EZ: An Efficient, Charge Conserving Current Deposition Algorithm for Electromagnetic Particle-In-Cell Simulations

Steiniger, K.; Widera, R.; Bastrakov, S.; Bussmann, M.; Chandrasekaran, S.; Hernandez, B.; Hübl, A.; Juckeland, G.; Kelling, J.; Leinhauser, M.; Pausch, R.; Rogers, D.; Schramm, U.; Young, J.; Debus, A.;

EZ: An Efficient, Charge Conserving Current Deposition Algorithm for Electromagnetic Particle-In-Cell Simulations

Klaus Steiniger[a,b,1,*], Rene Widera[a,1], Sergei Bastrakov[a], Michael Bussmann[a,b], Sunita Chandrasekaran[c], Benjamin Hernandez[d], Kristina Holsapple[c], Axel Huebl[e], Guido Juckeland[a], Jeffrey Kelling[a], Matt Leinhauser[c], Richard Pausch[a], David Rogers[d], Ulrich Schramm[a,f], Jeff Young[g], Alexander Debus[a]

[a]*Helmholtz-Zentrum Dresden – Rossendorf, Bautzner Landstraße 400, 01328 Dresden, Germany*
[b]*CASUS – Center for Advanced Systems Understanding, Untermarkt 20, 02826 Görlitz, Germany*
[c]*Department of CIS, University of Delaware, Newark, DE 19716, USA*
[d]*National Center for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*
[e]*Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA*
[f]*Technische Universität Dresden, 01062 Dresden, Germany*
[g]*Georgia Tech, School of Computer Science, Atlanta, GA 30332, USA*

Abstract

We present EZ, a novel current deposition algorithm for particle-in-cell (PIC) simulations. EZ calculates the current density on the electromagnetic grid due to macro-particle motion within a time step by solving the continuity equation of electrodynamics. Being a charge conserving hybridization of **E**sirkepov's method and **Z**igZag, we refer to it as "EZ" as shorthand for "Esirkepov meets ZigZag". Simulations of a warm, relativistic plasma with PIConGPU show that EZ achieves the same level of charge conservation as the commonly used method by Esirkepov, yet reaches higher performance for macro-particle assignment-functions up to third-order. In addition to a detailed description of the functioning of EZ, reasons for the expected and observed performance increase are given, and guidelines for its implementation aiming at highest performance on

---

[*]Corresponding author
Email address: `k.steiniger@hzdr.de` (Klaus Steiniger)
[1]These authors contributed equally

GPUs are provided.

1. Introduction

Electromagnetic particle-in-cell (PIC) [1, 2] simulation is the most widespread method of modeling kinetic plasma phenomena and is a quasi-gold standard for computer experiments. PIC simulations have a wide range of applications, such as laser-plasma acceleration [3–12], laser-driven light sources [13, 14], inertial confinement fusion [15], and astrophysical plasma phenomena [16–19]. Many state-of-the-art PIC codes have memory requirements that need high performance computing resources. Notably, simulations of solid-density plasmas commonly require substantial amounts of computing and storage capacity of current top supercomputers [20, 21].

Most of the computational time of a PIC iteration is spent on the *current deposition* (CD) stage, even in highly optimized codes. Leinhauser et al. measured that close to 60 % of an iteration was spent on the CD stage for a representative plasma physics case simulated by PIConGPU [22]. The current deposition stage computes the density values on the grid from charged macro-particle motion in the PIC simulation. This density distribution can determine the new electric and magnetic field strengths in a manner that respects Gauss's law. Widely adopted charge conserving current depositions schemes were developed by Eastwood [23], Villasenor et al. [24], Esirkepov [25], and Umeda et al. [26].

The allocations of compute resources for laser-plasma simulations can reach hundreds of million core hours on top supercomputers [27]. Considering these high computational demands, an improvement in current deposition run time without loss of precision would significantly reduce resources necessary to carry out these simulations.

Here, we present a novel current deposition method offering compute savings on the scale of such high resource demands. It is a hybridization of Esirkepov's

method [25] and ZigZag [26, 28], referred to as "EZ" henceforth. EZ combines the best of both methods by being exact charge conserving and applicable to arbitrary macro-particle *assignment-functions*[2] like Esirkepov's method, and maintaining a reduced computational effort like ZigZag. EZ can be implemented in existing PIC codes and replaces Esirkepov or ZigZag, resulting in at least similar, and often better, performance.

We implemented and tested EZ to deployment in PIConGPU, a popular PIC code. Our experiments verify the important charge conservation property of EZ and demonstrate performance improvements over Esirkepov's method across a variety of compute architectures.

PIConGPU is an open source, performance-portable implementation of the fully relativistic 3D3V particle-in-cell method written in C++ [17, 29]. It is used by a steadily growing and world-spanning scientific community, including fields such as particle physics, cancer research, and materials science. Since PIConGPU builds on the performance portability library *alpaka* [30, 31], there is a single source C++ implementation of EZ in PIConGPU, which additionally takes the macro-particle assignment-function as a compile-time template argument. The template argument allows a generic implementation for all assignment-functions and alpaka enables mapping of the single implementation to all supported compute platforms. This abstraction prevents the need for redundant implementations of the method for different macro-particle assignment-functions and compute platforms. As a result, EZ remains maintainable and portable, just as PIConGPU itself.

---

[2]We follow the naming of [1], but use symbol S where they use symbol W. The assignment function is also referred to as *shape factor* by [2] and *form-factor* by [25], both using symbol S. Hierarchy of assignment functions: CIC – first-order, TSC – second-order, PQS – third order.

2. EZ Single Macro-Particle Current Density Calculation Method

2.1. Introduction to the Method

First we introduce the principle idea of the method and why a performance increase is expected in comparison to existing methods.

EZ calculates the total current density in a similar way to existing methods. At each particle-in-cell iteration, EZ obtains the total current density on the grid by adding all single macro-particle contributions on the grid.

EZ's novelty lies in the computation of each individual macro-particle contribution. In general, the single macro-particle contribution to the current density is defined by the motion of the macro-particle within an iteration step. At the beginning of current deposition we know the old and new positions of a macro-particle. Existing methods differ in the assumption they make about which path a macro-particle takes from its old to new position. Whereas Esirkepov's method assumes a straight line motion [25], ZigZag assumes a non-straight line motion leading over a *relay point* [26]. EZ employs a macro-particle path splitting approach similar to ZigZag, but calculates the current density contribution for each segment with Esirkepov's method. Considering ZigZag has not been generalized to arbitrary macro-particle assignment functions, EZ introduces a novel macro-particle path splitting approach for arbitrary macro-particle assignment functions.

With EZ, the initial and final positions of a macro-particle are used to define a relay point. This relay point is where the path of the macro-particle is split. Then, the macro-particle path is decomposed into two smaller paths: the initial position to the relay point and the relay point to the final position. EZ then uses Esirkepov's method to calculate the current density of each smaller path.

EZ's path splitting with Esirkepov's method offers improved efficiency compared to Esirkepov's original method in two ways:

1. Computational loops require fewer iterations to calculate single macro-particle current density components.
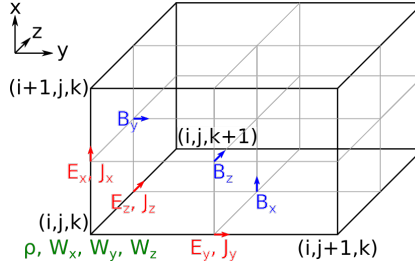
4

Figure 1: Yee cell with charge density ρ and current deposition vector $\mathbf{W}$ at a cell's origin, components of the magnetic field at the center of faces, components of both electric field and current density halfway along edges.

2. Macro-particle assignment-functions can be implemented for evaluation only on their support, sparing conditional statements required if care needed to be taken for out-of-support evaluations. Eliminating conditional statements and branching on highly parallel computer hardware avoids significant performance penalties [32].

2.2. Definition of the Relay Point and the Assignment Cell

The trajectory of a macro-particle is split in two parts only if the grid nodes to which its charge is assigned are different before and after movement. The discrete charge density $\rho|_{i,j,k}^{n}$ (at time step n) is calculated on a staggered Yee grid. The layout of the Yee cell and the location of the charge density values are visualized in fig. 1. Charge density values $\rho|_{i,j,k}^{n}$ are located at the cell origin at integer times.

In general, computing the discrete charge density associated with a macro-particle involves evaluating the macro-particle's assignment-function[3] S. The discrete charge density ρ of a single macro-particle on the grid node (i, j, k) is given by

$$\rho|_{i,j,k}^{n} = Q_p e S(x_p - i\Delta x)S(y_p - j\Delta y)S(z_p - k\Delta z)/(\Delta x \Delta y \Delta z)$$

---

[3]Ref. [1] uses symbol W.

where Q_p is the macro-particle's charge relative to e, (x_p, y_p, z_p) is the macro-particle's position at time step n, and Δx, Δy, Δz denote the grid steps.

In detail, a trajectory is split when a macro-particle leaves its assignment cell $\tilde{x}_{\text{assignment}-\text{cell}}$ (1), where $\tilde{x} = x/\Delta x$. The assignment cell encloses all possible locations of a macro-particle from which charge is assigned to the same set of grid nodes in the macro-particle's neighborhood. Accordingly, a macro-particle's assignment-cell location and boundaries depend on its position x_{old} before the movement and its assignment function S, respectively.

$$
\tilde{x}_{\text{assignment}-\text{cell}} \in
\begin{cases}
[\text{floor}(\tilde{x}_{\text{old}}), \text{floor}(\tilde{x}_{\text{old}}) + 1) \\
\qquad\qquad \text{if assignment-function is odd order,} \\
[\text{floor}(\tilde{x}_{\text{old}} + 0.5) - 0.5, \text{floor}(\tilde{x}_{\text{old}} + 0.5) + 0.5) \\
\qquad\qquad \text{if assignment-function is even order,}
\end{cases}
\tag{1}
$$

Expressions for the assignment-cell coordinates along the y and z axes are similarly obtained.

The trajectory is split at the relay point, which does not need to be on the straight line path between the macro-particle's old and new position. Several definitions for the relay point's coordinates are possible as long as the following conditions are fulfilled:

1. Along the axis where the macro-particle leaves the assignment cell, the relay point's coordinate equals the assignment cell's boundary which is traversed by the macro-particle.

2. Along the axes where the macro-particle does not leave the assignment cell, the relay point's coordinate can be anywhere within or on the boundaries of the assignment cell along this direction.

For example, the coordinate of the relay point along the x-axis can be chosen

as:

$$
\tilde{x}_r = \begin{cases} \max[\text{floor}(\tilde{x}_{\text{old}}), \text{floor}(\tilde{x}_{\text{new}})] \\ \qquad \text{if assignment-function is odd order,} \\ \max[\text{floor}(\tilde{x}_{\text{old}} + 0.5), \text{floor}(\tilde{x}_{\text{new}} + 0.5)] - 0.5 \\ \qquad \text{if assignment-function is even order,} \end{cases} \tag{2}
$$

which is valid for both the macro-particle leaving the assignment cell along this axis and not leaving it. Expressions for the relay point coordinates along the y and z axes are similarly obtained.

2.3. Current Density Calculation for a Macro-Particle Staying within its Assignment Cell

With Esirkepov's method at its core, EZ solves the electromagnetic continuity equation (3) on the staggered Yee grid to calculate the charge conserving single macro-particle current density $\mathbf{J}$ from the temporal change of the respective charge density ρ [33].

$$
\nabla \cdot \mathbf{J} + \frac{\partial \rho}{\partial t} = 0 \tag{3}
$$

Within the Yee cell, the discrete current density values $\mathbf{J}|^{n+1/2} = (J_x|_{i+1/2,j,k}^{n+1/2}, J_y|_{i,j+1/2,k}^{n+1/2}, J_z|_{i,j,k+1/2}^{n+1/2})$ are located halfway along the cell edges at times halfway between integer times, as visualized in fig. 1.

For the derivation of the current density calculation on the grid, eq. (3) is rescaled using elementary charge e, speed of light c, grid steps Δx, Δy, Δz, and the time step Δt:

- The scaled charge is defined as $\tilde{\rho} = \rho \frac{\Delta x \Delta y \Delta z}{e}$

- The current density field is defined as $\tilde{\mathbf{J}} = \mathbf{J} \frac{\Delta x \Delta y \Delta z}{ec}$

Then, the electromagnetic continuity equation in finite differences reads

$$
\frac{\tilde{\rho}|_{i,j,k}^{n+1} - \tilde{\rho}|_{i,j,k}^{n}}{c\Delta t} = -\frac{\tilde{J}_x|_{i+1/2,j,k}^{n+1/2} - \tilde{J}_x|_{i-1/2,j,k}^{n+1/2}}{\Delta x}
$$
$$
- \frac{\tilde{J}_y|_{i,j+1/2,k}^{n+1/2} - \tilde{J}_y|_{i,j-1/2,k}^{n+1/2}}{\Delta y} - \frac{\tilde{J}_z|_{i,j,k+1/2}^{n+1/2} - \tilde{J}_z|_{i,j,k-1/2}^{n+1/2}}{\Delta z} . \tag{4}
$$

7

Since the charge densities $\tilde{\rho}|_{i,j,k}^n$ and $\tilde{\rho}|_{i,j,k}^{n+1}$ are known from the macro-particle's position before and after movement, respectively, $\tilde{\mathbf{J}}|^{n+1/2}$ can be calculated from (4). Only current densities associated with the movement of a single macro-particle need to be considered, because the total current density on a grid node is obtained by summing current densities on that grid node of all macro-particles.

A charge-conserving current density of a single macro-particle is obtained from the continuity equation (4) by inserting this macro-particle's charge density before and after a movement. Let the macro-particle's position at time $t_n = n\Delta t$ before a movement be $(x_{\text{old}}, y_{\text{old}}, z_{\text{old}})$ and at time $t_{n+1} = (n+1)\Delta t$ after a movement be $(x_{\text{new}}, y_{\text{new}}, z_{\text{new}})$. Denoting the macro-particle's charge as Q, and abbreviating $S(x - i\Delta x) =: S_i(x)$ (y and z direction similarly), the scaled single macro-particle current density is related to the current deposition vector $\mathbf{W}|_{i,j,k}^{n+1/2}$:

$$
\begin{aligned}
\tilde{J}_x|_{i+1/2,j,k}^{n+1/2} - \tilde{J}_x|_{i-1/2,j,k}^{n+1/2} &= -Q\frac{\Delta x}{c\Delta t}W_x|_{i,j,k}^{n+1/2} \\
\tilde{J}_y|_{i,j+1/2,k}^{n+1/2} - \tilde{J}_y|_{i,j-1/2,k}^{n+1/2} &= -Q\frac{\Delta y}{c\Delta t}W_y|_{i,j,k}^{n+1/2} \\
\tilde{J}_z|_{i,j,k+1/2}^{n+1/2} - \tilde{J}_z|_{i,j,k-1/2}^{n+1/2} &= -Q\frac{\Delta z}{c\Delta t}W_z|_{i,j,k}^{n+1/2} ,
\end{aligned}
\tag{5}
$$

with the current deposition vector $\mathbf{W}|_{i,j,k}^{n+1/2}$ being defined from the continuity eq. (4) for the movement of a single macro-particle:

$$
S_i(x_{\text{new}})S_j(y_{\text{new}})S_k(z_{\text{new}}) - S_i(x_{\text{old}})S_j(y_{\text{old}})S_k(z_{\text{old}})
$$
$$
= W_x|_{i,j,k}^{n+1/2} + W_y|_{i,j,k}^{n+1/2} + W_z|_{i,j,k}^{n+1/2} .
$$

Assuming the macro-particle travels along a straight line by a distance not more than one cell and following the notation established by Esirkepov [25], the components of $\mathbf{W}$ are related to the assignment-function of the macro-particle

8

as follows. Defining the helper function

$$W(s_1, s_2, s_3, s_4, s_5, s_6) =$$
$$\frac{1}{3}\left(s_4 s_5 s_6 - s_1 s_5 s_6 + s_4 s_2 s_3 - s_1 s_2 s_3\right)$$
$$+ \frac{1}{6}\left(s_4 s_2 s_6 - s_1 s_2 s_6 + s_4 s_5 s_3 - s_1 s_5 s_3\right),$$

the components of $\mathbf{W}$ can be calculated by

$$
\begin{aligned}
W_x|_{i,j,k}^{n+1/2} &= \\
&W(S_i(x_{\text{old}}), S_j(y_{\text{old}}), S_k(z_{\text{old}}), S_i(x_{\text{new}}), S_j(y_{\text{new}}), S_k(z_{\text{new}})) \\
W_y|_{i,j,k}^{n+1/2} &= \\
&W(S_j(y_{\text{old}}), S_i(x_{\text{old}}), S_k(z_{\text{old}}), S_j(y_{\text{new}}), S_i(x_{\text{new}}), S_k(z_{\text{new}})) \\
W_z|_{i,j,k}^{n+1/2} &= \\
&W(S_k(z_{\text{old}}), S_j(y_{\text{old}}), S_i(x_{\text{old}}), S_k(z_{\text{new}}), S_j(y_{\text{new}}), S_i(x_{\text{new}})).
\end{aligned}
\tag{6}
$$

Current deposition by Esirkepov's method generally takes the macro-particle's location before and after movement as 'old' and 'new' coordinates. EZ makes the same choice as long as the macro-particle does not leave its assignment cell during movement. This already yields a current density for Esirkepov's method and EZ which is different from ZigZag's, as is shown in the supplemental material for macro-particle assignment-function CIC.

2.4. Current Density Calculation for a Macro-Particle Leaving its Assignment Cell

If the macro-particle leaves its assignment cell during movement, EZ makes a different choice for the 'old' and 'new' coordinates compared to Esirkepov's method. EZ takes inspiration from the ZigZag approach of splitting the trajectory in two parts. The total current density associated with the macro-particle movement is then obtained by summing the contributions of two virtual macro-particles, each virtual macro-particle traveling along its respective part of the split trajectory during the whole time step. That is, the 'old' and 'new' position of the first virtual macro-particle refer to the original macro-particle's

9

initial position and the coordinates of the relay point, while they refer to the relay point and the original macro-particle's final position for the second virtual macro-particle. Since each of the virtual macro-particle movements is restricted to the respective virtual macro-particle's assignment cell, the associated single virtual macro-particle current density can be calculated by Esirkepov's method as outlined above.

The procedure of splitting the trajectory at the relay point and adding up the current densities of two virtual macro-particles preserves the finite-difference continuity equation. Charge density value $\tilde{\rho}'|^{n+1}$ introduced at time t_{n+1} by the first virtual macro-particle located at the relay point and charge density value $\tilde{\rho}'|^n$ introduced at time t_n by the second virtual macro-particle located at the relay point are equal and together represent a static charge density distribution which does not generate a current.

Namely, the current density values produced by the virtual macro-particles separately adhere to their respective continuity equations with respective left-hand sides:

$$\frac{\tilde{\rho}'|^{n+1}_{i,j,k} - \tilde{\rho}|^n_{i,j,k}}{c\Delta t} \text{ and } \frac{\tilde{\rho}|^{n+1}_{i,j,k} - \tilde{\rho}'|^n_{i,j,k}}{c\Delta t}$$

Summing up these equations and taking into account $\tilde{\rho}'|^{n+1} = \tilde{\rho}'|^n$ yields eq. (4) for the combined current density. Thus, EZ current deposition is charge-conserving.

However, due to discretization, the implementation-specific choice of the relay point can result in different values for the components of the current deposition vector $\mathbf{W}$ and therefore different grid values of $\mathbf{J}$ for the farthest non-vanishing grid nodes. We emphasize that the finite-difference continuity eq. (4) is preserved with any of the relay point choices (satisfying the conditions). This fact has already been pointed out in the original article [25] with its uniqueness lemma being tied to the straightforward trajectory assumption.

Since numerical dispersion, as well as other particle-in-cell simulation properties, depend on the current deposition algorithm [34–36], it is possible that certain choices of the relay point's coordinates yield favorable numerical condi-

tions.

In case of 2D3V simulations, where all fields and assignment-functions depend only on x and y, the current density components J_x and J_y are calculated using the same trajectory splitting approach for movements along x, y. The corresponding components W_x and W_y of the current deposition vector are calculated from equation 6 with $S_k(z) = 1$ for all z. After simplification, it yields:

$$
\begin{aligned}
W_x|_{i,j,k}^{n+1/2} &= W(S_i(x_{\text{old}}), S_j(y_{\text{old}}), 1, S_i(x_{\text{new}}), S_j(y_{\text{new}}), 1) \\
&= \frac{1}{2}\left(S_i(x_{\text{new}})S_j(y_{\text{new}}) - S_i(x_{\text{old}})S_j(y_{\text{new}})\right. \\
&\quad \left. + S_i(x_{\text{new}})S_j(y_{\text{old}}) - S_i(x_{\text{old}})S_j(y_{\text{old}})\right) \\
W_y|_{i,j,k}^{n+1/2} &= W(S_j(y_{\text{old}}), S_i(x_{\text{old}}), 1, S_j(y_{\text{new}}), S_i(x_{\text{new}}), 1) \\
&= \frac{1}{2}\left(S_i(x_{\text{new}})S_j(y_{\text{new}}) - S_i(x_{\text{new}})S_j(y_{\text{old}})\right. \\
&\quad \left. + S_i(x_{\text{old}})S_j(y_{\text{new}}) - S_i(x_{\text{old}})S_j(y_{\text{old}})\right) .
\end{aligned}
$$

However, for movement along z there is no grid or coordinate change and therefore no trajectory splitting is possible. In this case we use the same way of calculating J_z as eq. (37) in [25], which in our notation becomes

$$
\begin{aligned}
\tilde{J}_z|_{i,j,k+1/2}^{n+1/2} = \\
Q\frac{v_z}{c}\left(\frac{1}{3}S_i(x_{\text{new}})S_j(y_{\text{new}}) + \frac{1}{6}S_i(x_{\text{old}})S_j(y_{\text{new}})\right. \\
\left. + \frac{1}{6}S_i(x_{\text{new}})S_j(y_{\text{old}}) + \frac{1}{3}S_i(x_{\text{old}})S_j(y_{\text{old}})\right) \quad (7)
\end{aligned}
$$

where v_z is velocity along z at time $t_{n+1/2}$.

3. Efficiency Increase of EZ Compared to Esirkepov's Method

EZ can significantly shorten the calculation time of the current density field compared to Esirkepov's method. This originates from a reduction of the necessary computations of the current deposition vector $\mathbf{W}$ as explained in the following.

For a macro-particle leaving its assignment cell during step from time t_n to t_{n+1}, fig. 2 visualizes the non-vanishing entries of the current deposition
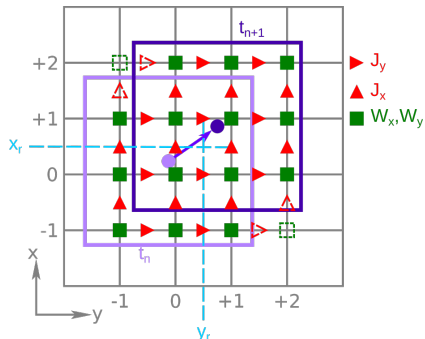
Figure 2: Visualization of discrete current deposition on two-dimensional grid for Esirkepov's method with second order assignment function TSC.

vector components W_x, W_y, as well as current density J_x, J_y on a 2D grid for Esirkepov's method. The relay point (x_r, y_r) marks the position where charge assignment extends over a different set of grid nodes when crossed by the macro-particle. The extent of the macro-particle's assignment-function at times t_n and t_{n+1} are highlighted for the second-order assignment-function TSC [1]. The number of nodes with non-vanishing values of $\mathbf{W}$ and $\mathbf{J}$ is within the bounded area. Albeit the values of the dashed entries are zero in the special case of this drawing, they are non-zero for a movement in another direction and therefore need to be calculated in a generally applicable implementation of Esirkepov's method. However, the last row (column) of W_x (W_y) at $x = +2$ ($y = +2$) does not need to be calculated, since the next row (column) of this quantity along $+x$ ($+y$) is zero and therefore the row (column) of current density J_x (J_y) at $x = +2 + 1/2$ ($y = +2 + 1/2$) vanishes according to eq. (5). For the general 2D case, there are in total $4 \times 3 = 12$ evaluations of each W_x and W_y required to calculate the respective current density component with Esirkepov's original method. See eq. (7) for the additional calculation of J_z that needs to be performed in 2D3V simulations.

In general, for an assignment-function of order l, where charge is assigned to $(l + 1)^d$ points in a d-dimensional simulation, this requires a maximum of $(l + 2)^{d-1} \times (l + 1)$ evaluations of a component of the current deposition vector
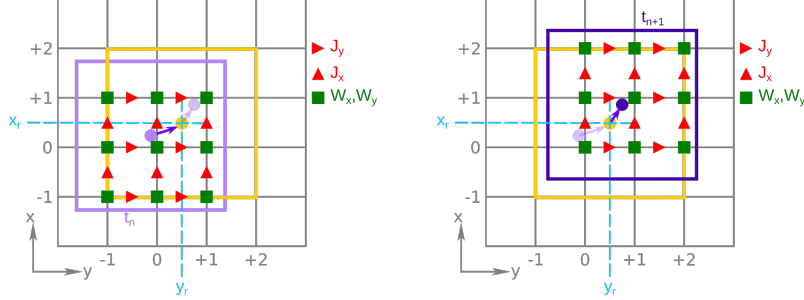
12

Figure 3: EZ-specific visualization of two-dimensional grid and second order assignment function TSC. The left hand side pictures the first part while the right hand side pictures the second part of the split trajectory.

W per macro-particle with Esirkepov's original method. This assumes the most general case where the nodes, to which a macro-particle's charge is assigned, change along every axis due to its movement.

For EZ, fig. 3 visualizes the non-vanishing entries of W_x, W_y, J_x, J_y similarly to fig. 2. Two independent but equal virtual macro-particles each travel along one part of the split trajectory. By the same argument as used for the classic method by Esirkepov, $3 \times 2 = 6$ evaluations of each W_x and W_y are required per part of the trajectory. Although this is the same number as for Esirkepov's method (in this 2D example), EZ offers improved performance compared to direct application of Esirkepov's method.

Esirkepov's traditional method must calculate the current density on all nodes to which charge of the macro-particle may be assigned before and after movement. This includes evaluations of the assignment-function out of its support making the assignment function implementation more costly compared to EZ. Due to path splitting in EZ, assignment function evaluations are always restricted to its support. While this reduces the complexity of assignment function implementations on the one hand, it also stimulates parallel calculation of the current density from many macro-particles by equalizing between all macro-particles the number of nodes at which the single contributions are calculated. That is, there is a fixed set of operations performed equally for all

13

macro-particles in the current density calculation without branching from the instruction sequence.

Moreover, in a d-dimensional simulation EZ reduces evaluations of the current deposition vector components to $2 \times (l+1)^{d-1} \times l$ points, as it spares calculations where the current density evaluates to zero. In 3D, these are fewer than the required amount of points for Esirkepov's method for assignment-functions of order $l = 3$ (PQS) and lower.

In summary, employing the trajectory splitting in principle allows EZ to surpass the performance of Esirkepov's method for two reasons.

1. EZ evaluates the current deposition vector $\mathbf{W}$ at fewer positions for lower order assignment-functions.

2. Reduced assignment-function definitions can be utilized where evaluations at positions outside the support do not need to be handled since all assignment-function evaluations are on-support with EZ.

As explained above, the latter equalizes the instruction sequence for all macro-particles which keeps thread branch divergence low, thereby promoting parallel computation via SIMT (single instruction, multiple threads) while avoiding serialization and idling threads. This is especially important when running on GPUs. Moreover, with the support size being a fixed value known at compile time, compilers may be able to optimize loops in the calculation of $\mathbf{W}$. On CPUS, these loops can be auto-vectorized promoting parallel computation via SIMD (single instruction, multiple data). Specifically for PQS, where the number of nodes with non-zero entries of the current deposition vector components is four in two directions, iterations of the $\mathbf{W}$ calculation loop and writes of the result to memory can be directly expressed with SSE4 (Streaming SIMD Extensions 4) intrinsics for single precision or AVX2 (Advanced Vector Extensions 2) intrinsics for double precision, explicitly enforcing usage of SIMD operations instead of relying on auto-vectorization.

4. EZ Implementation in PIConGPU

EZ's implementation in PIConGPU had the design goals:

1. Keep register footprint low to increase (GPU) occupancy [37].
2. Keep memory accesses, especially writes, low to avoid latency.

To address goal (1), the calculation of every component of the current deposition vector $\mathbf{W}$ is performed independently. This results in multiple calculations of the same assignment-function value instead of storing it in register memory, accomplishing the first goal.

Goal (2) provides the reason for the choice of the relay point. The initial coordinates of the relay point are the coordinates of the macro-particle's location after movement. Only if a macro-particle leaves its assignment cell along one axis, is the coordinate of the relay point on this axis changed. The relay point coordinates along the other axes remain unchanged as long as the macro-particle does not leave the assignment cell along these axes too. This allows cutting off calculations of whole components of the current density unless necessary.

For example, if the macro-particle is still within its assignment cell after particle push, the first virtual particle will move along the whole trajectory while the second virtual particle will not move at all. Therefore, the second virtual particle's current density calculation is completely spared in PIConGPU. While this possibly leads to branching of threads once per macro-particle, the performance impact is not as significant as branching during node level calculation of the single macro-particle current density.

Next, consider a macro-particle leaving its assignment cell only along one axis, say x. The relay point coordinates are at the assignment cell boundary along x and at the macro-particle's final position along y. Then the first virtual particle changes its position along x and y while it moves from the macro-particle's initial position to the relay point. The second virtual particle moving from the relay point to the macro-particle's final position then only travels parallel to the x-axis allowing to spare the computation of the y-component of

Table 1: Comparison of the number of nodes with non-zero entries of the current deposition vector for single macro-particle current deposition by Esirkepov's method and EZ for a second-order (TSC) macro-particle assignment-function.

macro-particle leaves along direction	number of nodes with non-zero entries of the current deposition vector component						total writes to current density array	
	Esirkepov			EZ				
	W_z	W_y	W_x	W_z	W_y	W_x	Esirkepov	EZ
none	$2 \times 3 \times 3$	$3 \times 2 \times 3$	$3 \times 3 \times 2$	$2 \times 3 \times 3$	$3 \times 2 \times 3$	$3 \times 3 \times 2$	54	54
x	$2 \times 3 \times 4$	$3 \times 2 \times 4$	$3 \times 3 \times 3$	$2 \times 3 \times 3$	$3 \times 2 \times 3$	$2 \times (3 \times 3 \times 2)$	75	72
$y,\ x$	$2 \times 4 \times 4$	$3 \times 3 \times 4$	$3 \times 4 \times 3$	$2 \times 3 \times 3$	$2 \times (3 \times 2 \times 3)$	$2 \times (3 \times 3 \times 2)$	104	90
$z,\ y,\ x$	$3 \times 4 \times 4$	$4 \times 3 \times 4$	$4 \times 4 \times 3$	$2 \times (2 \times 3 \times 3)$	$2 \times (3 \times 2 \times 3)$	$2 \times (3 \times 3 \times 2)$	132	108

its associated current density as it is zero. Moreover, a computation of the z component of the current density is not performed for either virtual particle.

Table 1 compares the number of nodes with non-zero entries of the components of the current deposition vector and the resulting total number of writes to the current density array between Esirkepov's method and EZ. A second-order (TSC) macro-particle assignment-function [1] is considered as an example. The number of nodes with non-zero entries, and of current density writes accordingly, depend on the number of directions along which the macro-particle leaves its assignment cell. In the given order, the cases correspond to not leaving the cell, or leaving along one, two, or three directions, respectively. The number of total writes given in the table is for a macro-particle moving along all three spatial directions, such that all current deposition vector components are non-zero. The total number of writes equals the number of nodes with non-zero current deposition vector entries summed over all components. A lower number of calculations and writes presumably results in better performance. The number of nodes with non-zero entries is given per dimension, where the order corresponds to (z, y, x).

Note, in this comparison an optimization of Esirkepov's method has been exploited as follows. If a macro-particle does not leave its assignment cell along all directions, there are nodes at which computations of the current-distribution vector can be spared, as values at these nodes are zero. Sparing these computations reduces computational time as well as memory writes. These nodes are oriented along the directions where the macro-particle stays within its assign-

ment cell. That is, if a macro-particle moves in a specific direction, but stays within its assignment cell along this direction, the number of nodes at which the current deposition vector components need to be evaluated are reduced by one along this direction. PIConGPU applies these optimizations in its implementation of Esirkepov's method, which is relevant for the performance comparisons in sec. 5.2.

As can be seen in tab. 1, EZ outperforms Esirkepov's method in all cases but the first one where they are equal. In this first case, they are equal due to the above described optimization applied to Esirkepov's method. Since optimized assignment-function implementations can be used for EZ, it can still be faster than Esirkepov's method for this case in practice. Overall, EZ shows potential for improving Esirkepov's performance.

Lower order assignment-functions, e.g. zeroth-order (NGP) or first-order (CIC), increase the difference between the Esirkepov's method and EZ with EZ requiring fewer calculations and writes.

In cases where assignment functions of fourth-order and higher are required to minimize aliasing effects, EZ's favorable properties of reduced assignment-function complexity and uniform instruction sequence can offer a benefit compared to Esirkepov's method.

5. Physics Acceptance Tests and Performance Results

5.1. Level of Charge Conservation

5.1.1. Single Macro-Particle Simulation

In order to evaluate EZ's level of charge conservation and to compare this to the optimized Esirkepov's method and ZigZag, a simple single moving macro-particle test case is set up in PIConGPU.

The test consists of one macro-particle of first-order assignment-function CIC [1] with charge $-e$ in a $24 \times 24 \times 24$ cells simulation volume. At the beginning it is located at $\mathbf{r} = (8.9\Delta x, 8.8\Delta y, 8.7\Delta z)$ and moving with constant normalized speed $v/c = \beta = 0.999$. The test comprises three scenarios, each

Table 2: Maximum absolute value λ_{SMP} of the normalized remainder $\tilde{R}_{\text{SMP}}$ of Gauss's law for simulations of a single moving macro-particle with different current deposition methods.

macro-particle	level of charge conservation λ_{SMP}		
leaves along	optimized Esirkepov's method	ZigZag	EZ
x	3.6e-08	3.6e-08	3.6e-08
$y,\ x$	5.9e-08	4.1e-08	4.1e-08
$z,\ y,\ x$	5.8e-08	1.3e-3	5.8e-08

characterized by the number of directions along which the macro-particle leaves its assignment cell. It does so either along the x axis only, within the x and y plane only, or diagonally across the volume. The respective velocity vectors are $\mathbf{v} = v \cdot (1,0,0)$, $\mathbf{v} = v \cdot (1,1,0)/\sqrt{2}$, or $\mathbf{v} = v \cdot (1,1,1)/\sqrt{3}$.

Within the PIC iteration step, the electric field $\mathbf{E}$ and charge density field ρ are calculated by PIConGPU. These are written to disk after one time step $\Delta t = 0.5\frac{\Delta x}{c}$ and then used to calculate the normalized remainder $\tilde{R}_{\text{SMP}}$ of Gauss's law. That is, the field $R|^n = \nabla \cdot \mathbf{E}|^n - \rho'|^n$ normalized to elementary charge per cell volume $\rho_{\text{SMP}} = e/(\Delta x \Delta y \Delta z)$ such that $\tilde{R}_{\text{SMP}} = R/\rho_{\text{SMP}}$. The charge density $\rho'|^n = \rho|^n - \rho|^0$ used in the calculation of $R|^n$ is a composition of the single macro-particle charge density $\rho|^n$ at time step n and the sign-inverted single macro-particle charge density at simulation start $-\rho|^0$, in order to account for the virtual mirror charge on the grid created by the finite-difference time-domain electromagnetic field solver employed by PIConGPU, as it assumes zero field at simulation startup. $\tilde{R}_{\text{SMP}}$ quantifies the level of charge conservation in the simulations. Ideally, $\tilde{R}_{\text{SMP}}$ is zero in case of perfect charge conservation and absent floating point round-off errors.

For this single macro-particle test case, the level of charge conservation λ_{SMP} is quantified by the largest absolute value of $\tilde{R}_{\text{SMP}}$ on the grid. That is, $\lambda_{\text{SMP}} = \|\tilde{R}_{\text{SMP}}\|_\infty := \max\{|\tilde{R}_{\text{SMP}}|_{ijk}|\}$. Results are displayed in Table 2. The source codes of simulations and analysis are available online [38].

Esirkepov's method and EZ reach comparable charge conservation in all three scenarios, where the absolute level of charge conservation is on the order of rounding error of floating-point arithmetic operations. ZigZag is on the same level as Esirkepov's method, as long as the macro-particle does not move diag-

onally through the volume. In the case of the macro-particle moving diagonally through the volume, ZigZag's charge conservation is orders of magnitude worse compared to Esirkepov's method and EZ. We suspect, the 3D eqs. (20) of ZigZag in [26] are actually only valid in 2D as they miss contributions to the current density J as in eqs. (35)-(37) in [24], which are proportional to the product of a macro-particle's displacements in all three dimensions. Due to this shortcoming of ZigZag, we will not take it into account in the following comparisons.

5.1.2. Warm Plasma Simulation

In order to evaluate EZ's level of charge conservation in a real world simulation and to compare it to the optimized Esirkepov's method, we set up a warm plasma simulation. The simulation consists of an initially field free space in which there are approximately 177 million electron-like macro-particles moving at a relativistic velocity. Due to macro-particle speeds close to the speed of light, we anticipate about half of the macro-particles to leave their assignment cell per step which is sufficient to compare charge conservation of EZ and Esirkepov's method.

The simulation domain extends over 192 cells in each direction and employs periodic boundary conditions on each side. There are 25 macro-particles per cell which sample a homogeneous distribution of relativistic electrons at a density of $\rho_{\mathrm{WP}} = -10^{20} \, e/\mathrm{m}^3$. Their initial momenta are normally distributed along each spatial dimension with a variance of $17.5 \, m^2 c^2$, resulting in a distribution of initial total momentum according to a Maxwell-Boltzmann distribution with a temperature of $8942 \, \mathrm{keV}$ and a most probable kinetic energy of $5 \, mc^2$. The grid resolution of $\Delta x = 57.8918 \, \mathrm{\mu m}$ is equal along all directions and chosen such that the simulation volume encompasses 5 Debye lengths along one axis at the given density and initial temperature. Time resolution is $\Delta t = 0.5 \Delta x / c$, providing variability with respect to the choice of Maxwell solver. The simulation setup and analysis code are available online [38].

Whereas a local estimate for the relative error of charge conservation at one time step can be defined by the normalized remainder $\tilde{R}_{\mathrm{WP}} = R/\rho_{\mathrm{WP}}$ of
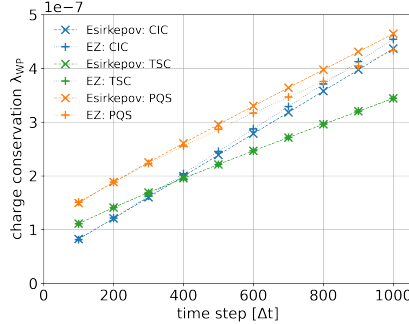
19

Figure 4: Evolution of charge conservation over time in the warm plasma simulation for EZ and Esirkepov's method and different macro-particle assignment-functions of increasing order. Charge conservation λ_{WP} is quantified by the standard deviation of the normalized Gauss' law remainder $\tilde{R}_{\mathrm{WP}}$ distribution on the grid, eq. (8).

Gauss' law at each grid node, a global estimate for the relative error of charge conservation at one time step can be obtained by analyzing the scattering of all $\tilde{R}_{\mathrm{WP}}$ values on the grid around the expectation value zero. We quantify the global charge conservation level λ_{WP} by the estimated standard deviation of the $\tilde{R}_{\mathrm{WP}}$ sample

$$\lambda_{\mathrm{WP}} = \sqrt{\frac{1}{N_{\mathrm{grid}}} \sum_{l=1}^{N_{\mathrm{grid}}} \tilde{R}_{\mathrm{WP},l}^2}\,, \tag{8}$$

where N_{grid} is the total number of $\tilde{R}_{\mathrm{WP}}$ values at one time step, i. e. $(192-1)^3$.

Evolution of global charge conservation λ_{WP} for EZ and Esirkepov's method during the course of simulation is depicted in Fig. 4 for the three macro-particle assignment-functions first-order CIC, second-order TSC, and third-order PQS [1]. Global charge conservation is on the same order of magnitude for both EZ and Esirkepov's method for all three macro-particle assignment-functions. At this order of magnitude a scattering of global charge conservation values is expected due to numerical floating point round off errors in parallel atomic summation of single macro-particle currents in the current deposition algorithm.

The uncertainty of λ_{WP} is estimated to be $\lambda_{\mathrm{WP}}/\sqrt{2n}$ [39]. For all data points in Fig. 4, this is below or equal to 1.2×10^{-10} which happens to be smaller than the markers of the data points and therefore is not visible.

20

From the results of the single macro-particle and the warm plasma simulation we conclude successful validation of EZ.

5.2. Performance Comparison

In order to examine the expected scaling of run time with macro-particle assignment-function order for the optimized Esirkepov's method and EZ, we run the above warm plasma simulation setup for several macro-particle assignment-functions and on two different GPUs, as well as on a CPU. Recall that optimizations applied to Esirkepov's method are laid out in Sec. 4.

5.2.1. Setup

Simulations ran for assignment-functions from first-order CIC to third-order PQS [1] on one NVIDIA Tesla V100 16GiB GPU at Summit [40], one AMD Instinct MI100 GPU at Spock [41], and one 64-core AMD EPYC 7662 CPU at Spock [41].

NVIDIA V100 simulations on Summit's IBM Power9 nodes were compiled with `nvcc` for architecture `sm_70` to target `NVIDIA V100` with `CUDA 11.0.3`, `spectrum-mpi 10.4.0.3-20210112`, and `boost 1.74.0` in the software environment.

AMD MI100 simulations on Spock's AMD EPYC 7662 64-core nodes were compiled with `hipcc` for architecture `gfx908` to target `AMD MI100` with `ROCm 4.5.0`, `cray-mpich 8.1.10`, and `boost 1.73.0` in the software environment.

AMD EPYC 7662 64-core simulations were compiled with `CCE Clang 12.0.3` with `-march=native` on the compute node with `cray-mpich 8.1.10`, as well as `boost 1.73.0` in the software environment and executed with 128 threads.

For each assignment-function and processing unit, ten simulations executing one hundred time steps were performed. Only the run time of the simulation steps was measured; simulation initialization and finalization are not included in the measured time.

Since current deposition is just one part of a particle-in-cell iteration step, whose fraction of time taken in the total simulation step increases with higher

Table 3: Performance metrics of EZ, as well as optimized Esirkepov's current deposition (CD) method for the warm plasma simulation with respect to macro-particle assignment-functions. The ratio of time taken by optimized Esirkepov's method over EZ defines Speedup.

Assignment	CD kernel run time [ms]			Avg. time per step [ms]		
function	Esirkepov	EZ	speedup	Esirkepov	EZ	speedup
NVIDIA V100 16GiB						
CIC	31.30	21.68	1.444	88.15 ± 0.07	80.04 ± 0.06	1.100
TSC	109.72	94.69	1.159	164.24 ± 0.06	151.65 ± 0.04	1.083
PQS	297.33	252.95	1.175	356.16 ± 0.03	319.91 ± 0.06	1.113
AMD MI100						
CIC	20.81	16.55	1.257	144.66 ± 0.52	139.43 ± 0.79	1.034
TSC	79.56	70.49	1.129	226.93 ± 0.79	217.02 ± 0.73	1.046
PQS	179.87	182.95	0.983	363.83 ± 0.80	366.70 ± 0.57	0.992
AMD EPYC 7662 64-core						
CIC	n/a			892.1 ± 0.4	864.9 ± 0.4	1.031
TSC	n/a			1904.3 ± 0.5	1674.2 ± 2.1	1.137
PQS	n/a			4658.0 ± 0.9	5037.6 ± 1.1	0.925

macro-particle assignment-function order, we additionally profiled the kernel on the different GPU architectures. Profiling allowed us to quantify the impact of EZ on the current deposition kernel alone. Profiles were taken for the first five time steps of the warm plasma simulation. After an initial warm-up phase of two steps, profiles did not change significantly in the following 3 steps. Profiling results presented in the following are taken from time step five.

All simulations for one compute architecture ran successively on the same node to avoid confounding hardware performance differences between nodes. GPU simulations were performed on a single GPU in single precision.

5.2.2. Results

Table 3 provides multiple metrics of interest:

- Run times of the current deposition kernel for GPU runs and for all macro-particle assignment-functions.

- Average time per total simulation step and their uncertainty.

- Calculated speedup with respect to the total simulation step duration when using EZ relative to Esirkepov's method. Speedup is the fraction of the average time per step required by Esirkepov's method in contrast to EZ. A value greater than 1 indicates an improved performance with EZ.

Run time of the current deposition kernel is obtained by profiling with `ncu` and `rocprof` on NVIDIA V100 and AMD MI100, respectively. Each data point of the average time per simulation step is an average of ten simulation runs executing one hundred steps. The uncertainty of the mean value is determined from estimated standard deviation of sample data multiplied with a coverage factor of two to reach a level of confidence of approximately 95 % for the time per simulation step to lie in the given interval (assuming measurement results are normally distributed).

For all macro-particle assignment-functions, EZ decreases run time of the current deposition kernel as measured on GPUs, except for the highest order assignment-function PQS on AMD MI100.

With current deposition being just one part of the total simulation step, average run times of total steps provide a better measure for overall time savings in a particle-in-cell simulation than just kernel run times. For this metric, EZ achieves a speedup on all three compute architectures.

For the first- and second-order macro-particle assignment-functions, CIC and TSC respectively, EZ consistently achieved speedup between 3.4 % and 13.7 %. For the third-order macro-particle assignment function PQS, EZ's speedup varies across compute architectures. For PQS on NVIDIA V100, 11.3 % speedup with EZ is the largest among all measured speedups on this compute architecture. However, for PQS on AMD MI100, there is almost no performance difference. Total simulation step run times differ by fewer than 1 %.

Table 4: Relative comparison of performance relevant profiling metrics between optimized Esirkepov's Method and EZ on AMD MI100. EZ is chosen as the baseline.

Assignment function	Scalar ALU Instr. SALUInsts	Shared Memory Ops. LDSInsts	Vector ALU Instr. VALUInsts
CIC	+ 89 %	+ 75 %	+ 29 %
TSC	+ 47 %	+ 21 %	- 2 %
PQS	+ 10 %	+ 0.03 %	- 8 %

Only for PQS on AMD EPYC CPU, EZ did not achieve comparable or better performance compared to Esirkepov's method.

5.2.3. Discussion

On GPUs, the observed differences between EZ and optimized Esirkepov's method in kernel run times are predominantly in line with our expectations from the scaling of compute and memory complexity, $2 \times (l + 1)^2 \times l$ for EZ and $(l + 2)^2 \times (l + 1)$ for optimized Esirkepov's method, where we expect and observe the largest difference for CIC, a smaller difference for TSC and a small or no difference for PQS. Only the much shorter run time with EZ for PQS on NVIDIA V100 is not in line with our expectation.

The profiles provide additional insight into the observed scaling of kernel run times. On AMD MI100, Esirkepov's method performs more scalar Arithmetic Logic Unit (ALU) instructions and shared memory operations than EZ for all three assignment functions as reported in Table 4.

For the slightly better performance of optimized Esirkepov's method with PQS on AMD MI100, the number of ALU calculations seems to make the critical difference due to the high arithmetic intensity for this assignment function at equal GPU occupancy of EZ and Esirkepov. High arithmetic and low memory intensity is indicated by the `MemUnitBusy` metric telling that memory operations take only about 3 % of the total GPU time. Equal GPU occupancy is indicated by equal usage of registers, as well as equal number of wavefronts of EZ and

Esirkepov. Summing vector and scalar instructions, Esirkepov's method performs about 6000 instructions (3 %) fewer than EZ, leading to fewer than 2 % kernel run time difference for PQS on AMD MI100.

On NVIDIA V100, a very similar scaling compared to AMD MI100 is observed in terms of shared memory instructions, arithmetic intensity, and occupancy. Again the total number of executed instructions is the striking difference between Esirkepov's method and EZ. Optimized Esirkepov's method performed more instructions for all three assignment functions (Table 5). The profile also shows that the larger number of instructions for Esirkepov's method originates from more `branch instructions` compared to EZ.

Table 5: Relative comparison of performance relevant profiling metrics between optimized Esirkepov's Method and EZ on NVIDIA V100. EZ is chosen as the baseline.

Assignment function	executed instructions	branch instructions
CIC	+88 %	+90 %
TSC	+96 %	+50 %
PQS	+116 %	+18 %

As the implementation of the current deposition and the macro-particle assignment-functions are independent of compute architecture in PIConGPU, we argue that the observed performance difference for PQS, for which we expect comparable calculation and memory complexity, highly depends on compiler optimizations of the current deposition kernel and possibly of the different assignment-functions. While the assignment-functions are mainly a series of `if`-conditions, with one condition fewer for EZ reflecting the on-support optimization, the current deposition consists of a nested triple `for`-loop and additional `if`-conditions providing room for a number of optimizations by the compiler such as unrolling or making use of predicates.

From the performance and profiling results of this exemplary real world simulation, we conclude that EZ provides, as expected, a substantial speedup for

assignment-functions of order two and lower independent of compute architecture. For assignment-functions of order three, EZ's performance depends on the details of the specific implementation, compute architecture and respective compiler implementations.

6. Conclusions

This work has presented a novel current decomposition method, EZ. We demonstrated its generality and utility for different macro-particle assignment functions within the performance-portable particle-in-cell code PIConGPU. We validated its charge conservation using standard deviation of the normalized Gauss' law remainder, showing that it has properties comparable to Esirkepov's method.

In addition, we verified the efficiency of EZ on several compute architectures and different accelerators by measuring performance, taking profiles, and comparing metrics to an optimized implementation of Esirkepov's method.

Performance measurements include current deposition kernel run time and average run time of a full particle-in-cell simulation step in a representative warm plasma simulation.

As expected from the scaling provided in the text, we observe a substantial speedup in these quantities with EZ on NVIDIA V100 and AMD MI100 GPUs, as well as AMD EPYC CPU for first (CIC) and second-order (TSC) macro-particle assignment-functions. For the third-order macro-particle assignment-function PQS, where we expected the performance of EZ and Esirkepov's method to be equal, we found that speedup strongly depends on the choice of compute architecture, compilers, and optimizations. For PQS, the highest order assignment function, the speedup of EZ over Esirkepov's method varied between lower (CPU), comparable (AMD MI100), and 11 % increased (NVIDIA V100) performance. Notably, we used just one implementation for each current deposition method. Due to usage of the performance portability library alpaka in PIConGPU, there is no need to implement hardware specific specializations of the

method.

Beyond the direct performance improvements demonstrated within PICon-GPU, every parallel implementation of the particle-in-cell method can benefit from EZ. It ensures a uniform instruction sequence in the computation of the current density between all macro-particles, independent of the choice of macro-particle assignment-function. This improves concurrency in the parallel computation of current density from many macro-particles. More generally, it promotes the design of novel workload distribution algorithms for cooperative execution units running in parallel, such as warps on GPUs and SIMD instructions on CPUs.

EZ allows scientists to use high-performance compute systems with comparable or higher efficiency than Esirkepov's method without loss of precision for all but one of the considered cases. This increase in efficiency enables a higher degree of sophistication in large-scale simulation campaigns modeling digital twins of lab experiments. Such developments have many applications for future work, such as predicting the femtosecond and nanometer scale dynamics of laser-driven electron or proton accelerators with increased precision by exploiting the efficiency increase to run higher resolution simulations. Likewise, the efficiency increase can be exploited to run more simulations in order to explore more of the possible parameter space of an experiment and identify interesting parameter regions to perform these experiments.

Acknowledgments

References

[1] R. W. Hockney, J. W. Eastwood, Computer Simulation Using Particles (1st ed.), CRC Press, 1988. `doi:10.1201/9780367806934`.

[2] C. K. Birdsall, A. B. Langdon, Plasma Physics via Computer Simulation (1st ed.), CRC press, 1991. `doi:10.1201/9781315275048`.

[3] R. A. Fonseca, S. F. Martins, L. O. Silva, J. W. Tonge, F. S. Tsung, W. B. Mori, One-to-one direct modeling of experiments and astrophysical scenarios: pushing the envelope on kinetic plasma simulations, Plasma Physics and Controlled Fusion 50 (12) (2008) 124034. `arXiv:0810.2460`, `doi:10.1088/0741-3335/50/12/124034`.

[4] H. Burau, R. Widera, W. Hönig, G. Juckeland, A. Debus, T. Kluge, U. Schramm, T. E. Cowan, R. Sauerbrey, M. Bussmann, PIConGPU: A fully relativistic particle-in-cell code for a GPU cluster, IEEE Transactions on Plasma Science 38 (10 PART 2) (2010) 2831–2839. `doi:10.1109/TPS.2010.2064310`.

[5] R. Lehe, M. Kirchen, I. A. Andriyash, B. B. Godfrey, J. L. Vay, A spectral, quasi-cylindrical and dispersion-free Particle-In-Cell algorithm, Computer Physics Communications 203 (2016) 66–82. `arXiv:1507.04790`, `doi:10.1016/j.cpc.2016.02.007`.

[6] J. P. Couperus, R. Pausch, A. Köhler, O. Zarini, J. M. Krämer, M. Garten, A. Huebl, R. Gebhardt, U. Helbig, S. Bock, K. Zeil, A. Debus, M. Bussmann, U. Schramm, A. Irman, Demonstration of a beam loaded nanocoulomb-class laser wakefield accelerator, Nature Communications 8 (1) (2017) 487. `doi:10.1038/s41467-017-00592-7`.
URL `http://www.nature.com/articles/s41467-017-00592-7`

[7] L. Obst-Huebl, T. Ziegler, F.-E. Brack, J. Branco, M. Bussmann, T. E. Cowan, C. B. Curry, F. Fiuza, M. Garten, M. Gauthier, S. Göde, S. H. Glenzer, A. Huebl, A. Irman, J. B. Kim, T. Kluge, S. D. Kraft, F. Kroll, J. Metzkes-Ng, R. Pausch, I. Prencipe, M. Rehwald, C. Roedel, H.-P. Schlenvoigt, U. Schramm, K. Zeil, All-optical structuring of laser-driven proton beam profiles, Nature Communications 9 (1) (2018) 5292. `doi:`

10.1038/s41467-018-07756-z.

URL http://www.nature.com/articles/s41467-018-07756-z

[8] J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, J. Dargent, C. Riconda, M. Grech, Smilei : A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation, Computer Physics Communications 222 (2018) 351–373. arXiv:1702.05128, doi:10.1016/j.cpc.2017.09.024.

URL https://linkinghub.elsevier.com/retrieve/pii/S0010465517303314

[9] A. Debus, R. Pausch, A. Huebl, K. Steiniger, R. Widera, T. E. T. Cowan, U. Schramm, M. Bussmann, Circumventing the Dephasing and Depletion Limits of Laser-Wakefield Acceleration, Physical Review X 9 (3) (2019) 031044. doi:10.1103/PhysRevX.9.031044.

URL https://link.aps.org/doi/10.1103/PhysRevX.9.031044

[10] G. Raj, O. Kononenko, M. F. Gilljohann, A. Doche, X. Davoine, C. Caizergues, Y.-Y. Chang, J. P. Couperus Cabadağ, A. Debus, H. Ding, M. Förster, J.-P. Goddet, T. Heinemann, T. Kluge, T. Kurz, R. Pausch, P. Rousseau, P. San Miguel Claveria, S. Schöbel, A. Siciak, K. Steiniger, A. Tafzi, S. Yu, B. Hidding, A. Martinez de la Ossa, A. Irman, S. Karsch, A. Döpp, U. Schramm, L. Gremillet, S. Corde, Probing ultrafast magnetic-field generation by current filamentation instability in femtosecond relativistic laser-matter interactions, Physical Review Research 2 (2) (2020) 023123. doi:10.1103/PhysRevResearch.2.023123.

URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.023123

[11] T. Kurz, T. Heinemann, M. F. Gilljohann, Y. Y. Chang, J. Couperus Cabadağ, A. Debus, O. Kononenko, R. Pausch, S. Schöbel, R. W. Assmann, M. Bussmann, H. Ding, J. Götzfried, A. Köhler, G. Raj, S. Schindler, K. Steiniger, O. Zarini, S. Corde, A. Döpp, B. Hidding, S. Karsch, U. Schramm, A. Martinez De La Ossa, A. Irman, Demonstra-

tion of a compact plasma accelerator powered by laser-accelerated electron beams, Nature Communications 12 (1). `arXiv:1909.06676, doi: 10.1038/s41467-021-23000-7`.

[12] J. P. Couperus Cabadağ, R. Pausch, S. Schöbel, M. Bussmann, Y.-Y. Chang, S. Corde, A. Debus, H. Ding, A. Döpp, F. M. Foerster, M. Gilljohann, F. Haberstroh, T. Heinemann, B. Hidding, S. Karsch, A. Koehler, O. Kononenko, A. Knetsch, T. Kurz, A. Martinez de la Ossa, A. Nutter, G. Raj, K. Steiniger, U. Schramm, P. Ufer, A. Irman, Gas-dynamic density downramp injection in a beam-driven plasma wakefield accelerator, Physical Review Research 3 (4) (2021) L042005. `doi:10.1103/PhysRevResearch.3.L042005`.
URL `www.hzdr.de/fwthttps://link.aps.org/doi/10.1103/PhysRevResearch.3.L042005`

[13] T. Haugbølle, J. T. Frederiksen, Å. Nordlund, Photon-Plasma: A modern high-order particle-in-cell code, Physics of Plasmas 20 (6) (2013) 062904. `arXiv:arXiv:1211.4575v3, doi:10.1063/1.4811384`.
URL `http://aip.scitation.org/doi/10.1063/1.4811384`

[14] R. Pausch, H. Burau, M. Bussmann, J. Couperus, T. E. Cowan, A. Debus, A. Huebl, A. Irman, A. Köhler, U. Schramm, K. Steiniger, R. Widera, Computing angularly-resolved far field emission spectra in particle-in-cell codes using GPUs, in: Proceeding of IPAC2014, Vol. MOPRI069, 2014, pp. 761–764. `doi:10.18429/JACoW-IPAC2014-MOPRI069`.
URL `http://accelconf.web.cern.ch/AccelConf/IPAC2014/papers/mopri069.pdf`

[15] A. Pukhov, Three-dimensional electromagnetic relativistic particle-in-cell code VLPL (Virtual Laser Plasma Lab), Journal of Plasma Physics 61 (3) (1999) 425–433. `doi:10.1017/S0022377899007515`.
URL `https://www.cambridge.org/core/product/identifier/S0022377899007515/type/journal_article`

[16] K.-I. Nishikawa, M. Medvedev, B. Zhang, P. Hardee, J. Niemiec, Å. Nordlund, J. Frederiksen, Y. Mizuno, H. Sol, G. J. Fishman, C. Meegan, C. Kouveliotou, N. Gehrels, Radiation from relativistic jets in turbulent magnetic fields, in: AIP Conference Proceedings, Vol. 1133, AIP, 2009, pp. 235–237. `doi:10.1063/1.3155889`.
URL `http://aip.scitation.org/doi/abs/10.1063/1.3155889`

[17] M. Bussmann, H. Burau, T. E. Cowan, A. Debus, A. Huebl, G. Juckeland, T. Kluge, W. E. Nagel, R. Pausch, F. Schmitt, U. Schramm, J. Schuchart, R. Widera, Radiative signatures of the relativistic Kelvin-Helmholtz instability, in: SC '13 Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2013, pp. 5–1 – 5–12. `doi:10.1145/2503210.2504564`.
URL `http://dl.acm.org/citation.cfm?doid=2503210.2504564`

[18] T. Grismayer, E. P. Alves, R. a. Fonseca, L. O. Silva, dc-Magnetic-Field Generation in Unmagnetized Shear Flows, Physical Review Letters 111 (1) (2013) 015005. `doi:10.1103/PhysRevLett.111.015005`.
URL `https://link.aps.org/doi/10.1103/PhysRevLett.111.015005`

[19] R. Pausch, M. Bussmann, A. Huebl, U. Schramm, K. Steiniger, R. Widera, A. Debus, Identifying the linear phase of the relativistic Kelvin-Helmholtz instability and measuring its growth rate via radiation, Physical Review E 96 (1) (2017) 013316. `doi:10.1103/PhysRevE.96.013316`.
URL `http://link.aps.org/doi/10.1103/PhysRevE.96.013316`

[20] P. Hilz, T. M. Ostermayr, A. Huebl, V. Bagnoud, B. Borm, M. Bussmann, M. Gallei, J. Gebhard, D. Haffa, J. Hartmann, T. Kluge, F. H. Lindner, P. Neumayr, C. G. Schaefer, U. Schramm, P. G. Thirolf, T. F. Rösch, F. Wagner, B. Zielbauer, J. Schreiber, Isolated proton bunch acceleration by a petawatt laser pulse, Nature Communications 9 (1) (2018) 423. `doi: 10.1038/s41467-017-02663-1`.

[21] A. Myers, A. Almgren, L. Amorim, J. Bell, L. Fedeli, L. Ge, K. Gott,

D. Grote, M. Hogan, A. Huebl, R. Jambunathan, R. Lehe, C. Ng, M. Rowan, O. Shapoval, M. Thévenet, J.-L. Vay, H. Vincenti, E. Yang, N. Zaïm, W. Zhang, Y. Zhao, E. Zoni, Porting WarpX to GPU-accelerated platforms, Parallel Computing 108 (2021) 102833. `doi:10.1016/j.parco.2021.102833`.
URL `https://www.sciencedirect.com/science/article/pii/S0167819121000818`

[22] M. Leinhauser, R. Widera, S. Bastrakov, A. Debus, M. Bussmann, S. Chandrasekaran, Metrics and design of an instruction roofline model for amd gpus, ACM Trans. Parallel Comput. 9 (1). `doi:10.1145/3505285`.
URL `https://doi.org/10.1145/3505285`

[23] J. W. Eastwood, The virtual particle electromagnetic particle-mesh method, Computer Physics Communications 64 (2) (1991) 252–266. `doi:10.1016/0010-4655(91)90036-K`.
URL `https://www.sciencedirect.com/science/article/pii/001046559190036K`

[24] J. Villasenor, O. Buneman, Rigorous charge conservation for local electromagnetic field solvers, Computer Physics Communications 69 (2) (1992) 306–316. `doi:10.1016/0010-4655(92)90169-Y`.
URL `https://www.sciencedirect.com/science/article/pii/001046559290169Y`

[25] T. Esirkepov, Exact charge conservation scheme for Particle-in-Cell simulation with an arbitrary form-factor, Computer Physics Communications 135 (2) (2001) 144–153. `doi:10.1016/S0010-4655(00)00228-9`.
URL `https://www.sciencedirect.com/science/article/pii/S0010465500002289`

[26] T. Umeda, Y. Omura, T. Tominaga, H. Matsumoto, A new charge conservation method in electromagnetic particle-in-cell simulations, Computer Physics Communications 156 (1) (2003) 73–85.

doi:10.1016/S0010-4655(03)00437-5.
URL https://www.sciencedirect.com/science/article/pii/S0010465503004375

[27] Partnership For Advanced Computing in Europe ("PRACE"), PRACE 15th Call continues to award outstanding research in HPC, https://prace-ri.eu/pr-prace-project-access-call15/, [Online; accessed 30-March-2022] (2017).

[28] J. Yu, X. Jin, W. Zhou, B. Li, Y. Gu, High-Order Interpolation Algorithms for Charge Conservation in Particle-in-Cell Simulations, Communications in Computational Physics 13 (4) (2013) 1134–1150. doi:10.4208/cicp.290811.050412a.

[29] PIConGPU: Particle-in-Cell Simulations for the Exascale Era, https://github.com/ComputationalRadiationPhysics/picongpu, [Online; latest release picongpu-0.6.0 at 21-December-2021] (2021).

[30] E. Zenker, B. Worpitz, R. Widera, A. Huebl, G. Juckeland, A. Knüpfer, W. E. Nagel, M. Bussmann, Alpaka - an abstraction library for parallel kernel acceleration, IEEE Computer Society, 2016. arXiv:1602.08477.
URL http://arxiv.org/abs/1602.08477

[31] A. Matthes, R. Widera, E. Zenker, B. Worpitz, A. Huebl, M. Bussmann, Tuning and optimization for a variety of many-core architectures without changing a single line of implementation code using the alpaka library, 2017. arXiv:1706.10086.
URL https://arxiv.org/abs/1706.10086

[32] NVIDIA Corporation & affiliates, CUDA C++ Best Practices Guide v12.1, https://docs.nvidia.com/cuda/archive/12.0.1/cuda-c-best-practices-guide/index.html#branching-and-divergence, [Online; accessed 24-March-2023] (2022).

[33] K. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, IEEE Transactions on Antennas and Propagation 14 (3) (1966) 302–307. `doi:10.1109/TAP.1966.1138693`.

[34] B. B. Godfrey, J.-L. Vay, Numerical stability of relativistic beam multidimensional PIC simulations employing the Esirkepov algorithm, Journal of Computational Physics 248 (2013) 33–46. `doi:https://doi.org/10.1016/j.jcp.2013.04.006`.
URL `https://www.sciencedirect.com/science/article/pii/S0021999113002556`

[35] M. Meyers, C.-K. Huang, Y. Zeng, S. Yi, B. Albright, On the numerical dispersion of electromagnetic particle-in-cell code: Finite grid instability, Journal of Computational Physics 297 (2015) 565–583. `doi:https://doi.org/10.1016/j.jcp.2015.05.037`.
URL `https://www.sciencedirect.com/science/article/pii/S002199911500371X`

[36] J. Xiao, H. Qin, Y. Shi, J. Liu, R. Zhang, A lattice Maxwell system with discrete space–time symmetry and local energy–momentum conservation, Physics Letters A 383 (9) (2019) 808–812. `doi:https://doi.org/10.1016/j.physleta.2018.12.010`.
URL `https://www.sciencedirect.com/science/article/pii/S0375960118312106`

[37] NVIDIA Corporation & affiliates, CUDA C++ Best Practices Guide v12.1, `https://docs.nvidia.com/cuda/archive/12.0.1/cuda-c-best-practices-guide/index.html#execution-configuration-optimizations`, [Online; accessed 24-March-2023] (2023).

[38] K. Steiniger, R. Widera, J. Young, EZ publication: source code, profiling, analysis and simulation data (2022). `doi:10.14278/rodare.1511`.

[39] P. Zyla, et al., Review of Particle Physics, PTEP 2020 (8) (2020) 083C01, Statistics Review; sec. 40.2. `doi:10.1093/ptep/ptaa104`.

[40] Oakridge Leadership Computing Facility, Summit User Guide, `https://docs.olcf.ornl.gov/systems/summit_user_guide.html#summit-nodes`, [Online; accessed 17-December-2021] (2021).

[41] Oakridge Leadership Computing Facility, Spock Quick-Start Guide, `https://docs.olcf.ornl.gov/systems/spock_quick_start_guide.html#spock-compute-nodes`, [Online; accessed 17-December-2021] (2021).