que je connaissais si peu en arrivant, ainsi que tous les autres : Catherine, Philippe, Jean-Paul, Hervé, et ceux que je n'ai pas cité. Merci à mes grand-parents qui ont toujours été là pour moi, à mes grand-mères qui ont su être fortes ces dernières années, et à mes grand-pères qui ne liront pas ces lignes mais auxquels je pense.

Je remercie ma petite soeur, Cassiopée. Tu t'es toujours intéressée et donnée beaucoup de mal pour comprendre ce que je faisais pendant ces trois années de thèse. Merci à Maximilien, mon grand-frère, et Krystel ma belle-mère. Votre présence m'a beaucoup aidé.

Merci à mon papa et ma maman de m'avoir toujours entièremment soutenu dans ce que je voulais faire, ce qui a beaucoup compté pour moi, et grâce à qui je peux aujourd'hui fièremment présenter ce travail.

Enfin, merci (à nouveau) à toi, Aurore, d'avoir été à mes côtés pendant ces trois années. Tu as su être constamment là pour moi, malgré la distance qui a pu bien souvent nous séparer. Je te promets maintenant de rester proche de toi à l'avenir, où que tu sois.

# Contents

# List of Figures

# List of Tables

# Notations

In this section, most of the notations used in the manuscript are indexed. Throughout the reading of the manuscript, the reader is invited to refer to this section if a notation is not clear for him.

Let $d \in \mathbb{N}^*$ be the dimension of the problem considered. For the applications targeted $d = 2$ or 3. The domain of interest is a phase space constituted of a spatial domain, a velocity domain and a time domain. Let the spatial domain be the $d$-dimensional periodic unit interval $\Omega_x = (\mathbb{R}/\mathbb{Z})^d$, also denoted $\Omega$ in the sequel, the velocity domain be $\Omega_v = \mathbb{R}^d$ and the time domain be $\mathbb{R}^+$ or $[0, T]$, where $T \in \mathbb{R}_+^*$ is the final time.

Let us introduce some notations for multi-variate functions and functional spaces. For a multi-index $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_d) \in \mathbb{N}^d$, a function $u$ defined on $\Omega$, we denote by $\partial_i^{\alpha_i} u$ the partial derivative of $u$ order $\alpha_i$ with respect to $x_i$ for $i \in \{1, ..., d\}$ and let $D^{\boldsymbol{\alpha}} u = \partial_1^{\alpha_1} ... \partial_d^{\alpha_d} u$. We introduce the following functional spaces:

$$C^\alpha(\Omega) := \left\{ u : \Omega \to \mathbb{R} \mid D^{\boldsymbol{\beta}} u \in C(\Omega),\ \forall |\boldsymbol{\beta}|_1 \leq \alpha \right\}, \tag{1}$$

$$X^\alpha(\Omega) := \left\{ u : \Omega \to \mathbb{R} \mid D^{\boldsymbol{\beta}} u \in C(\Omega),\ \forall |\boldsymbol{\beta}|_\infty \leq \alpha \right\}, \tag{2}$$

where $C(\Omega)$ denotes the space of continuous functions on $\Omega$, $C_0^\alpha(\Omega)$ and $X_0^\alpha(\Omega)$ the spaces of functions vanishing on the boundary. Let us consider the supremum norm for a function $u$ belonging to one of the previous spaces and the extension for a vector function $\mathbf{u} : \Omega \to \mathbb{R}^p$, $p \in \mathbb{N}$:

$$\|u\|_\infty := \sup_{\mathbf{x} \in \Omega} |u(\mathbf{x})|, \quad \|\mathbf{u}\|_\infty = \max_{1 \leq i \leq p} \|u_i\|_\infty \tag{3}$$

The following operators are introduced:

$$\begin{array}{rccc} \boldsymbol{\nabla} & : & C^\alpha(\Omega) & \to & \left(C^{\alpha-1}(\Omega)\right)^d \\ & & u & \mapsto & (\partial_i u)_{i=1,...,d} \end{array}, \text{ for } \alpha \geq 1, \tag{4}$$

$$\begin{array}{rccc} \boldsymbol{\nabla}\cdot & : & (C^\alpha(\Omega))^d & \to & C^{\alpha-1}(\Omega) \\ & & \mathbf{u} & \mapsto & \sum_{i=1}^d \partial_i u_i \end{array}, \text{ for } \alpha \geq 1, \tag{5}$$

$$\begin{array}{rccc} \Delta & : & C^\alpha(\Omega) & \to & C^{\alpha-2}(\Omega) \\ & & u & \mapsto & \sum_{i=1}^d \partial_i^2 u \end{array}, \text{ for } \alpha \geq 2. \tag{6}$$

Let us define the following order relations on multi-indexes $\mathbf{k}, \mathbf{l} \in \mathbb{N}^d$ by:

$$\mathbf{k} \leq \mathbf{l} \ \Leftrightarrow\ \forall i \in \{1, ...d\}\ k_i \leq l_i, \tag{7}$$

$$\mathbf{k} < \mathbf{l} \ \Leftrightarrow\ \mathbf{k} \leq \mathbf{l} \text{ and } \exists\, i \in \{1, ..., d\}\ \ s.t.\ k_i < l_i, \tag{8}$$

and , the $l^1$ norm, $l^\infty$ norm for a multi-index $\boldsymbol{\alpha} \in \mathbb{N}^d$:

$$|\boldsymbol{\alpha}|_1 := \sum_{i=1}^{d} |\alpha_i|, \quad |\boldsymbol{\alpha}|_1^{r,s} := \sum_{i=r}^{s} |\alpha_i|, \quad |\boldsymbol{\alpha}|_\infty := \max_{1 \le i \le d} |\alpha_i|, \tag{9}$$

where $1 \le r \le s \le d$ are integers.

Let us now describe the different domains (*i.e.* $\Omega_x, \Omega_v, \mathbb{R}^+$) discretizations. Let $\Delta t \in \mathbb{R}^{+,*}$ be the step of the time discretization such that $t^k = k\Delta t$, for $k \in \mathbb{N}$. The role of the space domain discretization is fundamental in this manuscript. Specifically, different meshes, called grids, are considered in the sequel.

**Definition 0.0.1 (Component indices)** *Let $\mathbb{L}_n$ be a set of indices, called the component indices, with respect to the discretization parameter $n \in \mathbb{N}$ and defined by:*

$$\mathbb{L}_n := \bigcup_{i \in [\![0, d-1]\!]} \mathbb{L}_{n,i}, \quad \mathbb{L}_{n,i} := \{\boldsymbol{l} \in \mathbb{N}^d \mid |\boldsymbol{l}|_1 = n + d - 1 - i,\ \boldsymbol{l} \ge \boldsymbol{1}\}, \tag{10}$$

Let us consider the family of d-dimensional anisotropic grids on the space domain $\Omega_x$ called *component grids*, or *sub-grids*:

**Definition 0.0.2 (Component grids)** *The component grids are defined for $\boldsymbol{l} \in \mathbb{L}_n$ by:*

$$\Omega_{h_{\boldsymbol{l}}} := \left\{\boldsymbol{j} h_{\boldsymbol{l}} \mid \boldsymbol{j} \in I_{h_{\boldsymbol{l}}}\right\}, \quad I_{h_{\boldsymbol{l}}} := [\![0, h_{l_1}^{-1} - 1]\!] \times \ldots \times [\![0, h_{l_d}^{-1} - 1]\!] \subset \mathbb{N}^d, \tag{11}$$

*where:*

$$h_{\boldsymbol{l}} := (h_{l_1}, \ldots, h_{l_d}) \in \mathbb{R}^d, \ \ h_{l_i} = 2^{-l_i} \text{ for } l_i \in \mathbb{N} \tag{12}$$

*is called the grid discretization and corresponds to the cell grid width.*

The number of component grids is given by:

$$|\mathbb{L}_n| := \ \mathrm{Card}(\mathbb{L}_n) = \sum_{i=0}^{d-1} \binom{n + d - 2 - i}{d - 1} = O\left(|\log h_n|^{d-1}\right), \tag{13}$$

where $\binom{i}{j}$, for $i \ge j$ integers, is the binomial coefficient defined as:

$$\binom{i}{j} := \frac{i!}{j!(i-j)!}. \tag{14}$$

Let us also consider a regular isotropic grid, named *Cartesian grid* or *full grid*, corresponding to a component grid of level $\mathbf{n} = n \cdot \mathbf{1}$ with uniform discretization $h_n$ for any direction:

**Definition 0.0.3 (Cartesian grid)** *The Cartesian grid, denoted $\Omega_{h_n}^{(\infty)}$, is defined by:*

$$\Omega_{h_n}^{(\infty)} := \left\{\boldsymbol{j} h_n \mid \boldsymbol{j} \in I_{h_n}\right\} \subset \Omega, \quad I_{h_n} := [\![0, h_n^{-1} - 1]\!]^d \subset \mathbb{N}^d, \tag{15}$$

*where $h_n = 2^{-n}$ is the Cartesian grid discretization.*

The terminology *sparse grid* refers to the grid constituted of all the component grid nodes and

is defined by:

$$\Omega_{h_n}^{(1)} := \bigcup_{\mathbf{l} \in \mathbb{L}_n} \Omega_{h_\mathbf{l}} \tag{16}$$

An illustration of the different grids is provided in figure 1.

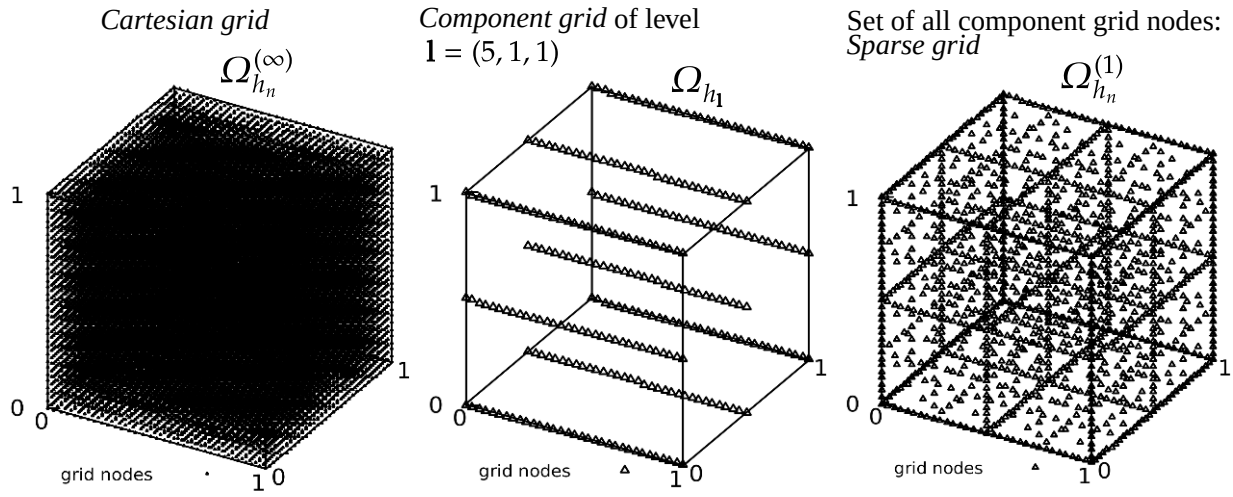


**Figure 1.** Representation of the Cartesian grid, a component grid and the sparse grid related to $n = 5$.

Let us now define discrete operators on the grids introduced in the previous section. Let $\boldsymbol{\nabla}_{h_\mathbf{l}}$ and $\Delta_{h_\mathbf{l}}$ be the discrete second order finite difference operators defined on the component grid $\Omega_{h_\mathbf{l}}$ by:

$$\Delta_{h_\mathbf{l}} u := \sum_{i=1}^{d} \delta_{i,h_{l_i}}^{+} \delta_{i,h_{l_i}}^{-} u, \quad \boldsymbol{\nabla}_{h_\mathbf{l}} u := \left( \delta_{i,h_{l_k}}^{0} u \right)_{i=1,\dots,d}, \tag{17}$$

$\delta_{i,h_{l_k}}^{-}$, $\delta_{i,h_{l_i}}^{+}$ are left-sided, right-sided differences and $\delta_{i,h_{l_i}}^{0}$ is the centered difference:

$$\delta_{i,h_{l_i}}^{+} u(\mathbf{j} h_\mathbf{l}) := \frac{u((\mathbf{j} + \mathbf{e}_i) h_\mathbf{l}) - u(\mathbf{j} h_\mathbf{l})}{h_{l_i}}, \quad \delta_{i,h_{l_i}}^{-} u(\mathbf{j} h_\mathbf{l}) := \frac{u(\mathbf{j} h_\mathbf{l}) - u((\mathbf{j} - \mathbf{e}_i) h_\mathbf{l})}{h_{l_i}}, \tag{18}$$

$$\delta_{i,h_{l_i}}^{0} u(\mathbf{j} h_\mathbf{l}) := \frac{u((\mathbf{j} + \mathbf{e}_i) h_\mathbf{l}) - u((\mathbf{j} - \mathbf{e}_i) h_\mathbf{l})}{2 h_{l_i}}, \tag{19}$$

for $\mathbf{j} \in I_{h_\mathbf{l}}$ and where $\mathbf{e}_i \in \mathbb{N}^d$ is the index whose value is 1 along the $i^{th}$ coordinate and 0 elsewhere. The definition may be extended to the Cartesian grid by setting $h_\mathbf{l} = (h_n, ..., h_n)$.

Let $u$ be a function, $u_{h_\mathbf{l}}$ an approximation of the function of the component grid $\Omega_{h_\mathbf{l}}$, and $(u_{h_\mathbf{l};\mathbf{j}})_{\mathbf{j} \in I_{h_\mathbf{l}}}$ the sequence of the approximations on the component grid nodes. The Discrete Fourier Transform (DFT) of this sequence is defined for $\mathbf{m} \in I_{h_\mathbf{l}}$ by:

$$\mathcal{F}_\mathbf{m}(u_{h_\mathbf{l}}) = \sum_{\mathbf{j} \in I_{h_\mathbf{l}}} u_{h_\mathbf{l};\mathbf{j}} e^{-2\pi i \mathbf{m} \mathbf{j} h_\mathbf{l}}, \tag{20}$$

where the notation $\mathbf{mj}h_{\mathbf{l}}$ stands for:

$$\mathbf{mj}h_{\mathbf{l}} = \sum_{i=1}^{d} m_i j_i h_{l_i}.$$

# Nomenclature

## Symbols and conventions

$d \in \mathbb{N}^*$: dimension of the problem.
$\mathbf{l} \in \mathbb{N}^d$ (sometimes $\mathbf{k}$): index level referring to the grid discretization $h_{\mathbf{l}}$.
$\mathbf{j} \in \mathbb{N}^d$ (sometimes $\mathbf{i}$): spatial index associated to a grid (*e.g.* index of a grid node).
$\Omega_x \subset \mathbb{R}^d$: $d$-dimensional spatial domain, $\Omega_v \subset \mathbb{R}^d$: $d$-dimensional velocity domain.
$n \in \mathbb{N}^*$: integer related to the maximum discretization of $\Omega_x$ in each direction.
$N \in \mathbb{N}^*$: number of numerical particles.
$h_n = 2^{-n} \in \mathbb{R}_+^*$: grid discretization, note that $|\log h_n| = n \log 2$.

**Table 1.** Meaning and definition of math symbols.

| symbol | designation | definition |
|---|---|---|
| $\mathbb{L}_n$ | set of component indices | (10) |
| $\Omega_{h_{\mathbf{l}}}, \Omega_{h_n}^{(\infty)}$ | component grid of level $\mathbf{l}$, full grid | (11),(15) |
| $I_{h_{\mathbf{l}}}, I_{h_n}$ | set of component grid indices, full grid indices | (11),(15) |
| $h_{\mathbf{l}}$ | grid discretization of level $\mathbf{l}$ (width of grid cells) | (12) |
| $\mathcal{S}_{h_{\mathbf{l}}}$ | shape function used for charge density accumulation | (2.22) |
| $\hat{\rho}_{h_{\mathbf{l}}}$ | statistical estimator of charge density | (2.30) |
| $\mathcal{V}(\hat{u}_h)$ | particle sampling error of the estimator $\hat{u}_h$ | (2.32) |
| $\mathrm{Bias}(\hat{u}_h)$ | grid-based error of the estimator $\hat{u}_h$ | (2.32) |
| $W_{h_{\mathbf{l}}}$ | basis function used for interpolation | (2.1) |
| $\hat{\rho}_{h_n}^c, \hat{\mathbf{E}}_{h_n}^c$,... | sparse grid reconstruction of $\rho, \mathbf{E}$,... | (2.11) |
| $\mathcal{I}_{V_{h_{\mathbf{l}}}}, \mathcal{I}_{V_{hn}^{(\infty)}}$,... | linear interpolation onto the space $V_{h_{\mathbf{l}}}, V_{h_n}^{(\infty)}$,... | (2.10) |
| $\mathcal{F}_{\mathbf{m}}(u_{h_l})$ | discrete Fourier transform of $u_{h_l}$ | (20) |
| $\Delta h_{\mathbf{l}}$ | volume of the component grid ($\Omega_{h_{\mathbf{l}}}$) cells | (5.24) |
| $P_c$ | mean number of particles per cell | (6.1),(6.2) |

## Physical quantity

$f_s : \Omega_x \times \Omega_v \times \mathbb{R}^+ \to \mathbb{R}^+$: distribution of particles of species $s$.
$\rho : \Omega_x \times \mathbb{R}^+ \to \mathbb{R}^+$: charge density.
$\mathbf{E} : \Omega_x \times \mathbb{R}^+ \to \mathbb{R}^d$: electric field.
$\mathbf{B} : \Omega_x \times \mathbb{R}^+ \to \mathbb{R}^d$: magnetic field.
$\Phi : \Omega_x \times \mathbb{R}^+ \to \mathbb{R}$: electric potential.
$\mathbf{J} : \Omega_x \times \mathbb{R}^+ \to \mathbb{R}^d$: charge current.
$\epsilon_0 \in \mathbb{R}_+^*$: vacuum permittivity.
$m_s, T_s \in \mathbb{R}_+^*$: mass and temperature of particle species $s$.
$q_s, n_s$: charge, and density of particle species $s$.

**Table 2.** PIC scheme abbreviations and significations.

| Abbreviations | Time discretization | Grid for... | | Sorting of particles | Basis for combination |
|---|---|---|---|---|---|
| | | Field resolution | Density/ current accumulation | | |
| PIC-Std, Ex-PIC-Std | explicit | $\Omega_{h_n}^{(\infty)}$ | $\Omega_{h_n}^{(\infty)}$ | | |
| PIC-SStd | explicit | $\Omega_{h_n}^{(\infty)}$ | $\Omega_{h_n}^{(\infty)}$ | yes | |
| PIC-UStd | explicit | $\Omega_{h_n}^{(\infty)}$ | $\Omega_{h_n}^{(\infty)}$ | no | |
| PIC-Sg,PIC-NSg Ex-PIC-Sg | explicit | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | no | nodal |
| PIC-Hg | explicit | $\Omega_{h_n}^{(\infty)}$ | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | no | nodal |
| PIC-HSg | explicit | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | no | hierachical |
| PIC-OHg | explicit | $\Omega_{h_n^{(\infty)}}$ | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_{n+\delta n}}$ | no | nodal |
| PIC-ESg | explicit | $\Omega_{h_{\mathbf{l}+\delta n}}$ | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | no | nodal |
| Imp-PIC-Std | implicit | $\Omega_{h_n}^{(\infty)}$ | $\Omega_{h_n}^{(\infty)}$ | no | nodal |
| Imp-PIC-Sg | implicit | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | no | nodal |
| Imp-PIC-Hg | implicit | $\left(\Omega_{h_{\mathbf{l}}}\right)_{\mathbf{l}\in\mathbb{L}_n}$ | $\Omega_{h_n}^{(\infty)}$ | no | nodal |

$\lambda_D = \sqrt{\varepsilon_0 T_e / q_e n_e} \in \mathbb{R}_+$: Debye length.
$\omega_p = \sqrt{q_e n_e / m_e \varepsilon_0} \in \mathbb{R}_+$: plasma period.

**Acronyms and abbreviations**

PIC: Particle-In-Cell
B: Bytes
HPC: High performance Computing
CPU: Central Processing Unit
GPU: Graphics Processing Unit
GPGPU: General-Purpose computing on Graphics Processing Units
ECSIM: Energy-Conserving Semi-Implicit Method
(Push): Evolution of particle (step of PIC scheme)
(F.Inter): Field interpolation (step of PIC scheme)
(Proj): Charge density accumulation (step of PIC scheme)
(Pois): Resolution of Poisson equation (step of PIC scheme)
(Comb): Sparse grid combination to reconstruct the solution (step of PIC scheme)
(Diff): Differentiation of the electric potential (step of PIC scheme)

# Chapter 1

# Introduction

*"Recently, a powerful new method for both types of investigation [a] has become possible through the advent of modern high-speed computers. This is the method of computer simulation or computer modeling."*

[a] *[the experimental techniques in which one disturbs the system in some controlled manner and observes its behavior, and the theoretical approach in which one uses analytical mathematical techniques to determine the behavior consistent with well-established physical laws]*

John M. Dawson, *Particle simulation of plasmas*, 1983.

Traditionally, physical complex systems have been studied with two well-known techniques: experimental techniques and mathematical analysis techniques. Most of the major advances in physics have been achieved thanks to the combination of both techniques. Nonetheless, there exists a large number of problems that cannot be overcome with any of these methods. Indeed, for various physical problems, experiments are difficult or even impossible to carry out and the large number of degrees of freedom of the problem makes analytical techniques impractical. In that scope, the emergence of computer simulations has made possible the study and understanding of a large variety of problems.

Computer simulations can be used to achieve results of practical interest, such as determining the performance of a fusion reactor, predicting the weather or study the strength of a manufactured material. It can also be used to better understand the physical laws of nature and some of its behaviors. The first step of a computer simulation method is to construct a numerical model of the system of interest. From this model, numerical experiments, in which the initial conditions are evolved according to the laws of the model, are conducted on a computer. Finally, the results of the numerical experiments can be compared to theoretical predictions based on simplified models, to experimental results or to observations of natural phenomena.

At the beginning of computer simulations, the experiments were restricted to problems of small sizes and performed on a few range of high performance computers. But with the massive increase of developments in the computer engineering area during the last decades, more and more complex problems have become achievable on high performance computers, as well as regular computers.

Particle models are among the most successful models for the simulation of plasmas at the core of this work. These models are based on the evolution of a large number of charged particles, which can be electrons, ions or others, in a self-consistent electromagnetic field. The main limitation of particle methods comes from the large number of particles to be considered.

Indeed, for most range of applications, the number of particles in plasmas largely exceeds the computational resources, even for world class supercomputers. Taking the example of plasma fusion application, the plasmas density considered is about $10^{20}$ particles per cubic meter for a tokamak of about $10^2$ cubic meters, corresponding to a total of $10^{22}$ particles. The current* most powerful supercomputer has a storage of about 1000PB, that is to say it can store the three-dimensional positions of roughly $10^{16}$ particles. For this reason, less detailed descriptions of the plasma are favored for computer simulations and alternative numerical model had to be conceived.

## Contents



# 1.1 Plasma physics background

> *"It has often been said that 99% of the matter in the universe is in the plasma state; that is, in the form of an electrified gas with the atoms dissociated into positive ions and negative electrons. On the other hand, in our everyday lives encounters with plasmas are limited to a few examples: the flash of a lightning bolt, the soft glow of the Aurora Borealis, the conducting gas inside a fluorescent tube or neon sign, and the slight amount of ionization in a rocket exhaust. It would seem that we live in the 1% of the universe in which plasmas do not occur naturally."*
>
> F.F. Chen, *Introduction to plasma physics and controlled fusion*, 1984.

When an atom or a molecule looses or gains an electron, it acquires a negative or positive charge and becomes an ion. This process is called *ionization* and is at the origin of the creation of a plasma. Nonetheless, any ionized gas is not necessarily called a plasma since there is always

*from the Top500 ranking in November 2022. https://www.top500.org/

a small degree of ionization in every gases. A plasma is different from regular ionized gas in that the particles should have collective behaviors. Let us consider the definition of a plasma introduced in [28].

**Definition 1.1.1 (plasma)** *A plasma is a quasineutral gas of charged (ions, electrons, etc.) and neutral particles which exhibits collective behavior.*

In a plasma, the distribution of charged particles and their motion create local concentrations of positive of negative charges, as well as currents. These local concentrations of charges and currents give rise to electric and magnetic fields which affect the motion of the particles in a large range of action. It results that the effects of charged particles on other particles are not restricted to local Coulomb interactions, but can occur for particles at large distances from each others. This long-ranged Coulomb force is the cause of the particle motion complexity in plasmas and make the understanding of their behaviors difficult. It is particularly true for the so-called collisionless plasmas in which long-range electromagnetic forces are so much larger than the forces due to ordinary local collisions that the latter can be neglected. In this manuscript, we have concentrated our efforts on collisionless plasma.

> *"Fusion energy could one day be a transformative energy source, because it will be clean, cheap, and nearly unlimited, with sea water supplying its basic fuel. A whole-device computer model can offer insights about the plasma processes that go on in the fusion device and predictions regarding the performance and optimizations of next-step experimental facilities."*
>
> A. Bhattacharjee, 2019.

The interest in plasma physics has significantly increased since the early 50's, when it was first propose to control the hydrogen bomb fusion reaction to create a reactor and produce energy. The process of fusing atomic nuclei by using high temperatures to bring them closer is called thermonuclear fusion. There exists two kind of thermonuclear fusion: uncontrolled fusion in which the energy released by the reaction is uncontrolled (*e.g.* thermonuclear weapons such as hydrogen bombs or the fusion in most stars); and controlled fusion in which the reaction takes place in a regulated environment and the energy released is controlled and can be harnessed (*e.g.* thermonuclear reactor). The principal and most accessible reaction is to fuse deuterium (D) and tritium (T) atoms, which are isotopes of hydrogen, to produce a helium atom and a neutron as follow:

$$D + T \rightarrow {}^{4}He + n + 17.6\ MeV. \tag{1.1}$$

## 1.2 Mathematical model: Vlasov-Maxwell/Poisson systems

> *"The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work."*
>
> J. Von Neumann, *Method in the Physical Sciences*, 1955.

As mentioned in the preceding lines, the description of a plasma based on the individual representations of each particle is in most applications impractical for computer simulations. Therefore, a different representation of the plasma shall be used to construct a numerical model on which numerical experiments can be performed. Usually, this is done by considering approximations leading to a less detailed descriptions of the plasma. Among the several plasma models existing, two are mostly used: the fluid description and the kinetic description.

The kinetic description is the second most fundamental way of representing a plasma, after the individual particle representation. It consists in considering a particle distribution function which takes its arguments in the phase space (made of positions and velocities of particles) and in time. The kinetic model is obtained by considering the Vlasov equation for the evolution of the particle distribution coupled to Maxwell's equations driving the changes in the electromagnetic field created by charge and current densities produced by the particles.

In fluid models, the description of the plasma is rather based on equations driving the evolution of macroscopic quantities (velocity moments of the distribution such as density, mean velocity, and mean energy) which are obtained from the moments of the Vlasov equation. The motivation is to reduce the complexity compared to the kinetic approach, macroscopic quantities being functions of only space and time variables.

The set of equation being of our concern in kinetic models is the Vlasov-Maxwell system. Let $f_s$ be a function attached to the distribution of the species $s$. These species could be for examples ions, electrons, negative ions, etc. The distribution of particles takes its arguments in the phase space and the time domain. It maps towards the real domain: $f_s : \Omega_x \times \Omega_v \times \mathbb{R}^+ \to \mathbb{R}^+$. The phase space consists of a $d$-dimensional spatial domain $\Omega_x \subset \mathbb{R}^d$ for the positions of the particles and a $d$-dimenisonal velocity domain $\Omega_v \subset \mathbb{R}^d$ for the velocity of the particles. The Vlasov equation describes the evolution of charged particles in an electromagnetic field which can either be self-consistent, that is to say, generated by the particles themselves, externally applied, or both. Assuming non-relativistic, non-collisional particles, the Vlasov equation falls to:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \boldsymbol{\nabla}_{\mathbf{x}} f_s + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \boldsymbol{\nabla}_{\mathbf{v}} \partial f_s = 0, \tag{1.2}$$

where we introduce the notation $(\boldsymbol{\nabla}_{\mathbf{x}} f)_i = \frac{\partial f}{\partial x_i}$, $(\boldsymbol{\nabla}_{\mathbf{v}} f)_i = \frac{\partial f}{\partial v_i}$, for $i \in \{1, ..., d\}$. In the above relation $q_s$, $m_s$ respectively states for the charge, mass of the particle of species $s$, $\mathbf{E}$ the electric field, $\mathbf{B}$ the magnetic field. The latters are decomposed into self-consistents fields, i.e. generated by the particles and external fields:

$$\mathbf{E} := \mathbf{E}_{self} + \mathbf{E}_{ext}, \tag{1.3}$$

$$\mathbf{B} := \mathbf{B}_{self} + \mathbf{B}_{ext}. \tag{1.4}$$

From the particle distribution function $f_s$, one can define the particle density $n_s$ related to the

species, the electrical charge and current densities $\rho$, $\mathbf{J}$:

$$n_s = \int_{\Omega_v} f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}, \tag{1.5}$$

$$\rho = \sum_s q_s n_s, \tag{1.6}$$

$$\mathbf{J} = \sum_s q_s \int_{\Omega_v} f(\mathbf{x}, \mathbf{v}, t) \mathbf{v} d\mathbf{v} \tag{1.7}$$

The self consistent electric and magnetic fields are generated by the particles from the electrical charge and current density according to Maxwell's equations:

$$\frac{1}{c^2}\frac{\partial \mathbf{E}_{self}}{\partial t} - \boldsymbol{\nabla} \times \mathbf{B}_{self} = -\mu_0 \mathbf{J}, \tag{1.8}$$

$$\frac{\partial \mathbf{B}_{self}}{\partial t} + \boldsymbol{\nabla} \times \mathbf{E}_{self} = 0, \tag{1.9}$$

$$\boldsymbol{\nabla} \cdot \mathbf{E}_{self} = \frac{\rho}{\varepsilon_0}, \tag{1.10}$$

$$\boldsymbol{\nabla} \cdot \mathbf{B}_{self} = 0, \tag{1.11}$$

where $c$ is the speed of light, $\mu_0$ the vacuum permeability and $\varepsilon_0$ the vacuum permittivity with $\mu_0 \varepsilon_0 c^2 = 1$. The Maxwell system consists of the Maxwell-Ampère (1.8), Maxwell-Faraday (1.9), Maxwell-Gauss (1.10), Maxwell-Thompson (1.11) equations. Initial and boundary conditions are needed in addition to fully determine the solution of the system.

The electrostatic regime is recovered from the Maxwell system when the magnetic field vanishes. The equations reduce then to:

$$\frac{1}{c^2}\frac{\partial \mathbf{E}_{self}}{\partial t} = \mu_0 \mathbf{J}, \tag{1.12}$$

$$\nabla \times \mathbf{E}_{self} = 0, \tag{1.13}$$

$$\nabla \cdot \mathbf{E}_{self} = \frac{\rho}{\varepsilon_0}. \tag{1.14}$$

Equation (1.12) and (1.14) are equivalent as soon as the continuity equation is satisfied. This equation writes:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{J} = 0. \tag{1.15}$$

The continuity equation is actually an outcome of the Vlasov equation. It is indeed obtained as the moment of zero order of the Vlasov equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{J} = \int_{\Omega_v} \left( \sum_s \frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \boldsymbol{\nabla}_{\mathbf{x}} f_s + \frac{q_s}{m_s} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \boldsymbol{\nabla}_{\mathbf{v}} \partial f_s \right) d\mathbf{v}. \tag{1.16}$$

Finally it is immediate to verify that if $\mathbf{E}$ derives from the gradient of a potential, equation (1.13) holds true. Therefore, in the electrostatic regime the electric field can equally be computed

thanks to:

$$\begin{cases} -\dfrac{\partial \mathbf{E}_{self}}{\partial t} = \dfrac{1}{\varepsilon_0}\nabla \cdot \mathbf{J}, \qquad \nabla \times \mathbf{E}_{self} = 0, \\ \text{or} \\ -\Delta \Phi_{self} = \dfrac{\rho}{\varepsilon_0}, \qquad \mathbf{E}_{self} = -\nabla \Phi_{self}, \end{cases} \tag{1.17}$$

The last equation is the so-called Poisson equation. Note that the electrostatic regime may also prevail for non vanishing magnetic fields [46]. The system of Vlasov-Poisson is then:

$$\begin{cases} \dfrac{\partial f_s}{\partial t} + \mathbf{v} \cdot \boldsymbol{\nabla}_{\mathbf{x}} f_s + \dfrac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}_{ext}) \cdot \boldsymbol{\nabla}_{\mathbf{v}} f_s = 0, \\ -\Delta \Phi_{self} = \dfrac{\rho}{\varepsilon_0}, \qquad \mathbf{E}_{self} = -\boldsymbol{\nabla} \Phi_{self}, \end{cases} \tag{1.18}$$

with adequate boundary conditions. Note that for this system, the electrical charge current $\mathbf{J}$ is no longer necessary to be computed. In the following of the manuscript the subscripts for the self-consistent and exterior fields are omitted.

**Proposition 1.2.1 (Conservation properties)** *The Vlasov-Poisson system (1.18) verifies the following conservation properties:*

- *Conservation of momentum:*

$$\frac{d}{dt}\left( \sum_s \iint_{\Omega_x \times \Omega_v} \boldsymbol{v} f_s(\boldsymbol{x}, \boldsymbol{v}, t) d\boldsymbol{x} d\boldsymbol{v} \right) = 0. \tag{1.19}$$

- *Conservation of $L^p$ norms, $p \in [\![1, \infty]\!]$:*

$$\frac{d}{dt}\left( \sum_s \iint_{\Omega_x \times \Omega_v} f_s(\boldsymbol{x}, \boldsymbol{v}, t)^p d\boldsymbol{x} d\boldsymbol{v} \right) = 0. \tag{1.20}$$

- *Conservation of energy:*

$$\frac{d}{dt}\left( \sum_s \frac{1}{2} \iint_{\Omega_x \times \Omega_v} m|\boldsymbol{v}|^2 f_s(\boldsymbol{x}, \boldsymbol{v}, t) d\boldsymbol{x} d\boldsymbol{v} + \frac{\varepsilon_0}{2} \int_{\Omega_x} |\boldsymbol{E}|^2 d\boldsymbol{x} \right) = 0. \tag{1.21}$$

## 1.3 Particle-In-Cell method

Particle-In-Cell (PIC) discretizations have been for years among the most used numerical methods in the simulation of kinetic plasmas [42, 15, 70, 102] and are still topical [47, 59, 55, 94]. The method consists in a coupling between a Lagrangian method for the Vlasov equation, based on the integration of numerical particle trajectories, and a mesh-based discretization of Poisson's equation (or Maxwell's system) for the computation of the self-consistent field. It results in a tightly coupled non-linear system whose solutions are proven to be challenging to determine. The specificity of PIC methods is the mixed discretization, made of both an Eulerian grid for the moments of the particle distribution and fields, in conjunction with individual Lagrangian particles discretizing the continuous phase space.

The principle of the methods is to represent the particle distribution function by a collection of numerical particles (macro-particles). The method is different from the individual particle

description of a plasma in that here a macro-particle represents a heap of physical particles. The numerical particles are advanced in time by solving the equations of motion, obtained from the Vlasov equation, with the electromagnetic field provided by Maxwell's equations.

Originally, and still in a large range of applications, PIC implementations are based on an explicit discretization (in time) of the Vlasov equation. In explicit formulations, the non-linear coupling between the particles and the self-consistent fields (generated by the particles) is broken down within a time step. Explicit time integrating benefits from simplicity of implementation, as well as a poor computational cost for one iteration. Nonetheless, explicit approaches suffer from temporal stability constraints imposing a limit on the time-step discretization, forcing the user to resolve the fastest wave described by the set of equations. The stability constraint in explicit PIC schemes is a well-known problem and has been extensively studied [70, 18]. In addition, these approaches usually feature spatial stability constraints, manifested by numerical instabilities called aliasing or finite grid instability [82, 72] occurring when the grid discretization (grid cell size) is equal or superior to the local Debye length of the plasma. Therefore, the application of explicit approaches to multidimensional problems, especially for three dimensional geometries, can be very computationally demanding and cumbersome.

In response to these issues, implicit formulations of PIC schemes have emerged and received a lot of attention, particularly thanks to their stability properties. A more detailed introduction of implicit methods is presented in chapter 5 of this manuscript.

An extensive literature on the PIC methods is available, including the fundamental, physics oriented works from Birdsall, Langdon [18] and Hockney, Eastwood [71]. The mathematics community has also been interested for years in the study of the PIC methods with, among others, the works from Cottet, Raviard [38, 39].

### 1.3.1 Numerical approximation of the particle distribution

The distribution of particles is represented by a collection of macro-particles. A macro-particle, also called numerical particle, refers to a heap of physical particles of the same species (electrons, ions, etc.). Let $N_s$ denotes the number of macro-particles attached to the species $s$. The positions and velocities of a particle are denoted $(\mathbf{x}_p, \mathbf{v}_p)$, $p = 1, ..., N_s$ being the index of the particles. We assume that all the numerical particles of one species have the same weight, defined by the ratio of physical particles ($n_s$) per numerical particle ($N_s$):

$$\omega_p = \frac{\int_{\Omega_x} n_s d\mathbf{x}}{N_s}, \qquad \forall p = 1, ..., N_s, \tag{1.22}$$

and the same charge and mass:

$$q_p = q_s \omega_p, \quad m_p = m_s \omega_p, \qquad \forall p = 1, ..., N_s. \tag{1.23}$$

The principle of the numerical approximation is to discretize the distribution of particles function with a sum of Dirac distribution centered at the positions and velocities of the macro-particles, according to the relation:

$$f_{N_s}(\mathbf{x}, \mathbf{v}, t) = \sum_{p=1}^{N_s} \omega_p \delta(\mathbf{x} - \mathbf{x}_p(t)) \delta(\mathbf{v} - \mathbf{v}_p(t)), \tag{1.24}$$

In the remaining of the chapter, we assume one specific type of particle, and omit the subscript $s$.

### 1.3.2 Depiction of the explicit scheme

Traditionally, the explicit Particle-In-Cell scheme consists of four steps repeated at each time iteration: charge density accumulation, computation of the electric field, interpolation of the electric field and update of particle positions and velocities.

First, the particle population is initialized according to the initial distribution function $f(\mathbf{x}, \mathbf{v}, t = 0)$. Then, at each time iteration, the charge density is approximated onto a grid $\Omega_h$ (usually the full grid defined in equation (15)); an approximation of the electric field is computed on the grid from the charge density with mesh-based numerical methods; the electric field is interpolated at the particle positions; and the particle population is updated by integrating Newton's laws. The scheme is summarized in algorithm 1 and illustrated on figure 1.1. The steps introduced here will be detailed in the following.

**Algorithm 1** PIC scheme

**Require:** Particle positions and velocities $(\mathbf{x}_p, \mathbf{v}_p)$, time step $\Delta t$, external magnetic field $\mathbf{B}$, grid $\Omega_h$.

**for each** time step $k\Delta t$ **do**

**Accumulate** the charge density onto the grid.

**Compute** the electric field from the charge density on the grid according to:

$$\mathbf{E} = -\boldsymbol{\nabla}_h \Phi, \quad -\varepsilon_0 \Delta_h \Phi = \rho. \tag{1.25}$$

**Interpolate** the electric field from the grid to the phase space at the particle positions.

**Update** the particle velocities and positions according to:

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p, \qquad \frac{d\mathbf{v}_p}{dt} = \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v}_p \times \mathbf{B})|_{\mathbf{x}=\mathbf{x}_p}. \tag{1.26}$$

**end for**

### 1.3.3 Update of the particles: method of characteristics

The principle is to transform the Vlasov equation, which is a partial differential equation with respect to variables $\mathbf{x}, \mathbf{v}, t \in \Omega_x \times \Omega_v \times \mathbb{R}^+$, into an ordinary differential equation. Let us introduce the functions $\mathbf{X} : \mathbb{R} \to \mathbb{R}^d$, $\mathbf{V} : \mathbb{R} \to \mathbb{R}^d$ so that the Vlasov equation can be recovered according to the relation:

$$\begin{aligned}\frac{d}{dt} f(\mathbf{X}(t), \mathbf{V}(t), t) = \frac{d}{dt}\mathbf{X}(t) \cdot \nabla_{\mathbf{x}} f(\mathbf{X}(t), \mathbf{V}(t), t) + \frac{d}{dt}\mathbf{V}(t) \cdot \nabla_{\mathbf{v}} f(\mathbf{X}(t), \mathbf{V}(t), t) \\ + \frac{\partial}{\partial t} f(\mathbf{X}(t), \mathbf{V}(t), t),\end{aligned}$$

where the notations $\nabla_{\mathbf{x}}$ and $\nabla_{\mathbf{v}}$ stand for the gradients with derivatives related to the space or the velocity domain. Therefore, the solutions of the Vlasov equation can be obtained by resolving

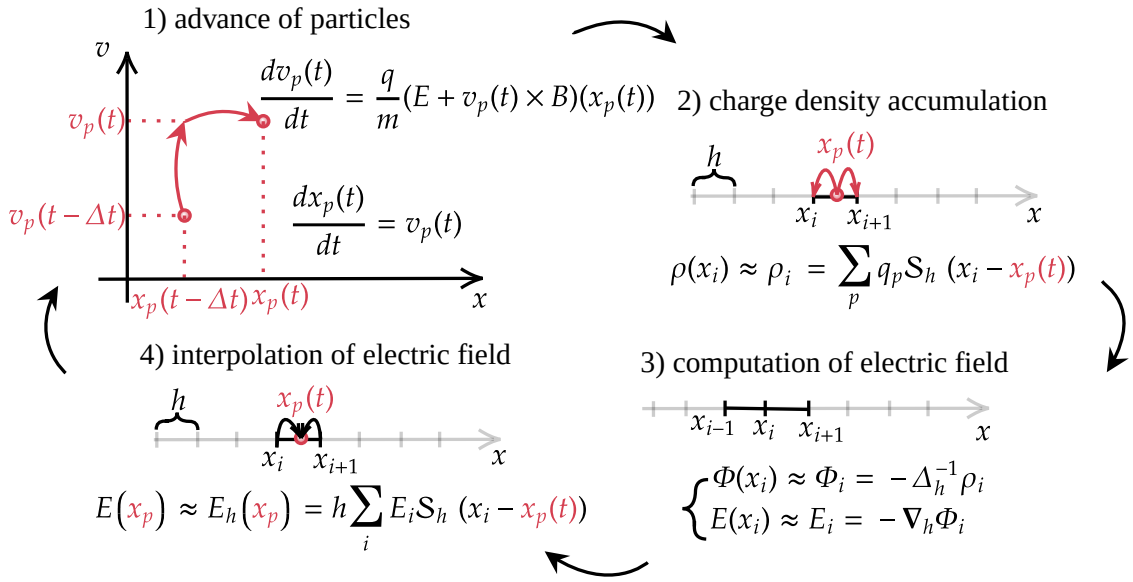


**Figure 1.1.** Schematic depiction of a uni-dimensional explicit PIC scheme: time $t - \Delta t$ to $t$.

the following system of ordinary differential equations:

$$\frac{d}{ds}\mathbf{X}(s) = \mathbf{V}(s) \tag{1.27}$$

$$\frac{d}{ds}\mathbf{V}(s) = \frac{q}{m}(\mathbf{E}(\mathbf{X}(s), s) + \mathbf{V}(s) \times \mathbf{B}(\mathbf{X}(s), s). \tag{1.28}$$

Vlasov equation yields the crucial result that if a distribution function $f$ is a solution of the Vlasov–Poisson system, then $f$ is constant along the characteristics of the Vlasov equation because $\frac{d}{dt}f(\mathbf{X}(t), \mathbf{V}(t), t) = 0$. Then in order to follow the evolution of the particle, i.e. the distribution function $f$ in time, one has to follow the characteristics and resolve the system (1.27)-(1.28), which can be discretized, for a vanishing magnetic field in an electrostatic regime, *e.g.* with a Euler explicit scheme:

$$\begin{cases} \mathbf{X}^{k+1} = \mathbf{X}^k + \Delta t \mathbf{V}^k, \\ \mathbf{V}^{k+1} = \mathbf{V}^k + \frac{q}{m}\Delta t \mathbf{E}^k(\mathbf{X}^k), \end{cases} \tag{1.29}$$

or a leap frog scheme:

$$\begin{cases} \mathbf{V}^{k+\frac{1}{2}} = \mathbf{V}^k + \frac{q}{m}\frac{\Delta t}{2}\mathbf{E}^k(\mathbf{X}^k), \\ \mathbf{X}^{k+1} = \mathbf{X}^k + \Delta t \mathbf{V}^{k+\frac{1}{2}}, \\ \mathbf{V}^{k+1} = \mathbf{V}^{k+\frac{1}{2}} + \frac{q}{m}\frac{\Delta t}{2}\mathbf{E}^{k+1}(\mathbf{X}^{k+1}), \end{cases} \tag{1.30}$$

or Runge-Kutta schemes, etc. The Euler explicit scheme is a first order method, that is to say the error scales with $O(\Delta t)$ whereas the leap frog scheme is a second order method and the error scales with $O(\Delta t^2)$. In the following of this manuscript the leap frog scheme is considered for explicit schemes. The stability of the scheme is studied with the Von Neumann analysis, *i.e.* by linearizing the equation and using Fourier analysis. The linear dispersion relation of the leap

frog for a cold plasma with no drift [71, 18] leads to:

$$\left(\frac{\Delta t}{2}\right)^2 \omega_p^2 = \sin^2\left(\omega\frac{\Delta t}{2}\right), \tag{1.31}$$

where $\omega_p = \sqrt{q_e n_0/m_e\varepsilon_0}$ is the plasma period. For $\omega_p\Delta t > 2$, the dispersion relation of the leap frog algorithm cannot be satisfied in the real axis, as the sin function amplitude cannot exceed 1. The resulting solutions for $\omega$ are complex conjugate, with one giving rise to an unphysical growth of the solution amplitude, characteristic of numerical instabilities. The stability constraint associated to the leap frog scheme is therefore:

$$\Delta t < \frac{2}{\omega_p}. \tag{1.32}$$

### 1.3.4 Charge density accumulation

In this manuscript, this step is equivalently referred to as the charge density accumulation or projection. The method is introduced here for the Cartesian grid of level $n \in \mathbb{N}^*$ (with an uniform discretization $h_n = 2^{-n}$). It will be extended to the more general framework of the component grids in the sequel.

Considering only one type of particle, the approximation of the distribution function defined in equation (1.24) yields the following approximation of the charge density:

$$\begin{aligned}\rho_N(\mathbf{x},t) = q\int_{\Omega_v} f_N(\mathbf{x},\mathbf{v},t)d\mathbf{v} &= q\sum_{p=1}^{N}\omega_p\delta(\mathbf{x}-\mathbf{x}_p(t))\int_{\Omega_v}\delta(\mathbf{v}-\mathbf{v}_p(t))d\mathbf{v}\\ &= \sum_{p=1}^{N} q_p\delta(\mathbf{x}-\mathbf{x}_p(t)).\end{aligned} \tag{1.33}$$

As a sum of Dirac distributions, the charge density is a positive distribution and thus it can be identified as a Radon measure, i.e. a linear form on the space of continuous functions with compact support $C_c^0(\Omega_x\times\mathbb{R}_+^*)$. The idea to approximate the density is to substitute the Dirac distribution, which is numerically impractical, by a continuous function with compact support.

Let us consider, as an ersatz of the Dirac distribution, a $d$-dimensional shape function based on the Cartesian grid discretization, denoted $\mathcal{S}_{h_n}$, which is constructed by tensor products of one dimensional hat functions:

$$\mathcal{S}_{h_n}(\mathbf{x}) := \left(\bigotimes_{i=1}^{d}\mathcal{S}_{h_n}\right)(\mathbf{x}), \quad \mathcal{S}_{h_n}(x) := h_n^{-1}W\left(h_n^{-1}x\right), \quad W(x) = \max\left(1-|x|,0\right). \tag{1.34}$$

Let $\bar{\mathcal{S}}_{h_n} : \Omega_x\times\Omega_x \to \mathbb{R}$ be defined by:

$$\bar{\mathcal{S}}_{h_n}(\mathbf{x},\mathbf{y}) = \mathcal{S}_{h_n}(\mathbf{x}-\mathbf{y}). \tag{1.35}$$

Then, for all $\mathbf{x}\in\Omega_x$, $\bar{\mathcal{S}}_{h_n}(\mathbf{x},\cdot)$ is a continuous function with compact support so that we can define an approximation of the electrical charge density, denoted $\rho_{h_n,N}$, by the relation:

$$\rho_{h_n,N}(\mathbf{x}) := \langle\rho_N, \bar{\mathcal{S}}_{h_n}(\mathbf{x},\cdot)\rangle = \sum_{p=1}^{N} q_p\mathcal{S}_{h_n}(\mathbf{x}-\mathbf{x}_p(t)). \tag{1.36}$$

Following a Monte-Carlo approach like in [99], the charge density approximation can alternatively be derived as a statistical estimator of the density. This Monte-Carlo approach is investigated for the accumulation on the component grids later in the manuscript. The local error between the density and its approximation can be recast into a grid-based error and a particle sampling error:

$$\rho_{h_n,N} - \rho = \underbrace{(\rho_{h_n,N} - \mathbb{E}(\rho_{h_n,N}))}_{\mathcal{V}(\rho_{h_n,N})} + \underbrace{(\mathbb{E}(\rho_{h_n,N}) - \rho)}_{\text{Bias}(\rho_{h_n,N})}, \tag{1.37}$$

where the square root variance of the particle sampling error and the grid-based error are given by:

$$\mathbb{V}(\rho_{h_n,N})^{\frac{1}{2}} = \left(\frac{2}{3}\right)^{\frac{d}{2}} \left(\frac{Q\rho}{Nh_n^d}\right)^{\frac{1}{2}} + O\left(N^{-\frac{1}{2}}\right), \quad \text{Bias}(\rho_{h_n,N}) = \frac{h_n^2}{12}\left(\sum_{i=1}^{d} \partial_i^2 \rho\right) + O\left(h_n^4\right), \tag{1.38}$$

$Q = \int_{\Omega_x} qn(\mathbf{x})d\mathbf{x}$ being the total charge of the system. The convergence of the estimator, *i.e.* the bias and the variance, requires respectively:

$$\lim_{n\to+\infty} h_n = 0, \qquad \lim_{n,N\to+\infty} Nh_n^d = +\infty. \tag{1.39}$$

Therefore an extremely large number of particles $N$ is necessary in order to achieve a low statistical error when refining the mesh size. In practice, the statistical noise is generally the most detrimental component of the error to the precision of the numerical approximation. In order to control the amount of statistical noise in the simulation, we refer to $P_c$, the average number of particles per cell. The number of numerical particles is provided by:

$$N = P_c h_n^{-d}, \tag{1.40}$$

setting a mean number of particle per cell in the simulation.

### 1.3.5 Computation of the electric field and interpolation

From the approximation of the charge density on the mesh introduced in the previous section, the electric field can be computed using mesh based methods such as finite differences (FD), finite element methods (FEM), or fast Fourier transform (FFT). In this manuscript, a finite difference approach is considered. Discretizing the Poisson equation with the finite difference discrete operators (introduced in equation (17)):

$$-\varepsilon_0 \Delta_{h_n} \Phi_{h_n} = \rho_{h_n,N}, \tag{1.41}$$

an approximation of the electric potential, denoted $\Phi_{h_n}$ is obtained by solving a linear system of size $2^{nd} \times 2^{nd}$. The electric field is obtained by applying the gradient finite difference operator to the electric potential:

$$\mathbf{E}_{h_n} = -\boldsymbol{\nabla}_{h_n} \Phi_{h_n}. \tag{1.42}$$

Finally, the electric field is interpolated at the particle positions according to:

$$\mathbf{E}(\mathbf{x}_p(t)) = \mathcal{I}_{h_n} \mathbf{E}_{h_n}(\mathbf{x}_p(t)), \tag{1.43}$$

where $\mathcal{I}_{h_n}$ is an interpolation operator based on the Cartesian grid discretization, $h_n$; an example of interpolation operator is given later in the manuscript.

## 1.4 Computer science and High-Performance-Computing (HPC) background

> *"A computing system is a (usually highly composite) device, which can carry out instructions to perform calculations of a considerable order of complexity—e.g. to solve a non-linear partial differential equation in 2 or 3 independent variables numerically."*
>
> J. Von Neumann, *First Draft of a Report on the EDVAC*, 1945.

Traditionally, computer science consists in the elaboration of algorithms executed in a serial stream of instructions, meaning that only one instruction is executed at a time. Since the early days of computer science, computing systems have been specifically designed to perform these kind of computations, following the well-known von Neumann model. In 1945, von Neumann [111] prescribed the architecture of one of the first stored-program computers, EDVAC. Since that time, most of the computers developed have followed its structure, which include:

- a Central Arithmetical (CA) part performing the elementary operations and a Central Control (CC) organizing the proper sequencing of the operations. These two components are nowadays more commonly known as Central Processing Unit (CPU).
- a Memory (M) that store the data and instructions.
- an outside recording medium (R), which is an external storage of the device.
- an organ that transfer information from R to its specific parts (CA,CC,M), which is called Input (I), and reciprocally one for transfers from CA,CC,M to R, called Output(O).

At the beginning of computing systems, the performance of the applications used to be most of the time dependent of the CPU speed (*i.e.* compute-bound). This is mostly due to the simple path between data and the low frequency of the processors. The frequency of a processor, or clock rate, is the frequency at which the clock generator of a process can generate pulses. It is often use to measure the speed of a processing unit.

**Definition 1.4.1 (Compute-bound, memory-bound)** *A program is said to be compute-bound when the performance, with respect to the runtime, is limited by the calculations and the speed of the central processor. On the opposite, a program is said to be memory-bound when the performance is limited by the number and the nature of memory accesses, that is when the runtime is dominated by the time required to move data.*

For years, computer performances have been enhanced by increasing the processors' frequency. An increase in frequency decreases runtime for all compute-bound programs. For instance, the first generation of computers has processor's frequency in range of hertz or kilohertz while nowadays it is measured in gigahertz. The increase of processors' frequency has ceased in May 2004, when Intel has canceled the development of their last microprocessor Tejas who was expected to achieve a clock rate range of 7-10 GHz. The project was abandoned due to power consumption and overheating issues. This event marks the end of the paradigm of frequency scaling and the emergence of multi-core processors, which are processor containing multiple cores that are independent and can access the same memory concurrently. With the

appearance of such architectures and, to fully profit from their potential, parallel computing has emerged and has quickly become predominant in computer science.

As a result of the huge developments made for years on the efficiency of processors (*e.g.* by increasing their frequency or their number of instructions per clock) and the emergence of supercomputers, modern computers are most of the time limited by the memory bandwidth (data availability). Therefore, limiting the number of memory accesses and optimizing each of them is the key to achieve high performance on modern computers. In order to limit the number of memory accesses, a lot of strategies have been conceived, based on data buffering, memory pre-fetching, branch predictors, etc. All modern computers are conceived to apply these strategies and include different levels of *cache memory* (L1, L2, L3, etc.) to increase the data availability.

**Definition 1.4.2 (Cache-memory)** *A cache is a hardware memory component associated to a computer CPU that reduces the average cost to access data from the main memory. Specifically, a cache stores data that can be required at future requests of the CPU so that it can access to it faster. The data stored results either from an earlier computation or from a copy of data. A cache memory is smaller, faster and closer to a processor core than the main memory.*

### 1.4.1 Parallelization

> *"Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously."*
>
> A. Gottlieb, G.S.Almasi, *Highly Parallel Computing*, 1989.

The idea of parallel computing has emerged to overcome the limitations of classical computer science and has followed the development of multi-core processor architectures.

> *"I do not think people have a good understanding yet of parallelism and what it is going to buy us. We have to develop ways of thinking about parallelism and languages for expressing parallel algorithms. There will be major activity in that area for 5 or 10 years to develop the science base that's needed to exploit it."*
>
> J. Hopcroft, *An interview with the 1986 A.M. Turing Award recipients*, 1987.

There exist multiple types of these architectures whose differences are based on the management of data and more specifically the organization of the mass (global) memory, the main classes being *distributed memory architectures* and *shared memory architectures*. Distributed memory systems are multiprocessor computer systems in which the processors have different private global memories. Shared memory systems are multiprocessor computer system in which all the processors can access the same (global) memory concurrently. With multi-core processors, the management of the data become more critical and nowadays a lot of applications are memory-bound. Therefore, a particular attention shall be paid to the efficiency of the memory transfers. The two most important notions to estimate the efficiency of a memory transfer are the bandwidth and latency. Though they are often confused with each other, they do not measure the same quantity.

**Definition 1.4.3 (Bandwidth)** *Bandwidth is the maximum amount of data you can manage in one memory transfer. It is measured in bytes per second [B/s].*

**Definition 1.4.4 (Latency)** *Latency is the amount of time it takes for data to reach a remote location and return to you. It measures the delay of memory transfers and is measured in second [s].*

Shared memory architectures are multiprocessors sharing a global memory: all processors have access to the same memory. The communications between the tasks executed on different processors are made by writing to and reading from the global memory. The coordinations and synchronizations between the processors is also performed via the global memory. The main advantage of shared memory architectures is the fast access to the data for all the processes, the latency is significantly reduced for these architectures in comparison to distributed memory architectures. Shared memory systems are various and many types exist such as:

- Uniform Memory Access (UMA) in which all the processors share the physical memory uniformly. That is to say each processor access memory with the same time, *i.e.* the latency is equal for all the processors.
- Non-Uniform Memory Access (NUMA) in which the memory access time depends on the memory location relative to a processor. For NUMA architectures the notion of data locality is crucial: a processor can access its own local memory faster than non-local memory (memory local to another processor or memory shared between processors), *i.e.* the latency is different between the processors.

## 1.5 State of the art, motivation and main contributions

### 1.5.1 State of the art

PIC method is one of the most broadly used numerical methods in the simulation of plasmas. A lot of efficient implementations in various plasma areas have been developed to perform numerical experiments. Some examples of popular implementations are provided:

- PICSAR (PArticle-In-Cell Scalable Application Ressource) [110].
  website: https://picsar.net/
- PIC-Vert: a Particle-in-Cell implementation for multi-core architectures [8].
- SMILEI (Simulating Matter Irradiated by Light at Extreme Intensities) [81, 52].
  website: https://smileipic.github.io/Smilei/
- TRISTAN (TRIdimensional STANford code) [86], par-T (PARallel Tristan).
  website: https://ntoles.github.io/tristan-mp-pitp/
- etc.

The popularity of PIC methods is explained by their simplicity, ease of parallelization and robustness. Nonetheless, PIC schemes still contain a significant weakness: the statistical error originating from the sampling of the distribution function by a limited number of numerical particles. This numerical noise decreases slowly with the increase of the average number of particles per cell. Therefore, a large number of particles may be required, necessitating tremendous computational resources, specifically for three dimensional simulations for which the desired precision may impose a number of cells as large as $10^9$, the number of particles exceeding $10^{11}$. This substantial number of particles is required to ensure a good statistical resolution in the simulation and achieve a fair representation of the particle distribution function. The storage requirements for two-dimensional and three-dimensional simulations up to a number

of $2^{10}$ (1024) grid cells in each direction and a mean number of particles per cell between 75 and 500 is provided in figure 1.2. The substantial memory requirements for three-dimensional PIC simulations (in GB or TB) clearly show the challenge of performing computer simulations for complex physical problems with these methods. In table 1.3, estimations of the grid-based error and the particle sampling error are provided for simulations corresponding to a perturbation of an equilibrium state with different mesh and particle configurations. In the examples provided here, as well as in the vast majority of simulations, the error is dominated by the particle sampling error. This is caused by the slow convergence of the statistical estimator (in $O\left(1/\sqrt{P_c}\right)$, where $P_c$ is the mean number of particles per cell) compared to the convergence of the grid-based error (in $O\left(h_n^2\right)$, where $h_n = 2^{-n}$ is the grid discretization). Indeed, an equivalent grid-based and particle sampling error would require an absurdly large number of particles. Consequently, PIC simulations are practically restricted to configurations with a dominant non negligible statistical error deteriorating the accuracy of the simulation [102].

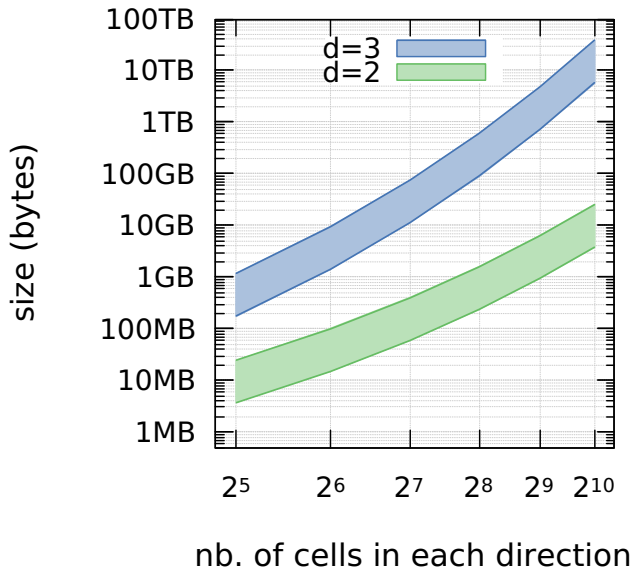


**Figure 1.2.** Memory storage footprint (in bytes) of the particle data for two dimensional and three dimensional PIC simulations, with a statistical error corresponding to $75 \leq P_c \leq 500$.

**Table 1.3.** Estimation of grid-based and particle sampling (noise) errors of the charge density in standard PIC simulations for a perturbation of an equilibrium state.

| $d$ | $n$ | $P_c$ | $N$ | noise | grid error |
|---|---|---|---|---|---|
| 2 | 6 | 75 | 3.07E+5 | 5.1320E−2 | 2.0345E−5 |
| 2 | 6 | 500 | 2.04E+6 | 1.988E−2 | 2.0345E−5 |
| 2 | 8 | 500 | 3.28E+7 | 1.988E−2 | 1.2715E−6 |
| 2 | 8 | 5000 | 3.28E+8 | 6.285E−3 | 1.2715E−6 |
| 3 | 6 | 500 | 1.31E+8 | 2.434E−2 | 2.0345E−5 |
| 3 | 8 | 500 | 8.39E+9 | 2.434E−2 | 1.2715E−6 |
| 3 | 8 | 5000 | 8.39E+10 | 7.698E−3 | 1.2715E−6 |

Noise reduction strategies aim at maintaining the accuracy of computations with a reduced set of particles. They have therefore received a lot of attention with, for instance, variance reduction methods such as the $\delta f$ method [51] or the quiet start initialization procedure [102] as well as filtering methods in either Fourier domain [18], wavelet domain [60], micro-macro decomposition [40] or variants [109].

In [97], the sparse grid techniques have been applied for the first time to a PIC scheme. Yet, the use of sparse grids for plasma applications is not exclusive to this work and has already been investigated many times in the literature [2, 66, 68, 79]. The motivation of the authors in [97] was to conceive a method whose complexity is nearly independent of the dimension of the problem. The method proposed is based on the sparse grid *combination technique*, an alternative characterization of the traditional and mostly used hierarchical-basis representation of sparse grids. Specifically, the combination technique provides an accurate representation of a function approximated on a variety of grids with coarse discretizations, and different resolutions

in each dimension. The grids considered in the combination technique are called *component grids*, and have been defined by equation (11). The accurate reconstruction is obtained with a specific linear combination of each coarse approximation. With an intelligent choice of the combination coefficients, one can achieve accuracy close to that of a well-resolved regular grid, that is the Cartesian grid defined by equation (15), but at a dramatically reduced cost. One crucial feature of the method is the large size of the component grid cells in comparison to those of the Cartesian grid, resulting in a significant increase of the number of particle per cell. This entails an improvement of the statistical resolution, without increasing the overall number of particles.

> *In this initial study, we have presented the use of the sparse grid combination technique in concert with PIC methodology. Although the method is still early in the development stage, initial results presented here demonstrate the method's potential to accelerate large scale kinetic plasma simulations.*
>
> L.F. Ricketson, A.J. Cerfon, *Sparse grid techniques for particle-in-cell schemes*, 2016.

The authors in [97] conclude on the memory requirement benefits of the method, which is manifest especially for three dimensional geometries. The conclusion on the computational time gain provided by the method is less evident, partly caused by the relatively simple implementation used. The authors also point the need to better understand the interplay between the statistical error and the field solve in the sparse grid context.

Following the work of [97], the sparse grid combination technique application to PIC method, which shall be named *sparse-PIC method* in the following of this manuscript, has been studied in the context of two-dimensional low temperature plasmas in [57, 58]. The simulation of plasmas in such conditions with explicit PIC methods is very challenging due to computational time constrains related to resolving both the electron Debye length in space and a fraction of the inverse plasma frequency in time. The authors have observed, for two-dimensional computations, that the sparse-PIC approach accurately reproduces the plasma profiles as well as the energy distribution functions compared to the standard PIC method.

For the work of this thesis, no existing implementations of PIC nor sparse grids methods have been used. All the developments and results presented in this manuscript were provided by implementations conceived and designed specifically for this work. All external implementations used, such as libraries, are listed in the numerical chapter 6.

### 1.5.2 Objectives and plan

The content of the previous section summarizes the context and the state of the art that could be drawn at the beginning of this thesis. The recent introduction of sparse grid techniques to PIC methods appeared promising. However, as pointed out in the seminal work [97], the method was still at its early stage of development and, as a result, the novelty of the method naturally raised numerous questions that were not answered. The motivation of the work presented in this manuscript is therefore to answer some of these issues. The rest of the manuscript is divided into five chapters: the first four ones (chapters 2, 3, 4, 5) correspond to the investigations of different kind of issues and the last one (chapter 6) gathers all the numerical results related to these issues.

**I. Properties of sparse grid reconstructions (chapter 2)**

The first questions raised were related to the accuracy and the faithfulness of the method:

**QI.1. What is the error made on the electric field carried out with sparse grid reconstructions embedded in PIC methods ? How is it close to the electric field computed with standard PIC methods ?**

Since the motion of particles is prescribed by the electric field, this is a fundamental question that must be answered in order to trust simulations embedding sparse-PIC reconstructions. It is well-known that explicit formulations of PIC methods conserve exactly the total momentum of the system. On the other hand, the total energy of the system is not exactly conserved, but depends on the grid discretization. If the grid discretization is smaller than the Debye length, the error of the energy conservation has small effects on the simulation in standard PIC methods.

**QI.2. Does the sparse-PIC methods conserves exactly the total momentum of the system ? Is the loss of total energy conservation controlled by the grid discretization as for standard explicit PIC schemes ?**

In [97], the authors observed that for some applications, the sparse-PIC methods fail to reproduce a precise approximation of the solution.

**QI.3. Is it possible to design alternative sparse-PIC methods or corrections of the existing methods to provide a better approximation ?**

First, as a prelude to the answers of these questions, we thought that a thorough description, as well as a summary, of the different sparse-PIC methods should be profitable to the community. In addition to this presentation, the questions **QI.1.**, **QI.2.** and **QI.3.** are answered in the chapter 2 of the manuscript and some of the results obtained lead to the following publication:

[48] Fabrice Deluzet, Gwenael Fubiani, Laurent Garrigues, Clément Guillet and Jacek Narski, *Sparse grid reconstructions for Particle-In-Cell methods*, ESAIM: M2AN, 56(5):1809–1841, September 2022.

In this article, the main motivation is to provide error estimates for sparse grid reconstruction methods. The error is separated into two contributions: a deterministic error (grid-based error) and a statistical error (particle sampling error). The first contribution to the error is thoroughly investigated, separated into several components depending on the solution derivatives, and bounds are provided for the grid-based error. On the other hand, the bounds provided for the particle sampling error are not optimal (especially for the electric potential and field), but sufficient to give an idea of the gain provided by sparse grid reconstructions.

**QI.4. Could we refine the particle sampling error estimations derived in [48] to better understand the benefit of sparse grid reconstruction methods ?**

The gain provided by the sparse grid reconstructions is characterized by the reduction of the number of particles required to achieve a given statistical resolution. Nonetheless, the number of particles used in sparse-PIC simulations is set by an heuristic relation, first introduced in [97] and derived from the average number of particles per cell used in tradition PIC simulations (see equation (1.40)).

**QI.5. Is it possible to provide an analytical justification for the heuristic relation prescribing the number of particles required in simulations ? If not, can we find another relation for that purpose ?**

The answers of theses questions are recent results, expected to be published in a future work and can be found at the end of chapter 2.

**II. CPU and shared memory implementation (chapter 3)**

Many issues concerning the efficiency of the method remain unanswered. Specifically, this aspect raises several questions. The optimization of the standard PIC schemes has been studied extensively for years [10, 11, 12] and a lot of efficient strategies have been conceived. According to [97], the benefits of the sparse-PIC method in term of memory requirements is manifest, but the benefit in term of computational time has not been clearly evidenced.

**QII.1. Are the optimizations developed for PIC methods also efficient for sparse-PIC methods, and are there novel strategies, more efficient, that may be designed specifically for sparse-PIC schemes ? Are sparse-PIC methods more efficient than the standard PIC methods regarding the computational time?**

As already discussed, the desire to understand more and more complex physical problems, together with the significant development of computer sciences during the last decades have massively increased the interest in developing numerical applications suited for supercomputers. In this thesis, we restricted ourselves to strategies designed for a single node of a supercomputer. Beyond this objective, an efficient implementation on shared memory architectures provides a building block for porting these methods on modern distributed memory architectures, where each node is a shared memory system with tens of cores. It is widely known that parallelization of PIC methods for shared memory architectures represents a considerable difficulty because of the numerous (random) memory accesses resulting from the large number of particles as well as the memory bound nature of PIC algorithm [9]. To address this issue, a lot of strategies has been developed [12, 9, 106, 52, 100, 110]. It naturally arises the questions of the portability of the sparse-PIC methods on shared memory platforms and its efficiency on such architectures. Since no application of the sparse-PIC method for shared-memory platforms was proposed at the beginning of this thesis, this aspect has raised the following questions:

**QII.2. To what extent the strategies proposed for standard PIC are efficient and relevant for sparse-PIC method ? Is it possible to conceive efficient parallelization strategies tailored for shared-memory architectures and specific to sparse-PIC methods that can tackle the memory-bound issue of PIC implementations ?**

Previous questions (**Q.II.1**,**Q.II.2**) are answered in chapter 3. The results obtained lead to the following publication:

[49] Fabrice Deluzet, Gwenael Fubiani, Laurent Garrigues, Clément Guillet and Jacek Narski, *Efficient parallelization for 3D-3V sparse grid Particle-In-Cell: shared memory systems architectures*

**III. GPGPU implementation (chapter 4)**

The last decades have seen the emergence of GPGPU applications and the appearance of accelerators with thousands of compute cores achieving substantial performance (in the range of several TFLOP/s). The interest in such architectures has therefore significantly increased as most of supercomputers now employ a lot of accelerators. One of the main difficulty with GPGPU programming is the management of the data between the host (CPU) and the device (GPU). The significant benefits of sparse-PIC methods observed on the memory footprint naturally motivate the use of accelerators. GPU parallelization is not new for PIC methods. A lot of implementations and different strategies have been conceived.

**QIII.1. Are the strategies proposed for PIC methods efficient for sparse-PIC applications on GPUs ? Are the strategies developed for sparse-PIC methods on shared memory architectures relevant for GPUs architectures ? If not, is it possible to design strategies specifically tailored for accelerators ?**

These questions (**Q.III.1**) are answered in chapter 4. The results obtained lead to the following publication:

[50] Fabrice Deluzet, Gwenael Fubiani, Laurent Garrigues, Clément Guillet and Jacek Narski, *Efficient parallelization for 3D-3V sparse grid Particle-In-Cell: single GPU architectures*.

**IV. Stability issues (chapter 5)**

The fourth point of interest during this thesis concerns the stability of the method. It is well known that standard explicit PIC methods are stable under constraints on the time step $\Delta t$, and the grid discretization $h_n$ (relative to the plasma period $\omega_p^{-1}$ and the Debye length $\lambda_D = \sqrt{\varepsilon_0 T_e / q_e n_e}$). Furthermore, explicit schemes do not conserve the total energy of the system and the stability is only linear (derived from a linearization of the equations). On the other hand, numerous implicit PIC methods alleviating these numerical constraints have emerged in the last decades [83, 30, 27].

**QIV.1. Are the numerical constraints for linear stability (time step, grid discretization) similar between the explicit sparse-PIC method and the standard PIC method ?**

A semi-implicit PIC scheme conserving exactly the total energy of the system (ECSIM) has been introduced in [83]. The non-linearity of the equations are linearized thanks to a specific time discretization. In the method, a mass matrix has to be computed, and the related linear system to be solved. The computational complexity is thus reduced in comparison to full implicit (non-linear) methods. The method is derived from Maxwell's equations in the context of electromagnetic fields.

**QIV.2. Is it possible to derive an electrostatic version of the ECSIM method?**

**QIV.3. Is it possible to merge the sparse grid reconstructions to the electrostatic/ electromagnetic ECSIM method and benefit from the advantages of both ?**

Theses questions are answered mostly in chapter 5 (**QIV.1** is answered in the chapters 2 and 6) but are still ongoing works.

### 1.5.3 Main contributions: answers to the questions

Let us now give a short answer for each of the question raised in the previous section. These answers define the main contribution of this thesis.

**I. Properties of sparse grid reconstructions (chapter 2)**

**AI.1. What is the error made on the electric field carried out with sparse grid reconstruction embedded in PIC method ? How is it close to the electric field computed with standard PIC methods ?**

Two main different sparse-PIC methods to reconstruct the electric field within Particle-In-Cell methods (PIC-Sg, PIC-Hg) have been first identified. They mainly differ in the

computation of the electric field. For both of them, the error and smoothness requirements of the solution have been explicited. The electric field error has been recast into two components: the grid-based error ($\text{Bias}(\mathbf{E}_h)$) and the particle sampling error ($\mathcal{V}(\mathbf{E}_h)$), where $\mathbf{E}_h$ is a shortcut notation for the approximation of the electric field. The results, detailed in the theorems 2.2.7 and 2.2.6, are summarized here. Even though, the grid-based and the particle sampling (square root of variance) error estimates provided are similar for the two schemes and scale as:

$$\text{Bias}(\mathbf{E}_h) = O\left(h_n^2 |\log h_n|^{d-1}\right), \tag{1.44}$$

$$\mathbb{V}\left(\mathcal{V}(\mathbf{E}_h)\right)^{\frac{1}{2}} \leq O\left((Nh_n)^{-\frac{1}{2}} |\log h_n|^{d-1}\right), \tag{1.45}$$

some differences between the schemes have been unraveled, especially for the dependencies on the cross derivatives of the solution:

- For the PIC-Hg scheme, the higher order term of the grid-based error depends only on the cross derivatives of the density:

$$\text{Bias}(\mathbf{E}_h^{Hg}) \propto \nabla \partial_1^2 ... \partial_d^2 \rho. \tag{1.46}$$

- For the PIC-Sg scheme, the higher order term of the grid-based error depends on the cross derivatives of the charge density and the electric potential:

$$\text{Bias}(\mathbf{E}_h^{Sg}) \propto \partial_1^4 ... \partial_d^4 \mathbf{E}, \nabla \partial_1^4 ... \partial_{d-1}^4 \partial_d^2 \rho, ... \tag{1.47}$$

  Note that the derivatives are of higher order than for the PIC-Hg scheme.

These estimates shall be compared to the ones obtained for the standard PIC method:

$$\text{Bias}(\mathbf{E}_h^{Std}) = O\left(h_n^2\right), \tag{1.48}$$

$$\mathbb{V}\left(\mathcal{V}(\mathbf{E}_h^{Std})\right)^{\frac{1}{2}} = O\left((Nh_n^d)^{-\frac{1}{2}}\right), \tag{1.49}$$

as well as the dependencies on derivatives of the solutions of the grid-based error higher order term:

$$\text{Bias}(\mathbf{E}_h^{Std}) \propto \partial_1^4 \mathbf{E}, ..., \partial_d^4 \mathbf{E}, \tag{1.50}$$

These estimates ascertain the gain brought by sparse grid reconstructions with respect to the statistical noise compared to standard methods. They also outline that the grid-based error is marginally less favorable with a dependence to cross derivatives for any of the sparse grid reconstructions.

**AI.2. Does the sparse-PIC methods conserves exactly the total momentum of the system ? Is the loss of total energy conservation controlled by the grid discretization as for standard explicit PIC schemes ?**

The conservation properties of the schemes have been investigated:

- The PIC-Sg scheme conserves exactly the total momentum of the system (see proposition 2.2.8) whereas the PIC-HG scheme does not.
- Both of the PIC-Hg and PIC-Sg schemes conserve exactly the total charge of the system (see proposition 2.2.3).

- None of the schemes conserve exactly the total energy of the system, but the increase has been observed to be similar to the standard PIC scheme on numerical examples.
- None of the schemes ensure the preservation of the positiveness of the solution, *e.g.* the reconstructed charge density is not nonnegative (see remark 2.2.4), but each component grid contribution is.

**AI.3. Is it possible to design alternative sparse-PIC methods or corrections of the existing methods to provide a better approximation ?**

Based on the analysis conducted on the different sparse-PIC methods, some techniques and corrections have been proposed to mitigate the major weaknesses of the method:

- The offset combination technique, which is a generalization of the truncated combination technique [88] and consists in both reducing the number of component grids considered within the combination and using component grids with increased minimum levels has been introduced. The motivation is to decrease the dominant component of the grid based error (related to the mixed derivatives) in sparse grid PIC methods. This framework permits to tune the balance between the different components of the error and improve the quality of PIC sparse grid approximations. This method has proven to achieve an improved numerical accuracy compared to existing sparse grid reconstructions on challenging configurations (see figure 1.4).
- Corrections for both the PIC-Hg and PIC-Sg methods, based on the analysis conducted for the different methods and consisting in oversampling the reconstructed charge density or enhancing the electric field on each component grid have been proposed in order to mitigate the major weakness of each method.

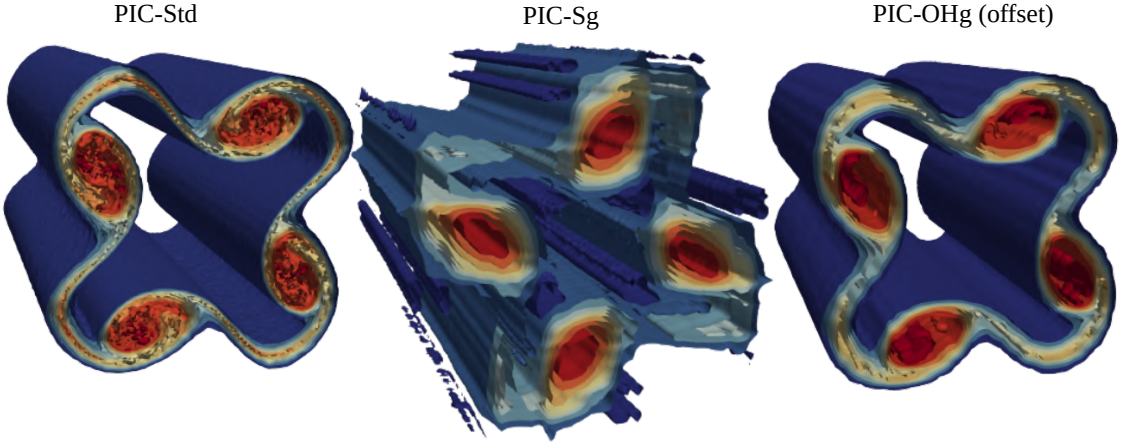


**Figure 1.4.** Representation of the electron density for a 3D-3V diocotron simulation, $h_n = 2^{-7}$, $P_c = 20$, corresponding, from left to right, to $N = 4.19\text{E+}7$, $N = 5.4\text{E+}5$, $N = 1.09\text{E+}7$.

**AI.4. Could we refine the particle sampling error estimations derived in [48] to better understand the benefit of sparse grid reconstruction methods ?**

The particle sampling error estimation of the recombined charge density has been refined (see theorem 2.2.15):

$$\mathbb{V}\left(\mathcal{V}(\rho_h)\right)^{\frac{1}{2}} \leq \begin{cases} O\left((Nh_n)^{-\frac{1}{2}} |\log h_n|^{d-1}\right) & \text{(old estimate)} \\ O\left((Nh_n)^{-\frac{1}{2}} |\log h_n|^{\frac{d-1}{2}}\right) & \text{(new estimate)} \end{cases} \tag{1.51}$$

where $\rho_h$ is a shortcut notation for the recombined charge density. Using Discrete Fourier Transform analysis to estimate the statistical noise in the simulation, a more accurate bound has been provided for the electric potential (see theorem 2.2.17), specifically:

$$\mathbb{V}\left(\mathcal{V}(\Phi_h)\right)^{\frac{1}{2}} \leq \begin{cases} O\left((Nh_n)^{-\frac{1}{2}}|\log h_n|^{d-1}\right) & \text{(old estimate)} \\ O\left(N^{-\frac{1}{2}}|\log h_n|^{d-1}\right) & \text{(new estimate)} \end{cases} \tag{1.52}$$

**AI.5. Is it possible to provide an analytical justification for the heuristic relation prescribing the number of particles required in simulations ? If not, can we find another relation for that purpose ?**

Let $P_c \in \mathbb{N}$ being an integer representing the mean number of particles per cell, and considering the total number of particles $N$ defined by the following equations for the sparse-PIC and standard schemes:

$$N_{std} = P_c h_n^{-d}, \quad N_{sparse} = P_c \left( \sum_{\mathbf{l}\in\mathbb{L}_n} |c_{\mathbf{l}}| h_{l_1} \dots h_{l_d} \right)^{-1}. \tag{1.53}$$

Then the particle sampling error on the charge density is proven to be comparable for both methods (see corollary 2.2.16).

**II. CPU and shared memory implementation (chapter 3)**

**AII.1. Are the optimizations developed for PIC methods also efficient for sparse-PIC methods, and are there novel strategies, more efficient, that may be designed specifically for sparse-PIC schemes ? Are sparse-PIC methods more efficient than the standard PIC methods regarding the computational time?**

First, a three dimensional parallel implementation of PIC method embedding sparse grid reconstruction has been designed. The efficiency of the implementation relies on novel developments and optimizations specific to sparse-PIC methods, including:

- A novel procedure for the sparse grid reconstruction, based on hierarchical subspace decomposition has been introduced. The method uses hierarchical basis functions for the combination and is different from the classical reconstruction using nodal basis functions. The reconstruction in hierarchical basis provides algebraically the same approximation than the reconstruction in nodal basis but it results from this procedure significant gains on computational time (for the interpolation of the electric field), the number of operations being reduced from $O\left(n^{d-1}2^{dn}\right)$ to $O\left(2^{dn}\right)$. The gains are particularly significant for three dimensional computations.
- A good memory management consisting in maximizing L1-cache reuse. It is well known that PIC methods suffer from irregular non-contiguous memory accesses when performing operations between the particles and the grid (*i.e.* charge density accumulation and field interpolation). Traditionally, the impact on efficiency is mitigated by converting a large number of these irregular memory accesses into contiguous accesses thanks to particle sorting. The particle array is periodically sorted so that contiguous cases of the array correspond to particles close to each other in the grid array. Nonetheless, sorting may be expensive for rapid particles dynamic. Thanks to the reduced size of the component grids ($O\left(2^{n+d-1}\right)$ nodes) in comparison to the full grid ($O\left(2^{dn}\right)$), one can benefit from the L1-cache memory

to mitigate the latency resulting from the irregular memory accesses. Indeed, for configurations up to $h_n = 2^{-9}$ (equivalent to a Cartesian grid with $512^3$ cells), the array containing any component grid entirely fits in the CPU L1-cache. This observation led us to develop an efficient implementation with an optimal L1-cache reuse and without any sorting of the particles.

The efficiency and performance of the novel implementation has been compared to a standard PIC implementation with traditional optimizations (*e.g.* sorting of particles) on several numerical configurations (test cases, hardware). Gains in terms of computational time have been demonstrated (see figure 1.5). We have observed that, for the sparse-PIC methods, the computational time is dominated by the cost of the charge density projection, which counts for roughly 90% of the total execution time (between 85% and 95% depending on the configuration). Indeed, thanks to hierarchization, the interpolation computational time is negligible, and the reduced number of particles is reflected in the small amount of computational time dedicated to the particle pusher .

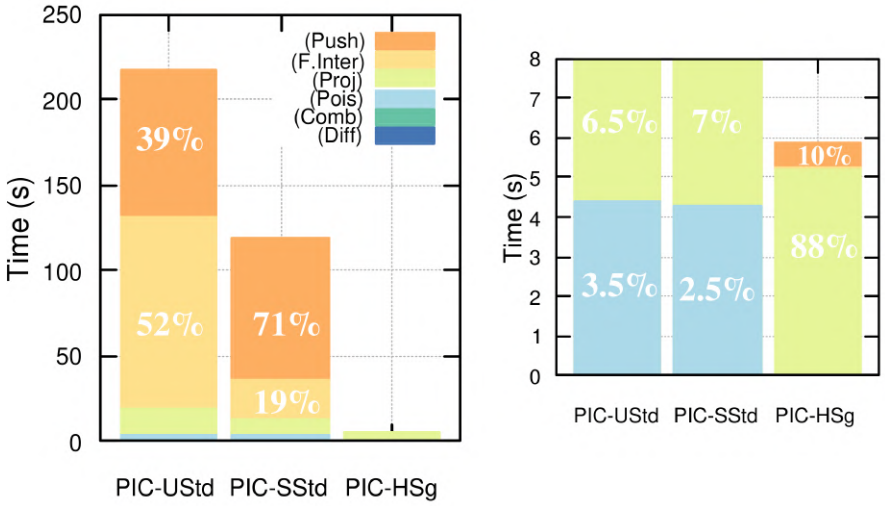


**Figure 1.5.** Computational time of one iteration for a three dimensional sequential execution of the PIC-UStd (without sorting of particles), PIC-SStd (with pre-sorting of particles, the time of sorting is not taken into account) and PIC-HSg scheme with comparable L2-norm errors on the electric field (Landau damping configuration).

**AII.2. To what extent the strategies proposed for standard PIC are efficient and relevant for sparse-PIC method ? Is it possible to conceive efficient parallelization strategies tailored for shared-memory architectures and specific to sparse-PIC methods that can tackle the memory-bound issue of PIC implementations ?**

The effort for the porting to shared memory architectures has been almost exclusively dedicated to the projection.

- Two strategies tailored to uniform (UMA) and non-uniform (NUMA) memory architectures have been conceived and mixed together to derive a two-levels paralelization strategy including coarse-grain and fine-grain parallelisms, achieving scalability close to optimal.

  i) The particle sample work sharing (coarse-grain) strategy: the particle population is divided into independent samples (one for each NUMA domain) and distributed onto the NUMA domains. Each core of a NUMA domain executes

operations related to the sample stored in the RAM local to the NUMA domain. This strategy takes full advantage of the (RAM) memory bandwidth increasing with the number of NUMA domains.

ii) The component grid work sharing (fine-grain) strategy: the component grids are distributed onto the cores within a NUMA domain. The operations (*i.e.* charge accumulation, resolution of Poisson equation,etc.) on different component grids are independent and executed concurrently by the cores of a NUMA domain. This strategy takes full benefit of the tiny memory footprint of the component grids. These arrays are stored in the L1-cache of the cores.

Eventually, a reduction operation, specific to NUMA domains and samples, is considered on the grid array (containing the value of the density on the grids). The overhead related to this operation is observed to be negligible because of the small number of grid nodes ($O\left(n^{d-1}2^n\right)$) in comparison with the number of particles. An important characteristic of the mixed strategy is that all the data written to memory (*i.e.* at the component grid nodes) are stored in the L1-cache, private to each core. Since these data accesses are usually non contiguous, the method benefits from the large bandwidth and low latency of the L1-cache memory. The only data that is accessed through the L2/L3 caches and RAM memory is the particle data which are read contiguously from the particle array and shared by all the cores within a NUMA domain. An illustration of the strategy is provided in figure 1.6.

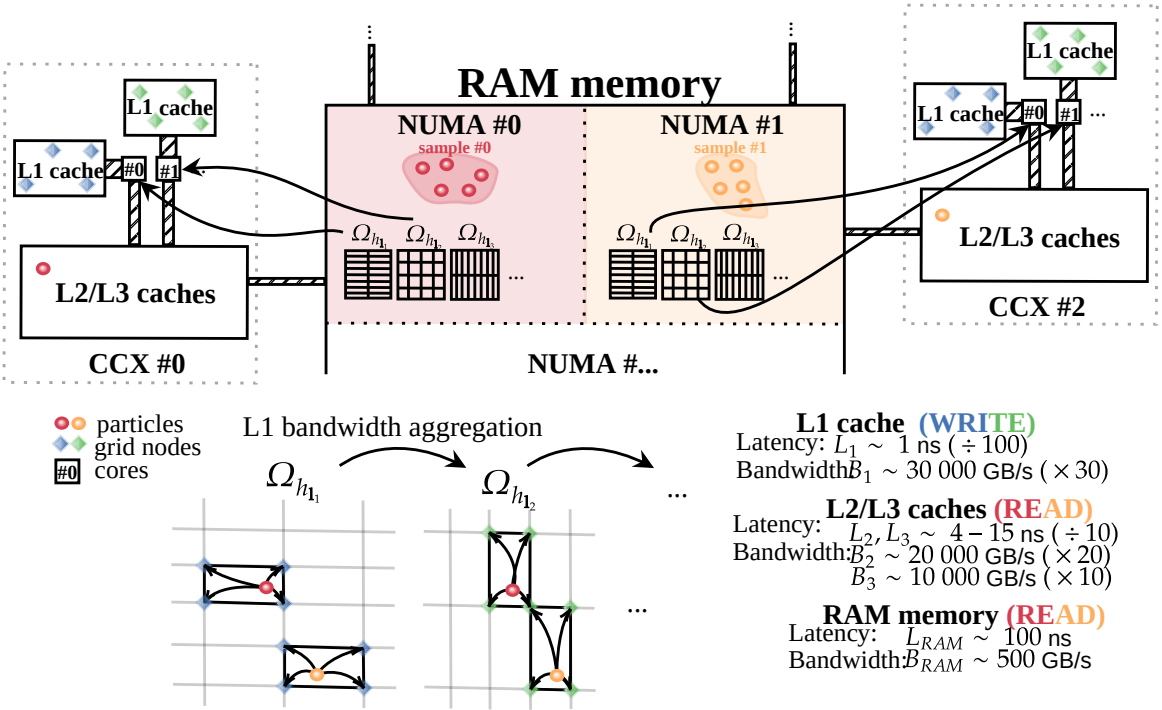


**Figure 1.6.** Charge accumulation parallelization strategy for NUMA architectures. The strategy is based on a decomposition of the data according to the type of access (read or write) and the memory hierarchy. All the data written to memory are in the L1-cache, increasing the bandwidth and decreasing the latency of the transfers. Bandwidth and latency are relative to the AMD EPYC™ 9004 (Genoa) architecture.

- A load balance strategy has been developed for UMA architectures to preserve the high scalability for general hardware configurations. Inside each NUMA domain, the particle sample associated to the domain is subdivided into as many clusters as

cores within the domain. The number of tasks, defined as the product of the number of component grids and number of clusters, is a multiple of the number of cores in the domain. The work load is therefore equally distributed onto the cores of the NUMA domain.

The efficiency of the parallelization is close to optimal (126/128) for the most costly step of the method (*i.e.* charge accumulation), and the global implementation achieves a speed-up of 100 on 128 cores (see figure 1.7).

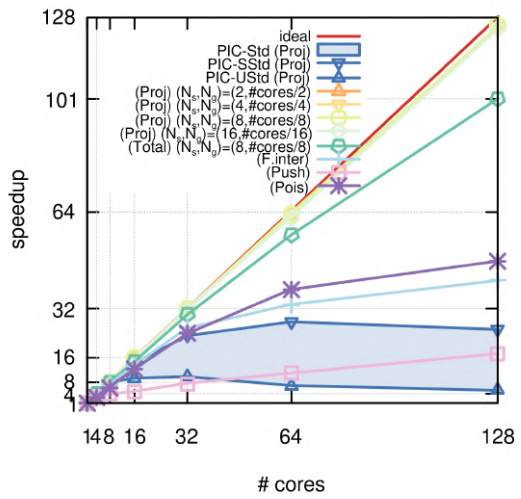


**Figure 1.7.** Strong scaling of different steps of the PIC-HSg scheme and the PIC-Std projection ($h_n = 2^{-7}$, $P_c = 500$) up to 128 cores. Different configurations of groups of grids and samples of particles ($N_s, N_g$) are represented. Computations carried out on either two AMD EPYC™ 7713 (Milan) CPUs.

### III. GPGPU implementation (chapter 4)

**AIII.1. Are the strategies proposed for PIC methods efficient for sparse-PIC applications on GPUs ? Are the strategies developed for sparse-PIC methods on shared memory architectures relevant for GPUs architectures ? If not, is it possible to design strategies specifically tailored for accelerators ?**

Traditionally, the efficient parallelization strategies for the standard PIC methods consist in particle sorting (on CPUs) and the use of shared memory (on GPGPUs). This is not a feature we enjoyed when porting sparse-PIC methods in GPGPU architectures. To benefit fully from the peak bandwidth of the GPU, the memory accesses shall be coalesced: consecutive threads shall access consecutive memory addresses. In order to achieve coalesced memory accesses with particle sorting, one sorting is required for each component grid at each iteration. This rules out the use of such strategies for sparse-PIC algorithms. The objective of this work was to unravel the genuine efficiency of sparse-PIC methods on generic GPUs rather than extract the maximal performance of a specific architecture. We therefore discarded CUDA implementation and choose on purpose a higher level API, namely OpenACC. This entails that a fine tuning of the GPU shared memory is not accessible for our implementation.

The NUMA parallelization implementation, referred to as CPU-inherited, offers poor performances on GPGPU architectures. This is explained by the fundamental differences

in the organization of the compute units in CPUs versus GPGPUs. Specifically, the CPU-inherited implementation is designed to optimize the L1-cache memory, private to each core. On GPUs, a large amount of compute units share the same L1-cache. Second, the NUMA implementation relies on a decomposition of the particle population into samples distributed onto the NUMA domains. This strategy reveals to be inefficient to organize a worksharing within the pool of Streaming Multiprocessors (SM) of the GPGPUs. This stems from the non coalesced memory accesses genuine to PIC algorithms. To mitigate the drops in performance, the SMs should be fed with a massive number of streams (independent tasks) to hide the memory latency. Porting the sampling worksharing strategies to GPGPUs would require reduction operation over thousand of arrays, penalizing the performance.

An implementation tailored to GPGPU architectures and parallelization strategies specific to sparse-PIC method has been developed. The key points of the charge accumulation implementation are the following:

- The implementation takes advantage of the significant reduction of memory footprint achieved by the sparse-PIC methods. All the data required during the simulation are stored on the GPU so that the only memory transfers between the host and the device are performed at the initialization and at the end of the computations. It is a crucial feature of the implementation since a lot of GPGPU applications are limited by the memory transfers between the host and the device (because of the limited bandwidth).
- The parallelization strategy is constituted of two levels of parallelism (coarse-grain parallelism and fine grain parallelism with a Single Instruction Multiple Thread (SIMT) execution model, see figure 1.8):
  i) particle work sharing (coarse-grain) strategy: the particle population is divided into clusters. A massive number of clusters (larger than the number of Streaming Multiprocessors (SM)), which can be as large 60k, is created. The clusters are distributed onto the SMs in a Single Program Multiple Data (SPMD) execution model, but are not bound to the SMs. The randomness of the memory accesses is mitigated thanks to the massive number of clusters that mask the latency of the non-coalesced memory accesses with computations: when a thread is stalled due to the unavailability of data (component grid nodes), a switch of context is operated to execute a different instruction stream with loaded data (interleave stream strategy).
  ii) Component grid work sharing (fine-grain) strategy inside the SMs: the component grids are processed at the same time by the different threads of a SM (SIMT execution model). The randomness of the memory accesses is also mitigated by reducing the memory accesses latency thanks to an optimization of the L2-cache of the GPU. Contrary to the CPU implementation, the interaction of one particle is computed with all the component grids at once, and repeated for all the particles. The data structure used to store all the component grids is small enough to be nursed into the L2-cache which contributes to extract good performances from the platform.

The GPGPU-specific implementation achieves speed-ups of 100 on a Tesla V100 (see table 1.10) compared to the CPU of the host.

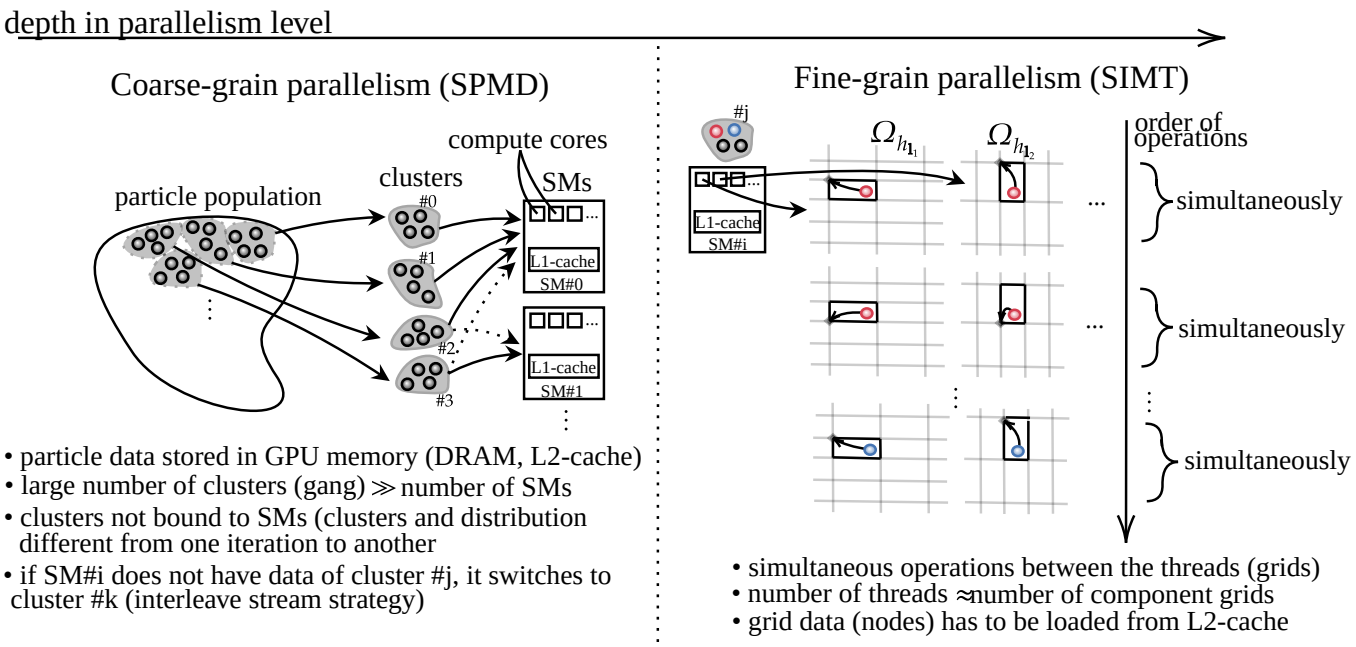


**Figure 1.8.** Parallelization strategy for charge accumulation on GPU. The strategy is based on two levels of parallelism: a coarse-grain level (SPMD execution model by means of the large number of clusters distributed onto the pool of SMs) and a fine-grain level (SIMT execution model within each SM).

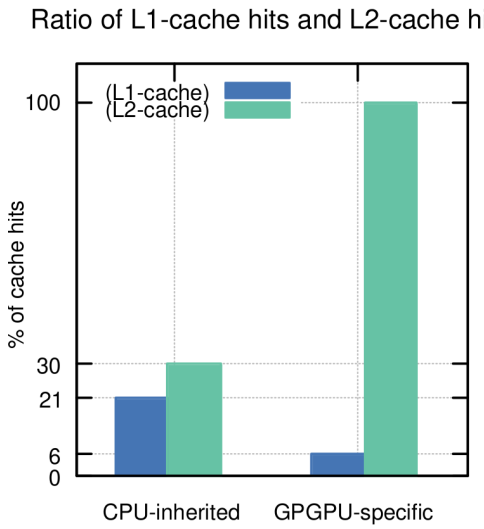


**Figure 1.9.** Relative amount of L1-cache and L2-cache hits for kernels running on the Tesla V100: The GPU parallel implementation is designed to maximize the L2-cache reuse.

**IV. Stability issues (chapter 6)**

**AIV.1. Are the numerical constraints for linear stability (time step, grid discretization) similar between the explicit sparse-PIC method and the standard PIC method ?**

The numerical constraints on the time step to guarantee the linear stability are the same for standard and sparse PIC methods, because the time step used to update the particles is

**Table 1.10.** Charge accumulation algorithm characteristics and performance on a Tesla V100, with $h_n = 2^{-7}$ and $P_c = 500$.

| Kernel | Effective performance [GFLOP/s] | Execution time [s] | Speed-up |
|---|---|---|---|
| CPU-inherited | 58.6 | 1.4 | 11.7 |
| GPU-specific | 213 (×4) | 0.1540 (÷10) | 106 |

similar for the two methods. On the other hand, the role played by the grid discretization on the stability is more intricate to analyze for the sparse grid methods. The stability has been investigated numerically and the same constraint as the standard scheme has been evidenced, *i.e.* the grid discretization must resolve the Debye length.

**AIV.2. Is it possible to derive an electrostatic version of the ECSIM method, which shall be cheaper than the original (electromagnetic) scheme?** Based on the divergence of the Ampere equation, an electrostatic ECSIM scheme has been derived. This method offers a genuine consistency with the constraint $\nabla \times \mathbf{E} = 0$ characteristic of the electrostatic regime. To the best of our knowledge this is the first numerical method embedding this property.

**AIV.3. Is it possible to merge the sparse grid reconstructions to the electrostatic/ electromagnetic ECSIM method and benefit from the advantages of both ?**

A sparse grid ECSIM scheme has been derived in an electrostatic regime, benefiting from the traditional ECSIM properties. Indeed, the scheme is implicit, *i.e.* no constraint on the time step is required for linear stability; the total energy is conserved exactly; the complexity of the scheme is reduced in comparison to fully implicit schemes (no non-linear system to solve). In addition, the scheme benefits from the sparse grid reconstructions: the particle sampling error is significantly reduced thanks to the projection of the current density on the component grids (with larger cells), resulting in a reduction of the number of particles for comparable statistical noise. The size of the linear system to solve is reduced because its size is based on the number of component grids nodes ($O\left(h_n^{-1}|\log(h_n)|^{d-1}\right)$) instead of the number of Cartesian grid for the original method ($O\left(h_n^{-d}\right)$). Nonetheless, like for the explicit sparse PIC method, the current density has to be accumulated onto $O\left(|\log(h_n)|^{d-1}\right)$ component grids, instead of one unique grid (note that the strategies based on L1 cache reuse introduced before hold for this scheme) and the fill-in of the resulting linear system is increased (the matrix has more non-zero entries). We have observed numerical instabilities manifested by the loss of the field energy positivity. The understanding and correction of this instability is still an ongoing work.

# Chapter 2

# Sparse grid reconstructions for Particle-In-Cell methods

*"The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience."*

Richard E. Bellman, *Adaptive Control Processes*, 1961

*Sparse grid* methods [24, 56] have been originally developed for the interpolation of high dimensional functions and extended to the approximation of partial differential equations [61, 23, 64, 62] with the aim of breaking the *curse of dimensionality*. The term, introduced by Bellman in [13], refers to the exponential dependence on the dimensionality $d$ of the problem for the cost of computing and representing an approximation of a function with a certain accuracy. Indeed for classic approximation methods, the complexity of the scheme scales as $O\left(\varepsilon^{-\alpha d}\right)$, where $\varepsilon$ is the prescribed accuracy, $\alpha > 0$ is a parameter depending on the approximation approach and the smoothness of the solution, and $d$ the dimension of the problem considered. For example, if we consider uniform grids with piecewise d-polynomial functions in a finite element or finite difference approach, the number of grid cells is $O\left(N^d\right)$, with a number of operations per cell scaling as $O(N^{-\alpha})$, where $\alpha$ depends on the smoothness of the solutions and the degree of the polynomial functions chosen. Hence the complexity of this procedure scales as $O\left(N^{-\alpha d}\right)$.

Recently sparse grids have been applied, in the framework of the so-called sparse grid *combination technique* [65, 63, 22], to PIC schemes [97, 88, 26, 58, 57]. The aim is to improve the properties of PIC methods with respect to the statistical error resulting from the particle sampling. In the sparse grid reconstructions, the numerical approximations are recomposed from partial representations carried out on a hierarchy of sparse grids with coarse resolutions. Compared to a regular Cartesian grid, the mean number of particles per cell is larger for any of the sparse grids. This crucial feature offers either a mitigation of the statistical noise or a decrease of the total number of numerical particles for a precision comparable to standard PIC discretizations. Besides, considering the thorough studies conducted during recent years to apply the combination technique to the resolution of PDEs, promising improvements in the computational efficiency are expected for the resolution of Poisson's equation (see [95, 22, 93]) providing the electric field in the PIC framework. Early implementations of sparse grids PIC schemes [97, 57, 58] show computational gains which are forecast substantial for three dimensional applications.

The objective of the chapter is to introduce PIC discretizations implementing the sparse grid combination technique, to conduct a formal analysis to explain their merits and weaknesses and support the development of new methods with improved efficiency.

The analyses of standard as well as sparse-PIC discretizations reveal that, for both methods, the approximation error may be decomposed into three contributions. The precision of the methods is characterized by the accuracy of the most probable value of the statistics associated to the particle sampling, this component being referred to as the bias. This is a grid-based error related to both the mesh size ($h$) and the smoothness of the solution with a component depending on the mixed derivatives of the solution and another contribution depending on non-mixed derivatives. The last error component is the so-called numerical noise or particle sampling error, providing the magnitude of the dispersion of the values attached to a sample of particles. The introduction of sparse grid reconstructions within PIC discretizations entails an increase of the grid error together with a significant mitigation of the statistical noise. This outlines the potential of these approaches: sparse grid reconstructions may be tailored to define different trades-off between the components of the error and finally mitigate the most detrimental one for the precision of PIC numerical approximations (the statistical noise). This leads to the derivation of the new sparse-grid methods introduced herein, with an improved numerical efficiency.

A specific attention is payed to the approximation of the electric field which can be computed thanks to two different approaches. The first one consists in computing the electric field on a refined Cartesian mesh thanks to the sparse interpolant of the charge density obtained by the combination technique. The second relies on a computation on each subgrid using the projected density. The electric field interpolant is then obtained by recombining the local approximants of the component grids. The analyses conducted in this section are aimed at providing error bounds for the electric field sparse grid interpolant and highlight the differences between these two approaches. The main results regarding the electric field, given by the propositions 2.2.7 and 2.2.6, point the strong dependance of the error on the mixed derivatives of the solution, especially for the second approach, which has proven to be more dependant of the smoothness of the solution. This is a major contribution since no convergence properties have already been proposed so far for the electric field, which is the critical quantity when determining the overall accuracy of PIC discretizations [104].

## Contents



# 2.1 Sparse grid techniques

## 2.1.1 Nodal basis and hierarchical basis representations

As a preliminary step to sparse grid techniques, one shall introduce some interpolation tools. Let $\mathbf{l} \in \mathbb{N}^d$, $\mathbf{j} \in I_{h_\mathbf{l}}$ be multi-indexes and consider basis functions defined by tensor products of

one-dimensional hat functions as follows:

$$W_{h_{\mathbf{l}};\mathbf{j}}(\mathbf{x}) := \left(\bigotimes_{i=1}^{d} W_{h_{l_i};j_i}\right)(\mathbf{x}), \quad W_{h_{l_j};j_i}(x) := W\left(h_{l_i}^{-1}(x_i - j_i h_{l_i})\right), \quad W(x) = \max\left(1 - |x|, 0\right), \tag{2.1}$$

where $h_{\mathbf{l}}$ is the grid discretization of the component grids defined in the notation section. These functions verify a partition of unity property:

$$\sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} W_{h_{\mathbf{l}};\mathbf{j}}(\mathbf{x}) = 1. \tag{2.2}$$

The space of $d$-dimensional hat functions with respect to the component grid $\Omega_{h_{\mathbf{l}}}$, denoted $V_{h_{\mathbf{l}}}$, is defined by:

$$V_{h_{\mathbf{l}}} := \text{span}\{W_{h_{\mathbf{l}};\mathbf{j}} \mid \mathbf{j} \in I_{h_{\mathbf{l}}}\}, \tag{2.3}$$

where $\{W_{h_{\mathbf{l}};\mathbf{j}} \mid \mathbf{j} \in I_{h_{\mathbf{l}}}\}$ is called the nodal basis of the space $V_{h_{\mathbf{l}}}$ and $I_{h_{\mathbf{l}}}$ the nodal basis index set. Additionally, we introduce hierarchical increments of $V_{h_{\mathbf{l}}}$ [24, 56], denoted by $U_{h_{\mathbf{l}}}$ and defined by:

$$U_{h_{\mathbf{l}}} := V_{h_{\mathbf{l}}} \setminus \bigoplus_{i=1}^{d} V_{h_{\mathbf{l}-\mathbf{e}_i}}, \quad \text{where } V_{h_{\mathbf{l}}} := 0 \ \text{ if } \exists i \in \{1, ..., d\} \ s.t. \ l_i = -1, \tag{2.4}$$

and $\mathbf{e}_i \in \mathbb{N}^d$ is the unit vector with the $i^{th}$ coordinate equal to one. The hierarchical increment contains all $W_{h_{\mathbf{l}};\mathbf{j}} \in V_{h_{\mathbf{l}}}$ that are not included in smaller $V_{h_{\mathbf{k}}}$, with $\mathbf{k} < \mathbf{l}$. It can also be expressed in the following form:

$$U_{h_{\mathbf{l}}} = \text{span}\{W_{h_{\mathbf{l}};\mathbf{j}} \mid \mathbf{j} \in \mathcal{B}_{h_{\mathbf{l}}}\}, \quad \mathcal{B}_{h_{\mathbf{l}}} := \left\{\mathbf{j} \in \mathbb{N}^d \mid \mathbf{0} \le \mathbf{j} \le h_{\mathbf{l}}^{-1}, \ \mathbf{j} \text{ odd}\right\}, \tag{2.5}$$

where $\{W_{h_{\mathbf{l}};\mathbf{j}} \mid \mathbf{j} \in \mathcal{B}_{h_{\mathbf{l}}}\}$ is called the hierarchical basis of the space $V_{h_{\mathbf{l}}}$ and $\mathcal{B}_{h_{\mathbf{l}}}$ the hierarchical basis index set. The space of piecewise d-linear functions of level $\mathbf{l}$ with respect to $\Omega_{h_{\mathbf{l}}}$ can be represented with its hierarchical basis:

$$V_{h_{\mathbf{l}}} = \bigoplus_{k_1 \le l_1} ... \bigoplus_{k_d \le l_d} U_{h_{\mathbf{k}}} = \bigoplus_{\mathbf{k}\le\mathbf{l}} U_{h_{\mathbf{k}}}, \tag{2.6}$$

Thus, each function $v_{h_{\mathbf{l}}} \in V_{h_{\mathbf{l}}}$ can be represented identically in the hierarchical basis of $V_{h_{\mathbf{l}}}$:

$$v_{h_{\mathbf{l}}} = \sum_{\mathbf{k}\le\mathbf{l}} \hat{v}_{h_{\mathbf{k}}} = \sum_{\mathbf{k}\le\mathbf{l}} \sum_{\mathbf{j}\in\mathcal{B}_{h_{\mathbf{k}}}} \alpha_{\mathbf{k},\mathbf{j}} W_{h_{\mathbf{k}};\mathbf{j}}, \tag{2.7}$$

or in the nodal basis of $V_{h_{\mathbf{l}}}$:

$$v_{h_{\mathbf{l}}} = \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \beta_{\mathbf{l},\mathbf{j}} W_{h_{\mathbf{l}};\mathbf{j}}, \tag{2.8}$$

where $\alpha_{\mathbf{k},\mathbf{j}}$ are the coefficients of $v_{h_{\mathbf{l}}}$ in the hierarchical basis, called hierarchical surplus, that shall be defined more precisely later; and $\beta_{\mathbf{l},\mathbf{j}}$ are the coefficients of $v_{h_{\mathbf{l}}}$ in the nodal basis which are the nodal values of the function $v_{h_{\mathbf{l}}}$. We introduce the space of $d$-dimensional piecewise

linear functions with respect to $\Omega_{h_n}^{(\infty)}$, denoted $V_{h_n}^{(\infty)}$.

$$V_{h_n}^{(\infty)} = \bigoplus_{|\mathbf{l}|_\infty \leq n} W_{\mathbf{l}} = \text{span}\{W_{h_{\mathbf{n}};\mathbf{j}} \mid \mathbf{j} \in \mathbb{N}^d \mid \mathbf{0} \leq \mathbf{j} \leq h_{\mathbf{n}}^{-1}\}. \tag{2.9}$$

Eventually, for $u$ a smooth function, we introduce the linear interpolation operators in nodal and hierarchical basis defined by:

$$\mathcal{I}_{V_{h_\mathbf{l}}}^{\mathcal{N}} u = \sum_{\mathbf{j} \in I_{h_\mathbf{l}}} u(\mathbf{j}h_\mathbf{l}) W_{h_\mathbf{l};\mathbf{i}}, \qquad \mathcal{I}_{V_{h_\mathbf{l}}}^{\mathcal{H}} u = \sum_{\mathbf{k} \leq \mathbf{l}} \sum_{\mathbf{j} \in \mathcal{B}_{h_\mathbf{l}}} \alpha_{\mathbf{k},\mathbf{j}} W_{h_\mathbf{k};\mathbf{j}}, \tag{2.10}$$

**Remark 2.1.1** *The linear interpolation in nodal and hierarchical basis provide the same approximation of a function $u$ and are equivalent, i.e. $\mathcal{I}_{V_{h_\mathbf{l}}}^{\mathcal{N}} u = \mathcal{I}_{V_{h_\mathbf{l}}}^{\mathcal{H}} u$. If not specified in the rest of the manuscript, the interpolation is assumed to be in nodal basis.*

### 2.1.2 Sparse grid combination technique

The sparse grid combination technique [65, 63, 22] is a method of interpolation using evaluations of the function on the nodes of component grids. The sparse grid interpolant is obtained by a linear combination of partial representations of the function on the component grids. Let us define the reconstruction of a given function $u$.

> *"The combination technique [...] uses the solutions of $O\left(\log(|h|^{-d})\right)$ different, on regular standard grids discretized problems with $O\left(h^{-1}\right)$ grid points and e.g. different mesh sizes in the x- and y-direction to produce a sparse grid solution. "*
>
> M. Griebel, M. Schneider, C. Zenger, *A combination technique for the solution of sparse grid problems*, 1992.

**Definition 2.1.2 (Reconstruction with the combination technique)** *Let $u$ be a function and $u_{h_\mathbf{l}}$ an approximation of this function in the space $V_{h_\mathbf{l}}$ (e.g. $\mathcal{I}_{V_{h_\mathbf{l}}} u$), then a sparse grid reconstruction, denoted $u_{h_n}^c$, is defined by linear combination of the contributions $u_{h_\mathbf{l}}$ of each component grid:*

$$u_{h_n}^c := \sum_{\boldsymbol{l} \in \mathbb{L}_n} c_{\boldsymbol{l}} u_{h_\mathbf{l}}, \quad \textit{where } c_{\boldsymbol{l}} = (-1)^i \binom{d-1}{i} \textit{ if } \boldsymbol{l} \in \mathbb{L}_{n,i}, \tag{2.11}$$

*and $\binom{d-1}{i} := \dfrac{(d-1)!}{i!(d-1-i)!}$ is the notation for the binomial coefficient.*

For two-dimensional and three dimensional spatial domains, the combination formula falls down to:

$$u_{h_n}^c = \sum_{|\mathbf{l}|_1 = n+1} u_{h_\mathbf{l}} - \sum_{|\mathbf{l}|_1 = n} u_{h_\mathbf{l}} \qquad \text{if } d = 2, \tag{2.12}$$

and:

$$u^c_{h_n} = \sum_{|\mathbf{l}|_1=n+2} u_{h_\mathbf{l}} - 2 \sum_{|\mathbf{l}|_1=n+1} u_{h_\mathbf{l}} + \sum_{|\mathbf{l}|_1=n} u_{h_\mathbf{l}} \qquad \text{if } d = 3. \tag{2.13}$$

An illustration of the two-dimensional combination and the component grids involved is provided on figure 2.1. The following theorem shows that the sparse grid reconstruction is a fair representation of the function.

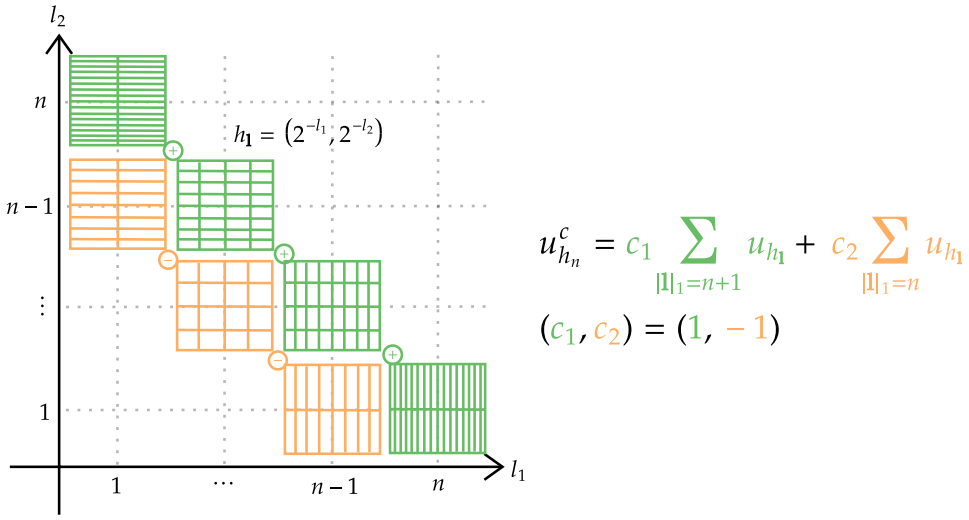


**Figure 2.1.** Illustration of the two-dimensional combination technique and the component grids used in the sparse grid approximation.

**Theorem 2.1.3 (Error of the combination technique)** *Let $u$ be a function and $u_{h_\mathbf{l}} \in V_{h_\mathbf{l}}$ be an approximation of $u$ such that the following pointwise error expression holds for $\mathbf{l} \in \mathbb{L}_n$:*

$$u_{h_\mathbf{l}}(\boldsymbol{x}) - u(\boldsymbol{x}) = \sum_{m=1}^{d} \sum_{\substack{\{i_1,\dots,i_m\} \\ \subset\{1,\dots,d\} \\ i_k \neq i_k}} a_{i_1,\dots,i_m}(\boldsymbol{x}; h_{l_{i_1}}, \dots, h_{l_{i_m}}) h^2_{l_{i_1}} \dots h^2_{l_{i_m}} \tag{2.14}$$

*where the $a_{i_1,\dots,i_m}(\cdot; h_{l_{i_1}}, \dots, h_{l_{i_m}})$ are bounded functions such that:*

$$a_{i_1,\dots,i_m}(\boldsymbol{x}; h_{l_{i_1}}, \dots, h_{l_{i_m}}) = \tilde{a}_{i_1,\dots,i_m}(\boldsymbol{x}) + O\left(h_{l_{i_1}}, \dots, h_{l_{i_m}}\right), \quad \|\tilde{a}_{i_1,\dots,i_m}\|_\infty \leq \kappa. \tag{2.15}$$

*Then the combination defined by equation (2.11) verifies the following local error:*

$$u^c_{h_n} - u = \tilde{a}_{1,\dots,d} h^2_n \sum_{r=0}^{d-1} 2^{-2(d-1-r)} (-1)^r \binom{d-1}{r} \binom{n+d-2-r}{d-1} + O\left(h^2_n \binom{n+d-3}{d-2}\right), \tag{2.16}$$

*i.e.*

$$u^c_{h_n} - u = O\left(h^2_n |\log h_n|^{d-1}\right). \tag{2.17}$$

The combination technique is remarkable for the reduction of the number of interpolation nodes, explicited by the relations:

$$|\Omega_{h_n}^{(\infty)}| = O\left(h_n^{-d}\right), \quad \sum_{\mathbf{l}\in\mathbb{L}_n} |\Omega_{h_\mathbf{l}}| = O\left(h_n^{-1}|\log h_n|^{d-1}\right), \tag{2.18}$$

while achieving nearly the same precision than the standard interpolation (with a negligible multiplicative term $|\log h_n|^{d-1}$). Indeed, for standard interpolation on the Cartesian grid with basis functions of degree one, *i.e.* hat functions, the interpolation error scales as $O\left(h_n^2\right)$.

**Remark 2.1.4** *The sparse grid reconstruction of a nonnegative function is not nonnegative.*

## 2.2 Application to PIC discretizations

> *"Crucially for PIC, the combination technique grids have very large cells relative to a comparable regular grid. This improves statistical resolution by increasing the number of particles per cell without increasing the overall particle number"*
>
> L.F. Ricketson, A.J. Cerfon, *Sparse grid techniques for particle-in-cell schemes*, 2016.

In this section, the application of the sparse grid combination technique to PIC methods is presented. In the regular PIC approximation, the main drawback is the statistical error decreasing with the number of particles per cell. Indeed, the particle sampling and the grid-based errors scale as:

$$\text{Bias}(\rho_{h_n,N})(\mathbf{x}) = O\left(h_n^2\right), \quad \mathbb{V}(\mathcal{V}(\rho_{h_n,N}(\mathbf{x})))^{\frac{1}{2}} = O\left(1/\sqrt{Nh_n^d}\right), \tag{2.19}$$

$h_n$ being the mesh size of the Cartesian grid, denoted $\Omega_{h_n}^{(\infty)}$, and $N$ the total number of particles. These two error estimates together lead to the following onerous conditions for convergence of the scheme $h_n^2 \ll 1, Nh_n^d \gg 1$ which can require an extremely large number of particles, specifically for three dimensional simulations. The combination technique achieves a representation of a function using a sequence of sparse grids, coarser than the standard full mesh. This ends up in a reduced number of interpolation nodes and, accordingly, an increased mean number of particles per cell. This feature motivates the application of sparse grid techniques to PIC methods.

### 2.2.1 Introduction to sparse PIC: charge accumulation onto the component grids

Let us now introduce in more details the motivation above mentioned for merging sparse grid techniques into PIC schemes. In order to do so, we investigate the error resulting from the charge density accumulation onto the component grids. Starting from the definition stated by equation (1.6), the density can be recast into the following integral:

$$\rho(\mathbf{x}) = Q \iint_{\Omega_x\times\Omega_v} \delta(\boldsymbol{\xi} - \mathbf{x}) f(\boldsymbol{\xi}, \mathbf{v}) d\mathbf{v} d\boldsymbol{\xi}, \tag{2.20}$$

where $Q = \int_{\Omega_x} qn(\mathbf{x})d\mathbf{x}$ is the total charge of the system. As already seen, an approximation based on the sampling of numerical particles can be constructed:

$$\rho_N(\mathbf{x}, t) = \sum_{p=1}^{N} q_p \delta(\mathbf{x} - \mathbf{x}_p(t)). \tag{2.21}$$

Let us consider a component grid with discretization $h_\mathbf{l}$ (corresponding to the cell width) and, as an ersatz of the convolution kernel (Dirac distribution), a $d$-dimensional shape function, denoted $\mathcal{S}_{h_\mathbf{l}}$ and represented in figure 2.2, which is constructed by tensor products of one dimensional hat functions:

$$\mathcal{S}_{h_\mathbf{l}}(\mathbf{x}) := \left( \bigotimes_{i=1}^{d} \mathcal{S}_{h_{l_i}} \right)(\mathbf{x}), \quad \mathcal{S}_{h_{l_i}}(x) := h_{l_i}^{-1} W\left(h_{l_i}^{-1} x\right), \quad W(x) = \max\left(1 - |x|, 0\right). \tag{2.22}$$

Let $\bar{\mathcal{S}}_{h_\mathbf{l}}(\mathbf{x}) : \Omega_x \times \Omega_x \longrightarrow \mathbb{R}$ be a function defined by:

$$\bar{\mathcal{S}}_{h_\mathbf{l}}(\mathbf{x}, \mathbf{y}) = \mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{y}). \tag{2.23}$$

Then for all $\mathbf{x} \in \Omega_x$, $\bar{\mathcal{S}}_{h_\mathbf{l}}(\mathbf{x}, \cdot)$ is a continuous function with compact support and we define an approximation of the charge density, denoted $\rho_{h_\mathbf{l},N}$, by the relation:

$$\rho_{h_\mathbf{l},N}(\mathbf{x}) := \langle \rho_N, \bar{\mathcal{S}}_{h_\mathbf{l}}(\mathbf{x}, \cdot) \rangle = \sum_{p=1}^{N} q_p \mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{x}_p(t)). \tag{2.24}$$

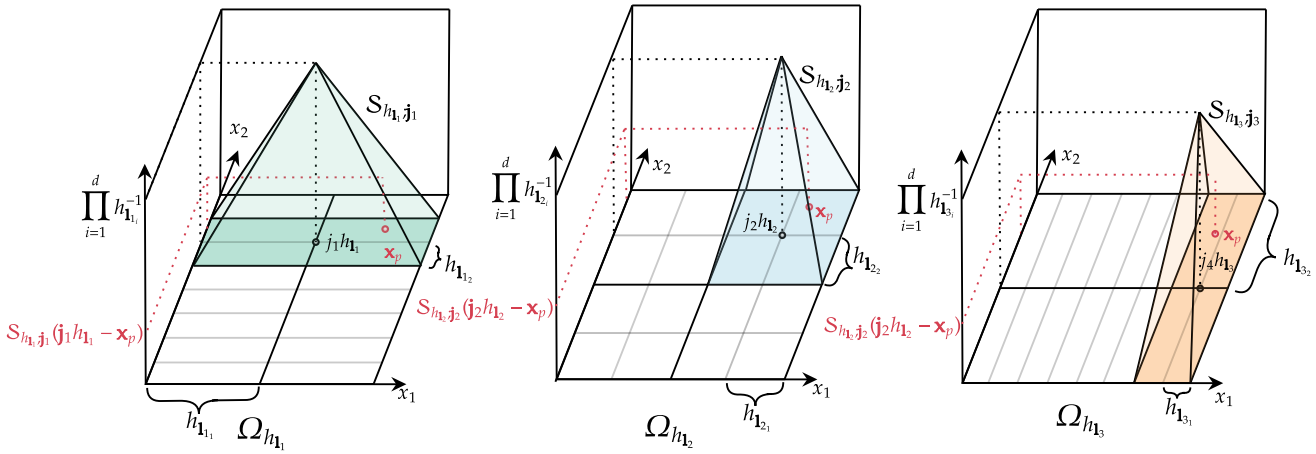


**Figure 2.2.** Example of two-dimensional hierarchical shape functions $\mathcal{S}_{h_\mathbf{l}}(\mathbf{j}h_\mathbf{l} - \mathbf{x}_p)$.

**Error estimation with Monte-Carlo method**

> *"To calculate the probability of a successful outcome of a game of solitaire is a completely intractable task. On the other hand, the laws of large numbers and the asymptotic theorems of the theory of probabilities will not throw much light even on qualitative questions concerning such probabilities. Obviously the practical procedure is to produce a large number of examples of any given game and then to examine the relative proportion of successes. We can see at once that the estimate will never be confined within given limits with certainty, but only if the number of trials is great with great probability."*
>
> N. Metropolis and S.Ulam, *The Monte Carlo method*, 1949.

Let us now estimate the error made by this approximation by considering a Monte Carlo approach. Let us recall the total charge of the system $Q = \int_{\Omega_x} qn(\mathbf{x})d\mathbf{x}$ and let $\tilde{f} = f/\int_{\Omega_x} n(\mathbf{x})d\mathbf{x}$ be the probability density function associated to the phase-space distribution of the particles, where $n$ is the physical particle density, defined in equation (1.6). It verifies an unit mean relation:

$$\iint_{\Omega_x\times\Omega_v} \tilde{f}(\mathbf{x},\mathbf{v},t)d\mathbf{x}d\mathbf{v} = 1. \tag{2.25}$$

Substituting the convolution kernel with the shape function in (2.20), we define an approximation of the density by:

$$\rho_{h_1}(\mathbf{x}) := Q \iint_{\Omega_x\times\Omega_v} \mathcal{S}_{h_1}(\mathbf{x}-\boldsymbol{\xi})\, f(\boldsymbol{\xi},\mathbf{v})d\mathbf{v}d\boldsymbol{\xi}. \tag{2.26}$$

In order to approximate this multidimensional integral, a Monte Carlo framework using statistical sampling techniques is considered. Let $\mathbf{X}$ be a random variable with the following probability density function, expected value and variance:

$$\mathbf{x} \mapsto \int_{\Omega_v} f(\mathbf{x},\mathbf{v})d\mathbf{v},\ \ \mathbb{E}(\mathbf{X}) := \iint_{\Omega_x\times\Omega_v} \boldsymbol{\xi} f(\boldsymbol{\xi},\mathbf{v})d\mathbf{v}d\boldsymbol{\xi},\ \ \mathbb{V}(\mathbf{X}) := \mathbb{E}\left((\mathbf{X}-\mathbb{E}(\mathbf{X}))^2\right). \tag{2.27}$$

The quantity $\mathcal{S}_{h_1}(\mathbf{x}-\mathbf{X})$ being a random variable and the function defined by $\boldsymbol{\xi} \mapsto \mathcal{S}_{h_1}(\mathbf{x}-\boldsymbol{\xi})$ being a mesurable function of $\Omega_x$, owing to Transfer theorem [99], the integral in (2.26) can be recast into an expected value:

$$\rho_{h_1}(\mathbf{x}) = Q\mathbb{E}\left(\mathcal{S}_{h_1}(\mathbf{x}-\mathbf{X})\right). \tag{2.28}$$

Using an independent random sample $(\mathbf{X}_1,...,\mathbf{X}_M)$ from the random variable $\mathbf{X}$, one can define an estimator of the integral by:

$$\hat{\rho}_{h_1}(\mathbf{x}) = \frac{Q}{M}\sum_{i=1}^{M}\mathcal{S}_{h_1}(\mathbf{x}-\mathbf{X}_i), \tag{2.29}$$

where $M$ is the number of random variables in the sample. To be able to make use of this result in particle simulations, we note that we can associate the values of the random sample $(\mathbf{X}_1,...,\mathbf{X}_M)$

with the particle positions [4] $(\mathbf{x}_1, ..., \mathbf{x}_M)$, yielding the following statistical estimator:

$$\hat{\rho}_{h_\mathbf{l}}(\mathbf{x}) := \frac{Q}{N}\sum_{p=1}^{N} \mathcal{S}_{h_\mathbf{l}}\left(\mathbf{x} - \mathbf{x}_p\right) = \rho_{h_\mathbf{l},N}. \tag{2.30}$$

This estimator is equal to the first approximation of the charge density, defined in equation (2.24). The expected value and variance of this estimator are defined from the independant random sample $(\mathbf{X}_1, ..., \mathbf{X}_N)$ by:

$$\mathbb{E}(\hat{\rho}_{h_\mathbf{l}}(\mathbf{x})) := \mathbb{E}\left(\frac{Q}{N}\sum_{p=1}^{N} \mathcal{S}_{h_\mathbf{l}}\left(\mathbf{x} - \mathbf{X}_p\right)\right), \quad \mathbb{V}(\hat{\rho}_{h_\mathbf{l}}(\mathbf{x})) := \mathbb{V}\left(\frac{Q}{N}\sum_{p=1}^{N} \mathcal{S}_{h_\mathbf{l}}\left(\mathbf{x} - \mathbf{X}_p\right)\right). \tag{2.31}$$

**Definition 2.2.1 (Grid-based and particle sampling errors)** *Let $p \in \mathbb{N}$ and let $\hat{u}_h$ be a statistical estimator of a function $u : \Omega_x \to \mathbb{R}^p$, defined on a grid $\Omega_h$, i.e. a quantity that depends on the sampling of the particles (the number of particles $N$) and the grid discretization $h$. The local error between the function and its statistical estimator is recast into two components:*

$$\hat{u}_h - u = \underbrace{(\hat{u}_h - \mathbb{E}(\hat{u}_h))}_{\mathcal{V}(\hat{u}_h)} + \underbrace{(\mathbb{E}(\hat{u}_h) - u)}_{\text{Bias}(\hat{u}_h)}. \tag{2.32}$$

*The first, refered to as the particle sampling error and denoted $\mathcal{V}(\hat{u}_h)$, is a centered random variable corresponding to the error stemming from the variance of the sampling with a finite number of particles. The second is the bias of the estimator, denoted $\text{Bias}(\hat{u}_h)$ also referred to as the grid-based error. It depends on both the mesh size and the solution smoothness and measures how close to the function $u$ the most probable value of the estimator is.*

The local error between the density and its statistical estimator is recast into a grid-based error and a particle sampling error:

$$\hat{\rho}_{h_\mathbf{l}} - \rho = \underbrace{(\hat{\rho}_{h_\mathbf{l}} - \mathbb{E}(\hat{\rho}_{h_\mathbf{l}}))}_{\mathcal{V}(\hat{\rho}_{h_\mathbf{l}})} + \underbrace{(\mathbb{E}(\hat{\rho}_{h_\mathbf{l}}) - \rho)}_{\text{Bias}(\hat{\rho}_{h_\mathbf{l}})}. \tag{2.33}$$

**Proposition 2.2.2** *Let $f(\cdot, \boldsymbol{v}) \in X^2(\Omega)$, then the particle sampling error is an unbiased random variable:*

$$\mathbb{E}\left(\mathcal{V}(\hat{\rho}_{h_l})\right) = 0. \tag{2.34}$$

*In addition, the variance of the particle sampling error and the grid-based error are given by:*

$$\mathbb{V}(\mathcal{V}(\hat{\rho}_{h_l})) = \frac{v_0(\cdot; h_{l_1}, ..., h_{l_d})}{N h_{l_1} ... h_{l_d}} + \sum_{m=1}^{d} \sum_{\substack{\{i_1,...,i_m\} \\ \subset\{1,...,d\} \\ i_j \neq i_k}} v_{i_1,...,i_m}(\cdot; h_{l_{i_1}}, ..., h_{l_{i_m}}) \frac{h_{l_{i_1}}^2 ... h_{l_{i_m}}^2}{N h_{l_1} ... h_{l_d}}, \tag{2.35}$$

$$\text{Bias}(\hat{\rho}_{h_\mathbf{l}}) = \sum_{m=1}^{d} \sum_{\substack{\{i_1,\dots,i_m\} \\ \subset\{1,\dots,d\} \\ i_j \neq i_k}} b_{i_1,\dots,i_m}(\cdot; h_{l_{i_1}}, \dots, h_{l_{i_m}}) h_{l_{i_1}}^2 \dots h_{l_{i_m}}^2, \tag{2.36}$$

*with:*

$$v_0(\cdot; h_{l_1}, \dots, h_{l_d}) = \left(\frac{2}{3}\right)^d Q\rho + O\left(h_{l_1} \dots h_{l_d}\right), \tag{2.37}$$

$$v_{i_1,\dots,i_m}(\cdot; h_{l_{i_1}}, \dots, h_{l_{i_m}}) = \left(\frac{2}{3}\right)^{d-m} \left(\frac{1}{15}\right)^m Q\partial_{i_1}^2 \dots \partial_{i_m}^2 \rho + O\left(h_{l_{i_1}}^2, \dots, h_{l_{i_m}}^2\right), \tag{2.38}$$

$$b_{i_1,\dots,i_m}(\cdot; h_{l_{i_1}}, \dots, h_{l_{i_m}}) = \left(\frac{1}{12}\right)^m \partial_{i_1}^2 \dots \partial_{i_m}^2 \rho + O\left(h_{l_{i_1}}^2, \dots, h_{l_{i_m}}^2\right). \tag{2.39}$$

The magnitude of the particle sampling error is given by square root of the variance which scale with:

$$\mathbb{V}(\mathcal{V}(\hat{\rho}_{h_\mathbf{l}}))^{\frac{1}{2}} = O\left(\left(N h_{l_1} \dots h_{l_d}\right)^{-\frac{1}{2}}\right). \tag{2.40}$$

Note that the bias of the charge density estimator on a component grid verifies the assumption of equation (2.14). Therefore, the combination of these projections onto the component grids according to equation (2.11) shall lead to cancellations for the grid-based error. This feature, resulting from the tensor product form of the shape function, is the motivation for the application of sparse grid combination to PIC methods.

### 2.2.2 Discretization on hybrid grids: PIC-Hg scheme, properties and error estimations

In this section, a first sparse grid application to PIC methods is presented. In order to take advantage of the cancellations exposed in the precedent section, a sparse grid interpolant of the density is constructed.

The density is accumulated onto each component grid, achieving a reduction of the statistical error thanks to the large cells of the component grids, then evaluated onto the Cartesian grid with the combination technique. The electric field is obtained by resolving the Poisson equation on the Cartesian grid. Let us introduce the sparse grid reconstruction of the density, denoted $\hat{\rho}^c_{h_n}$, and the resulting electric field, denoted $\hat{\mathbf{E}}_{h_n}$, as:

$$\hat{\rho}^c_{h_n} = \sum_{\mathbf{l} \in \mathbb{L}_n} c_\mathbf{l} \mathcal{I}_{V_{h_\mathbf{l}}} \hat{\rho}_{h_\mathbf{l}}, \qquad \hat{\mathbf{E}}_{h_n} = -\boldsymbol{\nabla}_{h_n} \hat{\Phi}_{h_n}, \qquad -\varepsilon_0 \Delta_{h_n} \hat{\Phi}_{h_n} = \hat{\rho}^c_{h_n}, \tag{2.41}$$

where $\boldsymbol{\nabla}_{h_n}$ and $\Delta_{h_n}$ are the discrete gradient and discrete laplacian finite differences operators, defined in equations (17). The corresponding scheme, named Hybrid grid PIC (PIC-Hg) scheme (owing to the Cartesian grid and component grids considered within the scheme) is detailed in algorithm 2. An illustration of the scheme is provided on figure 2.3. The following propositions hold true for the scheme.

**Proposition 2.2.3** *The sparse grid reconstruction of the density preserves the total charge:*

$$\int_\Omega \hat{\rho}^c_{h_n}(\boldsymbol{x}) d\boldsymbol{x} = Q. \tag{2.42}$$

**Algorithm 2** PIC-Hg scheme

**Require:** Particle positions and velocities $(\mathbf{x}_p, \mathbf{v}_p)$, time step $\Delta t$, external fixed magnetic field $\mathbf{B}$, component grids $\Omega_{h_\mathbf{l}}$ and Cartesian grid $\Omega_{h_n}^{(\infty)}$.
**for each** time step $k\Delta t$ **do**
  **for each** component grid of level $\mathbf{l} \in \mathbb{L}_n$ **do**
    **Accumulate** the charge density onto the component grid (2.30).
    **Interpolate** the charge density onto $V_{h_\mathbf{l}}$ (2.10).
  **end for**
  **Combinate** the charge density onto the Cartesian grid (2.41).
  **Compute** the electric field from the combined charge density on the Cartesian grid with finite differences (2.41).
  **Evaluate** $\mathcal{I}_{V_{h_n}^{(\infty)}} \hat{\mathbf{E}}_{h_n}$ at the particle positions.
  **Update** the particle velocities and positions according to the leap frog scheme (1.30).
**end for**

**Remark 2.2.4** *The positivity of the charge density is not preserved by the sparse grid interpolation (see remark 2.1.4).*

**Proposition 2.2.5** *The local error between the charge density sparse grid reconstruction and the solution is recast into:*

$$\hat{\rho}^c_{h_n} - \rho = \underbrace{\mathrm{Bias}(\hat{\rho}^c_{h_n})}_{\text{grid-based error}} + \underbrace{\mathcal{V}(\hat{\rho}^c_{h_n})}_{\text{particle sampling error}} . \tag{2.43}$$

*Assuming $\rho \in X^2(\Omega)$, then the grid-based error and the particle sampling error verify:*

$$\left\|\mathrm{Bias}(\hat{\rho}^c_{h_n})\right\|_\infty \leq K\|\partial_1^2 \ldots \partial_d^2 \rho\|_\infty h_n^2 |\log h_n|^{d-1} + O\left(h_n^2 |\log h_n|^{d-2}\right), \tag{2.44}$$

$$\left\|\mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right)^{\frac{1}{2}}\right\|_\infty \leq D\|\rho\|_\infty^{\frac{1}{2}} (Nh_n)^{-\frac{1}{2}} |\log h_n|^{d-1} + O\left(\left(\frac{|\log h_n|^{d-1}}{Nh_n}\right)^{\frac{1}{2}}\right) \tag{2.45}$$

*where $K$ and $D$ are constants depending on the dimension of the problem. For two-dimensional computations, it holds:*

$$\left\|\mathrm{Bias}(\hat{\rho}^c_{h_n})\right\|_\infty \leq K_1 h_n^2 |\log h_n| + K_2 h_n^2, \tag{2.46}$$

$$\left\|\mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right)^{\frac{1}{2}}\right\|_\infty \leq D|\log h_n|(Nh_n)^{-\frac{1}{2}}, \tag{2.47}$$

$$K_1 = \frac{3125}{46656}\|\partial_1^2\partial_2^2\rho\|_\infty, \quad K_2 = \frac{25}{108}(\|\partial_1^2\rho\|_\infty + \|\partial_2^2\rho\|_\infty), \quad D = \frac{2\sqrt{8}}{3\log(2)}\left(Q\|\rho\|_\infty\right)^{\frac{1}{2}}$$

The benefit of the sparse grid combination technique as a noise reduction strategy is pointed out by this proposition. Indeed, owing to the projection of the density onto the component grids,

the particle sampling error is significantly reduced:

$$\left\| \mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right)^{\frac{1}{2}} \right\|_\infty \leq O\left((Nh_n)^{-\frac{1}{2}} |\log h_n|^{d-1}\right) \leq O\left((Nh_n^d)^{-\frac{1}{2}}\right) = \left\| \mathbb{V}(\mathcal{V}(\hat{\rho}_{h_n}))^{\frac{1}{2}} \right\|_\infty ,$$

where $\mathcal{V}(\hat{\rho}_{h_n})$ (respectively $\mathcal{V}(\hat{\rho}^c_{h_n})$) is the particle sampling error of the density projection onto the Cartesian grid (respectively the sparse grid reconstruction). The profit is significant for refined grids as well as three dimensional problems. Conversely, an increase of the grid-based error results from the combination; the grid-based error is increased from $O\left(h_n^2\right)$ to $O\left(h_n^2 |\log h_n|^{d-1}\right)$. Though $h_n^2 \approx h_n^2 |\log h_n|^{d-1}$, the loss may appear negligible, however the dominant component of the density error depends on derivatives of order $2d$. As a comparison, the grid-based error of regular PIC methods is dominated by component with dependencies on derivatives of order 2. This feature indicates a drawback of the combination technique and may limit the efficiency of the method when the solution develops strong gradients not aligned with the Cartesian grid. A similar estimation can be stated for the electric field.

**Theorem 2.2.6** *The local error between the electric field approximation and the solution is recast into:*

$$\mathcal{I}_{V^{(\infty)}_{h_n}} \hat{\boldsymbol{E}}_{h_n} - \boldsymbol{E} = \underbrace{\text{Bias}\left(\mathcal{I}_{V^{(\infty)}_{h_n}} \hat{\mathbf{E}}_{h_n}\right)}_{\textit{grid-based error}} + \underbrace{\mathcal{V}\left(\mathcal{I}_{V^{(\infty)}_{h_n}} \hat{\boldsymbol{E}}_{h_n}\right)}_{\textit{particle sampling error}} , \tag{2.48}$$

*Assuming enough smoothness on the solutions, i.e.* $\Phi \in C^5_0(\Omega)$, $\boldsymbol{E} \in C^4(\Omega)$ $\rho \in X^2(\Omega) \cap C^3(\Omega)$, *then the grid-based error and the particle sampling error verify:*

$$\left\| \text{Bias}\left(\mathcal{I}_{V^{(\infty)}_{h_n}} \hat{\mathbf{E}}_{h_n}\right) \right\|_\infty \leq K \|\partial_1^2 ... \partial_d^2 \boldsymbol{\nabla}\rho\|_\infty h_n^2 |\log h_n|^{d-1} + O\left(h_n^2 |\log h_n|^{d-2}\right), \tag{2.49}$$

$$\left\| \mathbb{V}\left(\mathcal{V}\left(\mathcal{I}_{V^{(\infty)}_{h_n}} \hat{\boldsymbol{E}}_{h_n}\right)\right)^{\frac{1}{2}} \right\|_\infty \leq D \|\boldsymbol{\nabla}\rho\|_\infty^{\frac{1}{2}} (Nh_n)^{-\frac{1}{2}} |\log h_n|^{d-1} + O\left(\left(\frac{|\log h_n|^{d-1}}{Nh_n}\right)^{\frac{1}{2}}\right), \tag{2.50}$$

*where K and D are constants depending on the dimension of the problem. For two-dimensional computations, the following estimations hold true:*

$$\left\| \text{Bias}\left(\mathcal{I}_{V^{(\infty)}_{h_n}} \hat{\mathbf{E}}_{h_n}\right) \right\|_\infty \leq K_1 h_n^2 |\log h_n| + K_2 h_n^2, \quad \left\| \mathbb{V}\left(\mathcal{V}\left(\mathcal{I}_{V^{(\infty)}_{h_n}} \hat{\boldsymbol{E}}_{h_n}\right)\right)^{\frac{1}{2}} \right\|_\infty \leq D (Nh_n)^{-\frac{1}{2}} |\log h_n|, \tag{2.51}$$

*with*

$$\begin{aligned}
K_1 &= \frac{3125}{373248} \|\partial_1^2 \partial_2^2 \boldsymbol{\nabla}\rho\|_\infty, \quad D = \frac{2\sqrt{8}}{3\log(2)} \left(Q \|\boldsymbol{\nabla}\rho\|_\infty\right)^{\frac{1}{2}}, \\
K_2 &= \frac{1}{96} \left( \|\partial_1^4 \boldsymbol{E}\|_\infty + \|\partial_2^4 \boldsymbol{E}\|_\infty \right) + \frac{1}{3} \max(\|\partial_1^3 \Phi\|_\infty, \|\partial_2^3 \Phi\|_\infty) \\
&\quad + \frac{4}{27} (\|\partial_1^2 \boldsymbol{E}\|_\infty + \|\partial_2^2 \boldsymbol{E}\|_\infty) + \frac{25}{864} (\|\partial_1^2 \boldsymbol{\nabla}\rho\|_\infty + \|\partial_2^2 \boldsymbol{\nabla}\rho\|_\infty).
\end{aligned}$$

### 2.2.3 Discretization on component grids: PIC-Sg, PIC-NSg schemes, properties and error estimations

In this section, a second application of the sparse grid combination technique to PIC methods [97] is presented. This implementation does not use a reconstruction of the density onto the Cartesian grid. The idea is to deposit the charge density, solve the Poisson equation, differentiate the electric potential on each component grid and eventually interpolate the electric field from the component grids at particle positions by means of the combination technique. Let us introduce the sparse grid reconstruction of the electric field, denoted $\mathbf{E}^c_{h_n}$, defined by the relation:

$$\hat{\mathbf{E}}^c_{h_n} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} \mathcal{I}_{V_{h_\mathbf{l}}} \hat{\mathbf{E}}_{h_\mathbf{l}}, \tag{2.52}$$

where $\hat{\mathbf{E}}_{h_\mathbf{l}}$ is the approximation of the electrid field on a component grid with finite differences. Solving the Poisson problem on each component grid rather than on the Cartesian grid, is likely to speed-up the computations. Indeed, the gain may be coarsely estimated by the reduced number of cells in all the component grids compared to the Cartesian mesh (see equation 2.18), the result being a linear system with reduced size to solve for this scheme. Let us introduce the scheme in algorithm 3, named Sub-grid PIC (PIC-Sg) scheme in the following. An illustration of the scheme is provided on figure 2.3. The following propositions hold true for the scheme.

**Algorithm 3** PIC-Sg scheme

**Require:** Particle positions and velocities $(\mathbf{x}_p, \mathbf{v}_p)$, time step $\Delta t$, external fixed magnetic field $\mathbf{B}$, component grids $\Omega_{h_\mathbf{l}}$ and Cartesian grid $\Omega^{(\infty)}_{h_n}$.
**for each** time step $k\Delta t$ **do**
  **for each** component grid of level index $\mathbf{l} \in \mathbb{L}_n$ **do**
    **Accumulate** the charge density onto the component grid (2.30).
    **Compute** the electric field from the charge density on the component grid with finite differences.
    **Evaluate** $\mathcal{I}_{V_{h_\mathbf{l}}} \hat{\mathbf{E}}_{h_\mathbf{l}}$ at the particle positions (2.10).
  **end for**
  **Combinate** the electric field defined at the particle positions (2.52).
  **Update** the particle velocities and positions according to the leap frog scheme (1.30).
**end for**

**Theorem 2.2.7** *The local error between the electric field sparse grid reconstruction and the solution is recast into:*

$$\hat{\boldsymbol{E}}^c_{h_n} - \boldsymbol{E} = \underbrace{\mathrm{Bias}(\hat{\mathbf{E}}^c_{h_n})}_{\text{grid-based error}} + \underbrace{\mathcal{V}(\hat{\boldsymbol{E}}^c_{h_n})}_{\text{particle sampling error}}, \tag{2.53}$$

*Assuming enough smoothness on the solutions, i.e.* $\Phi \in X^4_0(\Omega) \cap C^5_0(\Omega)$, $\boldsymbol{E} \in X^4(\Omega)$, $\rho \in$

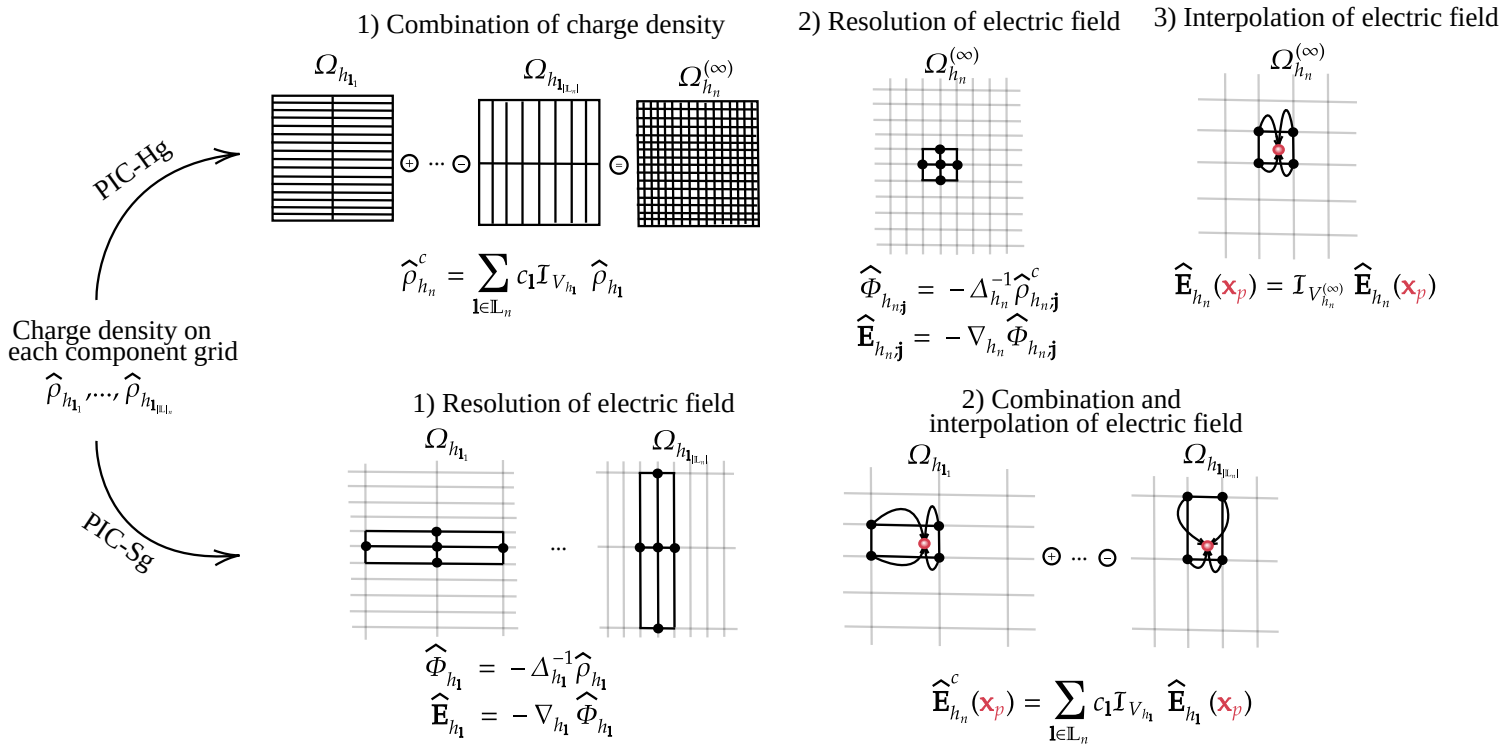


**Figure 2.3.** Schematic depiction of the differences between the PIC-Hg and the PIC-Sg schemes.

$X^4(\Omega) \cap C^5(\Omega)$, *then the grid-based error and the particle sampling error verify:*

$$\left\|\mathrm{Bias}(\hat{\mathbf{E}}^c_{h_n})\right\|_\infty \leq \left(K_{1,1}\|\partial_1^4 \ldots \partial_d^4 \boldsymbol{E}\|_\infty + K_{1,2}\|\partial_1^4 \ldots \partial_{d-1}^4 \partial_d^2 \boldsymbol{\nabla}\rho\|_\infty + \ldots\right) h_n^2 |\log h_n|^{d-1} + O\left(h_n^2 |\log h_n|^{d-2}\right), \tag{2.54}$$

$$\left\|\mathbb{V}\left(\mathcal{V}(\hat{\boldsymbol{E}}^c_{h_n})\right)^{\frac{1}{2}}\right\|_\infty \leq D\|\boldsymbol{\nabla}\rho\|_\infty^{\frac{1}{2}} (N h_n)^{-\frac{1}{2}} |\log h_n|^{d-1} + O\left(\left(\frac{|\log h_n|^{d-1}}{N h_n}\right)^{\frac{1}{2}}\right), \tag{2.55}$$

*and where* $K_{1,1}$, $K_{1,2}$,..., $D$ *are constants depending on the dimension of the problem. For two-dimensional computations, the following estimations hold true:*

$$\left\|\mathrm{Bias}(\hat{\mathbf{E}}^c_{h_n})\right\|_\infty \leq K_1 h_n^2 |\log h_n| + K_2 h_n^2, \quad \left\|\mathbb{V}\left(\mathcal{V}(\hat{\boldsymbol{E}}^c_{h_n})\right)^{\frac{1}{2}}\right\|_\infty \leq D(N h_n)^{-\frac{1}{2}} |\log h_n|, \tag{2.56}$$

*with*

$$
\begin{aligned}
K_1 &= \frac{20}{729}\|\partial_1^2\partial_2^2\boldsymbol{E}\|_\infty + \frac{5}{36864}\left(\|\partial_1^4\partial_2^2\boldsymbol{\nabla}\rho\|_\infty + 2\|\partial_1^4\partial_2^4\boldsymbol{E}\|_\infty + \|\partial_1^2\partial_2^4\boldsymbol{\nabla}\rho\|_\infty\right) \\
&+ \frac{5}{1152}\max\left(\|\partial_1^3\partial_2^2\rho\|_\infty + \|\partial_1^3\partial_2^4\Phi\|_\infty, \|\partial_1^2\partial_2^3\rho\|_\infty + \|\partial_1^4\partial_2^3\Phi\|_\infty\right) \\
&+ \frac{5}{2592}\left(\|\partial_1^4\partial_2^2\boldsymbol{E}\|_\infty + 2\|\partial_1^2\partial_2^2\boldsymbol{\nabla}\rho\|_\infty + \|\partial_1^2\partial_2^4\boldsymbol{E}\|_\infty\right) \\
&+ \frac{5}{81}\max(\|\partial_1^2\partial_2^3\Phi\|_\infty, \|\partial_1^3\partial_2^2\Phi\|_\infty), \\
K_2 &= \frac{4}{27}\left(\|\partial_1^2\boldsymbol{E}\|_\infty + \|\partial_2^2\boldsymbol{E}\|_\infty\right) + \frac{1}{96}\left(\|\partial_1^2\boldsymbol{\nabla}\rho\|_\infty + \|\boldsymbol{\nabla}\partial_2^2\rho\|_\infty\right) \\
&+ \frac{1}{96}\left(\|\partial_1^4\boldsymbol{E}\|_\infty + \|\partial_2^4\boldsymbol{E}\|_\infty\right) + \frac{2}{3}\max(\|\partial_1^3\Phi\|_\infty, \|\partial_2^3\Phi\|_\infty), \\
D &= \frac{2}{3\sqrt{8}}\left(Q\|\boldsymbol{\nabla}\rho\|_\infty\right)^{\frac{1}{2}}.
\end{aligned}
$$

**Proposition 2.2.8** *Assuming a periodic domain, the scheme does preserve the total momentum of the system, i.e*

$$
\frac{d}{dt}\left(m\iint_{\Omega\times\Omega_v}\boldsymbol{v}f_N(\boldsymbol{x},\boldsymbol{v},t)d\boldsymbol{x}d\boldsymbol{v}\right) = 0, \tag{2.57}
$$

*where:*

$$
f_N(\boldsymbol{x},\boldsymbol{v},t) = \sum_{p=1}^{N}\frac{n}{N}\delta(\boldsymbol{x}-\boldsymbol{x}_p(t))\delta(\boldsymbol{v}-\boldsymbol{v}_p(t)). \tag{2.58}
$$

The estimation of the error requires stronger assumptions on the smoothness of the electric potential $\Phi \in X_0^4(\Omega)\cap C_0^5(\Omega)$, electric field $\mathbf{E}\in X^4(\Omega)$ and density $\rho\in X^4(\Omega)\cap C^5(\Omega)$ than for the PIC-Hg scheme $\Phi\in C_0^5(\Omega)$, $\mathbf{E}\in C^4(\Omega)$, $\rho\in X^2(\Omega)\cap C^3(\Omega)$, especially on mixed derivatives. The significant differences with the PIC-Hg scheme are the additional dominant terms depending on higher mixed derivatives of the solution: $\|\partial_1^4\ldots\partial_{d-1}^4\partial_d^2\boldsymbol{\nabla}\rho\|_\infty$, ..., $\|\partial_1^4\ldots\partial_d^4\mathbf{E}\|_\infty$ which are density derivatives of order $4d-1$ and electric field derivatives of order $4d$. In comparison, the dominant terms in the PIC-Hg scheme estimation in equation (2.49) are of order $2d+1$ for the density, and the negligible terms are of order 4 for the electric field.

**Remark 2.2.9** *An alternative scheme, named PIC-NSg, in which the component grid contributions of the electric potential are combined on the full grid is also considered. The electric field is then computed on the full grid with differentiation and interpolated at the particle positions. The scheme is summarized in algorithm 4.*

**Proposition 2.2.10** *For all the sparse grid schemes (PIC-Hg,PIC-Sg,PIC-NSg), the linear dispersion relation for a cold plasma with no drift is similar to the standard PIC scheme (PIC-Std). The linear stability constraint associated to the schemes is then:*

$$
\Delta t < \frac{2}{\omega_p}. \tag{2.59}
$$

**Algorithm 4** PIC-NSg scheme

**Require:** Particle positions and velocities $(\mathbf{x}_p, \mathbf{v}_p)$, time step $\Delta t$, external magnetic field $\mathbf{B}$, component grids $\Omega_{h_\mathbf{l}}$ and Cartesian grid $\Omega_{h_n}^{(\infty)}$.
**for each** time step $k\Delta t$ **do**
  **for each** component grid of index $\mathbf{l} \in \mathbb{L}_n$ **do**
    **Accumulate** the charge density onto the component grid (2.30)
    **Compute** the electric potential from the charge density on the component grid.
    **Interpolate** the electric potential onto $V_{h_\mathbf{l}}$ (2.10).
  **end for**
  **Combine** the electric potential onto the full grid (2.11) in nodal basis.
  **Differentiate** the electric potential on the full grid.
  **Interpolate** the electric field at the particle positions.
  **Update** the particle positions and velocities (1.30).
**end for**

### 2.2.4 Improvements of the schemes

#### Offset combination technique

From the analysis conducted in the precedent sections, the following conclusions may be stated. Sparse grid reconstructions define a different trade off between the two components of the errors (grid-based and particle sampling errors) as compared to standard PIC approximations. The particle sampling error is significantly mitigated thanks to sparse grid approximations. This is an important feature since this component of the error is generally the most detrimental for the computations and ultimately limits the precision of the approximation. Contrariwise, the grid-based error is less favorable for the sparse grid approximations: the dominant term depends on mixed derivatives of the solutions (see tables 2.1, 2.1 for a comparison of the methods). In

**Table 2.1.** Grid-based error: dominant and marginal terms with dependances on $\rho$ or $\mathbf{E}$ derivatives.

i) Density grid-based error ($\|\hat{\rho}^c_{h_n} - \rho\|_\infty$), dependance on $\rho$ derivatives.

| Scheme | Dominant term | Negligible term |
|---|---|---|
| PIC-Std | $h_n^2 \left( \sum_{i=1}^d \|\partial_i^2 \rho\|_\infty \right)$ | $h_n^{2d} \|\partial_1^2 \ldots \partial_d^2 \rho\|_\infty$ |
| PIC-Hg / PIC-Sg | $h_n^2 \lvert \log h_n \rvert^{d-1} \|\partial_1^2 \ldots \partial_d^2 \rho\|_\infty$ | $h_n^2 \left( \sum_{i=1}^d \|\partial_i^2 \rho\|_\infty \right)$ |

ii) Electric field grid-based error ($\|\hat{\mathbf{E}}^c_{h_n} - \mathbf{E}\|_\infty$), dependance on $\mathbf{E}$ derivatives.

| Scheme | Dominant term | Negligible term |
|---|---|---|
| PIC-Std / PIC-Hg | $h_n^2 \left( \sum_{i=1}^d \|\partial_i^4 \mathbf{E}\|_\infty \right)$ | $h_n^{2d} \|\partial_1^4 \ldots \partial_d^4 \mathbf{E}\|_\infty$ |
| PIC-Sg | $h_n^2 \lvert \log h_n \rvert^{d-1} \|\partial_1^4 \ldots \partial_d^4 \mathbf{E}\|_\infty$ | $h_n^2 \left( \sum_{i=1}^d \|\partial_i^4 \mathbf{E}\|_\infty \right)$ |

this section, a general framework, that we shall referred to as offset combination technique, is introduced in order to reduce the grid-based error of sparse grid reconstructions. The offset combination technique is motivated by the property of the dominant term error to be an increasing

function of the number of sub-grids involved in the combination ($O\left(|\log h_n|^{d-1}\right)$) as well as the combined sum of the errors of the partial estimators. The offset combination consists therefore in both reducing the number of sub-grids considered within the combination and using sub-grids with increased minimum levels. In this respect, the offset combination borrows some ideas to the so-called truncated combination [14, 88]. However, a more subtle strategy is implemented within the offset combination in order to select efficiently the subset of sub-grids.

The main drawback of the truncated method is the additional statistical error introduced in the simulation, because of the smaller cells of the grids considered in the combination, the mean number of particles per cell is reduced. The offset combination is aimed at mitigating the increase of the statistical noise as well as the dominant component of the grid-based error in return of a deterioration of the error component depending on the non-mixed derivatives of the solution. The tuning of the balance between the different components of the error is implemented thanks to the two parameters $\tau_0, \tau_1 \in \mathbb{N}$. The index $\tau_0$, which is the truncation parameter, is used to parametrize the minimum discretization level for the sub-grids, with the aim of discarding the most anisotropic sub-grids from the combination. The parameter $\tau_1$, which is the offset parameter, sets the loss of discretization in the directions aligned with an axis (contributing to the negligible terms of the grid-based error with non-mixed derivatives) as illustrated on figure 2.4. For $\tau_1 = (d-1)\tau_0$ the offset method boils down to the truncated combination technique introduced in [14, 88]. The set of sub-grids for the offset combination is defined by:

**Definition 2.2.11 (Offset combination index)** *Let $\tau_0, \tau_1 \in \mathbb{N}$, the offset combination indexes are defined by:*

$$\mathbb{L}_n(\tau_0, \tau_1) := \bigcup_{i \in [\![0, d-1]\!]} \mathbb{L}_{n,i}(\tau_0, \tau_1), \quad \tau_0 \in [\![1, n]\!],\ \tau_1 \in [\![d-1, (d-1)\tau_0]\!] \tag{2.60}$$

$$\mathbb{L}_{n,i}(\tau_0, \tau_1) := \left\{ \boldsymbol{l} \in \mathbb{N}^d \mid \boldsymbol{l} \geq \tau_0,\ |\boldsymbol{l}|_1 = n + \tau_1 - i \right\}. \tag{2.61}$$

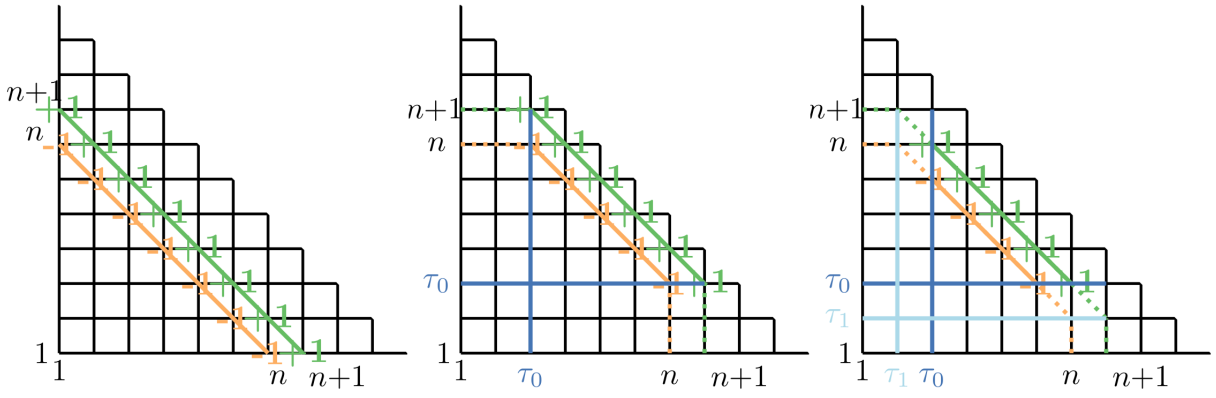


**Figure 2.4.** Schematic depiction of the two-dimensional classical combination (left), truncated combination with $\tau_0 = 3$ (middle) and offset combination with $\tau_0 = 3$, $\tau_1 = 2$ (right).

**Proposition 2.2.12** *The local error between the charge density sparse grid reconstruction with the offset combination technique and the solution is recast into:*

$$\hat{\rho}^c_{h_n} - \rho = \underbrace{\mathrm{Bias}(\hat{\rho}^c_{h_n})}_{\text{grid-based error}} + \underbrace{\mathcal{V}(\hat{\rho}^c_{h_n})}_{\text{particle sampling error}}, \tag{2.62}$$

*Assuming $\rho \in X^2(\Omega)$, then the grid-based error and the particle sampling error verify:*

$$\left\|\text{Bias}(\hat{\rho}^c_{h_n})\right\|_\infty \leq K\|\partial_1^2 \ldots \partial_d^2 \rho\|_\infty h_{n_2}^2 |\log h_{n_1}|^{d-1} + O\left(h_{n_2}^2 |\log h_{n_1}|^{d-2}\right), \tag{2.63}$$

$$\left\|\mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right)^{\frac{1}{2}}\right\|_\infty \leq D\|\rho\|_\infty^{\frac{1}{2}} (N h_{n_2})^{-\frac{1}{2}} |\log h_{n_1}|^{d-1} \tag{2.64}$$

*where the constants are the same constants as in proposition 2.2.5 and:*

$$n_1 = n - 2\tau_0 + \frac{\tau_1}{d-1}, \quad n_2 = n + \tau_1 - (d-1). \tag{2.65}$$

The proof of the above proposition is a simple extension of the proof of proposition 2.2.5. Similar results on the electric field can be demonstrated by following the proofs of theorems 2.2.7 and 2.2.6.

Compared to the estimation of the PIC-Hg scheme both the grid-based component and the particle sampling component of the error are mitigated in the offset method. This is obtained thanks to the reduced number of sub-grids in the combination which scales with $O\left(|\log h_{n_1}|^{d-1}\right)$ instead of $O\left(|\log h_n|^{d-1}\right)$, $n_1 \leq n$. Besides, the use of sub-grids with higher levels, *i.e.* parametrized by indices $\mathbf{l}$ with larger $|\mathbf{l}|_1$, entails an offset for both the particle sampling error and the grid-based error, if $n_2 \geq n$. Conversely, the negligible components in the grid-based error are increased by the elimination of the more anisotropic grids, however this is of no consequence on the accuracy of the numerical method since these components are negligible.

**Remark 2.2.13** *The offset combination technique can be applied to either the PIC-Hg scheme, the PIC-Sg scheme or the PIC-NSg scheme without the loss of their conservativity properties (total charge, total momentum, etc.).*

### Oversampled hybrid grid (PIC-OHg)

The analyses conducted within section 2.2 highlight a deterioration of the electric field approximations carried out by sparse grid PIC methods (see table 2.1) compared to standard PIC methods. Similarly to the density interpolant, this altered precision, more important for the PIC-SG scheme, is due to an increase of the grid-based error component depending on high order cross derivatives of the solution. The corrections proposed herein address this specific issue.

The PIC-Hg scheme does not take advantage of sparse-grid techniques for the resolution of the electric field, the potential being carrying out on a Cartesian mesh, unlike the PIC-Sg scheme, which makes the resolution more expansive than the latter. This computation may be expansive for refined discretizations and particularly for three dimensional problems. The benefit of this approach is a reduced dependency of the electric field approximation to the solution cross derivatives (compared to the PIC-Sg scheme, see table 2.1). The strategy introduced to alleviate the numerical cost of the electric field computation, consists in using grids with different resolutions for the charge density deposition and the electric field computation. Precisely, the electric field is carried out on a full mesh $\Omega^{(\infty)}_{h_n}$ while the density is projected onto a sequence of sub-grids associated to a more refined (oversampled) full mesh $\Omega^{(\infty)}_{h_{\tilde{n}}}$, where $h_{\tilde{n}} = h_n \cdot h_{\delta n}$, $\delta n \in \mathbb{N}$. The corrected scheme, that we shall name the oversampled hybrid grid PIC scheme, is similar to the PIC-Hg scheme, except that the sub-grids are considered in the following index set:

$$\tilde{\mathbb{L}}_n := \bigcup_{i \in [\![0, d-1]\!]} \tilde{\mathbb{L}}_{n,i}, \quad \tilde{\mathbb{L}}_{n,i} := \{\tilde{\mathbf{l}} \in \mathbb{N}^d \mid |\tilde{\mathbf{l}}|_1 = \tilde{n} + d - 1 - i,\ \mathbf{l} \geq \mathbf{1}\}, \tag{2.66}$$

Before the Poisson equation is solved, the density is deposited onto $\Omega_{h_n}^{(\infty)}$ from the values of the sparse grid interpolant on $\Omega_{h_{\tilde{n}}}^{(\infty)}$, which is denoted $\rho_{h_{\tilde{n}}}^c$:

$$\hat{\rho}_{h_n}^c(\mathbf{j}h_n) := \frac{\omega_{h_{\tilde{n}}}}{\omega_n} \sum_{\mathbf{i} \in I_{h_{\tilde{n}}}} \hat{\rho}_{h_{\tilde{n}}}^c(\mathbf{i}h_{\tilde{n}}) W_{h_n,\mathbf{j}}(\mathbf{i}h_{\tilde{n}}) \quad \text{for } \mathbf{j} \in I_{h_n}, \tag{2.67}$$

where $\omega_{h_n}$, $\omega_{h_{\tilde{n}}}$ correspond to the volume of a cell of the grid considered:

$$\omega_{h_n} = \left( \sum_{\mathbf{j} \in I_{h_n}} 1 \right)^{-1}, \quad \omega_{h_{\tilde{n}}} = \left( \sum_{\mathbf{j} \in I_{h_{\tilde{n}}}} 1 \right)^{-1} \tag{2.68}$$

**Remark 2.2.14** *The total charge of the density is conserved by the projection onto $\Omega_{h_{\tilde{n}}}^{(\infty)}$ and by reconstruction on $\Omega_{h_n}^{(\infty)}$, i.e*

$$\sum_{\boldsymbol{i} \in I_{h_{\tilde{n}}}} \hat{\rho}_{h_{\tilde{n}}}^c(\boldsymbol{i}h_{\tilde{n}}) \omega_{\tilde{n}} = Q. \tag{2.69}$$

**Enhanced sub-grids (PIC-ESg)**

This correction consists in enhancing the sub-grids for the resolution of the electric field by introducing sub-grids more refined than those used for the projection of the density. The sub-grids carrying the electric field are considered with discretization $h_{\tilde{\mathbf{l}}} = h_{\mathbf{l}} \cdot h_{\delta n}$, $\delta n \in \mathbb{N}$ being a parameter denoting the additional depth of these enhanced sub-grids. The corrected scheme, that we shall name enhanced sub-grid PIC scheme is similar to the PIC-Sg scheme except that after the projection, on each sub-grid, the partial representation of the density is interpolated to the enhanced sub-grid $\Omega_{h_{\tilde{\mathbf{l}}}}$ with standard interpolation:

$$\hat{\rho}_{h_{\tilde{\mathbf{l}}}}(\mathbf{i}h_{\tilde{\mathbf{l}}}) := \sum_{\mathbf{j} \in I_{h_{\mathbf{l}}}} \alpha_{\mathbf{l},\mathbf{j}} W_{h_{\mathbf{l}},\mathbf{j}}(\mathbf{i}h_{\tilde{\mathbf{l}}}), \quad \text{for } \mathbf{i} \in I_{h_{\tilde{\mathbf{l}}}}, \tag{2.70}$$

where the coefficients $\alpha_{\mathbf{l},\mathbf{j}}$ are determined by interpolation conditions. For linear interpolation the coefficients fall down to the values of the function evaluated at the grid nodes. Eventually, the electric field is computed on the enhanced sub-grids and reconstructed on the non-enhanced sub-grids in a way similar to equation (2.67).

### 2.2.5 Derivations of particle sampling error estimations

In the previous sections, different schemes and improvements of the schemes embedding sparse grid reconstructions have been introduced. For these schemes, the error associated to the sparse grid reconstruction, *i.e.* originating from the spatial discretization by a set of component grids and the sparse grid interpolation operator, has been separated into two main contributions and investigated: the grid-based error and the particle sampling error. On the one hand, the grid-based error has been thoroughly analyzed for both the charge density and the electric field, with estimations depending on the schemes (PIC-HG, PIC-Sg). On the other hand, the particle sampling error estimations derived in those sections are similar for all grid quantities (charge density, electric field) and all schemes. These estimations have allowed us to apprehend the gains offered by sparse grid reconstructions in comparison to traditional PIC schemes. Nonetheless,

the bounds provided in the previous theorems for the particle sampling error are not optimal and can be refined. Indeed, one shall expect a better statistical resolution for the electric potential than for the charge density, caused by the regularization of the inverse Laplacian operator.

The motivation of this section is to derive more accurate particle sampling error estimations. Unlike the previous ones, these estimations shall take into account the regularization effects of the grid operators (inverse Laplacian, gradient). In this section, the distinctions between the different sparse PIC schemes are no longer detailed; the estimations are derived for the grid quantities $\hat{\rho}^c_{h_n}$, $\hat{\Phi}^c_{h_n}$ (, $\hat{\mathbf{E}}^c_{h_n}$) corresponding to the reconstructions from the schemes PIC-Hg, PIC-NSg (and PIC-Sg).

First, let us recall the estimations obtained previously. The charge density reconstruction, defined by the first equation in (2.41), has a particle sampling error that verify the following bound:

$$\left\| \mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right)^{\frac{1}{2}} \right\|_\infty \leq D_1 \|\rho\|_\infty^{\frac{1}{2}} (N h_n)^{-\frac{1}{2}} |\log h_n|^{d-1} + O\left(\left(\frac{|\log h_n|^{d-1}}{N h_n}\right)^{\frac{1}{2}}\right), \tag{2.71}$$

and shall be compared to the standard method particle sampling error:

$$\left\| \mathbb{V}\left(\mathcal{V}(\hat{\rho}_{h_n})\right)^{\frac{1}{2}} \right\|_\infty = D_2 \|\rho\|_\infty^{\frac{1}{2}} (N h_n^d)^{-\frac{1}{2}} + O\left(N^{-\frac{1}{2}}\right), \tag{2.72}$$

where $D_1$, $D_2$ are constants depending on the dimension $d$. The following, more accurate, bound is established.

**Theorem 2.2.15** *The following bound for the particle sampling error of the reconstructed charge density holds true:*

$$\left\| \mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right)^{\frac{1}{2}} \right\|_\infty \leq D_3 \left( Q \|\rho\|_\infty \frac{|\log h_n|^{d-1}}{N h_n} \right)^{\frac{1}{2}} + O\left( |\log h_n|^{d-1} N^{-\frac{1}{2}} \right), \tag{2.73}$$

*where $D_3$ is a constant depending on the dimension $d$. For two dimensional computations it falls down to:*

$$D_3 = \sqrt{\left(\frac{24}{\log(2)}\right)}. \tag{2.74}$$

The constant $D_3$ of the estimation is not optimal. Indeed, the cancellations resulting from the positive and negative contributions in the combination have not been taken into account in the proof. Nonetheless, this estimation shall provide an upper bound for the statistical error. The estimation shall be compared with the previous bound provided in equation (2.71). The difference between this estimation and the new one is the $|\log h_n|^{d-1}$ term which is refined into a $|\log h_n|^{\frac{d-1}{2}}$ term. This novel estimation lead to the following result:

**Corollary 2.2.16** *Let $P_c \in \mathbb{N}$ being an integer representing the mean number of particles per cell, and considering a total number of particles $N$ defined by the following equations for the sparse-PIC and standard schemes:*

$$N_{std} = P_c h_n^{-d}, \quad N_{sparse} = P_c \left( \sum_{\boldsymbol{l} \in \mathbb{L}_n} |c_{\boldsymbol{l}}| h_{l_1} \dots h_{l_d} \right)^{-1}. \tag{2.75}$$

*Then the particle sampling error on the charge density is comparable between the two methods:*

$$\mathbb{V}\left(\mathcal{V}(\hat{\rho}_{h_n})\right)^{\frac{1}{2}} = \left(\frac{2}{3}\right)^{\frac{d}{2}} \left(\frac{Q\rho}{P_c}\right)^{\frac{1}{2}}, \quad \mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right)^{\frac{1}{2}} \leq \left(\frac{CQ\rho}{P_c}\right)^{\frac{1}{2}}, \tag{2.76}$$

*where $C$ is a constant that depends only on the dimension. E.g. for two dimensional computations:*

$$C = \sqrt{\frac{8}{\log(2)}}$$

The major consequence of this result is the following: by choosing the number of total particles according to equation (2.75) with a mean number of particle per cell $P_c$, the particle sampling error (for the charge density) of the standard and sparse grid methods shall be of the same order.

Let us now derive an estimation of the particle sampling error for the electric potential. Following the steps of theorem 2.2.7 proof, the following bound for the recombined electric potential can be obtained thanks to a maximum principle:

$$\left\| \mathbb{V}\left(\mathcal{V}(\hat{\Phi}^c_{h_n})\right)^{\frac{1}{2}} \right\|_\infty \leq D\|\rho\|_\infty^{\frac{1}{2}} (Nh_n)^{-\frac{1}{2}} |\log h_n|^{d-1} + O\left(\left(\frac{|\log h_n|^{d-1}}{Nh_n}\right)^{\frac{1}{2}}\right), \tag{2.77}$$

where $D$ is a constant depending on the dimension $d$. In order to derive more accurate estimations, we need some additional hypothesizes:

i) A decomposition of the Poisson problem for the grid-based error and the particle sampling error is assumed, *i.e.*

$$-\varepsilon_0 \nabla^2_{h_\mathbf{l}} \mathcal{V}(\hat{\Phi}_{h_\mathbf{l}}) = \mathcal{V}(\hat{\rho}_{h_\mathbf{l}}), \quad -\varepsilon_0 \nabla^2_{h_\mathbf{l}} \mathrm{Bias}(\hat{\Phi}_{h_\mathbf{l}}) = \mathrm{Bias}(\hat{\rho}_{h_\mathbf{l}}), \tag{2.78}$$

ii) The variance of the particle sampling errors is assumed to be constant on the space domain, *i.e.*

$$\mathcal{V}(\hat{u}_{h_\mathbf{l};\mathbf{j}}) = \mathcal{V}(\hat{u}_{h_\mathbf{l}}), \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}, \tag{2.79}$$

where $\hat{u}_{h_\mathbf{l}}$ is the statistical estimator of $u$, being the charge density, electric potential or field. This hypothesis is usually assumed in PIC simulations. Indeed, the statistical error is regulated by the average number of particles per cell $P_c$, assuming therefore the variance to be constant from one cell to another.

**Theorem 2.2.17** *Assuming the hypothesizes i) and ii), the following bound on the particle sampling error holds for the reconstructed electric potential:*

$$\left\| \mathbb{V}\left(\mathcal{V}(\hat{\Phi}^c_{h_n})\right)^{\frac{1}{2}} \right\|_\infty \leq D_1\|\rho\|_\infty^{\frac{1}{2}} |\log h_n|^{d-1} N^{-\frac{1}{2}} + O\left(|\log h_n|^{d-\frac{3}{2}} N^{-\frac{1}{2}}\right), \tag{2.80}$$

*where $D_1$ is a constant depending on the dimension.*

**Remark 2.2.18** *Assuming that the hypothesis i) is valid also for the differentiation of the field and applying the same arguments in the proof, a similar bound can also be obtained for the*

*reconstructed electric field:*

$$\left\| \mathbb{V}\left(\mathcal{V}(\hat{\boldsymbol{E}}^c_{h_n})\right)^{\frac{1}{2}} \right\|_\infty \leq \begin{cases} D_2 \|\rho\|_\infty^{\frac{1}{2}} |\log h_n|^{d-1} |\log h_n|^{\frac{1}{2}} N^{-\frac{1}{2}}, & d = 2 \\ D_3 \|\rho\|_\infty^{\frac{1}{2}} |\log h_n|^{d-1} h_n^{-\frac{1}{6}} N^{-\frac{1}{2}}, & d = 3 \end{cases}$$

*where $D_2$ and $D_3$ are constants depending on the dimension.*

# Proofs of chapter 2

Some of the proofs provided herein are limited to two-dimensional geometries. Nonetheless, they are readily extendable to an arbitrary number of dimensions, using the same tools, however with an additional complexity of notation avoided within the present chapter. First, let us explicit the notation used in theorem 2.1.3:

$$\sum_{m=1}^{d} \sum_{\substack{\{i_1,\dots,i_m\} \\ \subset\{1,\dots,d\} \\ i_j \neq i_k}} a_{i_1,\dots,i_m}(\mathbf{x}; h_{l_{i_1}}, \dots, h_{l_{i_m}}) h^2_{l_{i_1}} \dots h^2_{l_{i_m}}. \tag{2.81}$$

The second sum is constituted of all sets of $m$ different integers contained in the set $\{1, \dots, d\}$. For example, if $m = d$, the second sum has only one term corresponding to $i_1, \dots, i_m = \{1, \dots, d\}$. On the opposite, if $m = 1$, the second sum contains $m$ terms corresponding to $i_1 = 1, \dots, d$. Note that the sets of integers $\{i_1, \dots, i_m\}$, $\{i_2, i_1, \dots, i_m\}$ and others are equals and count as one term. Let us now explicit the notation for the cases of our interest, that is for $d = 2$, where the sums fall down to three terms:

$$\sum_{m=1}^{2} \sum_{\substack{\{i_1,\dots,i_m\} \\ \subset\{1,\dots,d\} \\ i_j \neq i_k}} a_{i_1,\dots,i_m}(\mathbf{x}; h_{l_{i_1}}, \dots, h_{l_{i_m}}) h^2_{l_{i_1}} \dots h^2_{l_{i_m}} =$$

$$a_1(\mathbf{x}; h_{l_1}) h^2_{l_1} + a_2(\mathbf{x}; h_{l_2}) h^2_{l_2} + a_{1,2}(\mathbf{x}; h_{l_1}, h_{l_2}) h^2_{l_1} h^2_{l_2}, \tag{2.82}$$

and for $d = 3$, where the sums fall down to seven terms:

$$\sum_{m=1}^{3} \sum_{\substack{\{i_1,\dots,i_m\} \\ \subset\{1,\dots,d\} \\ i_j \neq i_k}} a_{i_1,\dots,i_m}(\mathbf{x}; h_{l_{i_1}}, \dots, h_{l_{i_m}}) h^2_{l_{i_1}} \dots h^2_{l_{i_m}} =$$

$$\begin{aligned} &a_1(\mathbf{x}; h_{l_1}) h^2_{l_1} + a_2(\mathbf{x}; h_{l_2}) h^2_{l_2} + a_3(\mathbf{x}; h_{l_3}) h^2_{l_3} \\ &+ a_{1,2}(\mathbf{x}; h_{l_1}, h_{l_2}) h^2_{l_1} h^2_{l_2} + a_{1,3}(\mathbf{x}; h_{l_1}, h_{l_3}) h^2_{l_1} h^2_{l_3} + a_{2,3}(\mathbf{x}; h_{l_2}, h_{l_3}) h^2_{l_2} h^2_{l_3} \\ &+ a_{1,2,3}(\mathbf{x}; h_{l_1}, h_{l_2}, h_{l_3}) h^2_{l_1} h^2_{l_2} h^2_{l_3}. \end{aligned} \tag{2.83}$$

In order to prove the theorem 2.1.3, we introduce the following lemma.

**Lemma 2.2.19** *Let $l, m \in \mathbb{N}$ such that $0 \leq l \leq n$ and $0 \leq m \leq d-1$, then*

$$\sum_{r=0}^{d-1} (-1)^r \binom{d-1}{r} \binom{n+d-2-r-l}{d-1-m} = \begin{cases} 1 & \text{if } m = 0, \\ 0 & \text{else.} \end{cases} \tag{2.84}$$

**Lemma 2.2.20** *The combination coefficients verify an unit sum property:*

$$\sum_{\boldsymbol{l}\in\mathbb{L}_n} c_{\boldsymbol{l}} = 1. \tag{2.85}$$

*Proof of lemma 2.2.19.* First, one shall demonstrate the following relation by induction for $0 \leq i \leq d-2$ :

$$\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}r^j = 0, \quad \forall\, 0 \leq j \leq i. \tag{2.86}$$

The relation is immediate for $i = 0$ with the binomial theorem. Let $i \in [\![0, ..., d-3]\!]$ and assume that (2.86) holds for $i$, then:

$$\begin{aligned}
\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}r^{i+1} &= \sum_{r=1}^{d-1}(-1)^r\binom{d-1}{r}r^{i+1} \\
&= (d-1)\sum_{r=1}^{d-1}(-1)^r\binom{d-2}{r-1}r^i \\
&= (-1)(d-1)\sum_{r=0}^{d-2}(-1)^r\binom{d-2}{r}(r+1)^i \\
&= (-1)(d-1)\sum_{j=0}^{i}\binom{i}{j}\sum_{r=0}^{d-2}(-1)^r\binom{d-2}{r}r^j \\
&= 0.
\end{aligned}$$

where the relation $\dfrac{d-1}{r}\binom{d-2}{r-1} = \binom{d-1}{r}$, the binomial theorem and the induction hypothesis (where $i \leq d-3$) have been used in the last relations. From this, the expression falls down to:

$$\begin{aligned}
&\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}\binom{n+d-2-r-l}{d-1-m} \\
&\qquad = \sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}\frac{1}{(d-1)!}\underbrace{(n-r-l+m)...(n-r-l+d-2)}_{d-1-m \text{ terms}} \\
&\qquad = \begin{cases} \dfrac{1}{(d-1)!}\displaystyle\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}(-r)^{d-1} & \text{if } m = 0, \\ 0 & \text{else,} \end{cases}
\end{aligned}$$

because if $m \neq 0$ all exponent of $r$ in the product are less than $d-1$. Then, if $m = 0$, by applying $d-2$ times the previous calculations, one gets:

$$\begin{aligned}
\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}(-r)^{d-1} &= (-1)^{d-2}(-1)^{d-1}\frac{(d-1)!}{(d-1)!}\sum_{r=0}^{1}(-1)^r\binom{1}{r}r \\
&= 1. \quad \square
\end{aligned}$$

*Proof of lemma 2.2.20.* The sum can be recast into:

$$\begin{aligned}\sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} &= \sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}\sum_{|\mathbf{l}|=n+d-1-r}\\ &= \sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}\binom{n+d-2-r}{d-1}\\ &= 1,\end{aligned}$$

since there are $\binom{n+d-2-r}{d-1}$ ways of representing the sum $n+d-1-r$ with $d$ positive integer and owing to the lemma 2.2.19. □

*Proof of theorem 2.1.3.* With the lemma 2.2.20, the following relation holds according to the assumption (2.14):

$$u^c_{h_n} - u = \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}}(u_{h_{\mathbf{l}}} - u) = \sum_{m=1}^{d} B_m,$$

where:

$$B_m = \sum_{\substack{\{i_1,\dots,i_m\}\\ \subset\{1,\dots,d\}}}\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}\sum_{|\mathbf{l}|=n+d-1-r} a_{i_1,\dots,i_m}(\cdot;h_{l_{i_1}},\dots,h_{l_{i_m}})h^2_{l_{i_1}}\dots h^2_{l_{i_m}}.$$

Let us first consider the case $1 \le m \le d-1$, the term $B_m$ can be recast into:

$$\begin{aligned}B_m &= \sum_{\substack{\{i_1,\dots,i_m\}\\ \subset\{1,\dots,d\}}}\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}\sum_{\substack{\sum_{j=1}^m l_{i_j}\le\\ n+m-1-r}} a_{i_1,\dots,i_m}(\cdot;h_{l_{i_1}},\dots,h_{l_{i_m}})h^2_{l_{i_1}}\dots h^2_{l_{i_m}}D_{n,d,m}(r)\\ &= \sum_{\substack{\{i_1,\dots,i_m\}\\ \subset\{1,\dots,d\}}}\left(\sum_{\substack{\sum_{j=1}^m l_{i_j}\le\\ n+m-1-(d-1)}} a_{i_1,\dots,i_m}(\cdot;h_{l_{i_1}},\dots,h_{l_{i_m}})h^2_{l_{i_1}}\dots h^2_{l_{i_m}}\left(\sum_{r=0}^{d-1}(-1)^r\binom{d-1}{r}D_{n,d,m}(r)\right)\right.\\ &\quad\left.+\sum_{p=0}^{d-2}\sum_{\substack{\sum_{j=1}^m l_{i_j}=\\ n+m-1-p}} a_{i_1,\dots,i_m}(\cdot;h_{l_{i_1}},\dots,h_{l_{i_m}})h^2_{l_{i_1}}\dots h^2_{l_{i_m}}\sum_{t=0}^{p}(-1)^t\binom{d-1}{t}D_{n,d,m}(r)\right)\end{aligned}$$

where $D_{n,d,m}(r) = \binom{n+d-2-r-\sum_{j=1}^m l_{i_j}}{d-1-m}$. From lemma 2.2.19, applied with $l = \sum_{j=1}^m l_{i_j}$, $m \ge 1$, the first term in the last expression vanishes and the term $B_m$ falls down to:

$$B_m = h_n^2 E_{d,m}\sum_{\substack{\{i_1,\dots,i_m\}\\ \subset\{1,\dots,d\}}}\sum_{\substack{\sum_{j=1}^m l_{i_j}=\\ n+m-1-p}} a_{i_1,\dots,i_m}(\cdot;h_{l_{i_1}},\dots,h_{l_{i_m}}),$$

where $E_{d,m} = \sum_{p=0}^{d-2} 2^{-2(m-1-p)} \sum_{r=0}^{p} (-1)^r \binom{d-1}{r} \binom{d-m-1+p-r}{d-1-m}$. Since the outermost sum contains $\binom{d}{m}$ terms and because the functions $\tilde{a}_{i_1,\dots,i_m}$ are bounded, the following majoration holds:

$$\begin{aligned} |B_m| &\leq h_n^2 E_{d,m} \left( \kappa \binom{d}{m} \binom{n+m-2}{m-1} + O(1) \right) \\ &\leq h_n^2 E_{d,m} \left( \frac{\kappa \binom{d}{m}}{(m-1)!} (n+m-2)^{m-1} + O(1) \right) \end{aligned}$$

Eventually the last term for $m = d$ can be recast into:

$$\begin{aligned} B_d &= \sum_{r=0}^{d-1} (-1)^r \binom{d-1}{r} \sum_{|\mathbf{l}|=n+d-1-r} a_{1,\dots,d}(\cdot; h_{l_1}, \dots, h_{l_d}) h_{l_1}^2 \dots h_{l_d}^2 \\ &= h_n^2 \tilde{a}_{1,\dots,d} \left( \sum_{r=0}^{d-1} A_r \binom{n+d-1-r}{d-1} + O(1) \right), \end{aligned}$$

where $A_r = 2^{-2(d-1-r)} (-1)^r \binom{d-1}{r}$. □

*Proof of proposition 2.2.2.* First, the expected value of the estimator can be written as follow because of the linearity of the expected value and the symmetry of the basis function:

$$\mathbb{E}(\hat{\rho}_{h_\mathbf{l}}(\mathbf{x})) = \frac{Q}{h_{l_1} \dots h_{l_d}} \iint_{\Omega_x \times \Omega_v} W\left( \frac{\xi_1 - x_1}{h_{l_1}} \right) \dots W\left( \frac{\xi_d - x_d}{h_{l_d}} \right) f(\boldsymbol{\xi}, \mathbf{v}) d\mathbf{v} d\boldsymbol{\xi}. \tag{2.87}$$

Considering the following change of variable for $1 \leq i \leq d$:

$$y_i = h_{l_i}^{-1} (\xi_i - x_i), \tag{2.88}$$

one gets:

$$\mathbb{E}(\hat{\rho}_{h_\mathbf{l}}(\mathbf{x})) = Q \iint_{[-h_{l_i}^{-1}, h_{l_i}^{-1}] \times \Omega_v} W(y_1) \dots W(y_d) f(x_1 + y_1 h_{l_1}, \dots, \mathbf{v}) d\mathbf{v} d\mathbf{y}. \tag{2.89}$$

Let us now consider uni-dimensional Taylor expansions at second order in each dimension of the probability density function $f$ and combine them to obtain:

$$f(x_1 + y_1 h_{l_1}, \dots, \mathbf{v}) = \sum_{|\boldsymbol{\alpha}|_\infty \leq 2} \frac{\partial_1^{\alpha_1} \dots \partial_d^{\alpha_d} f(\mathbf{x}, \mathbf{v})}{\alpha_1! \dots \alpha_d!} y_1^{\alpha_1} h_{l_1}^{\alpha_1} \dots y_d^{\alpha_d} h_{l_d}^{\alpha_d} + o\left( h_{l_1}^2, \dots, h_{l_d}^2 \right). \tag{2.90}$$

Introducing the expansion in the last expression, separating the integrals and restricting the first

integrals on the supports of the basis functions, it holds:

$$\mathbb{E}(\hat{\rho}_{h_\mathbf{l}}(\mathbf{x})) = \sum_{|\boldsymbol{\alpha}|_\infty \leq 2} h_{l_1}^{\alpha_1} ... h_{l_d}^{\alpha_d} \left( \int_{[-1,1]} W(y_1) y_1^{\alpha_1} dy_1 \right) ... \left( \int_{[-1,1]} W(y_d) y_d^{\alpha_d} dy_d \right)$$
$$\left( Q \int_{\Omega_v} \frac{\partial_1^{\alpha_1} ... \partial_d^{\alpha_d} f(\mathbf{x}, \mathbf{v})}{\alpha_1! ... \alpha_d!} d\mathbf{v} \right) + o\left( h_{l_1}^2, ..., h_{l_d}^2 \right). \tag{2.91}$$

Because of the symmetry of the function $W$, the integrals vanish for all odd exponents, *i.e.*

$$\int_{[-1,1]} W(y) y^\alpha dy = 0 \quad \text{if } \alpha \text{ odd}, \tag{2.92}$$

and thus all terms with at least one odd exponent vanish in the sum. We conclude with the relations:

$$Q \int_{\Omega_v} \partial_1^{\alpha_1} ... \partial_d^{\alpha_d} f(\mathbf{x}, \mathbf{v}) d\mathbf{v} = \partial_1^{\alpha_1} ... \partial_d^{\alpha_d} \rho(\mathbf{x}), \quad \int_{[-1,1]} W(y) y^2 dy = \frac{1}{6}, \quad \int_{[-1,1]} W(y) dy = 1.$$

From the definition of the particle sampling error, it is obvious that its expected value is zero. On the other hand, owing to Bienaymé's equality, since the positions of the particles are supposed independent, the variance of the particle sampling error is:

$$\begin{aligned}
\mathbb{V}(\mathcal{V}(\hat{\rho}_{h_\mathbf{l}})(\mathbf{x})) = \mathbb{V}(\hat{\rho}_{h_\mathbf{l}}(\mathbf{x})) &= \mathbb{V}\left( \frac{Q}{N} \sum_{p=1}^{N} \mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{X}_p) \right) \\
&= \frac{Q^2}{N} \mathbb{V}(\mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{X})) \\
&= \frac{Q^2}{N} \mathbb{E}\left( (\mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{X}) - \mathbb{E}(\mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{X})))^2 \right) \\
&= \frac{Q^2}{N} \left( \mathbb{E}(\mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{X})^2) - \mathbb{E}(\mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{X}))^2 \right).
\end{aligned}$$

Combining uni-dimensional Taylor expansions at second order in each dimension and change of variables similar to the bias calculations, the first term can be written as:

$$Q\mathbb{E}(\mathcal{S}_{h_\mathbf{l}}(\mathbf{x} - \mathbf{X})^2) = \sum_{|\boldsymbol{\alpha}|_\infty \leq 2} \frac{h_{l_1}^{\alpha_1} ... h_{l_d}^{\alpha_d}}{h_{l_1} ... h_{l_d}} \left( \int_{[-1,1]} W^2(y_1) y_1^{\alpha_1} dy_1 \right) ... \left( \int_{[-1,1]} W^2(y_d) y_d^{\alpha_d} dy_d \right)$$
$$\left( Q \int_{\Omega_v} \frac{\partial_1^{\alpha_1} ... \partial_d^{\alpha_d} f(\mathbf{x}, \mathbf{v})}{\alpha_1! ... \alpha_d!} d\mathbf{v} \right) + o\left( h_{l_1}, ..., h_{l_d} \right). \tag{2.93}$$

Again, because of the symmetry of the function $W^2$, the integrals vanish for odd exponents, *i.e.*

$$\int_{[-1,1]} W^2(y) y^\alpha dy = 0, \quad \text{if } \alpha \text{ odd}. \tag{2.94}$$

Finally, with the relations:

$$\int_{[-1,1]} W^2(y)y^2dy = \frac{1}{15}, \quad \int_{[-1,1]} W^2(y)dy = \frac{2}{3}, \tag{2.95}$$

and neglecting the last term:

$$\mathbb{E}(\mathcal{S}_{h_\mathbf{l}}(\mathbf{x}-\mathbf{X}))^2 = \left(\rho(\mathbf{x}) + O\left(h_{l_1}^2, ..., h_{l_d}^2\right)\right)^2, \tag{2.96}$$

we conclude the proof. □

*Proof of proposition 2.2.3.* Recasting the sums and integral, it holds true:

$$\int_\Omega \hat{\rho}_{h_n}^c(\mathbf{x})d\mathbf{x} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_\mathbf{l} \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \frac{Q}{N} \sum_{p=1}^{N} \mathcal{S}_{h_\mathbf{l}}(\mathbf{j}h_\mathbf{l} - \mathbf{x}_p) \int_\Omega W_{h_\mathbf{l};\mathbf{j}}(\mathbf{x})d\mathbf{x}.$$

We obtain the result with following relations:

$$\sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \mathcal{S}_{h_\mathbf{l}}(\mathbf{j}h_\mathbf{l} - \mathbf{x}_p) = \left(\prod_{i=1}^{d} h_{l_i}\right)^{-1}, \quad \int_\Omega W_{h_\mathbf{l};\mathbf{j}}(\mathbf{x})d\mathbf{x} = \left(\prod_{i=1}^{d} h_{l_i}\right),$$

and thanks to the lemma 2.2.20. □

The following lemma is useful for the proof of proposition 2.2.5.

**Lemma 2.2.21** *Let $\boldsymbol{X}$ be the random variable defined by equation (2.27) and $\boldsymbol{k} \in \mathbb{L}_n$, then*

$$Q \sum_{(\boldsymbol{i},\boldsymbol{j})\in I_{h_\boldsymbol{k}}^2} \mathbb{E}\left(S_{h_\boldsymbol{k}}(\boldsymbol{i}h_\boldsymbol{k} - \boldsymbol{X})S_{h_\boldsymbol{k}}(\boldsymbol{j}h_\boldsymbol{k} - \boldsymbol{X})\right) W_{h_\boldsymbol{k},\boldsymbol{i}}W_{h_\boldsymbol{k},\boldsymbol{j}} \leq \left(\frac{2}{3}\right)^2 \frac{\|\rho\|_\infty}{h_{k_1}h_{k_2}} + O\left(h_{k_1}h_{k_2}\right). \tag{2.97}$$

*Proof of lemma 2.2.21.* Let us consider the one-dimensional problem which can be extended to the two-dimensional case by tensor product of the shape function. Owing to the symmetry of the basis functions, it holds:

$$\begin{aligned} Q\mathbb{E}(\mathcal{S}_{h_k}(ih_k - \mathbf{X})\mathcal{S}_{h_k}(jhk - \mathbf{X})) \\ &= Q \iint_{\Omega_x\times\Omega_v} \frac{1}{h_k^2} W\left(\frac{\xi - ih_k}{h_k}\right) W\left(\frac{\xi - jh_k}{h_k}\right) f(\xi, v, t)d\xi dv \\ &= Q \iint_{\Omega_x\times\Omega_v} \frac{1}{h_k} W\left(x + j - i\right) W\left(y\right) f(jh_k + xh_k, v, t)dydv \end{aligned}$$

where the change of variable $x = h_k^{-1}(\xi - jh_k)$ has been applied. There are two cases to consider either if $i = j$ or not:

i) If $i = j$, a Taylor expansion upon the probability density and because of the symmetry of

the basis functions all $x$-terms with odd exponent vanish:

$$\begin{aligned} Q\mathbb{E}(S_{h_k}(ih_k - X)S_{h_k}(jhk - X)) &= \frac{\rho(jh_k)}{h_k}\int_{-\frac{1}{h_k}}^{\frac{1}{h_k}} W(x)^2 dx + O\left(h_k^2\right) \\ &= \frac{2\rho(jh_k)}{h_k}\int_{-1}^{0}(1+x)^2 dx + O\left(h_k^2\right) \\ &= \frac{2}{3}\frac{\rho(jh_k)}{h_k} + O\left(h_k^2\right) \end{aligned}$$

because $\operatorname{supp} W = [-1, 1]$.

ii) If $i \neq j$, then $x$ and $x + j - i$ are in the support of $W$ if and only if $|j - i| = 1$. A Taylor expansion upon the probability density gives:

$$Q\mathbb{E}(S_{h_k}(ih_k - X)S_{h_k}(jh_k - X)) = \frac{\rho(jh_k)}{h_k}\left(\int_{-\frac{1}{h_k}}^{0}\Psi(x)dx + \int_{0}^{\frac{1}{h_k}}\Psi(x)dx\right) + O(h_k)\,,$$

where:

$$\Psi(y) := W(x)W(x + j - i).$$

Note that the $x$-terms with odd exponent no longer vanish because $\Psi$ is not symmetrical, so that the negligible terms scale as $O(h_k)$. We have:

$$\int_{-\frac{1}{h_k}}^{0}\Psi(x)dx = \begin{cases} 0 & \text{if } j - i = 1, \\ \frac{1}{6} & \text{if } j - i = -1, \end{cases}$$

$$\int_{0}^{\frac{1}{h_k}}\Psi(x)dx = \begin{cases} \frac{1}{6} & \text{if } j - i = 1, \\ 0 & \text{if } j - i = -1. \end{cases}$$

Eventually, using the partition of unit property of the basis function of equation (2.2), one gets:

$$\sum_{(i,j)\in I_{h_k}^2} Q\mathbb{E}(S_{h_k}(ih_k - X)S_{h_k}(jhk - X))W_{h_k,i}W_{h_k,j} \leq \frac{2}{3}\frac{\|\rho\|_\infty}{h_k} + O(h_k)\,. \;\square \tag{2.98}$$

*Proof of proposition 2.2.5.* The proof is provided in two dimensions and the interpolation is considered in nodal basis so that the superscript $\mathcal{N}$ is omitted ($\mathcal{I}_{V_{h_1}}^{\mathcal{N}} = \mathcal{I}_{V_{h_1}}$). In order to apply the theorem 2.1.3, the local error shall verify the relation stated by equation (2.14). The local error can be recast into:

$$\mathcal{I}_{V_{h_1}}\hat{\rho}_{h_1} - \rho = \mathcal{I}_{V_{h_1}}(\hat{\rho}_{h_1} - \rho) + \mathcal{I}_{V_{h_1}}\rho - \rho = \underbrace{\mathcal{I}_{V_{h_1}}\mathcal{V}(\hat{\rho}_{h_1})}_{\text{ii) particle sampling error}} + \underbrace{\mathcal{I}_{V_{h_1}}\text{Bias}(\hat{\rho}_{h_1}) + \mathcal{I}_{V_{h_1}}\rho - \rho}_{\text{i) grid-based error}}\,. \tag{2.99}$$

i) Grid-based error: from lemma 2.2.22, one gets:

$$\mathcal{I}_{V_{h_\mathbf{l}}}\mathrm{Bias}(\hat{\rho}_{h_\mathbf{l}}) = \frac{1}{12}(h_{l_1}^2\partial_1^2\rho + h_{l_2}^2\partial_2^2\rho) + \frac{41}{1296}h_{l_1}^2h_{l_2}^2\partial_{1,2}^2\rho + O\left(h_{l_1}^4, h_{l_2}^4\right). \tag{2.100}$$

$$\mathcal{I}_{V_{h_\mathbf{l}}}\rho - \rho = \frac{4}{27}(h_{l_1}^2\partial_1^2\rho + h_{l_2}^2\partial_2^2\rho) + \frac{16}{729}h_{l_1}^2h_{l_2}^2\partial_{1,2}^2\rho + O\left(h_{l_1}^4, h_{l_2}^4\right) \tag{2.101}$$

Eventually, the assumption (2.14) is verified for the grid-based error with:

$$a_1(\cdot, h_{l_1}) = \frac{25}{108}\partial_1^2\rho + O\left(h_{l_1}^2\right), \quad a_2(\cdot, h_{l_2}) = \frac{25}{108}\partial_2^2\rho + O\left(h_{l_2}^2\right),$$
$$a_{1,2}(\cdot, h_{l_1}, h_{l_2}) = \frac{625}{11664}\partial_1^2\partial_2^2\rho + O\left(h_{l_1}^2h_{l_2}^2\right).$$

From the theorem 2.1.3, we get the result for the grid-based error.

ii) Particle sampling error: the following estimate bound holds:

$$\mathbb{V}\left(\sum_{\mathbf{l}\in\mathbb{L}_n} c_\mathbf{l}\mathcal{I}_{V_{h_\mathbf{l}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}})\right) \leq \sum_{(\mathbf{k},\mathbf{l})\in(\mathbb{L}_n)^2} |c_\mathbf{k}||c_\mathbf{l}| \left|\mathrm{Cov}(\mathcal{I}_{V_{h_\mathbf{k}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{k}}), \mathcal{I}_{V_{h_\mathbf{l}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}}))\right|. \tag{2.102}$$

Let us look at the covariance of a pair of component grids. Since any of the random variables $I_{V_{h_\mathbf{l}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}})$ are centered, equations (2.33), (2.10) and the Cauchy-Schwarz inequality gives:

$$\begin{aligned}\left|\mathrm{Cov}(\mathcal{I}_{V_{h_\mathbf{k}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{k}}), \mathcal{I}_{V_{h_\mathbf{l}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}}))\right| &= \left|\mathbb{E}(\mathcal{I}_{V_{h_\mathbf{k}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{k}})\mathcal{I}_{V_{h_\mathbf{l}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}}))\right| \\ &\leq \left(\mathbb{V}(\mathcal{I}_{V_{h_\mathbf{k}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{k}}))\mathbb{V}\left(\mathcal{I}_{V_{h_\mathbf{l}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}})\right)\right)^{\frac{1}{2}},\end{aligned}$$

where, owing to the linearity of the expected value:

$$\mathbb{V}(\mathcal{I}_{V_{h_\mathbf{k}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{k}})) = \sum_{(\mathbf{i},\mathbf{j})\in I_{h_\mathbf{k}}^2} \mathbb{E}(\mathcal{V}(\hat{\rho}_{h_\mathbf{k}})(\mathbf{i}h_\mathbf{k})\mathcal{V}(\hat{\rho}_{h_\mathbf{k}})(\mathbf{j}h_\mathbf{k}))W_{h_\mathbf{k},\mathbf{i}}W_{h_\mathbf{k},\mathbf{j}}.$$

From equations (2.31) and (2.33), the linearity of the expected value, then recasting the sum upon $(p,q) \in [\![1,N]\!]^2$ into two sums, whether if $p$ is equal to $q$ or not, which contain $N$ and $N(N-1)$ terms, and by independancy of the random variables of the sample in the latter case, we have:

$$\begin{aligned}\mathbb{E}(\mathcal{V}(\hat{\rho}_{h_\mathbf{k}})(\mathbf{i}h_\mathbf{k})\mathcal{V}(\hat{\rho}_{h_\mathbf{k}})(\mathbf{j}h_\mathbf{k})) &= \mathbb{E}(\hat{\rho}_{h_\mathbf{k}}(\mathbf{i}h_\mathbf{k})\hat{\rho}_{h_\mathbf{k}}(\mathbf{j}h_\mathbf{k})) - \mathbb{E}(\hat{\rho}_{h_\mathbf{k}}(\mathbf{i}h_\mathbf{k}))\mathbb{E}(\hat{\rho}_{h_\mathbf{k}}(\mathbf{j}h_\mathbf{k})) \\ &= \frac{Q^2}{N^2}\Bigg(\sum_{(p,q)\in[\![1,N]\!]^2} \mathbb{E}\left(\mathcal{S}_{h_\mathbf{k}}(\mathbf{i}h_\mathbf{k} - \mathbf{X}_p)\mathcal{S}_{h_\mathbf{k}}(\mathbf{j}h_\mathbf{k} - \mathbf{X}_q)\right) \\ &\quad - \left(\sum_{p=1}^N \mathbb{E}(\mathcal{S}_{h_\mathbf{k}}(\mathbf{i}h_\mathbf{k} - \mathbf{X}_p))\right)\left(\sum_{q=1}^N \mathbb{E}(\mathcal{S}_{h_\mathbf{k}}(\mathbf{j}h_\mathbf{k} - \mathbf{X}_q))\right)\Bigg) \\ &= \frac{Q^2}{N}\Big(\mathbb{E}\left(\mathcal{S}_{h_\mathbf{k}}(\mathbf{i}h_\mathbf{k} - \mathbf{X})\mathcal{S}_{h_\mathbf{k}}(\mathbf{j}h_\mathbf{k} - \mathbf{X})\right) - \underbrace{\mathbb{E}(\mathcal{S}_{h_\mathbf{k}}(\mathbf{i}h_\mathbf{k} - \mathbf{X}))(\mathcal{S}_{h_\mathbf{k}}(\mathbf{j}h_\mathbf{k} - \mathbf{X}))}_{=O(1)}\Big).\end{aligned}$$

Finally, owing to lemma 2.2.21 we get:

$$\mathbb{V}\left(\sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}}\mathcal{I}_{V_{h_{\mathbf{l}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}}})\right) \leq |\mathbb{L}_n|^2\left(\max_{\mathbf{l}\in\mathbb{L}_n}|c_{\mathbf{l}}|\right)^2\left(\frac{2}{3}\right)^2\frac{Q\|\rho\|_\infty}{Nh_{n+1}} + O\left(\frac{1}{N}\right)$$
$$\leq \frac{16}{9}n^2\frac{Q\|\rho\|_\infty}{Nh_{n+1}} + O\left(\frac{n}{Nh_n}\right),$$

and we get the following bound on the square root variance:

$$\mathbb{V}\left(\sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}}\mathcal{I}_{V_{h_{\mathbf{l}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}}})\right)^{\frac{1}{2}} \leq \frac{2\sqrt{8}}{3\log(2)}(Q\|\rho\|_\infty)^{\frac{1}{2}}\left(\frac{|\log h_n|^2}{Nh_n}\right)^{\frac{1}{2}} + O\left(\left(\frac{|\log h_n|}{Nh_n}\right)^{\frac{1}{2}}\right). \square \tag{2.103}$$

*Proof of theorem 2.2.6.* Only the guidelines of the proof (in two dimensions) are provided, the details being specified for that of theorem 2.2.7. First the error can be recast into:

$$\mathcal{I}_{V_{hn}^{(\infty)}}\hat{\mathbf{E}}_{h_n} - \mathbf{E} = \mathcal{I}_{V_{hn}^{(\infty)}}(\hat{\mathbf{E}}_{h_n} - \mathbf{E}) + \mathcal{I}_{V_{hn}^{(\infty)}}\mathbf{E} - \mathbf{E} \tag{2.104}$$

Applying lemma 2.2.22 to each component of the electric field we get the following estimate:

$$\|\mathcal{I}_{V_{hn}^{(\infty)}}\mathbf{E} - \mathbf{E}\|_\infty \leq \frac{4}{27}\left(\|\partial_1^2\mathbf{E}\|_\infty + \|\partial_2^2\mathbf{E}\|_\infty\right) + \frac{16}{729}\left(\|\partial_1^2\partial_2^2\mathbf{E}\|_\infty\right). \tag{2.105}$$

From lemma 2.2.23 the following estimate can be stated:

$$\Delta_{h_n}\Phi(\mathbf{j}h_n) = -\rho(\mathbf{j}h_n) + \frac{h_n^2}{12}(\partial_1^4\Phi(\mathbf{j}h_n) + \partial_2^4\Phi(\mathbf{j}h_n)) + O\left(h_n^3\right), \tag{2.106}$$

$\Delta_{h_n}$ denoting the finite difference discrete Laplacian. Introducing the sparse grid reconstruction of the density and using invertibility of the finite difference operator, one gets:

$$\Phi(\mathbf{j}h_n) - \hat{\Phi}_{h_n}(\mathbf{j}h_n) \tag{2.107}$$
$$= (\Delta_{h_n})^{-1}\left(\hat{\rho}^c_{h_n}(\mathbf{j}h_n) - \rho(\mathbf{j}h_n) + \frac{h_n^2}{12}(\partial_1^4\Phi(\mathbf{j}h_n) + \partial_2^4\Phi(\mathbf{j}h_n)) + O\left(h_n^3\right)\right),$$

where $\Phi_{h_n}$ is the solution of the discrete problem. Applying the finite difference gradient operator, denoted $\boldsymbol{\nabla}_{h_n}$, to equation (2.107), owing to linearity and commutativity of the operators $\boldsymbol{\nabla}_{h_n}$ and $(\Delta_{h_n})^{-1}$ and introducing the following truncation error of the discrete gradient:

$$\boldsymbol{\nabla}\Phi(\mathbf{j}h_n) - \boldsymbol{\nabla}_{h_n}\Phi(\mathbf{j}h_n) = \frac{h_n^2}{3}\boldsymbol{\nabla}^3\Phi(\mathbf{j}h_n) + O\left(h_n^3\right), \tag{2.108}$$

one gets summing equations (2.107) and (2.108):

$$\hat{\mathbf{E}}_{h_n}(\mathbf{j}h_n) - \mathbf{E}(\mathbf{j}h_n) = (\Delta_{h_n})^{-1}\left(\boldsymbol{\nabla}_{h_n}(\hat{\rho}^c_{h_n}(\mathbf{j}h_n) - \rho(\mathbf{j}h_n)) + \frac{h_n^2}{12}\boldsymbol{\nabla}_{h_n}(\partial_1^4\Phi(\mathbf{j}h_n) + \partial_2^4\Phi(\mathbf{j}h_n))\right.$$
$$\left. + \boldsymbol{\nabla}_{h_n}O\left(h_n^3\right)\right) + \frac{h_n^2}{3}\boldsymbol{\nabla}^3\Phi(\mathbf{j}h_n) + O\left(h_n^3\right).$$

From lemma 2.2.24, we have:

$$\|(\Delta_{h_n})^{-1}\|_\infty \leq \frac{1}{8}. \tag{2.109}$$

Using Taylor expansions, we get:

$$\boldsymbol{\nabla}_{h_n}(\hat{\rho}^c_{h_n} - \rho) = \boldsymbol{\nabla}(\hat{\rho}^c_{h_n} - \rho) + O\left(h_n^3\right), \tag{2.110}$$

$$\boldsymbol{\nabla}_{h_n}\left(\partial_1^4\Phi + \partial_2^4\Phi\right) = \boldsymbol{\nabla}\left(\partial_1^4\Phi + \partial_2^4\Phi\right) + O\left(h_n^3\right), \quad \boldsymbol{\nabla}_{h_n} O\left(h_n^3\right) = O\left(h_n^3\right). \tag{2.111}$$

Eventually, applying lemma 2.2.22 to each component of last equation, one gets the result. □

The proof of the theorem 2.2.7 is provided in two dimensions and requires discrete operator notations, defined in the notation section, and semi-discrete notations that we shall introduce. Let us introduce the notation $\{i_1, ..., i_m\} \in \mathbb{I} := \{\emptyset, \{1\}, \{2\}, \{1,2\}\}$, for $0 \leq m \leq 2$, standing for

$$\{i_1, ..., i_m\} := \begin{cases} \emptyset & \text{if } m = 0, \\ \{1\} \text{ or } \{2\} & \text{if } m = 1, \\ \{1,2\} & \text{if } m = 2. \end{cases} \tag{2.112}$$

The quantities (functions, domains, operators, etc.) associated with this notation correspond either to continuous ones for $m = 0$, semi-discrete ones for $m = 1$ or discrete ones for $m = 2$. Then, we introduce the semi-discrete operators defined by:

$$(\Delta_{h_{\mathbf{l}}})^{(i_1,...,i_m)} u = \sum_{k \in \{i_1,...,i_m\}} \delta^+_{k,h_{l_k}} \delta^-_{k,h_{l_k}} u + \sum_{\substack{k \in \{1,...,d\} \\ k \notin \{i_1,...,i_m\}}} \partial_k^2 u, \tag{2.113}$$

where $u$ is defined on the following hyperplane:

$$\Omega^{(i_1,...,i_m)}_{h_{\mathbf{l}}} := \left\{ \mathbf{x} \in \Omega \mid x_{i_k} \in \{j h_{l_{i_k}} \mid 0 \leq j \leq h^{-1}_{l_{i_k}}\}, 1 \leq k \leq m \right\} \tag{2.114}$$

for $m \geq 1$ and $\Omega^{\emptyset}_{h_{\mathbf{l}}} := \Omega$. In order to prove the proposition 2.2.7, we need the following lemma:

**Lemma 2.2.22 (Semi-discrete interpolation error on $V_{h_{\mathbf{l}}}$)** *Let $f \in X^2\left(\Omega^{(i_1,...,i_m)}_{h_{\mathbf{l}}}\right)$, where $\{i_1, ..., i_m\} \in \mathbb{I}$ and $0 \leq m \leq 1$, then the local error at $\boldsymbol{x} \in \Omega^{(i_1,...,i_m)}_{h_{\mathbf{l}}}$ of the interpolation onto $V_{h_{\mathbf{l}}}$ is:*

$$\mathcal{I}_{V_{h_{\mathbf{l}}}} f(\boldsymbol{x}) - f(\boldsymbol{x}) = \tag{2.115}$$
$$\sum_{k=1}^{2-m} \sum_{\substack{\{r_1,...,r_k\} \subset \{1,2\} \\ s.t.\ \{r_1,...,r_k\} \cap \{i_1,...,i_m\} = \emptyset}} d_{i_1,...,i_m;r_1,...,r_k}(\boldsymbol{x}; h_{l_{i_1}}, ..., h_{l_{i_m}}; h_{l_{r_1}}, ..., h_{l_{r_k}}) h^2_{l_{r_1}} ... h^2_{l_{r_k}},$$

*where*

$$\|d_{i_1,...,i_m;r_1,...,r_k}(\cdot; h_{l_{i_1}}, ..., h_{l_{i_m}}; h_{l_{r_1}}, ..., h_{l_{r_k}})\|_\infty \leq \left(\frac{4}{27}\right)^k \|\partial^2_{r_1} ... \partial^2_{r_k} f\|_\infty \tag{2.116}$$

**Lemma 2.2.23 (Truncation error of semi-discrete laplacian)** *Let $f \in C^4\left(\Omega^{(i_1,...,i_m)}_{h_{\mathbf{l}}}\right)$, where*

$\{i_1, ..., i_m\} \in \mathbb{I}$ *and* $0 \leq m \leq 1$, *then:*

$$(\Delta_{h_l} - (\Delta_{h_l})^{(i_1,...,i_m)}) f = \sum_{\substack{k \in \{1,...,d\} \\ k \notin \{i_1,...,i_m\}}} \tau_k(\cdot; h_{l_k}) h_{l_k}^2, \tag{2.117}$$

*with:*

$$\|\tau_k(\cdot; h_{l_k})\|_\infty \leq \frac{1}{12} \|\partial_k^4 f\|_\infty \tag{2.118}$$

**Lemma 2.2.24** *Let* $f \in C^2\left(\Omega_{h_l}^{(i_1,...,i_m)}\right)$, $w \in C^0\left(\Omega_{h_l}^{(i_1,...,i_m)}\right)$, *where* $\{i_1, ..., i_m\} \in \mathbb{I}$ *and* $0 \leq m \leq 2$, *verifying the semi-discrete problem:*

$$(\Delta_{h_l})^{(i_1,...,i_m)} f = w, \quad f_{|\partial\Omega} = 0, \tag{2.119}$$

*then the following majoration holds:*

$$\|f\|_\infty \leq \frac{1}{8} \|w\|_\infty, \quad i.e. \quad \left\| \left( (\Delta_{h_l})^{(i_1,...,i_m)} \right)^{-1} \right\|_\infty \leq \frac{1}{8} \tag{2.120}$$

*Proof of lemma 2.2.22.* Let $\mathbf{x} \in \Omega_{h_\mathbf{l}}^{(i_1,...,i_m)}$, owing to the partition of unit property of the basis functions (2.2):

$$\mathcal{I}_{V_{h_\mathbf{l}}} f(\mathbf{x}) - f(\mathbf{x}) = \sum_{\mathbf{j} \in I_{h_\mathbf{l}}} (f(\mathbf{j}h_\mathbf{l}) - f(\mathbf{x})) W_{h_\mathbf{l},\mathbf{j}}(\mathbf{x}) \tag{2.121}$$

Let us introduce the notation $(y_{l_1,j_1}, y_{l_2,j_2}) \in \mathbb{R}^2$ defined by:

$$\mathbf{j}h_\mathbf{l} - \mathbf{x} = (y_{l_1,j_1}, y_{l_2,j_2}) h_\mathbf{l}, \quad y_{l_k,j_k} = \begin{cases} j_k - \frac{x_k}{h_{l_k}} & \text{if } k \notin \{i_1, ..., i_m\}, \\ 0 \quad \text{else.} \end{cases} \tag{2.122}$$

Because of the support of the basis functions:

$$|W_{h_\mathbf{l},\mathbf{j}}(\mathbf{x})| = 0 \iff \max(|y_{l_1,j_1}|, |y_{l_2,j_2}|) \geq 1, \tag{2.123}$$

and the sum in equation (2.121) falls down to four terms verifying $\max(|y_{l_1,j_1}|, |y_{l_2,j_2}|) < 1$. Let $k \in [\![1, ..., 2-m]\!]$ and $r_1, ..., r_k$ such that $\{r_1, ..., r_k\} \subset \{1, 2\}$ and $\{r_1, ..., r_k\} \cap \{i_1, ..., i_m\} = \emptyset$, then combining uni-dimensional Taylor expansions of $f$ at second order in dimensions $r_1, ..., r_k$, we obtain the relation:

$$f(\mathbf{j}h_\mathbf{l}) = \sum_{\substack{\boldsymbol{\alpha} \in \mathbb{N}^k \\ |\boldsymbol{\alpha}|_\infty \leq 2}} \frac{\partial_{r_1}^{\alpha_1} ... \partial_{r_k}^{\alpha_k} f(\mathbf{x})}{\alpha_1! ... \alpha_d!} y_{l_{r_1},j_{r_1}}^{\alpha_1} ... y_{l_{r_k},j_{r_k}}^{\alpha_k} h_{l_{r_1}}^{\alpha_1} ... h_{l_{r_k}}^{\alpha_k} + o\left(h_{l_{r_1}}^2, ..., h_{l_{r_k}}^2\right). \tag{2.124}$$

whose sum with basis functions vanishes for odd exponent:

$$\sum_{\mathbf{j} \in I_{h_\mathbf{l}}} y_{l_{r_1},j_{r_1}}^{\alpha_1} ... y_{l_{r_k},j_{r_k}}^{\alpha_k} W_{h_\mathbf{l},\mathbf{j}} = 0 \quad \text{if } \exists\, i \in \{1, ..., k\} \; s.t. \; \alpha_i \text{ odd.} \tag{2.125}$$

The first result follows with:

$$e_{i_1,\dots,i_m;r_1,\dots,r_k} = 2^{-k}\partial_{r_1}^2\dots\partial_{r_k}^2 f \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} y_{l_{r_1},j_{r_1}}^2\dots y_{l_{r_k},j_{r_k}}^2 W_{h_\mathbf{l},\mathbf{j}} + o(1)\,, \tag{2.126}$$

where the sum can be recast into:

$$\begin{aligned}\sum_{\mathbf{j}\in I_{h_\mathbf{l}}} y_{l_{r_1},j_{r_1}}^2\dots y_{l_{r_k},j_{r_k}}^2 W_{h_\mathbf{l},\mathbf{j}} &= y_{l_{r_1},j_{r_1}}^2\dots y_{l_{r_k},j_{r_k}}^2(1-y_{l_{r_1},j_{r_1}})\dots(1-y_{l_{r_k},j_{r_k}}) \\ &+ y_{l_{r_1},j_{r_1}}^2\dots y_{l_{r_{k-1}},j_{r_{k-1}}}^2 y_{l_{r_k},j_{r_k}}(1-y_{j_{r_1}})\dots(1-y_{l_{r_{k-1}},j_{r_{k-1}}})(1-y_{l_{r_k},j_{r_k}})^2 \\ &+ \dots + y_{l_{r_1},j_{r_1}}\dots y_{l_{r_k},j_{r_k}}(1-y_{l_{r_1},j_{r_1}})^2\dots(1-y_{l_{r_k},j_{r_k}})^2,\end{aligned} \tag{2.127}$$

and because the functions $y \mapsto y^2(1-y)$, $y \mapsto y(1-y)^2$ are bounded on $[0,1]$ by $\dfrac{4}{27}$, the following estimation holds:

$$\left|\sum_{\mathbf{j}\in I_{h_\mathbf{l}}} y_{l_{r_1},j_{r_1}}^2\dots y_{l_{r_k},j_{r_k}}^2 W_{h_\mathbf{l},\mathbf{j}}\right| \leq 2^k\left(\frac{4}{27}\right)^k \tag{2.128}$$

which gives the results. □

*Proof of lemma 2.2.23.* For all $k \in \{1,\dots,d\}$ such that $k \notin \{i_1,\dots,i_m\}$, since $f \in C^4(\Omega_{h_\mathbf{l}}^{(i_1,\dots,i_m)})$, a Taylor expansion in dimension $k$ gives the relations:

$$f((\mathbf{j}+\mathbf{i}_k)h_\mathbf{l}) - f(\mathbf{j}h_\mathbf{l}) = \sum_{\alpha=1}^{4}\frac{h_{l_k}^4}{\alpha!}\partial_k^4 f + O\left(h_{l_k}^5\right), \tag{2.129}$$

$$f((\mathbf{j}-\mathbf{i}_k)h_\mathbf{l}) - f(\mathbf{j}h_\mathbf{l}) = \sum_{\alpha=1}^{4}\frac{(-h_{l_k})^4}{\alpha!}\partial_k^4 f + O\left(h_{l_k}^5\right), \tag{2.130}$$

$$\tag{2.131}$$

Then adding the two relations, dividing by $h_{l_k}^2$ and summing upon the dimensions $k$, one gets the result. □

*Proof of lemma 2.2.24.* Let $\Psi \in C^2(\Omega_{h_\mathbf{l}}^{(i_1,\dots,i_m)})$ be a nonnegative function defined by:

$$\Psi(\mathbf{x}) = \frac{1}{4}\left(\left(x_1-\frac{1}{2}\right)^2 + \left(x_2-\frac{1}{2}\right)^2\right), \tag{2.132}$$

such that $0 \leq \Psi \leq \frac{1}{8}$ and $(\Delta_{h_\mathbf{l}})^{(i_1,\dots,i_m)}\Psi = 1$, because the finite difference scheme is exact for quadratic functions. Thus:

$$(\Delta_{h_\mathbf{l}})^{(i_1,\dots,i_m)}(f + \|w\|_\infty\Psi) > 0, \tag{2.133}$$

and a maximum principle gives a bound for the function:

$$\begin{aligned} f \leq \|f + \|w\|_\infty \Psi\|_\infty &\leq \|f_{|\partial\Omega} + \|w\|_\infty \Psi_{|\partial\Omega}\|_\infty \\ &\leq \|f_{|\partial\Omega}\|_\infty + \frac{1}{8}\|w\|_\infty \\ &\leq \frac{1}{8}\|w\|_\infty. \end{aligned} \tag{2.134}$$

because $f_{|\partial\Omega} = 0$. The same argument with $-f$ gives us the result. □

*Proof of theorem 2.2.7.* The proof is an outcome of theorem 2.1.3, we therefore need that the local error $(\hat{\mathbf{E}}^c_{h_n} - \mathbf{E})$ verify the assumption stated by equation (2.14). Unlike the problem discretized on the Cartesian grid (see proof of proposition 2.2.6), the resolution of the Poisson equation on the component grids requires more work because of the anisotropy of the grids, the operator of the problem depending on the discretizations in each direction, which are different. Thus, invertibility of the operator cannot provide directly an error expansion of the form of equations (2.14). This problem has been studied in [95, 22]. We therefore use the framework introduced in [95] for this proof.

First the local error can be recast into:

$$\mathcal{I}_{V_{h_\mathbf{l}}} \hat{\mathbf{E}}_{h_\mathbf{l}} - \mathbf{E} = \mathcal{I}_{V_{h_\mathbf{l}}} (\hat{\mathbf{E}}_{h_\mathbf{l}} - \mathbf{E}) + \mathcal{I}_{V_{h_\mathbf{l}}} \mathbf{E} - \mathbf{E} \tag{2.135}$$

In the following of this proof, we ommit the dependance upon the grid discretization in the coefficients of the form $b_{i_1,\dots,i_m}(\cdot; h_{l_{i_1}}, \dots, h_{l_{i_m}})$ for simplicity of notation. Applying lemma 2.2.23 for $m = 0$ and introducing the estimator of the density, one gets:

$$\Delta_{h_\mathbf{l}} \Phi + \hat{\rho}_{h_\mathbf{l}} = \sum_{i=1}^{2} (b_i + \tau_i) h_{l_i}^2 + b_{1,2} h_{l_1}^2 h_{l_2}^2 + \mathcal{V}(\hat{\rho}_{h_\mathbf{l}}). \tag{2.136}$$

Let us introduce the semi-discrete problems:

$$(\Delta_{h_\mathbf{l}})^{(i)} w_i = b_i + \tau_i, \quad w_i \in \Omega^{(i)}_{h_\mathbf{l}}, \ i = 1, 2. \tag{2.137}$$

The equation (2.136) can be recast into:

$$\begin{aligned} \Delta_{h_\mathbf{l}} \left( \Phi + \sum_{i=1}^{2} w_i h_{l_i}^2 \right) + \hat{\rho}_{h_\mathbf{l}} &= \sum_{i=1}^{2} \left( \Delta_{h_\mathbf{l}} - (\Delta_{h_\mathbf{l}})^{(i)} \right) w_i h_{l_i}^2 + b_{1,2} h_{l_1}^2 h_{l_2}^2 + \mathcal{V}(\hat{\rho}_{h_\mathbf{l}}) \\ &= (w_{1,2} + b_{1,2}) h_{l_1}^2 h_{l_2}^2 + \mathcal{V}(\hat{\rho}_{h_\mathbf{l}}), \end{aligned}$$

where the lemma 2.2.23 has been used on the right hand side and owing to the lemmata 2.2.23, 2.2.24, it holds for $i = 1, 2$:

$$\|w_i\|_\infty \leq \frac{1}{8}(\|b_i\|_\infty + \|\tau_i\|_\infty), \quad \|w_{1,2}\|_\infty \leq \frac{1}{12}(\|\partial_1^4 w_2\|_\infty + \|\partial_2^4 w_1\|_\infty). \tag{2.138}$$

Eventually applying the operators $(\Delta_{h_\mathbf{l}})^{-1}$, $\boldsymbol{\nabla}_{h_\mathbf{l}}$ which commute, lemma 2.2.24 and a Taylor expansion:

$$\boldsymbol{\nabla}_{h_\mathbf{l}} (\Phi - \hat{\Phi}_{h_\mathbf{l}}) = \Phi - \hat{\Phi}_{h_\mathbf{l}} + \mathbf{s}_{1,2} h_{l_1}^2 h_{l_2}^2 \tag{2.139}$$

$$= \mathbf{z}_1 h_{l_1}^2 + \mathbf{z}_2 h_{l_2}^2 + \mathbf{z}_{1,2} h_{l_1}^2 h_{l_2}^2 + \hat{\mathcal{Z}}_{N,h_\mathbf{l}} \tag{2.140}$$

where:

$$\|\mathbf{z}_i\|_\infty \leq \frac{1}{96}(\|\nabla\partial_i^2\rho\|_\infty + \|\nabla\partial_i^4\Phi\|_\infty), \quad \|\mathcal{Z}_{N,h_1}\|_\infty \leq \frac{1}{8}\|\nabla\mathcal{V}(\hat{\rho}_{h_1})\|_\infty, \tag{2.141}$$

$$\|\mathbf{z}_{1,2}\|_\infty \leq \frac{1}{9216}(\|\nabla\partial_1^4\partial_2^2\rho\|_\infty + 2\|\nabla\partial_1^4\partial_2^4\Phi\|_\infty + \|\nabla\partial_1^2\partial_2^4\rho\|_\infty) + \|\mathbf{s}_{1,2}\|_\infty, \tag{2.142}$$

$$\|\mathbf{s}_{1,2}\|_\infty \leq \frac{1}{288}\max\left(\|\partial_1^3\partial_2^2\rho\|_\infty + \|\partial_1^3\partial_2^4\Phi\|_\infty, \|\partial_1^2\partial_2^3\rho\|_\infty + \|\partial_1^4\partial_2^3\Phi\|_\infty\right). \tag{2.143}$$

On the other side, a Taylor expansion gives us the truncation error of the discrete gradient:

$$\nabla\Phi - \nabla_{h_1}\Phi = \begin{pmatrix} s_1 h_{l_1}^2 \\ s_2 h_{l_2}^2 \end{pmatrix}, \quad \|s_i\|_\infty \leq \frac{1}{3}\|\partial_i^3\Phi\|_\infty, \ i = 1, 2. \tag{2.144}$$

Lemma 2.2.22, applied first for $m = 1$ and $m = 0$, gives us:

$$\mathcal{I}_{V_{h_1}}(\hat{\mathbf{E}}_{h_1} - \mathbf{E}) = \hat{\mathbf{E}}_{h_1} - \mathbf{E} + \mathbf{f}_{1,2}h_{l_1}^2 h_{l_2}^2 \tag{2.145}$$

$$\mathcal{I}_{V_{h_1}}\mathbf{E} - \mathbf{E} = \mathbf{d}_1 h_{l_1}^2 + \mathbf{d}_2 h_{l_2}^2 + \mathbf{d}_{1,2}h_{l_1}^2 h_{l_2}^2, \tag{2.146}$$

where:

$$\|(f_{1,2})_j\|_\infty \leq \frac{4}{27}\left(\|\partial_1^2 z_2\|_\infty + \|\partial_2^2 z_1\|_\infty + \|\partial_k^2 s_j\|_\infty\right), \ k \neq j, \tag{2.147}$$

$$\|\mathbf{d}_i\|_\infty \leq \frac{4}{27}\left(\|\nabla\partial_i^2\Phi\|_\infty\right), \ i = 1, 2, \quad \|\mathbf{d}_{1,2}\|_\infty \leq \frac{16}{729}\left(\|\nabla\partial_1^2\partial_2^2\Phi\|_\infty\right). \tag{2.148}$$

Finally, summing the equations (2.139) and (5.4), applying to equation (2.145) and summing to equation (2.146), the local error verifies the assumption of equation (2.14):

$$\mathcal{I}_{V_{h_1}}\hat{\mathbf{E}}_{h_1} - \mathbf{E} = \mathbf{e}_1 h_{l_1}^2 + \mathbf{e}_2 h_{l_2}^2 + \mathbf{e}_{1,2}h_{l_1}^2 h_{l_2}^2 + \hat{\mathcal{Z}}_{h_1}, \tag{2.149}$$

where:

$$\|\mathbf{e}_1\|_\infty \leq \|\mathbf{d}_1\|_\infty + \|\mathbf{z}_1\|_\infty + \max(\|s_1\|_\infty, \|s_2\|_\infty) \tag{2.150}$$

$$\|\mathbf{e}_2\|_\infty \leq \|\mathbf{d}_2\|_\infty + \|\mathbf{z}_2\|_\infty + \max(\|s_1\|_\infty, \|s_2\|_\infty), \tag{2.151}$$

$$\|\mathbf{e}_{1,2}\|_\infty \leq \|\mathbf{d}_{1,2}\|_\infty + \|\mathbf{f}_{1,2}\|_\infty + \|\mathbf{z}_{1,2}\|_\infty. \tag{2.152}$$

Finally applying theorem 2.1.3 and a argument similar to the proof of proposition 2.2.5 for the particle sampling error, we get the result. □

*Proof of proposition 2.2.8.* The proof is given in two dimensions. The total momentum of the system is defined as

$$\mathcal{P} = m\iint_{\Omega\times\mathbb{R}^d}\mathbf{v}\sum_{p=1}^N\frac{n}{N}\delta(\mathbf{x} - \mathbf{x}_p(t))\delta(\mathbf{v} - \mathbf{v}_p(t))d\mathbf{x}d\mathbf{v} = \sum_{p=1}^N m\frac{n}{N}\mathbf{v}_p(t), \tag{2.153}$$

so

$$\frac{d\mathcal{P}}{dt} = \sum_{p=1}^N m\frac{n}{N}\frac{d\mathbf{v}_p(t)}{dt} = \sum_{p=1}^N q\frac{n}{N}\hat{\mathbf{E}}_{h_n}^c(\mathbf{x}_p(t)), \tag{2.154}$$

where $\hat{\mathbf{E}}^c_{h_n}$ is the sparse grid reconstruction of the electric field evaluated at particle positions, given by:

$$\hat{\mathbf{E}}^c_{h_n}(\mathbf{x}_p(t)) = \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \hat{\mathbf{E}}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}) \underbrace{\mathcal{S}_{h_{\mathbf{l}}}(\mathbf{x}_p(t) - \mathbf{j}h_{\mathbf{l}}) h_{l_1} h_{l_2}}_{= W_{h_{\mathbf{l}};\mathbf{j}}(\mathbf{x}_p(t))}, \tag{2.155}$$

Exchanging the sums, we get:

$$\frac{d\mathcal{P}}{dt} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{l_1} h_{l_2} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \hat{\mathbf{E}}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}) \underbrace{\frac{Q}{N} \sum_{p=1}^{N} \mathcal{S}_{h_{\mathbf{l}}}(\mathbf{x}_p(t) - \mathbf{j}h_{\mathbf{l}})}_{= \hat{\rho}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}})}, \tag{2.156}$$

Let us consider a component grid of level $\mathbf{l} = (l_1, l_2) \in \mathbb{L}_n$ and the notation $\hat{\Phi}^{j_1,j_2} = \hat{\Phi}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}})$ for a node $(j_1, j_2) \in I_{h_{\mathbf{l}}}$. Owing to periodic conditions on the field and the density:

$$\begin{aligned}
&\sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \hat{\mathbf{E}}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}) \hat{\rho}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}) = \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \boldsymbol{\nabla}_{h_{\mathbf{l}}} \hat{\Phi}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}) \Delta_{h_{\mathbf{l}}} \hat{\Phi}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}) \\
&= \sum_{\substack{(j_1,j_2)\in \\ [\![0,2^{l_1}]\!]\times[\![0,2^{l_2}]\!]}} \left( \frac{2\hat{\Phi}^{j_1,j_2}\hat{\Phi}^{j_1+1,j_2} - (\hat{\Phi}^{j_1+1,j_2})^2 + (\hat{\Phi}^{j_1-1,j_2})^2 - 2\hat{\Phi}^{j_1,j_2}\hat{\Phi}^{j_1-1,j_2}}{2h_{l_1}^3} \right) \\
&+ \left( \frac{2\hat{\Phi}^{j_1,j_2}\hat{\Phi}^{j_1,j_2+1} - (\hat{\Phi}^{j_1,j_2+1})^2 + (\hat{\Phi}^{j_1,j_2-1})^2 - 2\hat{\Phi}^{j_1,j_2}\hat{\Phi}^{j_1,j_2-1}}{2h_{l_2}^3} \right).
\end{aligned}$$

A change of index together with the periodicity of the problem yields:

$$\sum_{j_1=0}^{2^{l_1}} 2\hat{\Phi}^{j_1,j_2}\hat{\Phi}^{j_1+1,j_2} = \sum_{j_1=0}^{2^{l_1}} 2\hat{\Phi}^{j_1,j_2}\hat{\Phi}^{j_1-1,j_2}, \quad \sum_{j_1=0}^{2^{l_1}} (\hat{\Phi}^{j_1-1,j_2})^2 = \sum_{j_1=0}^{2^{l_1}} (\hat{\Phi}^{j_1+1,j_2})^2$$

Thus, owing to equivalent relations in $j_2$, we get the result. □

**Remark 2.2.25** *The proof on the conservation of the total momentum cannot be applied to the PIC-Hg and PIC-NSg schemes. Indeed, the source term of the Poisson equation does not appear in the field expression in equation (2.156).*

*Proof of proposition 2.2.10.* The proof is the same as the standard scheme and can be found in [18]. □

The proof of 2.2.15 is provided in dimension $d \in \mathbb{N}^*$. In order to prove the theorem, we introduce some some multi-index notations for $\mathbf{k}, \mathbf{l} \in \mathbb{N}^d$, $0 \leq r \leq s \leq d-1$:

$$\mathbf{k}^{r,s} = (k_r, ..., k_s) \in \mathbb{N}^{s-r+1}, \quad (\mathbf{k}, \mathbf{l})^{r,s} = (k_r, ..., k_s, l_r, ..., l_s) \in (\mathbb{N}^{s-r+1})^2, \tag{2.157}$$

and:

$$\mathbb{I}_n^r := [\![1,n]\!]^r \tag{2.158}$$
$$\mathbb{F}_n^r := \left\{(\mathbf{k},\mathbf{l}) \in \mathbb{I}_n^r \times \mathbb{I}_n^r \mid k_m = l_m,\ m \in [\![1,...,r]\!]\right\}, \tag{2.159}$$
$$\mathbb{G}_n^r := \left\{(\mathbf{k},\mathbf{l}) \in \mathbb{I}_n^r \times \mathbb{I}_n^r \mid k_m < l_m,\ m \in [\![1,...,r]\!]\right\}, \tag{2.160}$$
$$\mathbb{H}_n^r := \left\{(\mathbf{k},\mathbf{l}) \in \mathbb{I}_n^r \times \mathbb{I}_n^r \mid k_m > l_m,\ m \in [\![1,...,r]\!]\right\}. \tag{2.161}$$

From the definitions, it follows some equalities between the cardinal of the subspaces $|\mathbb{I}_n^r| = |\mathbb{F}_n^r| = O(n^r)$, $|\mathbb{G}_n^r| = |\mathbb{H}_n^r| = O\left(n^{2r}\right)$. The following lemmas have to be demonstrated. The first lemma is an extension of the lemma 2.2.21 to pairs of component grids with different levels $\mathbf{k},\mathbf{l} \in \mathbb{L}_n$.

**Lemma 2.2.26** *Let $\boldsymbol{k},\boldsymbol{l} \in \mathbb{L}_n$, then*

$$Q \sum_{\boldsymbol{i}\in I_{h_{\boldsymbol{k}}}} \sum_{\boldsymbol{j}\in I_{h_{\boldsymbol{l}}}} \mathbb{E}\left(S_{h_{\boldsymbol{k}}}(\boldsymbol{i}h_{\boldsymbol{k}} - \boldsymbol{X}) S_{h_{\boldsymbol{l}}}(\boldsymbol{j}h_{\boldsymbol{l}} - \boldsymbol{X})\right) W_{h_{\boldsymbol{k}};\boldsymbol{i}} W_{h_{\boldsymbol{l}};\boldsymbol{j}} \leq \|\rho\|_\infty \prod_{m=1}^{d} \frac{1}{h_{\kappa_m}} + O\left(h_{\lambda_m}\right) \tag{2.162}$$

*where $\boldsymbol{\lambda},\boldsymbol{\kappa} \in \mathbb{N}^d$ such that $\kappa_m = \min(k_m,l_m)$, $\lambda_m = \max(k_m,l_m)$, $m = 1,...,d$.*

**Lemma 2.2.27** *Let $\alpha \in \mathbb{N}^*$, $\beta \in [\![0,...,d-1]\!]$, $m > 0$ then*

$$\sum_{\substack{(k,l)\in\mathbb{I}_n\times\mathbb{I}_n,\\ k<l}} h_{l-k}(l-k)^\beta = n(-1)^\beta \left(\frac{2^{-x}}{1-2^{-x}}\right)^{(\beta)}\Bigg|_{x=1} + O(1), \tag{2.163}$$

$$\sum_{\substack{(k,l)\in\mathbb{I}_n\times\mathbb{I}_n,\\ k<l}} h_{l-k}^\alpha = \frac{n}{1-2^{-\alpha}} + O(1), \tag{2.164}$$

$$\sum_{\substack{(k,l)\in\mathbb{I}_n\times\mathbb{I}_n,\\ l\leq k-m}} h_{k-l} = 2nh_m + O(1). \tag{2.165}$$

*Proof of lemma 2.2.26.* Let us consider the one-dimensional case which can be extended to the d-dimensional case by tensor product of the shape function. Let us also assume that $l \geq k$ without loss of generality, then owing to the symmetry of the basis functions, it holds:

$$\begin{aligned}
&Q\mathbb{E}(S_{h_k}(ih_k - X)S_{h_l}(jhl - X)) \\
&\qquad = Q \iint_{\Omega_x\times\Omega_v} \frac{1}{h_k h_l} W\left(\frac{\xi - ih_k}{h_k}\right) W\left(\frac{\xi - jh_l}{h_l}\right) f(\xi,v,t)d\xi dv \\
&\qquad = Q \iint_{\Omega_x\times\Omega_v} \frac{1}{h_k} W\left(xh_{l-k} + jh_{l-k} - i\right) W(x)\, f(jh_l + xh_l, v, t)dxdv
\end{aligned}$$

where the change of variable $x = h_l^{-1}(\xi - jh_l)$ has been applied. There are three cases to consider either if $jh_{l-k} - i = 0$ or $jh_{l-k} - i < 0$ or $jh_{l-k} - i > 0$.
i) If $jh_{l-k} - i = 0$: with a Taylor expansion upon the probability density and because of the symmetry of the basis functions, all $x$-terms with odd exponent vanish, resulting in the $0^{th}$ order

term and negligible $2^{nd}$ order terms:

$$\begin{aligned}
Q\mathbb{E}(\mathcal{S}_{h_k}(ih_k - X)\mathcal{S}_{h_l}(jhl - X)) &= \frac{\rho(jh_l)}{h_k}\int_{-\frac{1}{h_l}}^{\frac{1}{h_l}} W(x)W(xh_{l-k})dx + O\left(h_l^2\right) \\
&= \frac{2\rho(jh_l)}{h_k}\int_{-1}^{0}(1+x)(1+xh_{l-k})dx + O\left(h_l^2\right) \\
&= \frac{\rho(jh_l)}{h_k}\left(1 - \frac{1}{3}h_{l-k}\right) + O\left(h_l^2\right) \\
&\leq \frac{\|\rho\|_\infty}{h_k} + O(h_l)
\end{aligned}$$

ii) If $jh_{l-k} - i > 0$: again with a Taylor expansion upon the probability density, one gets:

$$Q\mathbb{E}(\mathcal{S}_{h_k}(ih_k - X)\mathcal{S}_{h_l}(jhl - X)) = \frac{\rho(jh_l)}{h_k}\left(\int_{-\frac{1}{h_l}}^{0}\Psi(x)dx + \int_{0}^{\frac{1}{h_l}}\Psi(x)dx\right) + O(h_l)\,,$$

where:

$$\Psi(x) := W(x)W(xh_{l-k} + jh_{l-k} - i).$$

The $x$-terms with odd exponent no longer vanish because $\Psi$ is not symmetrical, so that the negligible terms scale as $O(h_l)$. Because $jh_{l-k} - i > 0$, we have that $xh_{l-k} + jh_{l-k} - i > 0$ for all $x \in \mathrm{Supp}(\Psi) = [-1, 1]$ and thus:

$$\begin{aligned}
\int_{-\frac{1}{h_l}}^{0}\Psi(x)dx &= \frac{1}{2}\left(1 - jh_{l-k} + i + \frac{1}{3}h_{l-k}\right) \\
\int_{0}^{\frac{1}{h_l}}\Psi(x)dx &= \begin{cases} 0 & \text{if } jh_{l-k} - i = 1, \\ \frac{1}{2}\left(1 - jh_{l-k} + i - \frac{1}{3}h_{l-k}\right) & \text{else.} \end{cases}
\end{aligned}$$

With $jh_{l-k} - i < 1$, the following bound holds true:

$$Q\mathbb{E}(\mathcal{S}_{h_k}(ih_k - X)\mathcal{S}_{h_l}(jhl - X)) \leq \frac{\|\rho\|_\infty}{h_k} + O(h_l)$$

iii) If $jh_{l-k} - i < 0$: by symmetry it is equivalent to $jh_{l-k} - i > 0$.

Eventually, using the partition of unit property of the basis function:

$$\sum_{i\in I_{h_k}}\sum_{j\in I_{h_l}} Q\mathbb{E}(\mathcal{S}_{h_k}(ih_k - X)\mathcal{S}_{h_l}(jhl - X))W_{h_k;i}W_{h_l;j} \leq \frac{\|\rho\|_\infty}{h_k} + O(h_l)\,. \;\square$$

*Proof of lemma 2.2.27.* Let $\beta \in [\![0, ..., d-1]\!]$, then:

$$\begin{aligned}
\sum_{\substack{(k,l)\in\mathbb{I}_n\times\mathbb{I}_n,\\ k<l}} h_{l-k}(l-k)^\beta &= \sum_{l=1}^{n}\sum_{k=1}^{l-1} k^\beta 2^{-k} = \sum_{l=1}^{n}(-1)^\beta \left(\frac{2^{-x}}{1-2^{-x}}\left(1-2^{-(l-1)x}\right)\right)^{(\beta)}\Big|_{x=1} \\
&= \sum_{l=1}^{n}(-1)^\beta \left(\sum_{m=0}^{\beta}\binom{\beta}{m}\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(\beta-m)}\left(1-2^{-(l-1)x}\right)^{(m)}\right)\Big|_{x=1} \\
&= \sum_{l=1}^{n}(-1)^\beta \left(\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(\beta)}\left(1-2^{-(l-1)x}\right)\right. \\
&\left. + \sum_{m=1}^{\beta}\binom{\beta}{m}\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(\beta-m)}(-1)^{m+1}(l-1)^m 2^{-(l-1)x}\right)\Big|_{x=1} \\
&= \sum_{l=1}^{n}\left((-1)^\beta\left(\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(\beta)}\right)\Big|_{x=1} + O\left(2^{-l}\right)\right) \\
&= n(-1)^\beta\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(\beta)}\Big|_{x=1} + O(1)\,.
\end{aligned}$$

Let $\alpha \in [\![1, ..., d-1]\!]$, then:

$$\begin{aligned}
\sum_{\substack{(k,l)\in\mathbb{I}_n\times\mathbb{I}_n,\\ k<l}} h^\alpha_{l-k} &= \sum_{l=1}^{n} 2^{-\alpha l}\sum_{k=1}^{l-1} 2^{\alpha k} = \sum_{l=1}^{n} 2^{-\alpha l}\frac{2^\alpha}{2^\alpha-1}(2^{\alpha(l-1)}-1) \\
&= \frac{n}{2^\alpha-1} - \frac{1-2^{\alpha n}}{(2^\alpha-1)(1-2^{-\alpha})}.
\end{aligned}$$

Let $m > 0$, then:

$$\sum_{\substack{(k,l)\in\mathbb{I}_n\times\mathbb{I}_n,\\ l\le k-m}} h_{k-l} = \sum_{k=1}^{n} 2^{-k}\sum_{l=1}^{k-m} 2^l = 2nh_m - (1-2^{-n}).\ \square$$

*Proof of theorem 2.2.15.* The following bound for the variance of the reconstructed charge density holds true:

$$\mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right) \le \sum_{(\mathbf{k},\mathbf{l})\in(\mathbb{L}_n)^2} |c_\mathbf{k}||c_\mathbf{l}|\left|\mathrm{Cov}(\mathcal{I}_{V_{h_\mathbf{k}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{k}}), \mathcal{I}_{V_{h_\mathbf{l}}}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}}))\right|. \tag{2.166}$$

First, let us have a look at the covariance of the pairs of component grids:

$$\begin{aligned}
&\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}}}),\mathcal{I}_{V_{h_{\mathbf{l}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}}})) = \mathbb{E}(\mathcal{I}_{V_{h_{\mathbf{k}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}}})\mathcal{I}_{V_{h_{\mathbf{l}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}}}))\\
&= \sum_{\mathbf{i}\in I_{V_{h_{\mathbf{k}}}}}\sum_{\mathbf{j}\in I_{V_{h_{\mathbf{l}}}}}\left(\frac{Q^2}{N^2}\sum_{p,q}\mathbb{E}\left(\mathcal{S}_{h_{\mathbf{k}}}(\mathbf{i}h_{\mathbf{k}}-\mathbf{X}_p)\mathcal{S}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}-\mathbf{X}_q)\right)-\mathbb{E}\left(\mathcal{S}_{h_{\mathbf{k}}}(\mathbf{i}h_{\mathbf{k}}-\mathbf{X}_p)\right)\mathbb{E}\left(\mathcal{S}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}-\mathbf{X}_q)\right)\right)W_{h_{\mathbf{k}};\mathbf{i}}W_{h_{\mathbf{l}};\mathbf{j}}\\
&= \sum_{\mathbf{i}\in I_{V_{h_{\mathbf{k}}}}}\sum_{\mathbf{j}\in I_{V_{h_{\mathbf{l}}}}}\left(\frac{Q^2}{N}\mathbb{E}\left(\mathcal{S}_{h_{\mathbf{k}}}(\mathbf{i}h_{\mathbf{k}}-\mathbf{X})\mathcal{S}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}-\mathbf{X})\right)+\frac{Q^2}{N^2}\sum_{p\neq q}\mathbb{E}(S_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}-\mathbf{X}_p))\mathbb{E}(S_{h_{\mathbf{k}}}(\mathbf{i}h_{\mathbf{k}}-\mathbf{X}_q))\right.\\
&\left.-\mathbb{E}\left(\mathcal{S}_{h_{\mathbf{k}}}(\mathbf{i}h_{\mathbf{k}}-\mathbf{X})\right)\mathbb{E}\left(\mathcal{S}_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}-\mathbf{X})\right)\right)W_{h_{\mathbf{k}};\mathbf{i}}W_{h_{\mathbf{l}};\mathbf{j}}
\end{aligned}$$

Since the extra-diagonal sum upon the particles (corresponding to the terms $p \neq q$) contains $N(N-1)$ terms which are independant because of the independance of the positions of particles, a part of it cancels with the last term and a negligible term remains left:

$$\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}}}),\mathcal{I}_{V_{h_{\mathbf{l}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}}})) = \frac{Q^2}{N}\sum_{\mathbf{i}\in I_{V_{h_{\mathbf{k}}}}}\sum_{\mathbf{j}\in I_{V_{h_{\mathbf{l}}}}}\mathbb{E}(S_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}-\mathbf{X})S_{h_{\mathbf{k}}}(\mathbf{i}h_{\mathbf{k}}-\mathbf{X}))W_{h_{\mathbf{k}};\mathbf{i}}W_{h_{\mathbf{l}};\mathbf{j}}+O\left(\frac{1}{N}\right).$$

On the other hand, according to lemma 2.2.26 and because $\mathrm{Card}(L_0) \geq \mathrm{Card}(L_i)$ for $0 \leq i \leq d-1$, the sum of the component grid pairs of covariance is bound by the sum on the more refined grids:

$$\begin{aligned}
\mathbb{V}\left(\mathcal{V}(\hat{\rho}^c_{h_n})\right) &\leq d^2 \sum_{\substack{(\mathbf{k},\mathbf{l})\in(\mathbb{N}^*)^{2d},\\ |\mathbf{k}|_1=n+d-1,\\ |\mathbf{l}|_1=n+d-1,}} |c_{\mathbf{k}}||c_{\mathbf{l}}|\left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}}}),\mathcal{I}_{V_{h_{\mathbf{l}}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}}}))\right|\\
&\leq d^2 \max_{\mathbf{l}\in\mathbb{L}_n}|c_{\mathbf{l}}|^2 \sum_{(\mathbf{k},\mathbf{l})\in\mathbb{I}_n^{d-1}\times\mathbb{I}_n^{d-1}}\left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}),\mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right|,
\end{aligned}$$

where:

$$\mathbf{k}^\star := (\mathbf{k}, n+d-1-|\mathbf{k}|_1^{1,d-1}),\ \ \mathbf{l}^\star := (\mathbf{l}, n+d-1-|\mathbf{l}|_1^{1,d-1}). \tag{2.167}$$

The conditions on the $l^1$-norms of the grid levels reduce the degrees of freedom from $2d$ to $2(d-1)$. Let us now consider all different configurations of component grid pairs, *i.e.* if $k_m = l_m$ or $k_m < l_m$ or $k_m > l_m$, for $m = 1, ..., d-1$. We consider the following decomposition of the index set:

$$\mathbb{I}_n^{d-1}\times\mathbb{I}_n^{d-1} = \bigcup_{0\leq r\leq s\leq d-1}\mathbb{F}_n^r\times\mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}, \tag{2.168}$$

so that the sum can be recast into:

$$\sum_{(\mathbf{k},\mathbf{l})\in\mathbb{I}_n^{d-1}\times\mathbb{I}_n^{d-1}} \left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}), \mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right|$$
$$= \sum_{0\le r\le s\le d-1} \sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}} \left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}), \mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right|.$$

Let $0 \le r \le s \le d-1$ be integers, and let us consider the innermost sum of the last expression. The $r$ first components are such that $k_m = l_m$, $m = 1, ..., r$. Then, the $s-r$ following are such that $k_m < l_m$, $m = r+1, ..., s$ and finally the last $d-1-s$ components are such that $k_m > l_m$, $m = s+1, ..., d-1$. Two cases are to distinguish: either if $k_d^\star > l_d^\star$ or not.

i) if $k_d^\star > l_d^\star$: then $|\mathbf{k}|_1^{1,d-1} < |\mathbf{l}|_1^{1,d-1}$, $s > r$ and according to equation (2.162) of lemma 2.2.26, one gets the following bound:

$$\sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}} \left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}), \mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right|$$
$$\le \|\rho\|_\infty \sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}} \frac{1}{h_{k_1}...h_{k_r}} \frac{1}{h_{k_{r+1}}...h_{k_s}} \frac{1}{h_{l_{s+1}}...h_{l_{d-1}}} \frac{1}{h_{l_d^\star}} + O\left(h_{n+1-d} h_{|\mathbf{l}-\mathbf{k}|_1^{1,s}}\right)$$
$$\le \frac{\|\rho\|_\infty}{h_{n+d-1}} \sum_{\substack{(\mathbf{k},\mathbf{l})^{1,s}\\ \in\mathbb{F}_n^r\times\mathbb{G}_n^{s-r}}} h_{|\mathbf{l}-\mathbf{k}|_1^{r+1,s}} \sum_{\substack{(\mathbf{k},\mathbf{l})^{s+1,d-1}\in\mathbb{H}_n^{d-1-s},\\ |\mathbf{k}|_1^{1,d-1}\le|\mathbf{l}|_1^{1,d-1}}} 1,$$

where the negligible terms have been omitted in the last expression and will also be in the following for clarity. In addition, for $\mathbf{k}, \mathbf{l} \in \mathbb{N}^{d-1}$ such that $k_m = l_m$, $m = 1, ..., r$ and $k_m < l_m$, $m = r+1, ..., s$, the following bound is verified:

$$\sum_{\substack{(\mathbf{k},\mathbf{l})^{s+1,d-1}\in\mathbb{H}_n^{d-1-s},\\ |\mathbf{k}|_1^{1,d-1}\le|\mathbf{l}|_1^{1,d-1}}} 1 \quad\le\quad \sum_{\substack{\mathbf{l}^{s+1,d-1}\in\mathbb{I}_n^{d-1-s}\\ (\mathbf{k}^{s+1,d-1})_i\in]\!](\mathbf{l}^{s+1,d-1})_i,(\mathbf{l}^{s+1,d-1})_i+\frac{1}{d-1-s}|\mathbf{l}-\mathbf{k}|_1^{r+1,s}]\!]}} 1 \quad=\quad \left(\frac{n}{d-1-s}|\mathbf{l}-\mathbf{k}|_1^{r+1,s}\right)^{d-1-s},$$

for $s \neq d-1$. If $s = d-1$, the inequality is trivially verified. Then, the previous expression can be written as:

$$\sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}} \left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}), \mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right|$$
$$\le \frac{\|\rho\|_\infty n^{d-1-s}}{h_{n+d-1}} \left(\frac{1}{d-1-s}\right)^{d-1-s} \sum_{\substack{(\mathbf{k},\mathbf{l})^{1,s}\\ \in\mathbb{F}_n^r\times\mathbb{G}_n^{s-r}}} h_{|\mathbf{l}-\mathbf{k}|_1^{r+1,s}} \left(|\mathbf{l}-\mathbf{k}|_1^{r+1,s}\right)^{d-1-s}.$$

Owing to binomial theorem,

$$\left(|\mathbf{l}-\mathbf{k}|_1^{r+1,s}\right)^{d-1-s} = \sum_{m=0}^{d-1-s}\binom{d-1-s}{m}(l_s-k_s)^m\left(|\mathbf{l}-\mathbf{k}|^{r+1,s-1}\right)^{d-1-s-m},$$

and from the equation (2.163) of lemma 2.2.27 applied with $\beta = m$, it holds:

$$\begin{aligned}
&\sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}}\left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}),\mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right| \\
&\leq \frac{\rho n^{d-1-s+1}}{h_{n+d-1}}\left(\frac{1}{d-1-s}\right)^{d-1-s} \\
&\sum_{\substack{(\mathbf{k},\mathbf{l})^{1,s}\\ \in\mathbb{F}_n^r\times\mathbb{G}_n^{s-r}}} h_{|\mathbf{l}-\mathbf{k}|_1^{r+1,s-1}}\sum_{m=0}^{d-1-s}\binom{d-1-s}{m}(-1)^m\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(m)}_{|x=1}\left(|\mathbf{l}-\mathbf{k}|^{r+1,s-1}\right)^{d-1-s-m} \\
&\underbrace{\leq \ldots \leq}_{\substack{\text{binomial theorem and}\\ \text{lemma 2.2.27}\\ \text{applied } s-r-1 \text{ times}}} D_{r,s}\|\rho\|_\infty\frac{n^{d-1-r}}{h_{n+d-1}}\sum_{(\mathbf{k},\mathbf{l})^{1,r}\in\mathbb{F}_n^r}1 \;=\; D_{r,s}\|\rho\|_\infty\frac{n^{d-1}}{h_{n+d-1}}.
\end{aligned}$$

The last equality is obtained with $\mathrm{Card}(\mathbb{F}_n^r) = n^r$. $D_{r,s}$ is a constant depending on the dimension $d$ and the integers $r, s$, resulting from the $s-r$ applications of the binomial theorem and lemma 2.2.27:

$$\begin{aligned}
D_{r,s} = &\left(\frac{1}{d-1-s}\right)^{d-1-s}\sum_{m_1=0}^{d-1-s}\binom{d-1-s}{m_1}(-1)^{m_1}\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(m_1)}_{|x=1} \\
&\sum_{m_2=0}^{d-1-s-m_1}\binom{d-1-s}{m_2}(-1)^{m_2}\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(m_2)}_{|x=1}\cdots\sum_{m_{s-r}=0}^{d-1-s-m_{s-r-1}}\binom{d-1-s}{m_{s-r}}(-1)^{m_{s-r}}\left(\frac{2^{-x}}{1-2^{-x}}\right)^{(m_{s-r})}_{|x=1}.
\end{aligned}$$

ii) if $k_d^\star \leq l_d^\star$: then $|\mathbf{l}|_1^{1,d-1} \leq |\mathbf{k}|_1^{1,d-1}$ and necessarily $s < d-1$, so that, according to equation (2.162) of lemma 2.2.26, one gets the following bound:

$$\begin{aligned}
&\sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}}\left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}),\mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right| \\
&\leq \|\rho\|_\infty\sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}}\frac{1}{h_{k_1}\ldots h_{k_r}}\frac{1}{h_{k_{r+1}}\ldots h_{k_s}}\frac{1}{h_{l_{s+1}}\ldots h_{l_{d-1}}}\frac{1}{h_{k_d^\star}} + O\left(h_{n+1-d}h_{|\mathbf{k}-\mathbf{l}|_1^{s+1,d-1}}\right) \\
&\leq \frac{\|\rho\|_\infty}{h_{n+d-1}}\sum_{\substack{(\mathbf{k},\mathbf{l})^{1,s}\\ \in\mathbb{F}_n^r\times\mathbb{G}_n^{s-r}}}\sum_{\substack{(\mathbf{k},\mathbf{l})^{s+1,d-1}\in\mathbb{H}_n^{d-1-s},\\ |\mathbf{l}|_1^{1,d-1}\leq|\mathbf{k}|_1^{1,d-1}}} h_{|\mathbf{k}-\mathbf{l}|_1^{s+1,d-1}}.
\end{aligned}$$

In addition, for $\mathbf{k},\mathbf{l}\in\mathbb{N}^{d-1}$ such that $k_m = l_m$, $m = 1,\ldots,r$ and $k_m < l_m$, $m = r+1,\ldots,s$, the

following bound is verified, according to lemma 2.2.27, equation (2.165):

$$\sum_{\substack{(\mathbf{k},\mathbf{l})^{s+1,d-1}\in\mathbb{H}_n^{d-1-s},\\ |\mathbf{l}|_1^{1,d-1}\leq|\mathbf{k}|_1^{1,d-1}}} h_{|\mathbf{k}-\mathbf{l}|_1^{s+1,d-1}} \leq \sum_{\substack{\mathbf{k}^{s+1,d-1}\in\mathbb{I}_n^{d-1-s}\\ (\mathbf{l}^{s+1,d-1})_i\in[\![1,(\mathbf{k}^{s+1,d-1})_i-\frac{1}{d-1-s}|\mathbf{l}-\mathbf{k}|_1^{r+1,s}]\!]}} h_{|\mathbf{k}-\mathbf{l}|_1^{s+1,d-1}} = \left(2nh_{\frac{|\mathbf{l}-\mathbf{k}|_1^{r+1,s}}{d-1-s}}\right)^{d-1-s}$$

$$= (2n)^{d-1-s} h_{|\mathbf{l}-\mathbf{k}|_1^{r+1,s}},$$

so that using the lemma 2.2.27, equation (2.164) and $\mathrm{Card}(\mathbb{F}_0^r) = n^r$ one gets the following bound:

$$\sum_{\substack{(\mathbf{k},\mathbf{l})\in\mathbb{F}_n^r\times\\ \mathbb{G}_n^{s-r}\times\mathbb{H}_n^{d-1-s}}} \left|\mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{k}^\star}}), \mathcal{I}_{V_{h_{\mathbf{l}^\star}}}\mathcal{V}(\hat{\rho}_{h_{\mathbf{l}^\star}}))\right|$$

$$\leq 2^{d-1-s}\|\rho\|_\infty \frac{n^{d-1-s}}{h_{n+d-1}} \sum_{\substack{(\mathbf{k},\mathbf{l})^{1,s}\\ \in\mathbb{F}_n^r\times\mathbb{G}_n^{s-r}}} h_{|\mathbf{l}-\mathbf{k}|_1^{r+1,s}}$$

$$\leq 2^{d-1-s}2^{s-r}\|\rho\|_\infty \frac{n^{d-1-r}}{h_{n+d-1}} \sum_{\substack{(\mathbf{k},\mathbf{l})^{1,r}\\ \in\mathbb{F}_n^r}} 1$$

$$\leq E_r\|\rho\|_\infty \frac{n^{d-1}}{h_{n+d-1}},$$

where $E_r$ is a constant that depend on the dimension $d$ and the integer $r$:

$$E_r = 2^{d-1-r}.$$

Finally, since there are $d(d+1)/2$ terms in the sum upon $0 \leq r \leq s \leq d-1$, the results is obtained by taking:

$$D = \frac{d^3(d+1)}{2\log(2)}\left(\max_{\mathbf{l}\in\mathbb{L}_n}|c_{\mathbf{l}}|^2\right)\max_{0\leq r\leq s\leq d-1}\max(D_{r,s}, E_r). \ \square$$

The proof of theorem 2.2.17 is inspired of the appendix in [104]. In order to prove the theorem 2.2.17, we need the following lemma.

**Lemma 2.2.28** *The discrete fourier transform of the electric potential and electric field particle sampling error follow the relations:*

$$\mathcal{F}_{\boldsymbol{m}}(\mathcal{V}(\hat{\Phi}_{h_l})) = \frac{1}{4\pi^2}\frac{\mathcal{F}_{\boldsymbol{m}}(\mathcal{V}(\hat{\rho}_{h_l})}{\tilde{m}_1^2 + \ldots + \tilde{m}_d^2 + O\left(h_{l_1}^2, \ldots, h_{l_d}^2\right)}, \tag{2.169}$$

$$\mathcal{F}_{\boldsymbol{m}}(\mathcal{V}(\hat{\boldsymbol{E}}_{h_l})) = \left(\left(-2\pi i\tilde{m}_r + O\left(h_{l_r}^2\right)\right)\mathcal{F}_{\boldsymbol{m}}(\mathcal{V}(\hat{\Phi}_{h_l}))\right)_{r=1,\ldots,d}, \tag{2.170}$$

*for* $\boldsymbol{m} \in I_{h_l}/\{\boldsymbol{0}\}$, $\tilde{m}_r = \min(m_r, 2^{l_r} - m_r)\mathrm{sign}(2^{l_r-1} - m_r)$ *and* $\mathcal{F}_{\boldsymbol{0}}(\mathcal{V}(\hat{\Phi}_{h_l})) = 0$.

**Lemma 2.2.29** *Let* $\boldsymbol{l} \in \mathbb{L}_n$, *and assume* $l_1 \geq l_i$, *for all* $i = 1, \ldots, d$, *then the following bounds*

*hold true:*

$$\sum_{\substack{\boldsymbol{m}\in I_{h_{l-1}}/\{\boldsymbol{1}\}\\ m_i\geq 1}} \frac{1}{\left(m_1^2+\ldots+m_d^2\right)^2} \leq \begin{cases} \frac{1}{3}\left(1-8h_{l_1}^3\right) & \text{if } d=1,\\ \frac{\pi}{2}\left(1-h_{l_1}^2\right) & \text{if } d=2,\\ \frac{\pi}{2}\left(1-\sqrt{2}h_{l_1}\right) & \text{if } d=3. \end{cases} \tag{2.171}$$

*Proof of lemma 2.2.29.* Because the function $\mathbf{x}\mapsto\|\mathbf{x}\|_2^{-4}$ is a decreasing function on $\mathbb{R}^d/\{\mathbf{0}\}$, the following bound resulting from a right rectangle integration rule holds true:

$$\sum_{\substack{\mathbf{m}\in I_{h_{\mathbf{l-1}}}/\{\mathbf{1}\}\\ m_i\geq 1}} \frac{1}{\left(m_1^2+\ldots+m_d^2\right)^2} \leq \sum_{\substack{\mathbf{m}\in[\![0,2^{l_1-1}-1]\!]\times\ldots\\ \times[\![0,2^{l_d-1}-1]\!]\times/\{\mathbf{0}\}}} \int_{m_1}^{m_1+1}\ldots\int_{m_d}^{m_d+1} \frac{dx_1....dx_d}{\left(x_1^2+\ldots x_d^2\right)^2} = \int_{\substack{[0,2^{l_1-1}]\times\ldots\times\\ [0,2^{l_d-1}]/[0,1]^d}} \frac{dx_1\ldots dx_d}{\left(x_1^2+\ldots+x_d^2\right)^2}.$$

The integral can be bounded by an integral in polar coordinates in two dimensions and spherical coordinates in three dimensions:

$$\int_{\substack{[0,2^{l_1-1}]\times\ldots\times\\ [0,2^{l_d-1}]/[0,1]^d}} \frac{dx_1\ldots dx_d}{\left(x_1^2+\ldots+x_d^2\right)^2} \leq \begin{cases} \int_0^{\frac{\pi}{2}}\int_1^{r^\star} \frac{1}{r^3}drd\theta = \frac{\pi}{2}\left(1-h_{l_1}^2\right) & \text{if } d=2,\\ \int_0^{\frac{\pi}{2}}\int_0^{\frac{\pi}{2}}\int_1^{r^\star} \frac{1}{r^2}\sin(\varphi)drd\varphi\theta = \frac{\pi}{2}\left(1-\sqrt{2}h_{l_1}\right) & \text{if } d=3, \end{cases}$$

where $r^\star=\sqrt{2}h_{l_1-1}^{-1}$. □

*Proof of lemma 2.2.28.* The proof is provided in one dimension for ease of notation; the extension to multiple dimensions follows the same arguments. Because the domain is periodic $\Omega_x=(\mathbb{R}/\mathbb{Z})$, we assume that the mean of $\mathcal{V}(\hat{\Phi}_{h_l})$ and $\mathcal{V}(\hat{\rho}_{h_l})$ is zero so that the Poisson problem is well posed. Therefore:

$$\mathcal{F}_0(\mathcal{V}(\hat{\Phi}_{h_l}))=\sum_{j\in I_{h_l}}\mathcal{V}(\hat{\Phi}_{h_l;j})=0,\quad \mathcal{F}_0(\mathcal{V}(\hat{\rho}_{h_l}))=\sum_{\mathbf{j}\in I_{h_l}}\mathcal{V}(\hat{\rho}_{h_l;j})=0.$$

Then, for $m\in I_{h_l}/\{0\}$, the discrete Fourier transform of $\mathcal{V}(\hat{\rho}_{h_l})$ is:

$$\mathcal{F}_m(\mathcal{V}(\hat{\rho}_{h_l}))=\sum_{j\in I_{h_l}}\mathcal{V}(\hat{\rho}_{h_l;j})e^{-2\pi imjh_l}.$$

From the finite difference discretization of the Poisson problem we have:

$$\mathcal{V}(\hat{\rho}_{h_l;j})=-\frac{\mathcal{V}(\hat{\Phi}_{h_l;j-1})-2\mathcal{V}(\hat{\Phi}_{h_l;j})+\mathcal{V}(\hat{\Phi}_{h_l;j+1})}{h_l^2}.$$

Substituting it into the discrete Fourier transform of $\mathcal{V}(\hat{\rho}_{h_l})$ expression and owing to periodicity,

one gets:

$$\begin{aligned}
\mathcal{F}_m(\mathcal{V}(\hat{\rho}_{h_l})) &= \sum_{j\in I_{h_l}} \mathcal{V}(\hat{\Phi}_{h_l;j}) \left( \frac{-e^{-2\pi i m(j-1)h_l} + 2e^{-2\pi i m j h_l} - e^{-2\pi i m(j+1)h_l}}{h_l^2} \right) \\
&= -\sum_{j\in I_{h_l}} \mathcal{V}(\hat{\Phi}_{h_l;j}) \left[ \frac{-e^{-2\pi i m(j-\frac{1}{2})h_l} \left( e^{2\pi i m h_l/2} - e^{-2\pi i m h_l/2} \right) - e^{-2\pi i m(j-\frac{1}{2})h_l} \left( e^{2\pi i m h_l/2} - e^{-2\pi i m h_l/2} \right)}{h_l^2} \right] \\
&= -i \frac{\sin(2\pi m h_l/2)}{h_l/2} \sum_{j\in I_{h_l}} \mathcal{V}(\hat{\Phi}_{h_l;j}) \left( \frac{e^{-2\pi i m(j-\frac{1}{2})h_l} - e^{-2\pi i m(j-\frac{1}{2})h_l}}{h_l} \right) \\
&= -i^2 \left( \frac{\sin(2\pi m h_l/2)}{h_l/2} \right)^2 \sum_{j\in I_{h_l}} \mathcal{V}(\hat{\Phi}_{h_l;j}) e^{-2\pi i m j h_l} \\
&= \left( 4\pi^2 \tilde{m}^2 - \frac{4}{3} \tilde{m}^4 h_l^2 \right) \mathcal{F}_m(\mathcal{V}(\hat{\Phi}_{h_l})),
\end{aligned}$$

where $\tilde{m} = \min(m, 2^l - m) \cdot \mathrm{sign}(2^{l-1} - m)$. The relation for the electric field is obtained with similar arguments. □

*Proof of theorem 2.2.17.* The following bound holds true for the variance of the electric potential particle sampling error:

$$\mathbb{V}(\mathcal{V}(\hat{\Phi}^c_{h_n})) \leq \sum_{(\mathbf{k},\mathbf{l})\in(\mathbb{L}_n)^2} |c_{\mathbf{k}}||c_{\mathbf{l}}| \left| \mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{k}}}), \mathcal{I}_{V_{h_{\mathbf{l}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}})) \right|.$$

$\mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}})$ is a centered random variable, so that from Cauchy-Schwarz inequality:

$$\begin{aligned}
\left| \mathrm{Cov}(\mathcal{I}_{V_{h_{\mathbf{k}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{k}}}), \mathcal{I}_{V_{h_{\mathbf{l}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}})) \right| &= \mathbb{E}\left( \mathcal{I}_{V_{h_{\mathbf{k}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{k}}}) \mathcal{I}_{V_{h_{\mathbf{l}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}}) \right) \\
&\leq \left[ \mathbb{V}\left( \mathcal{I}_{V_{h_{\mathbf{k}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{k}}}) \right) \mathbb{V}\left( \mathcal{I}_{V_{h_{\mathbf{l}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}}) \right) \right]^{\frac{1}{2}}.
\end{aligned}$$

In addition, for $\mathbf{l} \in \mathbb{L}_n$, by linearity of the expected value one gets:

$$\begin{aligned}
\mathbb{V}\left( \mathcal{I}_{V_{h_{\mathbf{l}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}}) \right) &= \mathbb{E}\left[ \left( \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}};\mathbf{j}}) W_{h_{\mathbf{l}};\mathbf{j}} \right)^2 \right] = \mathbb{E}\left( \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \sum_{\mathbf{i}\in I_{h_{\mathbf{l}}}} \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}};\mathbf{j}}) \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}};\mathbf{i}}) W_{h_{\mathbf{l}};\mathbf{j}} W_{h_{\mathbf{l}};\mathbf{i}} \right) \\
&= \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \sum_{\mathbf{i}\in I_{h_{\mathbf{l}}}} \mathbb{E}\left( \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}};\mathbf{j}}) \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}};\mathbf{i}}) \right) W_{h_{\mathbf{l}};\mathbf{j}} W_{h_{\mathbf{l}};\mathbf{i}} \\
&= \mathbb{V}\left( \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}}) \right) \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \sum_{\mathbf{i}\in I_{h_{\mathbf{l}}}} W_{h_{\mathbf{l}};\mathbf{j}} W_{h_{\mathbf{l}};\mathbf{i}} \\
&= \mathbb{V}\left( \mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}}) \right),
\end{aligned}$$

because $\mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}};\mathbf{j}})$ and $\mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}};\mathbf{i}})$ are centered random variables with the same variance, so that they follow the same law, *i.e.* the law of the random variable $\mathcal{V}(\hat{\Phi}_{h_{\mathbf{l}}})$. Let us find an upper

bound for the variance of this law. The DFT of $\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})$ is defined by:

$$\mathcal{F}_\mathbf{m}\left(\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})\right) = \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \mathcal{V}(\hat{\Phi}_{h_\mathbf{l};\mathbf{j}}) e^{-2\pi i \mathbf{m}\mathbf{j} h_\mathbf{l}}.$$

Then, the following equalities on the variance hold true:

$$\begin{aligned}
\mathbb{V}(\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})) &= h_{l_1}...h_{l_d} \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \mathbb{V}(\mathcal{V}(\hat{\Phi}_{h_\mathbf{l};\mathbf{j}})) = h_{l_1}...h_{l_d} \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \mathbb{E}\left(\mathcal{V}(\hat{\Phi}_{h_\mathbf{l};\mathbf{j}})^2\right) \\
&= \mathbb{E}\left( h_{l_1}...h_{l_d} \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \left|\mathcal{V}(\hat{\Phi}_{h_\mathbf{l};\mathbf{j}})\right|^2 \right) \\
&= \mathbb{E}\left( h_{l_1}^2...h_{l_d}^2 \sum_{\mathbf{m}\in I_{h_\mathbf{l}}} \left|\mathcal{F}_\mathbf{m}\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})\right|^2 \right) \\
&= h_{l_1}^2...h_{l_d}^2 \sum_{\mathbf{m}\in I_{h_\mathbf{l}}} \mathbb{V}\left(\mathcal{F}_\mathbf{m}\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})\right).
\end{aligned}$$

The first equality is obtained because the variance is constant on the domain, *i.e.* for all $\mathbf{j} \in I_{h_\mathbf{l}}$; the second and the third are obtained because the random variable is centered and by linearity of the expected value. The fourth equality results from Plancherel theorem and the last again because the random variable is centered and by linearity of the expected value. Using lemma 2.2.28 and omitting the negligible terms, one gets:

$$\mathbb{V}(\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})) = \frac{h_{l_1}^2...h_{l_d}^2}{16\pi^4} \sum_{\mathbf{m}\in I_{h_\mathbf{l}}} \frac{\mathbb{V}\left(\mathcal{F}_\mathbf{m}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}})\right)}{\left(\tilde{m}_1^2 + ... + \tilde{m}_d^2\right)^2},$$

where $\tilde{m}_i = \min(m_i, 2^{l_i} - m_i) \cdot \text{sign}(2^{l_i-1} - m_i)$, $i = 1, ..., d$. According to [96], for a random variable with variance $\sigma$, the variance of its DFT with $K$-point is $K\sigma$. Therefore the variance of the DFT of $\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})$ is:

$$\mathbb{V}\left(\mathcal{F}_\mathbf{m}\mathcal{V}(\hat{\rho}_{h_\mathbf{l}})\right) = \left(\frac{2}{3}\right)^d \frac{Q\|\rho\|_\infty}{N h_{l_1}^2...h_{l_d}^2} + O\left(\frac{1}{N h_{l_1}...h_{l_d}}\right).$$

Getting rid of the DFT notation $\tilde{m}$ and omitting negligible terms, the variance of the particle sampling error is then:

$$\mathbb{V}(\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})) = \left(\frac{4}{3}\right)^d \frac{1}{16\pi^4} \frac{Q\|\rho\|_\infty}{N} \sum_{\mathbf{m}\in I_{h_{\mathbf{l}-\mathbf{1}}}/\{\mathbf{0}\}} \frac{1}{\left(m_1^2 + ... + m_d^2\right)^2}.$$

Assuming $l_1 \geq l_i$, $\forall i = 1, ..., d$ without loss of generality and using lemma 2.2.29, the sum, for

$d = 2$, is recast into:

$$\sum_{\mathbf{m}\in I_{h_{\mathbf{l-1}}}/\{\mathbf{0}\}} \frac{1}{\left(m_1^2+m_2^2\right)^2} = \frac{1}{4} + \sum_{i=1}^{2}\sum_{m_i=1}^{2^{l_i-1}} \frac{1}{m_i^4} + \sum_{\substack{\mathbf{m}\in I_{h_{\mathbf{l-1}}}/\{\mathbf{1}\} \\ m_i\geq 1}} \frac{1}{\left(m_1^2+m_2^2\right)^2}$$
$$\leq \frac{\pi}{2} + \frac{35}{12} + O\left(h_{l_1}^2\right).$$

and for $d = 3$:

$$\sum_{\mathbf{m}\in I_{h_{\mathbf{l-1}}}/\{\mathbf{0}\}} \frac{1}{\left(m_1^2+m_2^2+m_3^3\right)^2} = \frac{1}{9} + \sum_{\substack{i,j=1 \\ i\neq j}}^{3} \sum_{\substack{m_i,m_j\in \\ [\![0,2^{l_i-1}]\!]\times \\ [\![0,2^{l_j-1}]\!]/\{0,0\}}} \frac{1}{\left(m_i^2+m_j^2\right)^2} + \sum_{\substack{\mathbf{m}\in I_{h_{\mathbf{l-1}}}/\{\mathbf{1}\} \\ m_i\geq 1}} \frac{1}{\left(m_1^2+m_2^2+m_3^2\right)^2}$$
$$\leq 2\pi + \frac{321}{36} + O\left(h_{l_1}\right).$$

Finally, bounding the last expressions by the one corresponding to component grids for which $l_i = l_j = \lfloor \frac{n}{d} \rfloor$, $i, j = 1, ..., d$, the following bounds hold true in two dimensions:

$$\mathbb{V}(\mathcal{V}(\hat{\Phi}_{h_n}^c)) \leq |\mathbb{L}_n|^2 \frac{Q\|\rho\|_\infty}{N} \frac{1}{9\pi^4} \left(\frac{\pi}{2} + \frac{35}{12} + O\left(h_{\lfloor \frac{n}{2} \rfloor}^2\right)\right)$$
$$\leq \frac{4}{9\pi^4 \log(2)^2} \left(\frac{\pi}{2} + \frac{35}{12} + O\left(h_{\lfloor \frac{n}{2} \rfloor}^2\right)\right) Q\|\rho\|_\infty \frac{|\log h_n|^2}{N} + O\left(\frac{|\log h_n|}{N}\right),$$

and in three dimensions:

$$\mathbb{V}(\mathcal{V}(\hat{\Phi}_{h_\mathbf{l}})) \leq |\mathbb{L}_n|^2 \frac{Q\|\rho\|_\infty}{N} \frac{4}{27} \frac{1}{\pi^4} \left(2\pi + \frac{321}{36} + O\left(h_{\lfloor \frac{n}{3} \rfloor}\right)\right)$$
$$\leq \frac{1}{3\pi^4 \log(2)^4} \left(2\pi + \frac{321}{36} + O\left(h_{\lfloor \frac{n}{3} \rfloor}\right)\right) Q\|\rho\|_\infty \frac{|\log h_n|^4}{N} + O\left(\frac{|\log h_n|^3}{N}\right).$$

□

# Chapter 3

# Optimization and shared memory parallelization

The present chapter is an extension of the chapter 2 to three dimensional geometries, with the aim to unravel the computational efficiency of sparse-PIC methods for sequential execution and parallel implementations on shared memory architectures. Those methods are characterized by an improved control of the statistical method, compared to the standard methods, allowing thus to significantly reduce the number of numerical particles used in simulations. The sparse-PIC schemes are particularly memory efficient, with respect to the size of the component grids used to accumulate the density and compute the electric field as well as the array used to store the particles properties. This limited memory footprint calls for the development of parallel implementations on shared memory architectures. Though these platforms offer a limited amount of memory compared to distributed architectures, this limitation is not an issue for sparse-PIC methods. Therefore implementations of these methods scalable on tens or hundreds (and tomorrow thousands) of cores open the way to 3d-3v simulations on simple and inexpensive platforms. Beyond this first objective, an efficient implementation on shared memory architectures provides a building block for porting these methods on modern distributed memory architectures, where each node is a shared memory system with tens of cores. The scope of the present chapter is therefore dedicated to CPU shared memory architectures, and the porting to GPU will be addressed in chapter 4.

Standard PIC methods are not well suited for scalable implementations onto shared memory architectures. This is mainly due to the fact that PIC methods are globally memory bound. Thus the multiplication of the cores without any significant increase of the memory bandwidth brings a poor speed-up. This issue is analyzed in [103] and received a lot of attention for years [9, 10, 12]. Different workarounds are proposed to favor the locality of the data, increasing the cache reuse, therefore mitigating the number of requests to the main memory. Strategies are also proposed for Non-Uniform Memory Access (NUMA) architecture platforms. The purpose there is to take advantage of pieces of memory, local to a subset of cores, to enhance the scalability of the algorithm. This is achieved thanks to a decomposition of the population of particles into samples, each of which being assigned to a NUMA domain and the related operations performed by the local subset of cores. The implementation of sparse-PIC methods on NUMA shared memory architectures has not been proposed so far. The existing developments are specific to distributed architectures and do not take full benefits of the sparse-PIC properties to improve both the sequential and parallel computational efficiency. In [26], the strategy proposed is tailored to distributed memory architectures and takes profit of the small sizes of the grids involved in sparse-PIC methods. The grids are replicated and independent samples of particles are also created on each node. The motivation is to use the memory local to each node and avoid the communications inherent to strategies based on domain decomposition.

The purpose of the present chapter is to propose a first efficient implementation of sparse-PIC

methods, combining memory management and parallelization strategies exploiting the genuine properties of sparse-PIC methods. The sparse grid reconstructions require the operations of each particles with numerous (tens) anisotropic grids, with coarse discretizations and different sparsity patterns. These grids are referred to as *component grids*. The set of nodes of all the component grids is designated by the *sparse grid* terminology. The sparsity of these grids reduces their memory footprint to few kilobytes (KB) which is small enough to be contained in the highest level of the cache hierarchy common to any modern CPU core. Furthermore, the interactions of the particles with two different grids are independent and can therefore be processed concurrently. This defines a new level of parallelism, specific to sparse-PIC methods, that is central in obtaining a good scalability. The issue there lies in the load balancing, the number of grids being not necessarily a multiple of the number of cores. A strategy is therefore proposed to preserve the scalability close to optimal and finally obtain speed-ups exceeding 100 on 128 cores.

On top of that a hierarchization of the data accumulated on the component grids is introduced. It amounts to decomposing the information onto a basis of hierarchical functions, with in the end, a very significant reduction of the complexity of the sparse-PIC algorithm.

## Contents



# 3.1 Combination in hierarchical basis

In the present section, an alternative representation of the sparse grid approximations based on the hierarchical decomposition of the basis functions is introduced. The motivation is to reduce the complexity of the combination operations. The approach presented here can be applied both to the PIC-Hg and the PIC-NSg schemes, yielding the same reconstruction than the original schemes. Nonetheless, the method cannot be applied to the PIC-Sg scheme because it relies on the full grid which is absent of the PIC-Sg scheme. In the following, the hierarchical basis combination is presented in the framework of the PIC-NSg.

Let us recall the main steps of the combination within the PIC-Nsg scheme. Let $\Phi_{h_\mathbf{l}}$, for a level index $\mathbf{l} \in \mathbb{L}_n$, be defined by the approximations of the electric potential computed on the component grid of level $\mathbf{l}$. The sparse grid reconstruction is defined by the combination of each contributions in nodal basis :

$$\Phi^c_{h_n} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_\mathbf{l} \mathcal{I}^{\mathcal{N}}_{V_{h_\mathbf{l}}} \Phi_{h_\mathbf{l}} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_\mathbf{l} \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \Phi_{h_\mathbf{l};\mathbf{j}} W_{h_\mathbf{l};\mathbf{j}}, \tag{3.1}$$

or equivalently, as stated by equation (2.7), in hierarchical basis:

$$\Phi^c_{h_n} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_\mathbf{l} \mathcal{I}^{\mathcal{H}}_{V_{h_\mathbf{l}}} \Phi_{h_\mathbf{l}} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_\mathbf{l} \sum_{\mathbf{k}\le\mathbf{l}} \sum_{\mathbf{i}\in\mathcal{B}_{h_\mathbf{l}}} \alpha_{\mathbf{k},\mathbf{i}} W_{h_\mathbf{k};\mathbf{i}}, \tag{3.2}$$

where $\alpha_{\mathbf{k},\mathbf{i}}$ are the hierarchical surplus.

The coefficients in the hierarchical basis $(\alpha_{\mathbf{k},\mathbf{i}})_{\mathbf{i}\in\mathcal{B}_{h_\mathbf{k}},\mathbf{k}\leq\mathbf{l}}$ of the function $\Phi_{h_\mathbf{l}} \in V_{h_\mathbf{l}}$ are determined by a transformation from the nodal basis (whose coefficients are plainly the values of $\Phi_{h_\mathbf{l}}$ on the nodes of the full grid) to the hierarchical basis and the transformation is called hierarchization. Hierarchization is done by applying a d-dimensional stencil constructed by tensor product of one-dimensional stencil [24]:

$$\mathcal{H}_{\mathbf{i}h_\mathbf{k},\mathbf{k}} = \bigotimes_{j=1}^{d} \mathcal{H}_{i_j h_{k_j},k_j}, \quad \mathcal{H}_{i_j h_{k_j},k_j} = \left[-\frac{1}{2} \;\; 1 \;\; -\frac{1}{2}\right]_{i_j h_{k_j},k_j}, \tag{3.3}$$

where the one-dimensional stencil stands for:

$$\mathcal{H}_{i_j h_{k_j},k_j} f = f(i_j h_{k_j}) - \frac{1}{2}\left[f\left(\frac{i_j-1}{2}h_{k_j-1}\right) + f\left(\frac{i_j+1}{2}h_{k_j-1}\right)\right]. \tag{3.4}$$

The hierarchical surpluses of a function are given by:

$$\alpha_{\mathbf{k},\mathbf{i}} := \mathcal{H}_{\mathbf{i}h_\mathbf{k},\mathbf{k}} f, \qquad \mathbf{i} \in \mathcal{B}_{h_\mathbf{k}},\ \mathbf{k} \in \mathbb{N}^d. \tag{3.5}$$

Let us introduce the transformation from the nodal basis to the hierarchical basis in matrix formulation. Let $\boldsymbol{\alpha}_\mathbf{l} = (\alpha_{\mathbf{k},\mathbf{i}})_{\mathbf{i}\in\mathcal{B}_{h_\mathbf{k}},\mathbf{k}\leq\mathbf{l}}$, $\boldsymbol{\Phi}_\mathbf{l} = (\Phi_{h_\mathbf{l}}(\mathbf{j}h_\mathbf{l}))_{\mathbf{i}\in I_{h_\mathbf{l}}} \in \bigotimes_{j=1}^{d} \mathbb{R}^{2^{l_j}+1}$ be vectors with coordinates arranged in ascending order according to their global position in each dimension, which are of the same size because of the definitions of $\mathcal{B}_{h_\mathbf{l}}$ and $I_{h_\mathbf{l}}$. The hierarchization can thus be written as follows:

$$\boldsymbol{\alpha}_\mathbf{l} = \boldsymbol{\mathcal{H}}_\mathbf{l}\boldsymbol{\Phi}_\mathbf{l}, \quad \boldsymbol{\mathcal{H}}_\mathbf{l} = \begin{pmatrix} \boldsymbol{\mathcal{H}}_{l_1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \boldsymbol{\mathcal{H}}_{l_d} \end{pmatrix}, \quad \boldsymbol{\mathcal{H}}_l = \boldsymbol{\mathcal{H}}_l^{(l-1)} \ldots \boldsymbol{\mathcal{H}}_l^{(1)} \tag{3.6}$$

where $\boldsymbol{\mathcal{H}}_l^{(k)} \in \mathcal{M}_{2^l+1}(\mathbb{R})$, $k \in \{1, \ldots, l-1\}$ is defined by:

$$(\boldsymbol{\mathcal{H}}_l^{(k)})_{i,j} = \begin{cases} 1 & \text{if } j = i \\ \frac{1}{2} & \text{if } j = i \pm 2^{k-1} \text{ and } i \in 2^k\mathbb{Z}/\{0, 2^l\} \\ 0 & \text{else} \end{cases}. \tag{3.7}$$

The inverse operation consisting in a transformation from the hierarchical basis to the nodal basis is named dehierarchization [67] and is also done by applying a d-dimensional stencil:

$$\mathcal{D}_{\mathbf{i}h_\mathbf{k},\mathbf{k}} = \bigotimes_{j=1}^{d} \mathcal{D}_{i_j h_{k_j},k_j}, \quad \mathcal{D}_{i_j h_{k_j},k_j} = \left[\frac{1}{2} \;\; 1 \;\; \frac{1}{2}\right]_{i_j h_{k_j},k_j}, \tag{3.8}$$

where the one-dimensional stencil is defined by:

$$\mathcal{D}_{i_j h_{k_j},k_j} f = f(i_j h_{k_j}) + \frac{1}{2}\left[f\left(\frac{i_j-1}{2}h_{k_j-1}\right) + f\left(\frac{i_j+1}{2}h_{k_j-1}\right)\right]. \tag{3.9}$$

**Proposition 3.1.1** *The sparse grid reconstruction defined in equation (3.2) can be expressed in*

*the following form:*

$$\Phi^c_{h_n} = \sum_{|\boldsymbol{k}|_\infty \leq n} \sum_{\boldsymbol{i} \in \mathcal{B}_{h_l}} \gamma_{\boldsymbol{k},\boldsymbol{i}} \varphi_{h_{\boldsymbol{k}},\boldsymbol{i}}, \tag{3.10}$$

*where:*

$$\gamma_{\boldsymbol{k},\boldsymbol{i}} = \sum_{\boldsymbol{l} \in \mathbb{L}_n} c_{\boldsymbol{l}} \beta_{\boldsymbol{k},\boldsymbol{i}}, \qquad \beta_{\boldsymbol{k},\boldsymbol{i}} := \begin{cases} \mathcal{H}_{\boldsymbol{i}h_{\boldsymbol{k}},\boldsymbol{k}} \Phi_{h_{\boldsymbol{k}}}, & \text{if } \boldsymbol{k} \leq \boldsymbol{l}, \\ 0 & \text{else.} \end{cases} \tag{3.11}$$

The result of proposition 3.1.1 allows us to conceive an algorithm for the combination of the electric potential based on the hierarchical basis representation, which is presented in algorithm 5.

**Algorithm 5** Combination in hierarchical basis (PIC-HSg)

**Require:** Approximation of the electric potential $\Phi_{h_{\mathbf{l}}}$ on the component grid.
**for each** component grid of index $\mathbf{l} \in \mathbb{L}_n$ **do**
**for each** node of the component grid $\mathbf{i} \in I_{h_{\mathbf{l}}}$ **do**
**Apply** the transformation into the hierarchical basis:

$$\alpha_{\mathbf{l},\mathbf{i}} \leftarrow \mathcal{H}_{\mathbf{i}h_{\mathbf{l}},\mathbf{l}} \Phi_{h_{\mathbf{l}}}. \tag{3.12}$$

**Determine** the full grid index node $\mathbf{j}$ corresponding to the index node $\mathbf{i}$.
**Add** the contribution $\gamma_{\mathbf{n},\mathbf{j}} := c_{\mathbf{l}} \alpha_{\mathbf{l},\mathbf{i}}$ to $\Omega^{(\infty)}_{h_n}$ at the node $\mathbf{j}h_n$.
**end for**
**end for**
**for each** node of the full grid $\mathbf{j} \in I_{h_n}$ **do**
**Apply** the transformation into the nodal basis:

$$\Phi^c_{h_n}(\mathbf{j}h_n) \leftarrow \mathcal{H}^{-1}_{\mathbf{j}h_{\mathbf{n}},\mathbf{n}} \gamma_{\mathbf{n},\mathbf{j}}. \tag{3.13}$$

**end for**

### 3.1.1 Unidirectional principle

The unidirectional principle is a way to perform hierarchization of a multiple dimension function by a series of one-dimensional hierarchizations and is detailed in [73, 74, 75]. We briefly recall the principle of the method in this section. The principle exploits the tensor structure of the basis functions; for the d-dimensional case, hierarchization is obtained by hierarchizing dimensions one after the other. The hierarchization with the unidirectional principle is presented in $d$ dimensions in algorithm 6. The outer loop iterates over the $d$ dimensions and constitutes the unidirectional principle. For a specific dimension $i$, the data is split into one-dimension poles upon which the operations are made. A pole in dimension $i$ consists of all points of the grid which only differ in the $i^{th}$ component, that is which lie on a line parallel to the $i^{th}$ coordinate axis. For each pole, the operation is solely the one-dimensional hierarchization introduced in equation (3.4).

**Algorithm 6** Hierarchization with unidirectional principle

**Require:** $\mathbf{l} = (l_1, ..., l_d) \geq 0$ level, nodal coefficients stored in array[:]
**Ensure:** hierarchical coefficients stored in array [:]
**for** $i$ **from** 1 **to** $d$ **do**
**for** $p = l_i$ **downto** 0 **do**
$\tilde{\mathbf{l}} \leftarrow (..., l_{i-1}, p, l_{i+1}, ...)$
**for all** nodes $\mathbf{x}$ of level $\tilde{\mathbf{l}}$ **do**
**Let** $\mathbf{x}_l$ be the left hierarchical ancestor of $\mathbf{x}$ in dimension $i$
**Let** $\mathbf{x}_r$ be the right hierarchical ancestor of $\mathbf{x}$ in dimension $i$
array[$\mathbf{x}$]$\leftarrow$ array[$\mathbf{x}$] $-\frac{1}{2}$(array[$\mathbf{x}_l$]+array[$\mathbf{x}_r$])
**end for**
**end for**
**end for**

### 3.1.2 Complexity of the different combinations

In the previous section, the sparse grid reconstruction of the solution from the component grid contributions has been equivalently exposed in nodal or hierarchical basis. The methods, though providing equivalent results, differ in the operations involved. The discussion provided in section 2.7.1 of [67] suggests that a combination in hierarchical basis is more efficient than a combination in nodal basis. In this section, the investigation of the complexity of both approaches is provided. The combination technique in the hierarchical basis follows four steps:

- *Hierarchization:* For each component grid $\Omega_{h_\mathbf{l}}$, $\mathbf{l} \in \mathbb{L}_n$, a transformation to the hierarchical basis is performed by applying $\mathcal{H}_\mathbf{l}$. According to the unidirectional principle, the hierarchization of each grid $\Omega_{h_\mathbf{l}}$ amounts to a number of operations $n_{op}(\mathbf{l})$[73]:

$$n_{op}(\mathbf{l}) = 2 \cdot \sum_{i=1}^{d} \left( (2^{l_i+1} - 2 \cdot l_i - 2) \cdot \prod_{\substack{m=1 \\ m \neq i}}^{d} (2^{l_m} - 1) \right), \tag{3.14}$$

  then, the complexity of the hierarchization is:

$$\sum_{\mathbf{l} \in \mathbb{L}_n} n_{op}(\mathbf{l}) = O\left(n^{d-1} \cdot 2^n\right). \tag{3.15}$$

- *Prolongation:* Since every component grid involved in the combination is included in the full grid, the hierarchical surplus of each component grid are prolonged onto the full grid. This step yields no computation operation.
- *Combination:* The hierarchical surplus from each component grid are combined onto the full grid. The number of operations $n_{op}(\mathbf{l})$ for each component grid $\Omega_{h_\mathbf{l}}$ is:

$$n_{op}(\mathbf{l}) = 2 \cdot \prod_{m=1}^{d} (2^{l_m} + 1),$$

  and the complexity is similar to equation (3.15).

- *Dehierarchization:* A transformation from the hierarchical basis to the nodal basis is performed on the full grid to recover the values of the sparse grid reconstruction. Applying equation (3.14) to the full grid, the number of operations for the dehierarchization on the full grid scales with $O\left(2^{nd}\right)$.

The complexities of the hierarchization and combination steps are negligible and thus the complexity of the method is dominated by the dehierarchization complexity, that is $O(2^{nd})$. On the other side, the combination in the nodal basis consists of the following steps:

- *Interpolation:* For each component grid, an interpolation of $\Phi_{h_\mathbf{l}}$ onto the space $V_{h_\mathbf{l}}$ is considered in nodal basis according to the relation (2.10). The number of operations $n_{op}$ for each component grid is:

$$n_{op}(\mathbf{l}) = (6d + d2^d)(2^n + 1)^d.$$

  Since there are $O\left(n^{d-1}\right)$* component grids in the combination, the number of operations for all the grids scales with $O\left(n^{d-1} \cdot 2^{nd}\right)$.
- *Combination:* The contribution from each component grid is added to the full grid. The number of operations $n_{op}(\mathbf{l})$ for each component grid $\Omega_{h_\mathbf{l}}$ is

$$n_{op}(\mathbf{l}) = 2 \cdot (2^n + 1)^d,$$

  and the complexity is similar to the interpolation.

From this investigation of complexity, it is manifest that the hierarchical representation leading to $O\left(2^{nd}\right)$ operations shall provide a more efficient method than the nodal representation with $O\left(n^{d-1} \cdot 2^{nd}\right)$ operations.

## 3.2 Optimization and parallelization for shared memory systems

### 3.2.1 A non exhaustive overview of optimizations and parallelizations of standard PIC methods on shared memory architectures

> *"The authors in [103] present a model able to predict execution time as a function of data transfers. This study shows that, to improve the performance of multi-core (intra-node) processing in PIC simulations, we must decrease the amount of costly memory accesses."*
>
> Y.A. Barsamian, A. Charguéraud, S.A. Hirstoaga, M. Mehrenberger, *Efficient strict-binning Particle-In-Cell algorithm for multi-core SIMD processors*, 2018.

In PIC simulations, the implementations are usually memory-bound rather than compute-bounded [9, 103]. The performance of the implementation, with respect to the computational time, is limited by the number of memory accesses. For these kind of applications, multiplying

*There are $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} + \frac{(n-1)(n-2)}{2}$ component grids for $d = 3$.

the number of cores involved in the computation increases the memory contention and in the end deteriorates the efficiency.

This feature of PIC methods lies in the interaction of the particles and the array used to accumulate their properties. A particle contributes to the values stored on a limited sub-set of array indices corresponding to the cell it is contained in. As the particle is randomly distributed onto the computational domain and free to move during the simulation, a naive implementation of PIC methods does not secure a contiguous access to both the array containing the particle properties and the (grid) array accumulating the density (see figure 4.1). This deteriorates the CPU cache management significantly, resulting in an important performance loss. Indeed, the charge density accumulation, sketched in algorithm 7, consists mainly in loading the coordinates of one particle, computing the grid cell it is contained in and, adding the contribution to the 8 nodes of this cell. Two consecutive particles (with regard to their indices in the particle coordinate array) may contribute to different cells in the density array. This entails either a contiguous memory access in the particle array or in the density (grid) array. A similar issue characterizes the interpolation of the forces from the grid onto the position of the particles. Though the density accumulation and the field interpolation are only two steps of the complete algorithm, they account for a significant part necessary to mitigate the statistical noise.

Different workarounds are proposed, mainly based on a so-called sorting [12] of the particles. To this end, each cell of the grid receives a rank, two cells contiguous in memory being associated to consecutive ranks. The particles properties (position, velocity, etc.) are then stored in the particle arrays by rank of the cell they are contained in. By this means, both the grid and the particle arrays may be accessed contiguously providing a better cache reuse for the density accumulation and the field interpolation. Nonetheless, the particles shall be periodically sorted, since during the computation they are likely to cross the cell boundaries. This is thus a trade-off between the cost of re-sorting the particle population and, increasing the cache-miss rate during iterations. Different elaborated data structures have been proposed to alleviate the cost of the periodic particle sorting (see [9, 106] and [10, 11]). Another approach to mitigate the randomness of the memory accesses is to consider domain decomposition [52, 100, 110] with subdomains so small that they can fit in the cache system.

From the implementation point of view, the parallelization of the particle-grid interaction (density accumulation) may give rise to race conditions. The most obvious strategy consists in organizing the particles into clusters and distribute these clusters onto the available cores. Different particles are then likely to provide contributions to a same grid node which entails a race condition between different cores when writing at the same memory address. This issue is overcame thanks to private copies of grid arrays and reduction operations.

Regarding the field interpolation and the particle pusher, the operations related to different particles are independent. Therefore the parallelization is quite straightforward but the scalability may be limited by the poor arithmetic intensity of these steps. On NUMA architectures, the multiplication of cores number comes with an increase of the memory bandwidth: the cores are organized in NUMA domains associated to a local memory for which the access is faster compared to that of an other NUMA domain. The parallelization strategy is tuned to take advantage of the hardware topology. A SPMD (Single Program Multiple Data) is commonly deployed. The particle population is split into multiple samples, one sample being stored in the memory of one NUMA domain and the related operations computed by the cores with a fast access to this memory. The operations on each of the samples are finally reduced over the NUMA domains to recover the complete statistic.

### 3.2.2 Sparse-PIC optimization and parallelization

Sparse grid applications to PIC methods are rather recent [48, 58, 97, 57, 88] and, to date, no efficient implementation combining efficient memory management and tailored parallelization

**Algorithm 7** Projection or charge accumulation

**Require:** Array particle[:]
**Ensure:** Array grid[:] containing the charge density
  **for all** particles **do**
    **Read** positions of particles in particle array [:]
    **Determine** $\xi_1, ..., \xi_8$ the eight nodes of the cell containing the particle
    **for** i **from** 1 **to** 8 **do**
      **Determine** the charge contribution $\rho_i$ of the particle at the node $\xi_i$
      **Add** the contribution $\rho_i$ in the grid array at position $\xi_i$: grid[$\xi_i$]$\leftarrow \rho_i$
    **end for**
  **end for**

strategies has been provided. It is therefore the purpose of the present article. It follows from equation (6.2), that the number of particles required to achieve a given statistical error is much smaller (by hundreds) for the sparse reconstruction, compared to standard PIC methods. As a consequence the particle operations (particle pusher) and the interactions between the full grid and the particles (field interpolation) are no longer the most time consuming operations. The interactions between the component grids and the particles (charge deposition) dominate the computational time because of the large number of those grids (about one hundred). As a result, the strategies proposed in this section are strongly motivated by the mitigation of the charge deposition computational load.

### Memory management

The strategy introduced within this thesis takes advantage of the extremely reduced memory footprint of sparse grids. Indeed, the component grids are stored in tiny arrays with $2^n$, $2^{n+1}$ or $2^{n+2}$ nodes when a full grid requires $2^{3n}$ nodes, $n$ defining the spatial discretization ($h_n = 2^{-n}$). To emphasize this huge memory savings, consider a discretization parameter $n$ ranging from 7 to 9 (corresponding to Cartesian grids with $128^3$ to $512^3$ nodes), the storage of the full Cartesian grid requires 17MB, 134MB, 1GB which does not fit even in the last level cache memory. Conversely, the storage of the largest component grid requires then 4KB, 8KB, 16KB which fits in the L1 data cache memory of a core of modern CPUs, 32KB being the standard for the L1 data cache. Therefore two memory management policies, depicted in the algorithm 8, may be proposed, none of them requiring any particle sorting. Contrariwise, the array used to store the component grids are assumed to fit in the highest level of cache memory, with random accesses to these arrays while the particle arrays are accessed contiguously. This entails random accesses to the component grid arrays, but these non contiguous memory accesses will not generate cache misses since the whole component grid fits in the L1-cache (see figure 4.1).

The first strategy relies on computing the interaction of one particle with all the component grids at once, and then proceed with the next particle. For each particle, the contributions are written successively in every array and thus, in order to maximize memory reuse, the data of all arrays must fit in the cache memory. The size of all these arrays in bytes is:

$$(Total\ no.\ of\ grid\ nodes) \cdot (size\ of\ datatype) = 2^n \left(7n^2 - n + 2\right) \cdot 8 \tag{3.16}$$

The second strategy is the opposite: first the interactions of all the particles are computed for one component grid and then the algorithm proceeds with the following component grid.

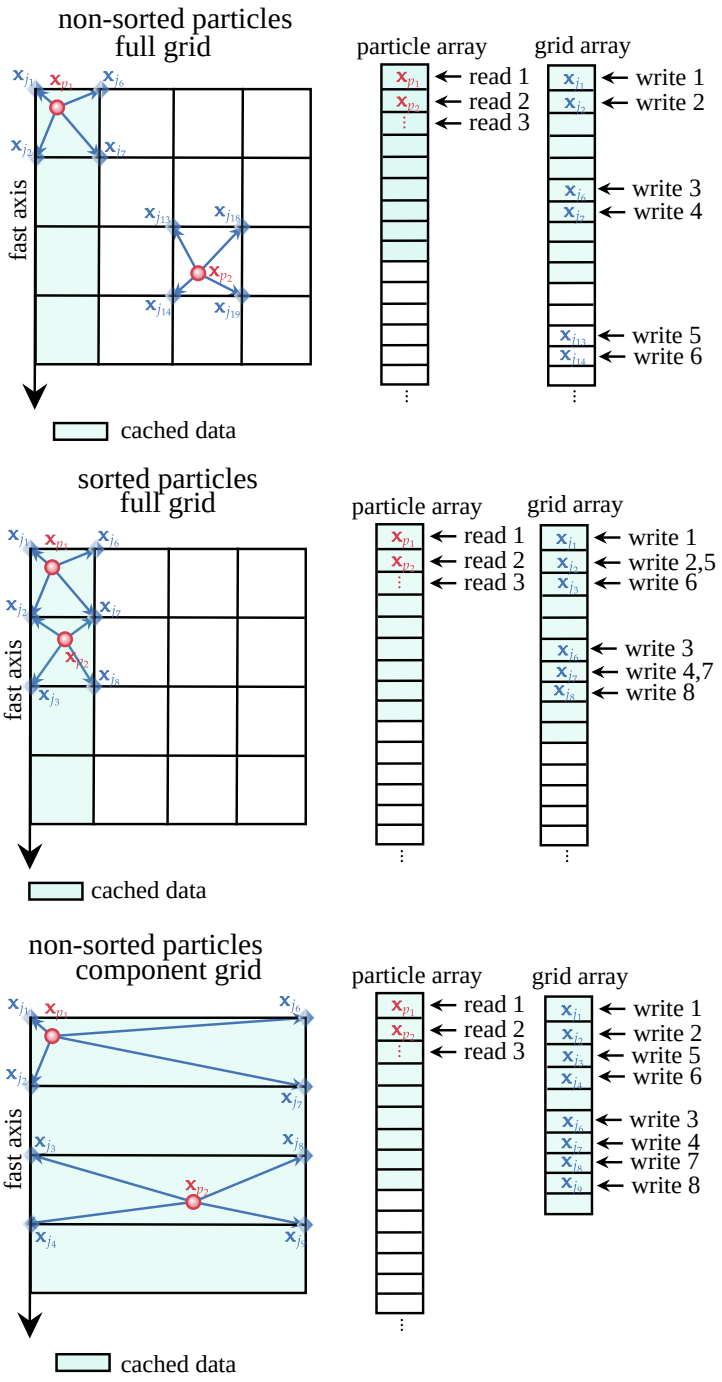


**Figure 3.1.** Cache memory management during charge deposition step. The particles are accessed in order $\mathbf{x}_{p_1}$-$\mathbf{x}_{p_2}$. On the top left the grid data are accessed non-contiguously and must be loaded from the main memory. On the top right the grid data are accessed contiguously and can be loaded from the cache. On the bottom, the grid data are accessed non-contiguously but can be loaded from the cache.

The size of the data that must fit in the cache memory to maximize memory re-use is bounded

by the size of the largest component grid:

$$(no.\ of\ component\ grid\ nodes) \cdot (size\ of\ datatype) \leq 2^{n+2} \cdot 8, \tag{3.17}$$

which is smaller than in the case of the first policy.

**Algorithm 8** Projection or charge accumulation (PIC-Sg)

**Option 1: Particles-component grids loops**

```
for all particles do
  Read positions of particles in particle array
  for all component grids do
    Determine ξ_1, ..., ξ_8 the eight nodes of the cell containing the particle
    for i from 1 to 8 do
      Determine the charge contribution ρ_i of the particle at the node ξ_i
      Add the contribution ρ_i in the grid array at position ξ_i: grid[ξ_i]← ρ_i
    end for
  end for
end for
```

**Option 2: Component grids-particles loops**

```
for all component grids do
  for all particles do
    Read positions of particles in particle array
    Determine ξ_1, ..., ξ_8 the eight nodes of the cell containing the particle
    for i from 1 to 8 do
      Determine the charge contribution ρ_i of the particle at the node ξ_i
      Add the contribution ρ_i in the grid array at position ξ_i: grid[ξ_i]← ρ_i
    end for
  end for
end for
```

The number of component grid varies from 60 to more than 120 for $n$ ranging from 7 to 10. The accumulation of the density on these component grids is therefore expected to be the most time consuming task of the sparse-PIC algorithm. Fortunately these tasks are arithmetic intensive and are therefore expected to offer a good scalability.

### Shared memory parallelization

The parallelization strategy for shared memory architectures exploits the genuine parallelism of the accumulation onto the different component grids. The component grids are arranged into groups and distributed onto the cores. Each core executes successively the tasks (accumulation, field solver operations) on the component grid assigned to it. Finally a reduction operation between the cores is performed to complete the combination step. This approach can however result in load imbalance, as the number of available cores rarely match the number of component grids. An alternative strategy, based on a suitable subdivision of the particle set can help to overcome this problem. Instead of dealing with all particles at once, we divide the particle population into as many clusters of particles as we have computational cores. This approach yields the number of tasks to be completed by the algorithm equal to a multiple of the number

of cores, a task being the accumulation of one particle cluster onto one component grid. Thanks to this procedure, the work load related to the density projection is distributed onto the cores with an ideal balance irrespective to the number of cores and component grids.

Different strategies may be considered for the resolution of the Poisson equation. Within the sparse-PIC method, a Poisson problem shall be solved on each of the component grid. The first strategy consists therefore to distribute the linear systems issued from the dicretization of the Poisson problem on the component grid onto the cores. The advantage of this strategy lies in the very small size of the linear systems. Furthermore, the systems are independent. Nonetheless, the load balance may be poor when the number of component grids is not a multiple of the number of cores.

The grid anisotropy and hence the number of grid nodes, takes an important part in the computational time imbalance during the resolution of Poisson equation: the convergence of iterative methods requires more iterations for larger systems and maximum refinement level $\|\mathbf{l}\|_\infty$. As a workaround, the grids are arranged in a decreasing order according to their complexity in three groups (grid of complexity $2^{n+2}$, $2^{n+1}$ or $2^n$); then, in each group, the grids are arranged in a decreasing order according to their maximum level of discretization $\|\mathbf{l}\|_\infty$.

The second strategy consists in gathering all these problems into a single (by block) linear system. Solving this single system is a more computational expensive task, however it offers a better tuning of the load balance at the level of the linear system solver.

Let us consider now the parallelization of the hierarchization (depicted in algorithm 6) and dehierarchization. Although the procedures are similar, different strategies are considered. The former operates on the component grids $\Omega_{h_\mathbf{l}}$ whereas the latter operates on the full grid $\Omega^{(\infty)}_{h_n}$. Since the hierarchization of the component grids is independent of each other, the parallelization is straightforward similarly to the resolution of the Poisson equation. Concerning the dehierarchization, whose algorithm resembles algorithm 6, one shall notice, when processing a specific dimension $i$, that the operations are independent for each pole (see section 3.1.1). Thus an immediate parallelization, consisting of a distribution of the poles onto the cores, may be conceived from this observation. However, the access of the data is not optimal for all but the innermost dimension. Indeed the grid nodes within a pole are potentially in different cache lines. Different methods such as unrolled unidirectional hierarchization algorithm [73] using blocks of poles or cache-oblivious hierarchization algorithm [74] subdividing the grid into smaller subproblems that completely fit into the cache has been conceived in order to circumvent the issue; however such refinements have not proven to be mandatory to obtain a good efficiency.

### Parallelization for NUMA architectures

NUMA refers to non-uniform memory architectures where the memory access time depends on the memory location of the processors. The memory resides in separate regions, named NUMA domains, and is assigned to groups of cores. A core assigned to a NUMA domain accesses data from its local memory (data stored on the memory of its NUMA domain) much faster than the non-local memory (data stored on another NUMA domain). In this section, we present a second parallelization strategy, designed for NUMA architectures and inspired by existing SPMD implementations for the standard method. It consists in a subdivision of the particle population into samples, each associated and bounded to one unique core or subset of cores. Private arrays and reduction operations are used to avoid race conditions (update of the same memory address by different cores). The efficiency of the strategy is limited by the memory architecture and bandwidth of the hardware since each core accesses simultaneously the particle data. Consequently the number of particle samples shall be chosen accordingly to the number of NUMA domains and stored into the memory banks of these NUMA domains. In order to make efficient use of this strategy, data should be as much as possible accessed within a NUMA

domain. Therefore, several implementation policies shall be respected *.

### Embedding the different parallelization strategies

The different parallelization strategies, though presented independently, shall be merged to define the most effective parallel implementation for a targeted hardware. The population of particles is subdivided into samples of particles of the same size. The number of samples is equal to the number of NUMA domains, each sample being assigned to a single NUMA domain. The parallelization strategy detailed in 3.2.2 is then implemented in every NUMA domain: the local particle sample is decomposed into clusters and the tasks cluster-component grids are distributed on the cores of the NUMA domain. The advantage of this strategy lies in the good exploitation of the increased memory bandwidth brought by the addition of NUMA domains. An illustration of the strategy is provided on figures 3.2, 3.3. On the other hand, this decomposition entails the reduction operation between different NUMA domains.

# Proof of the chapter 3

*Proof of proposition 3.1.1.* From the hierarchical basis representation of functions, it holds:

$$
\begin{aligned}
\Phi_n^c &= \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} I^{\mathcal{H}}_{V_{h_{\mathbf{l}}}} \Phi_{h_{\mathbf{l}}} \\
&= \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} \sum_{\mathbf{k}\le\mathbf{l}} \sum_{\mathbf{i}\in\mathcal{B}_{h_{\mathbf{k}}}} \alpha_{\mathbf{k},\mathbf{i}} \varphi_{h_{\mathbf{k}},\mathbf{i}} \\
&= \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} \sum_{|\mathbf{k}|_\infty\le n} \sum_{\mathbf{i}\in\mathcal{B}_{h_{\mathbf{k}}}} \beta_{\mathbf{k},\mathbf{i}} \varphi_{h_{\mathbf{k}},\mathbf{i}}, \qquad \text{where } \beta_{\mathbf{k},\mathbf{i}} = \begin{cases} \alpha_{\mathbf{k},\mathbf{i}} & \text{if } \mathbf{k}\le\mathbf{l}, \\ 0 & \text{else.} \end{cases} \\
&= \sum_{|\mathbf{k}|_\infty\le n} \sum_{\mathbf{i}\in\mathcal{B}_{h_{\mathbf{k}}}} \left( \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} \beta_{\mathbf{k},\mathbf{j}} \right) \varphi_{h_{\mathbf{k}},\mathbf{i}}. \ \square
\end{aligned}
$$

*The following implementation policies shall be respected:

- In order to access contiguously the particle data, the dimension dedicated to the ids of the particles inside the samples must be the fast axis, *e.g.* in Fortran, the particle array must have the following form:

  $$\text{double particle\_array}[1 : N_s, \ldots];$$

  where $N_s$ stands for the number of particles in a sample.
- A thread shall be bounded to an unique core throughout the simulation so it always has its data in cache and in the same locality region (NUMA domain). It is ensured by the OMP_PROC_BIND=TRUE environment variable.
- The data must be initialized in their respective NUMA domain according to the "first touch" data placement policy.
- The cores of a NUMA domain work on the data stored into the memory local to the NUMA domain thanks to the numactl -l command.

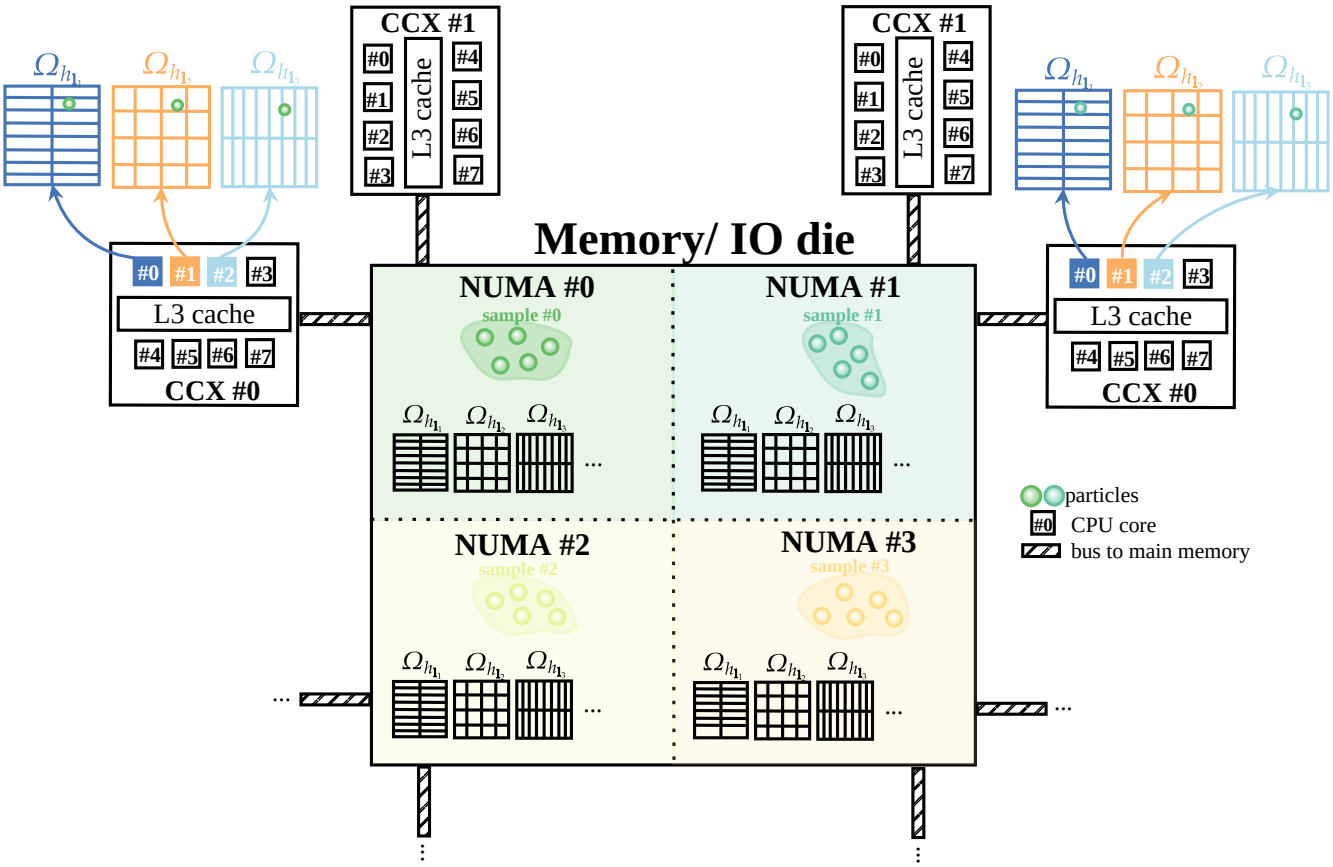


**Figure 3.2.** Embedded particle sample work sharing (a) and component grids work sharing (b) parallelization strategies applied to the AMD EPYC™ 7713 *Milan* architecture. The particles are subdivided into samples, as many as NUMA domains, and the component grids are distributed to the cores of the corresponding NUMA domain, e.g. the grids $\Omega_{h_{l_1}}$, $\Omega_{h_{l_2}}$ and $\Omega_{h_{l_3}}$ are distributed to the cores #0, #1 and #2 of the CCX #0.

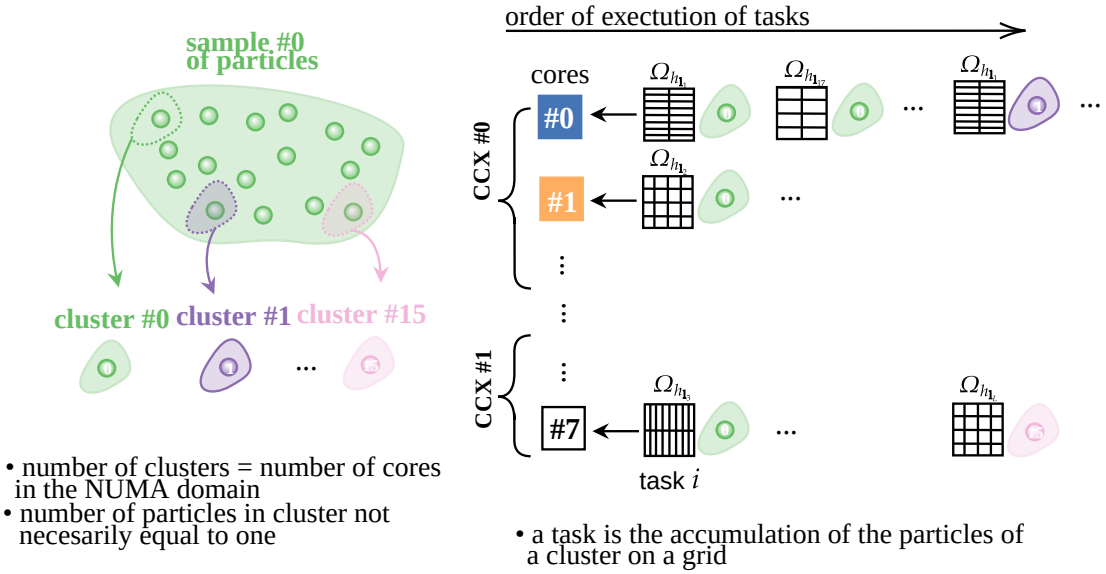


**Figure 3.3.** Load balance strategy within a NUMA domain (illustrated here for one NUMA domain within AMD EPYC™ 7713 *Milan* architecture composed with 2 CCX with 8 cores in a CCX). The particle sample associated to a NUMA domain is subdivided into as many clusters as core within the domain. The number of tasks, i.e. number of component grids*number of clusters (*e.g.* here $|\mathbb{L}_n| * 16$ tasks), is a multiple of the number of cores in the domain. The work load is equally distributed onto the cores.

# Chapter 4

# Parallelization on single GPU architectures

In the last decades, the emergence of General Purpose Graphics Processing Units (GPGPU) has dramatically disrupted the High Performance Computing (HPC) domain with the appearance of accelerators with thousands of compute cores achieving performance in the range of several TFLOP/s ($10^{12}$ instructions per second). Most of supercomputers now employ up to thousands of Graphical Processing Units (GPU), resulting in a total of millions of compute cores. Therefore, an increasing interest of GPGPU for massively parallel applications has emerged in past decades. For instance, several GPU implementations of 1D and 2D PIC simulations have been proposed, demonstrating speed-ups in the range of 20-100 [35, 43, 44, 45, 31, 77, 54, 101, 78].

Recently, sparse-PIC method optimization and parallelization have been investigated for shared memory architectures [49]. Two conclusions can be drawn from these investigations. First, sparse-PIC methods have been proven to be particularly memory efficient, with respect to the size of the grids used to accumulate the density and compute the electric field, as well as the storage of the particles properties. The number of those numerical particles is significantly reduced, compared to standard methods, owing to the better control of the statistical noise. This limited memory footprint calls for the development of parallel implementations on a single GPU architecture. Second, sparse-PIC has demonstrated substantial speed-ups both for sequential and parallel implementation with respect to the standard PIC method. The sequential computational time of the standard method may be reduced by two orders of magnitude with the sparse-PIC approach for an equivalent amount of statistical noise between the two simulations.

The present paper extends the works of [48, 49] to the parallelization of 3D-3V sparse-PIC methods on GPU architectures. Although GPUs offer a limited amount of memory in comparison to CPUs, this is not an issue for sparse-PIC implementations thanks to the substantial gains upon the memory footprint. The purpose of the present paper is to propose a first efficient GPU implementation of the sparse-PIC method with parallelization strategies tailored specifically for accelerator architectures. The sparse grid reconstructions require operations of each particle with numerous (tens) anisotropic grids, with coarse discretizations and different sparsity patterns. These grids are referred to as *component grids*, the set of nodes of all the component grids is being designated by the *sparse grid* terminology.

The efficient implementation introduced in this paper is compared to another one, which is merely the implementation on GPUs of sparse-PIC strategies introduced for shared memory CPU architecture in [49]. The novel implementation combines two level of parallelism for the charge density accumulation: a coarse-grain parallelism based on a particle population decomposition and a Single Instruction Multiple Data (SIMD) parallelism based on a component grid work sharing. This two-level parallelism exploits the architecture of the GPU organized into independent Streaming Multiprocessors (SM), each containing compute cores. The particle population is divided into a large number of particle clusters which are distributed onto the SMs. Within a cluster, each particle contribution is computed for all the component grids at once, enabling a Single Instruction Multiple Threads (SIMT) execution model for the GPU. In

addition, this approach offers a better management of the GPU cache memory (L2-cache) and helps to mitigate the non coalesced memory accesses, detrimental to GPU efficiency. One key element of the implementation is the absence of memory transfers between the host and the device during the simulation since all data fit on the GPU memory.

Sparse-PIC methods may be implemented to significantly increase the computational intensity compared to standard PIC methods known to be memory bounded. This feature is at the core of the GPU implementation to cope with GPUs designed to maximize the number of instructions treated at once.

## Contents



# 4.1 GPU architecture and programming background

The hardware architecture of a GPU differs from that of a Computing Processing Unit (CPU) in some key aspects, the differences being inherited from the initial field of application of GPUs (realtime graphics) where the same instruction has to be applied to a large amount of data [98].

A common CPU is optimized to minimize memory latency, since fetching data from the (off chip) main memory is a very time consuming operation. Therefore, CPU cores have a complex structure and involve out-of-order execution, branch prediction, memory pre-fetching and cache hierarchy, the purpose of all these optimizations being to improve the performance in a Single Instruction Single Data (SISD) fashion. By contrast, GPUs are optimized to maximize throughput, i.e. allowing to execute as many tasks as possible at once. In order to achieve this, a large number of cores, as simple as possible, is required, thus removing all logic that boosts single instruction stream performance but gaining the ability to put more cores on a chip.

A single GPU device consists of multiple Streaming Multiprocessors (SM). The streaming processors can be operated independently. One SM contains tens (*e.g.* 32) compute cores working in a Single Instruction Multiple Thread (SIMT) fashion, meaning that all instructions in all threads are executed in lock-step. A SM contains thousand of registers in order to run a large number of threads simultaneously and perform fast "context switching" between different warps. Typically a SM is assigned 8 thread blocks, consisting of tens of warps, a warp being composed of 32 threads (from the compute capability 2.x, the number of threads in a thread block may exceed 1024).

The SM is oversubscribed with thousands of threads (for tens of cores) in order to hide the memory latency: the warp scheduler of the SM switch quickly from warps stalled due to memory latency to resume a warp for which the data are ready; hence the need of rapid context switching.

The accesses to the GPGPU main memory are coalesced within the threads of a warp. The load and store from and to the main memory are organized in chunks of contiguous memory addresses, these chunks being aimed at feeding all the threads in a warp. To minimize the number of transactions with the main memory, memory accesses within the block shall be ordered to maximize performance: the $k^{th}$ thread in a warp shall access the $k^{th}$ element in the memory chunk.

Several programming languages are available for programmers who want to use GPU resources to accelerate general computational applications, the most widespread being the Compute Unified Device Architecture (CUDA) [91, 37, 34], specific to Nvidia GPUs. However, with the appearance of modern GPUs from competitive brands such as AMD or Intel the restriction to Nvidia GPUs could reveal limiting. Thus, other languages such as OpenMP 4., OpenCL [87] or OpenACC [76] have recently become popular. In the present paper, we implement our code with OpenACC which is an Application Programming Interface (API) written in C, C++, Fortran. OpenACC is a directive-based host driven language, meaning that the host (CPU) is responsible for launching every operations executed on the device (GPU) including execution of kernels (code running on a GPU), allocation of memory and data transfers. OpenACC is not a low-level programming language like CUDA, it allows more portability of the code thanks to an abstraction of the hardware. Though the fine tuning of the compute kernels to the GPU architecture is not in the scope of this API, OpenACC exposes the three levels of parallelism available on an accelerator, namely coarse-grain, fine-grain and SIMD. Gang parallelism is the highest level of parallelism (coarse-grain), equivalent to CUDA thread block, gangs being executed independently of each other without synchronization. The worker level (fine-grain) involves workers, similar to CUDA warps, that share data within a gang. The innermost level of parallelism, the vector parallelism, equivalent to CUDA thread concept, is based on SIMT execution model: an instruction is executed on vector of data.

### 4.1.1 Memory hierarchy

The memory architecture of a GPU also differs from that of a CPU in several aspects. The Nvidia Tesla V100 memory architecture is considered as a representative example in the following. Within GPGPU applications, the device (GPU) does not operate on the host (CPU) main memory but is connected to its own off-chip memory (DRAM). The data have to be transferred from the host global memory to the GPU specific memory. The bandwidth between the device (GPU) and its specific memory is much higher (897 GB/s) than the bandwidth between host memory (CPU) and device memory (16 GB/s)*. Hence, for best overall performance, it is capital to minimize data transfers between the host and the device, even if that means running kernels on the GPU that do not demonstrate any speedup compared to running them on the host CPU. This is the policy followed within this work.

Compared to a CPU, a GPU works with fewer, and relatively small, memory cache layers. The memory hierarchy is sketched in the following lines:

- The main memory (DRAM) consisting of 16GB accessed with a theoretical peak bandwidth of 897GB/s. Global memory can be read and written by all the threads on the GPU.
- The constant memory (64 KB read-only memory) which is faster than global memory because it is cached. Constant memory can be read by all the threads on the GPU.
- The L2 cache of 6.3MB shared by all SMs with a theoretical peak bandwidth of 4.1TB/s can be read and written by all threads.

*For a node of the supercomputer OLYMPE from CALMIP, equipped with a Nvidia Tesla V100 associated to an Intel® Skylake CPU with a PCI GEN3 16X bus of 15.8 GB/S.

- The L1 data cache of 128KB per SM, made of shared memory and texture memory. Shared memory provides high bandwidth and low latency but can only be read and written by the threads belonging to the same thread block (in CUDA terminology). It has a theoretical peak bandwidth of approximately 14TB/s.
- The register file of 256KB for each SM (16,384 32-bit registers on each processing block, 4 per SM). It allows fast read-write operations to the data stored in it. Registers are private to one thread and can only be accessed by the owning threads.

As mentioned before, the memory architecture is designed to optimize coalesced data transfers with the main memory. Consecutive threads from the same warp should access consecutive blocks of memory addresses in order to optimally exploit the memory bandwidth.

As a summary, in order to conceive an efficient GPU algorithm, the following policies shall be respected:

i) Limit the transfers between host (CPU) and device (GPU).
ii) Favor the locality of the data.
iii) Encourage coalesced data accesses with the memory.
iv) Create as many independent task as possible to mask the memory latency.

The GPGPU-specific algorithm proposed in the present article is compliant with i. and iv., and to some extent ii., but not to iii. because of the nature of the particle-grid operations. These points are outlined in the sequel.

## 4.2 Data management

Sparse-PIC methods dramatically reduce the memory footprint of PIC computations thanks to the significant diminution of the number of particles necessary to maintain an appropriate statistical noise. Let us investigate the memory requirements of both the standard and the Sparse PIC methods. The amount of data in bytes [B] to handle resulting from $N$ numerical particles is given by:

$$DataParticles\ [B] = N * 3 * (SizePositionData + SizeVelocityData + SizeAccelerationData). \tag{4.1}$$

Position, velocity and acceleration data are double precision, requiring 8 Bytes [B]. The data size for a contribution (charge density or electric potential) of all the component grids can be bounded by:

$$\begin{aligned} DataComponentGrids\ [B] &= SizeData * \sum_{\mathbf{l}\in\mathbb{L}_n} \left( \prod_{j=1}^{3} (2^{l_j} + 1) \right) \\ &\leq SizeData * |\mathbb{L}_n| * 9 * (2^n + 1). \end{aligned} \tag{4.2}$$

For instance for $n$ ranging from 7 to 10, corresponding to configurations equivalent to $128^3$ and $1024^3$ grids with respect to the standard method ($h_7 = 1/128$ to $h_{10} = 1/1024$), the data size of the component grid ranges from 594KB to 10MB. It results in a significant reduction compared to the Cartesian grid of the PIC method:

$$DataCartesianGrid\ [B] = SizeData * (2^n + 1)^3, \tag{4.3}$$

which requires 17MB to 8GB for $n$ ranging from 7 to 10. A comparison of the particle data requirements, relative to the number of particle per cell (defined in equations (6.1), (6.2)) and the grid data sizes is provided on figure 4.2.

This outlines one main advantage of Sparse PIC method over standard ones when porting to GPGPU: the whole data necessary during the simulation fit into the device memory of most accelerators (tens of GB capacity), even for configurations equivalent to $1024^3$ grids. Therefore our data management strategy shall capitalize on this observation. In order to avoid as much as possible data transfers between the host and the device, the data shall stay on the device memory throughout the whole simulation. One unique data transfer is realized at the initialization to send all the data on the device.

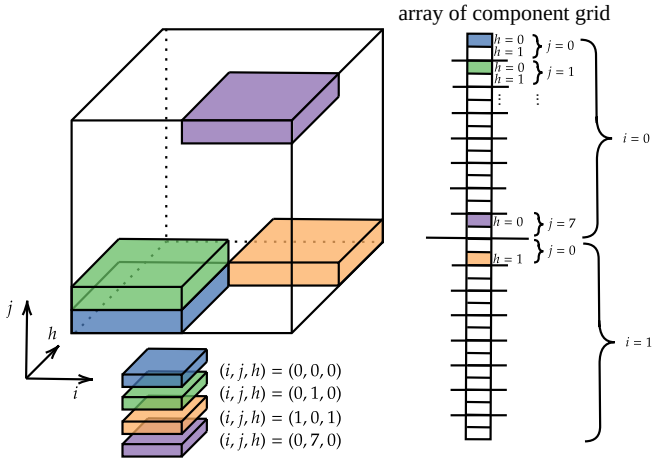


**Figure 4.1.** 1-dimensional structure array of a component grid. The cells of the three-dimensional grid are arranged in a one-dimensional array where the $z$-dimension is the fast axis and the $x$-dimension is the slow axis.

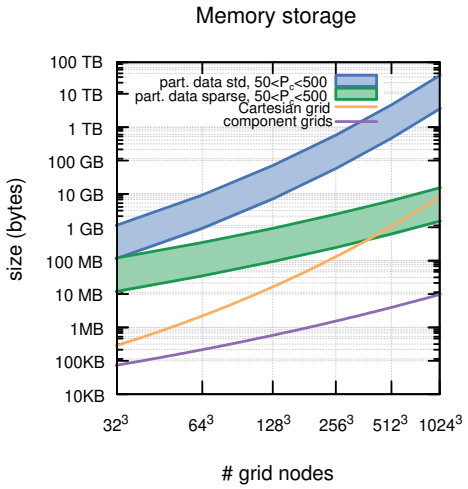


**Figure 4.2.** The storage size of data (particle data, Cartesian grid data and component grid data) is represented as a function of the number of grid nodes and particles per cell.

### 4.2.1 Data structure

The state of each numerical particle is described by its position, velocity and acceleration (corresponding to the electric field component) and is represented by the tuple $\langle x, y, z, v_x, v_y, v_z, a_x, a_y, a_z \rangle$. The position, velocity and acceleration coordinates are represented by double-precision floats. The particle data are represented by three arrays of $N$ rows and three columns:

$$double\ \mathbf{x}_p[1:N, 1:3], \mathbf{v}_p[1:N, 1:3], \mathbf{a}_p[1:N, 1:3];$$

where $N$ is the number of particles.

In order to represent the set of component grids, a three dimensional array (standing for one component grid) is required for all the grids. Since the size of the component grids differs from one to another, a more complex data structure than a four-dimensional array is required. In this paper, two different data structures policies representing the set of component grids are considered. The component grids are either represented by an Array of Structure (AoS) or a two-dimensional array.

The first policy, being the most natural and consisting in an AoS, has been introduced in [49] specifically for the parallelization of sparse-PIC methods on shared memory (CPUs) architecture.

With this policy, a data structure is created to store the dimensions and contributions (charge density, electric potential) of a component grid:

$$
\begin{aligned}
&type\ component\_grid\\
&\quad integer :: k, l, m;\\
&\quad double :: \rho(0:2^k, 0:2^l, 0:2^m), \Phi(0:2^k, 0:2^l, 0:2^m);\\
&end\ type
\end{aligned}
$$

where $k, l, m$ are the dimension of the component grid and $\rho$, $\Phi$ are the arrays containing the charge density and the electric potential contributions. All the information of the component grids are stored in a single array whose elements are component grid data structures:

$$type(component\_grid) \ :: comp\_grid[1 : |\mathbb{L}_n|],$$

where $L$ is the number of component grids, given by equation (13). Let us recall that for sparse-PIC parallelization strategies designed for CPUs, the particle properties are accumulated onto one component grid for all the particles, this repeated for each component grid (see [49] for details). This strategy, together with the AoS data structure, entails locality of the data. For the deposition of the particle density onto a component grid, the array used to store one component grid being small enough to be nursed in L1 cache of a CPU core. However, in the following, a GPU implementation based on the converse strategy, consisting in the accumulation of one particle property onto all the component grids, repeated for each particle, is introduced. Therefore, in order to preserve a good locality of the data between the component grids, a specific data structure is introduced. The rationale for this strategy is proposed in the next section.

This second policy is based on a transformation of the component grids from a three-dimensional structure into a one-dimensional structure. Each component grid is reshaped into a one-dimensional array (see figure 4.1) and all the one-dimensional component grid contributions are stored in an array of $|\mathbb{L}_n|$ rows and $9(2^n + 1) + 1$ columns:

$$double \ :: \rho[1 : |\mathbb{L}_n|, 1 : 9 * (2^n + 1)], \Phi[1 : L, 1 : 9 * (2^n + 1)].$$

This data structure is based on an upper estimation of the total number of component grid nodes. Indeed, the largest (in term of number of grid nodes) of the component grids are the most anisotropic ones, *i.e.* the ones corresponding to the levels $\mathbf{l} = (n, 1, 1); (1, n, 1); (1, 1, n)$, each containing $9 * (2^n + 1)$ grid nodes.

## 4.3 GPGPU implementation of charge deposition

The accumulation of the particle properties onto the component grids accounts for more than 90% of a sparse-PIC iteration (for a sequential execution on CPU). This step is therefore anticipated to be the key point to obtain an efficient parallel implementation on GPGPU. It consists mainly in reading one particle coordinates, computing the grid cell it is contained in, and accumulate the contribution onto the 8 nodes of this cell. These operations are repeated for each grid. During the procedure, two consecutive particles (with regard to their indices in the particle coordinate array) may contribute to different cells in the density array. Therefore either a contiguous memory access in the particle array or in the density (grid) array occurs. Standard implementations usually entail random memory accesses in the density array. In addition, the paralellization of the density accumulation may lead to race conditions when threads associated to different particles add their contributions to the same grid cell, writing at the same memory address. This issue is usually bypassed with either private copies of grid arrays and reduction operations or atomic operations.

### 4.3.1 Why sparse-PIC parallel implementations designed for shared memory (CPU) architectures are not efficient on GPUs

Let us first recall the main features of the sparse-PIC parallel implementation designed for CPUs, then, in the following of the section, we introduce the extension of the implementation to GPUs, named CPU-inherited implementation and finally investigate the limitations of the algorithm. The implementation details of the CPU-inherited algorithm are provided in algorithm 9.

Sparse-PIC parallelization efficiency on CPU is chiefly based on cache memory reuse. The non contiguous memory accesses to the grid data are mitigated by the large number of L1-cache hits. The cache reuse is maximized by considering the following charge density accumulation policy: the interactions of all the particles are computed with one component grid and repeated as many times as the number of component grids. Therefore, since each component grid fits in the L1-cache, the number of grid data cache misses is dramatically reduced, despite the irregular (non contiguous) accesses.

Sparse-PIC parallelization designed for CPUs takes benefit from the natural parallelism offered by the deposition onto the different component grids. The operations attached to two different component grids are independent and can therefore be processed concurrently (by different CPU-cores). Nonetheless this only level of parallelism is not sufficient to provide a good load balance for tens of cores. Therefore, a second level of parallelism is exploited: the particle sampling work sharing. It amounts to decompose the particle population into clusters, distributed onto the different threads. Each thread operates on its own sample of particles, accumulating the particle properties onto a private array. These private copies of the grid data are mandatory to avoid race conditions between threads assigned to different particles. Finally a reduction operation between the private copies is performed to gather the different contributions.

The hurdles of an extension to GPGPU are of different nature. First, the number of threads initiated within a SM (thousands) is large compared to CPUs (tens). This is particularly important for particle-grid operations, the memory accesses being genuinely non contiguous and non coalesced. Therefore, in order to mask the memory latency of the data transfer and achieve the best performance on GPU, one shall run a large number of instruction streams, *i.e.* a large number of gangs in OpenACC terminology. Nonetheless the number of gangs cannot be chosen as large as it shall be to maximize the performance of the CPU-inherited implementation because of the reduction operation requiring copies of the grids for each gang. The memory capacity, as well as the number of reduction operations to operate the array copies may be a limiting feature for a large number of gangs with this implementation. In addition, the method does not provide an effective memory access pattern of the component grid structure. The randomness of the spatial distribution for two consecutive particles in the particle array results in non-coalesced (random) memory accesses in the component grid array. The cumulative features of non efficient memory accesses and poor number of gangs (being the result of the reduction operation) may limit the efficiency of CPU-inherited implementation on GPU.

Second, the L1-cache is private to one CPU core while it is shared by all the SIMD processors within a SM. Each CPU core owns a private L1-cache of usually 32KB whereas GPU cores within a SM share a L1-cache of 128KB on the Tesla V100. As a result, based on the maximum number of threads within a SM for the Tesla V100, which is 2048, each thread has roughly 62B of available L1-cache memory which is significantly less than the 32KB L1-cache memory of the CPU cores. The private L1-cache being at the core of the efficiency of the CPU implementation of the sparse-PIC charge deposition (see [49]), it explains why an efficient GPGPU parallel implementation can not be conceived as a CPU parallel implementation run with thousand of threads rather than tens.

A third difficulty shall be pointed out here: OpenACC only provides a weak control of the cache hierarchy. On top of that, in the CPU parallel implementation, a reduction gathering all the particle contributions is performed for each component grids on a three-dimensional array

$\rho$ corresponding to the AoS component grid data structure (see section 4.2.1). OpenACC only offers reduction on the coarsest grain parallelism (gang level loop) via the "REDUCTION" clause, restricting the parallelization within the CPU-inherited policy (*i.e* the interactions of all the particles are computed with one component grid and repeated as many times as the number of component grids). It results from this implementation a number of independent kernels equal to the number of component grids ($L$). Therefore the two-level parallelization strategy, efficient on CPUs, is limited on GPUs.

### 4.3.2 Why CPU and GPU implementations of standard PIC methods are not suitable for sparse-PIC methods

The main objective of an efficient parallelization strategy for the charge density accumulation within PIC methods is to mitigate the randomness of the data access and entail contiguous (CPU) or coalesced (GPU) memory accesses.

The most natural strategy to deal with randomness of data accesses is inspired from parallelization strategies designed for CPUs and consists in a sorting of the particle population and distribution into clusters. Usually, on CPUs, the particle population is sorted so that most of the time consecutive particles write their contributions in the same density array cells, enabling efficient cache memory reuse.

CPU particle sorting is not applicable to sparse-PIC computations on GPUs though, because, in order to optimize memory transfer, one shall fetch data from the device memory in a coalesced fashion. Therefore, an effective sorting shall result in a particle array where consecutive particle correspond to consecutive grid cells, *i.e.* one sorted particle array for each component grid, which is a way too cumbersome data constraint.

The most efficient GPGPU parallelization strategy to reduce the effects of the irregular memory accesses is to consider the shared memory of the device [77, 31, 54, 101]. With this approach, a private copy of the density array is created in the shared memory of each SM (actually each gang). The accumulation of the particles properties onto the grid is performed with shared-memory atomic operations for threads from the same gang. Finally, a global reduction operates between the different copies of the density array from each gang. This strategy is usually performed along with a particle cluster work sharing strategy, where the particle population is divided into clusters of particles, each assigned to a thread block. In order to ensure that all particles in a cluster are stored contiguously and can deposit to the accumulating density array in the shared memory, a sorting based on the cluster index every time step may be necessary [78, 44].

Whereas CUDA provides a simple and effective way to use the shared memory of the GPU, this features is not available in a transparent and straightforward way on OpenACC. The "CACHE" directive allows to access the shared memory but only for simple memory access patterns and does not suit the algorithm specificity .

### 4.3.3 Sparse-PIC implementation for GPUs

The extension to GPU of the implementation designed for CPUs reveals an inefficient use of the computational capacities since the kernels (each dedicated to the accumulation of the particle properties onto one component grid) are executed in sequential, one after the others. This is the result of the CPU-inherited policy and OpenACC restrictions. In addition, the CPU-inherited implementation does not take advantage of the cache memory as it does on CPUs. We therefore propose a second implementation of the accumulation step for GPGPUs, named GPGPU-specific implementation, taking advantage both of the independent computations between the component grids and the large L2-cache memory capacity.

**Algorithm 9** GPGPU implementation of the CPU-inherited charge accumualtion algorithm.

**Require:** Particle position array $\mathbf{x}_p[1:N, 1:3]$, AoS $comp\_grid[1:|\mathbb{L}_n|]$, weight of particles $\omega$.
**Variables:** Integer: $i_x, i_y, i_z$, real: $i_{xr}, i_{yr}, i_{zr}, s_{x1}, s_{x2}, s_{y1}, s_{y2}, s_{z1}, s_{z2}$
**for** each component grid $i \in [\![1, |\mathbb{L}_n|]\!]$ **do**
$k \leftarrow comp\_grid[i]\%k$ ; $l \leftarrow comp\_grid[i]\%l$ ;$m \leftarrow comp\_grid[i]\%m$
!$ACC PARALLEL NUM_GANGS() VECTOR_LENGTH()
!$ACC LOOP REDUCTION (+:$\rho$) //Parallelism on the SMs and the cores of the SMs
**for** each particle $i_p \in [\![1, N]\!]$ **do**
// Read particle data
$i_{xr} \leftarrow \mathbf{x}_p[ip, 1]/2^{n-k}$;
$i_{yr} \leftarrow \mathbf{x}_p[ip, 2]/2^{n-l}$;
$i_{zr} \leftarrow \mathbf{x}_p[ip, 3]/2^{n-m}$;
// Determine grid cell containing particle
$i_x \leftarrow i_{xr}$ ; $i_{xr} \leftarrow i_{xr} - i_x$;
$i_y \leftarrow i_{yr}$ ; $i_{yr} \leftarrow i_{yr} - i_y$;
$i_z \leftarrow i_{zr}$ ; $i_{zr} \leftarrow i_{zr} - i_z$;
// Determine charge contribution of particle
$s_{x1} \leftarrow (1 - i_{xr}) * 2^k$ ; $s_{x2} \leftarrow i_{xr} * 2^k$;
$s_{y1} \leftarrow (1 - i_{yr}) * 2^l$ ; $s_{y2} \leftarrow i_{yr} * 2^l$;
$s_{x1} \leftarrow (1 - i_{zr}) * 2^m$ ; $s_{z2} \leftarrow i_{zr} * 2^m$;
// Add contribution to the grid
$comp\_grid[i]\%\rho[i_x, i_y, i_z] \leftarrow comp\_grid[i]\%\rho[i_x, i_y, i_z] + s_{x1} * s_{y1} * s_{z1} * \omega$;
$comp\_grid[i]\%\rho[i_x + 1, i_y, i_z] \leftarrow comp\_grid[i]\%\rho[i_x + 1, i_y, i_z] + s_{x2} * s_{y1} * s_{z1} * \omega$;
$comp\_grid[i]\%\rho[i_x, i_y + 1, i_z] \leftarrow comp\_grid[i]\%\rho[i_x, i_y + 1, i_z] + s_{x1} * s_{y2} * s_{z1} * \omega$;
$comp\_grid[i]\%\rho[i_x{+}1, i_y{+}1, i_z] \leftarrow comp\_grid[i]\%\rho[i_x{+}1, i_y{+}1, i_z] {+} s_{x2}{*}s_{y2}{*}s_{z1}{*}\omega$;

$comp\_grid[i]\%\rho[i_x, i_y, i_z + 1] \leftarrow comp\_grid[i]\%\rho[i_x, i_y, i_z + 1] + s_{x1} * s_{y1} * s_{z2} * \omega$;
$comp\_grid[i]\%\rho[i_x{+}1, i_y, i_z{+}1] \leftarrow comp\_grid[i]\%\rho[i_x{+}1, i_y, i_z{+}1] {+} s_{x2}{*}s_{y1}{*}s_{z2}{*}\omega$;

$comp\_grid[i]\%\rho[i_x, i_y{+}1, i_z{+}1] \leftarrow comp\_grid[i]\%\rho[i_x, i_y{+}1, i_z{+}1] {+} s_{x1}{*}s_{y2}{*}s_{z2}{*}\omega$;

$comp\_grid[i]\%\rho[i_x + 1, i_y + 1, i_z + 1] \leftarrow comp\_grid[i]\%\rho[i_x + 1, i_y + 1, i_z + 1] + s_{x2} * s_{y2} * s_{z2} * \omega$;
**end for**
!$ACC END LOOP
!$ACC END PARALLEL
**end for**

This strategy is based on a component grid work sharing principle in a SIMT pattern, *i.e.* the operations on the component grids are realized in a SIMT fashion: each thread within a gang operates on a different component grid. This strategy is coupled with a particle work sharing strategy at the gang level, where the particle population is divided into clusters and distributed into the gangs similarly to the CPU-inherited implementation (see figure 3.2):

- The first level (coarse-grain) of parallelism is based on the distribution of the particle population into clusters. The gangs (or thread blocks) are associated to the clusters and distributed to the SMs in a Single Program Multiple Data (SPMD) fashion.
- The second level (SIMT) of parallelism is based on the component grid work sharing principle. Within each gang, the threads operate on the component grid at the same time in a SIMT fashion. *E.g.* if there are 32 threads in a gang, the particle properties of the particle cluster are accumulated onto the first 32 component grids simultaneously by the threads.

The number of clusters is expected to be large (a lot larger than the number of SMs) so that a large number of thread blocks is enabled and the interleave stream strategy shall efficiently mask the non coalesced memory accesses.

The goal of the GPGPU algorithm is to exploit the GPU architecture so that the device is fed with a large number of similar instructions on multiple data (accumulation of a particle from one cluster onto all the component grids at once). Nonetheless, in order to avoid the limitations resulting from the reduction operation, an array shared between the gangs is considered along with ATOMIC operations. The details of the implementation are given in algorithm 10.

Unlike the CPU-inherited implementation, the interaction of one particle is computed with all the component grids at once, and repeated for all the particles. It provides an unique particle data memory access for all the component grids, reducing therefore the number of data transfer with the device memory. The two-dimensional component grid data structure is considered (see section 4.2.1) to ensure that threads from the same gang access memory addresses close to each other.

One of the benefits of the method concerns the cache memory reuse. Thanks to the shared status of the density array, it is mutual to all gangs and thus it can benefit from the large size of the L2 cache to mitigate the cost due to the random accesses to the grid array, the data size of the component grid ranging from 594KB to 10MB for $n = 7$ up to $n = 10$.

### 4.3.4 Resolution of Poisson equation, field interpolation, particle pusher, etc.

Sparse-PIC methods offer a significant alleviation of the grid operations with respect to the standard methods, resulting from a diminution of the grid nodes constituting the mesh of the method. Grid quantities are computed on each component grids, the operations being independent from one grid to another. It results in several independent linear systems to solve for the resolution of Poisson equation and necessitates novels parallelization strategies. It exists GPU-based libraries offering tools for the resolution of linear system such as AMGx, MAGMA, and CUDA libraries (CuSolver, CuBlas, CuSparse).

Different strategies may be considered: the first strategy consists in solving the linear systems issued from the dicretization of the Poisson problem on the component grids one after the other. The advantage of this strategy lies in the very small size of the linear systems. The second strategy consists in gathering all these problems into a single (by block) linear system. Solving this single system is a more computational expensive task, however it can better benefit from the computational capacity of the GPU.

For the field interpolation and the particle pusher, a straightforward parallelization, based on a decomposition of the particle population and distribution onto the threads, is proposed. No

**Algorithm 10** GPGPU-specific algorithm of charge density accumulation.

**Require:** Particle position array $\mathbf{x}_p[1:N, 1:3]$, 2d-structure $\rho[1:|\mathbb{L}_n|, 1:9*(2^n+1)]$, weight of particles $\omega$.
**Variables:** Integer: $i_x, i_y, i_z, i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8$, real: $i_{xr}, i_{yr}, i_{zr}, s_{x1}, s_{x2}, s_{y1}, s_{y2}, s_{z1}, s_{z2}, x_p, y_p, z_p$
!$ACC PARALLEL NUM_GANGS() VECTOR_LENGTH()
!$ACC LOOP GANG //Coarse-grain parallelism on SMs, Single Program Multiple Data (SPMD) fashion
**for** each particle $i_p \in [\![1, N]\!]$ **do**
// Read particle data from device memory
$x_p \leftarrow \mathbf{x}_p[ip, 1]$;
$y_p \leftarrow \mathbf{x}_p[ip, 2]$;
$z_p \leftarrow \mathbf{x}_p[ip, 3]$;
!$ACC LOOP VECTOR //Fine-grain parallelism on the cores of the SM, SIMT fashion
**for** each component grid $i \in [\![1, |\mathbb{L}_n|]\!]$ **do**
$k \leftarrow comp\_grid[i]\%k$ ; $l \leftarrow comp\_grid[i]\%l$ ;$m \leftarrow comp\_grid[i]\%m$;
// Adapt particle data to the grid
$i_{xr} \leftarrow x_p/2^{n-k}$;
$i_{yr} \leftarrow y_p/2^{n-l}$;
$i_{zr} \leftarrow z_p/2^{n-m}$;
// Determine grid cell containing particle
$i_x \leftarrow i_{xr}$ ; $i_{xr} \leftarrow i_{xr} - i_x$;
$i_y \leftarrow i_{yr}$ ; $i_{yr} \leftarrow i_{yr} - i_y$;
$i_z \leftarrow i_{zr}$ ; $i_{zr} \leftarrow i_{zr} - i_z$;
// Determine charge contribution of particle
$s_{x1} \leftarrow (1 - i_{xr}) * 2^k$ ; $s_{x2} \leftarrow i_{xr} * 2^k$;
$s_{y1} \leftarrow (1 - i_{yr}) * 2^l$ ; $s_{y2} \leftarrow i_{yr} * 2^l$;
$s_{x1} \leftarrow (1 - i_{zr}) * 2^m$ ; $s_{z2} \leftarrow i_{zr} * 2^m$;
// Determine cell of the 2d-structure
$i_1 \leftarrow i_z + i_y * (2^m + 1) + i_x * (2^m + 1) * (2^l + 1)$;
$i_2 \leftarrow i_1 + (2^m + 1) * (2^l + 1)$;
$i_3 \leftarrow i_1 + 2^m + 1$ ; $i_4 \leftarrow i_2 + 2^m + 1$;
$i_5 \leftarrow i_1 + 1$ ; $i_6 \leftarrow i_2 + 1$ ; $i_7 \leftarrow i_3 + 1$ ; $i_8 \leftarrow; i_4 + 1$
// Add contribution to the grid
!$ACC ATOMIC UPDATE
$\rho[i, i_1] \leftarrow \rho[i, i_1] + s_{x1} * s_{y1} * s_{z1} * \omega$;
!$ACC ATOMIC UPDATE
$\rho[i, i_2] \leftarrow \rho[i, i_2] + s_{x1} * s_{y2} * s_{z1} * \omega$;
...
!$ACC ATOMIC UPDATE
$\rho[i, i_8] \leftarrow \rho[i, i_8] + s_{x2} * s_{y2} * s_{z2} * \omega$;
**end for**
!$ACC END LOOP
**end for**
!$ACC END LOOP
!$ACC END PARALLEL

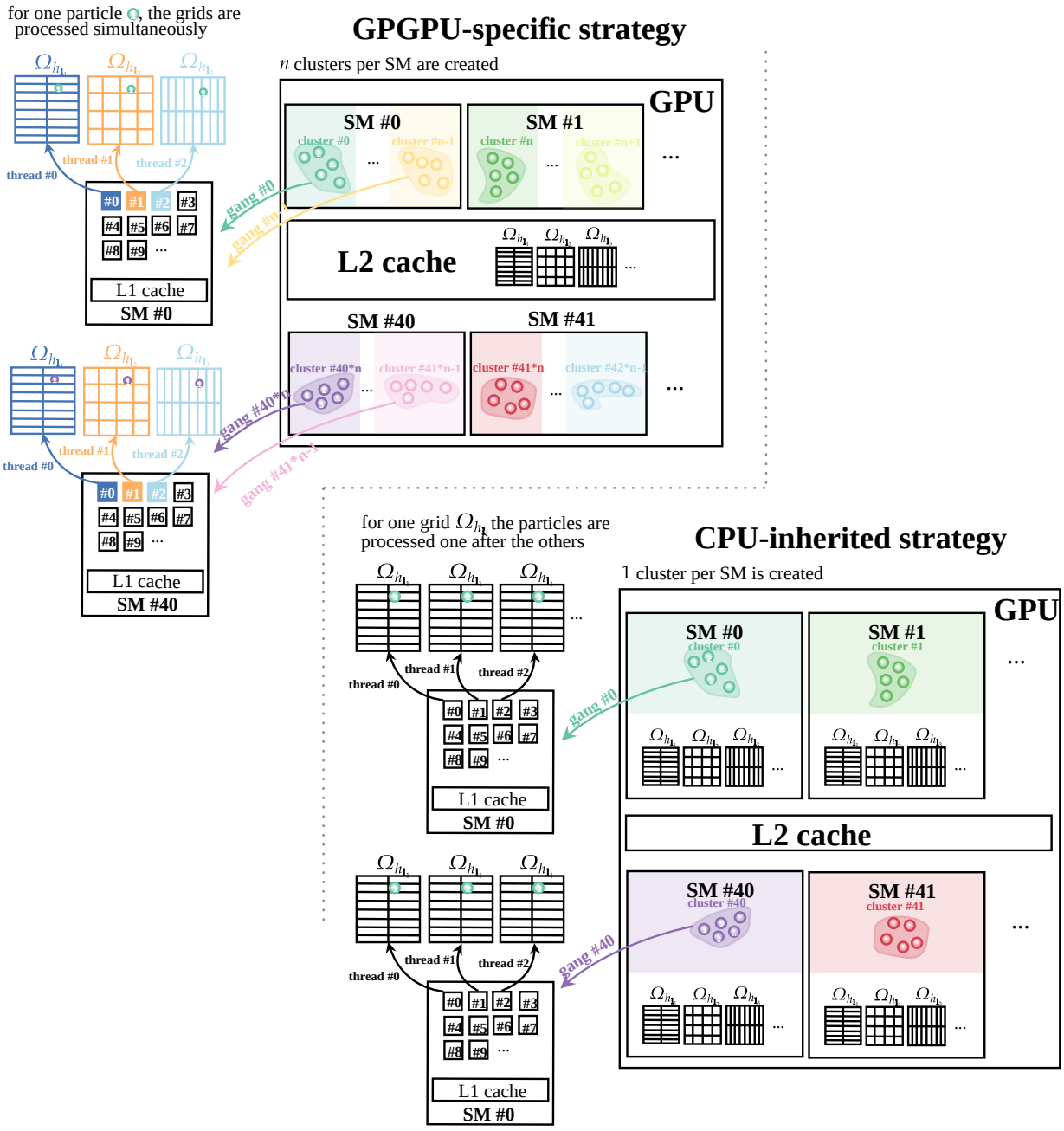


**Figure 4.3.** GPGPU-specific (top) and CPU-inherited (bottom) strategy of the charge deposition kernel. The operation (kernel) is repeated for all the component grids within the CPU-inherited method. In both implementations, the particle population is divided into clusters of particles and distributed onto the SMs (associated to the gangs/thread blocks). In the CPU-inherited algorithm (of the first component grid), the clusters are again divided and distributed to the threads of the SM (gangs/thread blocks). In the GPGPU algorithm, the threads from the same gang operate simultaneously on the component grids in a SIMT fashion.

competitive memory access between the threads (attached to different particles) leading to race

conditions is involved. For the combination, a similar strategy to the one introduced in [49] for the dehierarchization principle (transformation from the hierarchical basis to the nodal basis), is applied both to the hierarchization (transformation from the nodal basis to the hierarchical basis) and the dehierarchization. It exploits the tensor product structure of the basis functions. One-dimensional operations are performed on a collection of two-dimensional poles (see [49]), which are independent and therefore can be paralllelized. This parallelization strategy is not optimal but the combination step usually counts for thousandths of one iteration and therefore a finer parallelization has not proven to be mandatory to obtain a good efficiency.

The differentiation is also straightforward to parallelize. The nodes of the Cartesian grid are distributed to the threads and gangs in a way decided by the compiler.

# Chapter 5

# Semi-implicit sparse-PIC methods

The solutions of Vlasov-Maxwell equations verify some conservation properties, such as the conservation of the total energy and momentum of the system (see proposition 1.2.1), as well as the charge continuity equation which is a consequence of the Vlasov equation (moment of order 0). The question of conservation of these physical quantities in numerical simulations has been very popular for years. Explicit formulations of PIC methods, based on an explicit time integration of the characteristics of the Vlasov equation, are usually momentum-conserving but not energy-conserving. Conversely, PIC implementations based on an implicit formulation can be energy-conserving but not momentum-conserving. The question whether a numerical scheme preserving both energy and momentum is possible or not is addressed in [19].

Originally, and still in most applications, PIC implementations are based on an explicit discretization in time of the Vlasov equation, *e.g.* with a leap-frop scheme. An explicit time integration benefits from simplicity of implementation, as well as a poor computational cost per iteration. Nonetheless, explicit approaches suffer from temporal stability constraints, imposing a limit on the time-step discretization, forcing the user to resolve the fastest wave (see equation (1.32)). In addition, these approaches usually feature spatial stability constraints, manifested by numerical instabilities called *aliasing* or *finite grid instability* [82, 72] occurring when the grid discretization (grid cell size) is equal or superior to the Debye length of the plasma (defined by $\lambda_D = \sqrt{\varepsilon_0 T_e / q_e n_e}$). Therefore, the application of explicit approaches to multidimensional problems, especially for three dimensional geometries or large plasma densities, can be very computationally demanding and cumbersome.

In response to these issues, implicit formulations of PIC schemes have emerged and received a lot of attention, particularly thanks to their stability properties. Indeed, (semi-)implicit PIC methods such as the implicit-moment method [20, 85], direct-implicit method [36, 69] and their developments alleviate the numerical constraints, preserving stability with larger time-steps and grid discretizations. Ideally, in implicit formulations, the particle equations and the field equations shall be non-linearly coupled and shall require the resolution of a Newton or Picard iteration. Because of solver efficiency limitations at the early development of implicit methods, linear approximations have been favored at the expense of numerical approximations producing violation of energy conservation and resulting in significant artificial plasma heating or cooling. The methods using a linearization of the particle-field coupling are named semi-implicit methods. Later, a fully implicit approach [30], based on Newton–Krylov methods, in which field-particle couplings are converged to a tight nonlinear tolerance has been developed. In addition to the elimination of both temporal and spatial instability, the scheme offers valuable conservation properties, such as the exact conservation of the discrete energy of the system and exact conservation of the charge continuity equation. Nonetheless, the method requires the resolution of a non-linear system for the particles and field, which can be very computational expensive, especially for multidimensional computations. A few years ago, a semi-implicit

method preserving exactly the total energy of the system [83] has been developed. This Energy-Conserving Semi-Implicit Method (ECSIM) retains the simplicity of explicit schemes, *i.e.* it advances the particles first and then the fields without any iteration, and conserves discrete energy exactly. In this approach, the particle-field coupling is partially linearized, meaning that a part of the particle response to the field is comprised into the field equation, ensuring an exact discrete energy-conservation. Compared to the previous semi-implicit methods, namely the implicit-moment method and the direct-implicit method, the particle pusher and the derivation of the field equation are different. The mover does not require any inner iteration and its complexity is similar to that of explicit formulations. Nonetheless, the field solver presents a significantly more complex structure in comparison to that of explicit schemes in order to conserve energy to round-off. The major advantage over fully-implicit schemes is the reduced complexity of the algorithm, allowing development of the method for three dimensional simulations. Since then, the method has been applied extensively to large-scale kinetic simulations [108, 107, 29, 41]. However, the method is not consistent with the charge continuity equation as the fully-implicit method do. Therefore the error of conservation, or equivalently the consistency with the Gauss law for Vlasov-Ampere (VA) formulations, shall be verified throughout the simulation in order to avoid non-physical behavior of the plasma. Since then, developments addressing this charge continuity issue have been introduced. In [33], a correction inspired of the Boris ($\nabla \cdot \mathbf{E}$) correction, but operating on the particles instead of the field in order to preserve energy conservation is proposed. The method uses local linearization of the particle shapes and requires the resolution of an under-determined system on the particles with Lagrange multiplier method. Besides, a prediction-correction scheme [25] inspired both of the ECSIM scheme and of a charge-conserving scheme based on an averaging of grid quantities over interpolated trajectories of particles has been proposed.

It has already been extensively discussed in this manuscript that Particle-In-Cell schemes also contain another major hurdle: the statistical error originating from the sampling of the probability density function by a finite number of numerical particles. This numerical noise decreases slowly with the increase of the average number of particles per cell, scaling as the inverse square root of the mean number of particles per cell. Therefore, a large number of particles may be required, necessitating tremendous computational resources.

Sparse grid reconstructions in PIC methods aim at reducing the statistical error resulting from the particle sampling. In sparse grid reconstructions, the particle distribution moments are computed on a hierarchy of component grids with coarse resolution. Compared to standard grids, the mean number of particles per cell is larger for any of the component grids. This crucial feature offers either a mitigation of the statistical noise or a decrease of the total number of numerical particles required for a precision comparable to discretizations on a standard grid. The method has been applied to explicit Vlasov-Poisson formulation in the previous chapters first in two dimensions [48], then extended to three dimensional geometries [49, 50]. Substantial gains in term of memory consumption as well as computational time have been pointed out, by two or three orders of magnitude in comparison to approaches with standard grids. Besides, sparse grid reconstructions have proven to preserve exact momentum conservation of explicit formulations. These observations call for the development of an implicit formulation embedding sparse grid reconstructions and preserving energy conservation as well as improved spatial and temporal stability properties.

The present chapter is dedicated to the development of a semi-implicit scheme, inspired of ECSIM and based on a sparse grid reconstruction of the electric field. In this chapter, we consider an electrostatic regime in which Maxwell's equations fall down to Ampere equation without magnetic field. The objective is to develop an energy-conserving semi-implicit scheme with sparse grid reconstructions featuring the following properties:

i) The scheme is unconditionally stable with respect to the plasma period: the time step can be chosen irrespective to this value.

ii) Discrete energy is exactly conserved for any time-step or grid discretization.
iii) The aliasing or finite grid instability is eliminated, allowing the user to take any desired grid discretization without being forced to resolve the Debye length.
iv) The statistical error is significantly reduced compared to ECSIM with standard grid for the same number of particles.

Nonetheless, the first step is to derive an electrostatic ECSIM method based on a Vlasov-Ampere formulation. Indeed, to our knowledge, all the ECSIM methods described in the literature are based on Vlasov-Maxwell system and include a self-consistent magnetic field.

## Contents



# 5.1 Energy-conserving semi-implicit sparse PIC scheme

## 5.1.1 Electrostatic Vlasov-Ampere formulation

In this section, the non-relativistic Vlasov-Maxwell system is considered in the electrostatic regime, *i.e.* assuming no magnetic field $\mathbf{B} = 0$. In this context, the Vlasov-Maxwell system falls down to a Vlasov-Ampere (VA) formulation:

$$(VA): \begin{cases} \dfrac{\partial f_s}{\partial t}(\mathbf{x},\mathbf{v},t) + \mathbf{v}\cdot\nabla_{\mathbf{x}} f_s(\mathbf{x},\mathbf{v},t) + \dfrac{q_s}{m_s}\mathbf{E}(\mathbf{x},t)\cdot\nabla_{\mathbf{v}} f_s(\mathbf{x},\mathbf{v},t) = 0, \\ \nabla\times\mathbf{E}(\mathbf{x},t) = 0, \\ \epsilon_0 \dfrac{\partial \mathbf{E}}{\partial t}(\mathbf{x},t) = -\mathbf{J}(\mathbf{x},t). \end{cases} \tag{5.1}$$

The system is defined for $(\mathbf{x},\mathbf{v},t) \in \Omega\times\mathbb{R}^d\times\mathbb{R}^+$. In this problem, $f_s(\mathbf{x},\mathbf{v},t)$ is the phase-space distribution function attached to the species $s$; $q_s$, $m_s$ are the corresponding charge and mass, $\epsilon_0$ is the vacuum permittivity, $\mathbf{E}$ is the electric field and $\mathbf{J}$ is the plasma current density obtained from the phase-space distribution of each species:

$$\mathbf{J}(\mathbf{x},t) = \sum_s \mathbf{J}_s(\mathbf{x},t) = \sum_s q_s \int \mathbf{v} f_s(\mathbf{x},\mathbf{v},t) d\mathbf{v}. \tag{5.2}$$

The electric field is initialized with the Gauss law and requires the resolution of a Poisson equation for the electric potential, denoted $\Phi$:

$$\mathbf{E}(\mathbf{x},0) = -\boldsymbol{\nabla}\Phi(\mathbf{x},0), \qquad -\varepsilon_0\Delta\Phi(\mathbf{x},0) = \rho(\mathbf{x},0), \tag{5.3}$$

where $\rho(\mathbf{x},0)$ is the plasma charge density at initialization defined from the initial distribution of each species:

$$\rho(\mathbf{x},t) = \sum_s \rho_s(\mathbf{x},t) = \sum_s q_s n_s, \quad n_s = \int f_s(\mathbf{x},\mathbf{v},t) d\mathbf{v}. \tag{5.4}$$

**Remark 5.1.1** *Provided that the charge continuity equation is verified:*

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{J} = 0, \tag{5.5}$$

*the Vlasov-Ampere formulation (5.1) is equivalent to a Vlasov-Poisson formulation:*

$$(VP) : \begin{cases} \dfrac{\partial f_s}{\partial t}(\boldsymbol{x}, \boldsymbol{v}, t) + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f_s(\boldsymbol{x}, \boldsymbol{v}, t) + \dfrac{q_s}{m_s} \boldsymbol{E}(\boldsymbol{x}, t) \cdot \nabla_{\boldsymbol{v}} f_s(\boldsymbol{x}, \boldsymbol{v}, t) = 0, \\ -\epsilon_0 \Delta \Phi(\boldsymbol{x}, t) = \rho(\boldsymbol{x}, t), \quad \boldsymbol{E}(\boldsymbol{x}, t) = -\boldsymbol{\nabla} \Phi(\boldsymbol{x}, t). \end{cases} \tag{5.6}$$

By considering the charge continuity equation (5.5), one can derive from the Ampere equation an evolution equation for the electric potential $\Phi$:

$$\varepsilon_0 \frac{\partial \Delta \Phi}{\partial t}(\mathbf{x}, t) = \boldsymbol{\nabla} \cdot \mathbf{J}(\mathbf{x}, t). \tag{5.7}$$

The equation can alternatively be obtained by considering the divergence of the Ampere equation (third equation of the system (5.1)) and the Gauss law ($\epsilon_0 \nabla \cdot \mathbf{E} = \rho$). From this equation, a multi-dimensional electrostatic VA formulation is considered:

$$(VA^{\star}) : \begin{cases} \dfrac{\partial f_s}{\partial t}(\mathbf{x}, \mathbf{v}, t) + \mathbf{v} \cdot \nabla_{\mathbf{x}} f_s(\mathbf{x}, \mathbf{v}, t) + \dfrac{q_s}{m_s} \mathbf{E}(\mathbf{x}, t) \cdot \nabla_{\mathbf{v}} f_s(\mathbf{x}, \mathbf{v}, t) = 0, \\ \epsilon_0 \dfrac{\partial \Delta \Phi}{\partial t}(\mathbf{x}, t) = \boldsymbol{\nabla} \cdot \mathbf{J}(\mathbf{x}, t), \quad \mathbf{E}(\mathbf{x}, t) = -\boldsymbol{\nabla} \Phi(\mathbf{x}, t). \end{cases} \tag{5.8}$$

The formulation is equivalent to the first one (5.1) in multi-dimensions if the charge continuity equation (or Gauss law) is verified. Indeed, since the electric field is derived from a potential, its curl vanishes:

$$\boldsymbol{\nabla} \times \mathbf{E} = -\boldsymbol{\nabla} \times \boldsymbol{\nabla} \Phi = 0. \tag{5.9}$$

### 5.1.2 Description of the method

Our goal is to derive an electrostatic method, valid for multi-dimensions embedding sparse grid reconstructions, inspired of the Energy-Conserving Semi-Implicit Method (ECSIM) introduced by Lapenta in [83]. Lapenta's method benefits from an exact conservation of the total energy and avoids the need for a non-linear iteration between the field and particles. Indeed, the coupling between the particles and the field is linearized by considering a specific mover of the particles. A mass matrix is introduced to express the particle response to the electric field. The problem is then completely self-consistent and linear for the field. In addition, the implicit formulation of the scheme enables unconditional stability for time steps as large as desired. Nonetheless, the time step shall be chosen such that most particles do not move further than one grid cell $v_{the} \Delta t < \Delta x$, where $v_{the} = \sqrt{2 T_e * q_e / m_e}$ is the thermal velocity of the electrons, in order for the scheme to remain accurate. Note that it is not a stability but an accuracy constraint. Eventually, the so-called finite grid instability appearing when the Debye length is not resolved, *i.e.* when the grid discretization is larger than the Debye length of the plasma, is eliminated with the ECSIM scheme. As a result, the method can model problems with large system size using coarse discretization, much larger than the Debye length. Nonetheless, the ECSIM scheme still suffers from the complexity of Particle-In-Cells schemes, handling tremendous number of particles in order to get a fair statistical accuracy.

On the other hand, sparse grid reconstructions for explicit PIC schemes have demonstrated substantial gains on memory footprint [48], loosening significantly the constraints of the method by reducing drastically the number of particles in the simulation. The extension to three dimensional geometries in [49, 50] has proven the method to be far less computational expensive than the explicit approach with standard grid.

The motivation here is to preserve ECSIM properties of conservation and stability while benefiting from the advantages of the sparse PIC methods (reduction of computational time and memory footprint).

### Spatial discretization

In this section, the spatial discretization of the physical quantities (electric field, charge current, electric potential, etc.), as well as operators (gradient and divergence), on the component grids are explicited in two dimensions. The extension to three dimensional geometries contains no difficulty.

Let us consider a component grid $\Omega_{h_\mathbf{l}}$ with grid discretization $h_\mathbf{l}$. The scalar quantities, such as the electric potential, are defined at the vertices of the grid cells:

$$\Phi_{h_\mathbf{l};\mathbf{j}} = \Phi_{h_\mathbf{l};i,j} \in \Omega_{h_\mathbf{l}}, \text{ where } \mathbf{j} = (i,j) \in I_{h_\mathbf{l}}. \tag{5.10}$$

The notation $\Phi_{h_\mathbf{l};i,j}$ stands for the electric potential approximation at the grid node $\mathbf{j}h_\mathbf{l} = (ih_{l_1}, jh_{l_2})$.

The field quantities, such as the electric field and the current density, are defined at the centers and vertices of the grid cells according to the Yee discretization [113]. The Yee discretization of a component grid consists of two staggered component grids (one for each dimension), $\Omega^x_{h_\mathbf{l}}$ and $\Omega^y_{h_\mathbf{l}}$, defined by:

$$\Omega^x_{h_\mathbf{l}} := \left\{\mathbf{j}^x h_\mathbf{l} \mid \mathbf{j} \in I_{h_\mathbf{l}}\right\} \subset \Omega, \quad \Omega^y_{h_\mathbf{l}} := \left\{\mathbf{j}^y h_\mathbf{l} \mid \mathbf{j} \in I_{h_\mathbf{l}}\right\} \subset \Omega, \tag{5.11}$$

where we introduce the notation $\mathbf{j}^{xy}$ for an index $\mathbf{j} \in I_{h_\mathbf{l}}$, defined by:

$$\mathbf{j}^{xy} := \begin{pmatrix} \mathbf{j}^x \\ \mathbf{j}^y \end{pmatrix} := \begin{pmatrix} (i+1/2, j) \\ (i, j+1/2) \end{pmatrix}, \quad \text{for } \mathbf{j} = (i,j) \in I_{h_\mathbf{l}}. \tag{5.12}$$

Let $\Omega^{xy}_{h_\mathbf{l}} := (\Omega^x_{h_\mathbf{l}}, \Omega^y_{h_\mathbf{l}})$ denotes the staggered component grids. Then, the electric field and current density discretizations are written as:

$$\mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}} = \begin{pmatrix} E^x_{h_\mathbf{l};\mathbf{j}^x} \\ E^y_{h_\mathbf{l};\mathbf{j}^y} \end{pmatrix} = \begin{pmatrix} E^x_{h_\mathbf{l};i+1/2,j} \\ E^y_{h_\mathbf{l};i,j+1/2} \end{pmatrix} \in \Omega^{xy}_{h_\mathbf{l}}, \quad \mathbf{J}_{h_\mathbf{l};\mathbf{j}^{xy}} = \begin{pmatrix} J^x_{h_\mathbf{l};\mathbf{j}^x} \\ J^y_{h_\mathbf{l};\mathbf{j}^y} \end{pmatrix} = \begin{pmatrix} J^x_{h_\mathbf{l};i+1/2,j} \\ J^y_{h_\mathbf{l};i,j+1/2} \end{pmatrix} \in \Omega^{xy}_{h_\mathbf{l}}. \tag{5.13}$$

The notation $\mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}$ stands for the electric field approximation on the staggered grid; *i.e.* the first component of the field, $\mathbf{E}^x$, is defined at the grid nodes $\mathbf{j}^x h_\mathbf{l} = ((i+1/2)h_{l_1}, jh_{l_2})$ and the second component of the field is defined at the grid nodes $\mathbf{j}^y h_\mathbf{l} = (ih_{l_1}, (j+1/2)h_{l_2})$.

Let us introduce the discrete gradient and discrete divergence operators defined on the component grids. The discrete gradient is defined from $\Omega_{h_\mathbf{l}}$ to $\Omega^{xy}_{h_\mathbf{l}}$ and the discrete divergence

from $\Omega^{xy}_{h_\mathbf{l}}$ to $\Omega_{h_\mathbf{l}}$ by:

$$\left(\boldsymbol{\nabla}_{h_\mathbf{l}}\Phi_{h_\mathbf{l};\mathbf{j}}\right)_{h_\mathbf{l};\mathbf{j}^{xy}} = \begin{pmatrix} \dfrac{\Phi_{h_\mathbf{l};i+1,j} - \Phi_{h_\mathbf{l};i,j}}{h_{l_1}} \\ \dfrac{\Phi_{h_\mathbf{l};i,j+1} - \Phi_{h_\mathbf{l};i,j}}{h_{l_2}} \end{pmatrix}, \tag{5.14}$$

$$\left(\boldsymbol{\nabla}_{h_\mathbf{l}} \cdot \mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}\right)_{h_\mathbf{l};\mathbf{j}} = \frac{\mathbf{E}^x_{h_\mathbf{l};i+1/2j} - \mathbf{E}^x_{h_\mathbf{l};i-1/2j}}{h_{l_1}} + \frac{\mathbf{E}^y_{h_\mathbf{l};ij+1/2} - \mathbf{E}^y_{h_\mathbf{l};ij-1/2}}{h_{l_2}}. \tag{5.15}$$

The discrete Laplacian operator is recovered by applying successively the discrete gradient and divergence operators, *i.e.*:

$$\left(\Delta_{h_\mathbf{l}}\Phi_{h_\mathbf{l}}\right)_{h_\mathbf{l};\mathbf{j}} := \left(\boldsymbol{\nabla}_{h_\mathbf{l}} \cdot \left(\boldsymbol{\nabla}_{h_\mathbf{l}}\Phi_{h_\mathbf{l};\mathbf{j}}\right)_{h_\mathbf{l};\mathbf{j}^{xy}}\right)_{h_\mathbf{l};\mathbf{j}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}. \tag{5.16}$$

The motivation for the introduction of the staggered discretization is to retain some properties of the continuous gradient and divergence operators. Specifically, the discrete gradient and divergence operators shall verify a discrete integration by parts for exact conservation of energy.

**Lemma 5.1.2 (Discrete integration by parts)** *Let $A_{h_l;\boldsymbol{j}}$ be a scalar quantity defined on the periodic component grid $\Omega_{h_l}$ and $\boldsymbol{B}_{h_l;\boldsymbol{j}^{xy}}$ be a field quantity defined on the staggered periodic component grid $\Omega^{xy}_{h_l}$, then the following discrete integration by parts holds:*

$$\sum_{\boldsymbol{j}\in I_{h_l}} \boldsymbol{\nabla}_{h_l} A_{h_l;\boldsymbol{j}} \cdot \boldsymbol{B}_{h_l;\boldsymbol{j}^{xy}} = -\sum_{\boldsymbol{j}\in I_{h_l}} A_{h_l;\boldsymbol{j}} \boldsymbol{\nabla}_{h_l} \cdot \boldsymbol{B}_{h_l;\boldsymbol{j}^{xy}} \tag{5.17}$$

**Derivation of the scheme: Imp-PIC-Sg**

In PIC methods, the particle distribution $f_s$ is represented by a collection of $N_s$ numerical particles, representing many physical particles, whose positions and velocities are denoted $(\mathbf{x}_p, \mathbf{v}_p)$, $p = 1, ..., N_s$ being the index of the particles. All numerical particles from the same species have a similar charge $q_p$ (resp. mass $m_p$) defined from their physical charge $q_s$ (resp. mass $m_s$) and the ratio of physical particles per numerical particle:

$$q_p = \frac{q_s n_s}{N_s}, \quad m_p = \frac{m_s n_s}{N_s}, \qquad p = 1, ..., N_s. \tag{5.18}$$

Let $\Delta t$ be the step of the time discretization such that $t^k = k\Delta t$, for $k = 0, ..., T/\Delta t$ and $T$ final time. Starting from the ECSIM scheme, the particles are advanced according to:

$$\begin{cases} \mathbf{x}_p^{k+1/2} = \mathbf{x}_p^{k-1/2} + \Delta t \mathbf{v}_p^k, \\ \mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \dfrac{q_p \Delta t}{m_p} \mathbf{E}^{k+\theta}_{h_n}\left(\mathbf{x}_p^{k+1/2}\right). \end{cases} \tag{5.19}$$

In the above, $\mathbf{E}^{k+\theta}_{h_n}$ is the electric field reconstruction, defined in the space domain and evaluated at the particle positions, which are known explicitly from the first equation. It is constructed, according to the combination technique, from the electric field contributions of all component grids, averaged between time $k$ and $k+1$. Specifically, the electric field sparse grid reconstruction

is defined by:

$$\mathbf{E}_{h_n}^{k+\theta}\left(\mathbf{x}_p^{k+1/2}\right) = \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} \mathcal{I}_{V_{h_\mathbf{l}}}\left(\mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k+\theta}\right)\left(\mathbf{x}_p^{k+1/2}\right), \tag{5.20}$$

$$\text{where } \mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k+\theta} = \theta \mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k+1} + (1-\theta)\mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k}, \ \ \theta \in \left[\frac{1}{2}, 1\right] \tag{5.21}$$

We recall that $\mathcal{I}_{V_{h_\mathbf{l}}}$ stands for the interpolation onto the space associated to the component grid of discretization $h_\mathbf{l}$ and spanned by basis functions with support depending on $h_\mathbf{l}$ (see equation (2.10)).

The electric field contribution at the future time step is obtained from the one at the last time step by considering the divergence of the Ampere equation. An equation specific to each component grid is considered:

$$\boldsymbol{\nabla}_{h_\mathbf{l}} \cdot \mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k+1} - \boldsymbol{\nabla}_{h_\mathbf{l}} \cdot \mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k} = -\frac{\Delta t}{\epsilon_0} \boldsymbol{\nabla}_{h_\mathbf{l}} \cdot \mathbf{J}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k+1/2}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}. \tag{5.22}$$

In the above, $\boldsymbol{\nabla}_{h_\mathbf{l}}\cdot$ is the discrete divergence operator associated to the component grid $\Omega_{h_\mathbf{l}}$, and defined in equation (5.14). The average in time current density is defined for each component grid from the particle positions and velocities at present and future time step according to a Monte Carlo method (see section 2.2.1):

$$\mathbf{J}_{h_\mathbf{l};\mathbf{j}^{xy}}^{k+1/2} = \sum_s \mathbf{J}_{s;h_\mathbf{l};\mathbf{j}^{xy}}^{k+1/2} = h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} q_p \mathbf{v}_p^{k+1/2} W_{h_\mathbf{l};p\mathbf{j}^{xy}}, \ \ \forall \mathbf{j} \in I_{h_\mathbf{l}}, \tag{5.23}$$

where

$$h_\mathbf{l}^{-1} := \prod_{i=1}^{d} h_{l_i}^{-1} \tag{5.24}$$

is the cell volume of the component grid $\Omega_{h_\mathbf{l}}$. For ease of presentation, the following notations are introduced for the basis functions:

$$W_{h_\mathbf{l};p\mathbf{j}^{xy}} := W_{h_\mathbf{l};\mathbf{j}^{xy}}\left(\mathbf{x}_p^{k+1/2}\right), \quad W_{h_\mathbf{l};\mathbf{j}\mathbf{j}'} = W_{h_\mathbf{l};\mathbf{j}'\mathbf{j}} := W_{h_\mathbf{l};\mathbf{j}'}(\mathbf{j}h_\mathbf{l}). \tag{5.25}$$

The current density equation (2.121) and the Newton velocity equation (5.19) are linearly coupled. Indeed, the future velocities of the particles are determined from the implicit electric field,computed from the current density defined as a function of the particles implicit velocity. The particle mover (5.19) can be rewritten as follows:

$$\mathbf{v}_p^{k+1/2} = \mathbf{v}_p^k + \frac{q_p \Delta t}{2m_p} \mathbf{E}_{h_n}^{k+\theta}\left(\mathbf{x}_p^{k+1/2}\right). \tag{5.26}$$

This rewriting is the key point of the ECSIM method. The implicit velocity of the particles is decomposed into an explicit velocity and an implicit term corresponding to the particle response to the implicit electric field. Substituting this equation in equation (2.121), the current density

can be expressed as an explicit and an implicit components:

$$\mathbf{J}^{k+1/2}_{h_\mathbf{l};\mathbf{j}^{xy}} = \mathbf{J}^{k}_{h_\mathbf{l};\mathbf{j}^{xy}} + h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} \frac{q_p^2 \Delta t}{2m_p} \mathbf{E}^{k+\theta}_{h_n} \left( \mathbf{x}_p^{k+1/2} \right) W_{h_\mathbf{l};p\mathbf{j}^{xy}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}. \tag{5.27}$$

$\mathbf{J}^{k}_{h_\mathbf{l};\mathbf{j}^{xy}}$ is the explicit component of the current density, defined from the velocities $\mathbf{v}_p^k$ but at position $\mathbf{x}_p^{k+1/2}$:

$$\mathbf{J}^{k}_{h_\mathbf{l};\mathbf{j}^{xy}} = h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} q_p \mathbf{v}_p^k W_{h_\mathbf{l};p\mathbf{j}^{xy}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}. \tag{5.28}$$

Using the expression of the electric field sparse grids reconstruction (see equation (5.20)), and substituting it into the last equation, one gets:

$$\begin{aligned} \mathbf{J}^{k+1/2}_{h_\mathbf{l};\mathbf{j}^{xy}} &= \mathbf{J}^{k}_{h_\mathbf{l};\mathbf{j}^{xy}} + h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} \frac{q_p^2 \Delta t}{2m_p} \sum_{\tilde{\mathbf{l}} \in \mathbb{L}_n} c_{\tilde{\mathbf{l}}} P_{V_{h_{\tilde{\mathbf{l}}}}} \mathbf{E}^{k+\theta}_{h_{\tilde{\mathbf{l}}}} \left( \mathbf{x}_p^{k+1/2} \right) W_{h_\mathbf{l};p\mathbf{j}^{xy}} \\ &= \mathbf{J}^{k}_{h_\mathbf{l};\mathbf{j}^{xy}} + h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} \frac{q_p^2 \Delta t}{2m_p} \sum_{\tilde{\mathbf{l}} \in \mathbb{L}_n} c_{\tilde{\mathbf{l}}} \sum_{\tilde{\mathbf{j}} \in I_{h_{\tilde{\mathbf{l}}}}} \mathbf{E}^{k+\theta}_{h_{\tilde{\mathbf{l}}};\tilde{\mathbf{j}}^{xy}} W_{h_{\tilde{\mathbf{l}}};p\tilde{\mathbf{j}}^{xy}} W_{h_\mathbf{l};p\mathbf{j}^{xy}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}. \end{aligned} \tag{5.29}$$

Let us now derive an evolution equation for the electric potential. The electric potential and electric field are linked by the relation:

$$\mathbf{E}_{h_\mathbf{l};\mathbf{j}^{xy}} = -\boldsymbol{\nabla}_{h_\mathbf{l}} \Phi_{h_\mathbf{l};\mathbf{j}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}. \tag{5.30}$$

The electric potential is decomposed in the nodal basis $\{W_{h_\mathbf{l};\mathbf{j}} \mid \mathbf{j} \in I_{h_\mathbf{l}}\}$:

$$\Phi_{h_\mathbf{l};\mathbf{j}} = \sum_{\mathbf{j}' \in I_{h_\mathbf{l}}} \Phi_{h_\mathbf{l};\mathbf{j}} W_{h_\mathbf{l};\mathbf{j}\mathbf{j}'}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}. \tag{5.31}$$

Introducing the equations (5.29), (5.30) and (5.31) into the equation (5.22), one gets a relation between the electric potential at time $k+1$ and the electric potential at time $k$:

$$\begin{aligned} &\sum_{\mathbf{j}' \in I_{h_\mathbf{l}}} \left( \Phi^{k+1}_{h_\mathbf{l};\mathbf{j}'} - \Phi^{k}_{h_\mathbf{l};\mathbf{j}'} \right) \Delta_{h_\mathbf{l}} W_{h_\mathbf{l};\mathbf{j}\mathbf{j}'} = \\ &\qquad \frac{\Delta t}{\epsilon_0} \boldsymbol{\nabla}_{h_\mathbf{l}} \cdot \mathbf{J}^{k}_{h_\mathbf{l};\mathbf{j}^{xy}} - h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} \beta_p \sum_{\tilde{\mathbf{l}} \in \mathbb{L}_n} c_{\tilde{\mathbf{l}}} \sum_{(\tilde{\mathbf{j}},\mathbf{j}') \in I^2_{h_{\tilde{\mathbf{l}}}}} \Phi^{k+\theta}_{h_{\tilde{\mathbf{l}}};\mathbf{j}'} W_{h_{\tilde{\mathbf{l}}};p\tilde{\mathbf{j}}^{xy}} \boldsymbol{\nabla}_{h_{\tilde{\mathbf{l}}}} W_{h_{\tilde{\mathbf{l}}};\tilde{\mathbf{j}}\mathbf{j}'} \cdot \boldsymbol{\nabla}_{h_\mathbf{l}} W_{h_\mathbf{l};p\mathbf{j}^{xy}}, \end{aligned} \tag{5.32}$$

where $\beta_p = \dfrac{q_p^2 \Delta t^2}{2\epsilon_0 m_p}$.

Let us rewrite the equation (5.32) as a linear system. For a couple of component grids $(\mathbf{l}, \tilde{\mathbf{l}}) \in (\mathbb{L}_n)^2$, two local matrices corresponding to this couple, denoted $M^{(1)}_{h_\mathbf{l}}$, $M^{(2)}_{h_\mathbf{l},h_{\tilde{\mathbf{l}}}}$, are defined

by:

$$M_{h_{\mathbf{l}}}^{(1)} := \left(m_{\mathbf{j},\mathbf{j}'}^{(1)}\right)_{(\mathbf{j},\mathbf{j}')\in I_{h_{\mathbf{l}}}^2}, \qquad \text{where } m_{\mathbf{j},\mathbf{j}'}^{(1)} = \Delta_{h_{\mathbf{l}}} W_{h_{\mathbf{l}};\mathbf{j}\mathbf{j}'} \tag{5.33}$$

$$M_{h_{\mathbf{l}},h_{\tilde{\mathbf{l}}}}^{(2)} := \left(m_{\mathbf{j},\mathbf{j}'}^{(2)}\right)_{(\mathbf{j},\mathbf{j}')\in I_{h_{\mathbf{l}}}\times I_{h_{\tilde{\mathbf{l}}}}}, \qquad \text{where } m_{\mathbf{j},\mathbf{j}'}^{(2)} = \sum_{s}\sum_{p=1}^{N_s}\beta_p \sum_{\tilde{\mathbf{j}}\in I_{h_{\tilde{\mathbf{l}}}}} W_{h_{\tilde{\mathbf{l}}};p\tilde{\mathbf{j}}^{xy}} \boldsymbol{\nabla}_{h_{\tilde{\mathbf{l}}}} W_{h_{\tilde{\mathbf{l}}};\tilde{\mathbf{j}}\mathbf{j}'} \cdot \boldsymbol{\nabla}_{h_{\mathbf{l}}} W_{h_{\mathbf{l}};p\mathbf{j}^{xy}}. \tag{5.34}$$

The first local matrix $M_{h_{\mathbf{l}}}^{(1)}$ is a discretization of the Laplacian operator on the component grid $\Omega_{h_{\mathbf{l}}}$ and shall be named the local Laplacian matrix. The local Laplacian matrices are square, symmetric and of size depending on the number of nodes from the component grid, *i.e.* $M_{h_{\mathbf{l}}}^{(1)} \in \mathcal{M}^{2(2^{\mathbf{l}}+1)}$. The second local matrix $M_{h_{\mathbf{l}},h_{\tilde{\mathbf{l}}}}^{(2)}$ translates the energy exchange between the particles and the field and shall be denoted by the local stiffness matrix. Specifically, the stiffness matrix $M_{h_{\mathbf{l}},h_{\tilde{\mathbf{l}}}}^{(2)}$ represents the effect on the electric potential (computed on the grid $\Omega_{h_{\mathbf{l}}}$) of the electric potential (computed on the grid $\Omega_{h_{\tilde{\mathbf{l}}}}$) response to the particles. The local stiffness matrices are rectangular and of size depending on the number of nodes from each component grid of the couple, *i.e.* $M_{h_{\mathbf{l}},h_{\tilde{\mathbf{l}}}}^{(2)} \in \mathcal{M}^{(2^{\mathbf{l}}+1)\times(2^{\tilde{\mathbf{l}}}+1)}$. Let us recall $|\mathbb{L}_n| := Card(\mathbb{L}_n)$ the number of component grids. There are $|\mathbb{L}_n|^2$ local stiffness matrices but note that only a few more than a half of them shall be computed thanks to the symmetry. Let us numerate all the component grid levels $L = (\mathbf{l}_1, ..., \mathbf{l}_{|\mathbb{L}_n|})$ and let $\Phi_{\mathbb{L}_n}^k$ ($\boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}_{\mathbb{L}_n}^k$ resp.) be the vector of the electric potential (divergence of the current density resp.) approximations on all the component grids at time $k$:

$$\Phi_{\mathbb{L}_n}^k = \begin{pmatrix} \Phi_{h_{\mathbf{l}_1}}^k \\ \vdots \\ \Phi_{h_{\mathbf{l}_1};\mathbf{j}_{h_{\mathbf{l}_1}^{-1}}}^k \\ \Phi_{h_{\mathbf{l}_2};\mathbf{j}_2}^k \\ \vdots \\ \Phi_{h_{\mathbf{l}_{|\mathbb{L}_n|}};\mathbf{j}_{h_{\mathbf{l}_{|\mathbb{L}_n|}}^{-1}}}^k \end{pmatrix}, \quad \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}_{\mathbb{L}_n}^k = \begin{pmatrix} \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}_{h_{\mathbf{l}_1};\mathbf{j}_1^{xy}}^k \\ \vdots \\ \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}_{h_{\mathbf{l}_1};\mathbf{j}_{h_{\mathbf{l}_1}^{-1}}^{xy}}^k \\ \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}_{h_{\mathbf{l}_2};\mathbf{j}_2^{xy}}^k \\ \vdots \\ \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}_{h_{\mathbf{l}_{|\mathbb{L}_n|}};\mathbf{j}_{h_{\mathbf{l}_{|\mathbb{L}_n|}}^{-1}}^{xy}}^k \end{pmatrix} \tag{5.35}$$

From these local matrices, we construct by blocks two global matrices, containing all the component grid contributions. The first matrix, denoted $M_{\mathbb{L}_n}^{(1)}$, is a diagonal by blocks matrix corresponding to the discretization of the Laplacian on the component grids. The second matrix, denoted $M_{\mathbb{L}_n}^{(2)}$, corresponds to the electric potential response of the component grids to the particles, seen by other component grids. The matrices are defined by:

$$M_{\mathbb{L}_n}^{(1)} = \begin{pmatrix} M_{h_{\mathbf{l}_1}}^{(1)} & 0 & \cdots & 0 \\ 0 & M_{h_{\mathbf{l}_2}}^{(1)} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & M_{h_{\mathbf{l}_{|\mathbb{L}_n|}}}^{(1)} \end{pmatrix}. \tag{5.36}$$

$$M^{(2)}_{\mathbb{L}_n} = \begin{pmatrix} c_{\mathbf{l}_1} h^{-1}_{\mathbf{l}_1} M^{(2)}_{h_{\mathbf{l}_1}, h_{\mathbf{l}_1}} & c_{\mathbf{l}_2} h^{-1}_{\mathbf{l}_1} M^{(2)}_{h_{\mathbf{l}_1}, h_{\mathbf{l}_2}} & \cdots & c_{\mathbf{l}_{|\mathbb{L}_n|}} h^{-1}_{\mathbf{l}_1} M^{(2)}_{h_{\mathbf{l}_1}, h_{\mathbf{l}_{|\mathbb{L}_n|}}} \\ c_{\mathbf{l}_1} h^{-1}_{\mathbf{l}_2} M^{(2)}_{h_{\mathbf{l}_2}, h_{\mathbf{l}_1}} & c_{\mathbf{l}_2} h^{-1}_{\mathbf{l}_2} M^{(2)}_{h_{\mathbf{l}_2}, h_{\mathbf{l}_2}} & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ c_{\mathbf{l}_1} h^{-1}_{\mathbf{l}_{|\mathbb{L}_n|}} M^{(2)}_{h_{\mathbf{l}_{|\mathbb{L}_n|}}, h_{\mathbf{l}_1}} & \cdots & c_{\mathbf{l}_{|\mathbb{L}_n|-1}} h^{-1}_{\mathbf{l}_{|\mathbb{L}_n|}} M^{(2)}_{h_{\mathbf{l}_{|\mathbb{L}_n|}}, h_{\mathbf{l}_{|\mathbb{L}_n|-1}}} & c_{\mathbf{l}_{|\mathbb{L}_n|}} h^{-1}_{\mathbf{l}_{|\mathbb{L}_n|}} M^{(2)}_{h_{\mathbf{l}_{|\mathbb{L}_n|}}, h_{\mathbf{l}_{|\mathbb{L}_n|}}} \end{pmatrix}. \tag{5.37}$$

Two matrices are then constructed from these matrices according to:

$$M_{\mathbb{L}_n} = M^{(1)}_{\mathbb{L}_n} + \theta M^{(2)}_{\mathbb{L}_n}, \quad M^{\star}_{\mathbb{L}_n} = M^{(1)}_{\mathbb{L}_n} - (1-\theta) M^{(2)}_{\mathbb{L}_n}, \tag{5.38}$$

which are square and of order:

$$Order(M_{\mathbb{L}_n}) = Order(M^{\star}_{\mathbb{L}_n}) = \sum_{r=0}^{d-1} 2^{n+d-1-r} \binom{n+d-2-r}{d-1}. \tag{5.39}$$

Note that the global matrices $M_{\mathbb{L}_n}$ and $M^{\star}_{\mathbb{L}_n}$ are not symmetric because the global stiffness matrix $M^{(2)}_{\mathbb{L}_n}$ is not. The system can thus be rewritten as a linear system:

$$M_{\mathbb{L}_n} \Phi^{k+1}_{\mathbb{L}_n} = \frac{\Delta t}{\epsilon_0} \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}^{k}_{\mathbb{L}_n} + M^{\star}_{\mathbb{L}_n} \Phi^{k}_{\mathbb{L}_n}. \tag{5.40}$$

In order to solve the linear system (5.40) and obtain the updated electric potential at time $k+1$, the local stiffness matrices (5.34) need to be computed at each time step because of their dependence on the position of the particles. The resulting global stiffness matrix (and global matrix $M_{\mathbb{L}_n}$) is then different at each iteration.

The main steps of the method are summarized in the algorithm 11.

**Extension to the offset combination technique and standard grid**

In this section, the multi-dimensional electrostatic method is extended to the offset combination technique and also to standard grids. The offset combination, introduced in [48], consists in both reducing the number of sub-grids considered within the combination and using sub-grids with increased minimum levels. Let $\tau_0, \tau_1 \in \mathbb{N}$ be the indexes of the offset combination technique. The set of component grid is then parametrized by the set:

$$\mathbb{L}_n(\tau_0, \tau_1) := \bigcup_{i \in [\![0, d-1]\!]} \mathbb{L}_{n,i}(\tau_0, \tau_1), \quad \tau_0 \in [\![1, n]\!],\ \tau_1 \in [\![d-1, (d-1)\tau_0]\!] \tag{5.41}$$

$$\mathbb{L}_{n,i}(\tau_0, \tau_1) := \left\{ \mathbf{l} \in \mathbb{N}^d \mid \mathbf{l} \geq \tau_0,\ |\mathbf{l}|_1 = n + \tau_1 - i \right\}. \tag{5.42}$$

If $\tau_0 = n$ and $\tau_1 = (d-1)\tau_0$, then the set of component grids is constituted of only the Cartesian grid:

$$\mathbb{L}_n(n, (d-1)n) = \{(n, ..., n)\}. \tag{5.43}$$

Substituting $\mathbb{L}_n$ by $\mathbb{L}_n(\tau_0, \tau_1)$ in the last section, defines the method for the offset combination and also for the standard grid.

**Algorithm 11** Imp-PIC-Sg scheme

**Require:** Particle positions and velocities at previous time $(\mathbf{x}_p^{k-1/2}, \mathbf{v}_p^k)$, time step $\Delta t$, set of component grid $\Omega_{h_\mathbf{l}}, \mathbf{l} \in \mathbb{L}_n$.

**for each** time step $k\Delta t$ **do**

**Advance** the particle positions in time with an explicit contribution of the particle velocity:

$$\mathbf{x}_p^{k+1/2} = \mathbf{x}_p^{k-1/2} + \Delta t \mathbf{v}_p^k.$$

**for each** component grid of level index $\mathbf{l} \in \mathbb{L}_n$ **do**

**Accumulate** the explicit contribution of the current density onto the component grid:

$$\mathbf{J}^k_{h_\mathbf{l};\mathbf{j}^{xy}} = h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} q_p \mathbf{v}_p^k W_{h_\mathbf{l};p\mathbf{j}^{xy}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}, \quad \forall \mathbf{l} \in \mathbb{L}_n,$$

where $W_{h_\mathbf{l};p\mathbf{j}^{xy}} = W_{h_\mathbf{l};\mathbf{j}^{xy}}\left(\mathbf{x}_p^{k+1/2}\right)$ is defined on the staggered component grids $\Omega^{xy}_{h_\mathbf{l}}$.

**Compute** the local Laplacian matrix of the component grid $M^{(1)}_{h_\mathbf{l}}$:

$$M^{(1)}_{h_\mathbf{l}} = \left(\Delta_{h_\mathbf{l}} W_{h_\mathbf{l};\mathbf{j}\mathbf{j}'}\right)_{\mathbf{j}\mathbf{j}'} \quad \text{where } (\mathbf{j}, \mathbf{j}') \in I_{h_\mathbf{l}} \times I_{h_\mathbf{l}}.$$

**for each** component grid of level index $\tilde{\mathbf{l}} \in \mathbb{L}_n$ **do**

**Compute** the local stiffness matrices of the pair of component grids $M^{(2)}_{h_\mathbf{l},h_{\tilde{\mathbf{l}}}}$.

$$M^{(2)}_{h_\mathbf{l},h_{\tilde{\mathbf{l}}}} = \left( \sum_s \sum_{p=1}^{N_s} \beta_p \sum_{\tilde{\mathbf{j}} \in I_{h_{\tilde{\mathbf{l}}}}} W_{h_{\tilde{\mathbf{l}}};p\tilde{\mathbf{j}}^{xy}} \boldsymbol{\nabla}_{h_{\tilde{\mathbf{l}}}} W_{h_{\tilde{\mathbf{l}}};\tilde{\mathbf{j}}\mathbf{j}'} \cdot \boldsymbol{\nabla}_{h_\mathbf{l}} W_{h_\mathbf{l};p\mathbf{j}^{xy}} \right)_{\mathbf{j}\mathbf{j}'} \quad \text{where } (\mathbf{j}, \mathbf{j}') \in I_{h_\mathbf{l}} \times I_{h_{\tilde{\mathbf{l}}}}.$$

**end for**

**end for**

**Assemble** the global stiffness matrices $M_{\mathbb{L}_n}$, $M^\star_{\mathbb{L}_n}$ according to equations (5.36)-(5.38).

**Solve** the linear system associated to the global stiffness matrices:

$$M_{\mathbb{L}_n} \Phi^{k+1}_{\mathbb{L}_n} = \frac{\Delta t}{\epsilon_0} \boldsymbol{\nabla}_{h_\mathbf{l}} \cdot \mathbf{J}^k_{\mathbb{L}_n} + M^\star_{\mathbb{L}_n} \Phi^k_{\mathbb{L}_n}.$$

**for each** component grid of level index $\mathbf{l} \in \mathbb{L}_n$ **do**

**Compute** the electric field on the component grid from the electric potential:

$$\mathbf{E}^{k+1}_{h_\mathbf{l};\mathbf{j}^{xy}} = -\boldsymbol{\nabla}_{h_\mathbf{l}} \Phi^{k+1}_{h_\mathbf{l};\mathbf{j}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}, \quad \forall \mathbf{l} \in \mathbb{L}_n.$$

**end for**

**Interpolate** the averaged in time contribution of the electric field at the particle positions with the combination technique:

$$\mathbf{E}^{k+1/2}_{h_n}\left(\mathbf{x}_p^{k+1/2}\right) = \sum_{\mathbf{l} \in \mathbb{L}_n} c_\mathbf{l} \sum_{\tilde{\mathbf{j}} \in I_{h_{\tilde{\mathbf{l}}}}} \mathbf{E}^{k+1/2}_{h_{\tilde{\mathbf{l}}};\tilde{\mathbf{j}}^{xy}} W_{h_{\tilde{\mathbf{l}}};p\tilde{\mathbf{j}}^{xy}}, \qquad \text{where } \mathbf{E}^{k+1/2}_{h_\mathbf{l};\mathbf{j}^{xy}} = \frac{\mathbf{E}^{k+1}_{h_\mathbf{l};\mathbf{j}^{xy}} + \mathbf{E}^k_{h_\mathbf{l};\mathbf{j}^{xy}}}{2}.$$

**Advance** the particle velocities in time with the implicit contribution of the electric field:

$$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \frac{q_p \Delta t}{m_p} \mathbf{E}^{k+1/2}_{h_n}\left(\mathbf{x}_p^{k+1/2}\right).$$

**end for**

**Differences with the traditional ECSIM scheme**

Although the method presented is strongly inspired from the ECSIM scheme developed in [83] for standard PIC methods and Cartesian grids, there are some major differences between the two approaches. Let us emphasize these differences and give some hints on their impact on the computational costs and memory footprint of the methods.

The first difference between the traditional ECSIM scheme [83] (as well as its extensions [33, 25]) and the method introduced in this section is the electrostatic regime. As a result, the Ampere equation is substituted by the divergence of the Ampere equation. The resulting electrostatic stiffness matrices contain terms depending on the product of the basis functions gradients instead of products of the basis functions theirselves (for the electromagnetic mass matrices). Therefore the stiffness matrices have more non-zeros entries for the electrostatic method than for the traditional electromagnetic method. Considering a standard grid approach, the number of non-zero terms per row of the stiffness matrix is 21 for the electrostatic method whereas it is 9 for the electromagnetic method. In addition, there is no self-consistent magnetic field in our approach and therefore the rotation matrix used in the traditional ECSIM scheme does not have to be computed. It results in an unique stiffness matrix for each time step, instead of 9 mass matrices for the ECSIM scheme. In addition, the unknown for the electrostatic approach is scalar, so that the size of the linear system is reduced by six in comparison to the ECSIM scheme ($\Phi$ versus $E_x, E_y, E_z, B_x, B_y, Bz$). Note also that, contrary to the electrostatic approach, the condition $\nabla \times \mathbf{E}$ is not verified in the ECSIM scheme.

The second major difference is related to the embedding of sparse grid reconstructions. It results in the computation of numerous blocks (local stiffness matrices) to assemble the system matrix. In the standard ECSIM scheme derived for an electrostatic regime, an unique stiffness matrix shall be computed. The number of component grids and thus of stiffness matrices to compute for the sparse grid method depends on the dimension and the grid discretization, scaling as $O\left(|\log h_n|^{d-1}\right)$. The determination of a stiffness matrix can be expansive because of the computation of the particle interactions requiring a loop spanning the particle population. Nonetheless, the local stiffness matrices are less expensive to compute for sparse grid reconstructions since the number of particles is significantly reduced and, the array of a local stiffness matrix fits more easily in the cache memory of the CPU than the global stiffness matrix for standard grids. As demonstrated in [49], a good cache memory management is a capital feature increasing the performance of these kind of algorithms and enables an efficient parallelization.

An another significant difference between the standard approach and the sparse grid one is the profile of the global stiffness matrix. Indeed, one major advantage of the ECSIM scheme is the sparsity of the stiffness matrix. Because of the very localized support of the basis functions (hat function with one grid cell width support), the number of non-zero terms of the stiffness matrix is 21 per row (9 for the traditional electromagnetic method). The total number of non-zero terms in the global matrix is:

$$non\ zero\ terms = 21 * 2^{dn}. \tag{5.44}$$

Conversely, the matrix profile of the sparse grid reconstruction scheme is a bit more complex. The global matrix is constructed by assembling blocks of local matrices, each corresponding to a couple of component grids. For couples of the same component grids (corresponding to the diagonal blocks in the global matrix), the profile of the local stiffness matrix is similar to the standard matrix one with the same sparsity (only 21 non-zero terms per row). On the contrary, for couples of component grids with different discretizations, the profile may not be sparse at all. *E.g.* consider a couple of grids with levels $\mathbf{l}_1 = (1, n)$ and $\mathbf{l}_2 = (n, 1)$, for all grid nodes

$\mathbf{j}_1 \in I_{h_{\mathbf{l}_1}}$ and $\mathbf{j}_2 \in I_{h_{\mathbf{l}_2}}$ the support of the basis functions is not disjoint (see figure 5.1):

$$Supp(W_{h_{\mathbf{l}_1},\mathbf{j}_1}) \cap Supp(W_{h_{\mathbf{l}_2},\mathbf{j}_2}) \neq \emptyset. \tag{5.45}$$

In that configuration, all the entries in the local stiffness matrix $M^{(2)}_{h_{\mathbf{l}_1},h_{\mathbf{l}_2}}$ are non-zeros. Nonethe-

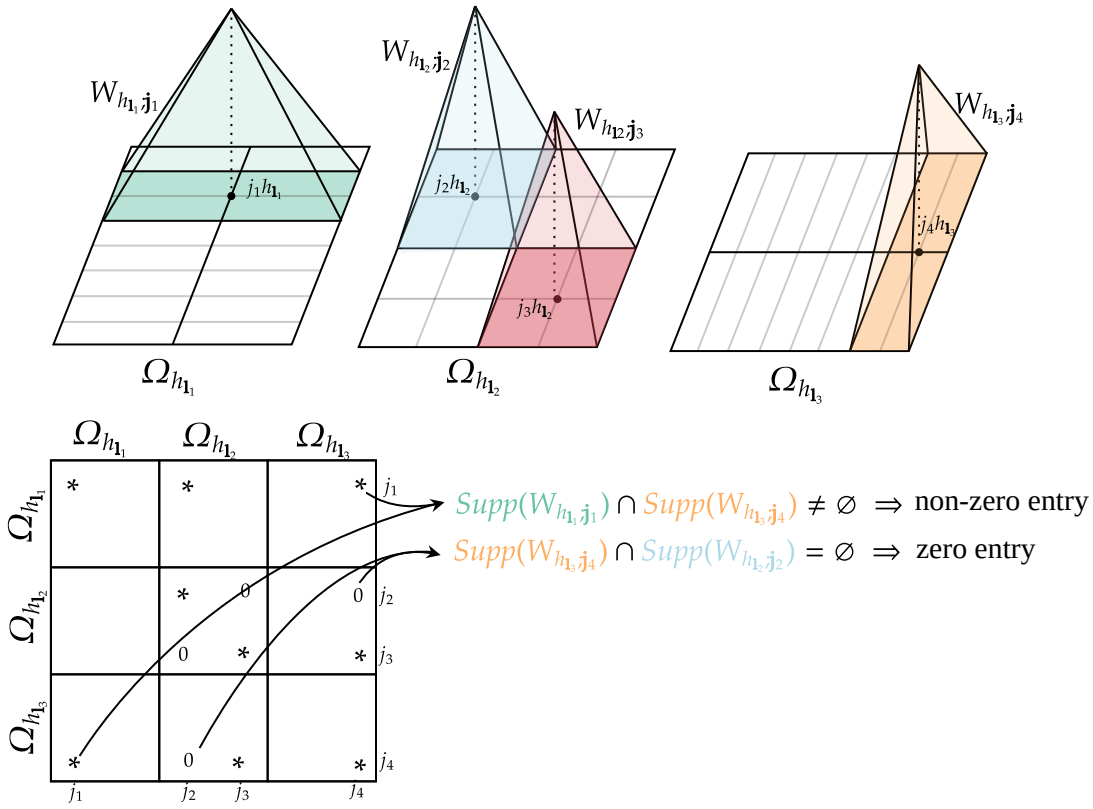


**Figure 5.1.** Support of basis functions for component grids and corresponding non-zero and zero entries in the stiffness matrix.

less, thanks to the sparse grid properties, the size of the linear system, *i.e.* the order of the global matrix, is reduced in comparison to the standard approach:

$$Order(M_{\mathbb{L}_n}) = \begin{cases} \displaystyle\sum_{r=0}^{d-1} 2^{n+d-1-r} \binom{n+d-2-r}{d-1} & \text{(sparse grid)}, \\ 2^{dn} & \text{(standard grid)}. \end{cases} \tag{5.46}$$

*E.g.* for three dimensional computations, the orders of the global matrices fall down to:

$$Order(M_{\mathbb{L}_n}) = \begin{cases} 2^n(4n^2 - 2n + 2) & \text{(sparse grid)}, \\ 2^{3n} & \text{(standard grid)}. \end{cases} \tag{5.47}$$

## 5.2 Properties of the scheme

The properties of the sparse grid reconstructions for the electrostatic and the electromagnetic ECSIM schemes are now investigated with respect to their conservation and stability properties. All the properties demonstrated in this section are valid for the schemes embedding sparse grids method with the classical and the offset combination techniques, as well as the standard scheme (embedding a Cartesian uniform grid).

Total energy conservation is crucial for stability in PIC schemes in order to avoid self-heating or self-cooling of the plasma that may lead to nonphysical behavior of the simulation. The exact discrete energy conservation is the main benefit of the ECSIM approach and our goal is to preserve this property. Exact energy conservation means that the amount of energy transferred between the particles and the fields is equivalent. In sparse grid reconstruction methods, the field equations are discretized on the set of component grids. Therefore, in order to quantify the loss in total energy, one shall define an interpolant of the electric field energy from the component grids. The electric field energy at time $k$ on the mesh is defined by the combination of the electric field energy from all the component grids:

$$\mathcal{E}_{\mathcal{F}}^{k} := \frac{\epsilon_0}{2} \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{\mathbf{l}}^{-1} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \left(\mathbf{E}_{h_{\mathbf{l}};\mathbf{j}}^{k}\right)^2. \tag{5.48}$$

**Remark 5.2.1** *The electric field energy reconstructed from the component grid $\mathcal{E}_{\mathcal{F}}$ is not non-negative (see remark 2.1.4).*

**Proposition 5.2.2 (Exact energy conservation)** *The total energy of the system is exactly conserved for $\theta = \frac{1}{2}$, i.e.*

$$\mathcal{E}_{\mathcal{K}}^{k+1} - \mathcal{E}_{\mathcal{K}}^{k} = -\left(\mathcal{E}_{\mathcal{F}}^{k+1} - \mathcal{E}_{\mathcal{F}}^{k}\right), \tag{5.49}$$

*where $\mathcal{E}_{\mathcal{F}}^{k}$ the electric field energy is defined in equation (5.48) and $\mathcal{E}_{\mathcal{K}}^{k}$ the kinetic energy is defined by:*

$$\mathcal{E}_{\mathcal{K}}^{k} := \frac{1}{2} \sum_{s} \sum_{p=1}^{N_s} m_p \left(\boldsymbol{v}_p^k\right)^2. \tag{5.50}$$

The particle mover is similar to the one introduced in [83]. It has the same accuracy than the standard leapfrog mover of explicit schemes, *i.e.* second order in time, and is linearly stable.

**Proposition 5.2.3 (Accuracy in time)** *The particle mover (5.19) is second order accurate for $\theta = \frac{1}{2}$. Let $\boldsymbol{x}$, $\boldsymbol{v}$ and $\boldsymbol{E}$ be solutions to the system of ODEs:*

$$\begin{cases} \dfrac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{v}(t), \\ \dfrac{d\boldsymbol{v}(t)}{dt} = \dfrac{q}{m}\boldsymbol{E}(\boldsymbol{x}(t), t), \end{cases} \tag{5.51}$$

*then we have:*

$$\boldsymbol{x}\left(t + \Delta t/2\right) = \boldsymbol{x}\left(t - \Delta t/2\right) + \Delta t \boldsymbol{v}(t) + O\left(\Delta t^2\right) \tag{5.52}$$

$$\boldsymbol{v}(t + \Delta t) = \boldsymbol{v}(t) + \frac{q\Delta t}{m}\left(\boldsymbol{E}\left(\boldsymbol{x}\left(t + \frac{\Delta t}{2}\right), t + \Delta t\right) + \boldsymbol{E}\left(\boldsymbol{x}\left(t + \frac{\Delta t}{2}\right), t\right)\right) + O\left(\Delta t^2\right) \tag{5.53}$$

**Proposition 5.2.4 (Linear stability)** *The dispersion relation of the schemes with $\theta = \frac{1}{2}$, resulting from the analysis of linear stability in time for a cold plasma with no drift, is :*

$$\tan(\omega\Delta t/2)\sin(\omega\Delta t/2) = \left(\omega_p \Delta t/2\right)^2, \quad \text{for } \Im(\omega) > 0. \tag{5.54}$$

*For all $\Delta t$, the roots are real and the scheme is stable.*

## 5.3 Semi-implicit scheme on hybrid grids

In this section, an alternative sparse PIC semi-implicit scheme is proposed. The idea is similar to the explicit PIC-Hg scheme based on the hybrid discretization in space (made of the component grids and the Cartesian grid). Recalling the discussions in chapter 2, the motivation for the hybrid discretization of space in the explicit approach was to mitigate the grid-based error deterioration due to the resolution of the Poisson equation on the component grids. The motivation here for introducing a hybrid discretization is different. Indeed, recalling the remark 5.2.1, the field energy conserved within the semi-implicit scheme is not a nonnegative quantity and thus the exact conservation does not ensure stability for the scheme, as we will see in the numerical applications. The goal is to design a sparse PIC semi-implicit method that is stable for all time and space discretizations. Nonetheless, because of the introduction of the hybrid discretization, the total energy exact conservation property of the scheme shall be lost.

Let us introduce the main steps of the scheme consisting of seven steps, summarized in the algorithm 12.

# Proof of the chapter 5

*Proof of lemma 5.1.2.* The result is obtained thanks to the periodicity of the component grids. □

*Proof of proposition 5.2.2.* The difference in the particle kinetic energy between two consecutive steps $k$ and $k+1$ is expressed by:

$$\mathcal{E}_{\mathcal{K}}^{k+1} - \mathcal{E}_{\mathcal{K}}^{k} = \sum_s \sum_{p=1}^{N_s} \frac{1}{2} m_p \left[ \left(\mathbf{v}_p^{k+1}\right)^2 - \left(\mathbf{v}_p^{k}\right)^2 \right] \tag{5.55}$$

$$= \Delta t \sum_s \sum_{p=1}^{N_s} q_p \mathbf{v}_p^{k+1/2} \mathbf{E}_{h_n}^{k+1/2}(\mathbf{x}_p^{k+1/2}) \tag{5.56}$$

$$= \Delta t \sum_s \sum_{p=1}^{N_s} q_p \mathbf{v}_p^{k+1/2} \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2,xy} W_{h_{\mathbf{l}};p\mathbf{j}}^{xy} \tag{5.57}$$

$$= \Delta t \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{\mathbf{l}}^{-1} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2,xy} \cdot \mathbf{J}_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2,xy}. \tag{5.58}$$

$$= -\Delta t \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{\mathbf{l}}^{-1} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \boldsymbol{\nabla}_{h_{\mathbf{l}}} \Phi_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2} \cdot \mathbf{J}_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2,xy}. \tag{5.59}$$

In the above, the particle velocity equation (5.19) has been used, as well as the definition of the recombined electric field (2.134) and the current density (2.121). The last equation is obtained

**Algorithm 12** Imp-PIC-Hg scheme

**Require:** Particle positions and velocities at previous time $(\mathbf{x}_p^{k-1/2}, \mathbf{v}_p^k)$, time step $\Delta t$, set of component grid $\Omega_{h_\mathbf{l}}$, $\mathbf{l} \in \mathbb{L}_n$ and the Cartesian, written as $\Omega_{h_\mathbf{l}}$, $\mathbf{l} \in \mathbb{L}_n^* = \mathbb{L}_n(n, (d-1)n)$.

**for each** time step $k\Delta t$ **do**

**Advance** the particle positions in time with an explicit contribution of the particle velocity:

$$\mathbf{x}_p^{k+1/2} = \mathbf{x}_p^{k-1/2} + \Delta t \mathbf{v}_p^k.$$

**for each** component grid of level index $\mathbf{l} \in \mathbb{L}_n$ **do**

**Accumulate** the explicit contribution of the current density onto the component grid:

$$\mathbf{J}^k_{h_\mathbf{l};\mathbf{j}^{xy}} = h_\mathbf{l}^{-1} \sum_s \sum_{p=1}^{N_s} q_p \mathbf{v}_p^k W_{h_\mathbf{l};p\mathbf{j}^{xy}}, \quad \forall \mathbf{j} \in I_{h_\mathbf{l}}, \quad \forall \mathbf{l} \in \mathbb{L}_n,$$

where $W_{h_\mathbf{l};p\mathbf{j}^{xy}} = W_{h_\mathbf{l};\mathbf{j}^{xy}}\left(\mathbf{x}_p^{k+1/2}\right)$ is defined on the staggered component grids $\Omega^{xy}_{h_\mathbf{l}}$.

**end for**

**Interpolate** the explicit contribution of the current density on the Cartesian grid from the component grids with the combination technique:

$$\mathbf{J}^{k,c}_{h_n;\mathbf{i}^{xy}} = \sum_{\mathbf{l}\in\mathbb{L}_n} c_\mathbf{l} \sum_{\mathbf{j}\in I_{h_\mathbf{l}}} \mathbf{J}^k_{h_\mathbf{l};\mathbf{j}^{xy}} W_{h_\mathbf{l};\mathbf{j}^{xy}\mathbf{i}^{xy}}, \qquad \forall \mathbf{i} \in I_{h_n}.$$

**Compute** the global stiffness matrices $M_{\mathbb{L}_n^*}$, $M^\star_{\mathbb{L}_n^*}$ according to equations (5.33),(5.34),(5.36)-(5.38).

**Solve** the linear system associated to the global stiffness matrices:

$$M_{\mathbb{L}_n^*} \Phi^{k+1}_{\mathbb{L}_n^*} = \frac{\Delta t}{\epsilon_0} \boldsymbol{\nabla}_{h_n} \cdot \mathbf{J}^k_{\mathbb{L}_n^*} + M^\star_{\mathbb{L}_n^*} \Phi^k_{\mathbb{L}_n^*}.$$

**Compute** the electric field on the Cartesian grid from the electric potential:

$$\mathbf{E}^{k+1}_{h_n;\mathbf{i}^{xy}} = -\boldsymbol{\nabla}_{h_n} \Phi^{k+1}_{h_n;\mathbf{i}}, \quad \forall \mathbf{i} \in I_{h_n}.$$

**Interpolate** the averaged in time contribution of the electric field at the particle positions:

$$\mathbf{E}^{k+1/2}_{h_n}\left(\mathbf{x}_p^{k+1/2}\right) = \sum_{\mathbf{i}\in I_{h_n}} \mathbf{E}^{k+1/2}_{h_n;\mathbf{i}^{xy}} W_{h_n;p\mathbf{i}^{xy}}, \qquad \text{where } \mathbf{E}^{k+1/2}_{h_n;\mathbf{i}^{xy}} = \frac{\mathbf{E}^{k+1}_{h_n;\mathbf{i}^{xy}} + \mathbf{E}^k_{h_n;\mathbf{i}^{xy}}}{2}.$$

**Advance** the particle velocities in time with the implicit contribution of the electric field:

$$\mathbf{v}_p^{k+1} = \mathbf{v}_p^k + \frac{q_p \Delta t}{m_p} \mathbf{E}^{k+1/2}_{h_n}\left(\mathbf{x}_p^{k+1/2}\right).$$

**end for**

with the equation (5.30). From lemma 5.1.2, a discrete integration by parts on (5.59) yields:

$$\mathcal{E}_{\mathcal{K}}^{k+1} - \mathcal{E}_{\mathcal{K}}^{k} = \Delta t \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{\mathbf{l}}^{-1} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \Phi_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2} \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \mathbf{J}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k+1/2}. \tag{5.60}$$

Then, using the divergence of Ampere equation (5.22), the difference in the kinetic energy is equal to:

$$\mathcal{E}_{\mathcal{K}}^{k+1} - \mathcal{E}_{\mathcal{K}}^{k} = -\epsilon_0 \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{\mathbf{l}}^{-1} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \Phi_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2} \boldsymbol{\nabla}_{h_{\mathbf{l}}} \cdot \left( \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k+1} - \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k} \right). \tag{5.61}$$

Eventually, a discrete integration by parts (see lemma 5.1.2) gives the result:

$$\mathcal{E}_{\mathcal{K}}^{k+1} - \mathcal{E}_{\mathcal{K}}^{k} = \epsilon_0 \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{\mathbf{l}}^{-1} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \boldsymbol{\nabla}_{h_{\mathbf{l}}} \Phi_{h_{\mathbf{l}};\mathbf{j}}^{k+1/2} \cdot \left( \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k+1} - \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k} \right) \tag{5.62}$$

$$= -\epsilon_0 \sum_{\mathbf{l}\in\mathbb{L}_n} c_{\mathbf{l}} h_{\mathbf{l}}^{-1} \sum_{\mathbf{j}\in I_{h_{\mathbf{l}}}} \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k+1/2} \cdot \left( \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k+1} - \mathbf{E}_{h_{\mathbf{l}};\mathbf{j}^{xy}}^{k} \right) \tag{5.63}$$

$$= -\left( \mathcal{E}_{\mathcal{F}}^{k+1} - \mathcal{E}_{\mathcal{F}}^{k} \right). \ \square \tag{5.64}$$

*Proof of proposition 5.2.3.* The result is obtained with Taylor expansions of $\mathbf{x}$ and $\mathbf{v}$. $\square$

*Proof of proposition 5.2.4.* A proof is provided in [83].

# Chapter 6

# Numerical applications

This last chapter is dedicated to the numerical investigation of the methods introduced in the previous chapters. The motivation is to assess numerically the accuracy and the efficiency of the sparse PIC methods for two-dimensional and three-dimensional geometries and to compare the results with the well-established standard PIC method.

As a preliminary section for the chapter, the general configurations used for the simulations are introduced: *e.g.* the test cases, the hardware configurations, etc.

First, numerical experiments are performed on various classical test cases in two-dimensions, consolidating the results of the formal analyses and illustrating conclusively the gain brought by sparse grid reconstructions to improve the sampling error without introducing a significant numerical diffusion. In this first part, the results presented are obtained with a two dimensional straightforward implementation of the methods presented in chapters 2 and 5, without any of the optimizations presented in chapter 3. Despite the simplicity of the implementation and the restriction to two dimensional geometries, the proposed numerical investigations provide a glimpse of the sparse PIC method efficiency. However the full potential of the method can only be achieved by three dimensional computations. This is strikingly illustrated by the plots of figures 1.2 and 4.2 relating the data required in either a two-dimensional or a three-dimensional computation to guarantee a similar statistical noise in both the regular and the sparse grid PIC methods. It is manifest that the reduction of particles with the sparse grid schemes is significantly larger for three dimensional computations: for grid with more than five hundred cells (in each dimension), the number of particles run in the standard method for two dimensional computations is larger than that of sparse grid methods for three dimensional computations. Though these projections may be mitigated by implementation issues, the perspectives offered by the sparse grid reconstruction promise a leap forward in the efficiency of PIC numerical methods for three dimensional computations.

Secondly, numerical investigations upon the efficiency and the parallelization of the method are conducted on three-dimensional classical test cases. The results presented are obtained with a three-dimensional implementation incorporating the optimizations introduced in chapter 3 and 4. The motivation is to assess the gains (in term of computational time), promised by the two-dimensional investigation, of the sparse-PIC methods over the standard PIC method. The better control of the statistical error offered by sparse-PIC reconstructions permits a substantial reduction of the number of particles (up to 2 orders of magnitude). Conversely, the interactions of one particle (restricted to the density projection for the algorithm implementing the hierarchization), shall be computed for tens to more than one hundred component grids with a very small number of nodes. Compared to standard PIC methods these features bring significant changes in the most consuming steps of the algorithm. The Poisson problem is discretized on each of the component grids issuing linear systems of tiny sizes which dramatically reduces the memory footprint as well as the computational effort attached to the electric potential approximation. The reduced number of particles, in addition to reducing massively the memory consumption,

lowers the computational cost of the particle pusher to a marginal contribution. In the end, the majority of the computations are dedicated to the projection of the particle properties onto the component grids, these operations being arithmetically intensive. The genuine parallelism of sparse grid reconstructions permits an implementation with a scalability close to optimal (the efficiency reaches 126 on 128 cores) providing speed-ups of the global algorithms up to 100 on 128 cores (on a NUMA architecture platform). On the other hand, the sparse-PIC GPU implementation achieves speed-ups close to 100 on a single GPU (Tesla V100) for three-dimensional PIC simulations, resulting in a computational time diminution of four orders of magnitude with respect to a single core CPU standard PIC execution.

## Contents



# 6.1 General settings

The space domain is a periodic cube $\Omega_x = (\mathbb{R}/L\mathbb{Z})^d$, of dimension $L$. The Debye length and the inverse plasma period are defined by $\lambda_D = \sqrt{\varepsilon_0 T_e/q_e n_0}$ and $\omega_p^{-1} = 1/\sqrt{q_e n_0/m_e\varepsilon_0}$.

Excepted for the section 6.2.1, where the electron charge, mass and temperature are $q_e = 1.602 \times 10^{-19}$ C, $m_e = 9.109 \times 10^{-31}$ kg, $T_e = 1$ eV, dimensionless variables are considered, the reference length being the Debye length and time units being the plasma period. Electron mass, temperature and charge are normalized to one and $\varepsilon_0 = \sqrt{1/4\pi}$.

Throughout this chapter we will refer to the mean number of particle per cell, denoted $P_c$, relating the amount of statistical noise in the simulation and introduced in [97]. For the standard method, it depends on the number of particles ($N$) and the Cartesian grid discretization ($h_n$):

$$P_c = \frac{NL^d}{h_n^d} = N2^{-dn}. \tag{6.1}$$

Note that the grid discretization depends now on the domain size ($h_n = 2^{-n}L$). An equivalent quantity for sparse-PIC methods can be defined by considering the number of particles for all component grid cells, *i.e.*:

$$P_c = N\left(\sum_{\mathbf{l}\in\mathbb{L}_n} |c_{\mathbf{l}}|\Delta h_{\mathbf{l}}\right)^{-1}, \tag{6.2}$$

which falls down to the following relations in 2-dimensional and 3-dimensional geometries:

$$P_c = N2^{-n}\left(3n-1\right)^{-1}, \quad P_c = N2^{-n}\left(\frac{9}{2}n^2 - \frac{3}{2}n + 1\right)^{-1}. \tag{6.3}$$

For the offset combination technique, the relation is obtained by taking $\mathbb{L}_n(\tau_0, \tau_1)$ in equation (6.2).

### 6.1.1 Test cases

**$1^{st}$ test case: Landau damping**

The first test case considered in this manuscript is the well-known Landau damping [1, 80]. When a plasma is slightly perturbed from an equilibrum state, it returns to its equilibrium with an exponential damping. For this test case, the electrons are immersed in a uniform, immobile, background of ions:

$$\rho_i = \frac{Q_e}{\int_{\Omega_x} d\mathbf{x}}. \tag{6.4}$$

A perturbation in the electron distribution of an equilibrium state is considered:

$$f_e(\mathbf{x}, \mathbf{v}, 0) = f_v^0(\mathbf{v}) f_x^0(\mathbf{x}), \tag{6.5}$$

where the initial velocity distribution is Maxwellian and the perturbation has the following form:

$$f_v^0(\mathbf{v}) = \left(\frac{1}{\sqrt{\pi v_{th,e}}}\right)^d e^{\frac{-\|\mathbf{v}\|_2^2}{v_{th,e}^2}}, \quad f_x^0(\mathbf{x}) = \prod_{i=1}^{d}\left(1 + \alpha_i \cos\left(k_i x_i\right)\right), \tag{6.6}$$

$v_{th,e} = \sqrt{2T_e q_e / me}$ is the thermal velocity of electrons and $\|\mathbf{v}\|_2^2 = \sum_{i=1}^{d} v_i^2$. $\boldsymbol{\alpha}$ is the magnitude and $\mathbf{k}$ is the period of the perturbation.

Two configurations of the perturbation are considered: a small perturbation regime and a large perturbation regime. In the small perturbation regime, the perturbation is considered uniform in each dimension, *i.e.* $k_i = k$, $k \in \mathbb{R}$ and the domain size depends on the perturbation:

$$L = \frac{2\pi}{k}. \tag{6.7}$$

By considering the roots of the dispersion function ($\varepsilon(\omega, k) = 0$), which is as follows:

$$\frac{1}{\varepsilon_0}\varepsilon(\omega, k) = 1 + \frac{1}{k^2}\left(1 + \frac{\omega}{\sqrt{2}k} Z\left(\frac{\omega}{\sqrt{2}k}\right)\right), \tag{6.8}$$

one can find the damping rate of the plasma ($\Im(\omega)$) for given values of $k \in \mathbb{R}$. *E.g.* for $k = 0.5$, the root with the largest imaginary part is $\omega = \pm 1.14156 - 0.1533i$; for $k = 0.3$, $\omega = \pm 1.1598 - 0.0126i$, etc. In the large perturbation regime, the perturbation is considered large enough to invalidate the linear dispersion theory.

### $2^{nd}$ test case: diocotron instability

In this test case, we consider a hollow profile in the electron distribution, confined by a uniform magnetic field $\mathbf{B}$ [92]. The electrons are immersed in a uniform, immobile, background of ions, with the following Maxwellian distribution of electrons :

$$f_e(\mathbf{x},\mathbf{v},0) = f_v^0(\mathbf{v})\frac{\gamma}{\sigma L(2\pi)^2}e^{-\frac{\left(\sqrt{(x_1-\frac{L}{2})^2+(x_2-\frac{L}{2})^2}-\frac{L}{4}\right)^2}{2(\sigma L)^2}}, \tag{6.9}$$

where $\gamma$ has to be chosen such that:

$$\iint_{\Omega\times\mathbb{R}^d} f(\mathbf{x},\mathbf{v})d\mathbf{x}d\mathbf{v} = 1. \tag{6.10}$$

$f_v^0$ is a Maxwellian velocity distribution and is defined in equation (6.6). The external magnetic field is considered uniform along the $z$-axis $\mathbf{B} = (0,0,B_z)$, with $B_z = 2.5\times10^{-5}$ T or $B_z = 15$ in dimensionless variables. It is supposed strong enough so that the electron dynamics is dominated by advection in the self-consistent field $\mathbf{E}\times\mathbf{B}$. The instability caused by the magnetic field deforms the initially axisymmetric electron density distribution, leading, in the nonlinear phase, to the formation of a discrete number of vortices. As we expect fine scale structure to form in the process, a high discretization in space is required to reproduce these structures. Again, this test case defines a demanding benchmark for sparse grid approximations very likely to outline the grid error introduced with these reconstructions.

### $3^{rd}$ test case: manufactured solutions

A 3-dimensional manufactured solution test case is considered, according to the framework introduced in [104], with both ion dynamics and electron dynamics. The ratio of mass, charge and temperature between ions and electrons are assumed to be one. Let the domain size be $L = 10$, the manufactured electron and ion distribution functions are then:

$$f_{e,M}(\mathbf{x},\mathbf{v},t) := f_v^0(\mathbf{v})\left(\frac{1}{1000} + P\sin(\pi t)\,e^{-\frac{\|\mathbf{x}-5\|_2^2}{8}}\left(x_1^2+x_2^2+x_3^2-10(x_1+x_2+x_3)+63\right)\right), \tag{6.11}$$

$$f_{i,M}(\mathbf{x},\mathbf{v},t) := \frac{f_v^0(\mathbf{v})}{1000}, \tag{6.12}$$

where $P = \dfrac{1}{12000}$ is a constant ensuring that $f_{e,M}(\mathbf{x},\mathbf{v},t)$ is non-negative. $f_v^0$ is defined in equation (6.6). The associated charge density and electric field are:

$$\rho_M = -P\sin(\pi t)\,e^{-\frac{\|\mathbf{x}-5\|_2^2}{8}}\left(x_1^2+x_2^2+x_3^2-10(x_1+x_2+x_3)+63\right), \tag{6.13}$$

$$\mathbf{E}_M = \begin{pmatrix} \frac{(x_1-5)}{4}P^\star\sin(\pi t)\,e^{-\frac{\|\mathbf{x}-5\|_2^2}{8}} \\ \frac{(x_2-5)}{4}P^\star\sin(\pi t)\,e^{-\frac{\|\mathbf{x}-5\|_2^2}{8}} \\ \frac{(x_3-5)}{4}P^\star\sin(\pi t)\,e^{-\frac{\|\mathbf{x}-5\|_2^2}{8}} \end{pmatrix}, \tag{6.14}$$

where $P^\star = \frac{4}{3000}$. The system associated with this manufactured solution is:

$$\frac{\partial f_e}{\partial t} + \mathbf{v}\cdot\nabla_x f_e + \frac{q_e}{m_e}\mathbf{E}\cdot\nabla_v f_e = S_e \tag{6.15}$$

$$\frac{\partial f_i}{\partial t} + \mathbf{v}\cdot\nabla_x f_i + \frac{q_i}{m_i}\mathbf{E}\cdot\nabla_v f_i = S_i \tag{6.16}$$

$$\nabla\cdot\mathbf{E} - \rho = 0, \tag{6.17}$$

where $S_e$ and $S_i$ are specific source terms defined by:

$$S_i = \frac{\partial f_{i,M}(\mathbf{x},\mathbf{v},t)}{\partial t} + \mathbf{v}\cdot\nabla_x f_{i,M} + \frac{q_e}{m_e}\mathbf{E}_M\cdot\nabla_v f_{i,M}, \tag{6.18}$$

$$S_e = \frac{\partial f_{e,M}(\mathbf{x},\mathbf{v},t)}{\partial t} + \mathbf{v}\cdot\nabla_x f_{e,M} + \frac{q_e}{m_e}\mathbf{E}_M\cdot\nabla_v f_{e,M}. \tag{6.19}$$

The sources terms added to the system lead to a modification of the algorithm. The particle weights are no longer constant throughout the simulation and must be updated at each iteration, from time $k\Delta t$ to $(k+1)\Delta t$:

$$\omega_{p,e}^{k+1} = \omega_{p,e}^{k} + \Delta t\frac{S_e\left((\mathbf{x}_p^{k+1}+\mathbf{x}_p^k)/2, \mathbf{v}_p^{k+1/2}, (k+1/2)\Delta t\right)}{f_v^0(\mathbf{v})}, \tag{6.20}$$

$$\omega_{p,i}^{k+1} = \omega_{p,i}^{k} + \Delta t\frac{S_i\left((\mathbf{x}_p^{k+1}+\mathbf{x}_p^k)/2, \mathbf{v}_p^{k+1/2}, (k+1/2)\Delta t\right)}{f_v^0(\mathbf{v})}, \tag{6.21}$$

where $\omega_{p,e}$ and $\omega_{p,i}$ are the weights of the $p^{th}$ electron and ion particle.

#### $4^{th}$ test case: two-streams instability

The two streams instability [17] consists of two opposing streams of charged particles. The electrons are immersed in a uniform, immobile, background of ions, with the following Maxwellian distribution of electrons :

$$f_e(\mathbf{x},\mathbf{v},0) = f_v^0(\mathbf{v})f_x^0(\mathbf{x}), \tag{6.22}$$

where the initial velocity distribution is Maxwellian and the perturbation has the following form:

$$f_v^0(\mathbf{v}) = \left(\frac{1}{\sqrt{\pi}v_{th,e}}\right)^d\left(e^{\frac{-\|\mathbf{v}-\mathbf{v}_0\|_2^2}{v_{th,e}^2}} + e^{\frac{-\|\mathbf{v}+\mathbf{v}_0\|_2^2}{v_{th,e}^2}}\right), \quad f_x^0(\mathbf{x}) = \prod_{i=1}^{d}\left(1+\alpha_i\cos\left(k_i x_i\right)\right), \tag{6.23}$$

$\alpha_i$ is the magnitude of the perturbation, $\mathbf{v}_0 = (v_0, 0, ...) \in \mathbb{R}^d$ the main velocity of the beams in opposite direction and the domain size is:

$$L = \frac{2\pi}{k}, \tag{6.24}$$

where $k_i = k \in \mathbb{R}$, for $i = 1, ..., d$. Depending on the values of $k$ and $v_0$, the configuration is stable or unstable. Indeed, when two streams move through each other so that one wavelength is traveled in one cycle of the plasma frequency, the perturbation of one stream is increased by the

other stream and the perturbation grows exponentially in time. The linear dispersion relation for this test case is:

$$\frac{1}{\varepsilon_0}\varepsilon(\omega, \mathbf{k}) = 1 - \frac{\omega_p^2}{(\omega - \mathbf{k}\cdot\mathbf{v}_0)^2} - \frac{\omega_p^2}{(\omega + \mathbf{k}\cdot\mathbf{v}_0)^2}. \tag{6.25}$$

The four roots of the linear dispersion relation are [18]:

$$\omega = \pm\left[k^2 v_0^2 + \omega_p^2 \pm \omega_p\left(4k^2 v_0^2 + \omega_p^2\right)^{\frac{1}{2}}\right]^{\frac{1}{2}}, \tag{6.26}$$

which can be imaginary, and lead to instability, for:

$$0 \leq \frac{k v_0}{\omega_p} \leq \sqrt{2}. \tag{6.27}$$

### 6.1.2 Hardware configurations

#### CPU platforms

To assess the performance of the implementations and parallelization strategies, three different CPU hardware are considered:

- The fist hardware is a laptop equipped with Intel® Core™ i9-10885H CPU with 8 cores @2.40 GHz sharing a L3 cache memory of 16MB. Random-Access Memory (RAM) size is 32GB. The memory is uniformly shared by all cores with two memory channels and a maximum memory bandwidth of 45.8GB/s. The cache memory is divided into a first level (L1 cache) dedicated for instructions (L1 I) and data (L1 D), both of 32KB, an intermediate level cache for instructions and data (L2 I+D) of 256KB, both three specific for each core; and a last level cache memory (L3 I+D) of 16MB shared by all the cores.
- The second hardware is a node of the supercomputer OLYMPE from CALMIP composed of two processors Intel® Skylake 6140 @2.3 GHz with 18 cores. Each socket has a RAM of 192GB. The memory architecture is uniform for the cores of one CPU with two controllers, six memory channels and a maximum memory bandwidth of 119.21GB/s per socket (NUMA domain). The relative memory latency between the different NUMA nodes is 21 in the Advanced Configuration and Power Interface System Locality Information Table (ACPI SLIT), that is to say the latency between cores from different NUMA nodes is 2.1 times higher than the latency between cores from the same node. Within each CPU, the cache memory is divided into a level L1 cache memories of 32KB dedicated for instructions (L1 I) and data (L1 D), an intermediate level cache for instructions and data (L2 I+D) of 1MB, the two first level of cache are specific to a core; the last level cache memory (L3 I+D) of 24.75MB is shared by all the cores.
- The last hardware consists of a single computational server equipped with 128 cores in two AMD EPYC™ 7713 *Milan* CPUs and RAM memory of 512GB. Each *Milan* socket has a maximum memory bandwidth of 190.73GB/s (4 controllers, each controller has 2 memory channels). The memory architecture is non-uniform within a CPU. It consists of four NUMA domains per socket, each containing sixteen cores and 64GB of memory. The relative memory latency is at best 12 between NUMA domains in the same socket and 32 between domains from different sockets (in the ACPI SLIT). Each NUMA domain contains two Core Complexes (CCX) of eight AMD Zen 3 cores @2.0 GHz (altogether 64 cores per socket). Within the CCX, the compute cores share a 32MB L3 cache, each core has

an optimized 32KB L1 write-back cache and private 512 KB Unified (Instruction/Data) L2 cache.

The compilers used for the three hardware are respectively GNU Fortran version 9.4.0, IFORT version 18.0.2 and GNU Fortran version 10.2.1 with options -fopenmp -cpp and optimizations -Ofast or -O3. Frequency boost and hyperthreading are disabled. The code is executed with PETSc [6, 5] library for Poisson solver, using BiConjugate Gradient Stabilized (BiCGSTAB) method with Geometric Algebraic MultiGrid (GAMG) preconditioner or MUMPS [3] library (sparse LU decomposition method).

**GPU configurations**

To assess the performance of the GPU implementation strategies, we consider two different platforms (see table 6.1) consisting of a single GPU and an associated host CPU:

- The first hardware is a laptop equipped with a Quadro T2000 with 32 (FP64) CUDA cores, at a base clock frequency of 1.575GHz. The size of the DRAM memory is 4.1 GB. The cache memory is divided into a first level (L1) of 64KB per SM and a second level cache (L2) of 1.024MB. The memory bandwidth between the device and its specific memory is 112.1GB/s. The GPU is associated to the Intel® Core™ i9-10885H CPU (host) with 8 cores @2.40 GHz .
- The second GPU considered belongs to a node of the supercomputer OLYMPE from CALMIP: the Tesla V100 with 2,560 (FP64) CUDA cores, and a base clock frequency of 1.245GHz. The size of the memory is 16GB. The cache memory is divided into a first level (L1) of 128KB per SM and a second level cache (L2) of 6MB. The memory bandwidth between the device and its specific memory is 897GB/s; the L2 cache memory bandwidth is 4.2TB/s and the L1 cache memory bandwidth is 14TB/s. It is associated to the Intel® Skylake CPU with 18 cores @2,3 GHz.

**Table 6.1.** Performance characteristics of the GPU hardware.

| GPU | FP64 cores | Instruction throughput (FP64) | DRAM bandwidth |
|---|---|---|---|
| Quadro T2000 | 32 | 103.7 GFLOP/s | 112.1 GB/s |
| Tesla V100 | 2560 | 7,800 GFLOP/s | 897 GB/s |

| GPU | L2-cache bandwidth | L1-cache bandwidth |
|---|---|---|
| Tesla V100 | 4.2 TB/s | 14 TB/s |

The speedups of the algorithms run on GPGPUs are measured against the most efficient sequential implementation run on one core of the CPU host (Intel® core i9 for the Quadro T2000 and the Intel® Skylake for the Tesla V100, alternatively one core of a bisocket AMD EPYC™ 7713*Milan* may be considered).

The first hardware operated with the compiler Nvidia nvhpc version 22.3 compiler and the flags *-fast -acc -ta=tesla* and CUDA Driver version 11.6. The second hardware operates with the nvhpc version 22.1 and the flags *-fast -acc -Minfo=all -ta=tesla -Mx,231,0x1*, the last flag allowing to correct bugs with atomic and reduction operations within the 22.1 version, and CUDA Driver version 11.5. The Poisson equation is solved with the CuSolver [91] library for the first hardware and either with CuSolver or AMGx [90] for the second one. We have noticed a bug in the Fortran random number generator "*random number()*", providing not enough randomness in the particle distribution when used along with the nvhpc compiler. Therefore,

the CUDA random number generator CuRand is considered in the following to initialize the particles.

## 6.2 Two-dimensional geometries

### 6.2.1 Accuracy of the explicit schemes

The objective of the present section is to investigate qualitatively the sparse-PIC methods introduced in this manuscript and compare them to the standard method. The numerical investigation is restricted to two-dimensional geometries. The efficiency and performance of the methods in term of computational time are not investigated in this section as it will be developed in the next sections.

A series of numerical tests is carried out in two dimensional geometries: a Landau damping in both the linear and the non-linear regimes as well as a diocotron instability. These tests are performed with the standard Particle-In-Cell scheme (PIC-Std), the sub-grid Particle-In-cell scheme (PIC-Sg), the hybrid grid Particle-In-cell scheme (PIC-Hg), the enhanced sub-grid Particle-In-Cell (PIC-ESg) scheme and the oversampled hybrid grid Particle-In-Cell scheme (PIC-OHg), respectively presented in algorithm 1, 2, 3 and sections 2.2.4, 2.2.4. The schemes are implemented either with the classical, when not specified, or the offset combination technique. The results are compared without any filtering methods for both sparse and standard schemes. All the simulations in this section are performed on the Intel® Core™ i9-10885H CPU with one core. At initialization, the particles are initialized according to the initial function distribution $f(\mathbf{x}, \mathbf{v}, 0)$ related to the test case configuration.

In any of the following test cases, the total charge of the density is exactly conserved (to machine precision $\approx 10^{-16}$) for the PIC-Std, PIC-Hg scheme and PIC-OHg schemes with any combination technique which confirms the propositions 2.2.3, 2.2.14. Recalling that within the PIC-Sg or the PIC-ESg scheme, because the density is never fully projected onto a grid since we only proceed a partial projection onto each sub-grid and combine the resulting partial electric fields, the conservation of density is therefore not assessed numerically for these schemes.

#### Linear Landau damping

The perturbation considered in the distribution of electrons has to be small enough so that a linear approximation is valid. Under this assumption, the electric field decreases exponentially fast in time according to a damping rate [80]. The motivation here is to recover this damping rate with the different schemes. Let us parametrize the perturbation with $\alpha_1 = \alpha_2 = 0.05$, $k$ such that $L = 22\lambda_D$ ($k \approx 0.286$), $\Delta t = 0.05\omega_p^{-1}$ and the final time $T = 25\omega_p^{-1}$. The grid discretization is chosen so that $h_n \simeq 0.34\lambda_D$, the configuration of the grid and particles is indicated in table 6.1. Interpolations in the combination technique are done with linear splines for all the methods. The numerical results are represented in figure 6.2 (a). Both the PIC-Hg, PIC-Sg, PIC-Std schemes agrees well with the damping rate. Besides, there are four times less total particles in the simulation for the sparse grid schemes.

**Table 6.1.** Linear Landau damping: Configuration of the methods.

| Scheme | Grid size $h_n$ | $P_c$ | $N$ | $N_{Std}/N$ |
|---|---|---|---|---|
| PIC-Std | $2^{-6}L$ ($64^2$ cells) | 500 | 2.048E+6 | 1 |
| PIC-Sg / PIC-Hg | $2^{-6}L$ ($64^2$ cells) | 500 | 5.440E+5 | 4 |

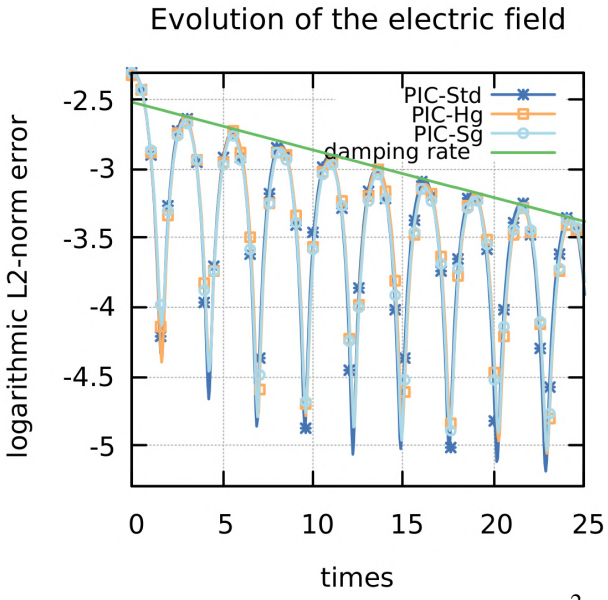


**Figure 6.2.** Linear Landau damping: evolution of the electric field $L^2$-norm in time, $k \approx 0.286$.

### Non-linear Landau damping

When the perturbation of the equilibrium state is considered large enough to invalidate the linear approximation, the precedent analytic damping rate is not available anymore. In order to assess the efficiency of the methods, the results will be compared to the PIC-Std scheme in the same configuration of grid discretization. A larger perturbation than for the linear case is considered with $\alpha_1 = 0.2$, $\alpha_2 = 0.15$, $\beta_1 = 4$, $\beta_2 = 3$ in equation (6.5) and let $L = 60\lambda_D$, $\Delta t = 0.05\omega_p^{-1}$. The grid discretization is chosen so that $h_n \simeq 0.47\lambda_D$, the configuration of the grid, particles and schemes is indicated in table 6.2. Interpolations in the combination technique are done with linear splines for each of the methods.

**Table 6.2.** Non-linear Landau damping: Configuration of the methods.

| Scheme | Parameters | Grid size $h_n$ | $P_c$ | $N$ | $N_{Ref}/N$ |
|---|---|---|---|---|---|
| Ref (PIC-Std) | | $2^{-7}L$ ($128^2$ cells) | 4000 | 6.553E+7 | 1 |
| PIC-Std | | $2^{-7}L$ ($128^2$ cells) | 1000 | 1.638E+7 | 4 |
| PIC-Sg / PIC-Hg | | $2^{-7}L$ ($128^2$ cells) | 1000 | 2.560E+6 | 25.6 |
| PIC-Sg / PIC-Hg (offset) | $(\tau^0, \tau^1) = (1, 0)$ | $2^{-7}L$ ($128^2$ cells) | 1000 | 1.792E+6 | 36.6 |

A first series of simulations is performed with different grid resolutions, numbers of particles per cell and total number of particles in order to get a comparison of the projection error between the methods at initial time. It is therefore possible to assess precisely the precision of the density projected onto the grid for the different methods by comparison with the analytic expression of the initial electron density ($\rho_{ex}$). The error of the density in $L^2$ or supremum norm is defined as:

$$\epsilon_p(\rho) := \frac{\|\rho - \rho_{ex}\|_p}{\|\rho_{ex}\|_p}, \quad p = 2, \infty \tag{6.28}$$

where $\rho_{ex}$ is the analytic density at initial time. The integrals in the $L^2$ norm expressions are approximated with a Riemann sum on the Cartesian grid. The numerical results are represented in figure 6.3. The results for the PIC-Sg scheme are not represented because at initial time the

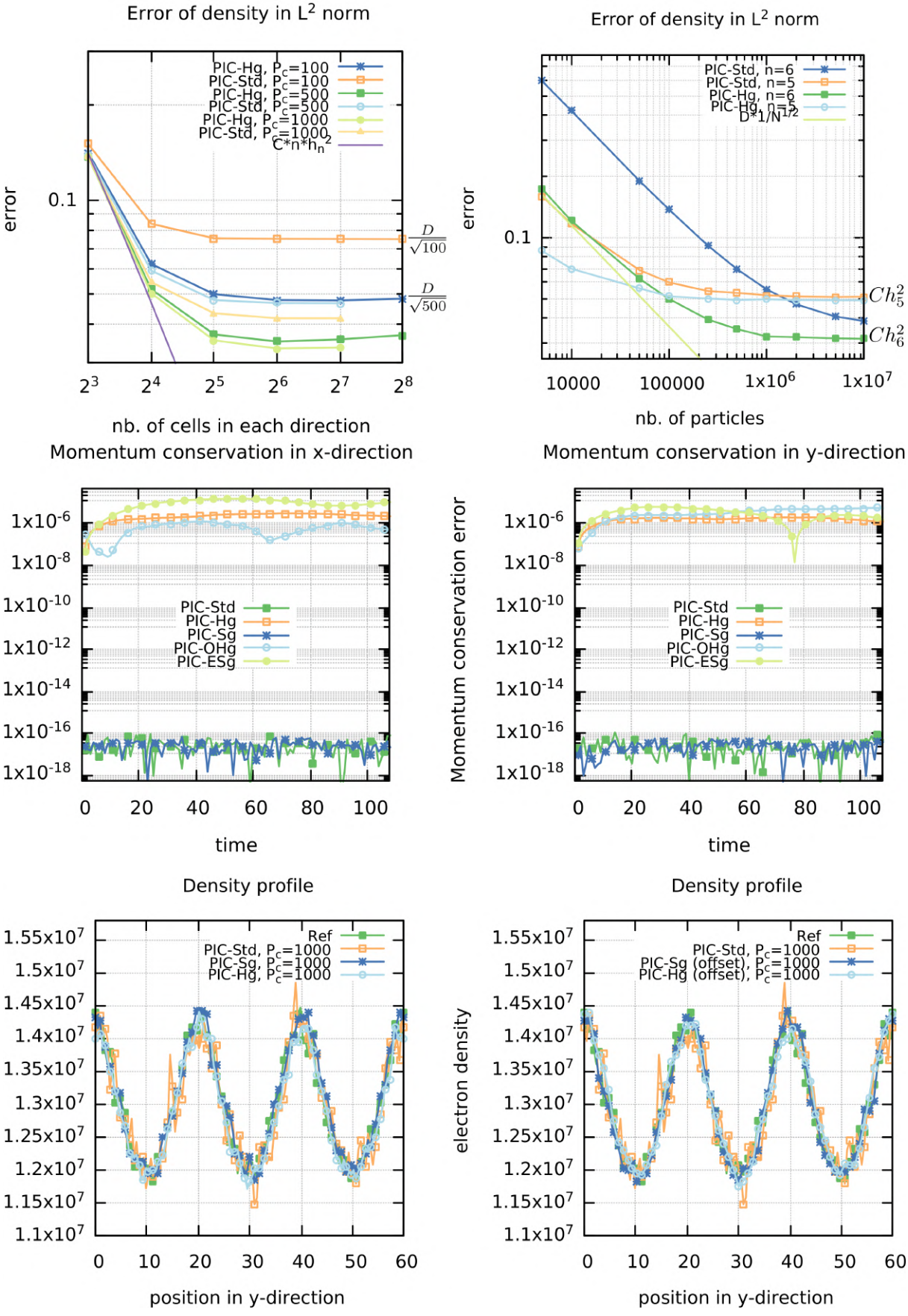


**Figure 6.3.** Non linear Landau damping: Density relative error $L^2$-norm $\epsilon_2(\rho)$ (top); momentum conservation (middle); density profile of the electron density (bottom).

scheme is equivalent to the PIC-Hg scheme. On the left figure, for any of the methods, the precision is limited by the particle sampling error for quite coarse grid resolutions, as soon as the number of cells is larger than $2^5$ in each direction. Though the total number of particles is increased with the mesh refinement, the mean number of particles per cell remains unchanged, which explains the non decreasing error observed on the plots of figure 6.3 (proportional to $\frac{1}{\sqrt{P_c}}$ for the standard method). This highlights that, though the number of particles per cell is consequent (five hundred or one thousand), it should be increased further to obtain an optimal precision and reduce the particle sampling error to a value comparable to the grid-based error. This outlines an important characteristic of PIC methods: the grid-based error is marginal compared to the particle error. This proves that gains may be expected from numerical methods with a better control of the statistical noise, hence the interest for sparse grid reconstructions. On the right figure the grid discretization is frozen so that the grid-based error is constant, proportional to $h_n^2$ for all the methods which testify that the multiplicative term $|\log h_n|^{d-1}$ in the error expression of the sparse grid methods is negligible. When the number of particle increases, the particle sampling error converges (in the $L^2$ norm) to zero at the rate $N^{-\frac{1}{2}}$. The global error converges to the grid-based error. For refined grids with more than $2^6$ cells in each direction, the amount of statistical noise of the PIC-Hg scheme with one hundred particles per cell is equal to that of the PIC-Std scheme run with five hundred particles per cell. This amounts to a total number of particles 32 times less for the sparse method compared to the standard method.

The second series of simulation for the non-linear Landau damping is dedicated to the time evolution of the perturbation. First, the conservation of the total momentum is investigated for the different methods. To this end, let us introduce the following error for the momentum:

$$\epsilon_{\mathcal{P}}(t) := \frac{1}{N}\sum_{p=1}^{N}\frac{m_e\mathbf{v}_p(t_0) - m_e\mathbf{v}_p(t)}{m_e v_{th}}, \tag{6.29}$$

where $\mathbf{v}_p(t)$ is the velocity of the $p^{\text{th}}$ particle at time $t$ and $v_{th} := \sqrt{2q_eT_e/m_e}$ is the thermal velocity of the electrons. The default of momentum conservation is represented as a function of time in figure 6.3 (middle) with $P_c = 500$. As predicted by the analysis conducted before, the total momentum is exactly conserved (to machine precision) for the PIC-Sg with any of the classical combination technique or the offset combination technique, as well as the PIC-Std scheme. The conservation default for the PIC-Hg, PIC-OHg and PIC-ESg schemes is observed to remain marginal ($\approx 10^{-6}$) and bounded independently of time.

The projection onto the Cartesian grid and a section in the $x$-direction of the electron density are proposed in figures 6.3 (bottom) after two periods of oscillation of the electric field (at time $T = 6.3\omega_p^{-1}$) for the different configurations of table 6.2. Since the density is never projected onto the Cartesian grid within the PIC-Sg and PIC-ESg schemes, we perform an interpolation on the Cartesian grid according to the combination technique for the diagnostics.

First, it appears that the reduction of the numerical noise is manifest for all the methods using sparse grid reconstructions. It is an essential property since, this error, due to the undersampling of the distribution function, is the most detrimental in the precision of numerical methods. This better control of the numerical noise is obvious on the density plots displayed on figures 6.3 (bottom) and 6.4. Though the estimated mean number of particles per cell is equal to 1000 for the PIC-Hg, PIC-Sg, PIC-OHg (offset combination) and PIC-ESg (offset), the magnitude of the dispersion is observed to be comparable, or even less, to that of the PIC-Std scheme with 4000 particles per cell which amounts to a total number of particles 25, or 36 times greater (see table 6.2). It is also noticeable that sparse grid approximations do not introduce too much numerical diffusion, the extrema of the numerical approximations being comparable whatever

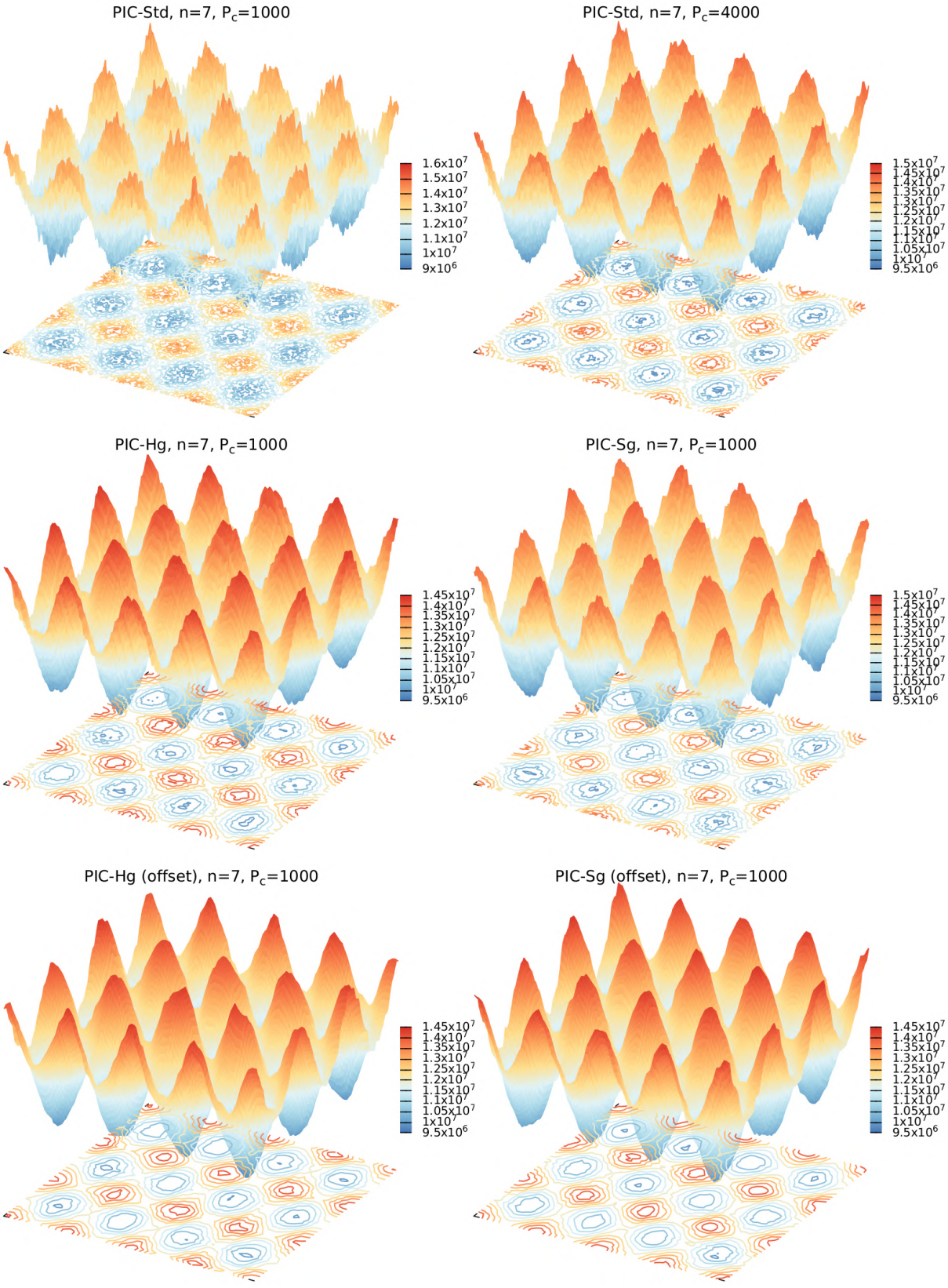


**Figure 6.4.** Non linear Landau damping: electron density.

the scheme. Second, the grid-based error can be observed, particularly on the plots of the PIC-Sg and PIC-Hg methods figure 6.4. This error reproduces the patterns of the coarsest grid levels. This error is specific to sparse grid approximations of the density and analysed to be dominated by the component scaling with $O\left(nh_n^2\right)$. This latter term results from the accumulation of error on the different levels of sub-grids and is reduced thanks to the offset combination technique. The smearing of the grid structure on the error plots related to the computations performed with the offset combination technique can be observed on figure 6.4. Indeed, in these computations the number of sub-grids considered for the reconstruction is reduced considering $n_1 < n$ (from $2n - 1 = 13$ sub-grids to $2n_1 - 1 = 7$ sub-grids). The control of the numerical noise (with approximately the same number of total particles), though expected deteriorated, due to the use of more refined grids compared to the original methods, remains very effective improving significantly the quality of the numerical results. For these methods, still a good control of the numerical diffusion shall be pointed out. Similarly, the increase of the negligible grid-based component (see proposition 2.2.12) reveals to remain marginal on the precision of the numerical approximation.

### Diocotron instability

Let the parameters be $\sigma = 0.03$, $L = 22\lambda_D$, $\Delta t = 0.1\omega_p^{-1}$, the system is observed at time $T = 54\omega_p^{-1}$. The numericals results will be compared to the PIC-Std scheme with a grid composed of $256 \times 256$ cells and $P_c = 200$ particles per cells as indicated in table 6.3. The interpolations are implemented using linear splines while splines of degree two are used for the density visualization. Following steps of proof of proposition 2.2.5, one can see that the use of spline of degree two, even restricted to the visualization of the density on the grid, provides a better representation of the density.

**Table 6.3.** Diocotron instability: Configuration of the numerical methods.

| Scheme | Parameters | Grid size $h_n$ | $P_c$ | $N$ | $N_{Ref}/N$ |
|---|---|---|---|---|---|
| Ref (PIC-Std) | | $2^{-8}L$ ($256^2$ cells) | 200 | 2.621E+7 | 1 |
| PIC-Std | | $2^{-8}L$ ($256^2$ cells) | 40 | 2.621E+6 | 5 |
| PIC-OHg | $\delta n = 2$ | $2^{-8}$ ($256^2$ cells) | 40 | 1.187E+6 | 11 |
| PIC-SG | | $2^{-10}L$ ($1024^2$ cells) | 40 | 1.187E+6 | 11 |
| PIC-ESg / PIC-OHg (offset) | $(\tau^0, \tau^1, \delta n) = (4, 4, 1)$ | $2^{-8}L$ ($256^2$ cells) | 40 | 1.802E+6 | 7.25 |

The projection of the electron density onto the Cartesian grid and a section in the $x$-direction (at $x = \frac{2L}{3}$) at time $T$ for the different configurations of table 6.3 are represented in figures 6.5, 6.6. The numerical results are presented only for the PIC-OHg scheme and not for the PIC-Hg scheme because the resolution of the Poisson problem for the latter is exceedingly costly in comparison to the others methods. Besides, the same accuracy is achieved with either the PIC-OHg scheme or the HG scheme, hilighting the benefit of the oversampled correction of the scheme. The discretizations of the sparse grid schemes with the classical combination technique ($h_n = 2^{-10}L$, $h_{\tilde{n}} = 2^{-10}L$) are chosen higher than the standard ones ($h_n = 2^{-8}L$) because the sparse grid schemes fail to reproduce the fine-scale structures depending on cross derivative terms. Indeed, we have shown in section 2.1 that the grid-based error component depending on mixed derivatives scales with $O\left(nh_n^2\right)$ for the sparse grid schemes (compared to $O\left(h_n^4\right)$ for the standard scheme).

Despite the use of a more refined grid discretization, we observe on the plots of the section in the $x$-direction (figure 6.6) that the PIC-Ohg and PIC-Sg schemes with the classical combination technique still fail to reproduce correctly the fine-scale structure of the density. Indeed, where the solution has steep gradients ($y \approx 5.5$, $y \approx 9$ in figure 6.6) the sparse grid schemes show a

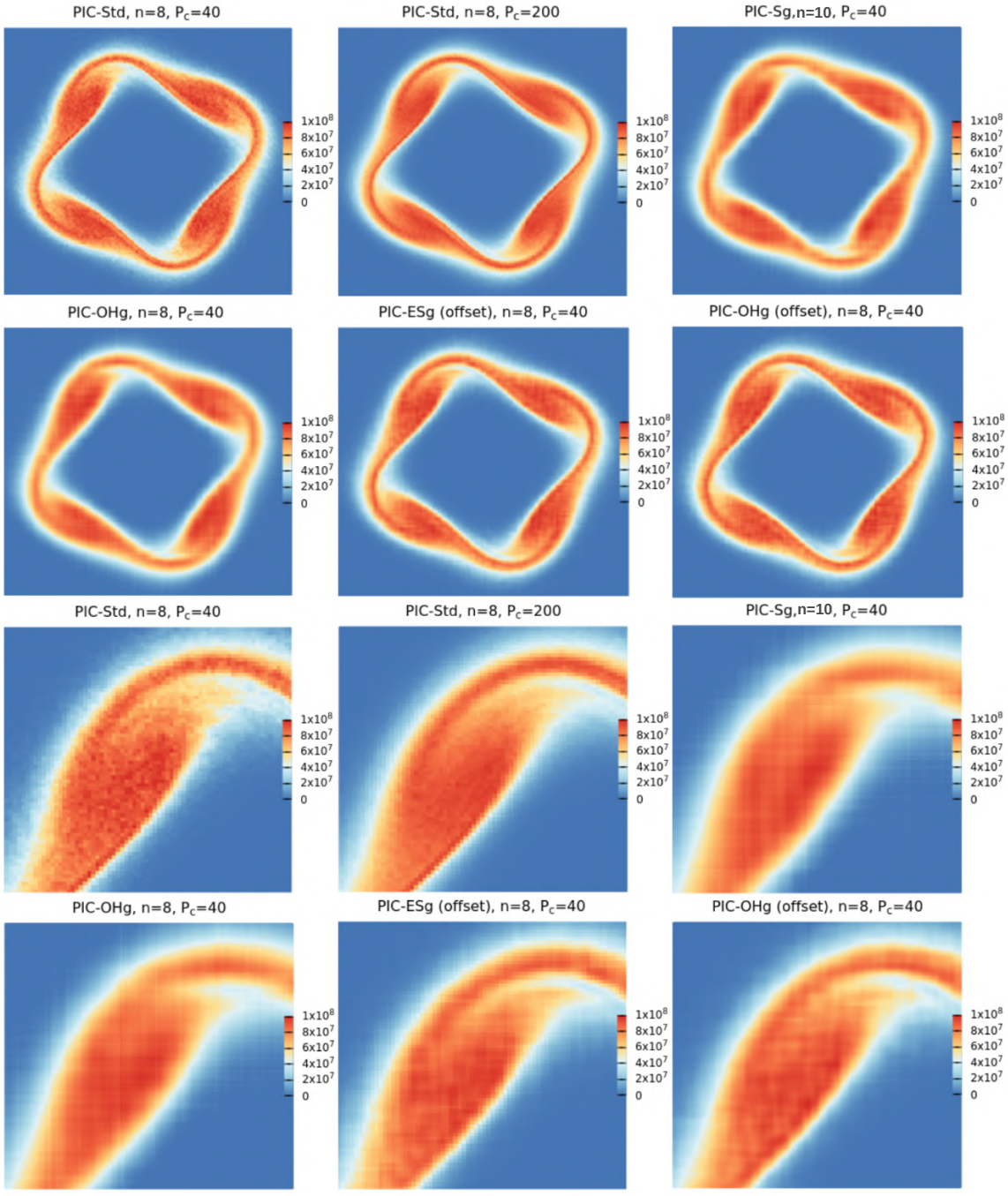


**Figure 6.5.** Diocotron instability: electron density, $t = 54\omega_p^{-1}$.

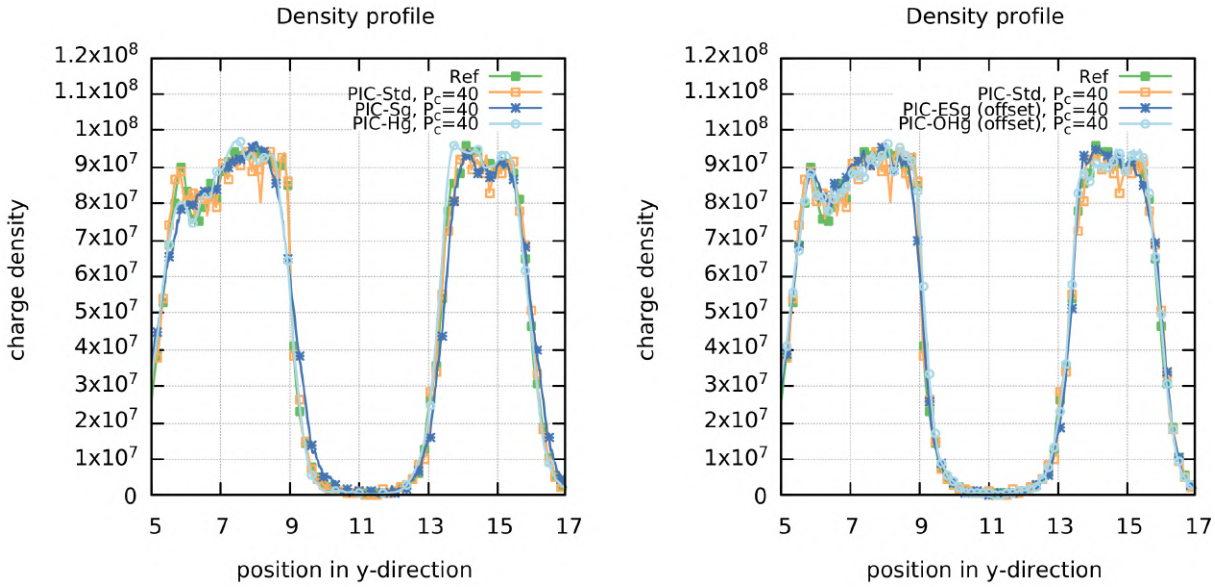


**Figure 6.6.** Diocotrom instability: density profile, $t = 54\omega_p^{-1}$.

bit of numerical diffusion. The correction of the schemes with the offset combination technique achieves a fair representation of the density and even reproduce some fine-scale structures that are faded by the statistical noise of the PIC-Std scheme with $P_c = 40$ (see the zooms on figure 6.5). The improvement is manifest on the plot of the section in the $x$-direction (figure 6.6) where the numerical diffusion introduced by the sparse grid schemes has been mitigated. Here again the sparse grid techniques achieve an improvement on the statistical noise with the same mean number of particles per cell (and thus less total particles in the simulation) than the regular PIC approximation (see the plot of the section in the $x$-direction for the density in figure 6.6). Though this test case is a very demanding benchmark for sparse grid approximations a reduction of the total number of particles is achieved (see table 6.3) and all the fine-scale structures appearing in the reference solution are well reproduced by the corrected schemes.

### 6.2.2 Numerical investigations of the semi-implicit schemes

#### Instability of the semi-implicit sparse-PIC scheme

While performing numerical investigations on the semi-implicit sparse grid method, we have experienced numerical instabilities not observed with the standard ECSIM method. The numerical instabilities are related neither to the time discretization nor the space discretization. We have found that it might be instead related to the loss of positivity of the field energy. Indeed, according to the remark 5.2.1, the reconstructed field energy is not guaranteed to be a non negative quantity, even though the field energy defined on each component grid is positive. This is a drawback of the combination technique used to reconstruct the field energy that does not preserve the positivity of the solution (see remark 2.1.4). As a result, when the field energy becomes negative, because of the total energy conservation (see theorem 5.2.2), the particle velocities are increased to compensate the loss in the field energy. We have observed that once the field energy is negative, it tends to become more and more negative so that the particle velocities increase in response, leading to numerical instabilities.

However, the origin of the loss of positivity and the context in which the instability is triggered is not yet well understood. These observations suggest us that future investigations shall be performed to better understand the nature and origin of this instability and maybe correct

it. In the following, as soon as the field energy is negative, the simulation is broken down so that no result is provided for consecutive times. As a first correction to this instability, we propose the Imp-PIC-HG scheme for which the instability is absent. Nonetheless the scheme does not conserve exactly the total energy of the scheme.

### Finite grid instability

The so-called aliasing or finite grid instability, first studied in [82], is a common numerical instability arising in PIC plasma simulations. It originates from the inconsistency between the discrete Eulerian discretization of the fields on a grid and the continuous discretization of Lagrangian particles in the phase-space [72].

This instability is manifested in simulations by a numerical heating of the plasma [18] related to the numerical parameters. Since the aliasing introduces artificial heat in the system, it is also characterized by a violation of the energy conservation. Usually in PIC simulations, the aliasing instability is avoided by choosing a grid discretization equal or smaller than the Debye length $h_n \leq \lambda_D$, including for problems with scales of interest much larger than the Debye length. For example, dense plasmas are well described by the quasi-neutral approximation in most of the domain and the simulation of the plasma physics does not required grid cells smaller than the Debye length. Therefore large gains could be obtained with coarse grid cells that do not resolve the Debye length, especially for three dimensional computations.

Various methods and numerical schemes have been proposed to mitigate this instability, including introducing grid interlacing [32], random jiggling [21], using higher order particle shapes [16] or temporal/spatial filtering [84]. Besides, implicit scheme with exact energy conservation, such as the ECSIM scheme [83] have proven numerically to preserve the simulations from aliasing.

Nonetheless, analysis of the aliasing instability is not straightforward. It has been conducted linearly for stationary plasmas and specific schemes, such as the fully-implicit energy-conserving scheme in [7]. For drifting plasma however, it has been shown that in principle the scheme is not exempted from the instability, but that in practice the scheme is almost always freed from it. In this section, we address to establish numerically that the ECSIM scheme embedding sparse grid reconstructions does not feature finite grid instability in classical configurations where the explicit discretizations does.

An initially Maxwellian and stable plasma is considered with the following distribution of electron:

$$f_v^0(\mathbf{x}, \mathbf{v}) = \left( \frac{1}{\sqrt{\pi} v_{th,e}} \right)^2 e^{-\|\mathbf{v}\|_2^2 / v_{th,e}^2}, \tag{6.30}$$

where $v_{th,e} = \sqrt{2}$ is the thermal velocity of electrons and $\|\mathbf{v}\|_2^2 = v_1^2 + v_2^2$. The electrons are considered immersed in a uniform background of ions. The size of the domain is $L = 5\pi, 10\pi, 15\pi, 20\pi$ and the grid discretization is $h_n = 2^{-5} L$. Some of these configurations shall lead to the development of the finite-grid instability for the momentum conserving explicit schemes, since the grid discretization is larger than the Debye length.

First, let us investigate the explicit schemes (Ex-PIC-Std, Ex-PIC-Sg). The first dimension of the particle phase space $(x_1, v_1)$ is represented at time $T = 100$ on figure 6.7 for the different configurations of the domain. The finite grid instability is visible for the explicit schemes with the configurations in which the grid discretization is larger than the Debye length. The implicit standard scheme is preserved from the finite grid instability for all configurations. Let the kinetic

and field energy error at time iteration $k+1$ be defined by:

$$\boldsymbol{\varepsilon}_E^{k+1} := \max_{i\in\{1,\dots d\}} \left| \left( \frac{\mathcal{E}_K^{k+1} + \mathcal{E}_F^{k+1} - \left(\mathcal{E}_K^0 + \mathcal{E}_F^0\right)}{\mathcal{E}_K^0 + \mathcal{E}_F^0} \right)_i \right|, \tag{6.31}$$

where $\mathcal{E}_K^k$ and $\mathcal{E}_F^k$ are the kinetic and field energy measured on the mesh at time $k$. The total energy errors at time $T$ for the different configurations are provided in table 6.4.

**Table 6.4.** Finite grid instability: total energy errors for the different configurations.

| Scheme | $L$ | $h_n/\lambda_D$ | $T$ | $\boldsymbol{\varepsilon}_E(T)$ | Scheme | $L$ | $h_n/\lambda_D$ | $T$ | $\boldsymbol{\varepsilon}_E(T)$ |
|---|---|---|---|---|---|---|---|---|---|
| Ex-PIC-Std | $5\pi$ | 0.49 | 100 | 3.0E−3 | Ex-PIC-Std | $15\pi$ | 1.47 | 200 | 1.0E+1 |
| Ex-PIC-Sg | $5\pi$ | 0.49 | 100 | 2.5E−3 | Ex-PIC-Sg | $15\pi$ | 1.47 | 200 | 1.8E+1 |
| Imp-PIC-Std | $5\pi$ | 0.49 | 100 | 1.8E−15 | Imp-PIC-Std | $15\pi$ | 1.47 | 200 | 3.5E−15 |
| Ex-PIC-Std | $10\pi$ | 0.98 | 100 | 9.4E−2 | Ex-PIC-Std | $20\pi$ | 1.97 | 100 | 7.0E−1 |
| Ex-PIC-Sg | $10\pi$ | 0.98 | 100 | 8.1E−2 | Ex-PIC-Sg | $20\pi$ | 1.97 | 100 | 1.3E+0 |
| Imp-PIC-Std | $10\pi$ | 0.98 | 100 | 1.1E−15 | Imp-PIC-Std | $20\pi$ | 1.97 | 100 | 1.3E−15 |

### Linear Landau damping

Let us parametrize the perturbation with $\alpha_1 = \alpha_2 = 0.05$, $k = 0.3$ or $k = 0.5$ such that the domain size is $L = \frac{20}{3}\pi$ ($k = 0.3$) or $L = 4\pi$ ($k = 0.5$). The time step is $\Delta t = 0.025, 0.25, 1$ and the final time $T = 30$, resulting in 1200, 120 and 30 iterations. The grid discretization is $h_n = 2^{-5}L$ so that the Debye length is resolved $h_n \simeq 0.39\lambda_D$ ($k = 0.5$), $0.65\lambda_D$ ($k = 0.3$) and the number of particles per cell is $P_c = 1000$. The evolution of the electric field $L^2$-norm in time for all configurations is provided in figures 6.8. For the perturbation parametrized with $k = 0.5$, the theoretical damping rate is well reproduced thanks to the semi-implicit schemes (Imp-PIC-Std, Imp-PIC-Sg) at any time step ($\Delta t = 0.25, 1$). On the other hand, the explicit standard scheme is unstable for relatively large $\Delta t$ (even for $\Delta t < 2\omega_p^{-1}$) and provide an accurate damping rate for $\Delta t = 0.025$. For $k = 0.3$, the explicit scheme is stable for all $\Delta t < 2\omega_p^{-1}$, as predicted by the linear dispersion theory (see equation (1.32)), and unstable for larger time steps. The damping rate is well reproduced at $\Delta t = 0.25$ for the implicit schemes, but seems to be more accurate for the sparse-PIC scheme at large time.

The evolution of the total momentum and total energy conservation errors are represented in figure 6.8. The momentum error in the simulation at iteration $k+1$ is measured by the infinite norm of the momentum error vector:

$$\boldsymbol{\varepsilon}_{\mathcal{P}}^{k+1} := \max_{i\in\{1,\dots d\}} \left| \left( \sum_s \frac{1}{N_s} \sum_{p=1}^{N_s} \frac{m_p \mathbf{v}_p^{k+1} - m_p \mathbf{v}_p^0}{m_s v_{th,s}} \right)_i \right|, \tag{6.32}$$

where $\mathbf{v}_p^k$, $m_p^k$ are the velocity and mass of the $p^{\text{th}}$ particle at time iteration $k$ and $v_{th,s} := \sqrt{2q_sT_s/m_s}$ is the thermal velocity of the species $s$. The mass is defined by $m_p = m_s n_0/N_p$. As expected, the semi-implicit schemes conserve exactly the total energy of the system but not the momentum, and the explicit scheme conserve exactly the total momentum of the system but not the energy. Nonetheless, even for large time step ($\Delta t = 1$), the default in the momentum conservation is below $\approx 10^{-5}$ for the implicit schemes.

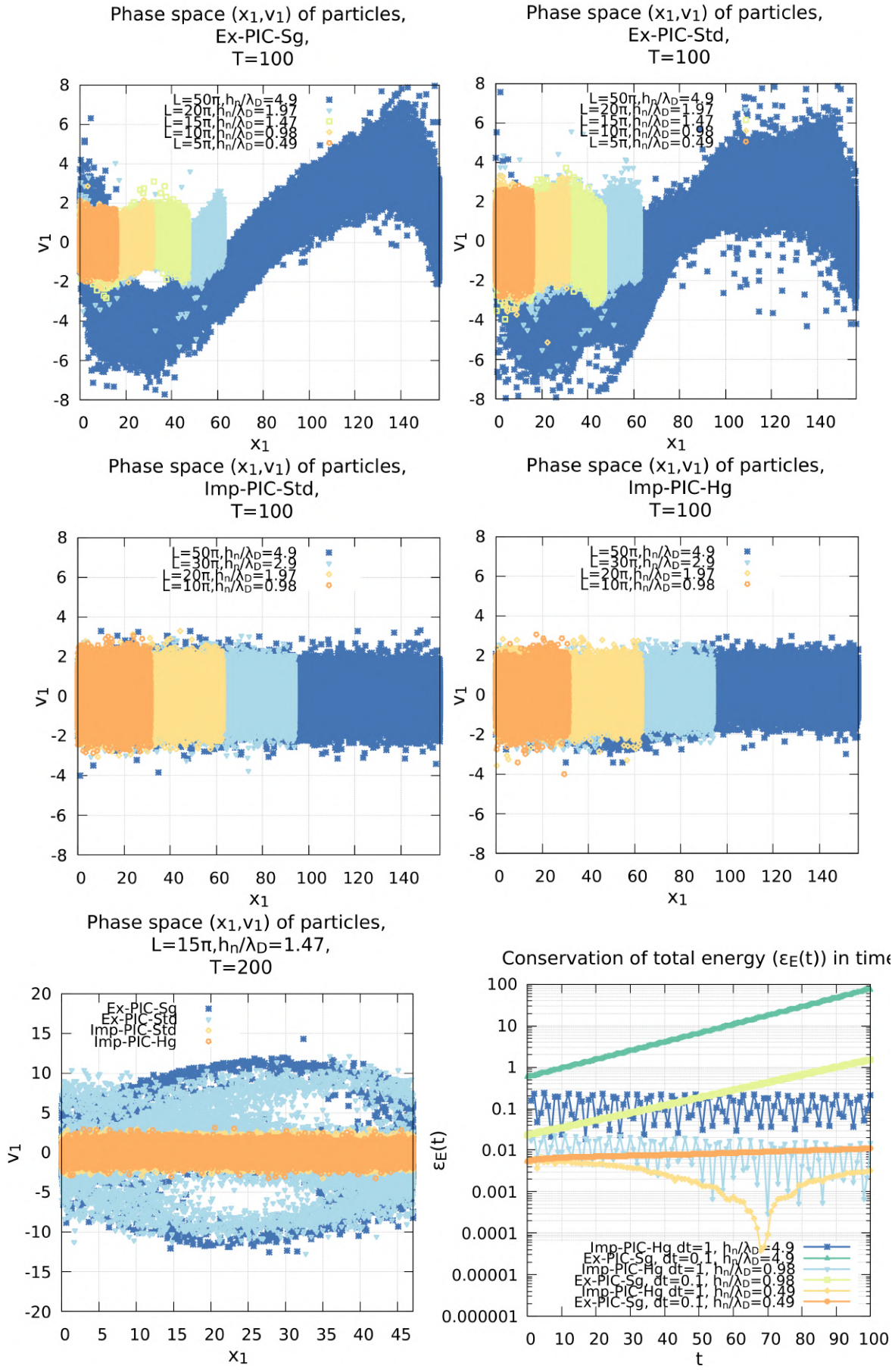


**Figure 6.7.** Finite grid instability: representation of the phase space $(x_1, v_1)$ of an initially Maxwellian distribution of electron and conservation error of the total energy in time. $\Delta t = 0.1$ for the explicit schemes and $\Delta t = 1$ for the implicit scheme, $P_c = 100$.

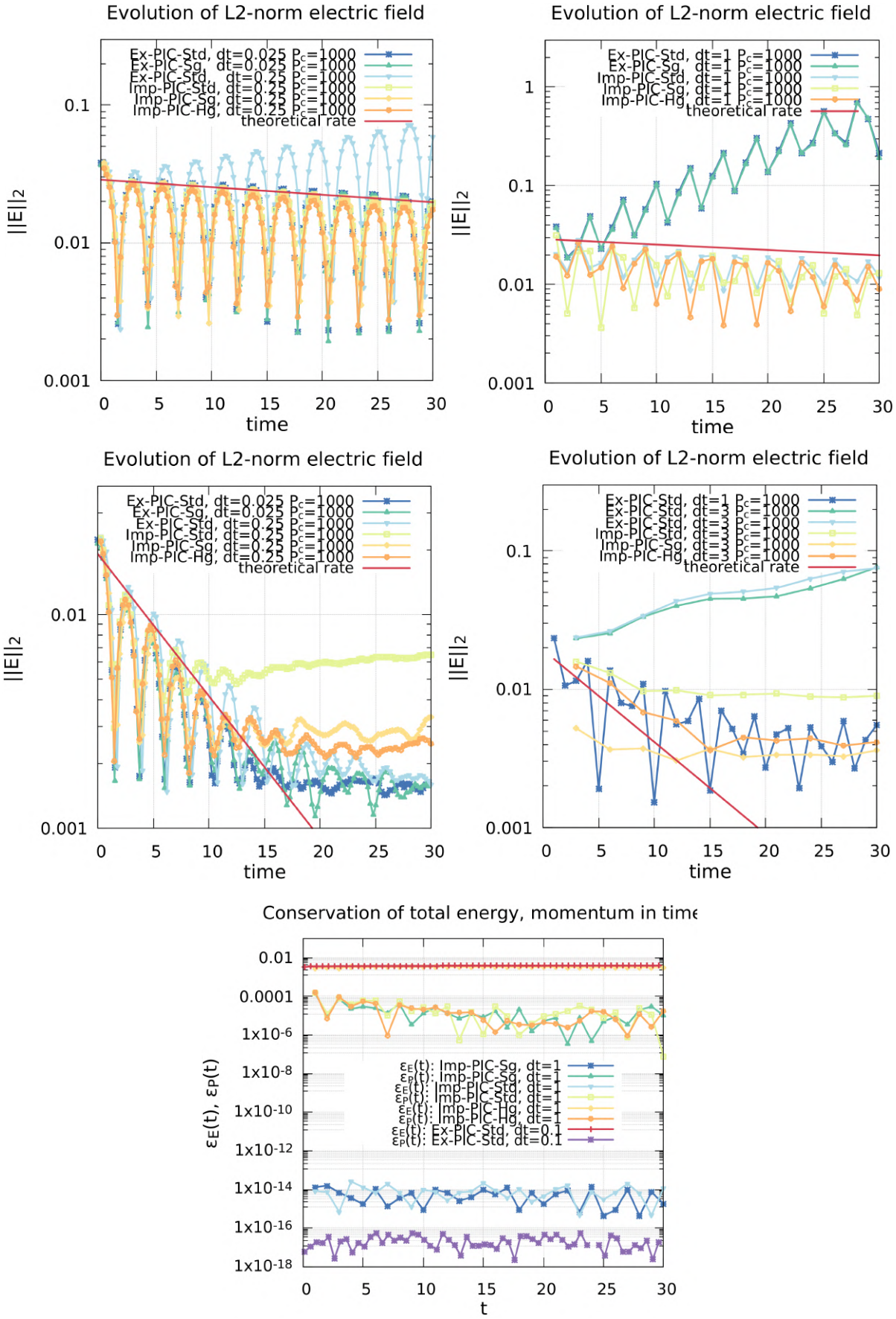


**Figure 6.8.** Linear Landau damping: evolution of the electric field $L^2$-norm in time, $k = 0.3$ (top), $k = 0.5$ (middle) and evolution of the total energy and total momentum conservation error in time, $k = 0.5$ (bottom).

#### Two-streams instability

Let us parametrize the perturbation with $\alpha_1 = \alpha_2 = 0.005$, $k = 0.2$ or $k = 0.05$ such that the domain size is $L = 10\pi$ ($k = 0.2$) or $L = 40\pi$ ($k = 0.05$). The mean velocity is $v_0 = 3$ or $v_0 = 12$ such that the growth is similar for the two configurations (similar $k^2 v_0^2$). The time step is $\Delta t = 0.1$ and the final time $T = 60$, resulting in 600 iterations. The grid discretization is $h_n = 2^{-5}L, 2^{-7}L$ so that the Debye length is resolved for explicit schemes: $2^{-5}10\pi = 2^{-7}40\pi \simeq 0.98\lambda_D$, and the number of particles per cell is $P_c = 100, 500$. The evolution of the electric field $L^2$-norm in time for all configurations is provided in figures 6.9. The simulation with the Imp-PIC-Sg scheme is stopped before the final time because the field energy becomes negative for $t \approx 23$. For the configuration with $k = 0.2$, all the schemes reproduce well the growth rate, expect for the Imp-PIC-Std slope which is slightly different from the theoretical one. Nonetheless, the accuracy of the results is increased with more particles per cell ($P_c = 500, 1000$) and the theoretical growth rate is recovered. For the second configurations, $k = 0.05$, all the schemes reproduces accurately the growth rate for $P_c = 100$.

## 6.3 Three-dimensional geometries

### 6.3.1 Sequential performances and shared memory parallelization

Throughout this section the different methods presented previously, namely the sparse grid schemes with hierarchical (PIC-HSg) or nodal combination (PIC-NSg), with first (PIC-HSg1) or second option (PIC-HSg2) (see algorithm 8) and the standard scheme (PIC-Std), are investigated and compared. In order to provide a fair comparison between the sparse grid methods and the standard methods, we introduce a naive implementation of the method (without sorting of particles), identified as PIC-UStd (Unsorted Standard), and an implementation based on initially sorted particle data, named PIC-SStd (Sorted Standard). These implementations provide lower and upper bounds of the method efficiency (the sorting time is not taken into account). The numerical results consist of a mean runtime of the first five iterations of the scheme at the end of which only 4% of the particles have moved to a different cell of the grid containing $128^3$ cells with 75 particles per cell in the standard case. This means that the benefits of the particle sorting performed at the initial time step are very effective for whole computations. In the following, the scheme is divided into six steps, namely the projection of the density (charge accumulation) onto the grid (Proj), the resolution of the Poisson equation (Pois), the differentiation of the electric potential (Diff), the pusher of particles (Push), the interpolation of the electric field (F.Inter), the combination of the component grid contributions in either nodal or hierarchical basis (Comb). An additional step is performed within the scheme for the review of the numerical results; the charge density is reconstructed from all the component grid contributions and represented on the full grid.

#### Sequential results and performances

#### Landau damping

First, let us consider the Landau damping test case. For such configurations, sparse grid reconstructions have already proven to be more efficient than standard PIC in terms of computational time and memory requirements [97]. In this section, the optimizations of the sparse-PIC implementation such as the hierarchization, memory management and Poisson solver for sparse-PIC are investigated and compared to standard PIC.

The results of the computational time of the Poisson solver for each scheme with different methods (direct, iterative) and configuration of the linear system (one large system or many small

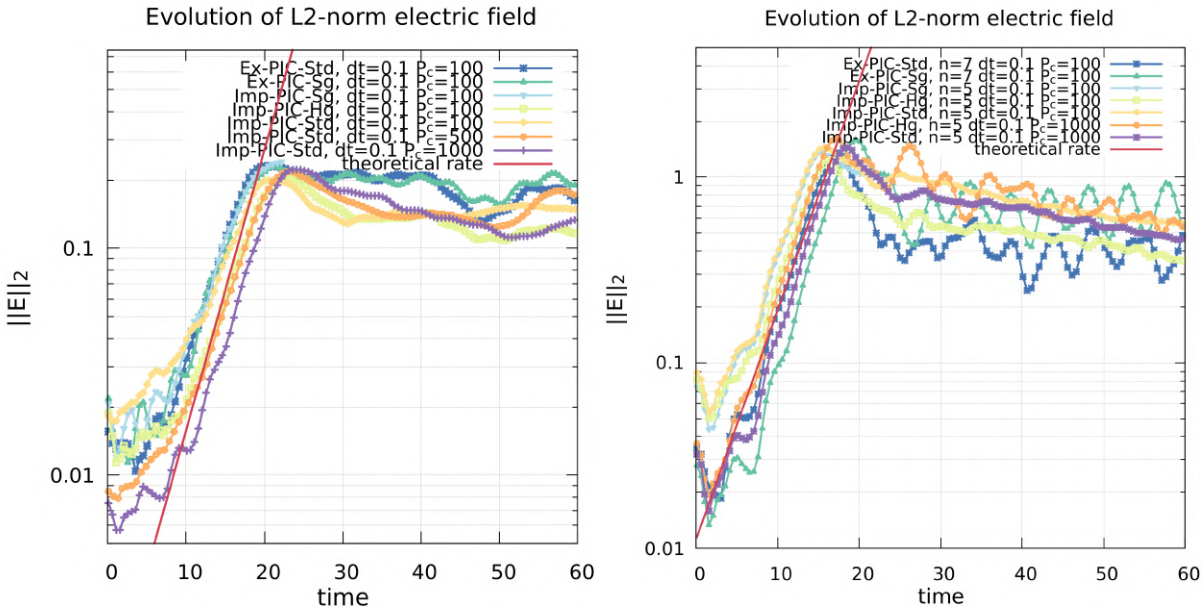


**Figure 6.9.** Two streams instability: evolution of the electric field $L^2$-norm in time, $k = 0.2$, $v_0 = 3$ $L = 10\pi$ (left), $k = 0.05$, $v_0 = 12$, $L = 40\pi$ (right).

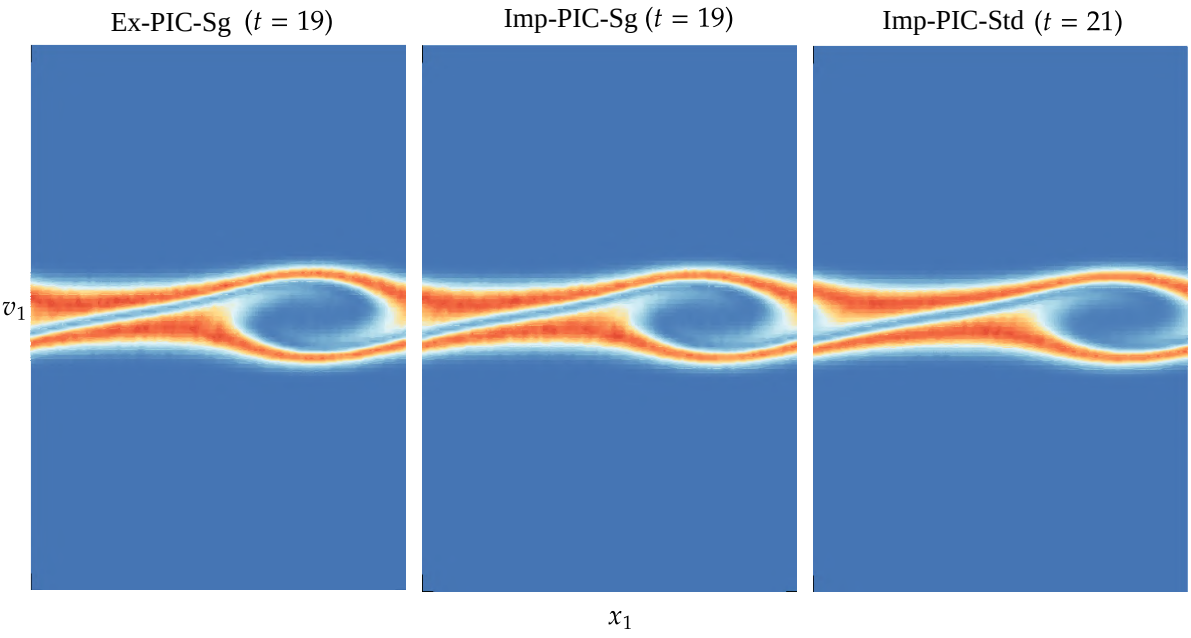


**Figure 6.10.** Two streams instability: evolution of phase space density (first dimension $(x_1, v_1)$), $n = 5$, $P_c = 1000$, $k = 0.2$, $v_0 = 3$ $L = 10\pi$.

systems, see section 3.2.2) are represented in figure 6.11. The direct method consists in a LU factorization performed at the initialization (not taken into account in the computational time) and followed by the resolution of a triangular system at each time step. The iterative method consists in a BiConjugate Gradient STABilized (BiCGSTAB) with a Geometric Algebraic MultiGrid (GAMG) preconditioner. The benefit resulting from the reduction of the grid nodes is manifest on the computational time of the linear system resolution. Indeed, dividing the mesh size by a factor 2 doubles the time of execution of the sparse grid scheme with LU decomposition whereas it is multiplied by ten for the standard scheme. It results in a significant difference between the standard and sparse-PIC approaches. The difference being of three orders of magnitude for $256^3$ grids. This efficiency deficit is predicted to deepen to four orders of magnitude for $1024^3$ grids: the resolution lasts more than one hour for the standard method whereas the sparse grid resolution is done in a tenth of a second. The conclusion is that for the sparse-PIC methods the most efficient configuration is the direct method performed either on a large system or independently on each system.

Also, the computational time of one time step for the PIC-HSg and PIC-NSg schemes is represented in figure 6.11 for comparable configurations of grid and number of particles per cell, according to formulae (6.1), (6.2). The observation upon the complexity of the combination in nodal and hierarchical basis presented in section 3.1.2 is verified here: the computational load of the combination has been drastically reduced (divided by 122 in the first and 240 in the second configuration) by the transformation into the hierarchical basis (from 12.5% to 0.11% and 91% to 4% of one iteration computational time). In the second test case, the combination in nodal basis takes a substantial amount of time because there are more grid nodes (in the Cartesian grid) than particles (about 12 times more grid nodes). As a result, the most efficient scheme on one core is the sparse grid scheme with the combination in hierarchical basis; and the combination in nodal basis is put aside of considerations in the following. As anticipated in the previous sections, the projection is by far the most costly operation within the sparse grid scheme (between 80% and 95 % of the time of the iteration) because of the large number of component grids involved in the charge deposition. The other steps are less time consuming than in the standard method because of the reduced number of particles and the smaller size of the linear systems. Theses observations confirm that our efforts should be concentrated into the optimization and parallelization of the projection.

The evolution in time of the momentum and total energy conservation errors are represented in figure 6.11. Note that, unlike the PIC-Std scheme, the PIC-HSg scheme does not preserve the momentum to round-off, yet the error remains negligible (about $10^{-7}$) throughout the simulation. This results from the choice made to interpolate the electric field first at the Cartesian grid before the interpolation at particle positions. To recover exact momentum conservation (to machine precision), the strategy shall be that of precedent work [48]: interpolate the electric field directly from the component grids to the particle positions. It may be however less efficient in term of computational time, especially for configurations with many particles per cell. The errors on the total energy conservation are comparable for the standard and sparse-PIC schemes.

The projection step is investigated in more details for the sparse grid schemes with both options of algorithm 8 on the one hand, and the standard scheme with sorted and unsorted particles on the other hand. The computational time of one step as a function of the grid discretization is represented on the panel a) in figure 6.13. The percentage of L1 data cache misses relative to the total number of data accesses during the projection is represented on the panel b) in figure 6.13. The storage requirements of the density array related to the memory size available on the different platforms is provided on the panel c) in figure 6.13. In order to extend the conclusions drawn in this section to the parallel case, in which each thread holds a copy of the (grid) array because of the competitive accesses between the threads (see section 3.2.1), the storage requirements of these copies are also represented when the data of each thread exceed the L2 cache memory (limitation between private and shared memory of the threads). As a

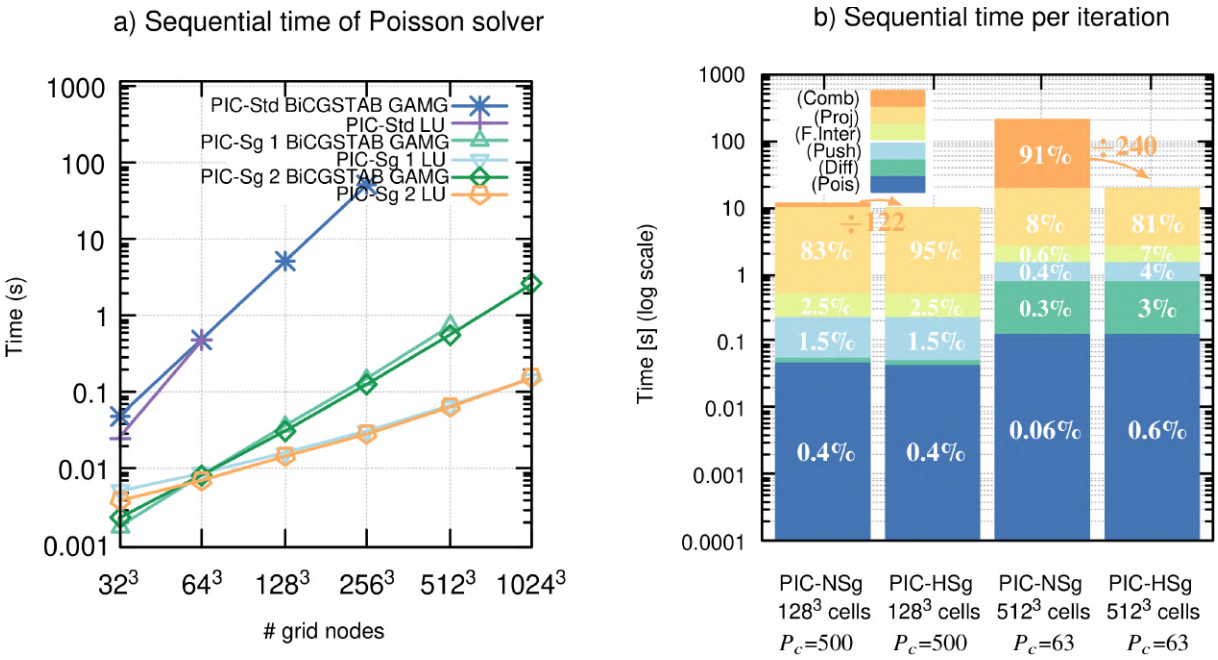


**Figure 6.11.** Landau damping: on panel a) the sequential time of Poisson solver: Direct sparse (MUMPS) and iterative BiConjugate Gradient STABilized (BiCGSTAB) with Geometric Algebraic MultiGrid (GAMG) preconditioner methods. PIC-Sg 1: The resolution of all sub-grids has been gathered in one large linear system. PIC-Sg 2: The linear systems issued from the component grids are solved independently one after another. On panel b) the sequential time of one time iteration (in logarithmic scale) for the PIC-NSg and PIC-HSg schemes. The use of the hierarchical basis shortens the combination step computational time by hundreds. Computations are carried out on one AMD Zen 3 core @2GHz.

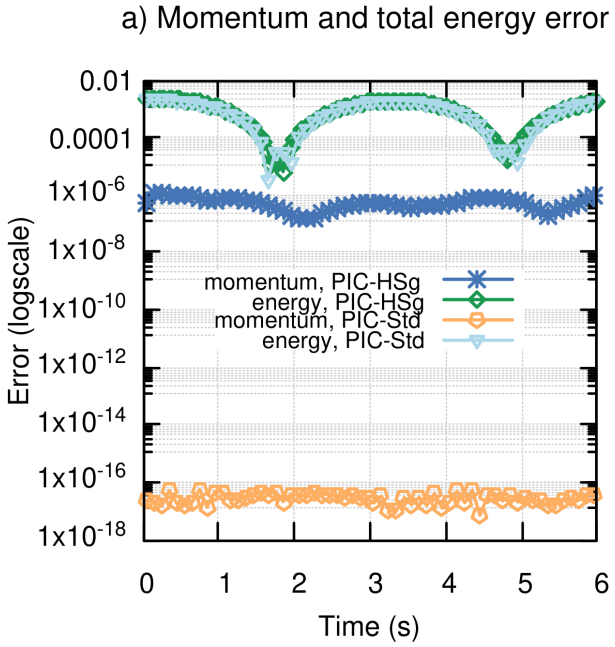


**Figure 6.12.** Landau damping: time history of the momentum and total energy conservation errors. conservation errors of momentum and total energy ($n = 7$, $P_c = 100$).

result of better memory reuse, the second option (see algorithm 8) is twice as fast as the first one for any grid discretization. The reason is the following: for the second strategy, the data fit in the L1 cache for all tested grids whereas for the first strategy the data must use slower L2 and L3 cache registers. As a consequence, the number of L1 cache misses during the projection step is significantly larger with the first option than with the second one, especially for fine discretizations. For the second strategy, the number of L1 cache miss per memory access is negligible and corresponds to the ratio of the standard scheme with the particle sorting, in which the data are accessed contiguously. This result demonstrates that within our approach the cache memory management is close to optimal for grids up to $1024^3$ cells.

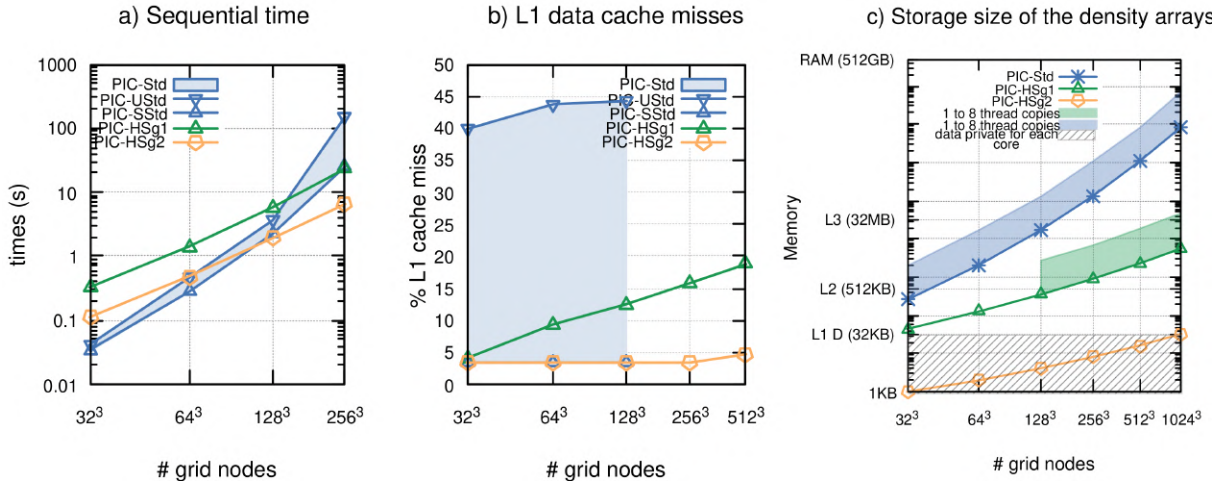


**Figure 6.13.** Landau damping: sequential time (panel a), ratio of L1 data cache miss per total data access (panel b) and storage of grid data (panel c) for the projection step with $P_c = 100$ (AMD Zen 3 core @1.7GHz).

### Manufactured solutions

Let us now investigate the manufactured solution test case. This process provides an analytic framework for a verification of PIC implementations [104]. Let $\Delta t = 0.05$ and $T = 5$.

The accuracy of the simulations is assessed by means of the L2-norm error of grid quantities (density and electric field). The L2-norm of the grid quantities (charge density and electric field) are defined by:

$$\|\mathbf{E} - \mathbf{E}_M\|_2 := \left( \sum_{\mathbf{j} \in I_{h_n}} \mathbf{E}^2_{\mathbf{j}h_n} h_n^3 \right)^{\frac{1}{2}}, \quad \|\rho - \rho_M\|_2 := \left( \sum_{\mathbf{j} \in I_{h_n}} \left( \rho^c_{\mathbf{j}h_n} \right)^2 h_n^3 \right)^{\frac{1}{2}}, \tag{6.33}$$

where $\rho_M$, $\mathbf{E}_M$ are the manufactured solutions and $\rho^c_{h_n}$ is the charge density reconstructed on the Cartesian grid with the combination technique. The PIC-UStd, PIC-SStd and PIC-HSg schemes are compared with a equivalent configuration, considering a $128^3$ grid (*i.e.* $n = 7$) and $P_c = 250$. The results are provided in table 6.24. The L2-norm errors of the electric field and charge density are represented in figure 6.14. The grid quantity errors of the two schemes are comparable. The computational time of one iteration, as well as the proportion of each step, is provided for each scheme in figure 6.14. The benefit of the sorting of particles is manifest on the grid/particles operations (i.e. the projection and the field interpolation) for the standard scheme and it results in an acceleration of 1.8 on the time iteration compared to the scheme

without sorting. This is the result of a better cache memory reuse achieved thanks to the sorting. Nonetheless this optimization concerns only the grid-particles operations, therefore it has no effects on the particle pusher which is the most costly operation.

The gain provided by the sparse grid reconstructions on the computational time is significant. The execution time of the sparse grid scheme, with an equivalent accuracy, is reduced by more than 20 times compared to the PIC-SStd. The substantial gain is due to the reduction of the number of particles in the sparse schemes (roughly 78 fewer particles). The large proportion (10%) of the particle pusher in the PIC-HSg total iteration is explained by the particle weight update, specific to the manufactured solution methods. In addition, the sparse grid scheme offers a significant reduction of the grid operations (thanks to 100 times fewer grid nodes for the sparse grid scheme).

**Table 6.5.** Results of the manufactured solution test case at $T = 5$ (after 100 iterations). Standard (sorted PIC-SStd and unsorted PIC-UStd) and sparse methods are considered with a $128^3$ grid, $P_c = 250$, on one AMD Zen 3 core @2GHz.

| Method | Particle memory footprint | $\|(\rho - \rho_M)(T)\|_2$ | $\|(\mathbf{E} - \mathbf{E}_M)(T)\|_2$ | Computational time |
|---|---|---|---|---|
| PIC-UStd | 75 GB | 8.9E−2 | 5.1E−2 | 100% (218 s) |
| PIC-SStd | 75 GB | 8.9E−2 | 5.1E−2 | 55% (120 s) |
| PIC-HSg | 972 MB | 7.6E−2 | 5.1E−2 | 2.7% (5.9s) |

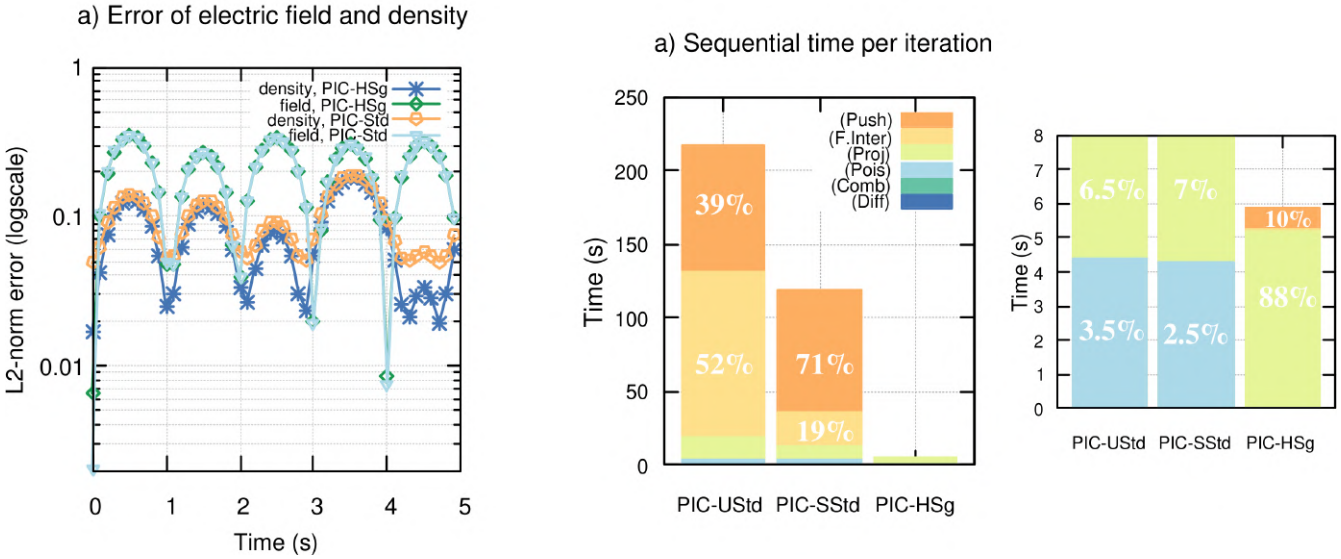


**Figure 6.14.** Manufactured solutions: a) L2-norm error of the grid quantities (charge density and electric field) as a function of time. b) computational time of one iteration on one AMD Zen 3 core @2GHz. All simulations have been performed on a $128^3$ grid and $P_c = 250$. The right panel is a zoom on the bottom of the left panel.

### Diocotron instability

As already put forward in [48, 97, 88, 105], sparse grid reconstructions may fail to reproduce non smooth solutions with for instance concentrated support and fine structures misaligned with

the axis as those developing with the diocotron instability. As an illustration of this feature, we investigate the diocotron test case in which instabilities caused by the magnetic field lead to the formation of a discrete number of vortices. Although the configuration is quasi two-dimensional because of the strong magnetic field in the z-direction and the two dimensional condition, which is favorable to sparse-PIC methods, it remains a challenging test case for sparse grid reconstructions since strong gradients in mixed directions (xy-direction) appear throughout the simulation. In [88, 48], improvements of the sparse grid reconstructions have been proposed, achieving a better approximation of the solution for challenging configurations like the diocotron instability. In this section, we investigate the performance of the PIC-HSg scheme implementing the offset combination technique [48] and compare it with the PIC-UStd and PIC-SStd schemes.

Let $\Delta t = 0.05$, $h_n = 2^{-7}L$ and $P_c = 80$. The offset combination technique, introduced in [48], is an alternative to the classical combination technique, consisting in an elimination of the most anisotropic grids from the combination. The component grids considered in the offset combination technique are more refined. As a result, the efficiency of the projection is deteriorated because the grids may no longer fit in the L1-cache memory. The computational time of the projection step and the Poisson solver step for the PIC-SStd, PIC-UStd and PIC-Hsg schemes with the offset combination technique ($\tau_0 = 1, 2, 3$, $\tau_1 = 6$) are represented in figure 6.15, panel a) and b). The storage size of the density array and the corresponding memory level is also represented for all the schemes within figure 6.15, panel c). We observe on figure 6.15 a deterioration of the projection efficiency when the size of the density arrays (corresponding to the number of component grid nodes) exceeds the L1-cache memory capacity. The computational time of the Poisson solver depends on the number of unknowns in the linear system and thus increases with more refined grids.

The computational time of one iteration, as well as the proportion of each step of the algorithm, is provided for the PIC-SStd, PIC-HSg with the classical and offset combination technique ($\tau_0 = 3$) in figure 6.16. As expected, the computational time increases when the offset combination technique is considered with more refined grids. Nonetheless, the method remains competitive since the simulation with the offset combination technique (in figure 6.23, bottom panel) is 10 times faster than the simulation with the PIC-SStd scheme (in figure 6.23, top panel).

**Parallelization on uniform memory architecture**

Let us now consider the parallelization on uniform memory architecture. The first hardware considered in this paper is a laptop with the Intel® Core™ i9-10885H CPU incorporating eight cores. In this section, the different parallelization strategies, namely particle sample and component grid work sharing, are investigated separately and compared. The resolution of the electric potential (Poisson) is not parallelized as it is assumed to be negligible (see figure 6.11 a). Since we are interested mainly in performance measures, we consider only few particles per cell in the simulation and thus a poor statistical resolution. This is necessary in order to be able to perform standard PIC simulation on fine meshes due to hardware memory limitations. The memory requirements of the methods are illustrated on panel a) of figure 6.17.

Let us focus on the parallelization strategies for both standard and sparse grid schemes. In this section, we consider the Landau damping test case. The conclusions drawn from the present investigations apply equally to the manufactured solution test case. The strong scaling of the projection, field interpolation and particle pusher up to eight cores is represented on the panels b) and c) of figure 6.17 for different configurations of grid ranging from $32^3$ to $512^3$ grid cells. For the standard scheme, the scalability of the projection seems rather good for grids with less than one hundred nodes in each dimension. For larger grids, the computational efficiency drastically deteriorates for the non-sorted data. This is caused by the size of the thread copies of the density array exceeding the size of the last level (L3) cache memory (see panel c) of figure 6.13). Indeed, since the array data is accessed with a random pattern and the data does not fit

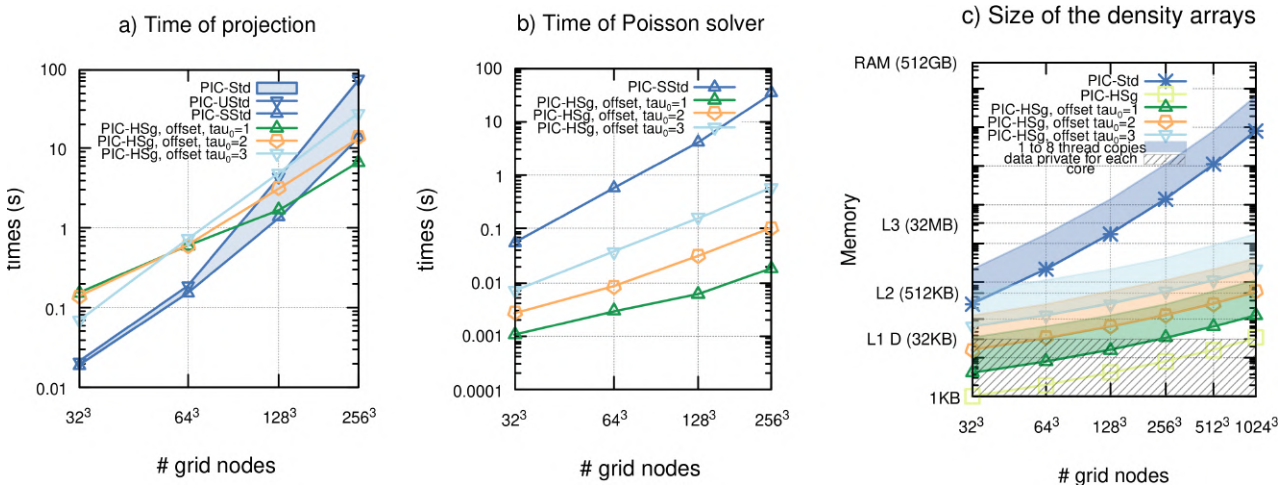


**Figure 6.15.** Diocotron instability: sequential time of the projection step (panel a)) and the Poisson solver for the PIC-SStd, PIC-UStd and PIC-HSg schemes with the offset combination technique (AMD Zen 3 core @2GHz).

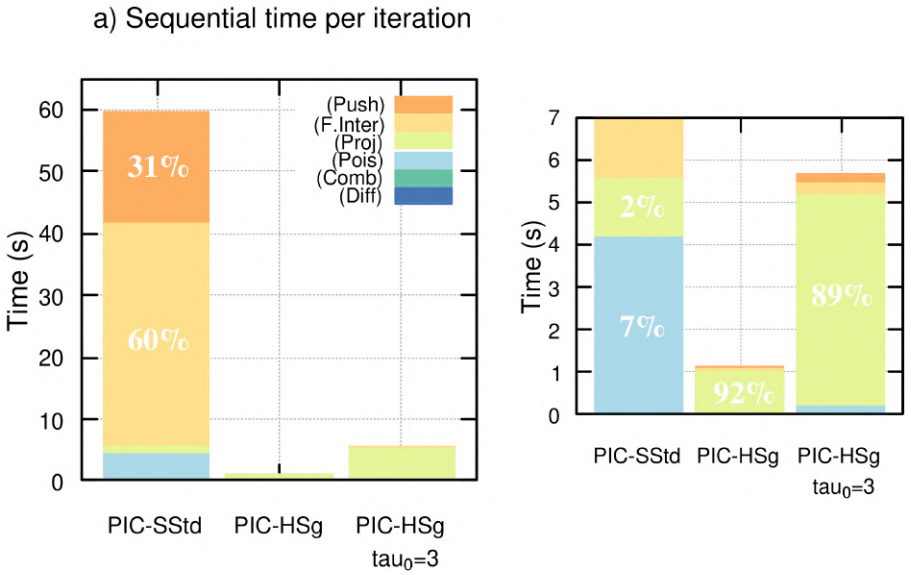


**Figure 6.16.** Diocotron instability: computational time of one iteration on one AMD Zen 3 core @2GHz. All simulations have been performed on a $128^3$ grid and $P_c = 80$. The right panel is a zoom at the bottom of the left panel.

in the cache, the number of cache misses is large and the multiplication of the number of cores increases the memory contention to access the data stored in the RAM. Conversely, the sparse grid scheme (with option 2 from alg. 8) demonstrates a scalability close to ideal. The grids involved in the projection fit in the first level cache memory (private to each core). The size of the data barely exceeds the first level cache memory for discretization equivalent to $1024^3$ grid cells (see panel c) of figure 6.13). The choice of the first option in alg. 8 reveals once more to be the most effective since within the second option the grid data do not fit in the shared L2 cache memory for discretizations above $128^3$ grid nodes. The scalability of the other steps, namely the pusher, combination and field interpolation is less favorable, similar to the standard scheme, but these operations are negligible for the sparse grid scheme due to the reduced number of particles and grid nodes. The loss of scalability for the combination step derived from the access of non-contiguous data within a pole (as explained in section 3.2.2) and small number of poles to parallelize (roughly 16 000). For the pusher of the particles the poor scalability is due to low arithmetic intensity characterizing these operations (nine writes and eighteen reads, as a comparison the projection step has only three reads upon the particle data).

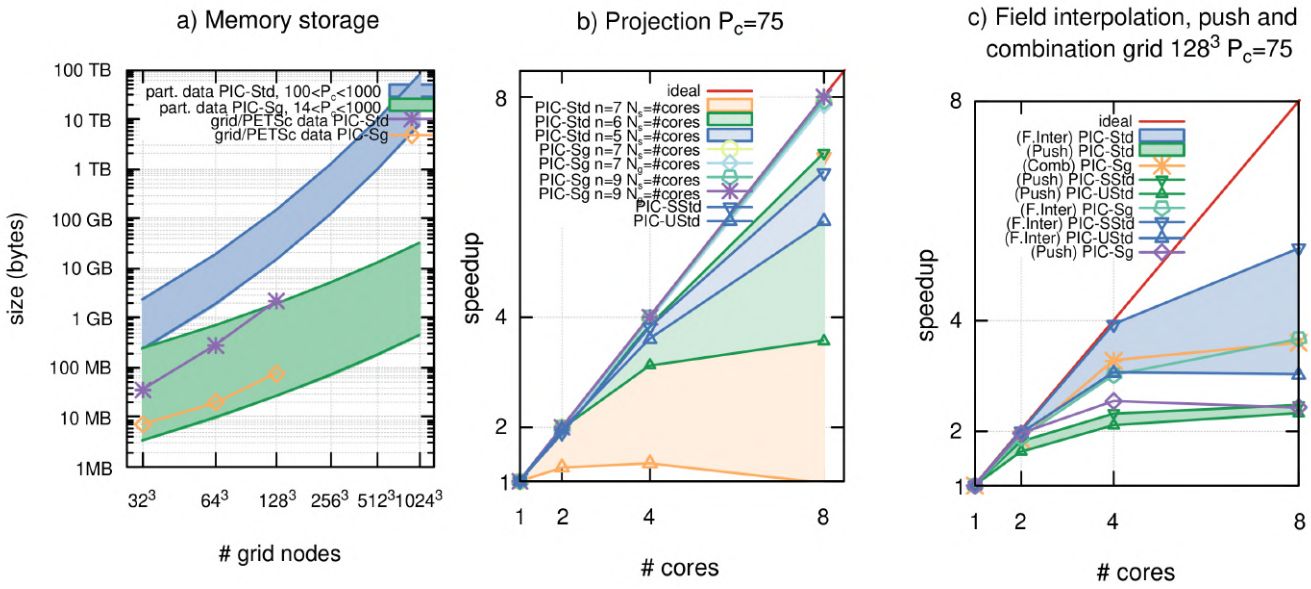


**Figure 6.17.** Landau damping: on panel a), the storage size of data (particle data, grid data, PETSc objects, etc.) is represented as a function of the number of grid nodes and particles per cell. On panel b) and c), the strong scaling of the projection, field interpolation and push steps up to 8 cores on a uniform memory architecture (Intel® Core™ i9-10885H) are represented($n$ is the grid discretization parameter $h_n = 2^{-n}L$). $N_g$ is the number of groups of component grids and $N_s$ the number of samples of particles.

### Parallelization on NUMA architecture

In this section, the different parallelization strategies, namely the particle sample and component grid work sharing, are investigated for the sparse grid scheme. Based on the hardware architecture and the reflections of section 3.2.1, the population of particles is distributed onto the NUMA domains and the component grids onto the cores of a NUMA domain. The set of component grids is replicated to match the number of particle samples and accumulate the density carried by the sample. In the following, we denote by $N_g$ the number of groups of component grids and

$N_s$ the number of samples of particles. A series of simulations is performed in order to assess the strong scaling (when adding cores to a fixed discretization) of a time iteration.

Let us now consider a first non-uniform memory architecture: the two sockets Intel® Skylake 6140 @2.3 GHz of 18 cores (36 cores), and the Landau damping test case. The numerical results of the NUMA parallelization strategy on the projection, as well as other configurations and the other most costly operations (field interpolation, particle pusher, differentiation) up to 36 cores for grids with $128^3$ and $256^3$ cells, 500 particles per cell are presented on left panel of figure 6.18. The parallelization strategies providing the best efficiency on the projection are the ones that do not stress the memory bandwidth: configurations with two samples of particles ( *i.e.* as many as NUMA domains) and the one without any sampling of particles. Indeed, when the number of samples of particles exceeds the number of NUMA domain, the threads access the memory in a competing way and the scalability is deteriorated. Although the scalability of the projection is close to ideal on 36 cores, the density projection remains by far the most costly operation within the scheme. As a result, the limited scalability of other steps such as the pusher, the differentiation or the sequential execution of the Poisson solvers has a marginal impact on the total speedup (of 27.3).

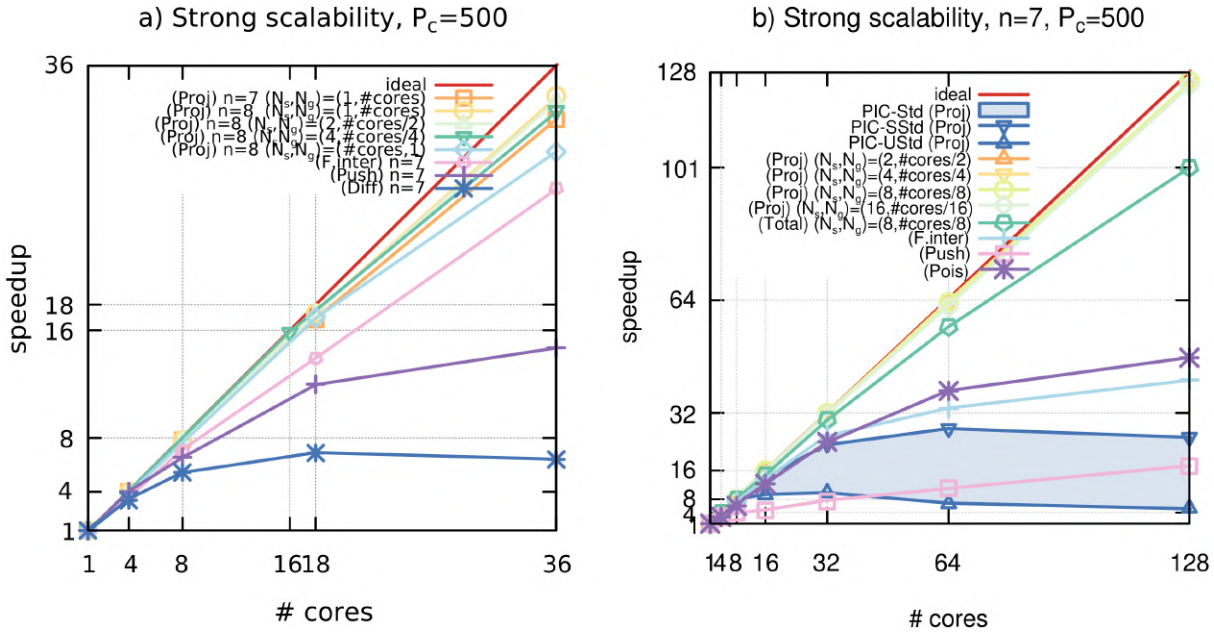


**Figure 6.18.** Landau damping: strong scaling of the most costly steps of the PIC-HSg scheme and the projection of the PIC-Std scheme up to 36 (panel a) and 128 (panel b) cores. $N_g$ is the number of groups of component grids and $N_s$ the number of samples of particles. Different configurations of groups of grids and samples of particles are represented (# cores denotes the number of cores). For the other steps than the projection, the configuration $(N_s, N_g)$ is not given since all provide similar results. $n$ is the grid discretization parameter $h_n = 2^{-n}L$. Computations carried out on either two Intel® Skylake 6140 CPUs (left) or two AMD EPYC™ 7713 CPUs (right).

A second non-uniform memory architecture is considered: the two sockets AMD EPYC™ 7713 *Milan* with 64 cores per socket (128 cores), still for the Landau damping test case. With this increased number of cores, the sequential Poisson solver used in the sparse-PIC methods is no longer negligible. Therefore, the parallelization strategy based on the component grid work sharing is implemented. The linear systems issued from the different component grids are distributed onto the cores. Though this strategy is not optimal, since it entails a load imbalance,

it permits to reduce the cost of this step of this algorithm and obtain a good scalability of the sparse-PIC method for tens of cores. A series of simulations is performed in order to assess the strong scaling of the projection, which is by far the most costly operation of the sparse grid scheme as evidenced by the computational time of an iteration on one core (see panel b) of figure 6.11). The projection remains the most costly step (nearly 80 % of the time iteration) of the PIC-HSg scheme on one hundred of cores. The scalability of the PIC-HSg most costly steps and the PIC-UStd, PIC-SStd projection steps up to 128 cores for grids with $128^3$ cells, $P_c = 500$ particles per cell are presented on panel b) in figure 6.18. The configuration that gives the best results and achieves a speedup of 126 (parallel efficiency of 0.984) is the one with eight samples of particles (as many as NUMA domains). For different configurations of samples, the speedup is still close to ideal, which lead to the conclusion that the parallelization strategy has not reached its scalability limit with one hundred of cores. The total speedup of one time iteration is 101 with 128 cores. For the standard scheme, the scalability of the projection above 32 cores is poor; the speedup ranges between 10 without sorting and 28 with sorted particles. The substantial difference between the standard and the sparse-PIC schemes can be explained by two observations: first, as already highlighted, the particle sample work sharing parallelization strategy on its own may lead to increasing the memory contention when a large number of cores shares the same sample; second, the sparse grid reconstructions offer a more favorable trade-off between memory accesses and compute operations, resulting in an increased arithmetic intensity. Indeed the charge of one particle is deposited onto each component grid, the array used to store these grids being nursed in the first level of cache memory, the contention of the RAM is alleviated and the scalability of sparse-PIC algorithms is favored.

The same conclusions than the uniform memory architecture can be drawn for the pusher and field interpolation steps. The scalability of the pusher is limited by the memory bandwidth, and thus increases with the number of NUMA domains used. The scalability of the resolution of Poisson is deteriorated by the load imbalance between the grids (complexity, anisotropy), but this step accounting for a marginal part of the total computational time, this coarse parallel implementation is sufficient.

Finally, the parallelization of the diocotron instability test case is investigated. The strong scaling of the projection, as well as one total sparse-PIC iteration up to 128 cores are represented for the PIC-HSg scheme with the classical and the offset combination technique in figure 6.19. The grid discretization and particle per cell parameters are $n = 7$, $P_c = 80$. The sizes of the component grids involved in the offset combination technique are provided in table 6.6. The scalability is deteriorated with the offset combination technique and the deterioration depends on the refinement parameter $\tau_0$ (the projection scalability is deteriorated from 112 to 95 for $\tau_0$ ranging from 0 to 3). Increasing the parameter $\tau_0$ results in a smaller number of component grids involved in the combination, each with more grid nodes (see table 6.6). The deterioration of the efficiency is the result of the more refined grids that do not fit in the L1-cache. Indeed, for $\tau_0 = 2$ only 19% of the component grids fit in the L1-cache, and for $\tau_0 = 3$ none of the component grids fit in the L1-cache. The speed-up of the total iteration is also deteriorated because the grids operations (less scalable) are less negligible due to the few number of particles per cell.

### 6.3.2 Single GPU performances

#### Performance metrics

The roofline performance model [53, 112] is a method for determining the maximum performance of an algorithm running on a given hardware. It is used to asses the performance of the different implementations, it is elaborated on metrics provided by the Nvidia Nsight Compute profiler. The theoretical instruction throughput performance is obtained from the number

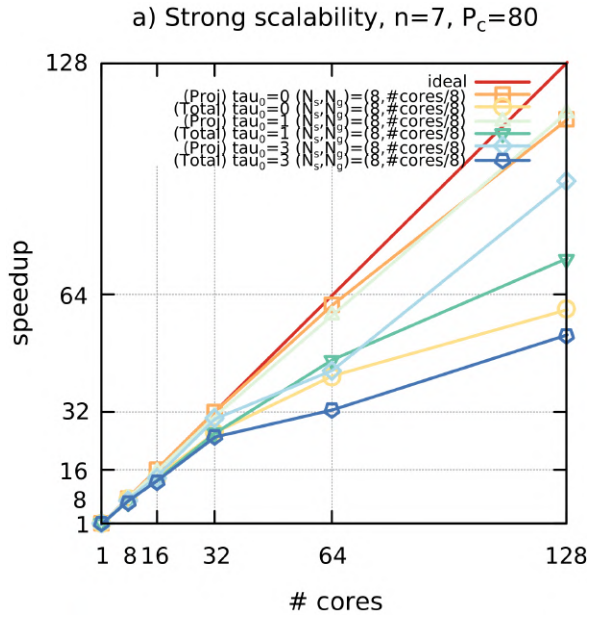


**Figure 6.19.** Diocotron instability: strong scaling of the projection and the total iteration up to 128 cores for the PIC-HSg scheme with the classical or the offset combination technique. Different configurations of groups of grids and samples of particles are represented (# cores denotes the number of cores). Computations carried out on two AMD EPYC™ 7713 CPUs.

**Table 6.6.** Diocotron instability: component grid sizes for the offset combination technique, number of component grids fitting in the L1 cache related to the total number of component grids and speed-ups of the projection on the two socket AMD EPYC™ 7713 CPU platform.

| $\tau_0$ | Grid sizes | Grids in L1-cache | Speed-up projection |
|---|---|---|---|
| 0 | 4KB,2KB,1KB | 100% (64/64) | 112.3 (100%) |
| 1 | 16KB,8KB,4KB | 100% (46/46) | 114.3 (100%) |
| 2 | 65KB,32KB,16KB | 19% (6/31) | 100.3 (89.3%) |
| 3 | 262KB,131KB,65KB | 0% (0/19) | 96.2 (85.6%) |

of cores, the clock frequency of the GPU and the number of instructions per cycle (1 operation for multiplication or addition and 2 operations per cycles for Fused Multiply-Add (FMA) instructions):

$$InstructionThroughput[GFlop/s] = nb.\,of\ cores * instruction\ per\ cycle * clock frequency. \tag{6.34}$$

It provides the upper bound of the computing capacity of the hardware. Nonetheless, in most cases the performance of the hardware is limited by its memory bandwidth. Therefore, the theoretical peak performance of the hardware is defined by:

$$PeakPerformance[GFlop/s] = \min(InstructionThroughput, Bandwidth * ArithmeticIntensity), \tag{6.35}$$

where the arithmetic intensity (AI), a metric characteristics of the algorithm, is defined by:

$$ArithmeticIntensity = \frac{nb.\,of\ arithmetic\ operations[FLOP]}{DataRead[B] + DataWritten[B]}. \tag{6.36}$$

The program is said to be compute-bounded when the minimum in the right side of equation (6.35) is the instruction throughput. In this case, peak performance of the hardware can be achieved. Otherwise the program is memory-bounded, that is the theoretical performance of the hardware is deteriorated by the memory accesses and relies upon the memory bandwidth efficiency of the hardware (see figure 6.20)*. The peak performance metric shall be compared to the effective performance of the algorithm, defined by:

$$EffectivePerformance[GFlop/s] = \frac{nb.\,of\ arithmetic\ operations[FLOP]}{time\ of\ execution[s]}. \tag{6.37}$$

A new metric is introduced computed as the ratio of the measured FLOP related to the measured Bytes transferred to and from the different memories: DRAM, L2 and L1 caches. Due to the analogy of the Arithmetic Intensity characterizing the algorithm, the new metrics are referred to as DRAM-AI, L2-AI and L1-AI. *E.g.* the DRAM-AI is defined by:

$$DRAM\text{–}AI = \frac{nb.\,of\ arithmetic\ operations[FLOP]}{DataReadFromDRAM[B] + DataWrittenIntoDRAM[B]}. \tag{6.38}$$

Based on this metric, a theoretical peak performance for each memory can be defined; *e.g.* the DRAM Peak Performance (DRAM-PP) is defined by:

$$DRAMPeakPerformance[GFlop/s] = \min(IntructionThroughput, Bandwidth * DRAM\text{–}AI), \tag{6.39}$$

**Performance analysis of charge density accumulation algorithms**

The charge density accumulation is the most predominant step in sparse-PIC methods with a computational cost counting for roughly 90% of the simulation (for a sequential execution on CPU). Therefore an efficient parallelization of the accumulation is capital to achieve significant performance on accelerators. In section 4.3, two different GPU implementations have been proposed for the charge deposition. These implementations shall now be investigated and compared. Let us consider the sparse-PIC method for the 3D-3V Landau damping in a configuration equivalent to a $128^3$ Cartesian grid (*i.e.* $n = 7$) and 500 particles per cell ($P_c$) with respect to the standard method, amounting to 1.3E+7 particles ($N$). As a comparison, the standard method on the Cartesian grid would require $10^9$ particles.

*These metrics are available with the following command line:

```
>ncu --metrics dram__bytes.sum,lts__t_bytes.sum,l1tex__t_bytes.sum,
  sm__sass_thread_inst_executed_op_dadd_pred_on.sum,
  sm__sass_thread_inst_executed_op_dfma_pred_on.sum,
  sm__sass_thread_inst_executed_op_dmul_pred_on.sum,
```

where *dram__bytes.sum* provides the data transfer with the device memory, *lts__t_bytes.sum* the data transfer with the L2 cache and *l1tex__t_bytes.sum* the data transfer with the L1 cache. The last three metrics provide the total number of double precision floating point operations (addition, multiplication and fused multiply-add operations).

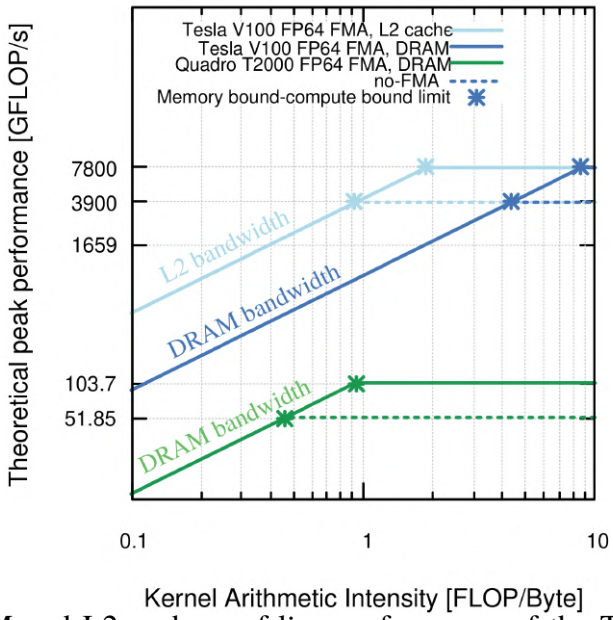


**Figure 6.20.** DRAM and L2 cache roof-line performance of the Tesla V100 and Quadro T2000.

### Impact of the number of gangs and vector size on the CPU-inherited and GPGPU-specific implementations

In this section, the gang, worker and vector length configurations are investigated. The number of workers is usually advised to be set to one [76], as Nvidia compilers do. The vector size shall be a multiple of the number of compute cores within a SM (32 for Nvidia hardware). The computational time of the charge accumulation step on the Quadro T2000 with different configurations of gangs and vector size is provided in figure 6.21 and the middle panel of figure 6.23. The same trend is observed on the second platform (Tesla V100). The number of gangs (or thread blocks in CUDA terminology) is a capital tuning to achieve good performance. For the CPU-inherited implementation, the number of gangs is optimal when it is equal to the number of SMs. The performance drastically decreases when a too large number of gangs is used. This is the consequence of the reduction operation performed on the three-dimensional grid. For the GPGPU-specific implementation introduced in this paper, the performance increases with the number of gangs, facilitating the interleave stream strategy to mask the memory latency. Therefore the number of gangs for the GPGPU specific implementation may be delegated to the compiler. The execution time increases significantly when the vector length is superior to 128 but no significant differences have been observed for vector size equal to 128 and below, so this parameter is delegated to the compiler (usually 128) in the following of the paper.

### Memory management and kernel analysis

Let us now investigate specifically the kernels of the CPU-inherited and the GPGPU-specific implementation. The optimal configuration of thread blocks (gangs) and threads within a thread block (vector size) for each method is chosen according to the following policy: the number of gangs is set to the number of SM for the CPU-inherited implementation; while this choice is delegated the compiler for the GPGPU implementation. The number of threads (vector size) is set to 128 for both methods. In the CPU-inherited implementation, one kernel (corresponding to the accumulation onto one component grid) is considered, *e.g.* the kernel associated to the first component grid (of level $\mathbf{l} = (1, 1, 7)$). The same conclusions can be drawn for all kernels by

| Implementation | Gangs | Vector length | Time [s] |
|---|---|---|---|
| CPU-inherited | 16 | 128 | 6.22 |
| * | 32 | 128 | 7.19 |
| * | 64 | 128 | 8.60 |
| * | 128 | 128 | 11.22 |
| GPGPU-specific | 16 | 128 | 1.49 |
| * | 128 | 128 | 0.92 |
| * | 1024 | 128 | 0.88 |
| * | 65,535 | 128 | 0.87 |
| * | 65,535 | 256 | 1.0 |
| * | 65,535 | 1024 | 1.42 |
| * | 65,535 | 64 | 0.85 |

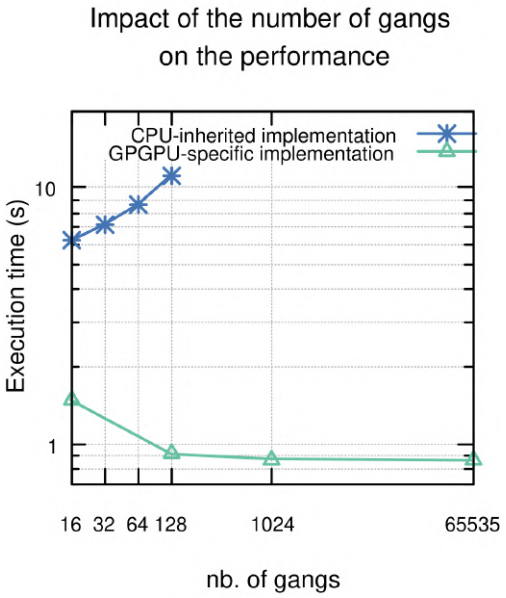


**Figure 6.21.** Impact of the number of gangs (thread blocks) and vector size (thread) on the performance of the charge accumulation algorithm for the Quadro T2000.

extension. The kernel is compared to the kernel of the GPGPU specific algorithm, implementing the accumulation of the particle properties onto all the component grids.

First, the arithmetical intensity (AI), the peak and effective performances of the kernels involved in both the CPU-inherited and the GPGPU-specific implementations are investigated on table 6.22. The AI is determined with the total number of FLOP relative to the total amount of data to be read and written (see equation 6.38). The peak performance is determined by the Roofline model from the AI and the DRAM bandwidth of the device. It is compared to the effective performance measured thanks to the profiler. Although the kernel AI of the two implementations are equivalent, we observe a significant discrepancy in the effective performance, the effective performance of the GPGPU implementation being roughly twice its theoretical peak performance. This surprising result is the consequence of the cache memory reuses occurring in the GPGPU implementation.

**Table 6.22.** Charge accumulation algorithm performance on Tesla V100 with $128^3$ grids and $P_c = 500$. Kernel CPU-inheirted corresponds to the kernel of the first component grid ($\Omega_{h_{(1,1,7)}}$) for the CPU-inherited algorithm. Kernel GPGPU-specific corresponds to the kernel of all the component grids for the GPGPU-specific algorithm.

| Kernel | AI | Peak performance | Effective performance (%) |
|---|---|---|---|
| CPU-inherited | 0.12 | 107.6 GFLOP/s | 58.6 GFLOP/s (54%) |
| GPGPU-specific | 0.14 | 125.5 GFLOP/s | 213 GFLOP/s (170%) |

Let us now consider a more thorough analysis of the kernels to better understand the memory managements of the methods. In the following we consider the metrics based on the different memories and introduced previously as DRAM-AI, L2-AI and L1-AI. These metrics are computed as the ratio of the total number of FLOP related to the effective amount of data handled by a specific memory at the execution (DRAM data, L2-cache data on table 6.24). All the characteristics and performance of the charge accumulation algorithms are provided on the

table 6.24. The relative amount of L1-cache and L2-cache hits are represented on the left panel of figure 6.23.

First, it is manifest that the GPGPU-specific kernel has a better L2-cache efficiency (more than 99% of L2-cache hit) than the CPU-inherited kernel (roughly 30% of L2-cache hit). We observe a non efficient use of the cache memory for the CPU-inherited method since more than one half of the L1 cache requests has to be fetched from the DRAM memory (and less than a third has to be fetched in the L2-cache). The amount of data transferred between the device and the DRAM memory is dramatically reduced for the GPGPU implementation, emphasizing the success of the strategy targeted by these implementation. However, the number of L1-cache hits remains significantly low for the GPGPU-specific implementation, this being the result of the irregular and non coalesced memory accesses genuine to the method.

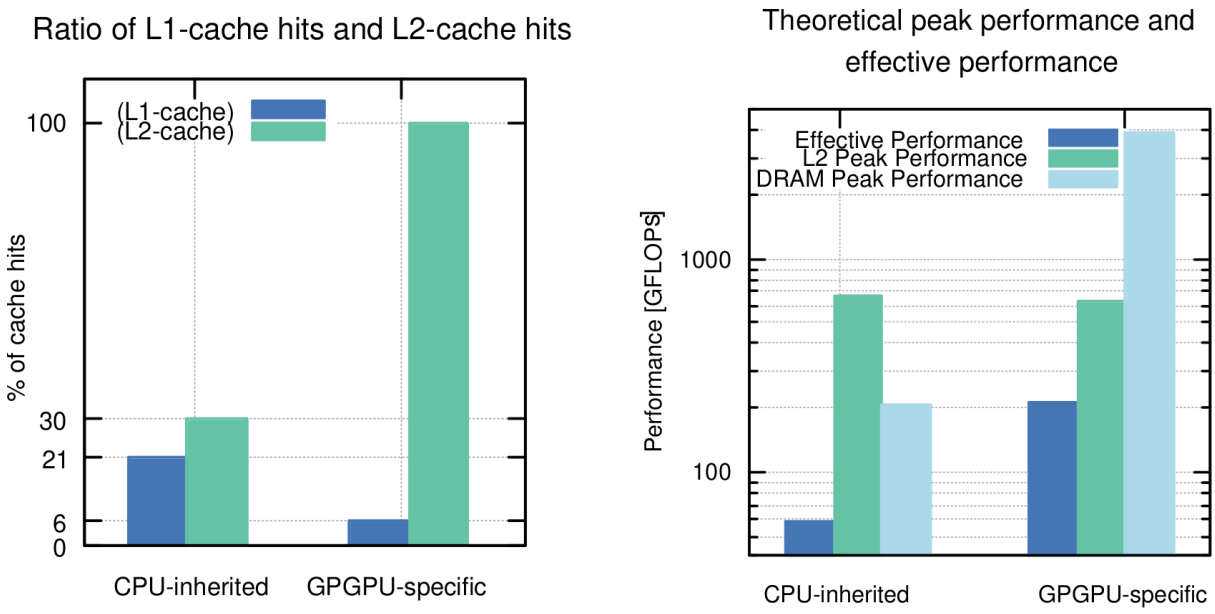


**Figure 6.23.** Relative amount of L1-cache and L2-cache hits (left panel) and theoretical/effective performance (right panel) for kernels running on the Tesla V100. Kernel CPU-inherited corresponds to the kernel of the component grid $\Omega_{h_{(1,1,7)}}$ for the CPU-inherited algorithm. Kernel GPGPU-specific corresponds to the kernel of all the component grids for the GPGPU-specific algorithm.

The theoretical peak performance specific to the DRAM (DRAM-PP) can be determined from the DRAM-AI of the kernel and compared to the effective performance. A high DRAM-PP compared to the theoretical Peak Performance means that a lot of data is loaded from the cache memory. For the Tesla V100, the kernel of the CPU-inherited algorithm is memory-bounded according to this metric, with a DRAM peak performance of roughly 206 GFLOP/s (see table 6.24) and effective performance of 58.6 GFLOP/s. The kernel of the GPGPU algorithm is compute-bounded according to this metric, with a DRAM peak performance of 3,900 GLOP/s (since no FMA operation is performed) and an effective performance of 213 GFLOP/s. The DRAM-PP is roughly 30 times higher than the theoretical Peak Performance, emphasizing on the good cache memory management of the method. The kernel is actually far from being compute-bounded and most of the memory transfers take place in the L2-cache and not the DRAM. Indeed, most of the data necessary in the kernel is fetched from the caches (L1 and mostly L2 layers) and a significantly small proportion of data is transferred from the main memory (The number of DRAM memory accesses represent 0.1% of those of L2-cache memory). For this reason, the DRAM-AI and DRAM-PP metrics are not relevant for the GPGPU-specific

implementation. For the Quadro T2000, the same conclusions can be drawn. Nonetheless, the effective performance of the GPGPU implementation is closer to the theoretical one (72%).

Let us now investigate the roofline model based on the L2-cache data transfers. The L2-AI of the two implementations are equivalent (of roughly 0.15) and so does the L2-Peak Performance. The effective performance is about 33% of the L2-PP for the GPGPU implementation and 8% for the CPU-inherited implementation. Because the peak performance is not reached in both cases, the implementations are rather latency-bound and the performance is limited by the irregular and non coalesced memory accesses.

Despite the randomness of the memory accesses increasing the latency between the device and its memory, a significant speed-up is achieved both on the Quadro T2000 and the Tesla V100. Although the computing capacity of the Quadro T2000 is poor, a speed-up close to 12 is reached in double precision, meaning that a GPGPU execution is more efficient than parallelization with the 8 cores Intel® Core™ i9-10885H of the laptop host. The Tesla V100 achieves a speed-up of more than 100 on the charge deposition in double precision, which is significant for such an inexpensive hardware. As a comparison, in [49] a speed-up of 126 is achieved for the charge accumulation on a hardware consisted of 2 sockets AMD EPYC™ 7713 *Milan* with 128 cores.

In conclusion, the detrimental effects of the irregular non coalesced memory accesses are mitigated by a good locality of the data, resulting in an efficient L2-cache memory management, as well as an efficient latency masking strategy based on a large number of gangs associated to clusters of particles.

**Investigation of the whole scheme**

Let us consider the parallelization of all the steps of the sparse-PIC scheme. The 3D-3V Landau damping is run on two platforms, both with a $128^3$ grid: on the one hand the Intel® Core™ i9-10885H with 1 core is compared to the Quadro T2000 run for 100 particles per cell; on the other hand the Intel® Skylake with 1 core is compared to the Tesla V100 run for 500 particles per cell. In the following, the GPGPU implementation is considered for runs on GPUs.

First, the different strategies for the resolution of the Poisson equation are investigated, namely: either solving the linear systems issued from the discretization of the problem on the component grids one after the other or gathering all the problems into a single (by block) linear system. The first strategy is considered with the CuSolver dense library and the second one with the AMGX library. The computational time of both strategies is represented on figure 6.26 for different grid discretization (ranging from a $32^3$ grid to a $512^3$ grid). For coarse discretizations, the first option is the most efficient despise the dense format of the solver. This is caused by the very small size of the linear system, which are roughly of one-dimension complexity. When the grid is refined, the second option along with the AMGX solver becomes less time consuming as a result of the increasing work load favorable to the GPU.

The mean computational time out of 5 iterations is represented on figure 6.25 for the two platforms, namely the Quadro T2000 associated to the Intel® Core™ i9-10885H CPU and the Tesla V100 associated to the Intel® Skylake CPU. The speed-ups obtained on the GPU devices with respect to the host runs for each step, as well as the total iteration, are provided in table 6.27. The GPU execution time is compared to a sequential CPU execution with the MUMPS library together with a GNU compiler (see [49] for more details).

The field interpolation and the particle pusher have the most significant speed-ups (of roughly 130 on the Tesla V100). Indeed, these steps, consisting in basic operations working on a large amount of data are well suited for SIMT processing of GPUs. The speed-ups of the combination, differentiation and Poisson solver are less substantial, mostly because of the insufficient kernel arithmetic intensity. The operations within these steps are made on a very small amount of data (component grid nodes) and the sequential computational time is insignificant, even on 1 core (the three steps represent less than 0.3% on the Intel® Skylake run with 1 core). The

**Table 6.24.** Charge accumulation algorithm characteristics and performance (DRAM and L2 cache Roofline model) on the Tesla V100 (a) and the Quadro T2000 (b), with a $128^3$ grid and $P_c = 500$. Kernel CPU corresponds to the kernel of the component grid $\Omega_{h_{(1,1,7)}}$ for the CPU-inherited algorithm. Kernel GPU corresponds to the kernel of all the component grids for the GPGPU-specific algorithm. DRAM, L2 cache and L1 cache data correspond to the amount of data transferred with respectively the main memory (DRAM), the L2 cache or the L1 cache during the execution; *e.g.*, the amount of L2 cache hit data is $L2\,cache\,data - DRAM\,data$. DRAM-AI and L2-AI correspond to metrics of the kernel based on the measured Bytes transferred to and from the different memories (DRAM, L2). The peak performance is defined with the DRAM-AI and L2-AI metrics and the effective performance is defined by equation (6.37).

(a) Tesla V100

| Kernel | Gangs (blockidx%x) | Vector length (threadidx%x) | DRAM data | L2-cache data | L1-cache data |
|---|---|---|---|---|---|
| CPU | 80 | 128 | 4.57GB | 6.59GB | 8.43GB |
| GPU | 65,535 | 128 | 326MB | 215GB | 229GB |

| Kernel | L2-cache hit (%) | L1-cache hit (%) |
|---|---|---|
| CPU | 2.02GB (30%) | 1.84GB (21%) |
| GPU | 214.6GB(99.8%) | 14GB (6%) |

| Kernel | FLOP | DRAM-AI | DRAM-PP | Effective performance | Execution time (all grids) | Speed-up |
|---|---|---|---|---|---|---|
| CPU | 1.05E+9 | 0.23 | 206 GFLOP/s | 58.6 GFLOP/s (28%) | 0.0179 s (1.4 s) | 11.7 |
| GPU | 3.28E+10 | 101 | 3,900 GFLOP/s | 213 GFLOP/s (5%) | 0.1540 s | 106 |

| Kernel | L2-AI | L2-PP | Effective performance |
|---|---|---|---|
| CPU | 0.16 | 672 GFLOP/s | 58.6 GFLOP/s (8%) |
| GPU | 0.15 | 630 GFLOP/s | 213 GFLOP/s (33%) |

(b) Quadro T2000

| Kernel | Gangs (blockidx%x) | Vector length (threadidx%x) | DRAM data | L2-cache data | L1-cache data |
|---|---|---|---|---|---|
| CPU | 16 | 128 | 4.45GB | 5.92GB | 7.5GB |
| GPU | 65,535 | 128 | 328MB | 215GB | 229GB |

| Kernel | FLOP | DRAM-AI | DRAM-PP | Effective performance | Execution time (all grids) | Speed-up |
|---|---|---|---|---|---|---|
| CPU | 1.0E+9 | 0.226 | 25.4 GFLOP/s | 12.3 GFLOP/s (48%) | 0.082 s (7.04 s) | 1.47 |
| GPU | 3.28E+10 | 100 | 51.85 GFLOP/s | 37.4 GFLOP/s (72%) | 0.874 s | 11.8 |

charge density accumulation, which is the most time consuming operations of the scheme have a significant speed-up, of roughly 100 on the Tesla V100. It results in a total speed-up for the whole iteration of about 12 on the Quadro T2000 and 95 on the Tesla V100.

### Comparison between GPGPU-specific sparse-PIC and sequential standard PIC

The sparse-PIC GPGPU-specific implementation are compared to a sequential CPU implementation of the standard scheme. The architecture considered for the standard PIC simulations consists of a single computational server equipped with two AMD EPYC™ 7713 *Milan* CPUs and RAM memory of 512GB. The standard scheme runs with a particle population sorted beforehand, i.e. the particle sorting time is not taken into account for the total execution time, which provides an upper bound of the method efficiency. As a first investigation, let us consider first the 3D-3V Landau damping on a standard PIC method configuration of a $128^3$ grid with 128 particles per cell, which is considered as a reference. sparse-PIC discretizations carried out on a $128^3$ grid (resp. a $512^3$ grid) with 128 (resp. 500) particles per cell are considered

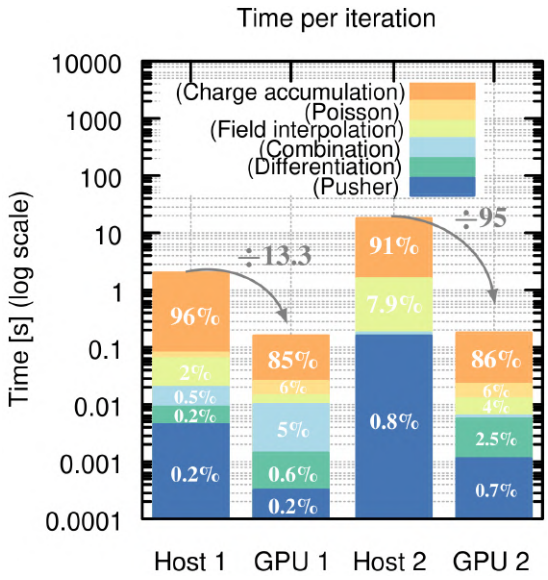


**Figure 6.25.** Time per iteration of the CPU and GPU sparse-PIC implementation. Host 1 and GPU 1 run with 100 particles per cells while host and GPU 2 run with 500 particles per cell. Host 1 is the Intel® Core™ i9-10885H with 1 core; GPU 1 is the Quadro T2000; Host 2 is the Intel® Skylake with 1 core; GPU 2 is the Tesla V100.

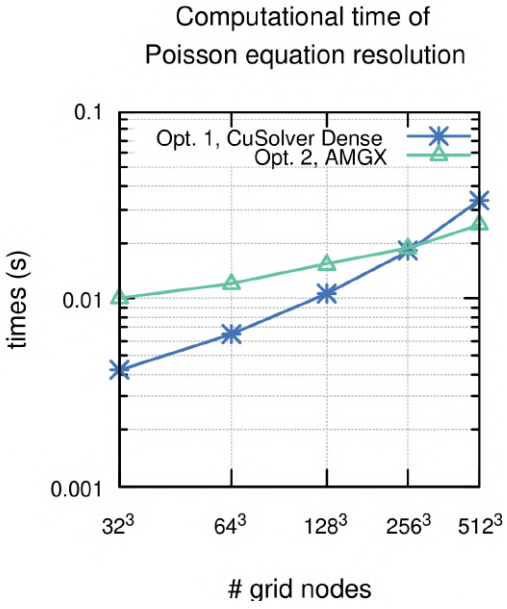


**Figure 6.26.** Computational time of the Poisson equation resolution as a function of the grid discretization. In option 1, all the systems issued from the discretization of the Poisson equation are solved one after the other. In option 2, all the problems are gathered into a single (by blocks) linear system.

**Table 6.27.** Speed-ups (and relative time) of the steps for the Quadro T2000 (GPU 1) and Tesla V100 (GPU 2). GPU 1 runs with 100 particles per cells while GPU 2 runs with 500 particles per cell. Poisson speed-up is provided with respect to a sequential CPU execution with MUMPS library.

| | Hardware | (Proj) | (F. Inter) | (Push) | (Comb) |
|---|---|---|---|---|---|
| Host | Intel® Core™ i9 | 1 (96%) | 1 (2%) | 1 (0.2%) | 1 (0.5%) |
| Device | Quadro T2000 | 14.1 (85%) | 12.33 (3.2%) | 13.7 (0.2%) | 1.2 (5%) |
| Host | Intel® Skylake | 1 (91%) | 1 (7.9%) | 1 (0.8%) | 1 (0.2%) |
| Device | Tesla V100 | 107 (86%) | 139 (4%) | 133 (0.7%) | 21.8 (0.8%) |

| | Hardware | (Diff) | (Pois) | (Total) |
|---|---|---|---|---|
| Host | Intel® Core™ i9 | 1 (0.2%) | 1 (1.1%) | 1 (100%) |
| Device | Quadro T2000 | 4.44 (0.6%) | 1.4 (6%) | 12.5 (100%) |
| Host | Intel® Skylake | 1 (0.1%) | 1 (0.1%) | 1 (100%) |
| Device | Tesla V100 | 29.8 (2.5%) | 1 (6%) | 95 (100%) |

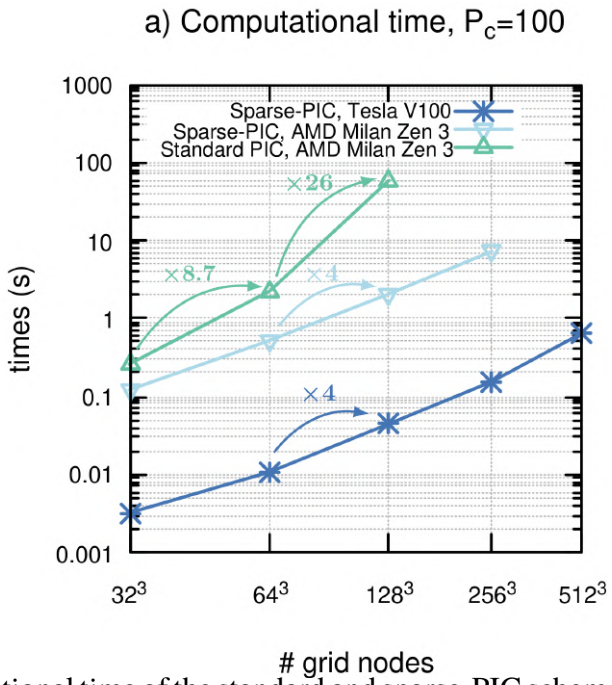


**Figure 6.28.** Computational time of the standard and sparse-PIC schemes for grids ranging from $32^3$ to $512^3$. Standard simulation is performed on the AMD Zen 3 core @2 GHz. Sparse-PIC simulations are performed on the AMD Zen 3 core and on the Tesla V100.

**Table 6.29.** Configurations and results of the 3D-3V Landau damping test case. AMD EPYC™ 7713 *Milan* with AMD ZEN 3 core @2.0 GHz and Intel® *Skylake* core @2,3 GHz are considered.

| Method | Figure | $h_n/P_c$ | Particles (N) | Memory | Time (1 iter.) |
|---|---|---|---|---|---|
| Standard (CPU AMD *Milan*) | 6.34 (left) | $2^{-7}L/128$ | 2.6E+8 | 19GB | 139.6 s |
| Sparse-PIC (CPU Intel® *Sylake*) | | $2^{-7}L/128$ | 3.45E+6 | 248MB | 5.4 s (÷25.8) |
| Sparse-PIC (CPU AMD *Milan*) | | $2^{-7}L/128$ | 3.45E+6 | 248MB | 2.6 s (÷53.7) |
| Sparse-PIC (GPU Tesla V100) | 6.34 (right) | $2^{-7}L/128$ | 3.45E+6 | 248MB | 0.05 s (÷2792) |
| Sparse-PIC (GPU Tesla V100) | | $2^{-9}/500$ | 9.01E+7 | 6.48GB | 1.739 s |

as a comparison. It results in a substantial reduction of the number of particles (two orders of magnitude) and thus of the memory footprint (from GB to MB). Besides the significant gain on the memory footprint offered by the sparse-PIC method, the computational time is also reduced (by one order of magnitude in that configuration, see table 6.29).

The computational time of the standard method on a single CPU core, the sparse-PIC method on a single CPU core and the GPGPU implementation are represented as a function of the grid discretization on figure 6.28. In the following, the sequential executions of the sparse-PIC method are provided for the most efficient algorithm on CPU, which is the algorithm 9. When the mesh is refined in the standard method, the number of particles and the number of Cartesian grid nodes are multiplied by 8 and therefore so does the computational time. Nonetheless, a significantly larger increase is observed when the mesh is refined from a $64^3$ grid to a $128^3$ grid. Actually, it is explained by the non-locality of the data: the simulation data for the $128^3$ simulation no longer fit on a single NUMA node of the AMD EPYC computational server. The computational time of both the CPU and GPU sparse-PIC simulations is multiplied by 4 when the mesh is refined, which is twice as much as the standard method.

### 6.3.3 Qualitative results of the three-dimensional test cases

The three-dimensional representation of the electron density is provided on figure 6.30 for the 3D-3V Landau damping test case. The results for the PIC-Std (left) and PIC-HSg (right) schemes are compared. For each configuration the mean number of particles per cell is $P_c = 128$. The reduction of the statistical noise is manifest on the density representation for the sparse grid scheme. In addition, this improvement, as compared to the standard method, is achieved with 75 times fewer particles in the simulation: a comparable mean number of particles per cell yielding here $N = 2.6\text{E+}8$ for PIC-Std, $N = 3.45\text{E+}6$ for PIC-HSg.

In [48, 49, 88], new sparse grid reconstructions have been introduced to improve the approximation of solutions with localized support and fine structures. As an illustration, we investigate the three-dimensional diocotron test in which instabilities caused by a magnetic field lead to the formation of a discrete number of vortices. These characteristics are used to emphasize the weaknesses of standard sparse-PIC reconstructions.

The three dimensional representation of the electron density is represented on figures 6.31, 6.32 and 6.33 at different times, with a grid discretization $h_n = 2^{-7}$, the mean number of particles per cell being $P_c = 20$. Both the PIC-Std scheme, PIC-HSg scheme with classical combination technique and PIC-HSg scheme with the offset (truncated) combination technique are represented. The offset combination technique [48] and truncated combination technique [88] are derivations of the classical combination technique, consisting in the elimination of the most anisotropic grids from the combination. With the offset combination technique, there are less, but more refined, component grids in the combination. As a result, more particles are required to obtain a good statistical resolution and the grid operations such as the resolution of Poisson equation are more costly. The sparse grid scheme with the classical combination technique fails to reproduce the fine structure of the density. One can see that the sparse grid reconstruction has a tendency to flatten the steep gradients of the solution. The offset reconstruction provides significant improvements of the sparse grid reconstructions and a mitigation of the statistical noise in comparison to the standard approach despite a significantly reduced number of particles. Even for such an unfavorable test case, sparse-PIC methods embedding the offset sparse grid reconstructions provides a gain over the standard method on memory footprint, as well as computational time, as it has been already highlighted in the previous sections. As already pointed out in [48], sparse grid reconstructions do not preserve the positivity of the solution. Indeed, on figure 6.32, the charge density reconstructed on the Cartesian grid is not non-negative as it should be. Nonetheless, it is important to emphasize that this reconstruction is used only for post processing verification in the PIC-HSg and PIC-NSg schemes. The charge density used to compute the electric field is defined on each component grid and this quantity being positivity preserving.

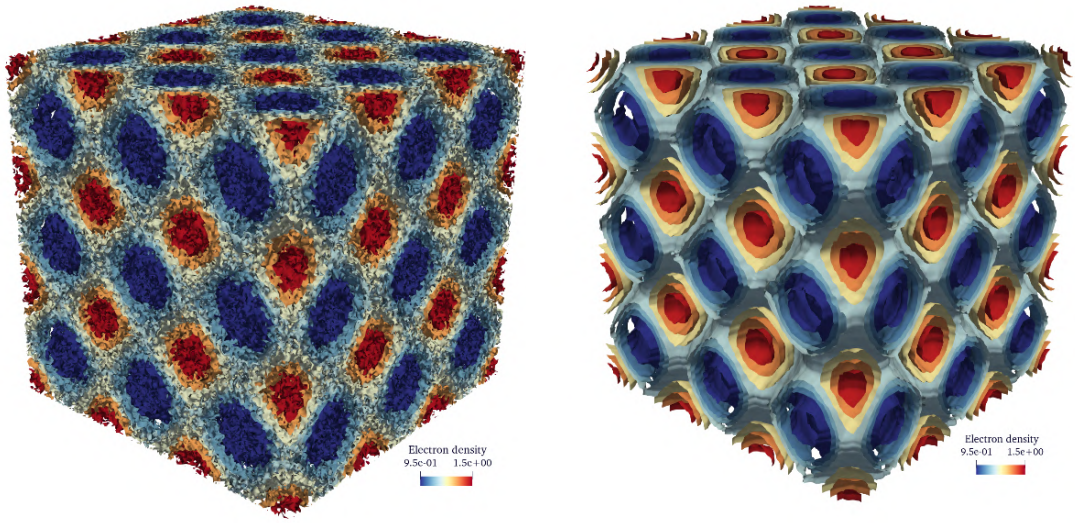


**Figure 6.30.** Representation of the electron density for the 3D-3V Landau damping simulation with $h_n = 2^{-7}L$, $P_c = 128$. Three dimensional representation of the standard method (left) ( $N = 2.68\text{E+8}$) and the sparse-PIC method (right) ($N = 3.4\text{E+6}$) after two oscillations of the electric field, at $t = 6$.

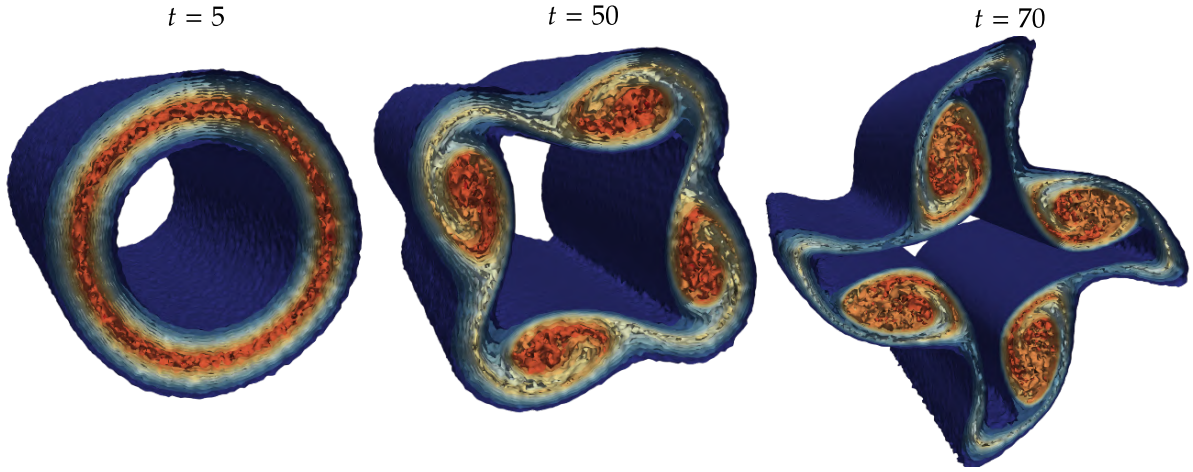


**Figure 6.31.** Representation of the electron density for the 3D-3V diocotron simulation, $\sigma = 0.03$. PIC-Std scheme, $h_n = 2^{-7}L$, $P_c = 20$, $N = 4.19\text{E+}7$.

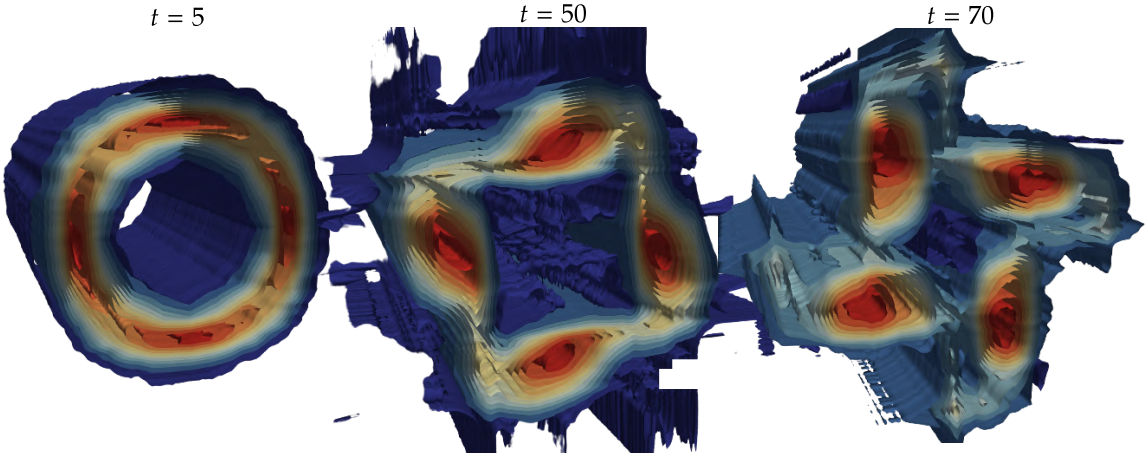


**Figure 6.32.** Representation of the electron density for the 3D-3V diocotron simulation, $\sigma = 0.03$. PIC-Sg scheme, $h_n = 2^{-7}L$, $P_c = 20$, $N = 5.4\text{E+}5$.

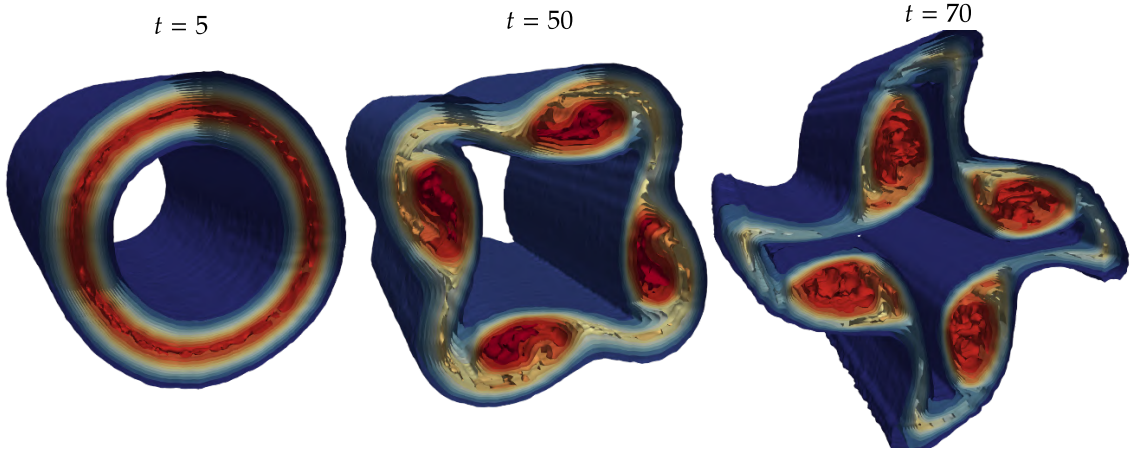


**Figure 6.33.** Representation of the electron density for the 3D-3V diocotron simulation, $\sigma = 0.03$. PIC-Sg scheme, offset combination technique, $h_n = 2^{-7}L$, $P_c = 20$, $(\tau_0, \tau_1) = (3, 6)$, $N = 1.09\text{E+}7$.

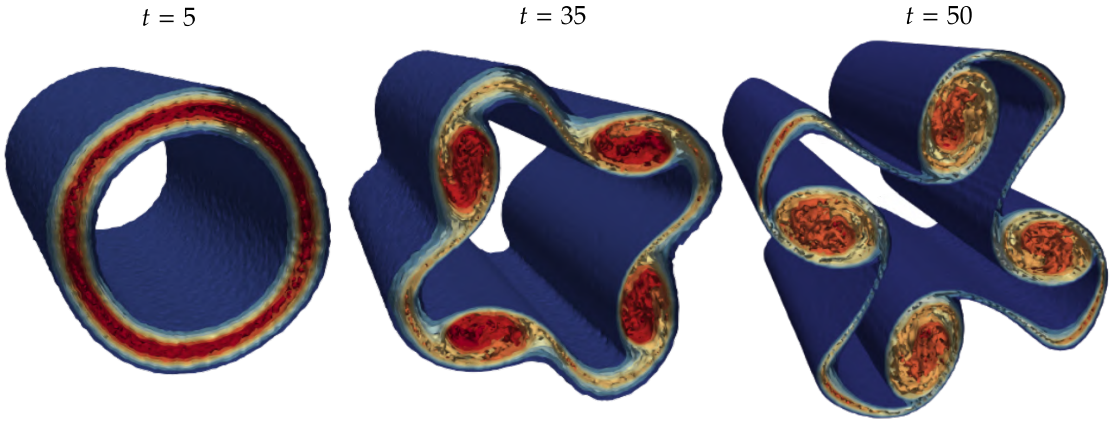


**Figure 6.34.** Representation of the electron density for the 3D-3V diocotron simulation, $\sigma = 0.02$. PIC-Std scheme, $h_n = 2^{-7}L$, $P_c = 20$, $N = 4.19\text{E+}7$.

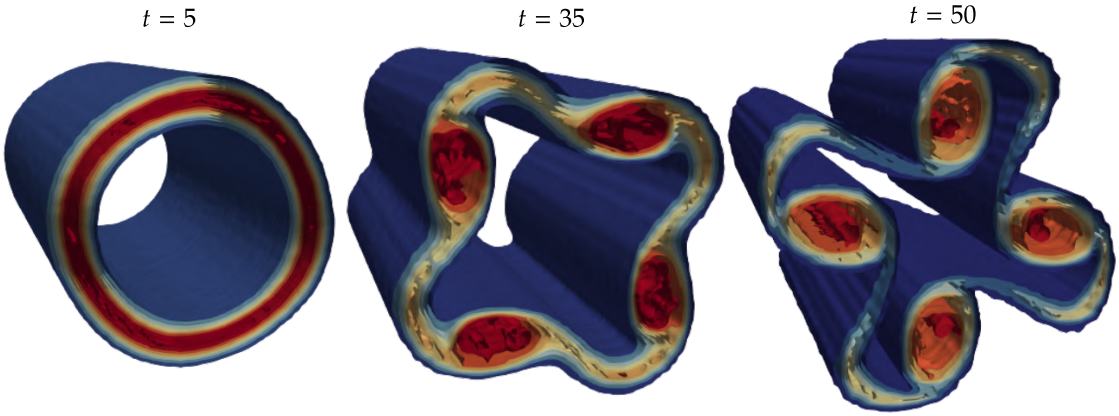


**Figure 6.35.** Representation of the electron density for the 3D-3V diocotron simulation, $\sigma = 0.02$. PIC-OHg scheme, offset combination technique, $h_n = 2^{-7}L$, $P_c = 20$, $(\tau_0, \tau_1) = (3, 6)$, $N = 1.09\text{E+}7$.

# Conclusion

In this thesis, we have presented, analyzed and proposed Particle-In-Cell numerical methods and implementations embedding sparse grid reconstructions by means of the combination technique. These methods have been numerically experienced and compared against regular PIC methods. Sparse-PIC discretizations offer a reduction of the memory requirements thanks to a better control of the statistical noise which entails a decrease of the particle number. The computation of the electric field is also less ressource consuming, owing to the reduced number of cells composing sparse-grids ($O\left(|\log h_n|^{d-1} h_n^{-1}\right)$ instead of $O\left(h_n^{-d}\right)$ for a Cartesian mesh).

The analyses conducted within the chapter 2 show that the approximation error may be decomposed into a particle error and a grid based error. Two components define the grid based error, one depending on mixed derivatives, the other depending on non-mixed derivatives of the solution, with a balance between these two contributions depending on the numerical methods. The particle error is related to the sampling of the distribution function by particles and characterizes the dispersion of the sampling. These analyses lead to two conclusions. First, the sparse-PIC schemes achieve a fair representation of the density with an improved control of the statistical noise compared to regular PIC schemes. Second, the grid based error is deteriorated, due to the dependence of the grid based error to the mixed derivatives. This component of the error is specific to sparse grid reconstructions, the mixed derivatives contributions of the grid-based error being negligible for standard PIC methods. This increase is more substantial for three dimensional computations. Similar conclusions may be drawn from the formal analysis stating the first error estimates for the electric field sparse grid approximation established within this work (see theorems 2.2.7 and 2.2.6). Furthermore, the PIC-Sg scheme is proved to be compliant with the conservation properties of standard PIC methods (total charge and momentum), except positivity. The offset technique is introduced to decrease the dominant component of the grid based error (related to the mixed derivatives) in sparse grid PIC methods. This framework permits to tune the balance between the different components of the error and improve the quality of PIC sparse grid approximations. Besides, enhancements of the PIC-Sg and PIC-Hg schemes are proposed in order to improve the efficiency of sparse grid methods regarding the computation of the electric field. The numerical experiments performed on various classical test cases, consolidate the results of the formal analyses and illustrate conclusively the gain brought by sparse grid reconstructions in particular when combined with the offset method, to mitigate the statistical noise without introducing a significant numerical diffusion.

In chapter 3, the question of the computational efficiency of sparse-PIC methods is investigated. The gains with respect to the memory requirements of such methods have been made unquestionable since their early introduction. These savings are clearly identified to be related to the reduction of the number of numerical particles permitted by the improved control on the statistical error. Nonetheless the number of component grids may be as large as tens to a hundred which calls into question the existence of any gains regarding the computational efficiency. This issue has been first investigated through the optimization on CPU, as well as shared memory architectures. Parallelization strategies tailored to Sparse-PIC methods on shared memory architectures have been introduced. The better control of the statistical error offered by Sparse-PIC reconstructions permits a substantial reduction of the number of particles for three dimensional

computations (up to 2 orders of magnitude). Conversely, the interactions of one particle (restricted to the density projection for the algorithm implementing the hierarchization), shall be computed for tens to more than one hundred component grids with a very small number of nodes. Compared to standard PIC methods these features bring significant changes in the most consuming steps of the algorithm. The Poisson problem is discretized on each of the component grids issuing linear systems of tiny sizes which dramatically reduces the memory footprint as well as the computational effort attached to the electric potential approximation. The reduced number of particles, in addition to reducing massively the memory consumption, lowers the computational cost of the particle pusher to a marginal contribution. In the end, the majority of the computations are dedicated to the projection of the particle properties onto the component grids. These operations are arithmetically intensive within sparse-PIC methods and contrary to standard PIC methods. The strategy proposed within this paper takes profit of the particularly reduced memory footprint of the arrays storing any of the component grids. These small arrays are entirely contained in the first cache level of the CPU, providing fast access to non contiguous memory addresses, thus removing the need for particle sorting routinely used in standard PIC methods. The genuine parallelism of sparse grid reconstructions permits an implementation with a scalability close to optimal (the efficiency reaches 126/128 on 128 cores) providing speed-ups of the global algorithms up to 100 on 128 cores.

In chapter 4, we proposed a GPGPU implementation of the sparse-PIC methods based on parallelization strategies specific to sparse grid reconstructions and tailored to GPU architectures. The implementation introduced in chapter 3 for shared memory architectures is extended to GPGPUs. This "CPU-inherited" implementation has proven to under exploit the potential of GPUs. This loss of efficiency is explained by the fundamental differences of architectures and memory layout of CPUs versus GPGPUs. This called for a specific GPGPU implementation. Coarse-grain and SIMT parallelisms are introduced within the GPGPU implementation thanks to particle sampling and component grid work sharing strategies. The algorithm is also designed to benefit from the large L2 cache memory of the GPU together with a strategy based on a massive number of concurrent tasks, with the aim of mitigating the negative impact of the non-coalesced memory accesses genuine to sparse-PIC methods. Sparse-PIC methods benefit from a substantial reduction of the memory footprint with respect to the standard PIC methods. Our implementation takes advantage of this feature: all the data lie on the device (a single GPU) throughout the simulation and the only data transfers between the host and the device occur at the initialization. The efficiency of the implementation has been investigated in chapter 6. The sparse-PIC GPU implementation achieves speed-ups close to 100 on a Tesla V100 for 3D-3V PIC simulations, resulting in a computational time diminution of four orders of magnitude with respect to a single core CPU standard PIC execution. As a result, 3D-3V PIC Simulations with a $512^3$ grid and 500 particles per cell have been performed on a single Tesla V100 GPU device equipped with 16GB of memory.

In chapter 5, an electrostatic ECSIM method based on a Vlasov-Ampere formulation and embedding sparse grid reconstructions has been derived. The scheme conserves exactly the total energy of the system and benefits from the mitigation of the statistical noise resulting from the sparse grid techniques. Nonetheless, a numerical instability, probably caused by the loss of positivy of the reconstrcuted field energy and deteriorating the accuracy of the method, has been observed numerically. A more thorough understanding and the proposition of possible corrections of this instability are still ongoing works.

# Perspectives

The work conducted during this thesis has not only provided some answers for the questions raised at the beginning of the thesis but also pointed out prospects that shall be investigated in

future works.

**Increasing the accuracy of sparse-PIC schemes**

As evidenced in chapter 2 and experienced numerically on the diocotron instability test case, sparse-PIC schemes may fail to reproduce fine scale structures with strong dependencies on mixed derivatives of the solutions. In addition, as a result of the reduced number of total particles, the accuracy of sparse-PIC methods can be deteriorated in configurations in which the plasma density can be very low (few particles per cell). A first alternative to these unfavorable configurations has been proposed with the offset combination technique applied to enhanced schemes (oversampled hybrid grid, enhanced sub-grid). Nonetheless, the gains provided by the sparse grid reconstruction and achieved thanks to the improvement of the statistical resolution are less consequent with these approaches than traditional sparse-PIC methods. The use of basis/shape functions of higher order (instead of the multi-linear hat functions considered in this work) shall increase the accuracy of the sparse grid reconstructions by reducing the grid-based error which may be dominant for sparse-PIC methods. In [89] a first fourth order PIC scheme in time and space has been introduced. The embedding of the fourth order basis and shape functions proposed in [89] to the sparse-PIC schemes shall be investigated in the future.

**Better understanding the gains**

The sparse grid reconstructions offer a better control of the statistical noise which entails a significant decrease of the particle number compared to standard approaches. Nonetheless, to quantify precisely the gains achieved by the sparse-PIC methods is not straightforward and seems to be problem dependent. Indeed, we have observed that the gains are very different between the Landau damping and the diocotron instability. A first achievement in this way is provided by the corollary 2.2.16, which state that if the total number of particles is chosen according to the formulae (2.75) then the particle sampling error of the reconstructed charge density is comparable for the sparse-PIC and standard PIC scheme. Since the reconstructed density is not involved into some sparse-PIC schemes (*e.g.* PIC-Sg), a similar estimation on the reconstructed electric field shall benefit the method. In addition, a more precise estimation of the number of particles required for sparse-PIC methods to achieve the same accuracy than the standard approach, depending on the configuration, shall be investigated.

**Functional and efficient code for super-computers**

The developments introduced in this manuscript concerning the parallelization of the method are relative to shared memory architectures and single GPUs. The goal was to propose an efficient implementation of sparse-PIC method on a node of a supercomputer. Therefore, the parallelization strategies introduced for shared memory architectures and GPUs in chapter 3 and 4 shall be, in a future work, embedded into a distributed memory parallelization strategy in order to create an efficient implementation for supercomputers and perform very large scale plasma simulations.

**Extension to electromagnetic regime**

To the best of our knowledge, all existing sparse-PIC methods are based on an electrostatic derivation of the Vlasov-Maxwell system. To extend the field of applications of the method, the sparse grid discretization of the Maxwell equations and the embedding of a self-consistent electromagnetic field into the sparse-PIC scheme shall be considered as a future work. Note that the extension to electromagnetic fields raises implementation issues related to the cache reuse for a three field vector ($\mathbf{J}_x$, $\mathbf{J}_y$, $\mathbf{J}_z$) compared to a scalar field ($\rho$).

### Implicit approaches

The work conducted in the chapter 5 and the numerical experiments associated have proven that semi-implicit approaches and sparse grid reconstructions can be merged together to benefit from improved stability properties and memory requirement reductions. We have experienced that the exact conservation of total energy in semi-implicit sparse-PIC method cannot provide stability without the preservation of the reconstructed field energy non-negativity. A deeper understanding of the instability related to energy-conserving semi-implicit sparse-PIC schemes shall benefit the method.

Full-implicit PIC schemes are more accurate than semi-implicit ones because no linear approximation are made, but they are more computational expansive because they require the resolution of a non linear system coupling the field and the particles. The reduction of the particle number offered by the sparse grid reconstruction could therefore bring significant gains to these approaches. The development of a full-implicit sparse-PIC scheme shall be a promising direction for future studies. Nonetheless, the instability issues related to the loss of non-negativity may also occur for the sparse full-implicit schemes.

# Bibliography


[1] 61 - on the vibrations of the electronic plasma. In D. TER HAAR, editor, Collected Papers of L.D. Landau, pages 445–460. Pergamon, 1965.

[2] Md Mohsin Ali, Peter E Strazdins, Brendan Harding, Markus Hegland, and Jay W Larson. A fault-tolerant gyrokinetic plasma application using the sparse grid combination technique. In 2015 International Conference on High Performance Computing & Simulation (HPCS), pages 499–507, Amsterdam, Netherlands, July 2015. IEEE.

[3] P.R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM Journal on Matrix Analysis and Applications, 23(1):15–41, 2001.

[4] A. Y. Aydemir. A unified Monte Carlo interpretation of particle simulations and applications to non-neutral plasmas. Physics of Plasmas, 1(4):822–831, April 1994.

[5] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. https://petsc.org/, 2022.

[6] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, Modern Software Tools in Scientific Computing, pages 163–202. Birkhäuser Press, 1997.

[7] D.C. Barnes and L. Chacón. Finite spatial-grid effects in energy-conserving particle-in-cell algorithms. Computer Physics Communications, 258:107560, January 2021.

[8] Yann Barsamian. Pic-vert: A particle-in-cell implementation for multi-core architectures. (pic-vert: Une implémentation de la méthode particulaire pour architectures multi-cœurs). 2018.

[9] Yann Barsamian, Arthur Charguéraud, Sever A. Hirstoaga, and Michel Mehrenberger. Efficient Strict-Binning Particle-in-Cell Algorithm for Multi-core SIMD Processors. In Marco Aldinucci, Luca Padovani, and Massimo Torquati, editors, Euro-Par 2018: Parallel Processing, volume 11014, pages 749–763. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[10] Yann Barsamian, Sever A. Hirstoaga, and Eric Violard. Efficient Data Structures for a Hybrid Parallel and Vectorized Particle-in-Cell Code. In 2017 IEEE International Parallel

and Distributed Processing Symposium Workshops (IPDPSW), pages 1168–1177, Lake Buena Vista, FL, May 2017. IEEE.

[11] Yann Barsamian, Sever A. Hirstoaga, and Éric Violard. Efficient data layouts for a three-dimensional electrostatic Particle-in-Cell code. Journal of Computational Science, 27:345–356, July 2018.

[12] A. Beck, J. Derouillat, M. Lobet, A. Farjallah, F. Massimo, I. Zemzemi, F. Perez, T. Vinci, and M. Grech. Adaptive SIMD optimizations in particle-in-cell codes with fine-grain particle sorting. Computer Physics Communications, 244:246–263, November 2019.

[13] Richard Bellman. Adaptive Control Processes: A Guided Tour. Princeton University Press, 1961.

[14] Janos Benk and Dirk Pflüger. Hybrid parallel solutions of the Black-Scholes PDE with the truncated combination technique. In 2012 International Conference on High Performance Computing Simulation (HPCS), pages 678–683, July 2012.

[15] Charles K Birdsall and Dieter Fuss. Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation. Journal of Computational Physics, 3(4):494–511, April 1969.

[16] Charles K. Birdsall and Neil Maron. Plasma self-heating and saturation due to numerical instabilities. Journal of Computational Physics, 36(1):1–19, June 1980.

[17] C.K. Birdsall. Interaction Between Two Electron Streams for Microwave Amplification. Department of Electrical Engineering, Stanford University, 1951.

[18] C.K. Birdsall and A.B Langdon. Plasma Physics via Computer Simulation. CRC Press, October 2018.

[19] J.U. Brackbill. On energy and momentum conservation in particle-in-cell plasma simulation. Journal of Computational Physics, 317:405–427, July 2016.

[20] J.U Brackbill and D.W Forslund. An implicit method for electromagnetic plasma simulation in two dimensions. Journal of Computational Physics, 46(2):271–308, May 1982.

[21] J.W. Brackbill and Giovanni Lapenta. A Method to Suppress the Finite-Grid Instability in Plasma Simulations. Journal of Computational Physics, 114(1):77–84, September 1994.

[22] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise Convergence Of The Combination Technique For Laplace's Equation. East-West J. Numer. Math, 2:21–45, 1994.

[23] Hans-Joachim Bungartz and Stefan Dirnstorfer. Higher Order Quadrature on Sparse Grids. In Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, Computational Science - ICCS 2004, Lecture Notes in Computer Science, pages 394–401, Berlin, Heidelberg, 2004. Springer.

[24] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. Acta Numerica, 13:147–269, May 2004.

[25] Martin Campos Pinto and Valentin Pagès. A semi-implicit electromagnetic FEM-PIC scheme with exact energy and charge conservation. Journal of Computational Physics, 453:110912, March 2022.

[26] Antoine Cerfon and Lee Ricketson. Sparse Grid Particle-in-Cell Scheme for Noise Reduction in Beam Simulations. In 13th International Computational Accelerator Physics Conference, May 2019.

[27] L. Chacón, G. Chen, and D.C. Barnes. A charge- and energy-conserving implicit, electrostatic particle-in-cell algorithm on mapped computational meshes. Journal of Computational Physics, 233:1–9, January 2013.

[28] Francis F. Chen. Introduction to Plasma Physics and Controlled Fusion. Springer US, Boston, MA, 1984.

[29] G. Chen and L. Chacón. A multi-dimensional, energy- and charge-conserving, nonlinearly implicit, electromagnetic Vlasov–Darwin particle-in-cell algorithm. Computer Physics Communications, 197:73–87, December 2015.

[30] G. Chen, L. Chacón, and D.C. Barnes. An energy- and charge-conserving, implicit, electrostatic particle-in-cell algorithm. Journal of Computational Physics, 230(18):7018–7036, August 2011.

[31] G. Chen, L. Chacón, and D.C. Barnes. An efficient mixed-precision, hybrid CPU–GPU implementation of a nonlinearly implicit one-dimensional particle-in-cell algorithm. Journal of Computational Physics, 231(16):5374–5388, June 2012.

[32] Liu Chen, A Bruce Langdon, and C.K Birdsall. Reduction of the grid effects in simulation plasmas. Journal of Computational Physics, 14(2):200–222, February 1974.

[33] Yuxi Chen and Gábor Tóth. Gauss's Law satisfying Energy-Conserving Semi-Implicit Particle-in-Cell method. Journal of Computational Physics, 386:632–652, June 2019.

[34] John Cheng. Professional Cuda C programming. Wrox programmer to programmer. John Wiley and Sons, Inc, Indianapolis, IN, 2014.

[35] J. Claustre, B. Chaudhury, G. Fubiani, M. Paulin, and J. P. Boeuf. Particle-In-Cell Monte Carlo Collision Model on GPU—Application to a Low-Temperature Magnetized Plasma. IEEE Trans. Plasma Sci., 41(2):391–399, February 2013.

[36] Bruce I Cohen, A.Bruce Langdon, and A Friedman. Implicit time integration for plasma simulation. Journal of Computational Physics, 46(1):15–38, August 1982.

[37] Shane Cook. CUDA Programming: a Developer's Guide to Parallel Computing with GPUs. Elsevier Science, Saint Louis, 2014. OCLC: 1045075687.

[38] G.-H. Cottet and P.-A. Raviart. Particle Methods for the One-Dimensional Vlasov–Poisson Equations. SIAM J. Numer. Anal., 21(1):52–76, February 1984.

[39] G. H. Cottet and P. A. Raviart. On particle-in-cell methods for the Vlasov-Poisson equations. Transport Theory and Statistical Physics, 15(1-2):1–31, February 1986.

[40] Anaïs Crestetto, Nicolas Crouseilles, and Mohammed Lemou. A particle micro-macro decomposition based numerical scheme for collisional kinetic equations in the diffusion scaling. Communications in Mathematical Sciences, 16(4):887–911, 2018.

[41] Lars K.S. Daldorff, Gábor Tóth, Tamas I. Gombosi, Giovanni Lapenta, Jorge Amaya, Stefano Markidis, and Jeremiah U. Brackbill. Two-way coupling of a global Hall magnetohydrodynamics model with a local implicit particle-in-cell model. Journal of Computational Physics, 268:236–254, July 2014.

[42] John M. Dawson. Particle simulation of plasmas. Rev. Mod. Phys., 55(2):403–447, April 1983. Publisher: American Physical Society.

[43] Viktor K. Decyk and Charles D. Norton. UCLA Parallel PIC Framework. Computer Physics Communications, 164(1-3):80–85, December 2004.

[44] Viktor K. Decyk and Tajendra V. Singh. Adaptable Particle-in-Cell algorithms for graphical processing units. Computer Physics Communications, 182(3):641–648, March 2011.

[45] Viktor K. Decyk and Tajendra V. Singh. Particle-in-Cell algorithms for emerging computer architectures. Computer Physics Communications, 185(3):708–719, March 2014.

[46] Pierre Degond and Fabrice Deluzet. Asymptotic-preserving methods and multiscale models for plasma physics. Journal of Computational Physics, 336:429–457, 2017.

[47] Pierre Degond, Fabrice Deluzet, and David Doyen. Asymptotic-preserving Particle-In-Cell methods for the Vlasov-Maxwell system near quasi-neutrality. arXiv:1509.04235 [physics], September 2015. arXiv: 1509.04235.

[48] Fabrice Deluzet, Gwenael Fubiani, Laurent Garrigues, Clément Guillet, and Jacek Narski. Sparse grid reconstructions for Particle-In-Cell methods. ESAIM: M2AN, 56(5):1809–1841, September 2022.

[49] Fabrice Deluzet, Gwenael Fubiani, Laurent Garrigues, Clément Guillet, and Jacek Narski. Efficient parallelization for 3d-3v sparse grid Particle-In-Cell: Shared memory architectures. Journal of Computational Physics, 480:112022, May 2023.

[50] Fabrice Deluzet, Gwenael Fubiani, Laurent Garrigues, Clément Guillet, and Jacek Narski. Efficient parallelization for 3d-3v sparse grid particle-in-cell: Single gpu architectures. Computer Physics Communications, page 108755, 2023.

[51] R. E. Denton and M. Kotschenreuther. {delta}f Algorithm. Technical Report DOE/ET/53088-629; IFSR-629, Texas Univ., Austin, TX (United States). Inst. for Fusion Studies, November 1993.

[52] J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, J. Dargent, C. Riconda, and M. Grech. Smilei : A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation. Computer Physics Communications, 222:351–373, January 2018.

[53] Nan Ding and Samuel Williams. An Instruction Roofline Model for GPUs. In 2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), pages 7–18, Denver, CO, USA, November 2019. IEEE.

[54] Shahab Fatemi, Andrew R. Poppe, Gregory T. Delory, and William M. Farrell. AMITIS: A 3D GPU-Based Hybrid-PIC Model for Space and Plasma Physics. J. Phys.: Conf. Ser., 837:012017, May 2017.

[55] G. Fubiani, L. Garrigues, J. P. Boeuf, and J. Qiang. Developpment of a hybrid MPI/OpenMP massivelly parallel 3D particle-in-cell model of a magnetized plasma source. In 2015 IEEE International Conference on Plasma Sciences (ICOPS), pages 1–1, May 2015. ISSN: 0730-9244.

[56] Jochen Garcke. Sparse Grids in a Nutshell. In Jochen Garcke and Michael Griebel, editors, Sparse Grids and Applications, volume 88, pages 57–80. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. Series Title: Lecture Notes in Computational Science and Engineering.

[57] L. Garrigues, B. Tezenas du Montcel, G. Fubiani, F. Bertomeu, F. Deluzet, and J. Narski. Application of sparse grid combination techniques to low temperature plasmas particle-in-cell simulations. I. Capacitively coupled radio frequency discharges. Journal of Applied Physics, 129(15):153303, April 2021. Publisher: American Institute of Physics.

[58] L. Garrigues, B. Tezenas du Montcel, G. Fubiani, and B. C. G. Reman. Application of sparse grid combination techniques to low temperature plasmas Particle-In-Cell simulations. II. Electron drift instability in a Hall thruster. Journal of Applied Physics, 129(15):153304, April 2021. Publisher: American Institute of Physics.

[59] Laurent Garrigues, Gwénaël Fubiani, and Jean-Pierre Boeuf. Negative ion extraction via particle simulation for fusion: critical assessment of recent contributions. Nuclear Fusion, 57(1):014003, January 2017. Publisher: IOP Publishing.

[60] Salimou Gassama, Éric Sonnendrücker, Kai Schneider, Marie Farge, and Margarete Domingues. Wavelet denoising for postprocessing of a 2D Particle - In - Cell code. http://dx.doi.org/10.1051/proc:2007013, 16, January 2007.

[61] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. Numerical Algorithms, 18(3):209, January 1998.

[62] Michael Griebel. Parallel Multigrid Methods on Sparse Grids. In W. Hackbusch and U. Trottenberg, editors, Multigrid Methods III, pages 211–221. Birkhäuser Basel, Basel, 1991.

[63] Michael Griebel. The combination technique for the sparse grid solution of pde's on multiprocessor machines. Parallel Process. Lett., 02(01):61–70, March 1992. Publisher: World Scientific Publishing Co.

[64] Michael Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. Computing, 61(2):151–179, June 1998.

[65] Michael Griebel, Micha Schneider, and Christoph Zenger. A combination technique for the solution of sparse grid problems. 1992.

[66] Wei Guo and Yingda Cheng. A sparse grid discontinuous Galerkin method for high-dimensional transport equations and its application to kinetic simulations, February 2016. arXiv:1602.02124 [math].

[67] Mario Heene. A massively parallel combination technique for the solution of high-dimensional PDEs. 2018. Publisher: Universität Stuttgart.

[68] Markus Hegland, Jochen Garcke, and Vivien Challis. The combination technique and some generalisations. Linear Algebra and its Applications, 420(2):249–275, 2007.

[69] Dennis W Hewett and A Bruce Langdon. Electromagnetic direct implicit plasma simulation. Journal of Computational Physics, 72(1):121–155, September 1987.

[70] Eastwood Hockney. Computer Simulation Using Particles. SIAM Rev., 25(3):425–426, July 1983. Publisher: Society for Industrial and Applied Mathematics.

[71] Roger W. Hockney and James W. Eastwood. Computer simulation using particles. A. Hilger, Bristol [England] ; Philadelphia, special student ed edition, 1988.

[72] C.-K. Huang, Y. Zeng, Y. Wang, M.D. Meyers, S. Yi, and B.J. Albright. Finite grid instability and spectral fidelity of the electrostatic Particle-In-Cell algorithm. Computer Physics Communications, 207:123–135, October 2016.

[73] Philipp Hupp. Performance of Unidirectional Hierarchization for Component Grids Virtually Maximized. Procedia Computer Science, 29:2272–2283, 2014.

[74] Philipp Hupp and Riko Jacob. A Cache-Optimal Alternative to the Unidirectional Hierarchization Algorithm. In Jochen Garcke and Dirk Pflüger, editors, Sparse Grids and Applications - Stuttgart 2014, volume 109, pages 103–132. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computational Science and Engineering.

[75] Riko Jacob. Efficient Regular Sparse Grid Hierarchization by a Dynamic Memory Layout. In Jochen Garcke and Dirk Pflüger, editors, Sparse Grids and Applications - Munich 2012, volume 97, pages 195–219. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computational Science and Engineering.

[76] Guido Juckeland and Sunita Chandrasekaran. OpenACC for programmers: concepts and strategies. 2018. OCLC: 1047846209.

[77] Zoltan Juhasz, Ján Ďurian, Aranka Derzsi, Štefan Matejčík, Zoltán Donkó, and Peter Hartmann. Efficient GPU implementation of the Particle-in-Cell/Monte-Carlo collisions method for 1D simulation of low-pressure capacitively coupled plasmas. Computer Physics Communications, 263:107913, June 2021.

[78] Xianglong Kong, Michael C. Huang, Chuang Ren, and Viktor K. Decyk. Particle-in-cell simulations with charge-conserving current deposition on graphic processing units. Journal of Computational Physics, 230(4):1676–1685, February 2011.

[79] Christoph Kowitz, Dirk Pflueger, Frank Jenko, and Markus Hegland. The combination technique for the initial value problem in linear gyrokinetics. 2012. Lecture Notes in Computational Science and Engineering.

[80] Nicholas A. Krall and Alvin W. Trivelpiece. Principles of Plasma Physics. American Journal of Physics, 41(12):1380–1381, December 1973. Publisher: American Association of Physics Teachers.

[81] L. Lancia, A. Giribono, L. Vassura, M. Chiaramello, C. Riconda, S. Weber, A. Castan, A. Chatelain, A. Frank, T. Gangolf, M. N. Quinn, J. Fuchs, and J.-R. Marquès. Signatures of the self-similar regime of strongly coupled stimulated brillouin scattering for efficient short laser pulse amplification. Phys. Rev. Lett., 116:075001, Feb 2016.

[82] A.Bruce Langdon. Effects of the spatial grid in simulation plasmas. Journal of Computational Physics, 6(2):247–267, October 1970.

[83] Giovanni Lapenta. Exactly energy conserving semi-implicit particle in cell formulation. Journal of Computational Physics, 334:349–366, April 2017.

[84] H.Ralph Lewis, A Sykes, and J.A Wesson. A comparison of some particle-in-cell plasma simulation methods. Journal of Computational Physics, 10(1):85–106, August 1972.

[85] Rodney J Mason. Implicit moment particle simulation of plasmas. Journal of Computational Physics, 41(2):233–244, June 1981.

[86] Peter Messmer. Par-t: A parallel relativistic fully 3d electromagnetic particle-in-cell code. In Tor Sørevik, Fredrik Manne, Assefaw Hadish Gebremedhin, and Randi Moe, editors, Applied Parallel Computing. New Paradigms for HPC in Industry and Academia, pages 350–355, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[87] A. Munshi, B. Gaster, T.G. Mattson, and D. Ginsburg. OpenCL Programming Guide. OpenGL. Pearson Education, 2011.

[88] Sriramkrishnan Muralikrishnan, Antoine J. Cerfon, Matthias Frey, Lee F. Ricketson, and Andreas Adelmann. Sparse grid-based adaptive noise reduction strategy for particle-in-cell schemes. Journal of Computational Physics: X, 11:100094, June 2021.

[89] A. Myers, P. Colella, and B. Van Straalen. A 4th-order particle-in-cell method with phase-space remapping for the vlasov–poisson equation. SIAM Journal on Scientific Computing, 39(3):B467–B485, 2017.

[90] Maxim Naumov, M. Arsaev, Patrice Castonguay, Jonathan M. Cohen, Julien Demouth, Joe Eaton, Simon K. Layton, N. Markovskiy, István Z. Reguly, Nikolai Sakharnykh, V. Sellappan, and Robert Strzodka. Amgx: A library for gpu accelerated algebraic multigrid and preconditioned iterative methods. SIAM J. Sci. Comput., 37, 2015.

[91] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[92] J. Petri. Non-linear evolution of the diocotron instability in a pulsar electrosphere: 2D PIC simulations. A&A, 503(1):1–12, August 2009. arXiv: 0905.1076.

[93] C. Pflaum. Convergence of the Combination Technique for Second-Order Elliptic Differential Equations. 1997.

[94] Alexander Philippov and Anatoly Spitkovsky. Ab-initio pulsar magnetosphere: three-dimensional particle-in-cell simulations of axisymmetric pulsars. ApJ, 785(2):L33, April 2014. arXiv: 1312.4970.

[95] C. Reisinger. Analysis of linear difference schemes in the sparse grid combination technique. 2007.

[96] Mark A Richards. The discrete-time fourier transform and discrete fourier transform of windowed stationary white noise. Georgia Institute of Technology, Tech. Rep, 2013.

[97] L F Ricketson and A J Cerfon. Sparse grid techniques for particle-in-cell schemes. Plasma Phys. Control. Fusion, 59(2):024002, February 2017.

[98] S. Soller. GPGPU Origins and GPU Hardware Architecture. Hochschule der Medien, 2011.

[99] Eric Sonnendrücker. Monte carlo methods with applications to plasma physics. 2014.

[100] Igor Surmin, Sergey Bastrakov, Zakhar Matveev, Evgeny Efimenko, Arkady Gonoskov, and Iosif Meyerov. Co-design of a Particle-in-Cell Plasma Simulation Code for Intel Xeon Phi: A First Look at Knights Landing. In Jesus Carretero, Javier Garcia-Blas, Victor Gergel, Vladimir Voevodin, Iosif Meyerov, Juan A. Rico-Gallego, Juan C. Díaz-Martín, Pedro Alonso, Juan Durillo, José Daniel Garcia Sánchez, Alexey L. Lastovetsky, Fabrizio

Marozzo, Qin Liu, Zakirul Alam Bhuiyan, Karl Fürlinger, Josef Weidendorfer, and José Gracia, editors, Algorithms and Architectures for Parallel Processing, volume 10049, pages 319–329. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computer Science.

[101] Junya Suzuki, Hironori Shimazu, Keiichiro Fukazawa, and Mitsue Den. Acceleration of PIC Simulation with GPU. Plasma and Fusion Research, 6:2401075–2401075, 2011.

[102] R. D. Sydora. Low-noise electromagnetic and relativistic particle-in-cell plasma simulation models. Journal of Computational and Applied Mathematics, 109(1):243–259, September 1999.

[103] William Tang, Bei Wang, Stephane Ethier, Grzegorz Kwasniewski, Torsten Hoefler, Khaled Z. Ibrahim, Kamesh Madduri, Samuel Williams, Leonid Oliker, Carlos Rosales-Fernandez, and Tim Williams. Extreme Scale Plasma Turbulence Simulations on Top Supercomputers Worldwide. In SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, pages 502–513, Salt Lake City, UT, USA, November 2016. IEEE.

[104] Paul Tranquilli, Lee Ricketson, and Luis Chacón. A deterministic verification strategy for electrostatic particle-in-cell algorithms in arbitrary spatial dimensions using the method of manufactured solutions. Journal of Computational Physics, 448:110751, January 2022.

[105] Lloyd N. Trefethen. Cubature, Approximation, and Isotropy in the Hypercube. SIAM Rev., 59(3):469–491, January 2017.

[106] D. Tskhakaya and R. Schneider. Optimization of PIC codes by improved memory management. Journal of Computational Physics, 225(1):829–839, July 2007.

[107] Gábor Tóth, Yuxi Chen, Tamas I. Gombosi, Paul Cassak, Stefano Markidis, and Ivy Bo Peng. Scaling the Ion Inertial Length and Its Implications for Modeling Reconnection in Global Simulations. JGR Space Physics, 122(10), October 2017.

[108] Gábor Tóth, Xianzhe Jia, Stefano Markidis, Ivy Bo Peng, Yuxi Chen, Lars K. S. Daldorff, Valeriy M. Tenishev, Dmitry Borovikov, John D. Haiducek, Tamas I. Gombosi, Alex Glocer, and John C. Dorelli. Extended magnetohydrodynamics with embedded particle-in-cell simulation of Ganymede's magnetosphere. J. Geophys. Res. Space Physics, 121(2):1273–1293, February 2016.

[109] J.-L. Vay, C.G.R. Geddes, E. Cormier-Michel, and D.P. Grote. Numerical methods for instability mitigation in the modeling of laser wakefield accelerators in a Lorentz-boosted frame. Journal of Computational Physics, 230(15):5908–5929, July 2011.

[110] H. Vincenti, M. Lobet, R. Lehe, R. Sasanka, and J.-L. Vay. An efficient and portable SIMD algorithm for charge/current deposition in Particle-In-Cell codes. Computer Physics Communications, 210:145–154, January 2017.

[111] J. von Neumann. First draft of a report on the edvac. IEEE Annals of the History of Computing, 15(4):27–75, 1993.

[112] Charlene Yang, Thorsten Kurth, and Samuel Williams. Hierarchical Roofline analysis for GPUs: Accelerating performance optimization for the NERSC-9 Perlmutter system. Concurrency Computat Pract Exper, 32(20), October 2020.

[113] Kane Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. IEEE Transactions on Antennas and Propagation, 14(3):302–307, 1966.

# Résumé


## Approche sur grilles parcimonieuses pour accélérer la méthode Particle-In-Cell.

Cette thèse porte sur l'étude des méthodes Particle-In-Cell (PIC) avec reconstructions sur grilles parcimonieuses. Ces méthodes permettent de réduire significativement les coûts en mémoire et les coûts de calcul des méthodes PIC. En effet, malgré leur simplicité, leur facilité de parallélisation sur plusieurs nœuds de calcul et leur robustesse, les schémas PIC présentent un inconvénient majeur, à savoir l'erreur statistique associée au bruit des particules, qui dépend du nombre moyen de particules par cellule et conduit à une complexité qui croît exponentiellement avec la dimension du problème. L'utilisation de grilles *sparse* dans les méthodes PIC, par le biais de la technique dite de combinaison où une fonction est approximée sur différentes grilles plus grossières, permet de réduire le bruit des particules, grâce aux cellules plus grandes des grilles, et donc de réduire les temps d'exécution élevés de la simulation. Dans le chapitre 2, des résultats principaux sont fournis sur la convergence de l'interpolant *sparse grid* en fonction de la discrétisation en espace et sur les propriétés de conservation de la méthode. En outre, des reconstructions *sparse grid* adaptées, dans le cadre de la technique de combinaison *offset*, sont proposées pour introduire des méthodes PIC avec une efficacité améliorée. Le chapitre 3 est consacré à l'introduction de nouvelles stratégies de parallélisation spécifiques aux architectures à mémoire partagée et adaptées aux méthodes PIC implémentant des reconstructions de grilles parcimonieuses. Ces stratégies exploitent les différents parallélismes propres aux méthodes Sparse-PIC afin d'obtenir une augmentation de vitesse supérieure à 100 sur 128 cœurs. En outre, des gains substantiels (deux ordres de grandeur) sont introduits pour les calculs séquentiels et parallèles du champ électrique grâce à la procédure de hiérarchisation. Elle consiste à décomposer l'information portée par les grilles parcimonieuses sur des fonctions de base hiérarchiques, ce qui permet de réduire considérablement le nombre d'opérations. Le chapitre 4 propose une mise en œuvre efficace des méthodes PIC sur processeurs avec carte graphique (GPU). La parallélisation, qui met en œuvre de nouvelles stratégies spécifiques aux méthodes PIC sparse et adaptées aux architectures GPU, permet d'obtenir des gains de vitesse par rapport à une exécution séquentielle sur un processeur de la même génération que le GPU. Ces gains de vitesse peuvent atteindre un facteur 100 sur un seul GPU Tesla V100, par rapport à une exécution séquentielle sur une unité de traitement informatique (CPU) ; et une réduction de quatre ordres de grandeur du temps de calcul par rapport à une simulation PIC séquentielle standard sur un CPU. Enfin le chapitre 5 concerne des travaux en cours et propose un nouveau schéma semi-implicite conservant l'énergie basé sur une discrétisation électrostatique Particle-In-Cell (PIC) du système Vlasov-Ampère (VA). La méthode est inspirée de la méthode semi-implicite de Lapenta (ECSIM). La nouveauté est la dérivation d'une approche électrostatique semi-implicite multidimensionnelle d'une formulation de Vlasov-Ampère, intégrant des techniques de reconstruction sur grilles parcimonieuses. L'objectif étant d'une part de bénéficier des avantages de la méthode de Giovanni Lapenta, *i.e* conservation exacte de l'énergie pour n'importe quel pas de temps ou discrétisation de grille ; élimination l'instabilité de grille finie; etc. D'autre part, les reconstructions *sparse grid* permettent de réduire considérablement l'empreinte mémoire en

diminuant drastiquement le nombre de particules pour une erreur statistique équivalente.