A Novel Method for Circuits of Perfect Electric Conductors in Unstructured Particle–In–Cell Plasma–Object Interaction Simulations

S. Marholm, D. Darian, M. Mortensen, R. Marchand and W. J. Miloch

***Abstract*—A novel numerical method has been developed that incorporates electrically conducting objects into Particle–In–Cell simulations of electrostatic plasma. The method allows multiple self-consistently floating circuits of objects connected through voltage and current sources, and arbitrary geometries can be approximated using an unstructured mesh. The constraints of the circuits are directly incorporated into the matrix formulation of the Poisson problem, and solved simultaneously.**

This method has been implemented in a new code, PUNC, suitable for rapid prototyping. The results for a conducting sphere immersed in Maxwellian plasma is in good agreement with that of Laframboise. PUNC has also been used to survey various methods of taking the gradient of the potential to obtain the electric field, and allows second-order accurate electric fields.

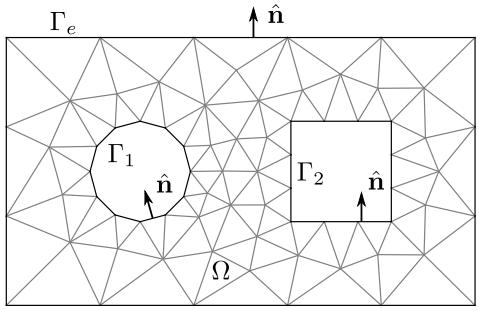

Fig. 1. Example of a mesh on a 2D simulation domain Ω with exterior boundary Γ_e and two objects delimited by two interior boundaries Γ_1 and Γ_2. The normal vector $\hat{\mathbf{n}}$ on the boundaries points away from the simulation domain (and thus into the objects). The mesh is created using gmsh [13].

I. Introduction

It is apparent that as computational power becomes ever more available, more and more problems are addressed using computer simulations. This includes studies of spacecrafts and their instruments in the upper atmosphere, which are often modelled as electric conductors. Understanding how they influence the surrounding plasma is crucial in the interpretation of their measurements. In order to study kinetic phenomena such as charging from the collection of background plasma particles, the approach of choice is to use kinetic simulation models such as the fully self-consistent *Particle–In–Cell* (PIC) method.

Several PIC models have been implemented for plasma-object interaction, each with its own strengths, for instance DiP3D [24], EMSES [25], MUSCAT [26], Nascap-2k [21], CPIC [9], SPIS [33] and PTetra [22]. Of these, DiP3D, EMSES and MUSCAT all uses uniform cartesian structured grids. This is restrictive both because objects must be approximated by staircased geometries, and because the mesh cannot be made finer near the object than far away. On the other hand, using an unstructured mesh such as the one depicted in Fig. 1, objects of arbitrary geometry can be approximated by piecewise linear facets (faces in 3D or edges in 2D). SPIS and PTetra uses the *Finite Element Method* (FEM) on such meshes. It should be mentioned that Nascap-2k and CPIC also fits the mesh to the object, but using different techniques. Nascap-2k uses the *Boundary* Element Method on an unstructured *surface* mesh for the spacecraft charging part, but for self-consistent calculation of particle trajectories in its external field Nascap-2k still uses a cartesian mesh, although with nested refinements. Finally, CPIC uses a curvilinear structured mesh that it boundary-fits to the object's geometry, and which it transforms to a logical space where the mesh is uniform and cartesian. While such a mesh certainly has its merits in a PIC code, it is not as flexible and generally applicable as unstructured meshes. It is perhaps this generality which has been responsible for the success of FEM, a success which has lead to continuous development, improvement and maintenance of many modern and accessible tools. It is this ecosystem of knowledge and tools we build upon when we present our novel method for incorporating objects.

Both FEM-PIC codes mentioned solves the Poisson equation with Dirichlet boundaries on the object and relies on ad-hoc techniques to self-consistently obtain the correct value on this boundary. SPIS introduces a separate circuit solver in addition to the Poisson solver, while PTetra introduces capacitances and solves the Poisson equation twice per time step. We present an alternative approach where the unknown object potential is treated as an integral part of the Poisson boundary value problem and incorporated directly into the resulting stiffness matrix. The method is generalized to multiple objects, possibly connected by voltage and current sources. Moreover, while we focus on the PIC application, the method really applies to all electrostatic simulations of perfectly conducting objects where the charge is known.

A new rapid prototyping code called PUNC — *Particles-in-Unstructured-Cells* — has been developed to verify the method. PUNC is implemented in Python and centered around FEniCS [19], [1], a modern and efficient environment for solving variational problems, including the assembly of matrices stemming from the FEM method as well as iterative solutions

through efficient third-party libraries. The flexibility inherited from both Python and FEniCS makes it easy to test new concepts, and to add custom diagnostics. At last, PUNC plays well with Gmsh [13] for mesh generation and ParaView [2] for visualization.

With PUNC we have also assessed a variety of methods for taking the gradient of the solution of the Poisson problem to obtain the electric field. In particular, achieving a second order accurate electric field allows the PIC simulations to be second order accurate overall.

First, in Sec. II we present background material on the PIC method. Then, since the *particle* methods and *mesh* methods in the PIC method are based on the separate disciplines of *ordinary differential equations* and *partial differential equations*, they are treated separately in Sec. III and Sec. IV, respectively. The multi-object method is mainly explained in the mesh section, and is more generally applicable than PIC simulations, but in the PIC setting input to the mesh method is derived from the particles. Subsequently, the interaction between the mesh and particles are explained in Sec. V, before verification and validation of the method is presented in Sec. VI. At last follows the discussion and conclusion in Sec. VII and Sec. VIII, respectively.

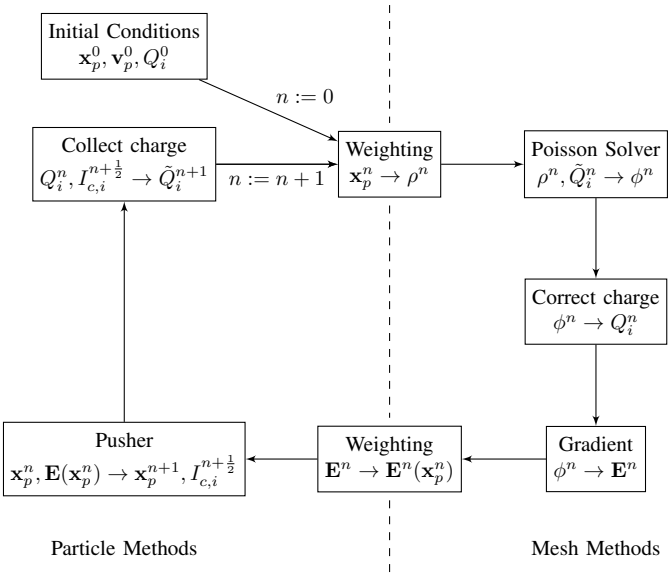

Fig. 2. Overview of the PIC cycle as used in PUNC. Each step indicates new quantities computed from previous quantities. Subscripts p and i are particle and object indices, respectively, and computations with these indices must be performed for all particles/objects. Quantities without subscripts are field-quantities. The time-step of each quantity is indicated as a superscript.

II. Background

We shall begin by giving an overview of the PIC method. Knowing the forces acting on a particle p (and its initial conditions), its position $\mathbf{x}_p$ can be obtained at any time by numerically integrating Newton's second law. However, if we were to add up the electric forces between each pair of N_P particles, known as a *particle–particle* approach, the computational complexity per timestep would be $\mathcal{O}(N_P^2)$. To reduce this, the PIC method uses a combined *particle–mesh* approach, where field(s) are introduced, and the force experienced by the particles are derived from these. More specifically, a particle in an electric field $\mathbf{E}$ and magnetic flux density $\mathbf{B}$ is governed by

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = \mathbf{v}_p, \qquad \frac{\mathrm{d}\mathbf{v}_p}{\mathrm{d}t} = \frac{q_p}{m_p}(\mathbf{E} + \mathbf{v}_p \times \mathbf{B}), \tag{1}$$

where $\mathbf{v}_p$ is the velocity of the particle, q_p its charge, and m_p its mass. t is time. The chief feature of a plasma as opposed to charged particles in externally imposed fields is the collective forces the particles exerts on one another. We shall here assume the electrostatic approximation, where any collective magnetic force are considered negligible, but allow a constant, homogeneous magnetic flux density $\mathbf{B}_0$ to be specified by the user. Regardless, $\mathbf{E}$ must still be determined from the position and charge of the particles using Maxwell's equations, which for electrostatics reduces to [7, p. 75]

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}, \qquad \nabla \times \mathbf{E} = \mathbf{0}, \tag{2}$$

where ρ is the charge density and ε_0 is the permittivity of vacuum. To obtain the charge density, space must be discretized onto a mesh, and the charge of each particle assigned to nodes in this mesh by some weighting scheme. When Maxwell's equations are solved, another weighting scheme is used to interpolate the electric field back to the particles, such that their positions can be advanced one time step. Since this again alters the charge density, the electric field must be re-computed before the next position update, and we have what is commonly referred to as the PIC cycle. See Fig. 2.

The operations count of one PIC cycle is $\mathcal{O}(N_P) + \beta(N_G)$ where N_G is the number of nodes in the grid and β is some function depending on the mesh solver [16, p. 21], typically $\mathcal{O}(N_G \log N_G)$ or even $\mathcal{O}(N_G)$ when multigrid methods are used [17, p. 137]. Usually it's necessary with many particles per cell, in the order of tens, to keep the statistical noise sufficiently low, hence $N_P \gg N_G$. Clearly, this is better than the particle–particle approach. Further details on the PIC method in general can be found in [5], [16], [18].

A. Simulation Particles

The number of particles in a simulation volume is often very high. For instance, the electron density is in the order of 10^{10}–$10^{12}\,\mathrm{m}^{-3}$ in the ionosphere [29, p. 161]. Integrating the trajectory of such vast numbers of particles is impractical both in terms of speed and memory, even for the particle–mesh approach. Therefore, it is customary in PIC codes to use *simulation particles*, each corresponding to w_s physical particles [5]. Given as inputs the number of simulation particles per specie at start-up, N_s, and the density of physical particles, w_s can be determined from the ratio of densities of simulation particles (n_s') to physical particles (n_s):

$$w_s = \frac{n_s}{n_s'} = \frac{n_s}{N_s}\,\mathrm{Vol}(\Omega) \tag{3}$$

where $\mathrm{Vol}(\Omega)$ is the volume of the domain.

The charge and mass of a simulation particle of species s is then

$$q'_s = w_s q_s, \qquad m'_s = w_s m_s. \tag{4}$$

It is easily verified that physical characteristic scales such as the plasma frequency are invariant under this transformation. The reduced number of particles will of course increase the statistical noise which is proportional to $(n'_s)^{-1/2}$ [30, p. 98].

III. Particle Methods

In prinicple any correct numerical integration of (1) can be used to update the position of the particles, but PUNC uses the Boris method for its cheap execution, long-term stability and $\mathcal{O}(\Delta t^2)$-accuracy, Δt being the timestep [31]. Let a superscript n inidcate that a quantity is approximated at time $t = n\Delta t$. The Boris method uses a staggered temporal grid where the position is stored at integer timesteps, $\mathbf{x}_p^n$, while the velocity is stored at half-integer timesteps, $\mathbf{v}_p^{n+\frac{1}{2}}$. To offset the velocity by half a timestep, a normal forward Euler step [35, p. 317] is performed during initialization. In addition to position and velocity, each particle is equipped with their own charge q_p and mass m_p. This is more flexible than dividing the particles into different species, although also a bit more memory demanding.

The particles are organized such that each cell in the mesh has one list of particles residing in that cell. This allows us to perform the weighting described in Sec. V in a cell-wise manner. After updating the particle positions one must therefore also update which cell they belong to. This is done following the approach in PTetra [22], described in the following.

Let $\hat{\mathbf{n}}_f$ be the outwards-pointing normal vector of a facet f of the old host cell, and $\mathbf{x}_f$ any arbitrary point on that facet, for instance one of the vertices. Further, let $x_{p,f} = (\mathbf{x}_p - \mathbf{x}_f) \cdot \hat{\mathbf{n}}_f$. Then the particle remains in the old cell if $x_{p,f} \leq 0$ for all facets of that cell. If not, it is likely that the new host cell is adjacent to the facet with the largest $x_{p,f}$. However, since the particle may have travelled beyond this cell, the algorithm is applied recursively until a cell is found with $x_{p,f} \leq 0$ for all f. Note that this requires pre-computing and storing which cell is adjacent to every facet of every cell in the mesh. If a particle passes through a facet of which there is no adjacent cell, we know that the particle crossed a boundary, and moreover, we know which boundary. In this case we store the adjacent boundary instead of the adjacent cell.

Since the position update per time step is usually small compared to the cell size, the recursion depth is on average very small. It is important that newly inserted particles are not just placed in the list of a random host cell, expecting the recursive algorithm to properly relocate it. Not only will the recursion depth be large for such particles, but if the domain is non-convex the algorithm may erroneously find that the particle has left the domain. Instead, we use an axis-aligned bounding box tree to locate the correct cell of newly inserted particles[1].

[1]Axis-aligned bounding box trees and adjacency information comes out-of-the-box in FEniCS.

A. Initial Conditions

For each species s, the simulation domain Ω is initially populated with the given number of simulation particles, N_s. By default, a uniform probability distribution $\mathcal{U}(\Omega)$, is used to generate a random position for each particle in the simulation domain. However, PUNC offers the user to specify the initial probability distribution function for the position of particles of each species. In order to assign initial velocities to simulation particles of species s, a velocity probability distribution function (pdf) $f_s(\mathbf{v})$, must be specified. One of the extensively used distributions in plasma simulations is the shifted-Maxwellian velocity distribution [22]:

$$f_{\mathrm{M},s}(\mathbf{v}) = \prod_{i=1}^{D} f_{\mathrm{M}}(v_i; v_{\mathrm{D},s,i}, v_{\mathrm{th},s}),$$

where

$$f_{\mathrm{M}}(v_i; v_{\mathrm{D},s,i}, v_{\mathrm{th},s}) = \frac{1}{\sqrt{2\pi} v_{\mathrm{th},s}} \exp\left[-\frac{(v_i - v_{\mathrm{D},s,i})^2}{2 v_{\mathrm{th},s}^2} \right], \tag{5}$$

is the distribution of the velocity component v_i. Here, D is the number of geometric dimensions of Ω, $v_{\mathrm{D},s,i}$ is the $i-$component of drift velocity, and $v_{\mathrm{th},s} = \sqrt{k_{\mathrm{B}} T_s / m_s}$ is the thermal velocity of species s, where T_s is the temperature and k_{B} is the Boltzmann constant. There exist different methods to generate random velocities from this distribution [6], [23]. In the inverse transform sampling method [11], first the cumulative distribution function (cdf) for each velocity component is computed. The analytical expression for the cdf of component v_i is given by [23]:

$$F_{\mathrm{M}}(v_i; v_{\mathrm{D},s,i}, v_{\mathrm{th},s}) = \int_{-\infty}^{v_i} f(v'_i)\, \mathrm{d}v'_i = \frac{1}{2} \operatorname{erfc}\left(\frac{v_{\mathrm{D},s,i} - v_i}{\sqrt{2} v_{\mathrm{th},s}} \right), \tag{6}$$

where erfc is the complementary error function [3]. Then, by inverting (6), the velocity component $v_{s,p,i}$ for each simulation particle is obtained by

$$v_{s,p,i} = v_{\mathrm{D},s,i} - \sqrt{2} v_{\mathrm{th},s} \operatorname{erfc}^{-1}(2F_{p,i}), \tag{7}$$

where $F_{p,i} \in [0,1]$ is a uniformly distributed random number, which is usually denoted as $F_{p,i} \sim \mathcal{U}(0,1)$.

The particle velocity distributions in space plasmas, as confirmed by spacecraft measurements [36], [20], are mostly non-Maxwellian. It is, therefore, also desirable to simulate plasma conditions where the velocity pdf is not the shifted-Maxwellian distribution, and the inverse transform sampling method may no longer be applicable. For this purpose, in the following, we propose a new efficient method to generate random velocities from virtually any pdf with any form. Our method is based on *rejection sampling* [12], which is a standard Monte Carlo technique that uses a simpler proposal distribution, $\pi_s(\mathbf{v})$, to generate samples. The generated samples are then accepted or discarded according to the ratio between the target and proposal distributions $f_s(\mathbf{v})/\pi_s(\mathbf{v})$. The easiest possible proposal distribution is obtained by using a uniform distribution as $\pi_s(\mathbf{v}) = \ell \mathcal{U}(\mathbf{v})$, where $\ell = \max\{f_s(\mathbf{v})\}$ is an upper bound for the target pdf $f_s(\mathbf{v})$. However, the main

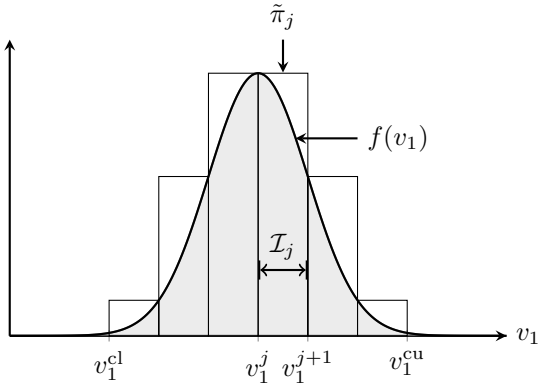

Fig. 3. Illustration of velocity subdomain $\mathcal{I}_j = \{v_1^j, v_1^{j+1}\}$ and its corresponding proposal pdf $\tilde{\pi}_j$ for $D = 1$.

issue with this chose of proposal distribution, is that many of the generated samples may be rejected. To decrease the number of rejected samples, and hence, increase the efficiency of generating random velocities, we divide the velocity domain into subdomains, each with its own proposal distribution constructed by the maximum value of the target distribution in the corresponding subdomain.

Let the target distribution $f_s(\mathbf{v})$ be a D-dimensional pdf, with the following upper and lower velocity cutoffs

$$\mathbf{v}^{\rm cu} = (v_1^{\rm cu}, v_2^{\rm cu}, \ldots, v_D^{\rm cu}),$$
$$\mathbf{v}^{\rm cl} = \left(v_1^{\rm cl}, v_2^{\rm cl}, \ldots, v_D^{\rm cl}\right).$$

To divide the velocity domain into smaller subdomains, for each component of the velocity vector, a set of support points is constructed by

$$\mathfrak{S}_i = \{v_i^1, v_i^2, \ldots, v_i^{N_i}\}, \quad \text{for} \quad i = 1, \ldots, D, \tag{8}$$

sorted in ascending order, where $v_i^1 = v_i^{\rm cl}$ and $v_i^{N_i} = v_i^{\rm cu}$ are the lower and upper cutoffs, respectively, and N_i is the number of support points for the velocity component v_i. The support points do not need to be equidistant. The Cartesian product of all the sets of support points, $\prod_{i=1}^{D} \mathfrak{S}_i$, gives $N_{\mathcal{I}} = \prod_{i=1}^{D}(N_i - 1)$ hyperrectangle intervals, which constitute the velocity subdomains. An illustration of the velocity subdomain $\mathcal{I}_j$ for $D = 1$ is given in Fig. 3.

The next step is to build the proposal pdf $\pi_s(\mathbf{v})$, which approximates $f_s(\mathbf{v})$ in each subdomain $\mathcal{I}_j$, and it is easy to sample from. As mentioned above, the simplest choice of such a proposal pdf is obtained by a sequence of piece-wise constant functions $\{\tilde{\pi}_j(\mathbf{v})\}_{j=1}^{N_{\mathcal{I}}}$, which are constructed by

$$\tilde{\pi}_j(\mathbf{v}) = \ell_j \mathcal{U}(\mathbf{v}), \tag{9}$$

where $\ell_j = \max\{f_s(\mathbf{v}) : \mathbf{v} \in \mathcal{I}_j\}$, i.e., the maximum value of $f_s(\mathbf{v})$ in $\mathcal{I}_j$, see Fig. 3.

The normalizing constant for $\pi_s(\mathbf{v})$ is given by

$$c_\pi = \sum_{j=1}^{N_{\mathcal{I}}} \tilde{\pi}_j(\mathbf{v})\, \mathrm{d}\mathbf{v}_j, \tag{10}$$

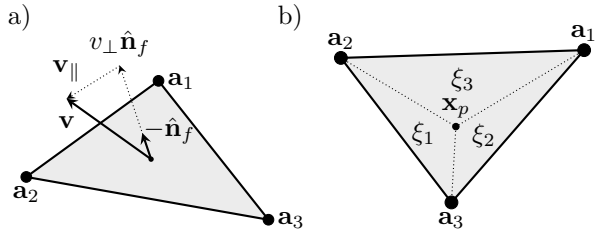

Fig. 4. Illustration of a) tangential and normal components of a velocity vector, and b) generation of a uniformly distributed random point $\mathbf{x}_p$, for a triangular boundary facet, f. $\hat{\mathbf{n}}_f$ is the outward-pointing (out of Ω) normal vector and $v_\perp = \mathbf{v} \cdot \hat{\mathbf{n}}_f$.

where $\mathrm{d}\mathbf{v}_j = \prod_{i=1}^{D}(v_i^{j+1} - v_i^j)$. Thus, the proposal pdf has the following form

$$\pi_s(\mathbf{v}) = \frac{1}{c_\pi} \begin{cases} \tilde{\pi}_1(\mathbf{v}), & \mathbf{v} \in \mathcal{I}_1, \\ \vdots & \\ \tilde{\pi}_{N_{\mathcal{I}}}(\mathbf{v}), & \mathbf{v} \in \mathcal{I}_{N_{\mathcal{I}}}. \end{cases} \tag{11}$$

The cdf of $\pi_s(\mathbf{v})$ is simply obtained by

$$\varpi_{s,k} = \frac{1}{c_\pi} \sum_{j=1}^{k} \tilde{\pi}_j(\mathbf{v}) d\mathbf{v}_j. \tag{12}$$

Once the proposal pdf and its cdf are constructed, random velocities are generated by the following three steps:

1) *Inverse transform step:* Draw a uniformly distributed random number $r_0 \sim \mathcal{U}(0, 1)$. Find index k such that
$$r_0 \leq \varpi_{s,k}. \tag{13}$$
The index k corresponds to subdomian $\mathcal{I}_k$.
2) *Sampling step:* Sample D random numbers in subdomain $\mathcal{I}_k$ by
$$v_{p,i} = v_i^k + r_i(v_i^{k+1} - v_i^k), \quad \text{for} \quad i = 1, \ldots, D, \tag{14}$$
where $r_i \sim \mathcal{U}(0, 1)$. Equation (14), gives the components of the sampled random velocity $\mathbf{v}_p$.
3) *Rejection step:* Sample r_{D+1} from $\mathcal{U}(0, 1)$. If $r_{D+1} \leq \frac{f_s(\mathbf{v}_p)}{\pi_s(\mathbf{v}_p)}$, then accept $\mathbf{v}_p$. Otherwise, reject $\mathbf{v}_p$ and go back to step 1.

With this method, owing to close tightness between $\pi_s(\mathbf{v})$ and $f_s(\mathbf{v})$, the majority of sampled velocities are accepted. The efficiency of this method is crucially dependent on the choice of proposal pdf. Here, we have presented the simplest possible way of constructing a proposal pdf that is close to the target pdf. Another possible chose, which is used for one dimensional pdfs in adaptive rejection sampling [14], would be piece-wise linear hyperplanes tangent to $f_s(\mathbf{v})$ at each support point. This would increase the acceptance rate of the sampled velocities, since the proposal pdf is even closer to the target pdf. However, it would be computationally more costly, as it would result in more function evaluations.

B. Exterior Boundary Conditions

The exterior boundaries of the simulation domain are assumed to be open, and placed far away from any object inside

the domain such that any plasma disturbances caused by the presence of the objects do not perturb the velocity distribution function of the ambient plasma at the boundaries. Since the boundaries are open, plasma particles may leave the domain, and therefore, new particles from each species must be injected into the domain in accordance with the velocity distribution function of the ambient plasma.

The injection process consists of first finding the number of particles to be injected through each exterior boundary facet, and then assigning a random position and velocity to each particle. Let $f_s(\mathbf{v})$ be the velocity distribution of the ambient plasma species s with particle number density n_s'. The flow of particles creates a flux through the facet f per time Δt and facet area ΔS_f, given by

$$\Phi_{s,f} = \Delta t \Delta S_f n_s' v_\perp, \tag{15}$$

where, with reference to Fig. 4 a), $v_\perp = \mathbf{v} \cdot \hat{\mathbf{n}}_f$. It is clear from this expression that only particles that have a negative velocity component normal to the exterior boundary facet f can enter the simulation domain. Thus, the total number of particles entering the simulation domain through the facet f with area ΔS_f, during a time-step Δt can be found by taking the expectation value of (15), i.e.,

$$N_{\Phi,s,f} = \Delta t \Delta S_f n_s' \int\limits_{\mathbb{R}^{D-1}} \int\limits_{-\infty}^{0} v_\perp f_s(\mathbf{v}) \, \mathrm{d}v_\perp \, \mathrm{d}\mathbf{v}_\parallel, \tag{16}$$

where $\mathbf{v}_\parallel$, as illustrated in Fig. 4 a), is the particle velocity tangent to the facet.

For the shifted-Maxwellian velocity distribution, (16) reduces to [22]

$$N_{\Phi_\mathrm{M},s,f} = \Delta t \Delta S_f n_s^{'} \left[\frac{v_{\mathrm{th},s}}{\sqrt{2\pi}} \exp\left(-\frac{v_{\mathrm{D},s,\perp}^2}{2v_{\mathrm{th},s}^2} \right) + \frac{v_{\mathrm{D},s,\perp}}{2} \left(1 + \mathrm{erf}\left(\frac{v_{\mathrm{D},s,\perp}}{\sqrt{2} v_{\mathrm{th},s}} \right) \right) \right], \tag{17}$$

where $v_{\mathrm{D},s,\perp}$ is the component of the drift velocity normal to the facet.

For each of the $N_{\Phi,s,f}$ particles, a uniformly distributed random position on the facet must be generated. In 2D, the exterior facet element is a straight line with two vertices, say, $\mathbf{a}_1$ and $\mathbf{a}_2$, and a uniformly distributed random position between the vertices is simply generated by

$$\mathbf{x}_p = \mathbf{a}_1 + r_p \left(\mathbf{a}_2 - \mathbf{a}_1 \right), \tag{18}$$

where $r_p \sim \mathcal{U}(0,1)$, is a uniformly distributed random number. In 3D, where the exterior facet element is a triangle with vertices, say, $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$, see Fig. 4 b), the random position is given by [28]

$$\mathbf{x}_p = \xi_1 \mathbf{a}_1 + \xi_2 \mathbf{a}_2 + \xi_3 \mathbf{a}_3, \tag{19}$$

where $\xi_1 = \sqrt{r_{p,2}} \left(1 - r_{p,1} \right)$, $\xi_2 = r_{p,1} \sqrt{r_{p,2}}$, and, $\xi_3 = \left(1 - \sqrt{r_{p,2}} \right)$ are the barycentric coordinates, with $\xi_1 + \xi_2 + \xi_3 = 1$ to ensure that $\mathbf{x}_p$ is within the triangle, and $r_{p,1}, r_{p,2} \sim \mathcal{U}(0,1)$ are two uniformly distributed random numbers.

Each generated particle must be given a random velocity based on the velocity distribution function, $f_s(\mathbf{v})$, of the ambient plasma before it is injected into the simulation domain. The probability distribution function for the particle flux through the exterior boundary facet f is given by

$$f_{\Phi,s,f}(\mathbf{v}) = \begin{cases} -v_\perp f_s(\mathbf{v}), & \text{if} \quad v_\perp < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{20}$$

where $v_\perp$ is the velocity component normal to the facet. The optimized rejection sampling method, described in the previous section, is then utilized to generate random velocities.

Once the random position $\mathbf{x}_p$ and velocity $\mathbf{v}_p$ for each particle are generated, the particles must be injected through the exterior facet and placed inside the simulation domain. There exist different injection methods with different order of accuracy [6]. In the simplest (first-order accurate) injection method, which ignores any forces acting on the particles, the generated particles are placed inside the domain at $\mathbf{x}_p + r_p \Delta t \mathbf{v}_p$, where $r_p \sim \mathcal{U}(0,1)$, to make sure the injection is as continuous as possible by taking into account that particles may have entered the domain during the last time-step with a uniform probability in the range $[\mathbf{x}_p - \Delta t \mathbf{v}_p, \mathbf{x}_p]$. This method is a good approximation if the plasma disturbances due to the presence of objects inside the domain are negligible at the exterior boundaries. In the presence of a uniform background magnetic flux density, a more accurate (second-order) particle injection can be obtained by using a Boris push [6]. For general spatially and temporally varying electric field and magnetic flux density, a third-order injection method has also been suggested in [6].

C. Interior Boundary Conditions

Objects in the plasma are represented on the mesh as a set of interior boundaries $\{\Gamma_\alpha\}$ where $\alpha \in \mathcal{S} = 1, 2, ...$ is the indexing of the objects. See e.g. Fig. 1. We shall limit the discussion to objects that are perfect electric conductors, but shall develop a generic theory for simulating circuits of such objects connected by ideal voltage and current sources. This allows simulations where for instance a probe has a certain bias voltage with respect to the spacecraft (e.g. Langmuir probes), or where a certain current is known to be imposed on an object by a device external to the simulation. An example circuit is shown in Fig. 5. Implementation-wise, this circuitry is specified in PUNC by a list of voltage sources, and a list of current sources, e.g.:

```
vsources = [[0, 1, V1 ],
            [1, 3, V13],
            [4, 6, V46]]
isources = [[1, 2, I12]]
```

Each source consists of two numbers specifying which objects it is connected to, and a third number specifying the voltage/current (which can be made to vary from time-step to time-step, if desirable). A zero indicates that the source is connected to system ground.

The method that is to be developed require certain information about the charge of the objects. We shall now derive

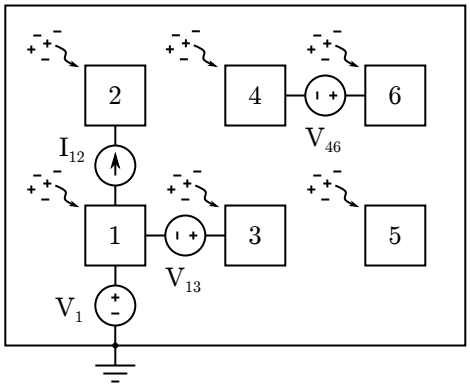

Fig. 5. Example of a simulation of three circuits consisting of six objects (square). The two rightmost circuits are floating. Objects 2, 4, 5 and 6 have unknown potentials, whereas objects 1 and 3 have known potentials, the potential of object 3 obviously being $V_1 + V_{13}$. The charges in the objects are collected from the surrounding plasma, as well as being delivered through the sources. Note: For drifting plasma with background magnetic flux density the outer boundary is not zero as indicated here, but still Dirichlet.

the required charges from particle quantities, while postponing the field solver theory, which applies also to cases where the charge may be known for other reasons.

Since the net current into an object α is defined as $I_\alpha = \mathrm{d}Q_\alpha/\mathrm{d}t$ the charge of an object can be numerically integrated to accuracy $\mathcal{O}(\Delta t^2)$ through

$$Q_\alpha^{n+1} = Q_\alpha^n + \Delta t I_\alpha^{n+\frac{1}{2}}, \tag{21}$$

where Q_α^0 is an initial condition for the object. $I_\alpha^{n+\frac{1}{2}}$ is the sum of all currents into the object. This includes:

1) Current collected from the plasma. Particles colliding with an object, as detected by it moving through a facet of the object, is absorbed and deleted. The particles collected by object α between timesteps n and $n+1$ adds a charge $\Delta Q_{c,\alpha}^{n+\frac{1}{2}}$ to the object equal to that of all collected particles. Hence the collected current for any object is $I_{c,\alpha}^{n+\frac{1}{2}} = \Delta Q_{c,\alpha}^{n+\frac{1}{2}}/\Delta t$.
2) Current due to current sources. Current sources have a prescribed current which is readily added to the collected current. For instance, object 2 in Fig. 5 has a total current consisting of both the collected current and that due to the current source: $I_2^{n+\frac{1}{2}} = I_{c,2}^{n+\frac{1}{2}} + I_{12}^{n+\frac{1}{2}}$.
3) Current due to voltage sources. Voltage sources have a prescribed voltage across them, and whichever current is necessary to maintain that voltage will flow through it instantaneously [27, p. 28]. Since this current is unknown, (21) cannot be evaluated to obtain Q_α^{n+1} for objects connected to voltage sources. In the example in Fig. 5 this would be objects 1, 3, 4 and 6.

We observe that although for instance Q_4^{n+1} in Fig. 5 cannot be determined from (21) due to the voltage source, the sum $Q_4^{n+1} + Q_6^{n+1}$ can, because the contribution from the voltage source cancel. The charge leaving object 4 through the voltage source enters object 6 (or vice versa). We will refer to this by saying that objects 4 and 6 *share the charge*. This identification of which objects share charge is fundamental for solving the field equations when voltage sources are present, and will be further explored in Sec. IV.

To simplify the implementation, the charge is integrated from time-step n to $n+1$ for each individual object α as if there were no voltage sources. The resulting "false charge" is called $\tilde{Q}_\alpha^{n+1}$. While this may not be the actual charge for individual objects, summing over charge sharing objects yields the correct net charge.

Since the rest of the paper discusses methods taking place at a single time-step $n+1$ we shall omit the superscript from now on.

IV. Mesh Methods

When solving Maxwell's equations for electrostatic problems it is commonplace to express the electric field $\mathbf{E}$ in terms of an electric potential ϕ [7, p. 92]:

$$\mathbf{E} = -\nabla\phi. \tag{22}$$

This makes the curl equation in (2) trivially satisfied, and the divergence equation reduces to the Poisson equation:

$$-\nabla^2\phi = \frac{\rho}{\varepsilon_0}. \tag{23}$$

Next, one must consider the boundary conditions for which to solve this partial differential equation. We first consider the exterior boundary condition, and subsequently treat two cases of interior boundary conditions; first, a single object at its floating potential, and second, the general case of arbitrarily many objects. We then proceed by discretizing the problem.

A. Exterior Boundary Conditions

For the general case of a Maxwellian plasma with a homogeneous background magnetic flux density $\mathbf{B}_0$ and a constant drift velocity $\mathbf{v}_D$ there must be a homogeneous background electric field $\mathbf{E}_0$ consistent with the $\mathbf{E}\times\mathbf{B}$ drift velocity:

$$\mathbf{v}_D = \frac{\mathbf{E}_0 \times \mathbf{B}_0}{\|\mathbf{B}_0\|^2}. \tag{24}$$

This implies that the component of $\mathbf{E}_0$ which is perpendicular to $\mathbf{B}_0$ must equal $-\mathbf{v}_D \times \mathbf{B}_0$. Any other components of $\mathbf{E}_0$ must be zero; otherwise the particles would be accelerated and not maintain the (homogeneous) drift velocity. Assuming the exterior boundary Γ_e (see Fig. 1) to be sufficiently far away from any perturbations in the fields, the potential at Γ_e is therefore given by the following Dirichlet boundary condition:

$$\phi = \phi_0 \stackrel{\text{def}}{=} (\mathbf{v}_D \times \mathbf{B}_0)\cdot\mathbf{x} \text{ on } \Gamma_e, \tag{25}$$

where $\mathbf{x}$ is the position on Γ_e and the integration constant is arbitrarily set to zero (the forces only depends on the gradient of ϕ anyway).

B. Interior Boundary Conditions – Single Object

For a perfect electric conductor, electromagnetic theory provides us boundary conditions for the normal and tangential

components of the electric field [7]. These can be written in terms of the potential as:

$$\nabla\phi \cdot \hat{\mathbf{n}} = \frac{\sigma}{\varepsilon_0}, \tag{26}$$

$$\nabla\phi \times \hat{\mathbf{n}} = \mathbf{0}, \tag{27}$$

where σ is the surface charge density. Note that $\hat{\mathbf{n}}$ by our definition points *into* the object.

Condition (27) implies that the objects' surface potential is constant:

$$\phi = V_\alpha \text{ on } \Gamma_\alpha. \tag{28}$$

However, for an object with floating potential, this constant is unknown. To resolve this unknown, one extra constraint is required. This comes from (26). While σ is unknown too (it will redistribute self-consistently), integrating (26) over the object's surface reveals the Gauss equation in integral form,

$$\varepsilon_0 \oint_{\Gamma_\alpha} \nabla\phi \cdot \hat{\mathbf{n}} \, \mathrm{d}s = Q_\alpha. \tag{29}$$

Recall that all the charge on a perfect electric conductor resides on the surface, and that Q_α is obtained by "counting" the collected particles as described in Sec. III-C. This constraint therefore closes the system of equations.

C. Interior Boundary Conditions – Multiple Objects

For multiple objects, each object α is equipotential according to (28). This means that we need as many constraints as there are objects to resolve the V_α's. To have multiple objects with independently floating potentials is simply to add one constraint like (29) for each of them. Connecting them by current sources also poses no problems, since this merely alters the value of the charge Q_α (c.f. Sec. III-C). What complicates the matter is the voltage sources, which may connect objects to each other, and objects to system ground (i.e. fix the potential of an object). Objects connected to voltage sources do not have a known charge, and therefore (29) cannot readily be imposed. To deal with this, we shall first identify two classes of objects:

1) Objects with fixed potential. An object α connected to ground through a voltage source has a fixed potential V_α. Objects also have fixed potentials if they are connected (through a voltage source) to another object with fixed potential. In the example in Fig. 5, objects 1 and 3 fall into this category; V_1 is specified from the outset, and $V_3 = V_1 + V_{13}$ from basic circuit theory. We define $\mathcal{S}^K$ to be the set of (indices of) objects whose potential is fixed.
2) Objects with non-fixed potential. All other objects have non-fixed potential, i.e. the objects in $\mathcal{S}^U = \mathcal{S} \setminus \mathcal{S}^K$.

In order to formulate constraints regarding the charge of objects with non-fixed potentials, the set $\mathcal{S}^U$ is partitioned into N_z disjoint (non-overlapping) sets of charge sharing objects:

$$\mathcal{S}^U = \mathcal{S}_1^U \cup \mathcal{S}_2^U \cup \cdots \cup \mathcal{S}_{N_z}^U \tag{30}$$

To state rigorously what we mean by "charge sharing" sets; an object in $\mathcal{S}_z^U$ (for some z) cannot be connected, through a voltage source, to an object in $\mathcal{S}_y^U$ where $y \neq z$. Moreover, the sets should be as small as the previous statement permits.

To consider a concrete example, for Fig. 5, the partition would be:

$$\begin{aligned} \mathcal{S}_1^U &= \{2\}, \\ \mathcal{S}_2^U &= \{4, 6\}, \\ \mathcal{S}_3^U &= \{5\}, \end{aligned} \tag{31}$$

In PUNC this is represented similarly;

```
groups = [ [2], [4,6], [5] ]
```

and it is automatically derived from `vsources` by following edges in a graph, the nodes being the objects and the edges the voltage sources. Subsets containing zero (ground) are removed from the list.

Recalling from Sec. III-C that the total charge of charge sharing objects is known, one can take the sum of (29) to obtain N_z constraints with known right hand sides[2]:

$$\varepsilon_0 \sum_{\alpha \in \mathcal{S}_z^U} \oint_{\Gamma_\alpha} \nabla\phi \cdot \hat{\mathbf{n}} \, \mathrm{d}s = \sum_{\alpha \in \mathcal{S}_z^U} Q_\alpha, \tag{32}$$

for all z.

The remaining constraints are the voltage sources. For a voltage source w imposing a voltage difference $V_{\alpha_w \beta_w}$ between two objects α_w and β_w the constraint is;

$$V_{\beta_w} - V_{\alpha_w} = V_{\alpha_w \beta_w}, \tag{33}$$

which applies for all w. Recall that V_α is really just $\phi(\mathbf{x}_i)$ where $\mathbf{x}_i$ can be any point on Γ_α, so this is indeed a constraint of ϕ.

To summarize, there are equally many constraints of form (32) or (33) as there are objects, and these provides closure for the unknowns V_α. For simplicity, we shall write constraints generically as

$$g_z(\phi) = 0 \tag{34}$$

where for $z = 1, ..., N_z$ this is the charge constraints (32), and for $z = N_z + 1, ..., |\mathcal{S}|$ it is the potential constraints (33). $|\cdot|$ denotes the number of elements in the set.

Once the solution ϕ is obtained, the correct charge Q_α can be computed on all objects by numerically evaluating (29). Currents through voltage sources can further be inferred from circuit theory, should they be of interest.

D. Discretization

To solve this boundary value problem using the FEM method we shall first rewrite the Poisson equation to variational form and discretize it. Subsequently, we shall close the system of equations with the boundary conditions.

To rewrite the Poisson equation, we multiply it by a *test function* ψ, integrate across the simulation domain Ω, and use integration by parts:

$$\varepsilon_0 \int_\Omega \nabla\phi \cdot \nabla\psi \, \mathrm{d}\mathbf{x} - \varepsilon_0 \int_{\partial\Omega} \psi \nabla\phi \cdot \hat{\mathbf{n}} \, \mathrm{d}s = \int_\Omega \rho\psi \, \mathrm{d}\mathbf{x}. \tag{35}$$

[2]This is in fact an extension to how the surface charge density σ is unknown in Sec. IV-B, but the surface integral of it equals the known total charge Q_α.

Here, $\mathrm{d}\mathbf{x}$ is a volume element, $\partial\Omega$ is the boundary of Ω, $\mathrm{d}s$ is a surface element on that boundary and $\hat{\mathbf{n}}$ is a normal unit vector pointing out of Ω (into objects). Although our boundary conditions are somewhat different than in the usual text book Dirichlet problem, we shall still enforce them separately, and hence take ψ to be zero on $\partial\Omega$. In that case, the second term vanishes and we're left with the variational form:

$$a(\phi, \psi) = \langle \rho, \psi \rangle, \tag{36}$$

where a is the symmetric, coercive and continuous bilinear form given by

$$a(\phi, \psi) = \varepsilon_0 \int_\Omega \nabla\phi \cdot \nabla\psi \, \mathrm{d}\mathbf{x} \tag{37}$$

and $\langle \cdot, \cdot \rangle$ is the standard L_2 inner product:

$$\langle \rho, \psi \rangle = \int_\Omega \rho\psi \, \mathrm{d}\mathbf{x}. \tag{38}$$

The inner product requires $\rho \in L_2(\Omega)$ (a physically sound assumption), in which case the exact solution of (23) is in the Sobolev space $H^1(\Omega)$ provided the boundary is sufficiently smooth [17, p. 92]. More specifically, it must satisfy the boundary conditions such that it will be in

$$\begin{aligned} H^1_V(\Omega) = \{\phi \in H^1(\Omega) : \phi &= \phi_0 \text{ on } \Gamma_e, \\ \phi &= \text{const on } \Gamma_\alpha, \quad \forall \alpha \in \mathcal{S}, \\ g_z(\phi) &= 0, \quad z = 1, ..., |\mathcal{S}|\}, \end{aligned} \tag{39}$$

Thence, the problem of finding $\phi \in H^1_V(\Omega)$ such that

$$a(\phi, \psi) = \langle \rho, \psi \rangle, \quad \forall \psi \in H^1_0(\Omega) \tag{40}$$

is equivalent of solving the boundary value problem. $H^1_0(\Omega)$ is defined as usual:

$$H^1_0(\Omega) = \{\phi \in H^1(\Omega) : \phi = 0 \text{ on } \partial\Omega\}, \tag{41}$$

For discretizing the domain, any curved boundaries are approximated by piecewise linear segments and the domain is partitioned into a mesh of disjoint, conformal[3], simplicial[4] cells. This is done externally to PUNC, e.g. using Gmsh [13]. Moreover, we discretize the function space $H^1(\Omega)$ into a finite dimensional space with basis functions $\{\psi_j(\mathbf{x})\}_{j\in\mathcal{I}}$ where $\mathcal{I} = \{1, 2, ...\}$ is the set of indices of the degrees-of-freedom. The approximate solution can then be written

$$\phi(\mathbf{x}) = \sum_{j\in\mathcal{I}} \phi_j \psi_j(\mathbf{x}), \tag{42}$$

where ϕ_j are the coefficients of $\phi(\mathbf{x})$ in the given basis.

The approximation in the FEM method is in replacing the function spaces. This means replacing ϕ with (42) in (40), and requiring it to be true for the basis functions of the discretized test function space (and hence all other functions in the test function space). Using linearity of a we write the matrix equation

$$\sum_{j\in\mathcal{I}} A_{ij}\phi_j = b_i, \quad i \in \mathcal{I} \tag{43}$$

[3]No vertex of one cell appear at the middle of the edge or face of another cell.

[4]Having as few vertices as necessary, e.g. triangles for 2D and tetrahedrons for 3D.

where

$$A_{ij} = a(\psi_j, \psi_i), \quad b_i = \langle \rho, \psi_i \rangle, \quad i, j \in \mathcal{I}. \tag{44}$$

It is important that the variational form (40) should hold for all test functions *that are zero on the boundary*. However, the assembled linear system (43) now includes test functions that are not zero on the boundary. Accordingly, the rows corresponding to basis functions that are non-zero on the boundary must be replaced by equations strongly imposing the boundary conditions. We denote by $\mathcal{I}_e$ (the index of) the basis functions being non-zero on Γ_e and $\mathcal{I}_\alpha$ the basis functions being non-zero on an object boundary Γ_α. The replacement of rows i where i is in one of these sets will be treated in subsequent sections.

For the electric potential, we use the Lagrange finite element space CG_r of continuous, piecewise polynomial functions of order r. The nodes $\mathbf{x}_i$ and basis functions ψ_j of this space is such that

$$\psi_j(\mathbf{x}_i) = \delta_{ij}, \tag{45}$$

and varying polynomially of order r within each cell. Fig. 6 a) shows an example of CG_1, and δ_{ij} is the Kronecker delta. The fact that the basis functions has local support (are non-zero only on the cells surrounding its associated node) means that the matrix A is sparse, which is beneficial for many linear algebra solvers. Moreover, clever ordering of the nodes in the mesh ensures that the bandwidth of the matrix is small, which is also beneficial. This is taken care of by FEniCS, and not considered further in this paper [34], [19].

E. Exterior Boundary Condition

The exterior Dirichlet boundary condition is imposed in the standard way [17]. From (45) and (42) we have that for a node $\mathbf{x}_i$, $\phi(\mathbf{x}_i) = \phi_i$ and since $\phi = \phi_0$ is known on Γ_e we set $\phi_i = \phi_0(\mathbf{x}_i)$. In the system of equations this means:

$$A_{ij} = \delta_{ij}, \quad b_i = \phi_0(\mathbf{x}_i), \quad \forall (i, j) : \mathcal{I}_e \times \mathcal{I}. \tag{46}$$

F. Interior Boundary Conditions – Equipotentiality

For all interior boundaries Γ_α equipotentiality means that ϕ_i are equal for all nodes on Γ_α. If, for the sake of illustration, we label the nodes on Γ_α as $1, 2, \ldots, N_\alpha$, we could say that

$$\begin{aligned} \phi_1 &= \phi_2 \\ \phi_2 &= \phi_3 \\ &\vdots \\ \phi_{N_\alpha - 1} &= \phi_{N_\alpha} \end{aligned} \tag{47}$$

which is $N_\alpha - 1$ equations. As is to be expected, one degree-of-freedom is left, which will be closed by the charge and potential constraints. Imposing (47) naïvely as rows in the matrix has certain issues: ϕ_1 may be far from ϕ_2, which may be far from ϕ_3, and so forth. This will create non-zero elements far from the diagonal of the matrix, and possibly inhibit the performance of the linear algebra solver. One could imagine trying to set ϕ_1 equal to one of its neighboring nodes instead of ϕ_2, and that node equal to one of *its* neighbors, and so

forth. On a 2D surface it is not obvious how to cover all nodes in such a "propagating" pattern. One should likely resort to graph theory. Instead, we propose another method, namely setting ϕ_1 equal to the average of all its neighbors on the surface, and so forth for all the other boundary nodes except one (i.e. $N_\alpha - 1$ equations). Leaving one is important since otherwise, they would be linearly dependent, as if we had included $\phi_{N_\alpha} = \phi_1$ in (47).

Let's state this method rigorously. Let nodes i and j be neighbors if they are nodes on a common cell. Let $\mathcal{N}_j$ be the set of all neighbors of node j also on Γ_α. Then we'd like to set

$$\phi_j = \frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \phi_i \tag{48}$$

for all ϕ_j on Γ_α but one. In the matrix this means,

$$A_{ij} = \begin{cases} |\mathcal{N}_j|, & i = j \\ -1, & i \in \mathcal{N}_j \\ 0, & \text{otherwise} \end{cases}, \quad b_i = 0, \tag{49}$$

for all $j \in \mathcal{I}$, for all $i \in \mathcal{I}_\alpha$ but one. Note that these rows has similar sparsity properties as (44). Moreover, the method is agnostic of dimensionality and the order r of the CG space.

G. Interior Boundary Conditions – Constraints

At last there's one more row to specify per potential and charge constraint. Substituting (42) into (32) for all z readily yields the charge constraints. The potential constraints (33) is also readily enforced by just choosing a node on the repsective boundaries as V_{β_w} or V_{α_w}. For voltage sources connected to ground, one of them is omitted, and it boils down to standard Dirichlet conditions.

These rows unfortunately do not preserve the sparsity pattern common in a Dirichlet problem. Nevertheless, we've had success in solving these using *Generalized Minimal Residual* (GMRES) as linear algebra solver preconditioned with *Algebraic Multigrid* (AMG).

Since the stiffness matrix A remains unaltered between timesteps its assembly and the computation of the preconditioning matrix is done only before the first time-step. Moreover, since the potential is assumed to change only by a small amount during Δt the solution from the previous time-step is used as an initial guess to speed up computations.

H. Electric Field

Once ϕ is determined, (22) can be used to obtain the electric field $\mathbf{E} = -\nabla\phi$. We note that with $\phi \in H^1(\Omega)$, the electric field will naturally be found in $[L_2(\Omega)]^D$. There are different methods to compute the electric field. In the following, five different methods, which are available in PUNC, are described:

1) The electric potential is approximated using continuous linear Lagrange elements of order one, i.e., $\phi \in \mathrm{CG}_1(\Omega)$. In this case the gradient is computed naturally using discontinuous Lagrange elements of order zero, and we use $\mathbf{E} \in [\mathrm{DG}_0(\Omega)]^D$. The gradient is found by L_2-projection: find $\mathbf{E} \in [\mathrm{DG}_0(\Omega)]^D$, such that

$$\int_\Omega \mathbf{E} \cdot \Psi \mathrm{d}\mathbf{x} = \int_\Omega -\nabla\phi \cdot \Psi \mathrm{d}\mathbf{x}, \quad \forall\, \Psi \in [\mathrm{DG}_0(\Omega)]^D. \tag{50}$$

2) In order to have a continuous electric field in the entire domain, we choose instead $\mathbf{E} \in [\mathrm{CG}_1(\Omega)]^D$. The gradient is again found by L_2-projection: find $\mathbf{E} \in [\mathrm{CG}_1(\Omega)]^D$, such that

$$\int_\Omega \mathbf{E} \cdot \Psi \mathrm{d}\mathbf{x} = \int_\Omega -\nabla\phi \cdot \Psi \mathrm{d}\mathbf{x}, \quad \forall\, \Psi \in [\mathrm{CG}_1(\Omega)]^D. \tag{51}$$

3) We choose again $\mathbf{E} \in [\mathrm{CG}_1(\Omega)]^D$ and $\phi \in \mathrm{CG}_1(\Omega)$. However, we now divert from the projection method and use interpolation instead. For each mesh node $\mathbf{x}_j$, and the set of all cells sharing the vertex $\mathbf{x}_j$, i.e., $\mathcal{M}_j = \bigcup \{T \in \Omega; \mathbf{x}_j \in T\}$, calculate the arithmetic mean (AM) of the constant electric fields in each cell of $\mathcal{M}_j$ [23]

$$\mathbf{E}(\mathbf{x}_j) = \frac{1}{|\mathcal{M}_j|} \sum_{T_k \in \mathcal{M}_j} (-\nabla\phi)_k. \tag{52}$$

Here the right hand side $-\nabla\phi \in [\mathrm{DG}_0(\Omega)]^D$ is the piecewise constant electric field, and $|\mathcal{M}_j|$ is the number of cells in $\mathcal{M}_j$.

4) Another method which is commonly used for interpolating non-smooth and discontinuous functions with continuous and piece-wise linear elements is *Clément-interpolation* (CI) [8]. Like previous case, the interpolation is done locally for each patch $\mathcal{M}_j$ in the domain. And we still want to find $\mathbf{E} \in [\mathrm{CG}_1(\Omega)]^D$. However, to find the degrees of freedom in this space, i.e., $\mathbf{E}(\mathbf{x}_j)$ for all vertices j in the mesh, we now compute local L_2-projections of the electric field to the local macroelements composed of $\mathcal{M}_j$. Mathematically, for each macroelement $\mathcal{M}_j$ we have exactly D degrees of freedom and find $\mathbf{E}(\mathbf{x}_j) \in [\mathrm{DG}_0(\mathcal{M}_j)]^D$, such that

$$\int_{\mathcal{M}_j} \mathbf{E} \cdot \Psi \mathrm{d}\mathbf{x} = \int_{\mathcal{M}_j} -\nabla\phi \cdot \Psi \mathrm{d}\mathbf{x}, \quad \forall\, \Psi \in [\mathrm{DG}_0(\mathcal{M}_j)]^D. \tag{53}$$

As such, the Clément-interpolation corresponds to a weighted average

$$\mathbf{E}(\mathbf{x}_j) = \frac{1}{\mathrm{Vol}(\mathcal{M}_j)} \sum_{T_k \in \mathcal{M}_j} (-\nabla\phi)_k \mathrm{Vol}(T_k). \tag{54}$$

5) Finally, in order to obtain higher accuracy when computing the electric field, we solve (23) using second order continuous Lagrange elements, i.e., $\phi \in \mathrm{CG}_2(\Omega)$. The electric field can then be obtained by projecting $-\nabla\phi$ onto $[\mathrm{CG}_1(\Omega)]^D$, exactly like in (51), only now with ϕ from a higher order space.

V. Particle-Mesh Interaction

Before solving Poisson's equation, the volume charge density of each species, appearing on the right-hand side of (23), must be determined from the position of particles relative to nearest mesh nodes. This process is commonly termed

weighting, and it is done by interpolating the charge of each particle inside a given cell element T_k to the nodes of T_k. The charge at each mesh node is then divided by the corresponding volume of Voronoi cell [10] to obtain the volume charge density. Once the volume charge density is found and the electric field is computed from the solution of Poisson's equation, the particles must be accelerated. For this purpose, the Lorentz force, known at each mesh node, must be weighted to the position of each particle.

A. Force weighting

In finding the Lorentz force at the position $\mathbf{x}_p$ of a given particle, the electric field $\mathbf{E}$ needs to be interpolated using the finite element representation. We have for each vector component E_k;

$$E_k(\mathbf{x}) = \sum_{j \in \mathcal{I}} E_k(\mathbf{x}_j)\psi_j(\mathbf{x}), \quad \forall k \in 1, \ldots, D. \tag{55}$$

For a simulation particle at position $\mathbf{x}_p$, the electric field is obtained with regular finite element interpolation

$$E_k(\mathbf{x}_p) = \sum_{j \in \mathcal{I}} E_k(\mathbf{x}_j)\psi_j(\mathbf{x}_p), \quad \forall k \in 1, \ldots, D. \tag{56}$$

Note that this expression is valid for any basis ψ, like CG_1, DG_0 or CG_2. Illustrations of the CG_1 basis function ψ_j associated with the vertex $\mathbf{x}_j$, and its evaluation at the particle position $\mathbf{x}_p$ inside the cell element T_i, for the 2-dimensional case, are given in Fig. 6 a) and b), respectively.

B. Charge density assignment

The volume charge density at each mesh vertex is computed in the following two steps: first, for each mesh vertex $\mathbf{x}_j$, the charge of each particle inside $\mathcal{M}_j$, i.e., the set of all cells sharing vertex $\mathbf{x}_j$, are interpolated to $\mathbf{x}_j$ by employing the same basis functions used in (55). Second, the interpolated charge at each mesh vertex $\mathbf{x}_j$ is then divided by the corresponding volume of the Voronoi cell R_j defined at $\mathbf{x}_j$, see Fig. 6 c). The volume charge density at $\mathbf{x}_j$ is then given by

$$\rho_j = \frac{1}{\mathrm{Vol}(R_j)} \sum_p q_p \psi_j(\mathbf{x}_p), \tag{57}$$

where q_p and $\mathbf{x}_p$ are the particle charge and position, respectively. The volume of the Voronoi cell defined at vertex $\mathbf{x}_j$ is obtained by using the following approximation [22]

$$\mathrm{Vol}(R_j) \approx \frac{1}{4} \sum_{T_k \in \mathcal{M}_j} \mathrm{Vol}(T_k), \tag{58}$$

which is a good approximation if the cell elements of the Delaunay mesh are nearly equatorial [10].

We note that among the Lagrangian family of elements, only first order Lagrangian shape functions are appropriate for weighting of particle charges to the element nodes. This is simply due to the fact that higher order Lagrangian shape functions might be negative in some regions of the cell elements. To obtain higher order weighting, one may utilize Bernstein basis functions, which have the desired property of being non-negative and form a partition of unity [4].

VI. Validation

Two cases are examined for validating the methods. First, the various methods for electric field computations are compared. And second, a full PIC simulation determining the floating potential of a sphere is presented.

A. E-field Comparison

To validate the order of accuracy of the electric field obtained by the methods introduced in Sec. IV-H, we will in the following consider the simple case of a closed system of two electrically conducting concentric spheres, where the analytical solutions are easy to obtain. The inner sphere has radius $a = 0.1\,\mathrm{m}$, and the outer sphere, which is hollow, has radius $b = 1.0\,\mathrm{m}$. The space between the spheres is assumed to be empty, and the potential on the spheres is set to $V_a = 1\,\mathrm{V}$ and $V_b = 0\,\mathrm{V}$, respectively. The analytical solution of the electric field is given by [15]

$$\mathbf{E}_e = \frac{(V_a - V_b)ab\mathbf{r}}{(b-a)r^3}, \tag{59}$$

where $r = \|\mathbf{r}\|$ is the radial distance measured from origin. In order to compare the convergence of the different methods, the problem is solved on a sequence of five meshes with increasing number of cells. The error field used here is defined as $\mathbf{e} = \mathbf{E}_e - \mathbf{E}_h$, where $\mathbf{E}_e$ and $\mathbf{E}_h$ are the exact and numerical solution of the electric field, respectively. The error is measured by using L_2 norm, which is defined by [32]

$$\|\mathbf{e}\|_{L_2} = \left(\int_\Omega \mathbf{e} \cdot \mathbf{e}\, \mathrm{d}\mathbf{x} \right)^{\frac{1}{2}}. \tag{60}$$

The L_2 error norm versus minimum edge-length, h, for the methods outlined above, are displayed in Fig. 7. As expected, the largest errors correspond to the electric field in the discontinuous finite element space DG_0, where the convergence rate is around 0.9. Although the error in the electric field is large compared to the other methods, this method requires least computational time among the methods presented in this paper. The continuous electric field obtained from the electric potential based on 1st order Lagrange elements, i.e, method 2), gives smaller errors compared to DG_0, and the error is remarkably close to $\mathcal{O}(h)$. Relatively similar results are obtained for the Clément-interpolation and the arithmetic mean method. When it comes to computational efficiency, both CI and AM methods are rather similar, and twice as efficient as method 2).

As can be seen from Fig. 7, the smallest errors are achieved with method 5), i.e., for the electric field computed from the electric potential based on 2nd order Lagrange elements. In this case, the error is relatively close to $\mathcal{O}(h^2)$. Based on our simulations, this method is more efficient than method 2). However, the Poisson solver used to obtain the electric potential is almost four times less efficient than the corresponding solver in method 2). Needless to say, the most efficient method for any particular problem should be the one which gives the desired accuracy with the least amount of computational work.

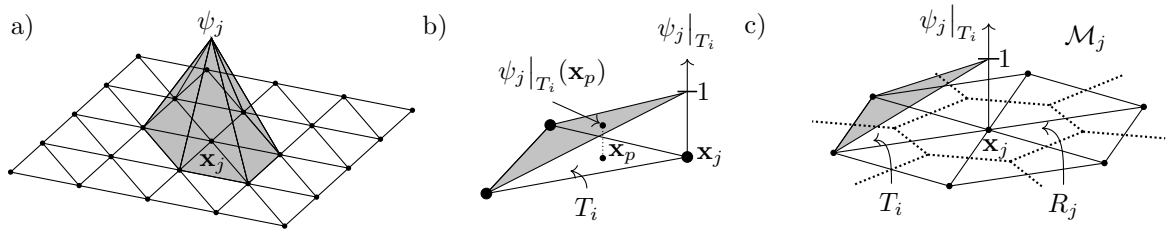

Fig. 6. Illustration of a) a 2D linear basis function ψ_j associated with the mesh vertex $\mathbf{x}_j$ on top of the triangular mesh, b) a triangular cell element T_i with the corresponding basis function associated with $\mathbf{x}_j$ restricted to T_i (the basis function is equal to one at the vertex $\mathbf{x}_j$, and linearly decreases to zero in the element T_i), c) the set $\mathcal{M}_j$, containing all cells sharing vertex $\mathbf{x}_j$, with the corresponding Voronoi cell R_j.

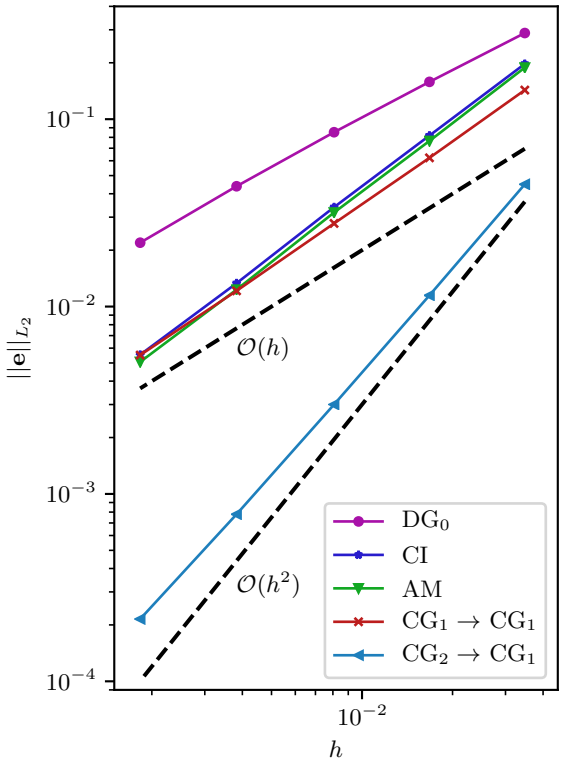

Fig. 7. The L_2 error norm versus minimum edge-length h, of the electric field computed by the methods outlined in section Sec. IV-H. The mesh was generated by Gmsh (version 3.0.5) with five different resolutions on the outer sphere, $\{1/5, 1/10, 1/20, 1/40, 1/80\}$, and the inner sphere, $\{1/30, 1/60, 1/120, 1/240, 1/480\}$, resulting in the following number of tetrahedral cells: $\{7\,463, 48\,852, 340\,714, 2\,708\,950, 20\,877\,296\}$.

B. Spherical Probe

Laframboise gives the relationship between applied voltage w.r.t. the background plasma and collected current for spherical probes with a significant radius compared to the Debye length λ_{De}. In this test we consider a spherical probe of radius $r_i = \lambda_{De}$. We choose the outer boundary to $r_o = 10\lambda_{De}$ to make sure the outer boundary is sufficiently far away from any perturbations in the plasma, since this is an assumption of the boundary condition. The mesh (generated using Gmsh) consist of 9061 tetraheddral cells, and has a resolution of $0.2\lambda_{De}$ at the probe and $2\lambda_{De}$ at the outer boundary. Moreover, we use CG_1 for both ϕ and $\mathbf{E}$, using the L_2-projection for $\mathbf{E}$. The simulation is initialized with 4 simulation particles per cell of each species (electrons and protons), and the time step is taken

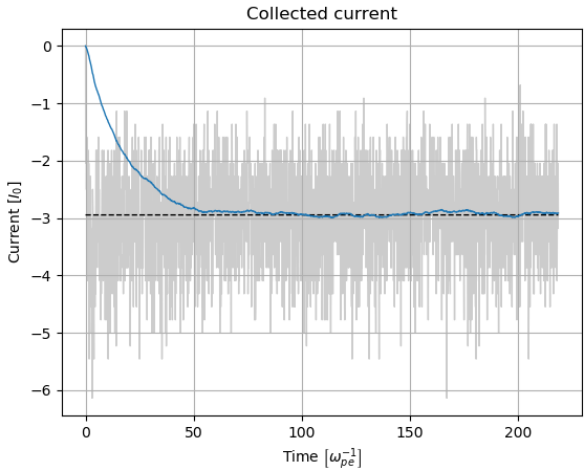

Fig. 8. Current collected by object connected to voltage source.

to be $0.1\omega_{pe}^{-1}$ where ω_{pe} is the electron (angular) frequency.

With this setup, we demonstrate good agreement with the results of Laframboise. We do this in two different ways: First, we fix the potential of the probe to $2e/k_BT_e$ using a voltage source. Here, e, k_B and T_e are the elementary charge, the Boltzmann constant and the electron temperature, respectively. According to Laframboise this should yield a collected current of $I = -2.945I_0$ in steady-state where $I_0 = en_e\sqrt{8\pi k_BT_e/m_e}$. Fig. 8 shows the result of this simulation (gray) along with the current predicted by Laframboise (dashed black). As mentioned in Sec. II-A the reduced number of particles in simulations enhances the noise, which is clearly the case in this simulation. Nonetheless, by employing an *exponential moving average* filter with relaxation time of $20\omega_{pe}^{-1}$ (blue) we are able to see that the result is indeed in good agreement with Laframboise.

Second, instad of fixing the potential, the potential is left floating and a current of $I = -2.945I_0$ is pulled from the probe using a current source. At steady state, one expects the probe to collect the same current from the surrounding plasma (otherwise charge would build up). By observing the collected current in Fig. 9, it is evident that this is the case. Then, according to Laframboise, one expects the potential to settle at $2e/k_BT_e$. Although somewhat noisy, Fig. 10 do show this behaviour of the object potential (more particles would reduce the noise).

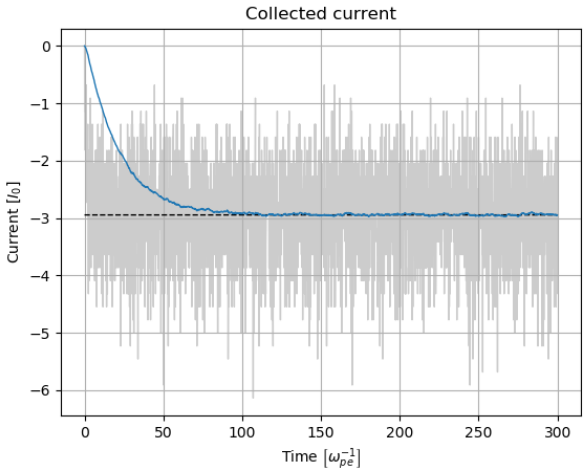

Fig. 9. Current collected by object connected to current source.

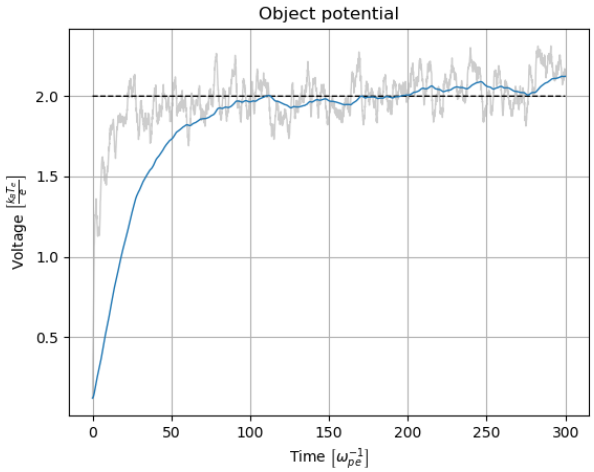

Fig. 10. Potential of object connected to current source.

VII. Discussion

That the simulation of a spherical probe with specified voltage shows results comparable to those of Laframboise indicates that the full FEM based PIC method works as intended. Moreover, similar results were also achieved by imposing a current, letting the object floating potential be self-consistently determined. This demonstrates that imposing the Gauss constraints as rows in the matrix also works as intended.

In the following, we would like to discuss order of accuracy. Many PIC codes, e.g. those presented in [5], [16], use numerical integrators that are second order accurate (have error $\mathcal{O}(\Delta t^2)$) and use Poisson solvers that are also second order accurate. However, in the PIC cycle one also takes the gradient of ϕ to obtain $\mathbf{E}$. The gradient effectively divides the spatial error by the step-size (this is true for finite element as well as finite difference based methods), such that the degree of accuracy of the whole PIC cycle is one less than that of the Poisson solver alone. This can be observed in Fig. 7. To achieve a PIC code that is second order accurate in both space and time it is therefore necessary to use a third order accurate Poisson solver. The method of this paper is agnostic of order, and what's more, we have surveyed various methods of computing the electric field. One of these method provides a second order accurate electric field, sufficient for a spatially second order accurate PIC code.

Of course, this is computationally more intensive than using linear finite elements for ϕ, in which case $\mathbf{E}$ cannot be more than first order accurate. It is interesting to observe, though, that obtaining a continuous electric field by taking the arithmetic mean gives practically as good results as any first order method, while being computationally cheaper than most, and only marginally more expensive than leaving the field discontinuous.

At last, it is also in place to discuss some of the established stability criteria of PIC codes in light of unstructured meshes and objects. The numerical integrators may have stability criteria of their own, such as, for the leapfrog method, $\omega\Delta t < 2$ for any angular frequency ω present in the system [16]. It is well known that due to the collective behaviour of the particles it is not enough to consider the stability of such integrators on their own, but must instead be considered for an ensemble of particles under mutual influence. [5] gives the criteria $\omega_{pe}\Delta t < 1.62$ for a warm, Maxwellian distributed plasma with electron plasma (angular) frequency ω_{pe}. However, we would like to emphasize that in the vicinity of objects, the particle distribution may very well deviate from Maxwellian, and therefore we cannot take the known stability criteria as an absolute truth. The authors are not aware of any theoretical or experimental work to obtain temporal stability criteria for simulations with objects. This could be considered future work. However, it seems reasonable, not just for stability but also for accuracy, to choose $\omega\Delta t \ll 1$.

The spatial discretization in PIC codes has the equivalent effect of replacing charged point particles by charge clouds of finite extent. When particles get sufficiently close to one another, the clouds overlap and the force between them diminishes instead of increases [5]. Therefore, to produce physically meaningful results the mesh must sufficiently resolve any characteristic lengths of the physical processes involved, the smallest typically being the Debye shielding happening at the scale of the electron Debye length λ_{De}.

For a uniform rectangular mesh with stepsize Δx, the charge clouds would be rectangular as well, and the literature typically reports $\Delta x < \varsigma\lambda_{De}$ with ς around 3 as a sufficient criteria [5], [18]. Failure to resolve the Debye length leads to artificial particle accelerations and in turn unstable growth of the thermal energy (numerical heating). Beware, however, that since the Debye length increases with increasing temperature, PIC simulations may stabilize at the wrong temperature once the Debye length is resolved [16]. Anyhow, for an unstructured mesh the charge clouds will not be rectangular or even constant [18], which will likely influence the stability constraint. More work on stability for unstructured meshes with objects would be highly relevant.

VIII. Conclusion

We have created a finite element based particle-in-cell method for plasma–object interaction simulation, which sup-

ports arbitrary number of perfectly electrically conducting objects. These objects can be arbitrarily connected by ideal voltage and current sources, and a novel method automatically formulates mathematical constraints and adds these to the stiffness matrix of the Poisson problem. This allows the Poisson equation to be solved only once per time step, compared to for instance the methods in [22] and [25].

Moreover, the method is implemented using Python and FEniCS, which allows for great flexibility and rapid prototyping. We have used this flexibility to survey various methods of obtaining the electric field from the electric potential, and discussed the benefits of the various methods. Moreover, the user can choose third order accurate Poisson solver, which renders the PIC method second order accurate in space as well as time (using the Boris algorithm for time integration).

At last, we have adressed the need for revising the stability constraints for PIC simulations for methods with unstructured meshes containing objects. Future work should also include a more efficient implementation of the method in a compiled programming language, as well as parallelization.

IX. Acknowledgement

We acknowledge support from the 4DSpace Strategic Research Initiative at the University of Oslo.

References

[1] Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[2] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., USA, 2015.

[3] AJ Barrett. Other functions defined by integrals. In *Special Functions of Mathematics for Engineers–Second edition, Andrews L. C., Oxford University Press, Great Clarendon Street, Oxford OX2 6DP, UK. 1998. 480pp. Illustrated.£ 50.*, volume 102, chapter 3, pages 110–117. Cambridge University Press, 1998.

[4] M Idrees Bhatti and P Bracken. Solutions of differential equations in a bernstein polynomial basis. *Journal of Computational and Applied Mathematics*, 205(1):272–280, 2007.

[5] C. K. Birdsall and A. B. Langdon. *Plasma Physics Via Computer Simulation*. Taylor & Francis Group, 2005.

[6] KL Cartwright, JP Verboncoeur, and CK Birdsall. Loading and injection of maxwellian distributions in particle simulations. *Journal of Computational Physics*, 162(2):483–513, 2000.

[7] David K. Cheng. *Field and Wave Electromagnetics*. Addison-Wesley, 2nd edition, 1989.

[8] Ph Clément. Approximation by finite element functions using local regularization. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):77–84, 1975.

[9] G. L. Delzanno, E. Camporeale, J. D. Moulton, J. E. Borovsky, E. A. MacDonald, and M. F. Thomsen. Cpic: A curvilinear particle-in-cell code for plasma material interaction studies. *IEEE Transactions on Plasma Science*, 41(12):3577–3587, Dec 2013.

[10] Pascal Frey and Paul Louis George. *Mesh Generation: Application to Finite Elements*. Wiley-ISTE, 2 edition, 4 2008.

[11] James E Gentle. Transformations of uniform deviates: General methods. In *Random number generation and Monte Carlo methods*, chapter 4, pages 101–109. Springer Science & Business Media, 2013.

[12] James E Gentle. Transformations of uniform deviates: General methods. In *Random number generation and Monte Carlo methods*, chapter 4, pages 113–124. Springer Science & Business Media, 2013.

[13] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[14] Walter R Gilks and Pascal Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, pages 337–348, 1992.

[15] David J Griffiths. Electrostatics. In *Introduction to electrodynamics*, chapter 2, page 105. Prentice Hall, 3 edition, 1999.

[16] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. Taylor & Francis Group, 1988.

[17] Claes Johnson. *Numerical solution of partial differential equations by the finite element method*. Studentlitteratur, 1987.

[18] Giovanni Lapenta. Particle simulations of space weather. *Journal of Computational Physics*, 2012.

[19] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[20] ATY Lui and SM Krimigis. Energetic ion beam in the earth's magnetotail lobe. *Geophysical Research Letters*, 10(1):13–16, 1983.

[21] M. J. Mandell, V. A. Davis, D. L. Cooke, A. T. Wheelock, and C. J. Roth. Nascap-2k spacecraft charging code overview. *IEEE Transactions on Plasma Science*, 34(5):2084–2093, Oct 2006.

[22] Richard Marchand. PTetra, a tool to simulate low orbit satellite–plasma interaction. *IEEE Transactions On Plasma Science*, 2012.

[23] Richard Marchand and Pedro Alberto Resendiz Lira. Kinetic simulation of spacecraft–environment interaction. *IEEE Transactions on Plasma Science*, 45(4):535–554, 2017.

[24] Wojciech J Miloch. Wake effects and mach cones behind objects. *Plasma Physics and Controlled Fusion*, 52(12):124004, nov 2010.

[25] Yohei Miyake and Hideyuki Usui. New electromagnetic particle simulation code for the analysis of spacecraft-plasma interactions. *Physics of Plasmas*, 16(6):062904, 2009.

[26] T. Muranaka, S. Hosoda, J. H. Kim, S. Hatta, K. Ikeda, T. Hamanaga, M. Cho, H. Usui, H. O. Ueda, K. Koga, and T. Goka. Development of multi-utility spacecraft charging analysis tool (muscat). *IEEE Transactions on Plasma Science*, 36(5):2336–2349, Oct 2008.

[27] James W. Nilsson and Susan A. Riedel. *Electric Circuits*. Pearson Prentice Hall, 7th edition, 2005.

[28] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[29] Gerd W. Prölss. *Physics of the Earth's Space Environment: An Introduction*. Springer, 2004.

[30] Hans L. Pécseli. *Fluctuations in Physical Systems*. Cambridge University Press, 1 edition, 8 2000.

[31] Hong Qin, Shuangxi Zhang, Jianyuan Xiao, Jian Liu, Yajuan Sun, and William M. Tang. Why is boris algorithm so good? *Physics of Plasmas*, 20(8):084503, 2013.

[32] Alfio Quarteroni and Silvia Quarteroni. Elements of functional analysis. In *Numerical models for differential problems*, volume 2, chapter 2, page 17. Springer, 2009.

[33] J. F. Roussel, F. Rogier, G. Dufour, J. C. Mateo-Velez, J. Forest, A. Hilgers, D. Rodgers, L. Girard, and D. Payan. Spis open-source code: Methods, capabilities, achievements, and prospects. *IEEE Transactions on Plasma Science*, 36(5):2360–2368, Oct 2008.

[34] Yousef Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. Society for Industrial and Applied Mathematics, 2003.

[35] Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 1 edition, 9 2003.

[36] Vytenis M Vasyliunas. A survey of low-energy electrons in the evening sector of the magnetosphere with ogo 1 and ogo 3. *Journal of Geophysical Research*, 73(9):2839–2884, 1968.