Submitted to the special edition of Physical Review Special Topic Accelerator and Beam

Efficient Numerical and Parallel Methods for Beam Dynamics Simulations

Jin Xu[1,2], Brahim Mustapha[1], and Misun Min[2]

[1] *Physics Division*
[2] *Mathematics and Computer Science Division*
Argonne National Laboratory
9700 S. Cass Ave., Argonne, IL 60439, USA

Abstract

Efficient numerical and parallel methods have been used at Argonne National Laboratory (ANL) to develop petascalable software packages for beam dynamics simulations. This paper is an summary of the numerical methods and parallel algorithms used and developed in the past five years at ANL. It is an extension to the paper "Developing petascalable algorithms for beam dynamics simulations," on proceeding of the first International Particle Accelerator Conference IPAC-2010 [33]. In this paper, numerical methods and parallel algorithms for our beam dynamics simulations are summarized. These include the standard particle-in-cell (PIC) method, direct Vlasov solvers and scalable Poisson solvers. Efficient parallel algorithms for these methods on supercomputers have been implemented and have successfully used tens of thousands processors on the IBM Blue Gene/P system at the Argonne Leadership Computing Facility (ALCF). Among them, scalable Poisson solvers in three dimensions, to account for space charge effects, are the most challenging. Several methods have been used to solve Poisson's equation efficiently in different situations. The ALCF provides a suitable environment to perform large scale beam dynamics simulations. High-order numerical methods have been adopted to increase the accuracy. Domain decomposition methods have been used for parallelizing the solvers, and good scaling has been achieved. Preliminary results for the direct Vlasov solvers have been obtained in up to four dimensions. The parallel beam dynamics code PTRACK, which uses the PIC method to solve Poisson's equation, has been used as a workhorse for end-to-end simulations and large-scale design optimizations for linear accelerators.

Key words: Poisson's Equation, Vlasov Equation, *hp*-FEM, Semi-Lagrangian Method, Discontinuous Galerkin Method, High Dimension, Multigrid Technique

PACS subject classifications: 02.60.-x; 02.70.-c; 07.05.Tp; 52.65.-y;

1 Introduction

Beam dynamics simulations (BDS) play a significant role in accelerator modeling, they are important for both the design and operation phases of an accelerator. Computational beam dynamics have become more powerful with the development of supercomputers capable of simulating more complex effects and phenomena in beam physics. As the computing era enters petascale, more accurate accelerator simulations can be conducted. In order to efficiently use this computing power, we need to develop more efficient numerical and parallel methods. This paper introduces our efforts in this direction that is to apply efficient numerical methods and develop scalable methods necessary for large scale BDS. We explain various numerical and parallel methods in a way that is easy to understand, and we direct the interested reader to related journals for more detailed information [28,30–32]. These efforts are limited and still on going, and there are many other efficient numerical methods not been included. We will try and report them in the future studies.

Charged particle beams are at the core of accelerator technology. Since charged beam is a kind of plasma, most of the methods used for plasma simulations could be adopted and used for charged beam simulations. The methods for simulating plasma can be divided into three categories: microscopic models, kinetic models, and fluid models. In the *microscopic* model, each charged particle is described by six variables (x, y, z, v_x, v_y, v_z). Therefore, for N particles, there are 6N variables in total. Since each two particles have mutual force interaction, this leads to a group of 6N equations which has 6N variables in each of them. Solving the beam dynamics equation in 6N dimensions exceeds the capability of current supercomputers for large N. The *fluid* model is the simplest, because it treats the plasma as a conducting fluid with electromagnetic forces exerted on it. This leads to solving the magnetohydrodynamics (MHD) equations in 3D (x, y, z); MHD techniques solve for average quantities, such as density and charge. Since charged beams are often accelerated and focused in bunches makes the fluid model, which is successful to describe a continuous flow of fluid, not suitable for charged beam simulations. Between these two models is the *kinetic* model, which is used by most current beam dynamics simulations. This model obtains the charge density function by solving the Boltzmann or Vlasov equations in 6 dimensions (x, y, z, v_x, v_y, v_z). The Vlasov equation describes the evolution of a system of particles under the effects of self-consistent electromagnetic fields. The Vlasov equation can be solved in two ways. The dominant way is the so-called particle-in-cell (PIC) method, which uses the motion of the particles along the characteristics of the Vlasov equation using an Euler-Lagrange approach. The PIC method is fast and easy to implement; and, with the arrival of petascale computing, one-to-one simulation can be realized for low-density beams. But for more intensive beams, the PIC method still uses macro particles, making it difficult to capture detailed beam structure. Furthermore, noise is associated with the finite number of particles in the simulations. Another way to solve the kinetic model is to solve the Vlasov equation directly, which requires solving in 2D dimensions (D is the dimension of the physical space). For example, in order to simulate beams in three dimensions, the Vlasov equation must be solved in 6 dimensions. Clearly, petascale computing is required, and petascale algorithms are critical to a successful solution. During the past 5 years, we have applied efficient numerical methods and developed several software packages to meet these demands. This paper presents our efforts in applying these efficient numerical methods to develop petascale software packages for BDSs for linear accelerators. The paper is organized as follows. The numerical methods are explained in Section 2 and the parallel methods in Section 3. Further comparisons and discussions of these numerical methods are presented in Section 4. We draw our conclusions in Section 5.

2 Numerical Methods

In this section, we present numerical methods for BDS. These methods include PIC and direct Vlasov methods. Two methods for time integration of the Vlasov equation are also presented. Since space charge effects are included in both the PIC and the Vlasov solutions, the numerical methods for solving the Poisson's equation is discussed separately at the end of the section.

2.1 Particle-in-Cell Method

The particle-in-cell method is the most widely used approach in the kinetic model for BDS. It uses macro particles to represent the real charged particles in the beam, integrating the beam dynamics equations under external and internal forces. The external force usually comes from electromagnetic fields of accelerator components and the internal force comes from the space charge effect, which is accounted for by solving the Poisson's equation on a fixed grid. Considering the relativistic effect, the beam dynamics equation should be solved in the appropriate coordinate system. Many ray-tracing codes are available, such as RAYTRACE, PARMILA and IMPACT. Another approach is to use the matrix formalism for the design and study of beam-optics systems, for example, TRANSPORT, TRACE3D, or GIOS. We will focus here on the ray-tracing method. A ray-tracing or particle-tracking code has many advantages over a matrix-based code, we mention for example: (i) the external field can be represented more accurately within the physical aperture of the device including fringe fields and field superpositions; (ii) the 6D particle coordinates are known at any point along the accelerator; (iii) non-electromagnetic elements, such as beam degraders or strippers, can be accurately simulated; (iv) beam space charge fields, especially for multi-components ion beams, can be calculated with high accuracy; and (v) beam losses can be calculated in the presence of complex sets of field errors and device misalignments. Our BDS research is based on the beam dynamics code TRACK, developed in the Physics Division at Argonne National Laboratory over the last 10 years. The basic method is presented below.

The transport of a charged particle is described by the equation of motion:

$$\frac{d\vec{p}}{dt} = Q(\vec{E} + \vec{v} \times \vec{B}), \tag{1}$$

,where $\vec{p}$ is the particle momentum and Q is its charge; $\vec{E} = \vec{E}_{ext} + \vec{E}_{int}$ and $\vec{B} = \vec{B}_{ext} + \vec{B}_{int}$ are the sums of external and internal electric and magnetic fields, respectively; and $\vec{v}$ is the particle velocity. TRACK uses 6 independent variables for tracking the phase-space coordinates of the particles, $(x, x' = dx/dz, y, y' = dy/dz, \beta = v/c, \phi)$, where $v = \mid \vec{v} \mid$ and ϕ is the particle phase shift with respect to a reference particle. Since the relativistic effect is being considered, the set of equations used for the step-by-step integration routine is

$$\frac{dx}{dz} = x', \frac{dy}{dz} = y', \frac{d\phi}{dz} = \frac{2\pi f_0 h}{\beta c} \tag{2}$$

$$\frac{dx'}{dz} = \chi \frac{Q}{A} \frac{h}{\beta\gamma} [\frac{h}{\beta c}(E_x - x'E_z) + x'y'B_x - (1 + x'^2)B_y + y'B_z] \tag{3}$$

$$\frac{dy'}{dz} = \chi \frac{Q}{A} \frac{h}{\beta\gamma} [\frac{h}{\beta c}(E_y - y'E_z) - x'y'B_x + (1 + y'^2)B_x - x'B_z] \tag{4}$$

$$\frac{d\beta}{dz} = \chi \frac{Q}{A} \frac{h}{\beta\gamma^3 c}(x'E_x + y'E_y + E_z), \tag{5}$$

where $\gamma = 1/\sqrt{1-\beta^2}; h = 1/\sqrt{1 + x'^2 + y'^2}; \chi = 1/m_a c^2; A$ is the mass number; m_a is the atomic mass unit; and $B_x, B_y, B_z, E_x, E_y, E_z$ are the components of magnetic and electric fields. TRACK uses the 4^{th} order Runge-Kutta method to integrate the equation (2), every time step is subdivided into 4 substeps. Currently, the external electromagnetic fields are computed from commercial software packages, such as CST MWS, EMS, ANSYS, etc. Several Poisson solvers have been developed and will be introduced in the following sections. The code TRACK supports the following electromagnetic elements for acceleration, transport and focusing of multi-component ion beams:

- Any type of RF accelerating resonator with realistic 3D fields.
- Radio Frequency Quadrupoles (RFQ).
- Solenoids with fringing fields.
- Bending magnets with fringing fields.
- Electrostatic and magnetic multipoles (quadrupoles,sextupoles, ...) with fringing fields.
- Multi-harmonic bunchers.
- Axial-symmetric electrostatic lenses.
- Entrance and exit of a high voltage (HV) deck.
- Accelerating tubes with DC distributed voltage.
- Transverse beam steering elements.

- Stripping foils or films (for FRIB, not general yet).
- Horizontal and vertical jaw slits for beam collimation.
- Static ion-optics devices with both electric and magnetic
- realistic three-dimensional fields.

More details can be found in [3,28].

2.2 Direct Vlasov Method

PIC solvers have their shortcomings, such as the noise associated with a finite number of macro particles and the difficulty in describing the beam tail and eventual beam halo formation. An alternative approach, which has the potential to compensate for the limitations of the PIC method, is to solve the Vlasov equation directly. The beam is described as a distribution function in the six-dimensional phase space (x, y, z, v_x, v_y, v_z). The evolution of the distribution function of particles $f(\vec{x}, \vec{v}, t)$ in the phase space $(\vec{x}, \vec{v}) \in \mathbf{R}^d \times \mathbf{R}^d$, with d =1,2,3 and time t, is given by the dimensionless Vlasov equation,

$$\frac{\partial f(\vec{x}, \vec{v}, t)}{\partial t} + \vec{v}(\vec{x}, t) \cdot \nabla_x f(\vec{x}, \vec{v}, t) + \vec{F}(\vec{x}, \vec{v}, t) \cdot \nabla_v f(\vec{x}, \vec{v}, t) = 0, \qquad (6)$$

where the force field $\vec{F}(\vec{x}, \vec{v}, t)$ can be coupled to the distribution function f. For the Vlasov-Poisson system, this coupling is done through the macroscopic beam density ρ.

$$\rho(\vec{x}, \vec{v}, t) = \int_{\mathbf{R}^d} f(\vec{x}, \vec{v}, t) dv \qquad (7)$$

The force field, which depends only on t and $\vec{x}$, makes the system nonlinear. It is determined by solving Poisson's equation,

$$\vec{F}(\vec{x}, \vec{v}, t) = \vec{E}(\vec{x}, t), \qquad \vec{E}(\vec{x}, t) = -\nabla_{\vec{x}} \phi(\vec{x}, t), \qquad -\Delta_{\vec{x}} \phi(\vec{x}, t) = \rho(\vec{x}, t) - 1, \quad (8)$$

where $\vec{E}$ is the electric field and ϕ the electric potential.

Since the Vlasov equation could involve higher dimensions than the PIC method, a time-splitting scheme has been used for time integration, as proposed by Cheng and Knorr [5]. This transforms the Vlasov equation from high dimension to two lower dimension equations. Details are given in the following subsections.

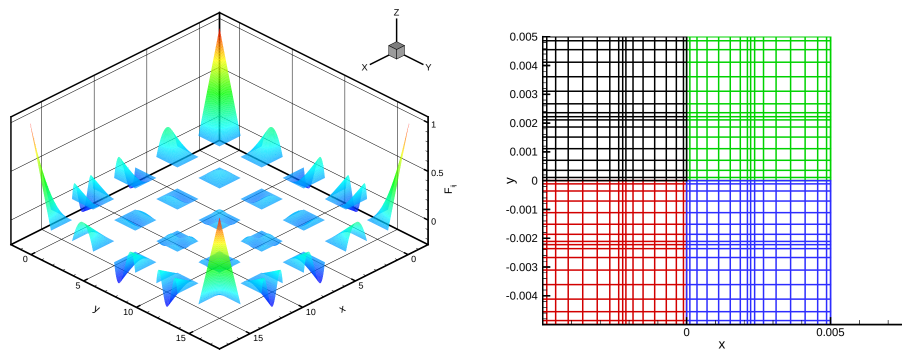

Fig. 1. Modal bases (left) and mesh (right) on quadrilateral.

2.2.1 Vlasov Equation in 1P1V Phase-Space

In 1P1V phase space, the normalized Vlasov equation can be written as follows [1].

$$\frac{\partial f(x,v,t)}{\partial t} + v(x,t)\frac{\partial f(x,v,t)}{\partial x} + E(x,t)\frac{\partial f(x,v,t)}{\partial v} = 0 \tag{9}$$

$$E(x,t) = -\frac{\partial \phi(x,t)}{\partial x}, -\Delta\phi(x,t) = \frac{\partial E(x,t)}{\partial x} = \rho(x,t) - 1 \tag{10}$$

$$\rho(x,t) = \int_{-\infty}^{\infty} f(x,v,t)dv \tag{11}$$

The distribution function $f(x,v,t)$ is expanded on a structured quadrilateral grid, as shown on the right of Fig. 1. Modal bases, shown on the left of Fig. 1, have been used. A semi-Lagrangian method, explained in Section 2.3, was also used. To increase the accuracy, we have adopted the algorithm proposed in [25]. Each time step has been subdivided into three substeps: the first and the third substeps are in velocity space, and the second substep is in physical space. The detailed procedure follows.

```
......
 Do istep=1,nstep:
 - Compute $j^n = q\int f^n(x^n,v^n)v^n dv$;
 - Compute $E^{pred} = E^n - j^n\Delta t$ from Ampere's law;
 - Do until $| E^{n+1} - E^{pred} | < \varepsilon$
       · Substep1: $v^{n+1/2} = v^{n+1} - E^{pred}(x^{n+1})\Delta t/2$
       · Substep2: $x^n = x^{n+1} - v^{n+1/2}\Delta t$;
       · Substep3: $v^n = v^{n+1/2} - E^n(x^n)\Delta t/2$;
       · Interpolate to compute charge density;
       · Solve Poisson's equation for $E^{n+1}$;
       · Update new $E^{pred} = E^{n+1}$.
 - Enddo
```

Enddo
......

Other methods can be developed to increase the order of accuracy for time integration. The interpolation is performed on each quadrilateral, and its accuracy depends on the polynomial order used. During each time step iteration, Poisson's equation must be solved on a structured grid. The method is explained in Subsection 2.5.3. Detailed information can be found in [31].

2.2.2 Vlasov Equation in 2P2V Phase-Space

Next, we study the Vlasov equation in higher dimensions, 2P2V. In beam dynamics, a simplified model has been developed in 2P2V [12] as a paraxial model based on the following assumptions:

- The beam is in a steady state: all particle coordinates derivatives with respect to time vanish.
- The beam is sufficiently long so that longitudinal self-consistent forces can be neglected.
- The beam is propagating at constant velocity v_b along the propagation axis z.
- Electromagnetic self-forces are included.
- $\vec{p} = (p_x, p_y, p_z), p_z \sim p_b$ and $p_x, p_y \ll p_b$, where $p_b = \gamma m v_b$ is the beam momentum. It follows in particular that

$$\beta \approx \beta_b = v_b/c, \gamma \approx \gamma_b = (1 - \beta_b^2)^{-1/2}. \tag{12}$$

- The beam is thin: the transverse dimensions of the beam are small compared to the characteristic longitudinal dimension.

The paraxial model can be written as

$$\frac{\partial f}{\partial z} + \frac{\vec{v}}{v_b} \cdot \nabla_{\vec{x}} f + \frac{q}{\gamma_b m v_b}\left(-\frac{1}{\gamma_b^2}\nabla\Phi^s + \vec{E}^e + (\vec{v}, v_b)^T \times \vec{B}^e\right) \cdot \nabla_{\vec{v}} f = 0 \tag{13}$$

coupled with the Poisson's equation

$$-\Delta_{\vec{x}}\Phi^s = \frac{q}{\epsilon_0}\int_{R^2} f(z, \vec{x}, \vec{v}) d\vec{v}, \tag{14}$$

where Φ^s is the self-consistent electric potential due to space charge; $\vec{E}^e$ and $\vec{B}^e$ are the external electric and magnetic fields, respectively; and v_b is the reference beam velocity.

In 2P2V simulations, since it is expensive to interpolate in four dimensions, the distribution function is updated after each substep. The distribution function

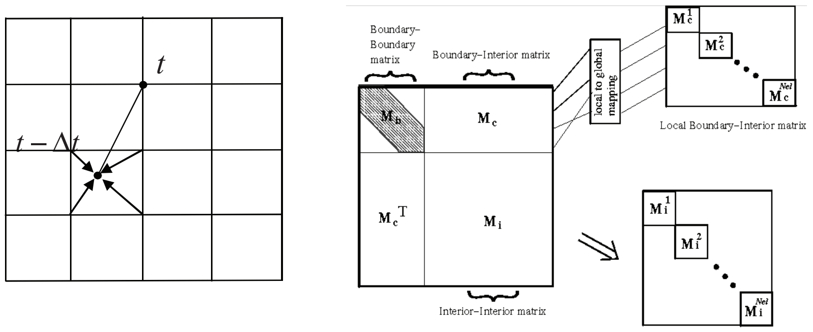

Fig. 2. Semi-Lagrangian method (left) and Shur complement method (right)

$f(x, y, v_x, v_y, t)$ is expanded on an unstructured triangular grid, as shown in the right side of Fig. 4. Nodal bases shown on the left of Fig. 4 have been used. A Discontinuous Galerkin (DG) method, explained in Section 2.4, has been used. Time splitting is the same as that proposed by Cheng and Knorr [5]. Each time step has been divided into three substeps: the first and third substeps are in physical space, and the second substep is in velocity space. The detailed procedure follows.

......

```
Do istep=1,nstep:
  - Substep 1: Perform a half time step shift in the (x,y) plane: $f^*(\vec{x},\vec{v}) = f(t^n, \vec{x} - \vec{v}\Delta t/2, \vec{v})$
  - Compute the electric field at time $t^{n+1/2}$ by substituting $f^*$ in the Poisson's equation;
  - Substep 2: Perform a full time step shift in the $(v_x, v_y)$ plane: $f^{**}(\vec{x},\vec{v}) = f^*(\vec{x}, \vec{v} - \vec{E}(t^{n+1/2}, \vec{x})\Delta t)$;
  - Substep 3: Perform a second half time step shift in the (x,y) plane: $f(t^{n+1}, \vec{x}, \vec{v}) = f^{**}(\vec{x} - \vec{v}\Delta t/2, \vec{v})$;
Enddo
```

......

On each two-dimensional space, the Vlasov equation transforms into a linear transport equation in 2D. The Semi-Lagrangian method has been used in 2P2V as in 1P1V. Instead of solving Poisson's equation and interpolating in 1D, they have been solved in 2D on structured quadrilaterals. Details can be found in [31]. As for the transport equation, researchers have successfully applied the discontinuous Galerkin method to transient Maxwell and Euler equations in 2D and 3D. Therefore, instead of using the semi-Lagrangian method, DG has been adopted on unstructured grid. Similarly, during the time step iteration, Poisson's equation must be solved on an unstructured grid. The method is explained in Subsection 2.5.3. Detailed information can be found in [32].

2.3 Semi-Lagrangian Method (SLM)

As shown on the left of Fig. 2, the semi-Lagrangian method consists of computing the distribution function at each grid point by following the characteristic curves backward and interpolating the distribution function at the previous time step. According to Liouville's theorem, the phase-space distribution function is constant along the trajectories of the system. Therefore, the interpolation at the previous time step equals the function value at the present time step. In contrast to the Eulerian framework, the semi-Lagrangian scheme allows the use of large time-steps without losing stability. The limitations for stability are that trajectories should not cross and that particles should not "overtake" one another. Therefore, the choice of time step size in the Semi-Lagrangian scheme is limited only by numerical accuracy.

2.4 Discontinuous Galerkin Method (DGM)

In 1973, Reed and Hill [21] introduced the first discontinuous Galerkin method for hyperbolic equations. Since that time there has been active development of DG methods for hyperbolic and nearly hyperbolic problems, resulting in a variety of methods. DG methods are locally conservative, stable, and high-order accurate methods. Originally, the DG method was realized with finite-difference and finite-volume methods. Later, it was extended to finite-element, *hp*-finite element, and spectral element methods. These make it easy to handle complex geometries, irregular meshes with hanging nodes, and approximations that have polynomials of different degrees in different elements. These properties have brought the DG method into many disciplines of scientific computing, such as computational fluid dynamics, especially for compressible flows, computational electromagnetics, computational plasma, semiconductor device simulation, chemical transport, and flow in porous media, as well as to a wide variety of problems such as Hamilton-Jacobi equations, elliptic problems, elasticity, and Korteweg-deVries equations. More details can be found in [6–8,17].

In the literature, the DG method has been used only in up to three dimensions. Since the Vlasov equation could be in higher dimensions, it brings new challenges to the DG method. Based on our successful experience of using a time-splitting scheme for solving the Vlasov equation directly [31], we have tried the DG method to substitute the semi-Lagrangian method in each substep, which is advanced in 2D phase spaces separately. Suppose that a 4D phase space Ω_K is composed of the tensor product of two 2D phase spaces, $\Omega^1_{K^1} \times \Omega^2_{K^2}$, and the total degrees of freedom (DOF) in $\Omega^i_{K^i}$ is F^i, then the total DOF in Ω_K is $F^1 \times F^2$.

Substituting t for z in Equation (13), we can write the time-splitting scheme combined with the DG method as follows:

$$\frac{\partial f(\vec{x},\vec{v},t)}{\partial t} + \frac{1}{2}\nabla_{\vec{x}} \cdot [\vec{V}(x,y) f(\vec{x},\vec{v},t)] = 0, \tag{15}$$
$$\frac{\partial f(\vec{x},\vec{v},t)}{\partial t} + \nabla_{\vec{v}} \cdot [\vec{U}(v_x,v_y) f(\vec{x},\vec{v},t)] = 0, \tag{16}$$
$$\frac{\partial f(\vec{x},\vec{v},t)}{\partial t} + \frac{1}{2}\nabla_{\vec{x}} \cdot [\vec{V}(x,y) f(\vec{x},\vec{v},t)] = 0. \tag{17}$$

The velocity in physical and velocity spaces is

$$\vec{V}(x,y) = \frac{\vec{v}}{v_b} = \frac{(v_x,v_y)}{v_b}, \tag{18}$$
$$\vec{U}(v_x,v_y) = \frac{q}{\gamma_b m v_b}[-\frac{1}{\gamma_b^2}\nabla\phi(x,y) + \vec{E}(x,y)]. \tag{19}$$

Another challenge in applying the DG method to solve the Vlasov equation is that the computer time increases as N^2, where N is the total DOF in physical and velocity spaces. Since in each subspace the transport equation is totally decoupled, it is well suited for large-scale parallel computing. Therefore, a highly scalable scheme could be developed, and the DG method matches this requirement, offering the added promise of high efficiency. More details can be found in [32].

2.5 Methods for Solving Poisson's Equation

Since the space charge effects are accounted for by solving the Poisson's equation, several numerical methods have been adopted. Each has advantages in specific conditions, which will be explained separately in the following. Poisson's equation is a standard second-order elliptic partial differential equation, which has been studied extensively in scientific computing. Some of the methods have been reported in our previous publications and will be presented here briefly. The other methods will be presented in more detail. The most common methods for solving the Poisson's equation, such as finite difference method, finite volume method, etc, are not included in the paper, since they are available in most text book of scientific computing. The methods presented below are numerical methods of high order accuracy.

2.5.1 Fourier Spectral Method (FSM)

The Fourier spectral method is the standard method for solving Poisson's equation in a box region in a cartesian coordinate system. The potential has been expanded in Fourier series in all three directions. Either periodic or Dirichlet boundary conditions can be applied in all three directions.

$$\phi(x,y,z,t) = \sum_{m=-M/2}^{M/2-1} \sum_{p=-P/2}^{P/2-1} \sum_{n=-N/2}^{N/2-1} \hat{\phi}(m,p,n,t) e^{-i\alpha m x} e^{-i\beta p y} e^{-i\gamma n z} \tag{20}$$

Then the Poisson's equation can be expressed as:

$$\begin{aligned} & f(x,y,z,t) = \Delta\phi(x,y,z,t) = \nabla^2\phi(x,y,z,t) \\ & = \sum_{m=-M/2}^{M/2-1} \sum_{p=-P/2}^{P/2-1} \sum_{n=-N/2}^{N/2-1} \hat{f}(m,p,n,t) e^{-i\alpha m x} e^{-i\beta p y} e^{-i\gamma n z} \\ & = - \sum_{m=-M/2}^{M/2-1} \sum_{p=-P/2}^{P/2-1} \sum_{n=-N/2}^{N/2-1} K(m,p,n)\hat{\phi}(m,p,n,t) e^{-i\alpha m x} e^{-i\beta p y} e^{-i\gamma n z} \qquad (21) \\ & K(m,p,n) = \alpha^2 m^2 + \beta^2 p^2 + \gamma^2 n^2 \qquad (22) \end{aligned}$$

From this it is easy to see that $\hat{\phi}(m,p,n,t) = \hat{f}(m,p,n,t)/K(m,p,n)$. Parallel algorithms for this method are given in Subsection 3.3.1.

2.5.2 Fourier hp-Finite Element Method (Fhp-FEM)

Since most accelerating devices have round apertures, solving Poisson's equation in a cylindrical coordinate system (CYLCS) is more appropriate in this case. In CYLCS, Poisson's equation has the following form:

$$\nabla^2\phi(r,\theta,z) = \frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2} + \frac{\partial^2\phi}{\partial z^2} = -\frac{\rho}{\epsilon_0}, \tag{23}$$

where ϕ is the electrostatic potential, ρ is the charge density, and ϵ_0 is the permittivity of vacuum. The 3D mesh is shown on the left of Fig. 3. The 2D mesh in the (r,θ) plane is shown on the right of Fig. 3. We use Gauss-Radau-Legendre quadrature points [18] in the first element close to the center to avoid the $1/r$ singularity. In this case, there are four unequal elements in the radial direction. Each element has nine points, and each pair of adjacent elements share the boundary points. Periodic boundary condition (B.C.) have been applied in the longitudinal and circumferential directions, and a natural B.C. has been applied in the center of the cylinder. A Dirichlet zero B.C. has been applied at $r = r_0$, where r_0 is the radius of the cylinder.

By transfroming from the (r, θ, z) space to the (r, m, n) space through Fourier Transform, we obtain

$$\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial\tilde{\phi}}{\partial r}) - \frac{m^2}{r^2}\tilde{\phi} - n^2\tilde{\phi} = -\frac{\tilde{\rho}}{\epsilon_0} \tag{24}$$

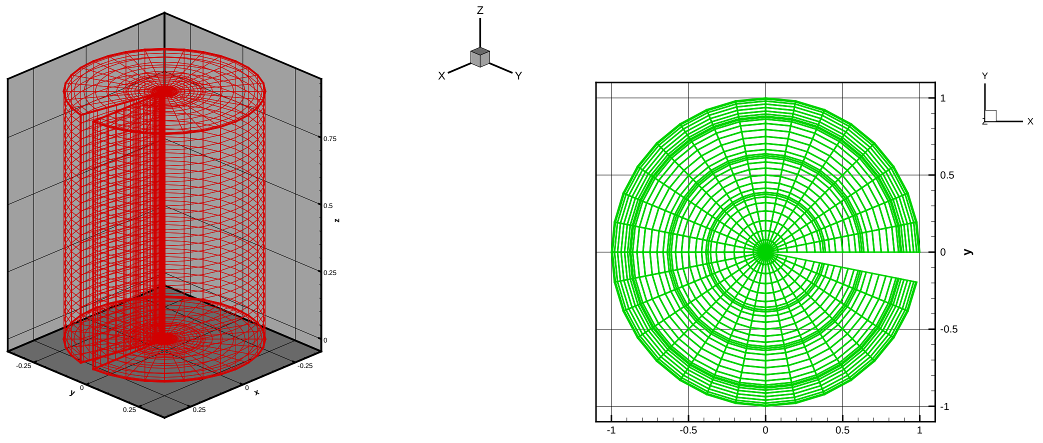
Fig. 3. 2D mesh in the (r,θ) plane

This yields a linear system of equations

$$A \cdot \hat{\phi} = \hat{f} \tag{25}$$

where A is a $(P+1) \times (P+1)$ matrix generated by the left-hand side terms, and $\hat{\phi}$, $\hat{f}$ are (P+1)-component vectors, where p,q=0,1,2,...,P. The equation (25) is solved directly as it is in one dimension. The size of A is not very large. Different Fourier modes m and n can be located on different processor to parallelize the solution. It will be explained in the next section and more details can be found in [30].

2.5.3 hp-Finite-Element Method (hpFEM)

The FEM originated in the 1940s. It was developed from elastic and structural analysis for civil and aeronautical engineering. The accuracy of FEM can be increased by using larger mesh, to be called *h*-FEM. It can also be improved by increasing the order of the bases, which is called *p*-FEM. The combination of these two approaches is called *hp*-FEM. It is an area in numerical methods which has attracted many computational mathematician. The *hp*-FEM has shown many advantages over popular, low-order methods in many applications [9,11,17,18,24]. The main advantages of *hp*-FEM are its flexibility in handling complex geometry and its high-order accuracy. Nearly all operations and data are local, including the derivative, interpolation, integration, solution of Poisson's equation, and transformation between physical and coefficient spaces. It can use two types of bases: nodal or modal bases. Both modal and nodal *hp*-FEM have been successfully applied to solve Poisson's and Vlasov equations. In our work, the modal *hp*-FEM uses a structured mesh in 2D, as shown on the left of Fig. 1, while the nodal *hp*-FEM uses the unstructured mesh in 2D geometry as shown on the right of Fig. 4.

With the Shur complement technique, the solution of a linear system $A \times x = b$ can be divided into two parts for boundary and internal modes, as shown on the right of Fig. 2. The boundary modes are solved iteratively by using a conjugate gradient

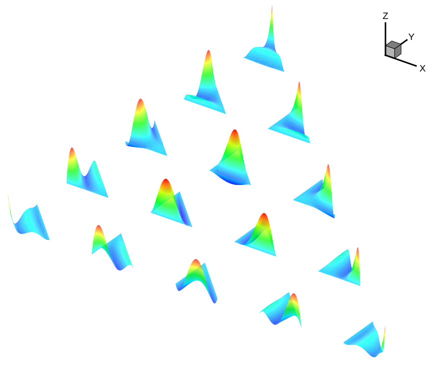
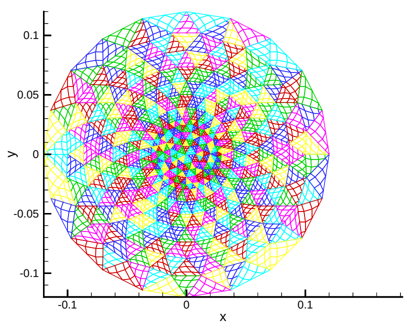

Fig. 4. Modal bases (left) and mesh (right) on triangle.

method. The internal modes in each element can then be solved directly. The discrete system for the Poisson equation can be written as follows:

$$\begin{pmatrix} A_{bb} & C_{bi} \\ C_{bi}^T & A_{ii} \end{pmatrix} \begin{pmatrix} u_b \\ u_i \end{pmatrix} = \begin{pmatrix} f_b \\ f_i \end{pmatrix}$$

, where b and i correspond to boundary and interior variables and u_b and u_i can be solved separately by the following equations:

$$(A_{bb} - C_{bi}A_{ii}^{-1}C_{bi}^T)u_b = f_b - C_{bi}A_{ii}^{-1}f_i \tag{26}$$

$$u_i = A_{ii}^{-1}(f_i - C_{bi}^T u_b) \tag{27}$$

More details can be found in [9,18].

Since the Vlasov solver on an unstructured grid is built on the DG framework, It is better to solve the Poisson's equation with the DG method too. The method been used in DG framework is called interior penalty (IP) method [2,10,26], more details can be found in [32].

3 Parallel Methods

In order to use petascale supercomputers, the numerical methods explained above must be parallelized. As in the preceding section, the parallel methods will be explained in three subsections. Since the Poisson solver is of special importance in BDS and its parallelization is the most challenging, it is explained separately.

3.1 Parallel Method for PIC Software

A BDS has two components: particle tracking and space charge (SC) calculation. For the PIC solver in PTRACK, we use domain decomposition only for the SC component of the calculation, which is the parallel Poisson solver explained below. The parallel algorithm for the PTRACK is shown in Fig. 5. At the beginning of the calculation, the internally generated or read-in initial particle distribution is equally distributed among all the processors. That is, each processor has only part of the total particles. But each processor has information about the full external fields for the beamline or accelerator element being simulated. The SC grid is also defined on all the processors. Before every tracking step the internal SC fields of the beam must be computed and combined with the external fields. The first step in the calculation of SC fields is the particle charge deposition on the nodes of the SC grid. This is done locally; that is, each processor deposits the charges of particles that are located on it. At the end of this step, every processor will have a partial SC distribution including only the charge of its particles. To calculate the SC distribution of the whole beam, we sum the partial SC distributions on the SC grid using the "MPI_Allreduce" routine of the MPI Library [23], To use domain decompositions of the Poisson solver shown in Fig. 7, we subdivide the full SC grid into smaller-scale SC grids using 1D, 2D, and 3D space decompositions. Each processor will have a local SC grid containing only part of the SC data. FSM has been used for all these three decompositions. Since the FFT is a global transformation, in order to perform FFT in one direction all data in that direction should be collected on one processor. Then after FFT, they will be distributed back to their original processors. After solving the Poisson equation, we have the solution in the form of potential data distributed on the local grids of each processor. In order to have all potential on each processor, a second global communication using "MPI_Allreduce" will bring the potential data from the local grids to the global grid to be ready for the tracking part of the calculation. The complete procedure is summarized in Fig. 5. The domain decomposition method has been used for the parallelization of other Poisson solvers, such as the Fourier *hp*-Finite Element Method. More details can be found in [28].

3.2 Parallel Method for direct Vlasov Solvers

For 1P1V, which is equivalent to a 2D simulation, 1D domain decomposition in both physical x and velocity v_x space has been adopted. Two MPI communicators, *comx* and *comv*, have been generated for operations in different spaces. For 2P2V simulation using the structured grid, the parallel model performs 2D domain decompositions in both physical (x, y) and velocity (v_x, v_y) planes; therefore, completely 4D domain decomposition has been used, as shown on the left of the Fig. 6. This makes it easy to use a large number of processors and is particularly helpful for direct solution of the Vlasov equation. Three splitting substeps are associated with different com-

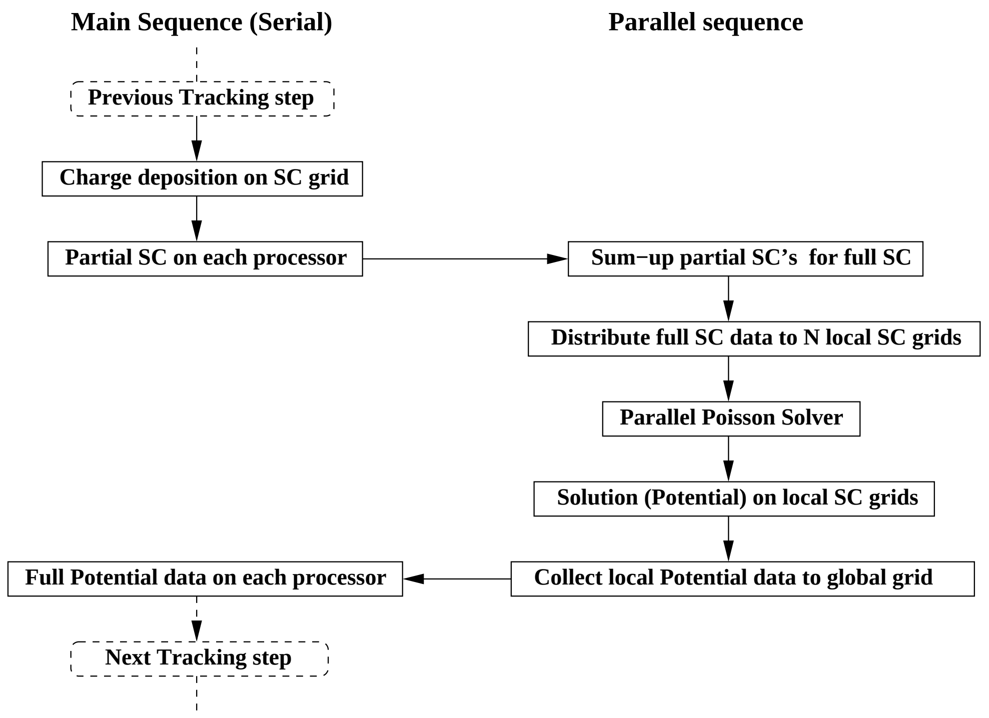
Main Sequence (Serial)
Parallel sequence
Previous Tracking step
Charge deposition on SC grid
Partial SC on each processor
Sum−up partial SC's for full SC
Distribute full SC data to N local SC grids
Parallel Poisson Solver
Solution (Potential) on local SC grids
Collect local Potential data to global grid
Full Potential data on each processor
Next Tracking step

Fig. 5. Parallel algorithm of PTRACK

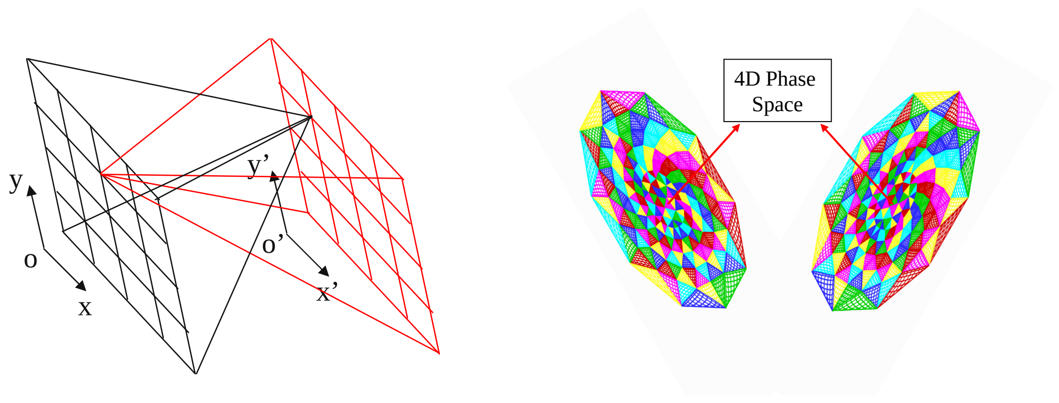

Fig. 6. Structured (left) and unstructured (right) grid in 2D plane

municators. This makes it possible to solve the Vlasov in high dimensions. In these simulations, two more communicators, *comxy* and *comvxvy*, have been generated for operations in different planes. The communicator *comxy* contains all processors with the same (v_x, v_y) location, and *comvxvy* contains all processors with the same (x, y) location. These two communicators are used for computing the beam statistics. When using the unstructured grid, only two communicators can be created, which are *comxy* and *comvxvy*, because it is impossible to distinguish x from y on the (x, y) plane and v_x from v_y on the (v_x, v_y) plane. Details in each case can be found in [31,32].

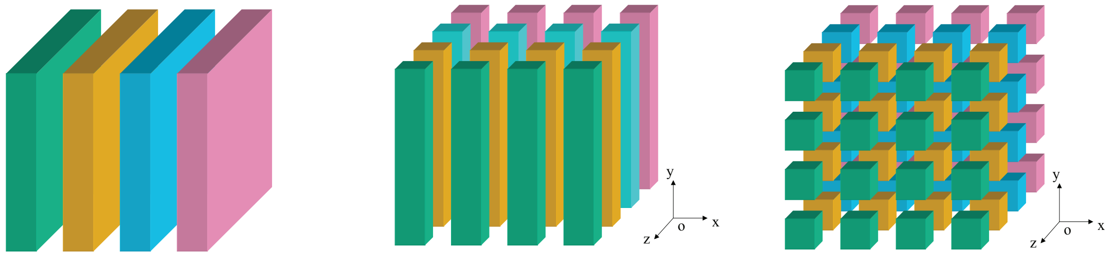

Fig. 7. Parallel models for the Fourier spectral method

3.3 Parallel Methods for Poisson Solvers

Because of the global nature of Poisson's equation, involving the whole charge distribution to calculate an effective self-field created by all the particles in the beam, its parallelization is the most challenging part in developing scalable algorithms for BDS. Especially when the grid is small and a large number of processors is used. In this section, we summarize the parallel algorithms developed with the domain decomposition method.

3.3.1 Parallel Method for FSM

Since FSM is the most popular method for solving Poisson's equation, more work has been devoted to this method. Three domain decomposition methods have been implemented, as shown in Fig. 7. With the appropriate model, it is easy to use tens of thousands of processors with a relatively small grid for the space charge calculation. For example, if the grid used for Poisson's equation is 32^3, then the maximum number of processors that can be used is 32 for the 1D decomposition model, $32^2 = 1024$ for the 2D decomposition model and $32^3 = 32,768$ for the 3D decomposition model. If the data is not located in one processor in any direction, a global $MPI\_Alltoall$ needs to be called to transfer the data into one processor. Another call is needed to transfer back the data to their original processors. Good scaling has been obtained with a relatively small grid for the space charge calculation, see [27–29].

3.3.2 Parallel Method for Fhp-FEM

For the Fhp-FEM in CYLCS, 2D domain decomposition has been developed, as shown in Fig. 8. The model on the right has the benefit of solving the 1D linear system $A \times x = b$ on each processor, as they are totally decoupled. While the model on the left has to solve the boundary modes first, then the internal modes in each elements in the r direction. Since the element number in the r direction is not very

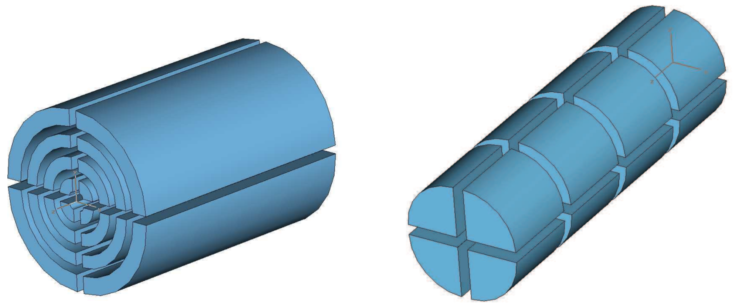

Fig. 8. Parallel models for the Fourier *hp*-finite element method

large, this model is also efficient and it is being used for BDS. It is also possible to develop 3D domain decomposition, as was done for FSM, which can use a large number of processors. Details can be found in [31].

3.3.3 Parallel Method for hp-FEM

Parallel Poisson solvers using *hp*-FEM are achieved by using the Shur complement technique, shown on the right of Fig. 2. The boundary mode, equation (26), has been solved using an iterative conjugate gradient (CG) method, whereas the internal mode of each element, equation (27), has been solved by the direct method. The mesh should be partitioned appropriately so that all processors have approximately the same numbers of elements to ensure load balance. In order to speedup the CG method, an efficient preconditioner can be used. For the multigrid technique, both the coarse and fine meshes use the same mesh partition. This makes it convenient to perform prolongation and restriction operations on each element. A good mesh partition is to minimize the modes located on the interfaces of different processors, as this will reduce the communication cost during global operations. More details can be found in [32].

4 Comparison and Discussion

In this section we compare the performance of the numerical methods presented in the paper. The comparison may help to understand these methods in more depth.

4.1 PIC vs. DVM

The common point of PIC and DVM is that they both need to solve the Poisson's equation during each time step. PIC and DVM have many differences, however. PIC uses a Lagrangian approach, while DVM uses an Euler approach. Using Lagrangian

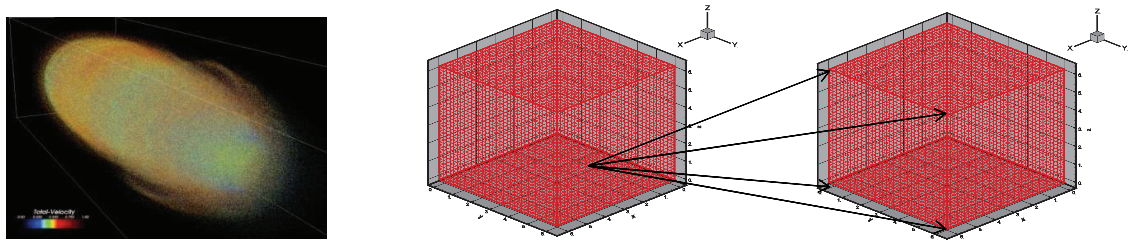

Fig. 9. Phase space for PIC (left) and Vlasov (right)

approach means that the macro particles are traced during acceleration. Using Euler approach means that the probablility function in phase space is studied instead. As shown in Fig. 9, PIC uses macro particles to represent a real beam, while DVM uses fixed grid points. The left of Fig. 9 shows a beam bunch of 10^8 particles in 3D physical space. The middle and right of Fig. 9 show that each point in the 3D physical space has a 3D velocity space associated with it. This makes DVM high dimensional while in the PIC method a single macro-particle has a very well defined space and velocity coordinates. In the PIC method, macro particles represent the same number of real beam particles, while the distribution function has different values on the grid points in DVM. Since PIC uses macro particles, the statistics of the beam is obtained on particles, while it is obtained on grid points for DVM. Moreover, PIC is at most in 3D, while DVM could be in 6D. In order to better understand their relation, we can shrink the velocity space in DVM to just one point, and transform the DVM grid in physical space to exact matching the beam bunch, DVM now translates to PIC method suppose that the DVM distribution function value at a grid point is equal to the particle numbers represented by the macro particle located at that position. Another difference between PIC and DVM is that DVM solves the transport equation in the velocity space, while PIC apply force kick to represent the integration in the velocity space. This greatly simplify the method which makes PIC easy to implement. From these comparisons, it is clear that both PIC and DVM are just different numerical methods for solving the Vlasov equation. They study same beam dynamics physics. Since macro particle represents same number of real particles, PIC method is difficult to capture beam halo, where less than particle number represented by one macro partile exists. DVM supposes to complement this by using Euler approach, but large derivative and interpolation errors prohibit its use in real applications. This needs more efficient numerical methods which can simulate the real physics more accurately and efficiently. This will be investigated more in the future.

4.2 SLM vs. DGM

SLM and DGM are two efficient methods for time integration. Both methods have strong stability. While DGM has stronger stability than the continuous Garlekin method (CGM), SLM is supposed to be unconditionally stable. DGM has strict limitation on the time step size while SLM is less limited. Therefore, with the appropriate choice of the time step size, SLM could be more stable than DGM. Since a beam is a concentrated bunch of particles, the distribution function could have sharp structures; this makes the errors of interpolation and derivation much larger than in usual plasma simulations. The greatest challenge comes from these large operator errors. In practical, SLM is easier to develop than the DGM. For the parallelization, as DG has completely independent bases on each element, the communication is less except for the DG Poisson solver. While the SLM has to redistribute all back tracing points on different processors, which makes it more complicated than DG. There is also a load balancing problem in SLM, since only small portion of the phase space has probability function larger than some value. Depending on the mesh partition, usually most back tracing points are located on some processors. This wastes the computing time, while DG has more balanced computing load as all processor have equal mount of work to be done.

4.3 Poisson Solvers

Several numerical methods to solve Poisson's equation have been presented briefly. Among them, FSM is the most popular and efficient one. Since the Fast Fourier Transform (FFT) algorithm can be used and public FFT libraries are easily available. The drawback of FSM is the global characteristics of Fourier Transformation. This requires the data in one direction must locate on one processor to complete the FFT. The large ratio of communication cost over the FFT cost makes the parallel efficiency low on large number of processors. As the total time is small compared to other time in PIC and direct Vlasov solvers, the overall parallel efficiency still good. This fits in all three parallel methods been developed for FSM, especially the one using 3D domain decomposition which can be run efficiently on tens of thousands of processors. Fhp-FEM is designed for CYLCS, which gives a more accurate solution for a cylindrical geometry because the B.C. is more precisely defined on a cylinder wall. Since the hp-FEM has been used in the radial direction, this limit the number of processors that can be used. The number of processors used in Fhp-FEM usually is less than that of FSM. High-order accuracy can be achieved by *hp*-FEM Poisson solvers when using high-order polynomials. Similar to FSM, there is some inherent restriction on scalability. Since the modes on interface between elements is solved globally through conjugate gradient method, the parallel efficiency usually become worse when the number of processors increases. The parallel efficiency usually becomes better as the polynomial order increases for both CG and DG methods. The convergence of CG usually is better than the DG method. Using an unstructured grid makes it easy to handle complex geometries, which also requires more work in the development. These Poisson solvers are more or less effective in different configurations and the choice of the appropriate solver should be based on the geometry and boundary conditions.

4.4 Structured vs. Unstructured meshes

Unstructured mesh is more appropriate for complex geometries, and finer mesh can be generated in the central part of the beam. Both structured and unstructured meshes can apply CG or DG methods, but of course the unstructured mesh is more complicated to develop. One significant difference between structured and unstructured meshes for the 2P2V Vlasov solvers is that we don't have access to the same statistical quantities. As explained in the previous section for parallel models for Vlasov solvers, using an unstructured mesh it is not possible to seperate the x and y directions. There is no way to find all (x, v_x) points for a given (y, v_y) point. Similarly There is no way to find all (y, v_y) points for a given (x, v_x) point. Therefore, the statistical quantities XX'_{rms} and YY'_{rms} can not be calculated, while the statistical quantities X_{rms}, Y_{rms}, X'_{rms}, and Y'_{rms} can be obtained on both meshes. The definitions of these statistics can be found in [31].

5 Summary

We presented a summary of efficient numerical and parallel methods used and developed in our research for large scale beam dynamics simulations during the past 5 years at ANL. Hope it will be helpful to other researchers in accelerator simulations. They have been explained in brief and readers should consult the cited journal articles for more details. These numerical methods have been discussed in three categories: PIC, DVM, and Poisson solvers. Parallel methods for all of them have also been described in following. These methods are intended to enable the numerical methods to take advantage of petascale supercomputers. The parallel methods have been implemented in software packages and successfully run on tens of thousands of processors of the IBM Blue Gene/P at the Argonne Leadership Computing Facility. The parallel PIC solver, PTRACK, has been used for large-scale beam dynamic optimization and real accelerator simulations. The parallel DVM solvers have been developed in 1P1V and 2P2V simulations, they have provided much more detail information on the physics, such as halo generation and filamentation in low dimension and with smooth initial distribution functions. Due to the large derivation and interpolation errors, they are difficult being used in real accelerator simulations now. More efficient numerical methods need to be developed to simulate complex beam dynamics more accurately. This will be investigated more in the future. More ever, direct Vlasov solevrs have more potential applications in plasma physics. Necessary for the success of PIC and VDM, several numerical methods for solving the Poisson's equation in different situations have been presented and compared. They are critical to perform the large scale simulations of BDS. Overall, these numerical and parallel methods serve as a basis for BDS, and they can be applied to more broad areas in the future. The numerical methods discussed in this paper are only summary in our research at ANL, there are many other efficient numerical methods which have not been included. In the future, more efficient numerical methods will be adopted to improve current BDS. At the same time, the current ones will be further optimized.

Acknowledgment

This paper is based on research conducted in the Physics and Mathematics and Computer Science Divisions at ANL during the past 5 years. Besides the co-authors listed, many other people have contributed to the work reported here, and we thank them for their contributions. We also thank Gail Pieper in MCS at ANL for the proofreading. This work was supported by the U.S. Department of Energy, Office of Nuclear Physics, under Contract No. DE-AC02-06CH11357.

References

[1] T.D. Arber and R.G.L. Vann, A critical comparison of Eulerian-grid-based Valsov solvers, *J. Sci. Comp.*, **180**, 339–357 (2002).

[2] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, *SIAM J. Numer. Anal.,* **39**, no. 5, 1749–1779 (2002).

[3] V.N. Aseev, P.N. Ostroumov, E.S. Lessner, and B. Mustapha, TRACK: The new beam dynamics code. In *Proceedings of PAC-05 Conference,* Knoxville, Tennessee, May 16–20, 2005.

[4] C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang, *Spectral Methods in Fluid Mechanics*, Springer-Verlag, New York, 1987.

[5] C.Z. Cheng and G. Knorr, The integration of the Vlasov equation in configuration space, *J. Sci. Comp.*, **22**, 330–351 1976.

[6] B. Cockburn, *Discontinuous Galerkin methods for convection-dominated problems.* In *High-Order Methods for Computational Physics*, edited by T. Barth and H. Deconink, Lecture Notes in Computational Science and Engineering, Vol. 9, Springer Verlag, Berlin, 1999, pp. 69–224.

[7] B. Cockburn, G. Karniadakis, and C.W. Shu, eds., The development of discontinuous Galerkin methods. In *Discontinuous Galerkin Methods. Theory, Computation and Applications*, Lecture Notes in Computational Science and Engineering, Vol. 11, Springer Verlag, Berlin, 2000, pp. 3–50.

[8] B. Cockburn and C.W. Shu, Runge-Kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comp., **16,** 173–261 (2001).

[9] M.O. Deville, P.F. Fischer, and E.H. Mund, *High-Order Methods for Incompressible Fluid Flow,* Cambridge University Press, Cambridge, 2002.

[10] J. Douglas, Jr., and T. Dupont, *Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods*, Lecture Notes in Physics, Vol. 58, Springer Verlag, Berlin, 1976.

[11] M. Dubiner, Spectral methods on triangles and other domains, *J. Sci. Comp.*, **6**, 345–390, (1991).

[12] F. Filbet and E. Sonnendrücker, Modeling and numerical simulation of space charge dominated beams in the paraxial approximation, *Mathematical Models and Methods in Applied Sciences*, **16**, no.5, 763–791 (2006).

[13] Paul Fischer, James Lottes, David Pointer, and Andew Siegel, Petascale algorithms for reactor hydrodynamics, *Journal of Physics: Conference Proceedings*, **125**, no. 1, 012976 (2008).

[14] D. Gottlieb and S.A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications,* SIAM-CMBS, 1977.

[15] L. Greengard and J. Lee, A direct adaptive Poisson solver of arbitrary order accuracy, *J. Comput. Phys.*, **125**, 415–424 (1996).

[16] J.S. Hesthaven and T. Warburton, Nodal high-order methods on unstructured grids, I: Time-domain solution of Maxwell's equations, *J. Comput. Phys.*, **181**, 186–221 (2002).

[17] J.S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis and Applications,* Springer, New York, 2008.

[18] G.E. Karniadakis and S.J. Sherwin, *Spectral/*hp *Element Methods for CFD,* Oxford University Press, London, 1999.

[19] T. Koornwinder, Two-variable analogues of the classical orthogonal polynomials. In *Theory and Applications of Special Functions*, edited by M. Ismail and E. Koelink, Academic Press, San Diego, 1975.

[20] A.T. Patera, A spectral element method for fluid dynamics: Laminar flow in a channel expansion, *J. Comput. Phys.*, **54,** 468–488 (1984).

[21] W.H. Reed and T.R. Hill, *Triangular mesh methods for the beutron transport equation*, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.

[22] G. Strang and G.J. Fix, *An Analysis of the Finite Element Method,* Wellesley-Cambridge Press, Wellesley, 2008.

[23] B. Szabó and I. Babuška, *Finite Element Analysis,* Wiley, New York, 1991.

[24] J. Shen and T. Tang, *Spectral and High-Order Methods with Applications,* Science Press of China, 2006.

[25] E. Sonnendrücker, F. Filbet, A. Friedman, E. Oudet, and J.-L. Vay, Vlasov simulations of beams with a moving grid, *Comput. Phys. Comm.*, **164**, 2390–395 (2004).

[26] M.F. Wheeler, An elliptic collocation-finite element method with interior penalties, *SIAM J. Numer. Anal.*, **15**, 152–161 (1978).

[27] J. Xu, Benchmarks on Tera-scalable Models for DNS of Channel Turbulent Flow, *Parallel Computing.* **33/12**, 780-794, 2007.

[28] J. Xu, B. Mustapha, V.N. Aseev, and P.N. Ostroumov, Parallelization of a beam dynamics code and first large scale radio frequency quadrupole simulations, *Physics Review Special Topic–Accelerator and Beams,* **10**, 014201 (2007).

[29] J. Xu, B. Mustapha, V. N. Aseev, and P.N. Ostroumov, Simulations of high-intensity beams using BG/P supercomputer at ANL. In *Proceedings of HB 2008,* Nashville, Tennessee, Aug. 25–29, 2008.

[30] J. Xu and P.N. Ostroumov, Parallel 3D Poisson solver for space charge calculation in cylinder coordinate system, *Computer Physics Communications,* **178,** 290–300 (2008).

[31] J. Xu, B. Mustapha, P.N. Ostroumov, and J. Nolen, Solving 2D2V Vlasov equation with high order spectral element method, *Communication in Computational Physics,* **8**, 159–184 (2010).

[32] J. Xu, P.N. Ostroumov, J. Nolen, and K.J. Kim, Scalable direct Vlasov solver with discontinuous Galerkin method on unstructured mesh, *SIAM Journal on Scientific Computing.*

[33] J. Xu , P. N. Ostroumov, B. Mustapha, M. Min and J. Nolen Developing Peta-scalable Algorithms for Beam Dynamic Simulations, *Proceeding of IPAC10*, Kyoto, Japan, May 23-28, (2010).