INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 14, Number 2, Pages 175–200

AN IMMERSED-FINITE-ELEMENT PARTICLE-IN-CELL SIMULATION TOOL FOR PLASMA SURFACE INTERACTION

YUCHUAN CHU, DAORU HAN, YONG CAO, XIAOMING HE, AND JOSEPH WANG

Abstract. A novel Immersed-Finite-Element Particle-in-Cell (IFE-PIC) simulation tool is presented in this paper for plasma surface interaction where charged plasma particles are represented by a number of simulation particles. The Particle-in-Cell (PIC) method is one of the major particle models for plasma simulation, which utilizes a huge number of simulation particles and hence provides a first-principle-based kinetic description of particle trajectories and field quantities. The immersed finite element method provides an accurate approach with convenient implementations to solve interface problems based on structured interface-independent meshes on which the PIC method works most efficiently. In the presented IFE-PIC simulation tool, different geometries can be treated automatically for both PIC and IFE through the geometric information specified in an input file. The set of parameters for plasma properties is also assembled into a single input file which can be easily modified for a variety of plasma environments in different applications. Collisions between particles are also incorporated in this tool and can be switched on/off with one parameter in the input file. Efficient modules are adopted to integrate PIC and IFE together into the final simulation tool. Hence our IFE-PIC simulation package offers a convenient and efficient tool to study the microcosmic plasma features for a wide range of applications. Numerical experiments are provided to demonstrate the capability of this tool.

Key words. Particle-In-Cell, immersed finite elements, plasma surface interaction, electric propulsion.

1. Introduction

The problem of plasma interactions with a complex surface is typically very complicated to solve. Numerical simulations play an essential role in the study of plasma characteristics and have become an efficient tool to provide accurate performance predictions of the plasma devices. The standard tool to solve the interactions of a rarefied plasma with a complex surface is Particle-in-Cell (PIC) method, which has been widely used in many engineering and applied physics problems involving plasma-surface interaction, such as electric propulsion systems [41, 57, 73] and fusion reactors [20, 62]. Furthermore, the PIC method with Monte Carlo Collisions (MCC) [7, 36, 60, 64] and Direct Simulation Monte Carlo (DSMC) methods [3–5, 9, 33, 34, 58, 59] can simulate the interactions between different plasma species.

In the PIC method, a "gather" step is used to interpolate the electric field, which is solved by a numerical method, from mesh nodes to particle positions to push the particles. A "scatter" step is used to deposit particle charge to mesh nodes for the numerical method to solve for the electric field. In a typical PIC simulation, there are millions of simulation particles in the computation domain. Hence these two main steps cost a significant portion of computing time. Therefore, it is necessary to find an efficient way to perform these two steps for each particle. Since the particle locations can be easily identified via indexing in a Cartesian mesh, it is preferable to use a Cartesian mesh on which the PIC method works most efficiently. However, most PIC simulations in practical applications have

Received by the editors February 25, 2016 and, in revised form, October 8, 2016.
2010 Mathematics Subject Classification: 68N19, 65N30.

objects immersed in the plasma, which makes the problem an *interface* problem. For such problems, traditional numerical methods need to use body-fitting meshes, which are unstructured for non-trivial object geometry hence not very efficient for locating particles in the PIC method. Therefore, it is critical to develop a new numerical method to accurately solve the interface problem with complex interfaces on Cartesian meshes for PIC method.

The immersed finite element (IFE) method which adapts the finite element method for interface problems so that a structured mesh independent of the interface can be used to accurately solve interface problems [1, 8, 11–13, 16, 18, 22, 27, 28, 31,32,35,43,44,46–50,52,53,56,63]. Based on the IFE method, the Immersed-Finite-Element Particle-In-Cell (IFE-PIC) method [37, 39, 40, 54, 55] has been developed to provide a promising approach for plasma simulations and applied to different applications [10, 15, 17, 36, 66, 67, 69, 70]. The algebraic system arising from the IFE method is symmetric positive definite, which is a critical property for employing many fast solvers. While minimizing the extra efforts to modify the traditional finite element packages, IFE methods can also easily deal with complex interface with an optimal order of accuracy. These features make the IFE methods competitive for accurately solving the interface problems on structured meshes independent of the interface and providing the electric field to the PIC method.

In this paper we will present a featured simulation tool of the IFE-PIC method. The IFE and PIC methods will be briefly reviewed and the dynamic interactions between IFE and PIC modules are illustrated in details. Several critical features of the simulation tool will be discussed, such as the automatic treatment of the different geometries, the convenient input for different types of plasma, efficient modules for interactions between IFE and PIC methods, and optional modules for particle collisions. In order to illustrate the features of this simulation tool, two numerical experiments are carried out for the electric propulsion. Based on the automatic treatment of object surfaces and high-efficient particle simulation of our tool, various spacecraft/satellite parts of different geometries and high density plasma in electric propulsion can be modeled with our IFE-PIC package. Meanwhile, the physical mechanism of discharging process in electric propulsion and the erosion of key components, such as the grid erosion caused by charge-exchange ions, can be analyzed in detail with the particle collision module.

The rest of the paper is organized as follows. In Section 2 we will review the IFE method and discuss about the modules for obtaining the IFE solution. In Section 3 we will review the PIC method and introduce the modules and parameters for PIC. In Section 4 we will integrate PIC and IFE together under a dynamic framework with efficient modules for the interactions between PIC and IFE. In Section 5 numerical examples will be provided to illustrate the features of the simulation tool. Finally brief conclusions are given in Section 6.

2. Modules for immersed finite element method

In this section, we will introduce the modules of the immersed finite element method in the plasma environment simulation tool. First, we will briefly review the basic idea of IFE and the definition of the 3D linear IFE method. Then we will introduce the input file and the modules for dealing with the geometries which are needed to form the IFE basis functions and compute the integrals by Gauss quadratures. Moreover, we will introduce the modules to form the linear system arising from the immersed finite element method based on the charge density provided by

PIC and the modules to solve the linear system for the electrostatic potential field which is needed by PIC.

2.1. Review for immersed finite element. The immersed finite element (IFE) method is an extended finite element method that can use a structured mesh to solve a partial differential equation (PDE) with discontinuities in the coefficients. The structured mesh is independent of the interface and allows complex interfaces to cut through the interior of the elements. Therefore, the mesh in an IFE method consists of interface elements whose interiors are cut through by the interface and the rest that are called non-interface elements. For all non-interface elements, standard finite element functions are used, while for interface elements, special piecewise polynomials satisfying interface jump conditions are employed. Therefore, the only difference between IFE and traditional finite elements is the choice of different basis functions on the interface elements. This leads to a convenient implementation of the IFE method based on the traditional finite element method. Furthermore, the IFE method has the optimal convergence rates expected from the utilized polynomials and provides a symmetric positive definite matrix for fast solvers [2, 14, 19, 26, 30, 42, 45, 51, 74]. In the following we will briefly recall the 3D linear IFE method [37, 38].

Consider the following 3D interface elliptic PDE for solving the electrostatic potential:

$$-\nabla \cdot \big(\varepsilon \nabla \phi(X)\big) = \rho(X),\ X = (x, y, z) \in \Omega, \tag{1}$$

together with the jump conditions across the interface Γ:

$$[\phi(X)]|_{\Gamma} = 0, \tag{2}$$

$$\left[\varepsilon \frac{\partial \phi(X)}{\partial \mathbf{n}_{\Gamma}}\right]\Big|_{\Gamma} = 0, \tag{3}$$

and the boundary conditions:

$$\phi(X)|_{\Gamma_D} = g(X), \tag{4}$$

$$\frac{\partial \phi(X)}{\partial \mathbf{n}_{\Gamma_N}}|_{\Gamma_N} = p(X). \tag{5}$$

Here, without loss of generality, we assume that $\Omega \subset R^3$ is a box domain, the interface Γ is a curved surface separating Ω into two sub-domains Ω^-, Ω^+ such that $\overline{\Omega} = \overline{\Omega^-} \cup \overline{\Omega^+} \cup \Gamma$, Γ_D and Γ_N are the Dirichlet and Neumann boundaries such that $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\mathbf{n}_{\Gamma_N}$ is the unit outer normal vector of Γ_N, $\mathbf{n}_{\Gamma}$ is the unit normal vector of Γ pointing from Ω^- to Ω^+, and the material-dependent coefficient $\varepsilon(x, y)$ is a piecewise constant function defined by

$$\varepsilon(X) = \begin{cases} \varepsilon^-,\ X \in \Omega^-, \\ \varepsilon^+,\ X \in \Omega^+. \end{cases}$$

In the mesh generation module, we first partition the simulation domain into uniform cubes whose edges are parallel to the coordinate axes, and then further partition each cube into five tetrahedra in the way such that the vertices of the tetrahedra are also those of the cubes, as illustrated in Figure 1.

All the parameters for the mesh generation are provided in a single input file `mesh.inp`. After these parameters are read into the code from the input file, the module *Basic_Tetra_Partition*, which calls the module *Cubic_Partition* for the partition of the domain into uniform cubes, is called to obtain the basic information array for all tetrahedron elements.

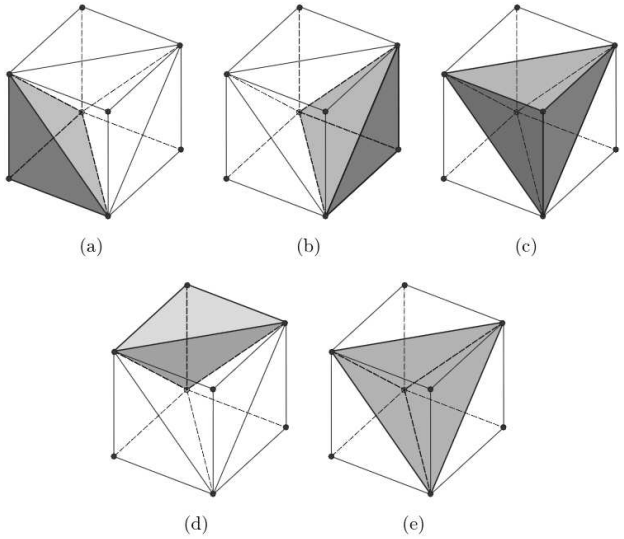


FIGURE 1. The five tetrahedra partitioned from a cubic cell [37].

In a typical interface tetrahedron T with vertices A_i, $(i = 1, 2, 3, 4)$, we use a three-point cut $\triangle P_1P_2P_3$ to approximate the interface Γ in this element where P_j $(j = 1, 2, 3)$ are three intersection points between the element edges and the interface Γ. Then the three-point cut $\triangle P_1P_2P_3$ divides T into two sub-elements T^+ and T^-. Then four piecewise 3D linear immersed finite element basis functions can be introduced [37, 38]:

$$\psi_i(x,y) \quad = \quad \begin{cases} \psi_i^+ = a_1x + a_2y + a_3z + a_4, (x,y) \in T^+ \\ \psi_i^- = a_5x + a_6y + a_7z + a_8, (x,y) \in T^- \end{cases} \quad (i \ = \ 1, 2, 3, 4),$$

which satisfy the following constraints:

(1) Nodal value specifications:

$$\psi_i(A_j) \quad = \quad \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \quad (i, j \ = \ 1, 2, 3, 4);$$

(2) The continuity across the three-point cut $\triangle P_1P_2P_3$:

$$\psi_i^+(P_j) = \psi_i^-(P_j), \ i = 1, 2, 3, 4 \quad \text{and} \quad j = 1, 2, 3;$$

(3) The flux continuity across three point cut $\triangle P_1P_2P_3$:

$$(\beta^+ \frac{\partial \psi_i^+}{\partial \mathbf{n}} - \beta^- \frac{\partial \psi_i^-}{\partial \mathbf{n}}) = 0, \ i = 1, 2, 3, 4$$

where $\mathbf{n}$ is the normal of $\tilde{\Gamma}_T$.

For more details about the computation of the coefficients in the basis functions and the formulation of the immersed finite element method, we refer the readers to [37, 38].

2.2. Object data structure. In the IFE-PIC simulation tool, two general data structures are utilized in the object input file `object.inp` to define different types of objects. One general data structure is to use up to 9 parameters to describe the properties of an object. The other one is to define an algebraic equation $z = z(x, y)$ for describing the surface of an object.

The first data structure defines 9 parameters to provide a general way to describe different types of objects [37]. The first parameter, $objects(i)\%Shape$, is an positive integer to identify the type of an object. For example, 1 indicates a cylinder, 2 indicates a sphere or spherical cap, 3 indicates a box, and so on. The second parameter $objects(i)\%Axis$ is an integer to specify the orientation axis of an object. In particular, 0 indicates that there is no need to specify axis; 1, 2, and 3 indicate that the axis is parallel to the coordinate axes x, y, or z, respectively; 4, 5, and 6 indicate that the axis is perpendicular to the axes x, y, or z, respectively. The parameter set $objects(i)\%Dimensions$ defines the size of an object. For example, this parameter set contains the length, width and height for a box. The parameter set $objects(i)\%Locations$ is used to determine the location of a reference point of an object. For example, it is the center point for a sphere. Up to three reference points can be specified for one object. The parameter set $objects(i)\%Region$ provides the index numbers of the regions in and outside of an object. The parameter $objects(i)\%Phi$ represents the reference potential of an object if applicable. The parameter $objects(i)\%Eps$ provides the dielectric coefficient of an object.

Based on the designed object data structure, we have established a shape library for most frequently used objects such as boxes, circular cylinders, circular truncated cones, spheres, spherical caps, thin plates, circular plates, and so on. Meanwhile, according to the variable $objects(i)\%Axis$, a series of objects, which have the same shape but are parallel or perpendicular to the coordinate axes x, y,or z, can be specified separately. In addition, a number of complex geometries can be constructed by applying Boolean operations to the objects which are defined in the shape library. In the following we will provide the information of several typical examples but not all types of the objects in order to shorten the presentation of the paper.

Data 1 presents the object data information for a $22 \times 22 \times 22$ box with the low and high corners located at point $(46, -11, 30)$ and $(68, 11, 52)$, respectively. The mesh nodes located in this cubic box are marked to be -2 and other mesh nodes outside of the cubic box are marked to be -1 to represent vacuum region. The potential and dielectric coefficient for this box are set to be -2 and 1×10^6, respectively. In addition, all the input parameters for our tool are normalized by reference variables which can be found in [37].

Data 1. *Object information for a cubic box*

```
3              !   objects(i)%Shape : Cube Box
0              !   objects(i)%Axis : NA
22., 22., 22.  !   objects(i)%Dimensions(1:3) : Length, Width, Height
46., -11., 30. !   objects(i)%Locations(1,:) : Lower Corner
68., 11., 52.  !   objects(i)%Locations(2,:) : Upper Corner
0., 0., 0.     !   objects(i)%Locations(3,:) : Dummy, Dummy, Dummy
-2, -1         !   objects(i)%Regions(1:2) : Inside the Cube Box, Outside the Cube
                   Box:Vacuum
-2.            !   objects(i)%Phi : Potential of the Cube Box
1.E6           !   objects(i)%Eps : Dielectric Coefficient of the Cube Box
```

The object data information in Data 2 is designed to model a circular-truncated cone into the simulation domain. The axis of the circular-truncated cone is parallel to the x-axis and the radius for the bottom and top circle surfaces of the circular-truncated cone are set to be 3.54 and 5.26, respectively. The coordinates of the center of the two circle surfaces are $(32, 0, 45)$ and $(40, 0, 45)$.

Data 2. *Object information for a circular-truncated cone with axis parallel to x axis*

4	!	objects(i)%Shape : Circular-Truncated Cone
1	!	objects(i)%Axis : Axis Parallel to x Axis
3.54, 5.26, 0.	!	objects(i)%Dimensions(1:3) : Radius, Radius, Dummy
32., 0., 45.	!	objects(i)%Locations(1,:) : Circle Center for Bottom Surface
40., 0., 45.	!	objects(i)%Locations(2,:) : Circle Center for Top Surface
0., 0., 0.	!	objects(i)%Locations(3,:) : Dummy, Dummy, Dummy
-5, -1	!	objects(i)%Regions(1:2) : Inside and Outside the Circular -Truncated Cone
-2.	!	objects(i)%Phi : Potential of the Circular Truncated Cone
1.E6	!	objects(i)%Eps : Dielectric Coefficient of the Circular Truncated Cone

Similarly, we can define a spherical cap in Data 3.

Data 3. *Object information for a spherical cap*

2	!	objects(i)%Shape : Spherical Cap
1	!	objects(i)%Axis : Sphere Cut by the Plane Perpendicular to x Axis
16.64, 0., 0.	!	objects(i)%Dimensions(1:3) : Radius, Dummy, Dummy
28., 0., 33.08	!	objects(i)%Locations(1,:) : Center for Sphere
39., 0., 0.	!	objects(i)%Locations(2,:) : x Coordinate of Cutting-Plane, Dummy, Dummy
0., 0., 0.	!	objects(i)%Locations(3,:) : Dummy, Dummy, Dummy
-4, -1	!	objects(i)%Regions(1:2) : Inside Spherical Cap, Outside Spherical Cap: Vacuum
-2.	!	objects(i)%Phi : Potential of Spherical Cap
1.E6	!	objects(i)%Eps : Dielectric Coefficient of Spherical Cap

On the other hand, a curved surface with arbitrary geometry (referred to as "z-surface" in the following of this paper) is also defined in the object shape library. The shape index of the new object is 2013. This object index will lead to the module *GetLocalHeight_zSurface* which specifies the geometry of the surface via a given math equation described as $z = z(x, y)$. Then all the geometric operations will be based on this equation, such as clarifying the location of mesh nodes and computing the intersection points between the object surface and the mesh lines. The regions, potential, and dielectric coefficient parameters can be passed into the IFE modules in the same way as other objects.

The object shape library, which utilizes both of the above two general data structures and the Boolean operations, contains both a series of basic shapes and the options to define customized geometries. Hence this simulation tool is capable of modeling most shapes appeared in the applications of plasma dynamics to astronautical and space studies. For instance, a complex-shaped spacecraft may be

assembled from the basic shapes that represent each key component of the spacecraft. On the other hand, a curved surface can be specifically and flexibly modeled by the object of "z-surface" to represent the rugged/irregular surface topography. Furthermore, complicated geometries such as the ion optics grids of an ion thruster can be constructed by boxes and cylinders with appropriate Boolean operations.

2.3. Modules for object geometry and IFE basis functions. As reviewed in Section 2.1, all the mesh elements need to be classified to be either interface elements or non-interface elements in order to define appropriate basis functions on each element. Therefore, the mesh nodes are first classified to be either in a specific object or out of all objects. Then this classification of the mesh nodes is used to classify the elements and locate the intersection points between the Cartesian mesh lines and the object surfaces, which provides the critical interface information to define the IFE basis functions as discussed above.

The classification of mesh nodes is done via the module *Locate_Nodes* which calls the module *Locate_Point*. For each mesh node, the module *Locate_Point* determines in which element this point is located. In *Locate_Nodes*, all the mesh nodes are classified by *Locate_Point* one by one and then assigned a flag indicating which object/region it belongs to.

For an object defined by the algebraic equation $z = z(x, y)$, a point is inside the object ("z-surface") if

$$z_p < z(x_p, y_p) \tag{6}$$

where x_p, y_p, and z_p are the coordinates of the given point while $z(x, y)$ is the function specifying the z-surface topography.

For an object defined by the general data structure using up to 9 parameters, the algorithm in the module *Locate_Point* adopts the fundamental vector geometry operations, such as the dot product and cross product of vectors, to decide whether a mesh point locates in the object.

After the mesh nodes are classified as above, each element will be classified as either an "interface element" or a "non-interface element" in the module *Classify_Elements*. To do this, one would need the location information of all four vertices of the tetrahedron element. The algorithm of classifying elements and calculating the intersection information is summarized as follows:

- Step 1: If all four vertices of the element belongs to the same object or they do not belong to any object, then this element is a non-interface element, done; otherwise, this element is an interface element, then go to Step 2;
- Step 2: For each edge of an element, calculate the possible intersection with the objects by performing the "line-object intersection calculation". This will determine if any edge of the element has any intersection with any object and then compute the coordinates of the intersection point if there is one. Summarizing such information of all six edges of an element will determine the intersection type of the element: either a three-edge-cut interface element or a four-edge-cut interface element (see Figure 2) [37,38].

The intersection points are searched in an efficient way such that each object is only checked against all the tetrahedron elements that lie in its vicinity, which is pre-specified such that all elements in the vicinity are enclosed by a virtual box. Basically, for every object in the simulation, a minimum box region is defined to contain this object. Then each element will be checked for whether it lies in the minimum box. If yes, the module *El_Object_Intersection* will be called to obtain the possible intersection points of this element with the object, inside which the

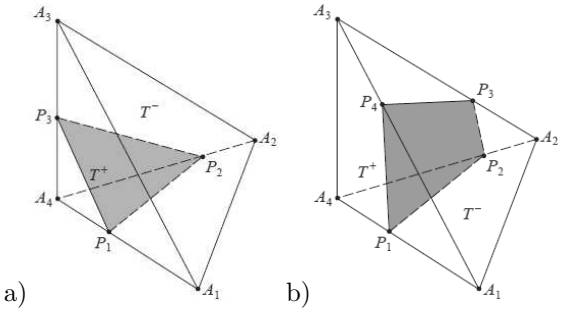


FIGURE 2. Intersection topologies of tetrahedral elements: a) Three-edge-cut b)Four-edge-cut [37].

module *Line_Object_Intersection* is called to obtain the intersection point for each edge of the element.

Once an element is classified (a non-interface element or an interface element) and the intersection points are obtained, the corresponding basis functions will be defined in the module *IFE_Assembler* based on the definition in Section 2.1. For example, for an interface element where vertex A_1 is in T^+ and vertices A_2, A_3, and A_4 are in T^-, the definition of the 3D linear IFE basis functions in Section 2.1 leads to the following eight linear equations for ψ_1:

$$a_1x_1 + a_2y_1 + a_3z_1 + a_4 = 1 \tag{7}$$

$$a_5x_2 + a_6y_2 + a_7z_2 + a_8 = 0 \tag{8}$$

$$a_5x_3 + a_6y_3 + a_7z_3 + a_8 = 0 \tag{9}$$

$$a_5x_4 + a_6y_4 + a_7z_4 + a_8 = 0 \tag{10}$$

$$a_1x_{p1} + a_2y_{p1} + a_3z_{p1} + a_4 = a_5x_{p1} + a_6y_{p1} + a_7z_{p1} + a_8 \tag{11}$$

$$a_1x_{p2} + a_2y_{p2} + a_3z_{p2} + a_4 = a_5x_{p2} + a_6y_{p2} + a_7z_{p2} + a_8 \tag{12}$$

$$a_1x_{p3} + a_2y_{p3} + a_3z_{p3} + a_4 = a_5x_{p3} + a_6y_{p3} + a_7z_{p3} + a_8 \tag{13}$$

$$\beta^+(n_1a_1 + n_2a_2 + n_3a_3) = \beta^-(n_1a_5 + n_2a_6 + n_3a_7) \tag{14}$$

where $(x_i,\ y_i,\ z_i)$ $(i = 1,\ 2,\ 3,\ 4)$ are coordinates of vertices A_i $(i = 1,\ 2,\ 3,\ 4)$, (x_{pj}, y_{pj}, z_{pj}) $(j = 1, 2, 3)$ are coordinates of intersection points P_j $(j = 1, 2, 3)$, and (n_1, n_2, n_3) is the normal vector $\mathbf{n}$.

Inside the module *IFE_Assembler*, for an interface element, the module *Linear_IFE_Basis_Coeff* inputs and solves these linear equations for all IFE basis functions in a general way to obtain the coefficients of the IFE basis functions. Then the module *Linear_IFE_Basis_Eval* use these coefficients to evaluate the IFE basis functions at the given points, which will be the Gauss points in the module to assemble the stiffness matrix and right hand side vector. Similarly, for a non-interface element, a module *Linear_FE_Basis_Coeff* is called to form the standard finite element basis functions, which are used in the module *Linear_FE_Basis_Eval* for the evaluations of the basis functions at give points.

2.4. Modules for immersed finite element solution. In this section, we will discuss the modules to form the linear system arising from the immersed finite element method based on the charge density provided by PIC and the module to solve the linear system for the electrostatic potential field which is needed by PIC. We mainly follow the regular local assembly procedure of the regular finite element

methods to form the stiffness matrix and the right hand side vector in the linear system. The only major difference is that we use the 3D linear IFE basis functions on the interface elements while we still use the traditional 3D linear finite elements on the non-interface elements, which makes it convenient to implement the IFE method. We apply the preconditioned conjugate gradient (PCG) method to solve the linear system since the stiffness matrix is symmetric and positive-definite.

In order to reduce the computational overhead when forming the global stiffness matrix A in the linear system arising from the finite element discretization, the computation of the integrals of the local stiffness matrix and local load vector on the non-interface elements are hardwired since the standard finite element basis functions are used. Basically the integrals are computed by Gauss quadratures only once on one non-interface element and then re-scaled to other non-interface elements. When the mesh is fine enough, most of the elements are in fact non-interface elements. Therefore, when forming the global stiffness matrix A in the module *Global_Stiff*, most of the computations are the same as those of the traditional finite element method.

For the local stiffness matrix on the interface elements, the corresponding integrals are computed one by one based on the Gauss quadrature and the definition of the IFE basis functions. Each interface element is divided into two sub-elements in the piecewise definition of the IFE basis functions. Due to the regularity requirement of the Gauss quadrature, an integral needs to be computed on the two sub-domains separately and then summed up over the whole interface element. Since the shapes of some sub-elements are not very regular, an interface element is further partitioned into a number of sub-tetrahedra for the convenience of the implementation of the Gauss quadrature. For a three-edge-cut interface element and a four-edge-cut interface element, the element is partitioned into four and six sub-tetrahedra respectively, shown in Figure 3. Based on the Gauss quadrature on a tetrahedron provided by the module *Gauss_Nodes* and the coefficients of the basis function provided by the module *Linear_IFE_Basis_Coeff*, the integrals will be numerically computed on each sub-tetrahedron and then summed up over the whole interface element. The module *IFE_Stiff* is called by module *Global_Stiff* to compute the integrals in this way by using the module *Linear_IFE_Basis_Eval* and then form the local stiffness matrix for each element.

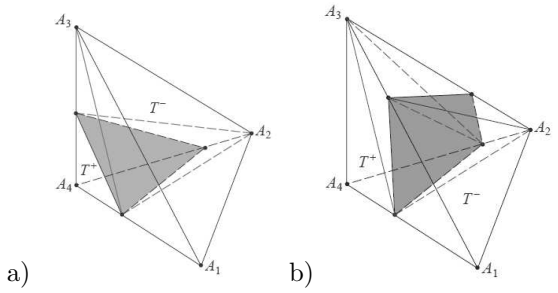


FIGURE 3. Partition a typical interface tetrahedron into sub-tetrahedra: a) Three-edge-cut b)Four-edge-cut [37].

Furthermore, based on the charge density provided by PIC, the right hand side vector of the linear system is formed through a similar procedure in the module *Global_RHS*. Then, the Dirichlet and Neumann boundary conditions are applied

in the modules *Global_RHS_EBC* and *Global_RHS_NBC* respectively. By using the preconditioned conjugate gradient method, the module *PCG_solver* is called to solve the sparse symmetric positive definite linear system for obtaining the electrostatic potential distribution in the domain, which will be used to push the plasma particles in the PIC simulation. All the parameters and boundary conditions which are related to form and solve the linear system of the IFE method are incorporated into a single input file `ife.inp`. Since these modules are relatively standard, we omit the related details to shorten the presentation of this paper.

3. Modules for Particle-In-Cell method

In this section, we will briefly review the basic ideas of the PIC method and then introduce the main PIC modules in the simulation tool, including the plasma parameter input, particle pushing, particle collision, particle injection, and particle collection. The PIC parameter input file with a general and complete data structure for specifying the plasma parameters and PIC control options, together with the automatic treatment of the plasma properties based on these parameters, allow users to conveniently change the type of the simulated plasma only in the input file. The efficient modules for the different operations on particles are one of the key factors for the high efficiency of this simulation tool.

3.1. Review for Particle-In-Cell method. The Particle-In-Cell method, which uses simulation particles (also referred as "macro-particles") to represent real plasma particles, is one of the most popular simulation method in the plasma simulation community because of its noise-reduction capability and moderate computational expense. This method works most efficiently on structured Cartesian meshes independent of complex interfaces. Important components of this method include: 1) deposition of the charge density from the particles to the mesh nodes, which provides the right hand side (RHS) vector of the interface Poisson's equation of the electric field; 2) solution of the interface Poisson equation for the electric field; 3) interpolation of the electric field at particle locations; 4) solving the equation of motion for moving particles over a short time period, and other related operations on the particles, such as collision, injection, and collection.

The PIC simulation for plasma environment is often a large-scale simulation since millions of simulation particles and thousands of PIC iteration steps are often needed for realistic applications. For each particle, the 3D coordinates of its location, the 3D velocity vector, and the species flag of the particle are saved in the computer memory. After the initial loading of particles in the simulation domain, a typical PIC iteration step consists of the following main stages:

(1) Deposit the charge density (see section 4.2):

The main task of this stage of PIC is to provide the electric charge density ρ on the right hand side of the equation (1), which will be solved by the IFE method in the next stage. For each particle, the mesh element, in which the particle locates, needs to be first identified through a fast searching algorithm since there are usually millions of particles in the simulation. Then the charge of the particle is deposited onto the mesh nodes through the linear or higher order interpolation/weighting based on the coordinates of the particle location. This is the main reason why PIC method works most efficiently on Cartesian meshes since it is trivial to locate the particle

and do the interpolation onto the Cartesian mesh.

(2) Solve the electric field (see section 2):

Once the electric charge density ρ is provided in the previous stage, the electrostatic potential ϕ, can be solved self-consistently from the interface Poisson's equation (1)-(5) for the electric field which is needed in the next stage. As reviewed in Section 2, through a convenient implementation of the IFE method based on the traditional finite element method, the IFE method can accurately solve this equation with optimal convergence rates on a structured mesh independent of the interface, which is much favorable for the efficiency of the PIC method.

(3) Interpolate electric field at particle locations (see section 4.3):

In this stage, the IFE solution of the interface Poisson's equation (1)-(5) is utilized to compute the electric field which needs to be evaluated at the locations of all particles in PIC for the next stage of pushing particles. The electric field $\mathbf{E}$ can be easily obtained from the electrostatic potential ϕ. It is strongly recommended to use the same interpolation/weighting as in the first stage, which can significantly reduce the associated numerical noise [6].

(4) Push particles (see section 3.5):

For a short time interval which is specified as the time step size in the PIC parameter input file, all the particles are pushed based on the electric field provided from the previous stage, which provides the particle locations for the next PIC iteration step. The trajectories of an individual charged particle is governed by the Newton-Lorentz equation. The equation of motion needs to be solved many times for pushing millions of particles at thousands of PIC iteration steps, which requires an efficient scheme with acceptable accuracy order and less memory requirement. Therefore, we adopted one commonly used scheme, leap-frog, in our IFE-PIC simulation tool.

(5) Implement particle collision (see section 3.3):

For certain types of applications, such as the discharge process in electric propulsion problems and interactions between plasma and materials, particle collision may need to be addressed in the simulation to accurately resolve the complex plasma environments. And different physical problems may lead to different collisions between different types of particles. Based on Monte Carlo Collision (MCC) method and Direct Simulation Monte Carlo (DSMC) method, most types of particle collisions, such as electron-neutral collision, ion-neutral collision, neutral-neutral collision, electron-electron collision, electron-ion collision, etc., can be simulated.

(6) Inject particles (see section 3.5):

In order to correctly simulate the effect of the external plasma source, photoelectrons, secondary electrons, etc., new plasma particles need to be appropriately injected into the simulation domain based on the velocity distribution specified in the PIC parameter input file, which is very critical to the accuracy of the simulation results.

(7) Handle particles hitting an object (see section 3.4):

At each PIC simulation step, some particles may be pushed to hit an object. Then it is critical to decide how to handle these particles for plasma surface interaction in order to keep a correct simulation particle set and apply the effect of these particles to the electric field. The general data structures of the object geometry, which is discussed in section 2.2, will play a key role in the efficient procedure to decide if a particle hits an object.

(8) Handle particles going out of the boundary (see section 3.5):

After the above operations on the particles, some particles may go out of the simulation domain. Different particle boundary conditions, which are specified in the PIC parameter input file, will provide different ways to deal with these particles. After this stage, the simulation will go to stage 1 to continue the next PIC iteration step.

3.2. PIC parameter input file. The PIC control options and plasma parameters are specified in an input file `pic.inp` as shown in Data 4. This input file is read by a PIC module at the beginning of the simulation and then automatically handled by the corresponding modules in the rest of the simulation. Therefore, by simply changing the parameters in this input file, different plasma environments can be automatically modeled in the IFE-PIC simulation tool. This feature provides a very convenient way to apply this tool to a wide range of plasma environments for different practical applications.

The following example shows the structure of the PIC parameter input file. The first line is a dummy string for the problem description. The second line specifies the total number of PIC iteration steps and the time step size for PIC iteration. The inclusion of photoelectrons can be specified via a logical switch ("T" or "F") as listed in the third line. Then the auto-save option/frequency is specified by the fourth line. The total number of species considered in the problem as ambient plasma is given in the fifth line, which is followed by the index, density, number of particles in each dimension, drifting velocity, and thermal velocity of each species.

Data 4. *Plasma parameter information specified in input file* `pic.inp`

```
! problem statement      ! a dummy line for problem statement
nt, dt                  ! total steps and step size
.T.                     ! switch of photoelectron species
1000                    ! output data every 1000 steps
2                       ! number of ambient species
1                       ! index for the first species: solar wind electrons
0.1359                  ! density for the first species
```

```
480, 40, 120                ! number of particles in each dimension
0.751928, 0, -0.065785      ! drifting velocities in each dimension
2.3350, 2.3350, 2.3350      ! thermal velocities in each dimension
2                           ! index for the first species: solar wind ions
0.1359                      ! density for the first species
480, 40, 120                ! number of particles in each dimension
0.751928, 0, -0.065785      ! drifting velocities in each dimension
0.04976, 0.04976, 0.04976   ! thermal velocities in each dimension
```

3.3. Module for particle collision. Different types of particle-particle collisions are taken into consideration and the collision in our simulation tool can be switched on/off with a parameter in the input file. The dynamics of electron-neutral scattering collisions is simulated with Monte Carlo Collision (MCC) method [7, 36, 60, 64] in the module *MCC_Collision*. Which type of electron-neutral excitation, electron-neutral ionization, and electron-neutral elastic scattering occurs is determined by the cross sections for these types of electron-neutral collisions. The ion-neutral charge exchange collisions, described by

$$Xe_{slow} + Xe^{+}_{fast} \longrightarrow Xe^{+}_{slow} + Xe_{fast}, \tag{15}$$

are also simulated with the MCC method. On the other hand, Direct Simulation Monte Carlo (DSMC) methodology [3–5, 9, 33, 34, 58, 59] in rarefied gas field is adopted to simulate the dynamics of neutral-neutral collisions in the module *DSMC_Collision*. Based on the probability theory and statistical approach, the DSMC method, shown in Figure 4, is capable of simulating the molecular motion and the inter-molecular collisions which are uncoupled over the time interval [36].

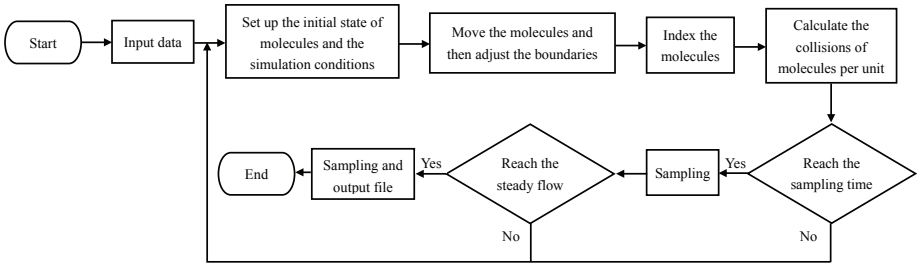


FIGURE 4. Flow chart of DSMC method.

3.4. Module for handling the particles hitting an object. During the movement of plasma particles in the simulation domain, particles may hit the surface of an object. The module *AdjustObjects* uses Algorithm 1 to decide whether a particle will collide with the surfaces of objects in the simulation domain based on the object data and the particle location. If yes, then the particle is either removed from the system by calling the module *RemovePart* or reflected by calling the module *DiffuseReflection* or *SpecularReflection* according to the different particle conditions on the object surface.

Algorithm 1. *Decide if a particle hits an object and deal with particle conditions on the object surface.*

DO i_part = 1, N_Particle
X_p $\leftarrow$ part(i_part, 1 : 3), X_po $\leftarrow$ X_p $-$ part(i_part, 4 : 6) $*$ dt
DO Iobj = 1, N_Objects
IF (Line segment $X\_p - X\_po$ intersects with the surface of object Iobj) **THEN**
call corresponding modules to deal with different particle conditions on the object surface
ENDIF
ENDDO
ENDDO

Here the parameter N_Particle is the total number of the simulation particles and N_Objects is the total numbers of objects in the simulation domain. X_p and X_po are the current and previous locations of the particle with index i_part, respectively. part(i_part, 1 : 3) and part(i_part, 4 : 6) store the coordinates of the location and the velocity vector of the particle with index i_part, respectively. dt is the time step size of PIC iteration.

Moreover, the charge of the removed particles can be also deposited on the interface, which can provide the flux jump condition on the interface. The accumulated surface charge density on the interface is critical to the interface problem with non-homogeneous flux jump condition which can be solved by non-homogeneous IFE methods [21, 23–25, 29, 72].

3.5. Other important modules for PIC. In this section, we will briefly introduce several other important modules for PIC, including pushing particles, injecting particles, handling particles going out of the boundary, and loading particles.

After the electric field in the simulation domain is obtained, the particles will be pushed in the simulation domain by numerically solving the following equations of motion:

$$m\frac{d\mathbf{v}}{dt} = \mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \tag{16}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}. \tag{17}$$

The above equations can be approximated by the well known leap-frog scheme:

$$m\frac{\mathbf{v}^{n+1/2} - \mathbf{v}^{n-1/2}}{\Delta t} = \mathbf{F}^{n}, \tag{18}$$

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^{n}}{\Delta t} = \mathbf{v}^{n+1/2}. \tag{19}$$

According to this algorithm, a module *Vel_ES* is adopted to update the three dimensional velocity for each particle based on the electric filed which is obtained from the IFE solution of the electric potential. Then the new position for each particle can be easily calculated for one time step size of PIC iteration.

During the plasma simulation, particles emigrate through the simulation domain. Hence new particles need to be appropriately injected into the simulation domain. The velocity distribution of injected particles follows a drifting Maxwellian velocity distribution, which can be sampled according to the method discussed

in [5]. Furthermore, it is critical for certain types of applications to appropriately inject photoelectrons and/or secondary electrons into the simulation. By calling corresponding modules *Inject_Ambient*, *Inject_Beam*, *Inject_PhotoElectrons*, *Inject_SecondaryElectrons*, etc. at each PIC iteration step, different types of plasma particles are injected into the simulation domain to continue the simulation process based on the plasma density, drifting velocities, and thermal velocities which are provided in the PIC parameter input file.

The moving simulation plasma particles may go out of the boundary of the simulation domain. These particles are normally absorbed and their associated data is removed from the particle array part by calling the module *RemovePart*. However, if a particle hits a periodic domain boundary, another particle is injected back correspondingly into the simulation domain from the opposite boundary with the same velocity. Therefore, the module *AdjustOuter* is performed to deal with the particles going out of the boundary according to the boundary conditions of the simulation domain which are provide in the PIC parameter input file.

The particle loading module is not very crucial for the solution accuracy if only the steady-state solution is what we seek from the particle simulation. However it may significantly reduce the computational cost if an appropriate set of particles are loaded before the PIC iteration starts. Hence, we provide a convenient module *Load_Particles* in our simulation tool to pre-load particles in the simulation.

4. Integration of IFE and PIC in the simulation tool

In this section, we will introduce how to integrate the IFE and PIC methods efficiently in our plasma simulation tool. The PIC method, which works most efficiently on Cartesian meshes, provides the charge density to the IFE method to form the right hand side vector of the linear system of IFE. The IFE method, which can accurately solve the interface problems on Cartesian meshes, provides the electrostatic potential field to PIC for pushing the plasma particles. Furthermore, the object information, which is discussed in Section 2.2, is critical to both IFE and PIC because the intersection points between the mesh lines and object surfaces are the key to define the IFE basis functions (see section 2.1) and the particles hitting an object need to be appropriately handled in PIC (see section 3.4). We will first use the general coding framework to show how to integrate the modules discussed in the previous two sections into a unified system. Then two efficient modules, which mainly serve as the key connection between IFE and PIC, will be discussed for the dynamic interactions between IFE and PIC.

4.1. Code structure of IFE-PIC simulation tool. The general framework of the IFE-PIC method is shown in Figure 5, which dynamically integrates the IFE method and PIC method together and clearly illustrates the roles of the modules discussed in this paper.

4.2. Deposit the particle charge density to mesh nodes for IFE. In order for the IFE method to solve the interface Poisson's equation (1)-(5) for the electrostatic potential ϕ, the electric charge density ρ on the right hand side of Equation (1) needs to be provided by the PIC method by accumulating the charge of all the particles on the mesh nodes for IFE. Since millions of simulation particles need to be located in the mesh for thousands of PIC iteration steps, it is necessary to utilize a very efficient method to deposit the charge density of the particles. Again, this is the main reason why PIC method works most efficiently on Cartesian meshes since it is trivial to locate the particle and deposit the charge onto a Cartesian mesh.

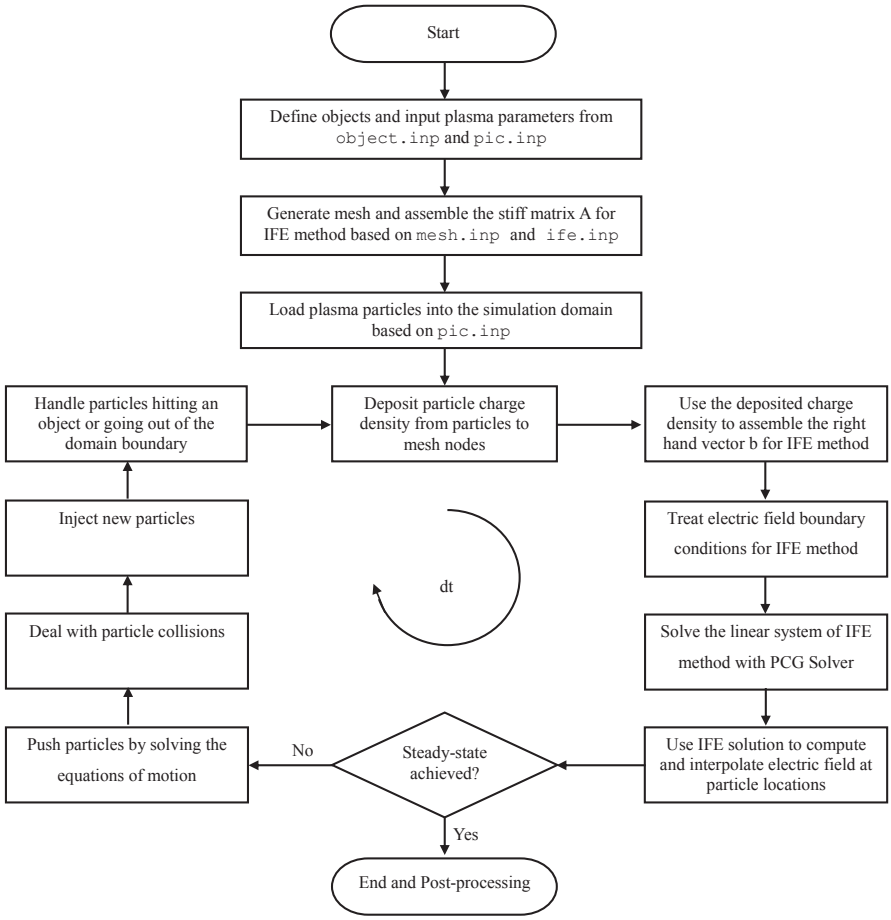


FIGURE 5. Flow chart of IFE-PIC method.

In a 3D domain, the contribution of a particle, which is located at the position $\mathbf{x_p}$, to the charge density at node (i, j, k) is calculated via linear weighting scheme based on volumes, as described below:

$$q_{i,j,k} = q_p \frac{V(\mathbf{x}_{i+1,j+1,k+1}, \mathbf{x}_p)}{V(\mathbf{x}_{i,j,k}, \mathbf{x}_{i+1,j+1,k+1})}, \tag{20}$$

where $\mathbf{x}_{i,j,k}$ is the mesh node with index (i, j, k), respectively, q_p is the charge density of the particle, and $V(\mathbf{x}, \mathbf{y})$ represents the volume of the box which can be uniquely determined by its two diagonal vertices $\mathbf{x}$ and $\mathbf{y}$. We can similarly compute the contribution of this particle to the charge density at other nodes of the box whose two diagonal vertices are $\mathbf{x}_{i,j,k}$ and $\mathbf{x}_{i+1,j+1,k+1}$.

As shown in Algorithm 2, the module *PIC-IFE* is called to deposit the particle charge density to mesh nodes in order to form the right hand side vector in the linear system of IFE in the IFE modules.

Algorithm 2. *Locate plasma particles and deposit charge to mesh nodes.*

DO i_part = 1, N_Particle

$x_{pos}, y_{pos}, z_{pos} \leftarrow$ part(i_part, 1 : 3)
isp $\leftarrow$ part(i_part, 7)
i, j, k $\leftarrow$ INT(x_{pos}/h_x), INT(y_{pos}/h_y), INT(z_{pos}/h_z)
dx, dy, dz $\leftarrow$ $x_{pos}/h_x - I, y_{pos}/h_y - J, z_{pos}/h_z - K$
rho_s(i, j, k, isp) = rho_s(i, j, k, isp) + (1 − dx) ∗ (1 − dy) ∗ (1 − dz)
rho_s(i + 1, j, k, isp) = rho_s(i + 1, j, k, isp) + dx ∗ (1 − dy) ∗ (1 − dz)
rho_s(i, j + 1, k, isp) = rho_s(i, j + 1, k, isp) + (1 − dx) ∗ dy ∗ (1 − dz)
rho_s(i, j, k + 1, isp) = rho_s(i, j, k + 1, isp) + (1 − dx) ∗ (1 − dy) ∗ dz
rho_s(i, j + 1, k + 1, isp) = rho_s(i, j + 1, k + 1, isp) + (1 − dx) ∗ dy ∗ dz
rho_s(i + 1, j, k + 1, isp) = rho_s(i + 1, j, k + 1, isp) + dx ∗ (1 − dy) ∗ dz
rho_s(i + 1, j + 1, k, isp) = rho_s(i + 1, j + 1, k, isp) + dx ∗ dy ∗ (1 − dz)
rho_s(i + 1, j + 1, k + 1, isp) = rho_s(i + 1, j + 1, k + 1, isp) + dx ∗ dy ∗ dz
ENDDO

DO isp=1,ispe_tot
  DO k=1,nz
    DO j=1,ny
      DO i=1,nx
        rho(i, j, k) = rho(i, j, k) + qs(isp) ∗ rho_s(i, j, k, isp)
      ENDDO
    ENDDO
  ENDDO
ENDDO

Here $(x_{pos}, y_{pos}, z_{pos})$ is the location of the particle with index i_part, isp is the species index, h_x, h_y, h_z are the mesh sizes in (x, y, z) direction, rho_s is the charge density of each species of plasma particle on mesh nodes, ispe_tot is the total number of species of plasma particles, (nx, ny, nz) are the number of mesh nodes in (x, y, z) direction, and qs is the unit charge of different types of plasma particle. Hence, the parameter rho is the charge density on mesh nodes which is used to form the right hand side vector in the linear system of IFE.

4.3. Interpolate electric field at particle locations. After the IFE method is applied to solve the interface Poisson's equation (1)-(5), the IFE solution of the electrostatic potential ϕ is utilized to compute the electric field which needs to be evaluated at the locations of all particles in PIC for pushing particles. The electric field **E** can be easily obtained from the electric potential ϕ as follows:

$$\mathbf{E}(x, y, z) = -\nabla\phi(x, y, z). \tag{21}$$

It is strongly recommended to use the same interpolation/weighting scheme as in the section 4.2 for depositing the electric field from the mesh nodes to the locations of all particles, which can significantly reduce the associated numerical noise [6]. The module *IFE-PIC* is utilized to interpolate the electric field at particle positions, shown in Algorithm 3.

Algorithm 3. *Interpolate electric field at particle positions.*

DO i_part = 1, N_Particle
  $x_{pos}, y_{pos}, z_{pos} \leftarrow$ part(i_part, 1 : 3)
  isp $\leftarrow$ part(i_part, 7)
  i, j, k $\leftarrow$ INT(x_{pos}/h_x), INT(y_{pos}/h_y), INT(z_{pos}/h_z)

$$dx, dy, dz \leftarrow x_{pos}/h_x - I, y_{pos}/h_y - J, z_{pos}/h_z - K$$

$$\begin{aligned}
f &= efx(i,j,k) + dx * (efx(i+1,j,k) - efx(i,j,k)) \\
f &= f + dy * (efx(i,j+1,k) + dx * (efx(i+1,j+1,k) - efx(i,j+1,k)) - f) \\
g &= efx(i,j,k+1) + dx * (efx(i+1,j,k+1) - efx(i,j,k+1)) \\
g &= g + dy * (efx(i,j+1,k+1) + dx * (efx(i+1,j+1,k+1) \\
&\quad - efx(i,j+1,k+1)) - g) \\
efx &= f + dz * (g - f)
\end{aligned}$$

The computation of efy and efy is similar to that of efx.

ENDDO

Here efx, efy, efz are the (x, y, z) components of the electric field **E**.

5. Numerical experiments

In this section we present two numerical examples for the ion thruster accelerator grid erosion and plasma thruster plume-spacecraft interaction. The examples involve different types of plasma, complex objects, collisions, and so on, which illustrate the applicability and the discussed features of the IFE-PIC plasma simulation tool.

5.1. Ion thruster accelerator grid erosion. The ion optics system is made of electrically biased multi-aperture grids. The ions in the discharge chamber are extracted and accelerated through the optics using the high potential between the grids, which is how the ions offer specific impulses for the spacecraft. The erosion of accelerator grid in ion thruster is a key issue in the design of grid which is very important in ion-thruster system. It has been found that the erosion of the accelerator is caused by the CEX ions that resulted from the charge-exchange collisions between fast beam ions and neutral atoms. Through the IFE-PIC simulation tool, the beam extraction process and the charge exchange collisions can be simulated self-consistently to study the erosion mechanism of accelerator grid.

Wang et al. developed the first ground test data validated, 3D PIC simulation model of ion thruster accelerator grid erosion [71]. The finite difference based PIC model [71] was extended by using the IFE-PIC method in [37] to simulate the grid erosion for an entire sub-scale ion optics [66]. The IFE-PIC simulation setup of an entire sub-scale ion optics of [37, 66] is shown in Figure 6. Recently, Cao et al. [9] further extended the simulation model to investigate the barrel erosion of the ion thruster accelerator.

In table 1, the potential V_s of the screen grid is set slightly below the plasma potential V_N to extract the ions and screen out the electrons. The potential V_a of the accelerator grid is set at a negative potential to provide the accelerating field and prevent electrons from back-streaming. The total accelerating voltage is denoted by V_T. These parameters and the geometry information for the ion optic model can be found in [71].

Since the upstream boundary surface is set to be immersed in the upstream plasma, we apply Dirichlet potential boundary condition on the upstream surface. All side surfaces satisfy the Neumann boundary condition due to a symmetric potential boundary condition. Since the downstream boundary surface is set to be immersed in the downstream plasma, we assume a Neumann potential boundary condition on the downstream surface such that the potential on the downstream

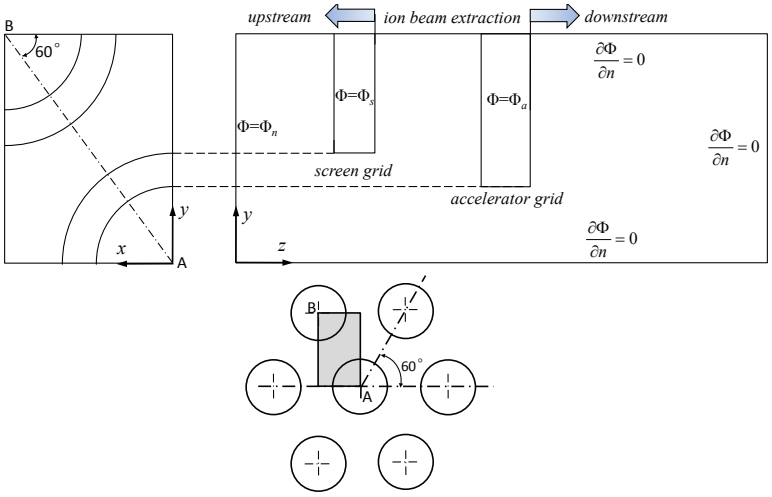


FIGURE 6. Two-quarter aperture simulation domain [37, 66].

TABLE 1. Operating conditions for the ion optics.

V_N	V_s	V_a	V_T
1800 V	1770 V	-190 V	1990 V

surface is determined self-consistently. The particles hit the $z = 0$ and $z = z_{max}$ surfaces are absorbed, while those hitting the other four surfaces are reflected. More details for the problem set-up and plasma parameters can be found in [9].

Figure 7 shows the selected potential contour lines and the beamlet ion density contours for the plasma density in the upstream region with n_0 equal to $0.5 \times 10^{17} m^{-3}$. It can be observed that under well-focused operating condition, the ions which impact on the downstream surface of the accelerator grid are all CEX ions. Figure 8 shows the erosion of the downstream surface caused by CEX ions after 1000 hour. The generation of CEX ions in this example is simulated with the collision module *MCC_Collision*. The pit and groove pattern on the downstream surface of accelerator grid agree with the phenomenon which has been observed via simulation and experiment in [71]. Both the existing results and the new results clearly illustrate the capability of the IFE-PIC simulation tool for this type of plasma problems with particle collisions and non-trivial geometry.

5.2. Plasma thruster plume-spacecraft interaction. An electric thruster propels a spacecraft by continuously emitting a partially ionized gas. However, the interaction between the thruster plume and the spacecraft surface may adversely affect spacecraft operation. Therefore, it is necessary to study the interaction between the thruster plume and the spacecraft which is crucial in the design of a satellite with electric propulsion.

Wang et al. developed the first in-flight data validated, 3D PIC simulation model to study ion thruster plume-spacecraft interactions for the Deep Space 1 spacecraft [65]. The finite-difference based PIC model was later extended by using the IFE-PIC method for the Dawn spacecraft so it can resolve the more details of the complicated surface geometry [37, 39, 68]. As showed in [37, 39, 68], even

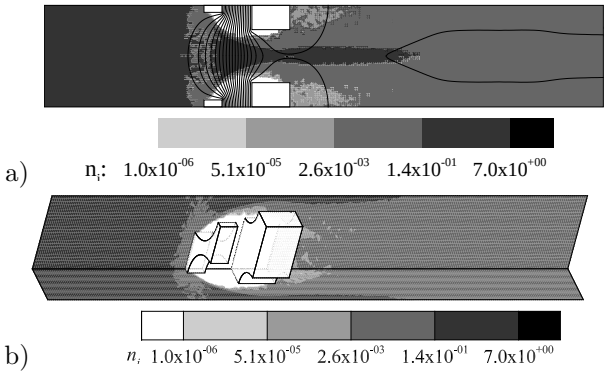


FIGURE 7. Potential and ion density (with CEX) results for $n_0 = 0.5 \times 10^{17} m^{-3}$ [9].

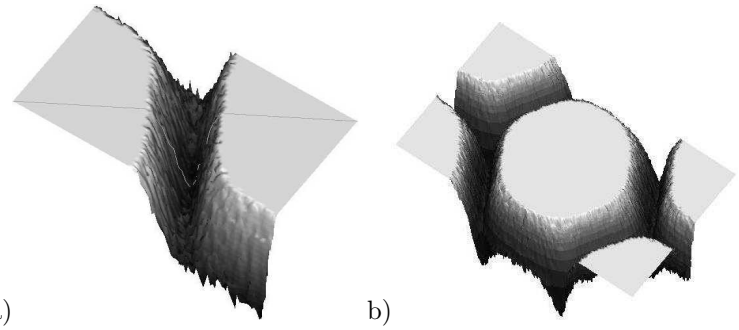
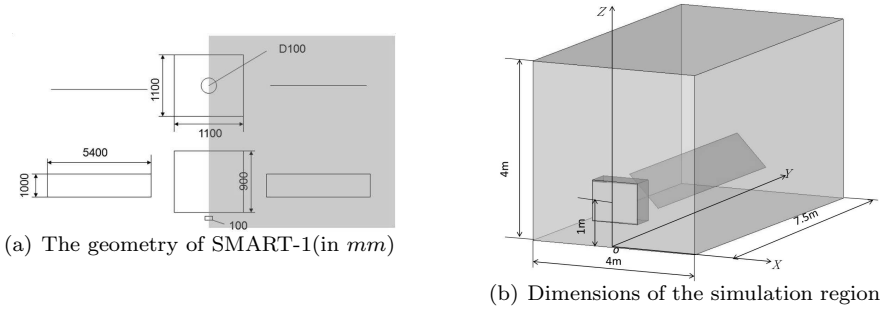


FIGURE 8. Erosion depth on the downstream surface of accelerator grid: a) erosion profile after 1000h b) pit and groove pattern of the downstream surface erosion.

a simplified Dawn spacecraft model will need to include various object shapes in order to retain the geometrical details that may affect the results. The numerical experiments show that different types of objects, plasmas, and collisions can be easily input in the object data structure and the PIC parameter input file of the IFE-PIC simulation tool, and then automatically handled by the corresponding modules to obtain physically valid results.

In the rest of this section, we present another numerical example which simulates the Hall thruster plasma plume interactions with the SMART-1 spacecraft [60, 61]. The geometry and dimensions of SMART-1 spacecraft model are illustrated in Figure 9(a). Based on the SMART-1 spacecraft model in [61], the main body of SMART-1 can be considered as a cubic shape with the dimensions of $l \times w \times h = 1100mm \times 1100mm \times 900mm$. The Hall thruster is simplified as a cylinder with the diameter of $100mm$ and the height of $50mm$. Two thin rectangle plates with length= $5400mm$ and width= $1000mm$ are utilized to represent the solar arrays which can rotate around the satellite. Let θ denote the divergence angle of Hall thruster and γ denote the angle between the normal direction of the solar arrays and the center line of the beam flow.

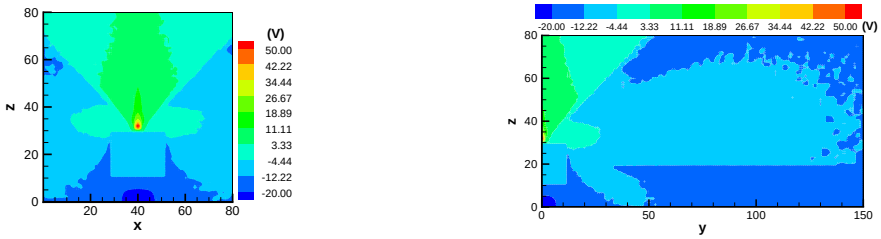
Due to the symmetry of SMART-1, half of the geometry model is employed to simulate the interaction between the plume and the surface of SMART-1, as shown in Figure 9(b). The simulation domain has a size of $4.0m \times 7.5m \times 4.0m$ with



(a) The geometry of SMART-1(in mm)

(b) Dimensions of the simulation region

FIGURE 9. Geometry of SMART-1 and simulation domain [60].



(a) (b)

FIGURE 10. Plasma potential distribution (at $\theta = 45°$,$\gamma = 0°$) [60].

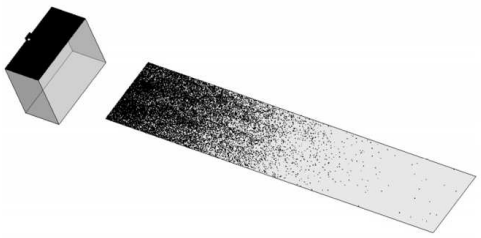
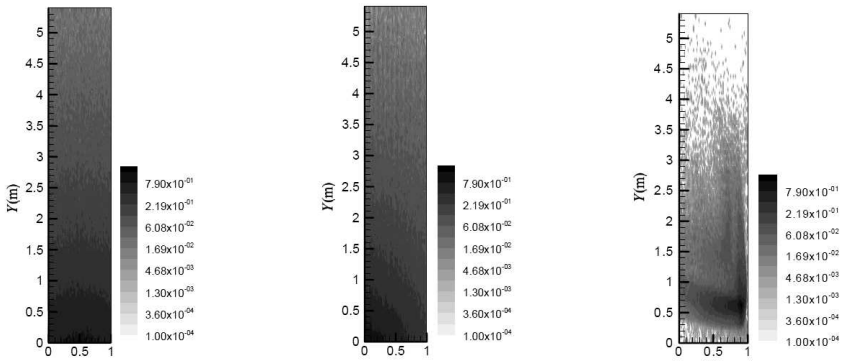


FIGURE 11. CEX ions distribution on the surface of SMART-1 main body and solar array.

$81 \times 151 \times 81$ grid points. The ion-neutral charge exchange collisions are simulated with Monte Carlo method.

In this numerical example, we consider three different plasma species, including Xe neutral atoms, Xe^+ beam ions, and charge-exchange ions. The plasma parameters can be found in [60]. Zero-Neumann boundary conditions is applied on all the domain boundaries and the particles going out of domain boundary are removed. Dirichlet boundary conditions, $\phi = -2V$, are applied to the thruster and satellite body and all the particles colliding on the object surfaces are absorbed.



(a) $\gamma = 0°$ (b) $\gamma = 45°$ (c) $\gamma = 90°$

FIGURE 12. Heat flux density distribution by CEX ions on the solar array of SMART-1(at $\theta = 45°$)(in W/m^2).

Figure 10 presents the distribution of the potential under the condition of $\theta = 45°$ and $\gamma = 0°$. The backflow of the CEX ions result in that the iso-potential surface expand toward a direction opposite to the ion beam movement. This makes the pattern of the distribution looks like a mushroom which has been observed for SMART-1. Figure 11 shows the CEX ions distribution on the surface of both the main body and the solar array. Figure 12 compares the heat flux density distribution by CEX ions on the solar array for different γ with $\theta = 45°$. As expected, it can be easily observed that the solar arrays receive much more CEX ions and bear much more heat flux at $\gamma = 0°$ than $\gamma = 90°$. All of the existing and new physically valid results well illustrate the applicability of the IFE-PIC simulation tool to this problem and the features of the tool discussed in this paper.

6. Conclusions

In this paper we presented the IFE-PIC simulation tool for plasma simulation. Both the IFE method and the PIC method are briefly reviewed. The featured modules and data structures of the IFE-PIC package are discussed in detail and the other modules are briefly introduced to illustrate the formation of the whole package. The IFE and PIC methods are integrated into a dynamical system through their interactions for charge density and electric potential field. Two numerical experiments are provided to illustrate the applicability and features of the IFE-PIC simulation tool.

Acknowledgments

The authors would like to thank Dr. Raed Kafafy for his significant and fundamental contribution to the IFE-PIC simulation tool as one of the major developers of this tool and Dr. Tao Lin for his long-term support on the IFE method as one of the major developers of IFE. This work is partially supported by National Natural Science Foundation of China (11175052), Shenzhen fundamental research program (JCYJ20160226201347750) and University of Missouri Research Board.

References

[1] S. Adjerid, M. Ben-Romdhane, and T. Lin. Higher degree immersed finite element methods for second-order elliptic interface problems. Int. J. Numer. Anal. Model., 11(3):541?66, 2014.

[2] S. Adjerid and T. Lin. p-th degree immersed finite element for boundary value problems with discontinuous coefficients. Appl. Numer. Math, 59(6):1303–1321, 2009.

[3] M. Andrenucci, L. Biagioni, and A. Passaro. PIC/DSMC models for Hall effect thruster plumes: Present status and ways forward. In AIAA-2002-4254, 38th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Inianapolis, Indiana, 2002.

[4] M. Auweter-Kurtz, M. Fertig, D. Petkow, T. Stindl, M. Quandt, C. Munz, P. Adamis, M. Resch, S. Roller, and D. D'Andrea. Development of a hybrid PIC/DSMC code. In IEPC-2005-71, 29 th International Electric Propulsion Conference, Princeton, USA, 2005.

[5] G. A. Bird. Molecular Gas Dynamics and the Direct Simulation of Gas Flows. Clarendon Press, 1994.

[6] C. Birdsall and A. Langdon. Plasma Physics via Computer Simulation. CRC Press, 2005.

[7] C. K. Birdsall. Particle-in-cell charged-particle simulations, plus monte carlo collisions with neutral atoms, pic-mcc. IEEE Trans. Plasma Sci., 19(2):65–85, 1991.

[8] B. Camp, T. Lin, Y. Lin, and W. Sun. Quadratic immersed finite element spaces and their approximation capabilities. Adv. Comput. Math., 24(1-4):81–112, 2006.

[9] H. Cao, Y. Chu, E. Wang, Y. Cao, G. Xia, and Z. Zhang. Numerical simulation study on barrel erosion of ion thruster accelerator grid. J. Propul. Power, 31(6):1785–1792, 2015.

[10] H. Cao, Q. Li, K. Shan, Y. Cao, and L. Zheng. Effect of preionization on the erosion of the discharge channel wall in a hall thruster using a kinetic simulation. IEEE Trans. Plasma Sci., 43(1):130–140, 2015.

[11] Y. Cao, Y. Chu, X.-M. He, and T. Lin. An iterative immersed finite element method for an electric potential interface problem based on given surface electric quantity. J. Comput. Phys., 281:82–95, 2015.

[12] Y. Cao, Y. Chu, X. Zhang, and X. Zhang. Immersed finite element methods for unbounded interface problems with periodic structures. J. Comput. Appl. Math., 307:72–81, 2016.

[13] S. Chou. An immersed linear finite element method with interface flux capturing recovery. Discrete Contin. Dyn. Syst. Ser. B, 17(7):2343–2357, 2012.

[14] S. Chou, D. Y. Kwak, and K. T. Wee. Optimal convergence analysis of an immersed interface finite element method. Adv. Comput. Math., 33(2):149–168, 2010.

[15] Y. Chu. Numerical Simulation Research About Sheath Characteristics and Lunar Dust Levitation in Terminator Region of Lunar Surface. PhD. Dissertation, Harbin Institute of Technology, 2015.

[16] Y. Chu, Y. Cao, X.-M. He, and M. Luo. Asymptotic boundary conditions with immersed finite elements for interface magnetostatic/electrostatic field problems with open boundary. Comput. Phys. Commun., 182(11):2331–2338, 2011.

[17] D. Depew, D. Han, J. Wang, X.-M. He, and T. Lin. Immersed-Finite-Element Particle-In-Cell simulations of lunar surface charging, #199. Proceedings of the 13th Spacecraft Charging Technology Conference, Pasadena, California, June 23-27, 2014.

[18] R. E. Ewing, Z. Li, T. Lin, and Y. Lin. The immersed finite volume element methods for the elliptic interface problems. Modelling '98 (prague). Math. Comput. Simulation, 50(1-4):63–76, 1999.

[19] W. Feng, X.-M. He, Y. Lin, and X. Zhang. Immersed finite element method for interface problems with algebraic multigrid solver. Commun. Comput. Phys., 15(4):1045–1067, 2014.

[20] G. Fubiani and J. P. Boeuf. Role of positive ions on the surface production of negative ions in a fusion plasma reactor type negative ion sourceinsights from a three dimensional particle-in-cell monte carlo collisions model. Phys. Plasmas, 20(11):113511–1–10, 2013.

[21] Y. Gong, B. Li, and Z. Li. Immersed-interface finite-element methods for elliptic interface problems with non-homogeneous jump conditions. SIAM J. Numer. Anal., 46:472–495, 2008.

[22] Y. Gong and Z. Li. Immersed interface finite element methods for elasticity interface problems with non-homogeneous jump conditions. Numer. Math. Theory Methods Appl., 3(1):23–39, 2010.

[23] D. Han, J. Wang, and X.-M. He. A non-homogeneous immersed-finite-element particle-in-cell method for modeling dielectric surface charging in plasmas. IEEE Trans. Plasma Sci., 44(8):1326–1332, 2016.

[24] D. Han, P. Wang, X.-M. He, T. Lin, and J. Wang. A 3D immersed finite element method with non-homogeneous interface flux jump for applications in particle-in-cell simulations of plasma-lunar surface interactions. J. Comput. Phys., 321:965–980, 2016.
[25] X.-M. He. Bilinear immersed finite elements for interface problems. Ph.D. Dissertation, Virginia Polytechnic Institute and State University, 2009.
[26] X.-M. He, T. Lin, and Y. Lin. Approximation capability of a bilinear immersed finite element space. Numer. Methods Partial Differential Equations, 24(5):1265–1300, 2008.
[27] X.-M. He, T. Lin, and Y. Lin. A bilinear immersed finite volume element method for the diffusion equation with discontinuous coefficients, dedicated to richard e. ewing on the occasion of his 60th birthday. Commun. Comput. Phys., 6(1):185–202, 2009.
[28] X.-M. He, T. Lin, and Y. Lin. Interior penalty bilinear IFE discontinuous Galerkin methods for elliptic equations with discontinuous coefficient, dedicated to David Russell's 70th birthday. J. Syst. Sci. Complex., 23(3):467–483, 2010.
[29] X.-M. He, T. Lin, and Y. Lin. Immersed finite element methods for elliptic interface problems with non-homogeneous jump conditions. Int. J. Numer. Anal. Model., 8(2):284–301, 2011.
[30] X.-M. He, T. Lin, and Y. Lin. The convergence of the bilinear and linear immersed finite element solutions to interface problems. Numer. Methods Partial Differential Equations, 28(1):312–330, 2012.
[31] X.-M. He, T. Lin, and Y. Lin. A selective immersed discontinuous Galerkin method for elliptic interface problems. Math. Methods Appl. Sci., 37(7):983–1002, 2014.
[32] X.-M. He, T. Lin, Y. Lin, and X. Zhang. Immersed finite element methods for parabolic equations with moving interface. Numer. Methods Partial Differential Equations, 29(2):619–646, 2013.
[33] M. Ivanov and S. Rogasinskii. Theoretical analysis of traditional and modern schemes of the dsmc method. In Rarefied Gas Dynamics, volume 1, pages 629–642, 1991.
[34] M. Ivanov and S. Rogasinsky. Analysis of numerical techniques of the direct simulation monte carlo method in the rarefied gas dynamics. RUSS. J. NUMBER. ANAL. M., 3:453–466, 1988.
[35] H. Ji, J. Chen, and Z. Li. A symmetric and consistent immersed finite element method for interface problems. J. Sci. Comput., 61(3):533–557, 2014.
[36] H. Jian, Y. Chu, H. Cao, Y. Cao, X.-M. He, and G. Xia. Three-dimensional IFE-PIC numerical simulation of background pressure's effect on accelerator grid impingement current for ion optics. Vacuum, 116:130–138, 2015.
[37] R. Kafafy. Immersed finite element Particle-In-Cell simulations of ion propulsion. Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2005.
[38] R. Kafafy, T. Lin, Y. Lin, and J. Wang. Three-dimensional immersed finite element methods for electric field simulation in composite materials. Int. J. Numer. Meth. Engrg., 64(7):940–972, 2005.
[39] R. Kafafy and J. Wang. A hybrid grid immersed finite element particle-in-cell algorithm for modeling spacecraft-plasma interactions. IEEE Trans. Plasma Sci., 34(5):2114–2124, 2006.
[40] R. Kafafy, J. Wang, and T. Lin. A hybrid-grid immersed-finite-element particle-in-cell simulation model of ion optics plasma dynamics. Dyn. Contin. Discrete Impuls. Syst. Ser. B Appl. Algorithms, 12:1–16, 2005.
[41] M. Keidar, I. D. Boyd, and I. I. Beilis. Plasma flow and plasma-wall transition in hall thruster channel. Phys. Plasmas, 8(12):5315–5322, 2001.
[42] D. Y. Kwak, K. T. Wee, and K. S. Chang. An analysis of a broken p_1-nonconforming finite element method for interface problems. SIAM J. Numer. Anal., 48(6):2117–2134, 2010.
[43] T. Lee, Y. Chang, J. Choi, D. Kim, W. Liu, and Y. Kim. Immersed finite element method for rigid body motions in the incompressible Navier-Stokes flow. Comput. Methods Appl. Mech. Engrg., 197(25-28):2305–2316, 2008.
[44] Z. Li. The immersed interface method using a finite element formulation. Appl. Numer. Math., 27(3):253–267, 1997.
[45] Z. Li, T. Lin, Y. Lin, and R. C. Rogers. An immersed finite element space and its approximation capability. Numer. Methods Partial Differential Equations, 20(3):338–367, 2004.
[46] Z. Li, T. Lin, and X. Wu. New Cartesian grid methods for interface problems using the finite element formulation. Numer. Math., 96(1):61–98, 2003.
[47] Z. Li and X. Yang. An immersed finite element method for elasticity equations with interfaces. Recent advances in adaptive computation. Contemp. Math., 383:285–298, 2005.
[48] T. Lin, Y. Lin, and W. Sun. Error estimation of a class of quadratic immersed finite element methods for elliptic interface problems. Discrete Contin. Dyn. Syst. Ser. B, 7(4):807–823, 2007.

[49] T. Lin, Y. Lin, and X. Zhang. A method of lines based on immersed finite elements for parabolic moving interface problems. Adv. Appl. Math. Mech., 5(4):548–568, 2013.
[50] T. Lin, Y. Lin, and X. Zhang. Immersed finite element method of lines for moving interface problems with nonhomogeneous flux jump. Contemp. Math., 586:257–265, 2013.
[51] T. Lin, Y. Lin, and X. Zhang. Partially penalized immersed finite element methods for elliptic interface problems. SIAM J. Numer. Anal., 53(2):1121–1144, 2015.
[52] T. Lin and D. Sheen. The immersed finite element method for parabolic problems with the Laplace transformation in time discretization. Int. J. Numer. Anal. Model., 10(2):298–313, 2013.
[53] T. Lin, D. Sheen, and X. Zhang. A locking-free immersed finite element method for planar elasticity interface problems. J. Comput. Phys., 247:228–247, 2013.
[54] T. Lin and J. Wang. An immersed finite element electric field solver for ion optics modeling. In Proceedings of AIAA Joint Propulsion Conference, Indianapolis, IN, July, 2002. AIAA, 2002-4263.
[55] T. Lin and J. Wang. The immersed finite element method for plasma particle simulation. In Proceedings of AIAA Aerospace Sciences Meeting, Reno, NV, Jan., 2003. AIAA, 2003-0842.
[56] T. Lin and X. Zhang. Linear and bilinear immersed finite elements for planar elasticity interface problems. J. Comput. Appl. Math., 236(18):4681–4699, 2012.
[57] M. Nakano. Three-dimensional simulations of grid erosion in ion engines. Vacuum, 83(1):82–85, 2008.
[58] D. Oh and D. Hastings. Three dimensional PIC-DSMC simulations of Hall thruster plumes and analysis for realistic spacecraft configurations. In AIAA-96-3299, 32nd Joint Propulsion Conference and Exhibit, Lake Buena Vista, FL, 1996.
[59] V. V. Serikov, S. Kawamoto, and K. Nanbu. Particle-in-cell plus direct simulation monte carlo (pic-dsmc) approach for self-consistent plasma-gas simulations. IEEE Trans. Plasma Sci., 27(5):1389–1398, 1999.
[60] K. Shan, Y. Chu, Q. Li, L. Zheng, and Y. Cao. Numerical simulation of interaction between hall thruster cex ions and smart-1 spacecraft. Math. Prob. Eng., Accepted.
[61] M. Tajmar, R. Sedmik, and C. Scharlemann. Numerical simulation of smart-1 hall-thruster plasma interactions. J. Propul. Power, 25(6):1178–1188, 2009.
[62] Y. Todo, H. L. Berk, and B. N. Breizman. Simulation of intermittent beam ion loss in a tokamak fusion test reactor experiment. Phys. Plasmas, 10(7):2888–2902, 2003.
[63] S. Vallaghè and T. Papadopoulo. A trilinear immersed finite element method for solving the electroencephalography forward problem. SIAM J. Sci. Comput., 32(4):2379–2394, 2010.
[64] E. Wang, Y. Chu, Y. Cao, and J. Li. Numerical simulation of erosion mechanism for ion thruster accelerator grid aperture walls based on ife-pic and mcc methods. High Volt. Eng., 39(7):1763–1771, 2013.
[65] J. Wang, D. Brinza, and M. Young. Three-dimensional particle simulations of ion propulsion plasma environment for deep space 1. J. Spacecraft Rockets, 38(3):433–440, 2001.
[66] J. Wang, Y. Cao, R. Kafafy, R.A. Martinez, and J.D. Williams. Numerical and experimental investigations of cross-over ion impingement for sub-scale ion optics. J. Propul. Power, 24(3):562–570, 2008.
[67] J. Wang, Y. Cao, R. Kafafy, J. Pierru, and V. Decyk. Ion propulsion simulations using parallel supercomputers. In 29th International Electric Propulsion Conference, Princeton, NJ, Oct. 31-Nov.4, 2005. IEPC, 2005-271.
[68] J. Wang, Y. Cao, R. Kafafy, J. Pierru, and V. K. Decyk. Simulations of ion thruster plume-spacecraft interactions on parallel supercomputer. IEEE Trans. Plasma Sci., 34(5):2148–2158, 2006.
[69] J. Wang, X.-M. He, and Y. Cao. Modeling spacecraft charging and charged dust particle interactions on lunar surface. Proceedings of the 10th Spacecraft Charging Technology Conference, Biarritz, France, 2007.
[70] J. Wang, X.-M. He, and Y. Cao. Modeling electrostatic levitation of dusts on lunar surface. IEEE Trans. Plasma Sci., 36(5):2459–2466, 2008.
[71] Joseph Wang, James Polk, John Brophy, and Ira Katz. Three-dimensional particle simulations of ion-optics plasma flow and grid erosion. Journal of Propulsion and Power, 19(6):1192–1199, November-December 2003.
[72] P. Wang. Immersed finite element particle-in-cell modeling of surface charging in rarefied plasmas. Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2010.
[73] D. Yu, H. Liu, Y. Cao, and H. Fu. The effect of magnetic mirror on near wall conductivity in hall thrusters. Contrib. Plasma Phys., 48(8):543–554, 2008.

[74] X. Zhang. Nonconforming immersed finite element methods for interface problems. Ph.D. Dissertation, Virginia Polytechnic Institute and State University, 2013.

Department of Mechanical Engineering & Automation, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, Guangdong 518055, P. R. China, ychuan.chu@hitsz.edu.cn

Department of Astronautical Engineering, University of Southern California, Los Angles, CA 90089, USA, daoruhan@usc.edu

Department of Mechanical Engineering & Automation, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, Guangdong 518055, P. R. China, yongc@hitsz.edu.cn, corresponding author

Department of Mathematics and Statistics, Missouri University of Science and Technology, Rolla, MO 65409, USA, hex@mst.edu, corresponding author

Department of Astronautical Engineering, University of Southern California, Los Angles, CA 90089, USA, josephjw@usc.edu