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

Sigvald Marholm*, Diako Darian†, Mikael Mortensen†, Richard Marchand‡, and Wojciech J. Miloch*
*Department of Physics, University of Oslo, P.O. Box 1048 Blindern, N-0316 Oslo, Norway †Department of Mathematics, University of Oslo, P.O. Box 1048 Blindern, N-0316 Oslo, Norway ‡Department of Physics, University of Alberta, Edmonton AB, Canada

***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 objects connected by voltage and current sources in an arbitrary circuit topology. Moreover, by means of an unstructured mesh, the objects can have arbitrary shape. The electric potential of the objects are solved self-consistently by incorporating charge constraints into the finite element discretization of the Poisson equation.**

This method has been implemented in a new code, PUNC, suitable for rapid prototyping. The flexibility of this code has proven convenient to survey various methods, and an issue of reduced convergence rate of todays unstructured plasma–object interaction codes is highlighted. The results for a conducting sphere immersed in Maxwellian plasma are in good agreement with previous studies.

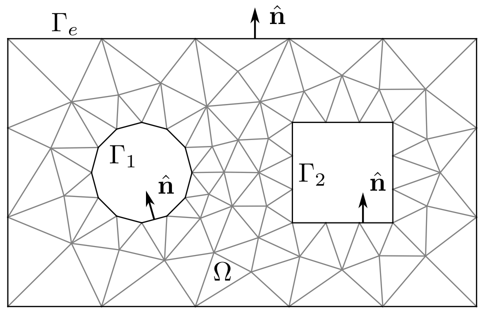


Fig. 1. Example of a mesh on a 2D simulation domain Ω with exterior boundary Γ_e and two objects delimited by two interior boundaries Γ_1 and Γ_2. The normal vector $\hat{\mathbf{n}}$ on the boundaries points away from the simulation domain (and thus into the objects). The mesh is created using Gmsh [9].

I. Introduction

Numerical simulations of objects interacting with plasma is one of the main challenges in plasma simulation techniques, which are now becoming more feasible with increasing computational power. 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 electric 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 studies of plasma-object interactions, each with its own strengths, for instance DiP3D [1], EMSES [2], MUSCAT [3], PDP2 [4], Nascap-2k [5], CPIC [6], SPIS [7] and PTetra [8]. Of these, DiP3D, EMSES, MUSCAT and PDP2 all use uniform Cartesian structured grids. This is restrictive in the sense that 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, when 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 use the *Finite Element Method* (FEM) on such meshes. It should be mentioned that Nascap-2k and CPIC also fit 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 also desirable to arrange the conducting parts in circuits. Spacecrafts and rockets can have many instruments, and parts of the instruments often consists of exposed metals that are electrically connected to the spacecraft [10], [11], [12]. This makes support for generic circuits, as well as unstructured meshes desirable. RLC circuits have already been treated in [4], [13]. However, this treatment does not allow for a generic circuit topology, since circuits are always considered between an object/electrode and a reference electrode (system ground). In addition, the derivation assumes a Cartesian mesh. A more generic topology which involves voltage sources is considered in MUSCAT [3]. It uses the *capacitance matrix method* to enforce interior boundary conditions, which requires solving the Poisson equation twice per time-step. This method is not necessary for unstructured meshes, where interior boundaries are easily generated. Generic support for circuits on

Corresponding author: S. Marholm (email: sigvaldm@fys.uio.no).

unstructured meshes is offered by SPIS [7], which introduces an ad-hoc circuit solver. This solver needs the capacitance of the spacecraft, which is computed from the last time-step. This limits the method to first-order temporal accuracy. PTetra [8] also implements a generic topology with voltage sources, by using a linear decomposition of the Poisson equation to achieve easily enforced Dirichlet boundaries. Such decompositions require two solutions of the Poisson equation per time-step, or alternatively, storing as many field quantities as there are conducting parts in the spacecraft or geometry to be simulated.

We present a new method for circuits of arbitrary topology in PIC simulations. It includes voltage and current sources, and can be extended to include other components through voltage-dependent current sources. It is not limited to first-order accuracy, and it still only requires one solution of the Poisson equation per time-step, with no additional storage. The method is derived for continuous fields and thus independent of the mesh, but requires implementation of new boundary conditions in the field solver. We present how these boundary conditions can be implemented for the finite element method, in a new open source code called PUNC (*Particles-in-Unstructured-Cells*).

While insulating objects are also of great practical importance, they are considered outside the scope of the present paper. In so far as their effect can be modelled by conducting objects and circuits, the methods herein may be applicable. Marchand also describes a method of partitioning the surface of an insulating object into several conducting segments, in order to approximate insulating objects [8], and SPIS shows an example of how to treat an insulating coating on an object. The more accurate representations of dielectric objects, however, allow for charge re-distribution throughout the object (for non-perfect insulators), and a non-uniform potential in the object, see, e.g., [4]. This requires a mesh in the interior of the object, which is in contrast with our treatment.

PUNC has been developed as a rapid prototyping tool to study various methods, including the new object method. It is implemented in Python and centered around FEniCS [14], [15], 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 one of several efficient third-party libraries. The flexibility inherited from both Python and FEniCS makes it easy to test new concepts, and to add custom diagnostics. The usefulness of such a tool should be evident from the variety of cases considered herein, such as a comparison study of different methods for obtaining the electric field, and the easy transition between different finite element spaces.

The layout of the paper is as follows: in Sec. II we briefly present background material on the PIC method. Then, in Sec. III we describe the methods related to particle quantities, including a novel method for sampling new particle velocities. The new circuit method is presented in Sec. IV, and the implementation of the resulting boundary conditions in Sec. V. Subsequently, the interaction between the mesh and particles are explained in Sec. VI, before numerical experiments on convergence and correctness are presented in Sec. VII. A discussion and conclusion are given in Sec. VIII and Sec. IX, respectively.

II. Background

We shall begin by giving an overview of the PIC method. The basic idea behind the method is well established [16], but we choose to provide it here for completeness. 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 its equations of motion. 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 time-step would be $\mathcal{O}(N_P^2)$ [17, pp. 18–20]. To reduce this complexity, 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 the equations of motion:

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

where q_p, m_p and $\mathbf{v}_p$ is, respectively, the charge, mass and velocity of the particle, and 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 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. This can for instance be the magnetic flux density of the Earth. The electric field $\mathbf{E}$, however, depends on contributions from all particles, and is given by Poisson's equation, along with the definition of the electric potential ϕ:

$$-\nabla^2\phi = \frac{\rho}{\varepsilon_0}, \qquad \mathbf{E} = -\nabla\phi. \tag{2}$$

Here, 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 needs to be assigned to nodes in this mesh by some weighting scheme. When equations (2) 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 [17, p. 21], typically $\mathcal{O}(N_G \log N_G)$ or even $\mathcal{O}(N_G)$ when multigrid methods are used [18, p. 137]. Usually it is necessary to have many particles per cell, in the order of tens, to keep the statistical noise sufficiently low, hence $N_P \gg N_G$. Clearly, the computational complexity is significantly lower than in the particle–particle approach. Further details on the PIC method can be found in [16], [17], [19], [20].

A. Simulation Particles

The number of particles in a simulation volume is often very high. For instance, the electron density is in the order

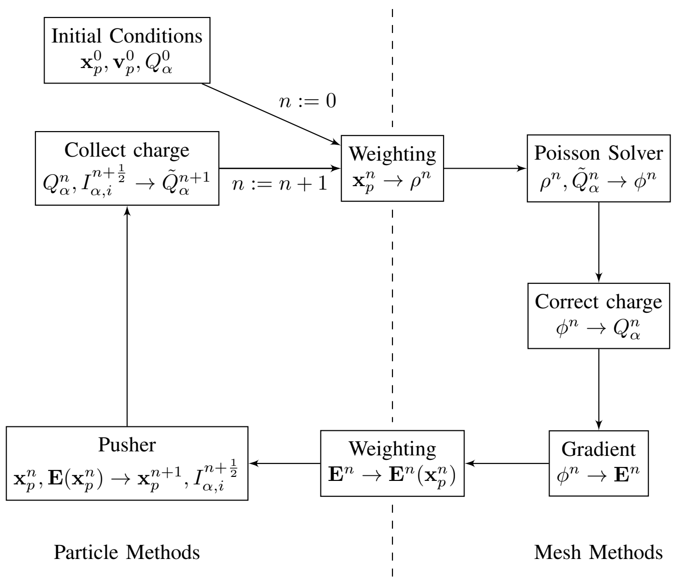

Initial Conditions
$\mathbf{x}_p^0, \mathbf{v}_p^0, Q_\alpha^0$
$n := 0$
Collect charge
$Q_\alpha^n, I_{\alpha,i}^{n+\frac{1}{2}} \to \tilde{Q}_\alpha^{n+1}$
$n := n + 1$
Weighting
$\mathbf{x}_p^n \to \rho^n$
Poisson Solver
$\rho^n, \tilde{Q}_\alpha^n \to \phi^n$
Correct charge
$\phi^n \to Q_\alpha^n$
Pusher
$\mathbf{x}_p^n, \mathbf{E}(\mathbf{x}_p^n) \to \mathbf{x}_p^{n+1}, I_{\alpha,i}^{n+\frac{1}{2}}$
Weighting
$\mathbf{E}^n \to \mathbf{E}^n(\mathbf{x}_p^n)$
Gradient
$\phi^n \to \mathbf{E}^n$
Particle Methods
Mesh Methods

Fig. 2. Overview of the PIC cycle as used in PUNC. Each step indicates new quantities computed from previous quantities. Subscripts p and α 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.

of 10^{10}–$10^{12}\,\mathrm{m}^{-3}$ in the ionosphere [21, 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 to use *simulation particles* [16], each corresponding to w_s physical particles, as specified by the user. The charge, mass and density, respectively, of simulation particles of a species s is then,

$$q_s' = w_s q_s, \qquad m_s' = w_s m_s, \qquad n_s' = \frac{n_s}{w_s}. \tag{3}$$

n_s is the density of physical particles. The velocity distribution is the same for the simulation particles as for the physical particles. It is then easily verified that physical characteristic scales, such as the (angular) plasma frequency ω_{ps}, the Debye length λ_{Ds} and the thermal speed $v_{\text{th},s}$ [22],

$$\omega_{ps} = \sqrt{\frac{q_s^2 n_s}{\varepsilon_0 m_s}}, \quad \lambda_{Ds} = \sqrt{\frac{\varepsilon_0 k_B T_s}{q_s^2 n_s}}, \quad v_{\text{th},s} = \sqrt{\frac{k_B T_s}{m_s}}, \tag{4}$$

are invariant under this transformation. Here, k_B is the Boltzmann constant and T_s is the temperature (beware that the temperature of the simulation particles is not the same as for the physical particles). The reduced number of particles will of course increase the relative statistical noise, which is proportional to $(n_s')^{-1/2}$ [20], [23, p. 98].

III. Particle Methods

In principle any correct numerical integration of (1) can be used to update the position of the particles, but PUNC uses the Boris method for its numerical efficiency, long-term stability and second order accuracy [17, p. 97], [24]. Let a superscript n indicate that a quantity is approximated at time $t = n\Delta t$, Δt being the time-step. The Boris method uses a staggered temporal grid where the position is stored at integer time-steps, $\mathbf{x}_p^n$, while the velocity is stored at half-integer time-steps, $\mathbf{v}_p^{n+\frac{1}{2}}$. To offset the velocity by half a time-step, a normal forward Euler step [25, p. 317] is used the first time. In addition to position and velocity, each particle has its 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.

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. VI for all particles in a cell as a batch. After updating the particle positions one must therefore also update which cell they belong to. This is done following the approach in PTetra [8], described in the following.

Let $\hat{\mathbf{n}}_f$ be the outwards-pointing unit 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 facet has no adjacent cell, i.e., it is part of a boundary, we instead store which boundary it is part of so we can easily identify which boundary the particles cross.

The recursion depth, and hence the speed of this algorithm, is directly related to $v_p \Delta t / h_p$ of the average particle, where h_p is the mesh resolution (cell diameter) in the vicinity of the particle. If, on average, $v_p \Delta t \gg h_p$, most particles will cross many cells per time-step and the algorithm will be slow. However, in this case the mesh is either needlessly fine, or Δt too large, because the particle trajectories are not able to resolve the mesh. It is therefore customary to choose Δt such that $v_p \Delta t / h_p \lesssim 1$ for the majority of the particles at any region of the domain. Many particles, however, will reside in areas of a coarser mesh, and travel at slower speeds (especially ions). Consequentially, the average $v_p \Delta t / h_p$ is usually significantly smaller than one, resulting in far fewer than one cell crossing per particle per time-step. The algorithm then only requires a few arithmetic operations per particle, and is usually comparable to the cost of advancing the particles' positions.

The algorithm is not suitable for newly inserted particles, however. Not only would it lead to deep and costly recursion, but if the domain is non-convex, e.g., it has objects, starting the recursion on the opposite side of an object of where the particle actually is would lead to deletion of the particle. To locate the host cell of newly inserted particles we instead use an axis-aligned bounding box tree[1].

A. Random Number Sampling

One of the components of the PIC method is to efficiently sample random particle positions and velocities for particle loading and injection of new particles into to the simulation domain. Depending on the given probability density function

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

(pdf) there exists several different methods [26] to sample random numbers from the distribution function. One of the distributions often used in plasma simulations is the D-dimensional shifted-Maxwellian distribution [8],

$$\operatorname{pdf}_{\mathrm{M}}(\mathbf{v};\mathbf{v}_{\mathrm{d}},v_{\mathrm{th}}) = \left(\frac{1}{\sqrt{2\pi}v_{\mathrm{th}}}\right)^{D} \exp\left(-\frac{\|\mathbf{v}-\mathbf{v}_{\mathrm{d}}\|^2}{2v_{\mathrm{th}}^2}\right), \quad (5)$$

of the velocity $\mathbf{v}$, where $\mathbf{v}_{\mathrm{d}}$ is the drift velocity. There exists several algorithms [27] to generate random numbers from the shifted-Maxwellian distribution. One of these methods, which is implemented in PUNC, is the *inverse transform sampling* method [26], [28].

However, the particle velocity distributions in space plasmas, as confirmed by spacecraft measurements [29], [30], are mostly non-Maxwellian. It is, therefore, also desirable to simulate plasma conditions where the velocity pdf is neither the shifted-Maxwellian distribution, nor separable, and the inverse transform sampling method may no longer be applicable. For this purpose, in the following, we present a new and optimized rejection sampling method to generate random velocities from effectively any pdf with any form. Our method is based on the *standard rejection sampling method* [31], which is a standard Monte Carlo technique that uses a simpler proposal distribution, ppdf, to generate samples. A sample $\mathbf{v}$ is then accepted with probability $\operatorname{pdf}(\mathbf{v})/\operatorname{ppdf}(\mathbf{v})$, or discarded otherwise. This process is repeated until a sample is accepted. The easiest possible proposal distribution is a uniform distribution, see Fig. 3. However, the main issue with this choice of ppdf, is that many of the generated samples may be rejected. An improved method is presented in [32]. However, this method is limited to one-dimensional and log-concave distribution functions. Instead, to decrease the number of rejected samples, and hence, increase the efficiency of generating random velocities, we divide the velocity domain into subdomains or bins. In each bin, a function which approximates the target pdf will be constructed. The set of all of these piece-wise functions constitutes the proposal pdf. Below, we describe a simple way of constructing such a proposal pdf, which is both efficient and it is easy to generate random samples from. With random velocities generated from the proposal pdf, the standard rejection sampling method will be employed in order to accept those velocities whose distribution is the target pdf.

Let $\operatorname{pdf}(\mathbf{v})$ be the target distribution, with $\mathbf{v} \in \mathbb{R}^D$. Due to numerical considerations, it is customary to introduce upper and lower velocity cutoffs, such that $\mathbf{v} \in \Xi \subset \mathbb{R}^D$. The computational velocity domain Ξ, is partitioned into N_B disjoint (non-overlapping) bins $\{B_j\}_1^{N_B}$. Although there is no restriction on how the velocity bins are constructed, for simplicity, we have used a non-uniform Cartesian mesh. An illustration of the partitioning of the velocity domain for $D = 1$ is given in Fig. 3.

The next step is to build the proposal pdf, which approximates $\operatorname{pdf}(\mathbf{v})$ in each bin B_j, and it is easy to sample from. The simplest choice is a piece-wise constant function with value

$$\xi_j = \max_{\mathbf{v}\in B_j} \operatorname{pdf}(\mathbf{v}) \quad (6)$$

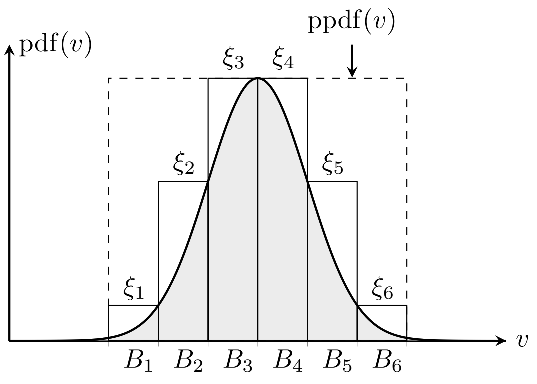


Fig. 3. Illustration of partitioning of velocity domain into disjoint bins, and the corresponding proposal distributions in each bin for $D = 1$. The proposal distribution $\operatorname{ppdf}(v)$, used in standard rejection sampling, is also shown.

in bin B_j. We have evaluated (6) by taking the maximum of $\operatorname{pdf}(\mathbf{v})$ in the corners of B_j. This assumes that the pdf is monotonous in each bin, see Fig. 3. The discrete (unnormalized) cdf of the proposal pdf is then obtained by

$$\bar{\xi}_0 = 0, \quad (7)$$

$$\bar{\xi}_k = \sum_{j=1}^{k} \xi_j \operatorname{Vol}(B_j), \quad k = 1, \ldots, N_B, \quad (8)$$

where $\operatorname{Vol}(B_j)$ is the volume of bin B_j. Note that although the pdf is D-dimensional, the discrete cdf is considered one-dimensionally.

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 in $(0, \bar{\xi}_{N_B})$. Find index k such that

$$\bar{\xi}_{k-1} \le r_0 < \bar{\xi}_k. \quad (9)$$

The index k corresponds to bin B_k. Since $\bar{\xi}_k$ is in ascending order, a binary search makes this very fast.

2) *Sampling step:* Sample a random velocity uniformly within B_k, i.e., $\mathbf{v}_p \sim \mathcal{U}(B_k)$, and a uniformly distributed random number $r_1 \sim \mathcal{U}(0, 1)$.

3) *Rejection step:* If

$$r_1 \le \frac{\operatorname{pdf}(\mathbf{v}_p)}{\xi_k},$$

then accept $\mathbf{v}_p$. Otherwise, reject $\mathbf{v}_p$ and go back to step 1.

With this method, owing to close tightness between the proposal and target pdfs, the majority of sampled velocities are accepted. Another possible way of constructing a proposal pdf would be using piece-wise linear hyperplanes tangent to $\operatorname{pdf}(\mathbf{v})$, but this will not be considered here.

B. Initial Conditions

For each species the simulation domain Ω is initially populated with a prespecified number of simulation particles. By default, a uniform probability distribution $\mathcal{U}(\Omega)$, is used to generate a random position for each particle in the simulation

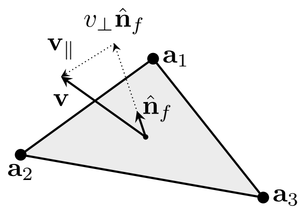


Fig. 4. Illustration of tangential and normal components of a velocity vector. Here, $\hat{\mathbf{n}}_f$ is the outward-pointing (out of Ω) unit normal vector, and $v_\perp = \mathbf{v} \cdot \hat{\mathbf{n}}_f$.

domain. However, PUNC offers the user to specify the initial probability distribution function for the position of particles of each species. Given a probability distribution other than the uniform distribution, the standard rejection sampling method [31] is used to generate random particle positions. In order to assign initial velocities to simulation particles, a velocity probability distribution function must be specified for each species. For the special case of a shifted-Maxwellian distribution function the inverse transform sampling method is implemented. For other types of velocity distribution functions the optimized rejection sampling method in Sec. III-A can be used.

C. Exterior Boundary Conditions

The exterior boundaries of the simulation domain are assumed to be open. They are placed far away in terms of Debye lengths 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 new particles from each species must be injected into the domain in accordance with the velocity distribution function of the ambient plasma. We note, however, that in some cases, such as a supersonic flow around an object, acceptable results are attained even when the velocity distribution is perturbed at the boundary [33]. Long extending perturbations of the wake in supersonic flows can lead to a local error at the downstream boundary, but it is being carried out of the domain with the flow and does not propagate upstream more than a few Debye lengths.

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 $\mathrm{pdf}(\mathbf{v})$ be the velocity distribution (with $\int_{\mathbb{R}^D} \mathrm{pdf}(\mathbf{v})\,\mathrm{d}\mathbf{v} = 1$) for a given ambient plasma species with particle number density n'. The flow of particles creates a flux through the facet f per time Δt and facet area ΔS_f, given by

$$\Phi_f = \Delta t \Delta S_f n' v_\perp, \tag{10}$$

where, with reference to Fig. 4, $v_\perp = \mathbf{v} \cdot \hat{\mathbf{n}}_f$. It is clear that only particles that have $v_\perp < 0$ 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 (10), i.e.,

$$N_{p,f} = \Delta t \Delta S_f n' \int_{\mathbb{R}^{D-1}} \int_{-\infty}^{0} v_\perp \,\mathrm{pdf}(\mathbf{v})\,\mathrm{d}v_\perp\,\mathrm{d}\mathbf{v}_\parallel, \tag{11}$$

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

For the shifted-Maxwellian velocity distribution, (11) evaluates to an analytical expression [8], however, for other types of distributions, if an analytical expression is not attainable, $N_{p,f}$ must be obtained by numerically evaluating (11).

For each of the $N_{p,f}$ particles of each species, a uniformly distributed random position on the facet must be generated. In 2D, the exterior facet element is a straight line, and in 3D, it is a triangle. Detailed descriptions of generating uniformly distributed random points on a straight line or a triangle can be found in, e.g., [8], [34].

Each generated particle must be given a random velocity based on the velocity distribution function $\mathrm{pdf}(\mathbf{v})$, of the corresponding ambient plasma species, 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

$$\mathrm{pdf}_f(\mathbf{v}) = \begin{cases} -v_\perp \,\mathrm{pdf}(\mathbf{v}), & \text{if} \quad v_\perp < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

where $v_\perp$ is the velocity component normal to the facet. The optimized rejection sampling method, described in the Sec. III-A, 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 [27], [35]. 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 entire 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 [27]. For general spatially and temporally varying electric field and magnetic flux density, a third-order injection method has also been suggested [27]. Because we assume that perturbations do not reach the boundary, we have only implemented the simplest method.

IV. Circuit Methods

In this section we present a general framework for circuits between perfectly conducting objects in PIC codes, where objects can be connected in an aribtrary circuit topology through voltage and current sources. An example is shown

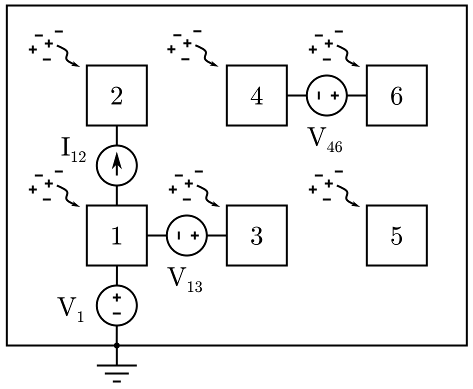


Fig. 5. Example of a simulation of three circuits consisting of six objects (squares). Objects 2, 4, 5 and 6 have unknown potentials, whereas objects 1 and 3 have known potentials, the potential of object 3 obviously being $V_1 + V_{13}$. The unknown potentials are solved self-consistently using accumulated charges. 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.

in Fig. 5. Other components, such as resistors, inductors and capacitors, can be implemented as voltage-dependent current sources. Circuits in PIC codes have already been extensively studied, although with less general support for circuit topologies. Verboncoeur *et al.* [13] considered a series RLC-circuit between the left and right boundary of a one-dimensional code. The right boundary was grounded with a fixed zero potential. Vahedi *et al.* [4] extended this to include objects represented as interior boundaries in a two-dimensional domain. However, the circuits were still limited to be between an object and a grounded node, and not, for instance, between two floating objects. Especially the case of an ideal voltage source between two objects poses new challenges. The papers about SPIS [7], MUSCAT [3] and PTetra [8] describe the problem of a satellite at a floating potential, where parts of the satellite may be biased with respect to the main body of the satellite. These support fairly generic circuits.

There are also fundamental differences in the way circuits are implemented. SPIS introduces a fairly generic ad-hoc circuit solver in addition to the Poisson solver, that in essence integrates the current collected by the plasma, $I_c = C_{\text{sat}}\mathrm{d}V/\mathrm{d}t$, to yield the potential at every time-step. This requires some means of determining the satelite's capacitance C_{sat}. In SPIS it can either be specified by the user, or can alternatively be computed by evaluating Gauss' integral equation for electric charge at every time-step (the capacitance may vary due to the plasma)[2]. Since C_{sat} from the last time-step is used, the temporal integration can only be first-order accurate. While this can be sufficient in many cases, we choose to focus on methods where solving the unknown potential is an integral part of the field solver, and which therefore do not have this inherent limitation. Structured, rectangular methods also use special techniques for enforcing interior boundaries. Verboncoeur uses a boolean masking of grid nodes on the boundary, whereas MUSCAT uses the capacitance matrix method by Hockney and Eastwood [17]. These techniques are not necessary on unstructured methods, and are therefore considered outside the scope of this paper. Other differences are highlighted throughout the text, but we mention already now that our method is the only one modelling circuits of arbitrary topology, in multiple dimensions, entierly through enforcing the right boundary conditions on a single solution of the Poisson equation instead of relying upon linear decomposition. In the following we derive these boundary conditions from first principles, in continuous form, such as not to depend on any specific field solver.

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. Implementation-wise, the circuitry is specified by a list of voltage sources, and a list of current sources, e.g., for the circuitry in Fig. 5:

```
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.

For a perfect electric conductor at rest[3], electromagnetic theory provides us boundary conditions for the tangential and normal components of the electric field [36, p. 103]. These can be written, respectively, in terms of the potential as:

$$\nabla\phi \times \hat{\mathbf{n}} = \mathbf{0}, \qquad \nabla\phi \cdot \hat{\mathbf{n}} = \frac{\sigma}{\varepsilon_0}, \tag{13}$$

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

The first condition implies that the objects' surface potential is constant:

$$\phi = V_\alpha \text{ on } \Gamma_\alpha. \tag{14}$$

However, in general, V_α is unknown. This leaves as many unknowns as there are objects, and we need one constraint for each of them in order to close the system of equations. The simplest constraint is to fix the voltage of the objects to a known voltage V_α. In this case (14) is a standard Dirichlet boundary condition, and the Poisson equation can easily be solved. If, on the other hand, V_α is not known a-priori, the second condition in (13) must also be used. Integrating it over the surface of the object reveals the Gauss equation in integral form:

$$\varepsilon_0 \oint_{\Gamma_\alpha} \nabla\phi \cdot \hat{\mathbf{n}}\, \mathrm{d}s = Q_\alpha. \tag{15}$$

If Q_α is known, this closes the system of equations. Notice that in this case the boundary condition is no longer a Dirichlet boundary condition, since Dirichlet boundary conditions require

[2]In addition to the scientific publications, the official SPIS documentation is available on `http://spis.org`.

[3]For a conductor *not* at rest, the electric field will not be zero inside the conductor, but $-\mathbf{v} \times \mathbf{B}$, where $\mathbf{v}$ is the velocity of the conductor and $\mathbf{B}$ the magnetic flux density. Consequently, there will be a tangential electric field on the surface of the object, and the voltage will *not* be constant.

the solution to be known on the boundary [37, p. 311], [38, p. 558].

Methods for circuits in PIC simulations differ on how they compute Q_α, as well as on how they impose (15). For one-dimensional problems, (15) reduces to a Neumann boundary condition and can be solved using standard techniques. This is done by Verboncoeur *et al.* [13]. Vahedi *et al.* [4] (in their method "without decomposition") solve the Poisson equation multiple times with Dirichlet boundary conditions, refining the boundary value at each step. When (15) is discretized it contains values of ϕ on the boundary, as well as inside the domain. Using previous values of ϕ inside the domain they compute a refined value of ϕ on the boundary to use in the next iteration. Perhaps the most common approach, however, is to decompose the potential,

$$\phi = \phi_0 + \sum_\alpha V_\alpha \phi_\alpha, \tag{16}$$

and use the linearity of the Poisson equation. Here, ϕ_0 is the solution of the Poisson equation with $\phi_0 = 0$ on the objects' boundaries, while ϕ_α is the solution with $\phi_\alpha = 1$ on object α's boundary, $\phi_\alpha = 0$ on the other boundaries, and $\rho = 0$. Since ϕ_0 and ϕ_α have Dirichlet boundary conditions, they can be solved using standard techniques. ϕ_α needs only be solved during initialization, whereas ϕ_0 must be solved each time-step. Substitution of (16) into (15) yields a matrix equation that can be used to compute V_α from ϕ_0. ϕ can then be obtained either by performing the summation in (16) [4], or by solving the Poisson equation again with the correct values on the boundaries [8]. The former is obviously faster, but requires storing ϕ_α for all objects in memory, which may be prohibitive for large meshes and complex geometries. Spacecraft simulations, for instance, may consist of tens of objects. We recognize (14) and (15) as a different kind of boundary condition than Dirichlet or Neumann, and in Sec. V we implement this for the finite element method.

To obtain Q_α, we integrate the current $I_\alpha = \mathrm{d}Q_\alpha/\mathrm{d}t$ into the object to second-order accuracy:

$$Q_\alpha^{n+1} = Q_\alpha^n + \Delta t I_\alpha^{n+\frac{1}{2}}, \tag{17}$$

Notice that I_α is the sum of *all* currents into the object, including charges collected from the plasma. Various contributors to this current are discussed in the following subsections.

A. Current Collected from Plasma

A particle is deleted from the simulation when the cell-crossing algorithm described in Sec. III finds that it has crossed the boundary of an object. Moreover, the particles collected by object α between timesteps n and $n+1$ add a charge $\Delta Q_{c,\alpha}^{n+\frac{1}{2}}$ to the object equal to that of all collected particles (the subscript c stands for collection). Hence, the current collected from the plasma is $I_{c,\alpha}^{n+\frac{1}{2}} = \Delta Q_{c,\alpha}^{n+\frac{1}{2}}/\Delta t$.

B. Ideal Current Source

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: $I_2^{n+\frac{1}{2}} = I_{c,2}^{n+\frac{1}{2}} + I_{12}^{n+\frac{1}{2}}$. Current sources of course contribute with an equal current to both objects they are attached to (unless one end is grounded), but with opposite sign.

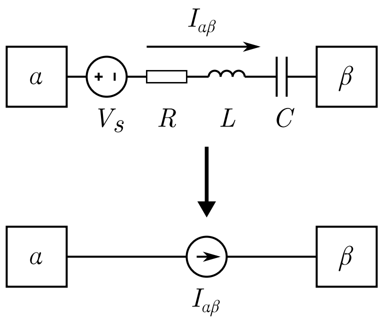


Fig. 6. More complex circuits such as a voltage-driven RLC circuit (top) can be represented as a voltage-dependent current source (bottom).

C. Voltage-Driven Series RLC Circuit

Ideal current sources are more generic than what they may at first appear, since other circuit components can be considered to be voltage-dependent current sources. As an example, we shall demonstrate how a voltage-driven series RLC circuit between two objects α and β can be replaced by an equivalent voltage-dependent current source as depicted in Fig. 6.

The circuit is described by the following ordinary differential equation [39, p. 355]:

$$L\frac{\mathrm{d}^2 I_{\alpha\beta}}{\mathrm{d}t^2} + R\frac{\mathrm{d}I_{\alpha\beta}}{\mathrm{d}t} + \frac{I_{\alpha\beta}}{C} = \frac{\mathrm{d}V_\alpha}{\mathrm{d}t} - \frac{\mathrm{d}V_\beta}{\mathrm{d}t} - \frac{\mathrm{d}V_s}{\mathrm{d}t}. \tag{18}$$

R, L and C are the resistance, inductance and capacitance in the circuit, respectively, and V_s is the driving voltage. By using appropriate finite differences centered around $n-\frac{1}{2}$ this can be discretized as,

$$\begin{aligned} L&\frac{I_{\alpha\beta}^{n+\frac{1}{2}} - 2I_{\alpha\beta}^{n-\frac{1}{2}} + I_{\alpha\beta}^{n-\frac{3}{2}}}{\Delta t^2} + R\frac{I_{\alpha\beta}^{n+\frac{1}{2}} - I_{\alpha\beta}^{n-\frac{3}{2}}}{2\Delta t} \\ &+ \frac{I_{\alpha\beta}^{n+\frac{1}{2}} + I_{\alpha\beta}^{n-\frac{3}{2}}}{2C} = \frac{V_\alpha^n - V_\alpha^{n-1}}{\Delta t} \\ &- \frac{V_\beta^n - V_\beta^{n-1}}{\Delta t} - \left.\frac{\mathrm{d}V_s}{\mathrm{d}t}\right|^{n-\frac{1}{2}} + \mathcal{O}(\Delta t^2). \end{aligned} \tag{19}$$

By rearranging this, $I_{\alpha\beta}^{n+\frac{1}{2}}$ can be computed from previous voltages at integer time-steps and currents at half-integer time-steps. The derivative of the voltage source V_s may either have an analytic expression which can be evaluated at $n-\frac{1}{2}$, or the same discretization as for the other voltages can be used. If the circuit is connected between an object α and system ground instead of between two objects, $V_\beta = 0$.

The numerical stability of the scheme is investigated using von Neumann analysis [37, pp. 47–57]. Let $I_{\alpha\beta}^n = I_{\alpha\beta}^0 \zeta^n$ where the superscript on ζ is an exponent and not an index. Substituting this into (19) and omitting the source terms on the right hand side results in a quadratic equation with the following roots:

$$\zeta = \frac{1 \pm \sqrt{1-(1+\tau_1+\tau_2^2)(1-\tau_1+\tau_2^2)}}{1+\tau_1+\tau_2^2} \tag{20}$$

where

$$\tau_1 = \frac{\Delta t}{2}\frac{R}{L}, \qquad \tau_2 = \frac{\Delta t}{\sqrt{2LC}}. \tag{21}$$

$|\zeta| \leq 1$ for all real, positive τ_1 and τ_2 which means the scheme is unconditionally stable. If the third term in (18) had been discretized as $I_{\alpha\beta}^{n-\frac{1}{2}}/C$ instead of using the average of two currents the scheme would have been unstable.

The resistor, inductor or capacitor can be removed (short-circuited) from this circuit simply by setting $R = 0$, $L = 0$ or $C \to \infty$, respectively. However, if all three are removed such that only the voltage source is left, there is no longer any time constant in the circuit, and the scheme breaks down. In fact, whichever charge needs to move from object α to object β in order to maintain the voltage of the voltage source should flow instantaneously.

Another scheme for voltage-driven series RLC-circuits is considered in Verboncoeur *et al.* [13]. That scheme directly discretizes the charge instead of considering the circuit as a current source, but have otherwise similar properties. That scheme also breaks down when all circuit components are short circuited. Finally, parallel circuits, e.g., a parallel RLC circuit, can be obtained simply by summing over the currents from each branch.

D. Ideal Voltage Source

We have seen that schemes for series RLC-circuits such as the ones presented in Sec. IV-C and in Verboncoeur *et al.* [13] break down when the components are short circuited and we are left with only a voltage source. Vahedi *et al.* [4] also present a scheme with a voltage source. In that case the voltage source is in series with a capacitor, but again, it breaks down when $C \to \infty$. This suggests using a separate method for ideal voltage sources that are not in series with other components.

An object α has an a-priori known potential V_α when it is connected to ground through a voltage source. Likewise, V_α is known when the object is connected through a voltage source to *another* object with known potential. This is the case for objects 1 and 3 in Fig. 5, object 1 having the potential V_1 and object 3 the potential $V_3 = V_1 + V_{13}$. For these objects (14) reduces to a Dirichlet boundary condition, which can be solved using standard techniques. Indeed, this is how for instance Verboncoeur *et al.* [13] deal with a short-circuited RLC circuit, since in their case, the circuit is always between an electrode and ground. It is not so simple, however, when a voltage source connects two objects with unknown potentials, such as objects 4 and 6 in Fig. 5. This could for instance represent a spacecraft at an unknown floating potential, and a Langmuir probe with fixed bias voltage with respect to the spacecraft. As such it is a highly relevant practical problem. PTetra and MUSCAT are able to solve such problems using linear decomposition of the Poisson equation and solving it twice per time-step. We present another method which formulates correct boundary conditions to the Poisson equation directly, and which therefore can obtain the solution in one step. To our knowledge there is no other method that can do it.

We first observe that since currents flow instantaneously through ideal voltage sources, Q_α cannot be determined (a-priori) for objects connected to voltage sources. As a consequence, (15) cannot be used as a constraint to close the system, and a modification of the method is needed. Let $\mathcal{S}^K$ be the set of (indices of) objects whose potential is known a-priori, and $\mathcal{S}^U$ be all other objects. We further partition $\mathcal{S}^U$ into N_z disjoint (non-overlapping) sets $\{\mathcal{S}_z^U\}$ where the objects within each subset $\mathcal{S}_z^U$ are connected to one another through ideal voltage sources. To be precise, an object in $\mathcal{S}_z^U$ cannot be connected through an ideal voltage source to an object in $\mathcal{S}_y^U$ where $y \neq z$, and the subsets should be as small as possible. To consider a specific example, for Fig. 5, the partition would be:

$$\begin{aligned} \mathcal{S}_1^U &= \{2\}, \\ \mathcal{S}_2^U &= \{4,6\}, \\ \mathcal{S}_3^U &= \{5\}. \end{aligned} \tag{22}$$

In PUNC this is represented similarly,

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

and it is automatically derived from `vsources` by following edges in a graph where nodes represent the objects and edges the voltage sources. If a subset contains zero (ground) it is removed from the list since this will be the set $\mathcal{S}^K$ and not a subset of $\mathcal{S}^U$.

A voltage source can instantaneously deliver an arbitrary current to an object in $\mathcal{S}_z^U$ in order to maintain the prescribed voltage difference. However, this current must also be drawn from another object in the same set and thus does not affect the total charge in the set. $\mathcal{S}_z^U$ is a *charge-sharing set*. We can use this to form N_z constraints by taking the sum of (15) over $\mathcal{S}_z^U$:

$$\varepsilon_0 \sum_{\alpha\in\mathcal{S}_z^U} \oint_{\Gamma_\alpha} \nabla\phi \cdot \hat{\mathbf{n}}\, \mathrm{d}s = \sum_{\alpha\in\mathcal{S}_z^U} Q_\alpha, \quad z = 1,...,N_z. \tag{23}$$

Implementation-wise the charge of each object is computed separately using (17), ignoring the voltage sources. This causes no error in the above expression. Once a solution ϕ has been computed, (15) can be evaluated to obtain the correct charge of each individual object. Currents through voltage sources can also be inferred, should they be of interest.

Voltage sources also impose constraints. A voltage source w constraints the voltage between two objects α_w and β_w to be V_w. For N_w voltage sources:

$$V_{\beta_w} - V_{\alpha_w} = V_w, \quad w = 1,...,N_w. \tag{24}$$

Recall that V_{α_w} is just ϕ on Γ_{α_w}, and likewise for V_{β_w}, so this is indeed a constraint of ϕ. For a voltage source connected between an object β_w and system ground, $V_{\alpha_w} = 0$.

All together, there are equally many constraints of form (23) or (24) as there are objects, and these provide closure for the unknowns V_α. For simplicity, we shall write the constraints generically as

$$g_z(\phi) = 0, \quad z = 1, ..., |\mathcal{S}| \tag{25}$$

where for $z = 1, ..., N_z$ these are the charge constraints (23), and for $z = N_z + 1, ..., |\mathcal{S}|$ these are the potential constraints (24). $|\cdot|$ denotes the number of elements in the set.

To summarize, the circuit method consists of 1) updating the charge of each object every time-step using (17), taking into account all currents except due to ideal voltage sources, 2) solving the Poisson equation (2) with boundary conditions on the objects given by equipotentiality (14), along with the constraints (25). The boundary condition on the exterior boundary is not dictated by the circuit method. Finally, 3) correct the charge on the objects by evaluating (15). This provides a general framework for circuits that can in principle be used for any field solver. However, equipotential boundary conditions with charge and voltage constraints must be implemented specifically for the method at hand. In Sec. V we develop these for the finite element method.

V. Mesh Methods

To complete the boundary value problem we also need an exterior boundary condition. This is independent of the circuit method, and our choice is described below. The problem is then discretized using the finite element method [18], [40], taking into account the special boundary conditions outlined in Sec. IV.

A. Exterior Boundary Conditions

In deriving open exterior boundary conditions, we consider a decomposition of the plasma into a constant, homogeneous background, plus local perturbations due to objects. If the exterior boundary is sufficiently far away from any object, the local perturbations in the electric potential can be considered negligible at the boundary, and we only need to take into account the background. This is a suitable boundary condition for instance for studying spacecraft charging phenomena, where the goal is a non-oscillatory, steady-state solution, possibly with a background magnetic flux density due to the Earth. Phenomena which are not local in nature, such as plane wave propagation, would require a different treatment at the boundary.

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

$$\mathbf{v}_\mathrm{d} = \frac{\mathbf{E}_0 \times \mathbf{B}_0}{\|\mathbf{B}_0\|^2}. \tag{26}$$

This implies that the component of $\mathbf{E}_0$ which is perpendicular to $\mathbf{B}_0$ must equal $-\mathbf{v}_\mathrm{d}\times\mathbf{B}_0$. Any parallel component of $\mathbf{E}_0$ must be zero; otherwise all particles in the domain would be accelerated along this component and not maintain the constant drift velocity of the background. There may of course still be local variations in the electric field and bulk velocity, for instance due to the acceleration caused by charged objects, but not near the boundary where only the background plasma is assumed to be important. 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 can therefore be approximated by the following Dirichlet boundary condition [8]:

$$\phi = \phi_0 \overset{\mathrm{def}}{=} (\mathbf{v}_\mathrm{d}\times\mathbf{B}_0)\cdot\mathbf{x} \text{ on } \Gamma_e, \tag{27}$$

where $\mathbf{x}$ is the position on Γ_e and the integration constant is arbitrarily set to zero (the forces only depend on the gradient of ϕ anyway). Another approximation in this boundary condition is the neglected particle noise which would have caused the background potential to deviate from (27) at the boundary. However, since the differential operator ∇^2 in (2) smooths out the particle noise in ϕ, this noise is of lesser concern, especially for large numbers of simulation particles.

It is also worth mentioning that by transforming the reference frame by a velocity $-\mathbf{v}_d$, i.e., to the frame of the plasma, the outer boundary simply becomes zero. However, this leads to a gradient $-\mathbf{v}_d\times\mathbf{B}_0$ in the potential of the objects, which would complicate the description of multiple, connected objects.

Were we to use (only) Neumann or periodic outer boundaries instead of Dirichlet boundaries, a voltage source could be used to prescribe the voltage on one of the objects, since otherwise, the solution would not be unique.

B. Discretization

Assuming $\rho\in L_2(\Omega)$ and that the boundaries are sufficiently smooth, the solution ϕ is in the Sobolev space $H^1(\Omega)$ [18, p. 92]. More specifically, for our problem with objects and circuits, it is in a subspace $H^1_V(\Omega)\subset H^1(\Omega)$ constructed to fulfill the exterior boundary condition, equipotentiality, charge constraints and potential constraints:

$$\begin{aligned} H^1_V(\Omega) = \{\phi\in H^1(\Omega) : \phi = \phi_0 &\text{ on } \Gamma_e,\\ \phi = \text{const on } \Gamma_\alpha, &\quad \forall\alpha\in\mathcal{S},\\ g_z(\phi) = 0, &\quad z = 1,...,|\mathcal{S}|\}. \end{aligned} \tag{28}$$

Using standard techniques [18, pp. 26–30], the boundary value problem can then equivalently be stated as the variational problem of finding $\phi\in H^1_V(\Omega)$ such that

$$a(\phi,\psi) = \langle\rho,\psi\rangle, \quad \forall\psi\in H^1_0(\Omega), \tag{29}$$

where,

$$a(\phi,\psi) = \varepsilon_0\int_\Omega \nabla\phi\cdot\nabla\psi\,\mathrm{d}\mathbf{x}, \quad \langle\rho,\psi\rangle = \int_\Omega \rho\psi\,\mathrm{d}\mathbf{x}, \tag{30}$$

and,

$$H^1_0(\Omega) = \{\phi\in H^1(\Omega) :\ \phi = 0 \text{ on } \partial\Omega\}. \tag{31}$$

To solve this, the domain is first partitioned into an unstructured mesh using, e.g., Gmsh [9]. $H^1(\Omega)$ is then discretized into a finite dimensional space with basis functions $\{\psi_j(\mathbf{x})\}$ such that members of this space can be written in terms of their coefficients $\{\phi_j\}$ as

$$\phi(\mathbf{x}) = \sum \phi_j\psi_j(\mathbf{x}). \tag{32}$$

We use the Lagrange finite element spaces CG_r of continuous, piecewise polynomials of order r. The mesh nodes $\mathbf{x}_i$ and basis functions ψ_j of this space are such that $\psi_j(\mathbf{x}_i) = \delta_{ij}$, where δ_{ij} is the Kroneker delta, and varying polynomially of order r within each cell. Substituting this into (29) and using the linearity of a reveals a discretized version of the variational problem:

$$\sum_{j\in\mathcal{J}} A_{ij}\phi_j = b_i, \quad \forall i \in \mathcal{I}, \tag{33}$$

where

$$A_{ij} = a(\psi_j, \psi_i), \quad b_i = \langle \rho, \psi_i \rangle, \quad \forall i \in \mathcal{I}, \quad \forall j \in \mathcal{J}. \tag{34}$$

Here, $\mathcal{J}$ is the set of (indices of) all basis functions, and $\mathcal{I}$ is the subset being zero on the boundaries, i.e., those corresponding to $H^1_0(\Omega)$. Since the latter set is smaller, the system of equations is as of now underdetermined. We shall constrain the solution space in subsequent sections by strongly imposing the boundary conditions, and thereby close the system of equations.

C. Exterior Boundary Condition

The exterior Dirichlet boundary condition is imposed in the standard way [40, pp. 201–203]. For CG_r elements we have that on a boundary node $\mathbf{x}_i$, $\phi(\mathbf{x}_i) = \phi_i$, and since $\phi = \phi_0$ on Γ_e we set $\phi_i = \phi_0(\mathbf{x}_i)$. In the system of equations this means:

$$A_{ij} = \delta_{ij}, \quad b_i = \phi_0(\mathbf{x}_i), \quad \forall i \in \mathcal{I}_e, \quad \forall j \in \mathcal{J}, \tag{35}$$

where $\mathcal{I}_e$ is the subset of basis functions being non-zero on Γ_e.

D. Interior Boundary Conditions – Equipotentiality

Equipotentiality on an interior boundary Γ_α means that ϕ_i are equal for all nodes on Γ_α. We propose the following method to achieve this: Let nodes i and j be neighbors if they are nodes on a common cell. Let $\mathcal{N}_i$ be the set of all neighbors of node i also on Γ_α. We would like to set ϕ_i to the average of all its surrounding boundary nodes, i.e.,

$$\phi_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} \phi_j, \tag{36}$$

for all ϕ_i on Γ_α except one. To omit one is crucial, since otherwise the equations would be linearly dependent. In matrix terms this means,

$$A_{ij} = \begin{cases} |\mathcal{N}_i|, & j = i \\ -1, & j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases}, \quad b_i = 0, \tag{37}$$

$\forall i \in \mathcal{I}_\alpha, \forall j \in \mathcal{J}, \forall \alpha \in \mathcal{S}$, where $\mathcal{I}_\alpha$ is the set of all basis functions being non-zero on Γ_α except one. We shall denote the index of the remaining one by i_α for later. A more naïve approach would be to set one boundary node equal to another, and so forth. However, it would be cumbersome to make sure all equations are linearly independent, and that any equation only consists of neighboring nodes. With the proposed method, the sparsity pattern normally used in finite element discretization is preserved. Moreover, the method is agnostic of dimensionality and the order r of the CG_r space.

E. Interior Boundary Conditions – Charge Constraints

Finally, there is one row i_α missing in the matrix equation for each object α, and these are the charge and potential constraints. We shall take the charge constraints as the first N_z of these rows. They are found by substituting (32) into (23):

$$A_{ij} = \varepsilon_0 \sum_{\alpha\in\mathcal{S}^U_z} \oint_{\Gamma_\alpha} \nabla\psi_j \cdot \hat{\mathbf{n}}\,\mathrm{d}s, \quad b_i = \sum_{\alpha\in\mathcal{S}^U_z} Q_\alpha, \tag{38}$$

$i = i_z, z = 1, ..., N_z, \forall j \in \mathcal{J}$.

F. Interior Boundary Conditions – Potential Constraints

To impose the N_w potential constraints in (24), we pick a node k_w on Γ_{β_w} and another node l_w on Γ_{α_w}, and require $\phi_{k_w} - \phi_{l_w} = V_w$, for each voltage source w. The coefficients of these equations go into rows i_α, with α-indices succeeding those of the charge constraints:

$$A_{ij} = \begin{cases} 1, & j = k_w \\ -1, & j = l_w \\ 0, & \text{otherwise} \end{cases}, \quad b_i = V_w \tag{39}$$

$i = i_{N_z+w}, w = 1, ..., N_w, \forall j \in \mathcal{J}$. For a voltage source between β_w and ground, -1 is replaced by 0. The charge and potential constraints unfortunately do not preserve the sparsity pattern of a common Dirichlet problem. Nevertheless, we have had success in solving this matrix equation using various linear algebra solvers as presented in Sec. VII.

Since the stiffness matrix A remains unaltered between time-steps, its assembly and the computation of the preconditioning matrix are 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.

G. Electric Field

Once ϕ is determined, the electric field $\mathbf{E} = -\nabla\phi$ can be obtained. 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 Sec. VII-B, we shall do a comparison study of five different methods, all available in PUNC. They are briefly described below:

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 [14, pp. 19–21]: find $\mathbf{E} \in [\mathrm{DG}_0(\Omega)]^D$, such that
$$\int_\Omega \mathbf{E}\cdot\Psi\,\mathrm{d}\mathbf{x} = \int_\Omega -\nabla\phi\cdot\Psi\,\mathrm{d}\mathbf{x}, \quad \forall\,\Psi \in [\mathrm{DG}_0(\Omega)]^D. \tag{40}$$
This is equivalent to evaluating $(-\nabla\phi)_T$ in each cell T, which makes it computationally cheap.
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, but with $\mathbf{E}$

and Ψ in $[\mathrm{CG}_1(\Omega)]^D$. In this case one must solve the variational form of the projection, which is more costly.

3) To obtain higher accuracy when computing the electric field, we may solve Poisson's equation (2) 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 $[\mathrm{CG}_1(\Omega)]^D$. Again one must solve the variational form, exactly as the method above, only now with ϕ from a higher order space.
4) 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 $\mathcal{M}_j$ [28]

$$\mathbf{E}(\mathbf{x}_j) = \frac{1}{|\mathcal{M}_j|}\sum_{T\in\mathcal{M}_j}(-\nabla\phi)_T. \tag{41}$$

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 $\mathcal{M}_j$. This method is also used by PTetra.
5) Another method which is commonly used for interpolating non-smooth and discontinuous functions with continuous and piece-wise linear elements is the *Clément-interpolation* (CI) [41]. Like the 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 for all $\Psi \in [\mathrm{DG}_0(\mathcal{M}_j)]^D$

$$\int_{\mathcal{M}_j}\mathbf{E}\cdot\Psi\,\mathrm{d}\mathbf{x} = \int_{\mathcal{M}_j} -\nabla\phi\cdot\Psi\,\mathrm{d}\mathbf{x}. \tag{42}$$

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

$$\mathbf{E}(\mathbf{x}_j) = \frac{1}{\mathrm{Vol}(\mathcal{M}_j)}\sum_{T\in\mathcal{M}_j}(-\nabla\phi)_T\mathrm{Vol}(T). \tag{43}$$

VI. Particle-Mesh Interaction

As is mentioned in Sec. II, the volume charge density must be determined from the position of particles relative to nearest mesh nodes. In traditional structured PIC methods based on finite differences, the charge of each particle inside a given grid cell is distributed to the grid vertices by using some weighting scheme. Using the finite element method, however, the charge density must be expressed in one of the finite element spaces, and is not necessarily defined at the mesh vertices.

In the following, we describe how the electric field is obtained at the position of each particle and discuss several weighting schemes, which may be utilized to determine the volume charge density.

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 [40, pp. 62]. For a simulation particle at position $\mathbf{x}_p$, the electric field is obtained with regular finite element interpolation:

$$E_i(\mathbf{x}_p) = \sum_{j\in\mathcal{J}} E_i(\mathbf{x}_j)\psi_j(\mathbf{x}_p), \quad \forall i \in 1,\dots,D, \tag{44}$$

where ψ_j are the finite element basis functions. To evaluate this expression one must know which cell hosts the particle, and the sum is then taken only across the basis functions being non-zero in that cell. Note that the expression above is valid for any finite element space, e.g., DG_0, CG_1 or CG_2.

B. Charge density assignment

In general, given any appropriate basis function ψ_j associated with the element node $\mathbf{x}_j$, the following weighting scheme can be used to obtain the volume charge density at each element node [42, p. 68]:

$$\rho_j = \frac{1}{\mathcal{V}_j}\sum_p q'_p\psi_j(\mathbf{x}_p), \qquad \mathcal{V}_j = \int_\Omega \psi_j(\mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{45}$$

where $\mathcal{V}_j$ is simply a weighted volume of the cells surrounding node $\mathbf{x}_j$. This also coincides with the scheme considered for curvilinear coordinates in [43] but with finite element basis functions as weight functions.

As a first step, a finite element function space must be specified. The choice of the function space depends on the desired numerical accuracy and efficiency. The simplest finite element space for expressing the volume charge density is DG_0, where a single constant value is assigned to ρ in each cell element, and it is discountinuous from cell to cell. The error of this method is $\mathcal{O}(h)$, which will naturally put restrictions on the accuracy of the electric potential, and hence, the electric field. Among the Lagrangian family of continuous elements, only first order Lagrangian basis functions are appropriate for weighting of particle charges to the element nodes. This is simply due to the fact that higher order Lagrangian basis functions might be negative in some regions of the cell elements. Expressing ρ in the CG_1 function space results in an error of $\mathcal{O}(h^2)$. 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 [44]. In PUNC, the user can choose to express ρ in either DG_0 or CG_1. Bernstein elements are not implemented in PUNC and are not considered further.

For CG_1, the weighted volume can also be easily computed as [42, p. 69]

$$\mathcal{V}_j = \frac{1}{D+1}\sum_{T\in\mathcal{M}_j}\mathrm{Vol}(T). \tag{46}$$

This also happens to be an approximation of the volume of the *Voronoi cell* centered at node $\mathbf{x}_j$, provided the mesh is nearly regular [45], [42]. This point of view is taken in [8].

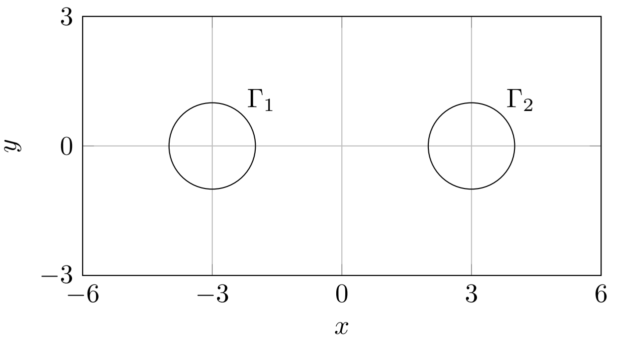


Fig. 7. Domain with two unit-radii circles (actually inscribed polygons) used for validation of the Poisson solver.

VII. Numerical Experiments

Three numerical experiments are performed using PUNC. First, convergence of finite element Poisson solvers taking into account objects is studied. Second, different methods of computing the electric fields are compared. And finally, a validation of a full PIC simulation of a spherical probe at its floating potential. Throughout these examples we change solvers, preconditioners, dimensionality and order of discretization with ease. This demonstrates the unique flexibility in PUNC compared to other PIC codes, much of which is inherited from Python and FEniCS.

A. Multi-Object Poisson Problems

In order to study accuracy and convergence of the method of solving Poisson's equation with objects and various types of constraints, we consider a 2D test case with two circular objects and a rectangular outer boundary like the one depicted in Fig. 7. The source term is inhomogeneous, $\rho/\varepsilon_0 = 100y$, to properly test that the solution is indeed constant on the object boundaries. Since analytical solutions are generally not available for such exotic configurations, a numerical reference solution $\phi_r \in \mathrm{CG}_3(\Omega)$ was made on a fine mesh using standard Dirichlet boundary conditions on all boundaries: $V_1 = 1\,\mathrm{V}$ on the left object, $V_2 = 2\,\mathrm{V}$ on the right object, and $0\,\mathrm{V}$ on the outer boundary. Then, the charges Q_1 and Q_2 of the objects were determined by numerical evaluation of (15). We emphasize that this reference solution only employs standard FEM techniques. With the charges computed from the reference solution, we are able to formulate an equivalent problem. In the equivalent problem, there is a voltage source $V_{12} = 1\,\mathrm{V}$ between the objects, and the objects are co-floating with a known total charge of $Q_1 + Q_2$. This problem tests both charge and voltage constraints (implementation-wise we put the whole charge on object 1 to make sure the voltage constraint is actually applied). To summarize, we have two equivalent problems, a standard Dirichlet problem (fixed potential), and a floating potential problem using our method with both voltage and charge constraints. To test our method we solve ϕ for the floating potential problem on $\mathrm{CG}_1(\Omega)$ and $\mathrm{CG}_2(\Omega)$ for increasingly fine meshes, and compute the L_2 error norm [40, pp. 17]:

$$\|\phi - \phi_r\|_{L_2} = \left(\int_\Omega (\phi - \phi_r)^2 \,\mathrm{d}\mathbf{x}\right)^{\frac{1}{2}}. \tag{47}$$

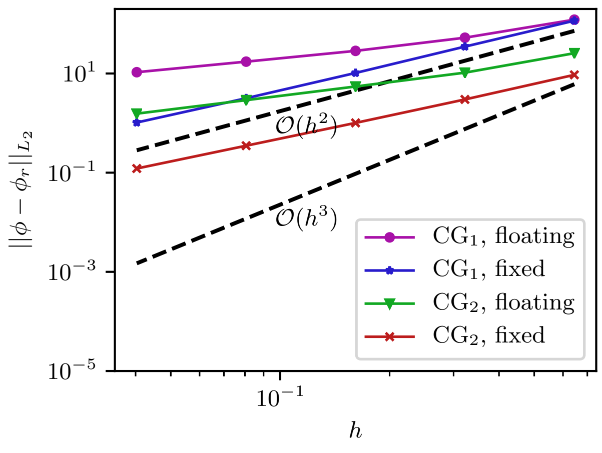
For evaluation of this integral ϕ_r is interpolated onto the same mesh as ϕ, although still of third order. For comparison, we also solve the standard Dirichlet problem on CG_1 and CG_2 for increasingly fine meshes. The solutions in this experiment were obtained using *Incomplete LU* (ILU) preconditioned *Bicongjugate Gradient Stabilized* (BiCGSTAB) linear algebra solver.

A couple of remarks should be made concerning the mesh refinements. First, replacing a curved boundary by line segments is a first order approximation and this will limit the solution to first order regardless of which finite element space is used on the mesh [18, p. 239]. To avoid this restriction, the objects considered are not actually circles, but regular convex polygons with the same number of segments on every refinement of the mesh as on the reference solution. Second, every mesh but the coarsest is made by bisecting the edges of the next coarser mesh (uniform refinement). This ensures that the mesh gets refined throughout the domain. Simply changing the resolution and regenerating the mesh will not suffice, since the cell size close to the object may be limited by the object's geometry, rather than desired resolution.

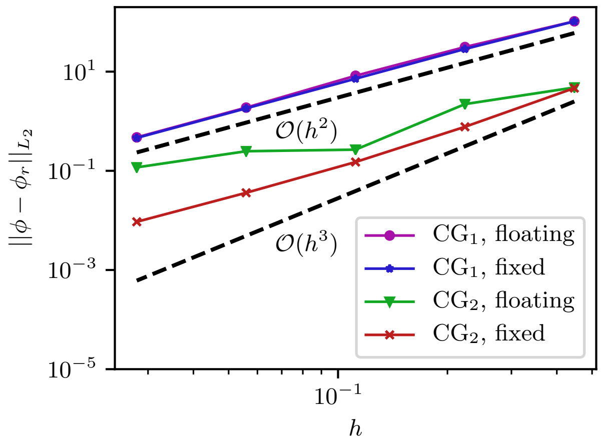
Since it is instructive to consider what happens to the convergence as varying number of segments are used in the polygons, we consider both 5-, 20- and 70-gons, which leads to 226, 282 and 642 cells in the coarsest mesh, respectively. This mesh is refined 4 times (each quadrupling the cell count) and the reference solution is computed on the 6th refinement. It is important that the reference solution is on a much finer mesh, since otherwise the computed charge may not be accurate enough to properly demonstrate convergence. The resulting convergence plots are seen in Fig. 8.

Overall, the solutions seem more accurate for finer meshes, but admittedly, the error is not necessarily $\mathcal{O}(h^2)$ for CG_1 and $\mathcal{O}(h^3)$ for CG_2 which might have been anticipated. It is interesting to observe that this reduced rate of convergence is not exclusive to our method, but is in fact present *even* for the standard Dirichlet FEM method (fixed potential)! Notice for instance that no solution has even quite $\mathcal{O}(h^2)$ for 5 segments. Increasing the number of segments to 20 appears to be good enough to achieve $\mathcal{O}(h^2)$ accuracy, whereas 70 segments are needed for $\mathcal{O}(h^3)$.

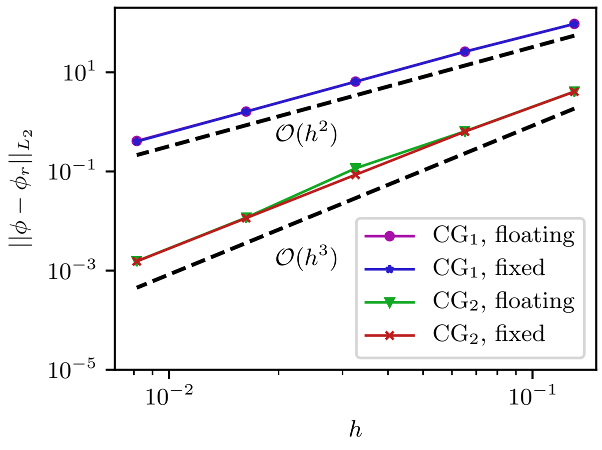
We offer an explanation of this behavior. If order P elements are used to solve a Poisson problem with Dirichlet boundary conditions, it can be shown that the error is $\mathcal{O}(h^{P+1})$ *if* the exact solution is in $H^P(\Omega)$ [18, pp. 90–92]. For smooth boundaries or convex polygonal boundaries, the exact solution can be proved to be in $H^2(\Omega)$ when the source term ρ/ε_0 is in $L_2(\Omega)$ [18, pp. 92–93]. Hence $\mathcal{O}(h^2)$ or $\mathcal{O}(h^3)$ can be achieved using first or second order elements, respectively. However, as Johnson points out, when the boundary Ω has corners with angles greater than 180°, e.g. when the domain has holes, this is not generally the case [18, pp. 92–93]. Since we have no guarantee of the regularity of the exact



a) 5 segments (252° angles)



b) 20 segments (198° angles)



c) 70 segments (185° angles)

Fig. 8. Rate of convergence of the solution ϕ of the Poisson equation on a domain with holes for our method (floating) and the standard Dirichlet FEM problem (fixed).

solution, we have no guarantee of the rate of convergence. This limitation is important, because it applies to all PIC codes using (conventional) finite element methods with objects inside the domain, thereby being non-convex. This includes codes such as SPIS [7] and PTetra [8].

Intuitively, one might expect that as the angles of the interior boundaries decrease towards 180°, the numerical solution should become more accurate. From Fig. 8 this appears to be the case; the more segments, the less the angles on the object boundaries will overshoot 180°, and the better the rate of convergence. We remark that this is only one case study, and that convergence may depend on other simulation parameters as well. Nevertheless, it demonstrates that up to $\mathcal{O}(h^3)$ convergence is achievable under the right circumstances, but that generally, convergence will take place at a reduced rate. We also point out that in the cases of reduced convergence, i.e., when CG$_1$ or CG$_2$ are used on 5 segments or when CG$_2$ are used on 20 segments, it is not clear which method is actually closer to the truth, since the standard Dirichlet method is used as reference.

We suggest further studies to resolve the reduced convergence of plasma–object interaction simulations using FEM-PIC methods. Meanwhile, when the reduced rate of convergence can be tolerated and compensated for by a finer mesh, FEM-PIC codes can still produce valuable results and be the method of choice. Indeed, SPIS and PTetra have already proven useful in a variety of applications requiring great flexibility in mesh geometry, and a full PIC simulation demonstrating the validity of PUNC is presented in Sec. VII-C. For better convergence, smoother object boundaries is advisable.

B. *Comparison of E-field Methods*

To compare the order of accuracy of the electric field obtained by the methods introduced in Sec. V-G, we will in the following consider the simple case of a closed system of two electrically conducting concentric spheres, where the analytical solution is easy to obtain. The inner sphere has a radius $a = 0.1\,\mathrm{m}$, and the outer sphere a 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 [46]

$$\mathbf{E}_e = \frac{(V_a - V_b)ab\mathbf{r}}{(b-a)r^3}, \tag{48}$$

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 using the L_2 norm, which is defined by [40, p. 17]

$$\|\mathbf{e}\|_{L_2} = \left(\int_\Omega \mathbf{e} \cdot \mathbf{e} \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{2}}. \tag{49}$$

The L_2 error norms $\|\mathbf{e}\|_{L_2}$ versus minimum edge-length h for the different methods are displayed in Fig. 9. As expected, using a piecewise constant electric field (DG$_0$) is the least accurate, with approximately a first-order rate of convergence. All other methods have a continuous, piece-wise linear electric field. However, the methods of using the arithmetic mean (AM), Clément-interpolation (CI) and L_2-projection from CG_1 to CG_1 are still only first-order accurate. This is to be expected, since ϕ is only second-order accurate, and its gradient can at most be one degree lower [18, pp. 90–92]. The same is true for finite difference methods, where the error of ϕ is divided by the step-size when taking the gradient. As a consequence, the acceleration of particles is spatially only first-order accurate in PIC codes with second-order accurate Poisson solvers, which includes most electrostatic PIC codes. To have second-order spatial accuracy in all quantities, the code must therefore have a third-order accurate Poisson solver. Indeed, when ϕ is solved to third-order accuracy on CG_2, the electric field can be computed to second-order accuracy for instance by L_2-projection. We remark that again the domain is non-convex, but having a smooth interior boundary, proper accuracy is achieved.

The DG_0 method is the cheapest, followed closely by the AM and CI methods. L_2-projection from CG_1 to CG_1 is about twice as time-consuming, but does not lead to better results than AM or CI. L_2-projection from CG_2 to CG_1 was actually found to be somewhat more efficient than from CG_1 to CG_1, but at the same time, it requires solving Poisson equation on CG_2, which makes the Poisson solver roughly four times as time-consuming. To conclude, the AM method is recommended for most use cases due to its simplicity and improved accuracy at negligible cost compared to DG_0. When a higher order of accuracy is required the potential must be third-order accurate and an L_2-projection to CG_1 can be used. If discontinuous electric fields can be accepted, L_2-projection to DG_1 would likely be faster, although not investigated here.

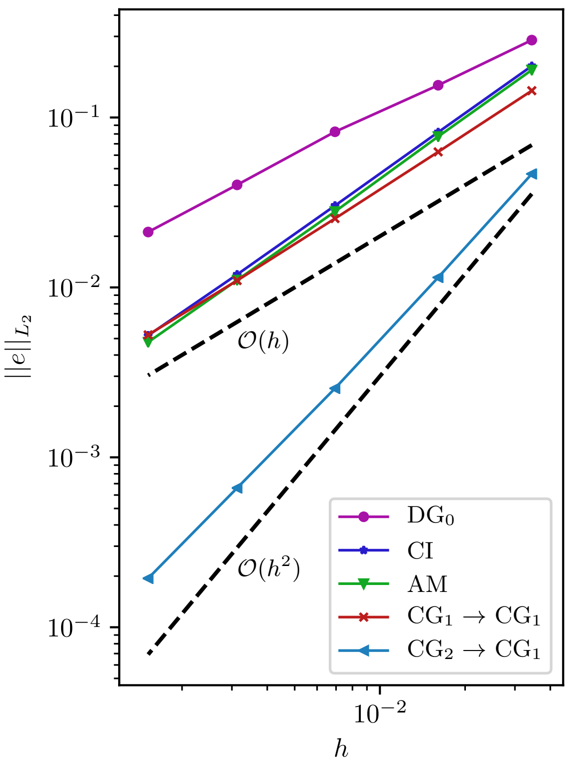


Fig. 9. The L_2 error norm versus minimum edge-length h, of the electric field computed by the methods outlined in section Sec. V-G. The mesh was generated by Gmsh (version 3.0.5) with five different resolutions on the outer sphere, $\{1/5, 1/10, 1/20, 1/40, 1/80\}$, and the inner sphere, $\{1/30, 1/60, 1/120, 1/240, 1/480\}$, resulting in the following number of tetrahedral cells: $\{7\,463, 48\,852, 340\,714, 2\,708\,950, 20\,877\,296\}$.

C. Floating Potential of Spherical Probe

Laframboise [47] 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 electron 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) consists of $37\,510$ tetrahedral 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 ρ, ϕ and $\mathbf{E}$, using the L_2-projection for $\mathbf{E}$. ϕ is solved using *Algebraic Multigrid* (AMG) preconditioned *Generalized Minimum Residual* (GMRES) linear algebra solver. The simulation is initialized with $300\,000$ simulation particles of each species (electrons and protons) using the inverse transform sampling, and in subsequent time-steps particles are injected using the optimized rejection sampling. The particles are Maxwellian distributed without drift, and the species are of equal temperature. The time step is taken to be $0.05\omega_{pe}^{-1}$ where ω_{pe} is the electron plasma frequency.

With this setup, we demonstrate good agreement with the results of Laframboise [47]. We do this in two different ways: First, we fix the potential of the probe to $2k_BT_e/e$ using a voltage source. Here, e and T_e are the elementary charge and the electron temperature, respectively. According to Laframboise this should yield a collected current of $I = -2.945I_0$ in steady-state where $I_0 = en_e\sqrt{8\pi k_BT_e/m_e}$. m_e and n_e are the electron mass and density, respectively. Fig. 10 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 $10\omega_{pe}^{-1}$ (blue) we are able to see that the result is indeed in good agreement with Laframboise. For reference, the algorithm updating the host cells of the particles (Sec. III) counted on average 0.154 cell crossings per particle per time-step for this simulation. This is indeed a very low mean recursion depth.

Second, instead 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. 11, it is evident that this is the case. Then, according to Laframboise [47], one expects the potential to settle at $2k_BT_e/e$. The simulation result in Fig. 12 agrees well with this. The methods described herein are thus capable of

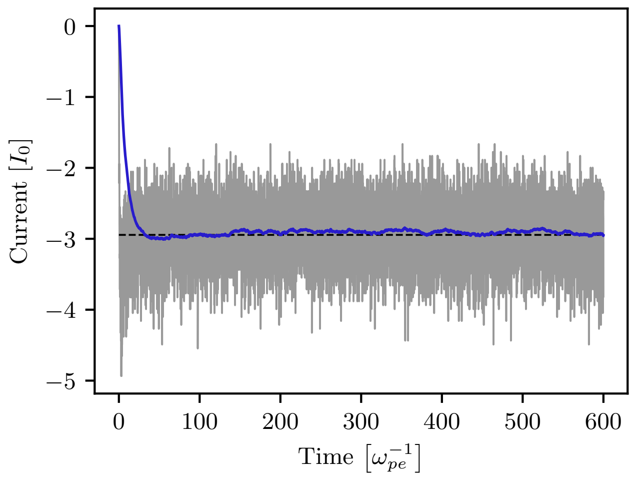


Fig. 10. Current collected by object connected to voltage source. Both raw (gray line) and smoothed data (blue line) are shown. Horizontal dashed line indicates a theoretical expectation.

producing correct kinetic results in plasma physics, including when the floating potential is self-consistently determined using charge constraints.

VIII. Discussion

The numerical experiments demonstrate the validity of the method, as well as a limitation. Namely, it is capable of reproducing kinetic results in plasma physics, but it does so at a reduced order of accuracy for simulations of objects with too sharp edges. This is necessarily also the case for other FEM-PIC codes suitable for plasma–object interaction studies. While this can be remedied using smoother objects, or one can accept the reduction of accuracy, we do propose further investigation into this topic to improve future plasma–object interaction simulations.

It is also worthwhile to revisit some established results about PIC simulations in the context of unstructured simulations with objects. Let us start with temporal stability criteria. Stability of the time-integration schemes used in the PIC cycle is often studied using the von Neumann analysis or an amplification matrix [17, pp. 96–106]. According to such analyses, the leapfrog and Boris methods commonly used to advance particle positions are stable when $\omega\Delta t < 2$ for any angular frequency ω present in the system [17, pp. 111–114]. Moreover, the RLC circuit schemes presented both in Sec. IV-C and in Verboncoeur [13] are unconditionally stable by the von Neumann analysis. These analyses, however, only considers the time-integration separately, and do not take into account that the driving (non-homogeneous) term is, in a PIC simulation, indirectly dependent on the solution at previous time-steps. This creates a feedback which may affect stability. Indeed, Birdsall *et al.* found that the leapfrog scheme became unstable due to collective effects when $\omega_{pe}\Delta t > 1.62$ [16, pp. 183–196], despite the von Neumann stability criteria being fulfilled. This derivation assumed a Maxwellian plasma. In the vicinity of strongly attracting or repelling objects, however, the velocity distribution is usually far from Maxwellian. The authors are not aware of any rigorously obtained criteria that guarantee stability in the presence of objects.

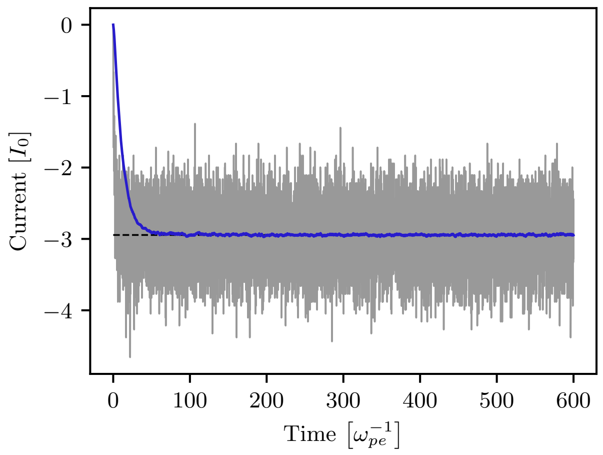


Fig. 11. Current collected by object connected to current source. Both raw (gray line) and smoothed data (blue line) are shown. Horizontal dashed line indicates a theoretical expectation.

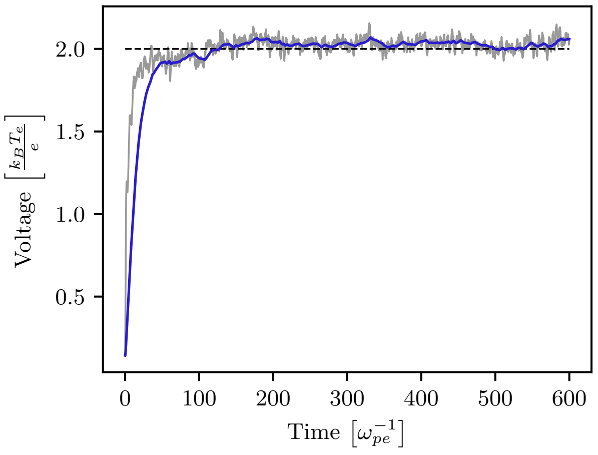


Fig. 12. Potential of object connected to current source. Both raw (gray line) and smoothed data (blue line) are shown. Horizontal dashed line indicates a theoretical expectation.

Let us also consider effects due to the mesh. Birdsall *et al.* derived the dispersion relation for a *simulation plasma*, which incorporates a structured mesh. It turns out that the mesh causes coupling between different wave modes (aliasing), which results in a strong increase in kinetic energy (numerical heating) when the mesh do not properly resolve the Debye length [17, pp. 231–241], [16, pp. 175–181]. The heating causes the Debye length to increase, until it is resolved by the mesh, at which point the heating more or less stops. This effect is also verified by others [48], [49], [50]. Accordingly, these references suggest that the spatial step-size should be no more than roughly 3 Debye lengths to avoid mesh instabilities, the exact figure depending on the reference and the case considered. We emphasize that all of these studies assumed a uniform, structured, rectangular mesh and not an unstructured one. These results should therefore not be applied naïvely to the methods presented herein. Furthermore, the studies assumed cold or Maxwellian plasmas. As already stated, the plasma is non-Maxwellian close to objects.

To summarize, neither spatial or temporal criteria previously established are, strictly speaking, applicable. This does not necessarily mean that FEM-PIC simulations of plasma–object interactions should be avoided, but that care should be exercised to verify that the simulations produce physically meaningful results. As a starting point, we recommend cell diameters of roughly 1–2 electron Debye lengths, or a fifth of the radius of curvature of the geometry, whichever is smaller. As for the time-step, we recommend it to be small enough that particle orbits (on average) have comparable spatial resolution as the mesh, taking into account that particles may be accelerated near objects. It should also be small compared to the plasma period, say, a tenth. The simulations in Sec. VII-C are along these guidelines, and have sound physical results. Finally, we remark that even when the Debye length *is* sufficiently resolved, randomly directed errors in the electric field causes a random walk process, which means that *some* numerical heating always occurs [17, pp. 316–318].

IX. Conclusion

We have created a finite element based particle-in-cell method for plasma–object interaction simulations. This method supports an arbitrary number of objects that can be connected in arbitrary circuit topologies with voltage sources, current sources, resistors, inductors and capacitors. The algorithm automatically formulates mathematical constraints and embeds these into the finite element stiffness matrix. In addition, this new method does not rely on decomposing the Poisson equation, which would require solving the Poisson equation twice per time-step, or storing large amounts of field data in case of many objects.

The methods are implemented in PUNC. Whereas for instance SPIS is a mature, high performance program with a graphical user interface, PUNC is an experimental code, intended for rapid prototyping and research on methods. Being implemented using Python and FEniCS, it comes with great flexibility, which is apparent from the wide range of solvers, preconditioners, etc. considered herein. In particular a comparison of methods for obtaining the electric field revealed that the arithmetic mean method is as accurate as more costly methods, if fist-order spatial accuracy is deemed sufficient. The method is also agnostic of the polynomial order of the finite elements, although we note that FEM-PIC codes in general suffer from reduced order of accuracy when coarse objects are present. Future efforts to improve this are recommended.

X. Acknowledgment

We acknowledge support from the 4DSpace Strategic Research Initiative at the University of Oslo, and the Natural Sciences and Engineering Research Council of Canada. Diako Darian gratefully thanks Miroslav Kuchta for his help with general finite element theory and details of FEniCS implementations throughout the course of this work.

References

[1] W. J. Miloch, "Wake effects and mach cones behind objects," *Plasma Physics and Controlled Fusion*, vol. 52, no. 12, p. 124004, nov 2010. [Online]. Available: https://doi.org/10.1088/0741-3335/52/12/124004

[2] Y. Miyake and H. Usui, "New electromagnetic particle simulation code for the analysis of spacecraft-plasma interactions," *Physics of Plasmas*, vol. 16, no. 6, p. 062904, 2009. [Online]. Available: https://doi.org/10.1063/1.3147922

[3] 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*, vol. 36, no. 5, pp. 2336–2349, Oct 2008.

[4] V. Vahedi and G. DiPeso, "Simultaneous potential and circuit solution for two-dimensional bounded plasma simulation codes," *Journal of Computational Physics*, vol. 131, no. 1, pp. 149–163, 1997.

[5] 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*, vol. 34, no. 5, pp. 2084–2093, Oct 2006.

[6] 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*, vol. 41, no. 12, pp. 3577–3587, Dec 2013.

[7] 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*, vol. 36, no. 5, pp. 2360–2368, Oct 2008.

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

[9] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009. [Online]. Available: http://dx.doi.org/10.1002/nme.2579

[10] M. Rapp and I. Strelnikova, "Measurements of meteor smoke particles during the ecoma-2006 campaign: 1. particle detection by active photoionization," *Journal of Atmospheric and Solar-Terrestrial Physics*, vol. 71, no. 3, pp. 477 – 485, 2009, global Perspectives on the Aeronomy of the Summer Mesopause Region.

[11] T. A. Bekkeng, E. S. Helgeby, A. Pedersen, E. Trondsen, T. Lindem, and J. I. Moen, "Multi-needle langmuir probe system for electron density measurements and active spacecraft potential control on cubesats," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–1, 2019.

[12] T. A. Bekkeng, A. Barjatya, U.-P. Hoppe, A. Pedersen, J. I. Moen, M. Friedrich, and M. Rapp, "Payload charging events in the mesosphere and their impact on langmuir type electric probes," *Annales Geophysicae*, vol. 31, pp. 187–196, 02 2013.

[13] J. Verboncoeur, M. Alves, V. Vahedi, and C. Birdsall, "Simultaneous potential and circuit solution for 1d bounded plasma particle simulation codes," *Journal of Computational Physics*, vol. 104, no. 2, pp. 321–328, 1993.

[14] A. Logg, K.-A. Mardal, G. N. Wells *et al.*, *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[15] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells, "The fenics project version 1.5," *Archive of Numerical Software*, vol. 3, no. 100, 2015.

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

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

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

[19] G. Lapenta, "Particle simulations of space weather," *Journal of Computational Physics*, 2012.

[20] J. P. Verboncoeur, "Particle simulation of plasmas: review and advances," *Plasma Physics and Controlled Fusion*, vol. 47, no. 5A, pp. A231–A260, apr 2005. [Online]. Available: https://doi.org/10.1088/0741-3335/47/5a/017

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

[22] H. L. Pécseli, *Waves and Oscillations in Plasmas*. CRC Press, 2012.

[23] ——, *Fluctuations in Physical Systems*, 1st ed. Cambridge University Press, 8 2000. [Online]. Available: http://amazon.com/o/ASIN/0521655927/

[24] H. Qin, S. Zhang, J. Xiao, J. Liu, Y. Sun, and W. M. Tang, "Why is boris algorithm so good?" *Physics of Plasmas*, vol. 20, no. 8, p. 084503, 2013. [Online]. Available: https://doi.org/10.1063/1.4818428

[25] E. Süli and D. F. Mayers, *An Introduction to Numerical Analysis*, 1st ed. Cambridge University Press, 9 2003.

[26] J. E. Gentle, "Transformations of uniform deviates: General methods," in *Random number generation and Monte Carlo methods*. Springer Science & Business Media, 2013, ch. 4, pp. 101–109.
[27] K. Cartwright, J. Verboncoeur, and C. Birdsall, "Loading and injection of maxwellian distributions in particle simulations," *Journal of Computational Physics*, vol. 162, no. 2, pp. 483–513, 2000.
[28] R. Marchand and P. A. R. Lira, "Kinetic simulation of spacecraft–environment interaction," *IEEE Transactions on Plasma Science*, vol. 45, no. 4, pp. 535–554, 2017.
[29] V. 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*, vol. 73, no. 9, pp. 2839–2884, 1968.
[30] A. Lui and S. Krimigis, "Energetic ion beam in the earth's magnetotail lobe," *Geophysical Research Letters*, vol. 10, no. 1, pp. 13–16, 1983.
[31] J. E. Gentle, "Transformations of uniform deviates: General methods," in *Random number generation and Monte Carlo methods*. Springer Science & Business Media, 2013, ch. 4, pp. 113–124.
[32] W. R. Gilks and P. Wild, "Adaptive rejection sampling for gibbs sampling," *Applied Statistics*, pp. 337–348, 1992.
[33] D. Darian, W. J. Miloch, M. Mortensen, Y. Miyake, and H. Usui, "Numerical simulations of a dust grain in a flowing magnetized plasma," *Physics of Plasmas*, vol. 26, no. 4, 2019.
[34] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 4, pp. 807–832, 2002.
[35] H.-C. Kim, Y. Feng, and J. P. Verboncoeur, "Algorithms for accurate collection, ejection, and loading in particle simulations," *Journal of Computational Physics*, vol. 223, no. 2, pp. 629–642, 2007.
[36] D. K. Cheng, *Field and Wave Electromagnetics*, 2nd ed. Addison-Wesley, 1989.
[37] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, 2nd ed. Society for Industrial and Applied Mathematics, 2004.
[38] E. Kreyszig, *Advanced Engineering Mathematics*, 9th ed. John Wiley & Sons, Inc., 2006.
[39] J. W. Nilsson and S. A. Riedel, *Electric Circuits*, 7th ed. Pearson Prentice Hall, 2005.
[40] A. Quarteroni, *Numerical models for differential problems*, 3rd ed. Springer, 2017.
[41] P. Clément, "Approximation by finite element functions using local regularization," *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, vol. 9, no. R2, pp. 77–84, 1975.
[42] Y. N. Grigoryev, V. A. Vshivkov, and M. P. Fedoruk, *Numerical" particle-in-cell" Methods: Theory and Applications*. Walter de Gruyter, 2002.
[43] J. Verboncoeur, "Symmetric spline weighting for charge and current density in particle simulation," *Journal of Computational Physics*, vol. 174, no. 1, pp. 421–427, 2001.
[44] M. I. Bhatti and P. Bracken, "Solutions of differential equations in a bernstein polynomial basis," *Journal of Computational and Applied Mathematics*, vol. 205, no. 1, pp. 272–280, 2007.
[45] P. Frey and P. L. George, *Mesh Generation: Application to Finite Elements*, 2nd ed. Wiley-ISTE, 4 2008.
[46] D. J. Griffiths, "Electrostatics," in *Introduction to electrodynamics*, 3rd ed. Prentice Hall, 1999, ch. 2, p. 105.
[47] J. G. Laframboise, "Theory of spherical and cylindrical langmuir probes in a collisionless, maxwellian plasma at rest," Ph.D. dissertation, University of Toronto, 1966.
[48] H. Okuda, "Nonphysical noises and instabilities in plasma simulation due to a spatial grid," *Journal of Computational Physics*, vol. 10, no. 3, pp. 475 – 486, 1972.
[49] C. K. Birdsall and N. Maron, "Plasma self-heating and saturation due to numerical instabilities," *Journal of Computational Physics*, vol. 36, no. 1, pp. 1 – 19, 1980.
[50] L. Chen, A. B. Langdon, and C. Birdsall, "Reduction of the grid effects in simulation plasmas," *Journal of Computational Physics*, vol. 14, no. 2, pp. 200 – 222, 1974.

Sigvald Marholm was born in Norway 1987. He received an MSc in electrical engineering at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway in 2012, and is currently pursuing a PhD in computational plasma physics at the University of Oslo, Norway.

Diako Darian is a PhD candidate in Mechanics at the Department of Mathematics, University of Oslo, Norway. His research interests include scientific computing, ionospheric plasma, dusty plasmas, and object-plasma interactions.

Mikael Mortensen is currently working as a professor at the Department of Mathematics, University of Oslo, where he is performing research into several aspects of turbulence and Computational Fluid Dynamics.

Richard Marchand is currently a Professor of Physics with the University of Alberta, Edmonton, AB, Canada, where he carries out research in computational physics, space physics, and spacecraft–environment interaction.

Wojciech J. Miloch received the M.Sc. degree in space and plasma physics and the Ph.D. degree from the University of Oslo, Norway, in 2006 and 2009, respectively. He is currently a professor and head of 4DSpace Strategic Research Inititative at the Department of Physics, University of Oslo. His research interests include space and astrophysical plasmas, space weather, plasma interactions with finite sized objects, such as spacecrafts, probes, or dust grains, complex plasmas and numerical modeling.