Scientific Programming 15 (2007) 83–94
IOS Press


# Electric propulsion plume simulations using parallel computer

Joseph Wang[a,b,*], Yong Cao[a], Raed Kafafy[a] and Viktor Decyk[b,c]
[a]*Department of Aerospace & Ocean Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*
[b]*Jet Propulsion Laboratory, Pasadena, CA 91109, USA*
[c]*Department of Physics, University of California, Los Angles, CA 90095, USA*

**Abstract**. A parallel, three-dimensional electrostatic PIC code is developed for large-scale electric propulsion simulations using parallel supercomputers. This code uses a newly developed immersed-finite-element particle-in-cell (IFE-PIC) algorithm designed to handle complex boundary conditions accurately while maintaining the computational speed of the standard PIC code. Domain decomposition is used in both field solve and particle push to divide the computation among processors. Two simulations studies are presented to demonstrate the capability of the code. The first is a full particle simulation of near-thruster plume using real ion to electron mass ratio. The second is a high-resolution simulation of multiple ion thruster plume interactions for a realistic spacecraft using a domain enclosing the entire solar array panel. Performance benchmarks show that the IFE-PIC achieves a high parallel efficiency of $\geqslant 90\%$.

## 1. Introduction

Electric propulsion is a critical enabling technology for future deep space missions. An electric propulsion (EP) device propels a spacecraft by continuously emitting a high speed plasma flow over long periods of time. One of the important issues in the application of electric propulsion is the interactions from thruster exhaust plume. In addition to being an interesting plasma physics problem, the thruster plume has long raised both engineering and science concerns as it may contaminate spacecraft surfaces and sensors, induce complex plasma interactions, and interfere with plasma and magnetic field measurements from spacecraft. Due to the complex nature of the physics involved, the difficulty of creating in-flight conditions in ground tests, and a lack of flight testing opportunity, physics based modeling is increasingly being used to predict EP plume effects. EP plume modeling has been a subject of extensive recent research. Some recent plume modeling studies can be found in [1–7] and references therein.

The underlying physics of many EP plume interaction problems is that of a collisionless plasma flow. Most physics based plume models are developed using the particle-in-cell (PIC) method. A PIC code models a plasma as many macro-particles and follows the evolution of the orbits of individual macro-particles in the self-consistent electromagnetic field [9]. In a PIC code, the particles are located anywhere the simulation domain but the field quantities are defined only on discrete mesh points. A PIC code uses a “gather" step to interpolate fields from mesh points to particle positions to push particles and a “scatter" step to deposit particle quantities to mesh points for solving the electric field. The particle trajectories and the electric fields are solved iteratively between a field solver and a particle pusher. While the particle simulation method allows one to study a problem from the very fundamental level, the scope of the physics that can be resolved in a simulation study critically depends on the computational power.

*Corresponding author: Joseph Wang, Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24061-0203, USA. Tel.: +1 540 231 8114; Fax: +1 540 231 9632; E-mail: jowang@vt.edu.

ISSN 1058-9244/07/$17.00 

While recent advances in parallel computing technology have provided computational possibilities that were previously not conceivable, parallel computing has not been widely used in EP plume modeling. This is in part due to specific difficulties in parallel implementation. Previously, various 3-D electrostatic and electromagnetic PIC codes have been implemented on parallel supercomputers (see for example [10–16] and references therein). Domain decomposition is typically used to divide the computations among processors. Many parallel electromagnetic PIC codes are designed to use a local, non-iterative method such as the finite-difference time-domain (FDTD) method or discrete-volume time-domain method to minimize the overhead of inter-processor communications. The fast Fourier transform (FFT) method has also been widely used in parallel electrostatic and electromagnetic PIC codes. High parallel efficiencies have been demonstrated for PIC codes using such field solution techniques. However, most EP plume simulations require one to solve an electrostatic boundary value problem involving complex boundary conditions inside the simulation domain. For such problems, a local non-iterative field solve method cannot be applied because of the elliptical nature of the Poisson's equation. The FFT method also can not be easily applied because of the presence of complex internal boundaries. Hence, one's choices on the field solution methods are limited to global, iterative methods. In addition, an unstructured body-fit mesh is also typically needed in order to solve the electric field accurately in the presence of a complex object. Developing a field solver that is capable of solving the Poisson's equation under complex boundary conditions with high parallel efficiency has been a major challenge in the parallel implementation of electrostatic PIC codes.

This paper presents a parallel PIC code developed for large-scale, 3-dimensional electrostatic PIC simulations involving complex internal boundaries. This parallel code incorporates a newly developed immersed-finite-element (IFE) algorithm [17] to solve the electric field. The IFE algorithm uses a structured Cartesian mesh for problems involving arbitrarily shaped boundaries. It is shown that the IFE solution accuracy is comparable to that of the standard finite element method using a body-fit unstructured mesh [17]. The IFE formulation allows one to retain the same particle-push and particle-mesh interpolation of the standard finite difference based PIC. Moreover, with a Cartesian mesh, the algebraic systems in the IFE method are always symmetric and positive definite, maintaining the desirable banded structure. This allows easy deployment of fast solution techniques such as preconditioned conjugate gradient, ADI, and multi-grid methods. Domain decomposition techniques can also be easily incorporated.

To demonstrate the effectiveness of the parallel, immersed finite element PIC (IFE-PIC) method for EP plume modeling, we present two ion thruster plume simulations in this paper. The first is a full particle PIC simulation of plume interaction in the near-thruster region using real ion to electron mass ratio. Due to computational constrains, almost all existing PIC simulations of EP plumes use the hybrid approach where only the ions are treated as particles and the electrons are simplified using various assumptions. However, as the physics in the near thruster region is primarily determined by electron dynamics, one must adopt the kinetic approach for both ions and electrons in order to resolve the near-thruster plume characteristics. The second is a high-resolution simulation of multiple ion thruster plume interactions with spacecraft. Due to computational constraints, previous PIC simulations of EP plume interactions are often limited to simplified spacecraft configurations and a relatively small simulation domain enclosing a fraction of spacecraft body. The simulation presented here considers a realistic spacecraft configuration modelled after the DAWN spacecraft with three ion thrusters, and uses a large, high resolution simulation mesh which resolves the charge-exchange plasma Debye length in a domain enclosing the entire spacecraft and the solar array panel.

Section 2 briefly discusses the algorithm and parallel implementation. Section 3 presents simulation model and simulation results. Section 4 presents performance benchmarks of running the parallel code on a Dell cluster. Section 5 contains a summary and conclusions.

## 2. Algorithm and parallel implementation

### *2.1. IFE-PIC*

The basic function of an electrostatic PIC code is to solve the electric field self-consistently with the boundary conditions and the space charge of the particles from the Poisson's equation

$$\nabla \cdot (\epsilon_0 \nabla \Phi) = e(n_e - n_i) \, . \tag{1}$$

and the trajectories of each charged particle from Newton's second law

$$\frac{d}{dt}(m\mathbf{v}) = \mathbf{F} = q\mathbf{E} \, , \quad \mathbf{v} = \frac{d\mathbf{x}}{dt} \, . \tag{2}$$

A PIC based model of EP plume interactions requires one to resolve accurately the effects from spacecraft boundary on plasmas. Complex geometries are usually best handled by a body-fit unstructured mesh using tetrahedral elements for example. However, an unstructured mesh based particle code can be computationally expensive compared to a standard Cartesian mesh PIC code. In a standard Cartesian mesh PIC code, the location of memory of quantities defined in neighboring cells can be found trivially via indexing. This is in contrast to an unstructured mesh where the neighbors of a given cell must be found by lookups in a table or other methods requiring additional memory references and computation time. Moreover, for an unstructured mesh, a fairly complex scheme is typically needed to determine a particle's new cell. These added complexities not only makes large-scale 3-D simulations expensive computationally but also complicates the parallel implementation. On the other hand, a finite difference method based field solver using the Cartesian meshes is susceptible of losing accuracy in the fields in the vicinity of a irregular boundary.

The IFE-PIC is a hybrid finite-element and finite-difference particle-in-cell algorithm. In IFE-PIC, the Poisson's Eq. (1) is solved using the recently developed immersed-finite-element method [17,18]. A major feature of the IFE method is that it uses a Cartesian based mesh to solve the electric field in the presence of complex boundaries. This greatly simplifies the search for macro-particle locations and particle-mesh interpolations for PIC. Unlike body-fitted, unstructured mesh-based field solvers, the IFE solver treats the internal bounday as part of the simulation domain and solves the electric field as an "interface" problem. As illustrated in Fig. 1, since the mesh used by IFE-PIC is generated independently of the object boundary (i.e. the interface) inside the simulation domain, some of the cells in the mesh will intersect with the object boundary. Hence, the elements used by IFE include (1) interface elements, i.e. those elements whose interiors are cut through by the object, and (2) non-interface elements.

Following the same procedure applied in standard finite element method, the solution space in IFE method is discretized using a finite number of local basis functions defined on each element of the mesh. In a non-interface tetrahedron, the standard linear local nodal basis functions can be used to span the local finite element space. In an interface element, the physical jump conditions at interface are used to determine the IFE basis function.

If the simulation domain is of rectangular shape, then a 3D Cartesian mesh with brick or tetrahedral

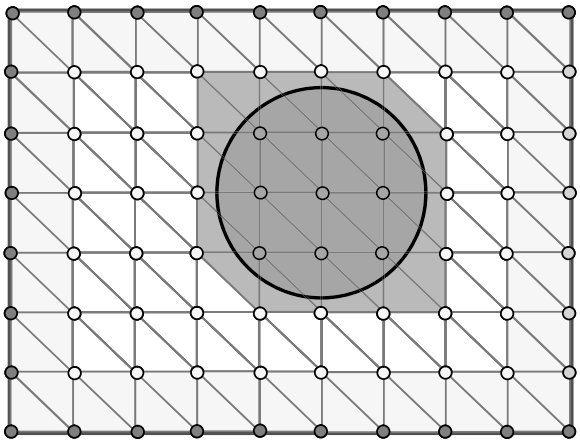

Fig. 1. The Cartesian-tetrahedral mesh used by IFE-PIC (2-D projection). The circular internal object boundary represents an interface surface in the simulation domain. Those cells containing the interface are interface cells. The physical jump conditions at interface are used to determine the IFE basis functions for the two portions of the interface cell.

elements may be naturally chosen for the IFE field solver. Based on the intersection topology of interface cells, we find that a tetrahedral element possesses favorable topological features compared with a brick element while keeping the total number of mesh nodes the same. Hence, the IFE-PIC code presented here uses a Cartesian-tetrahedral mesh, where the IFE field solver uses the tetrahedral elements rather than brick elements. The primary mesh of IFE-PIC is the Cartesian mesh. Each Cartesian cell is further divided into five tetrahedral elements as shown in Fig. 2. The primary Cartesian mesh is used by PIC. The secondary structured tetrahedral mesh is used by the IFE field solver. By investigating the possible intersection topologies of a typical tetrahedral element, we found that there are only two possible intersection topologies if the mesh size is small enough compared to the interface curvature, as shown in Fig. 2. The details of the IFE method, including the construction of the basis functions for the interface elements, are discussed in [17].

The particle push and charge deposit in IFE-PIC are identical to that in a finite difference based PIC. If a particle is located in a non-interface cell, the force interpolation is also identical to that in a finite difference PIC. For all interface cells, an extra search is performed to identify which tetrahedron a particle is located in, and the electric field obtained from the IFE solution for that tetrahedron element is used directly to push particles. The accuracy of the IFE-PIC algorithm has been investigated using various test cases. For instance, the accuracy and the convergence of the IFE field solver are

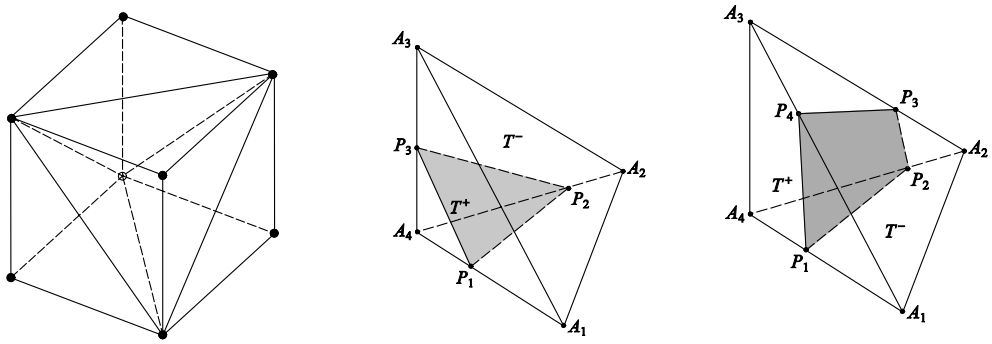


Fig. 2. Partition of a Cartesian cell into five tetrahedra and intersection topology.

discussed in detail in [17] where the IFE solution of the electric field was compared against both the analytical solution and the standard finite element solution.

### 2.2. Parallel implementation

#### 2.2.1. Domain decomposition

The IFE-PIC is parallelized using domain decomposition and the General Concurrent PIC (GCPIC) approach [10]. Each processor is assigned a subdomain and all the particles and grid points in it. When a particle moves from one subdomain to another, it must be passed to the appropriate processors, which requires interprocessor communication. To ensure that the gather/scatter steps can be performed locally, each processor also stores guard cells (neighboring grid points surrounding a processor's subdomain which belong to another processor's subdomain). Interprocessor communication is necessary to exchange guard cell information. The message passing interface (MPI) is used for interprocessor communications.

As the IFE-PIC is based on the use of a Cartesian based tetrahedral mesh and the particles are pushed only in the Cartesian mesh, domain decomposition for IFE-PIC is the same as the standard finite difference based PIC (FD-PIC). The IFE mesh as well as the mesh-object intersections are generated locally for each subdomain. The finite element system is constructed and stored locally on each processor. Figure 3 illustrates domain decomposition of the IFE mesh and node classifications. For the particular application considered in this paper, the computation time is dominated by IFE solution of the electric field (see Section IV). Hence, a domain decomposition of subdomains with equal number of nodes is sufficient to ensure load balance among processors.

#### 2.2.2. Parallel IFE field solver

To account for the assembly of the local finite element system on the elements at subdomain boundaries, the subdomain IFE mesh extends to include the guard cells from neighboring processors. The subdomain IFE mesh nodes are classified as local nodes (the nodes that are assigned to the current processor) and external nodes (the adjacent nodes assigned to neighboring processors). Local nodes are further classified as local internal nodes (the nodes that have no connectivity with external nodes), and local boundary nodes (the nodes that have element connectivity with external nodes). Figure 3 illustrates domain decomposition of the IFE mesh and node classification.

The finite element system is constructed and stored locally on local nodes. The application of the preconditioned-conjugate gradient solver inside the IFE solver requires the parallel implementation of vector-vector inner products and matrix-vector products. The inner products of local vectors are first performed by each processor, and then the local inner products are summed over all processors. The implementation of matrix-vector multiplications is more cumbersome. The local stiffness matrix, or mass matrix, is divided into three sub-matrices: local sub-stiffness matrix (which include the entries associated with interconnectivity of local nodes), external sub-stiffness matrix (which includes entries associated with the connectivity of local boundary nodes with external nodes), and zero matrix (which includes zero entries). During matrix-vector multiplication, the local sub-stiffness matrix is multiplied with the associated local vectors, the external sub-stiffness matrix is multiplied with the vectors associated with external nodes, and the zero matrix is simply ignored. Communication among neighboring processors is required to send information from local boundary nodes, and receive information from external nodes.

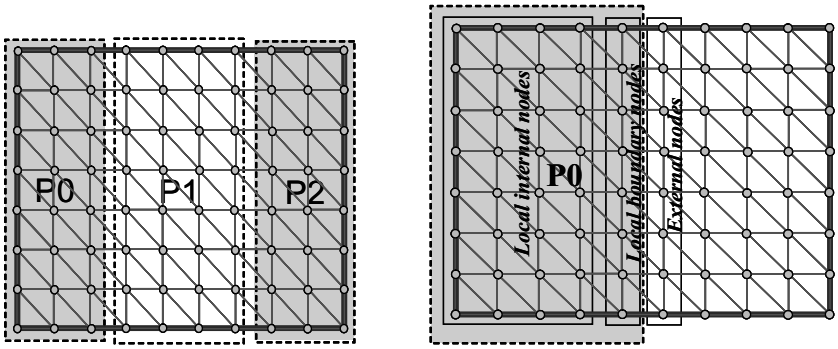


Fig. 3. Domain decomposition of the IFE mesh and node classification.

*2.2.3. Parallel PIC*

The construction of the parallel code includes the development of a parallel IFE field solver, and parallel implementation of the problem setup and PIC components of the legacy sequential PIC codes. The parallel implementation of PIC is carried out using the UCLA Parallel PIC Framework (UPIC) [16]. The UPIC Framework provides the common components that many PIC codes share, making use of object-oriented design in Fortran95. The Framework supports multiple numerical methods, different physics approximations, different numerical optimizations and implementations on different hardware. It is designed to hide the complexity of parallel processing and with "defensive" programming in mind, meaning that it contains many error checks and debugging helps. The UPIC Framework has been used to build a number of new parallel PIC codes. The parallel implementation is done through the merger of the UPIC Framework and the application subroutines. Specific components from UPIC were customized for this new code, primarily those involving management of particles on parallel processors. Only 1-dimensional domain decomposition is implemented in the current version of the code.

## 3. Simulation studies

### *3.1. Ion thruster plume*

In a typical ion thruster, the propellant xenon ions are accelerated through an ion optics grid to achieve a kinetic energy $E_b$ up to 1100 eV (exit beam velocity of $v_b \simeq 3.5 \times 10^6$ cm/s). The temperature of the ions corresponds to the thruster wall temperature, typically $T_w \sim 500$ K (0.04 eV). Outside the thruster, the propellant ions form a "cold", high speed beam with a divergence half angle of about 15 degree to 20 degree due to the curvature of thruster exit surface. The ion beam is kept quasi-neutral by electrons emitted from the neutralizer. Ground measurements show that the electrons near thruster exit typically have a temperature of $T_e \sim 2$ eV.

The propellant that remains un-ionized flows out of the thruster exit in free molecular flow with a thermal speed corresponding to the thruster wall temperature $T_w$. Charge-exchange (CEX) collisions occur between the beam ions and the neutrals, generating slow moving charge exchange ions and fast moving neutrals. CEX ions can backflow within the plume to impinge on thruster components, causing ion sputtering of materials. CEX ions can also be pushed out of the plume and backflow to interact with spacecraft. Hence, an ion thruster plume is composed of propellant efflux (high energy beam ions, neutralizing electrons, and un-ionized neutrals that escaped through the ion optics and from the neutralizer), nonpropellant efflux (material sputtered from thruster components and the neutralizer), and a low-energy charge-exchange plasma generated between the beam ions and the neutrals within the plume.

### *3.2. Full particle simulation of plume in the near thruster region*

We first study ion thruster plume in the near-thruster region. The physical process associated with the plume near thruster exit has the following characteristics: the plasma flow is essentially collisionless (the mean-free path of charged particle collisions is much larger than the thruster dimension), the flow is mesothermal (the ion beam velocity $v_b$ is much larger than the ion thermal speed $v_{ti}$ but is much less than the electron thermal speed $v_{te}$, $v_{ti} << v_b << v_{te}$), and there are strong interactions between electrons and ions, and the local electric field.

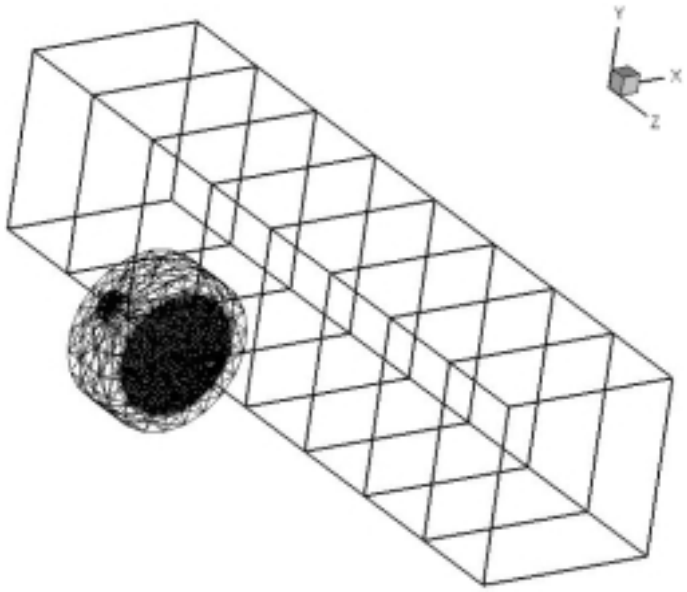


Fig. 4. Setup and domain decomposition for near-thruster plume simulation.

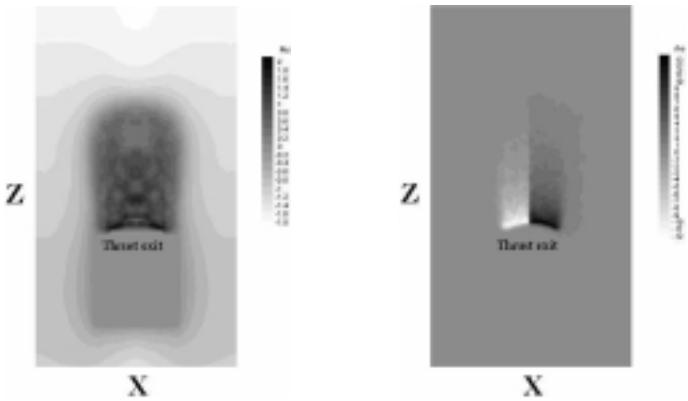


Fig. 5. Normalized potential contours (left) & electron and ion density contours (right). The potential is normalized by Te $= 2$ eV. On the right panel: electron density is shown on the left half and ion density is shown on the right half panel.

Few studies have attempted to simulate this problem using the full particle PIC approach primarily due to computational constraints. This is because, in order to simulate the physics correctly, a full particle PIC simulation must be carried out using an ion to electron mass ratio very close to the real mass ratio (and thus extremely small time steps for ion motion) in order to maintain the correct mesothermal velocity ratio. Moreover, one must also use a sufficiently fine mesh resolution to resolve the plasma Debye length at thruster exit, $\lambda_D$, and a relatively large simulation domain to minimize the effects of the simulation boundary.

The simulation setup is shown in Fig. 4. The thruster exit has a curved beam emission surface with a divergence angle of $\sim 19^\circ$. Due to the symmetry, we only need to simulate $1/4$ of the configuration. The purpose of this simulation is to analyze electron distributions and plume potential in a quasi-neutral plume. Hence, in the simulation, both the ions and electrons are injected into the simulation domain from the beam emission surface at thruster exit. The electrons are assumed to have an initial drifting Maxwellian distribution with a temperature of $T_e = 2$ eV. The ions are assumed to have an initial drifting Maxwellian distribution with a temperature of $T_i = 0.04$ eV. To speed up the simulation, we use the mass ratio of proton to electron, $m_i/m_e \simeq 1836$. Even using the parallel computer, simulations still need to be performed for "scaled-

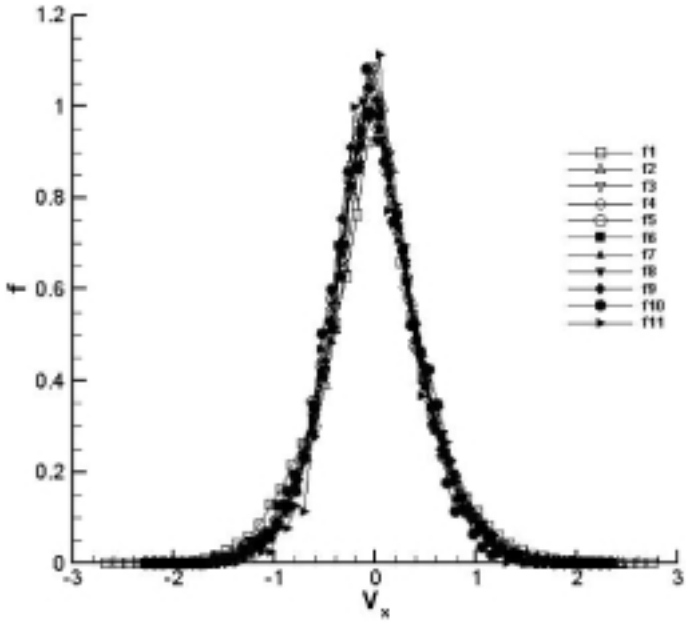


Fig. 6. Electron distribution functions at serveral downstream locations ($V_x$ is normalized by electron thermal speed $(Te/m_e)^{1/2}$.

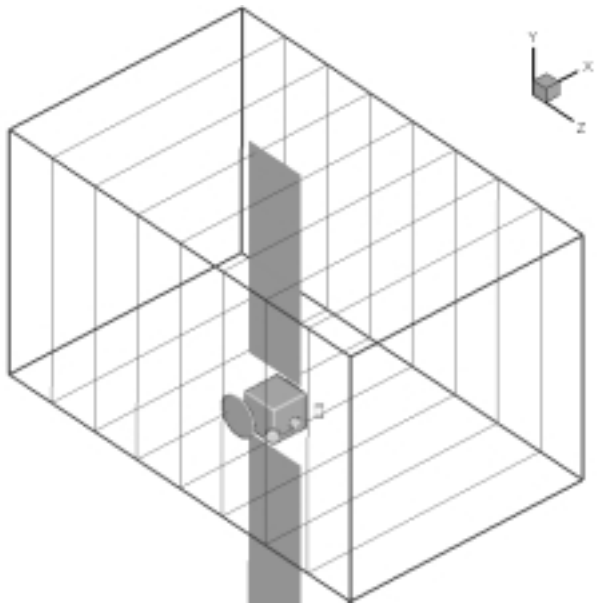


Fig. 7. Setup and domain decomposition for multiple-thruster plume spacecraft interaction simulation.

down" problems. The ion beam exit radius is taken to be $r_b/\lambda_D = 13$ and the thruster body radius is taken to be $r_T/\lambda_D = 15.5$. (If the thruster in the simulation had a real physical dimension of the 30cm diameter NSTAR thruster, the simulation parameters would correspond to a pseudo operating condition that generates a beam ion density of $\sim 10^6$ cm$^{-3}$ at the thruster exit.)

Both the thruster firing direction and domain decomposition is along the z direction. The mesh resolution is taken to be the Debye length $\lambda_D$ at thruster exit. In the simulation presented here, the PIC mesh is taken to be $60 \times 60 \times 256$. The number of micro-particles used is more than 100 million. Simulations are carried out using 8 processor of the JPL Dell Xeon cluster.

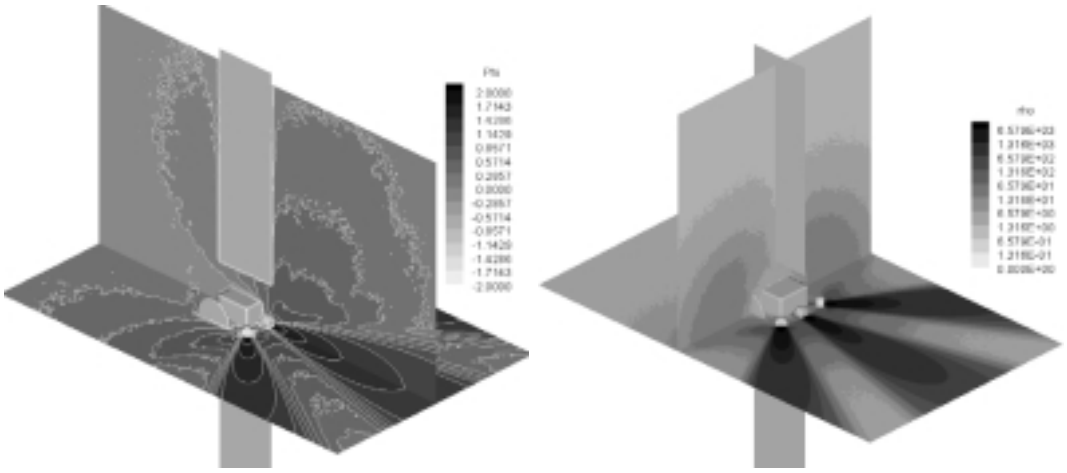

Fig. 8. Normalized potential contours (left) & ion density contours (right). (The unit for potential contour levels is 5 V. The unit for ion density contour levels is $n = 0.76 \times 10^5/\text{cm}^3$.

Simulation results are shown in Figs 5 and 6. Figure 5 shows a snapshot of the potential contours (left panel) and ion and electron density contours (right panel) at the time when the ion beam reaches beyond one thruster diameter. The contours are shown on a z-x plane cutting through the thruster center. The results of Fig. 5 show a neutralized ion beam with electrons confined within the ion beam. As the electrons are much more mobile than the ions due to their large thermal velocity, the plume center has a slightly more positive space charge and a small positive potential. The electron distribution functions for the velocity component $v_x$, $f_e(v_x)$, at different downstream locations from the thruster exit are shown in Fig. 6. The distributions functions are calculated for electrons within every 4 cells along the z direction. Because of the small value of the plume potential, the electrons in a neutralized beam largely maintain their initial distributions as expected.

### *3.3. High resolution simulation of multiple ion thruster plume interactions with spacecraft*

We next study ion thruster plume surrounding the entire spacecraft. Previously, Wang et al. [1] developed a 3-dimensional finite-difference based PIC model for ion thruster plume plasma interactions which showed excellent agreement with in-flight measurements from the Deep Space 1 (DS1) spacecraft. The model presented here adopts the same physics formulation as in [1] and uses a similar simulation setup.

We consider a spacecraft configuration similar to the DAWN spacecraft, shown in Fig. 7. The spacecraft consists of a $1.32 \times 1.32 \times 1.32$ m cube spacecraft bus, a spherical dish antenna, and three cylindrical ion thrusters. The ion thruster considered is taken to be the 30cm diameter NSTAR (NASA Solar Electric Propulsion Technology Application Readiness) ion thruster used on the Deep Space 1 spacecraft. We consider ion thruster plume due to all three thrusters firing simultaneously.

The focus here is to simulate the CEX ion backflow. Hence, only the CEX ions are treated as particles. The density distribution of the propellant ion beam $n_b(\mathbf{x})$ is modeled by an analytical profile of parabolic axisymmetric core and an exponential decay wing. The density distribution of the neutral plume $n_n(\mathbf{x})$ is modeled analytically as that of a free molecular flow from a point source located at one thruster radius $r_T$ behind the thruster exit. The electrons density is modelled by the Boltzmann distribution

$$n_e = n_{b0} \exp\left(\frac{\Phi - \Phi_{p0}}{T_e}\right) \tag{3}$$

where $n_{b0}$ is the average plasma density at thruster exit and $\Phi_{p0}$ is the plume potential near thruster exit. Charge-exchange ions are introduced into the simulation domain according to $n_b(\mathbf{x})$, $n_n(\mathbf{x})$, beam ion velocity $v_b$, and the charge-exchange collision cross section $\sigma_{\text{cex}}$:

$$\frac{dn_{cex}}{dt} = n_b(\mathbf{x}) n_n(\mathbf{x}) v_b \sigma_{\text{cex}} \tag{4}$$

The input parameters for the charge-exchange ion simulation include the average ion beam density $n_{b0}$ and neutral plume density $n_{n0}$ at thruster exit, the potential difference between the plume and spacecraft $\Phi_{p0} - \Phi_{scg}$, the electron temperature in the plume $T_e$,

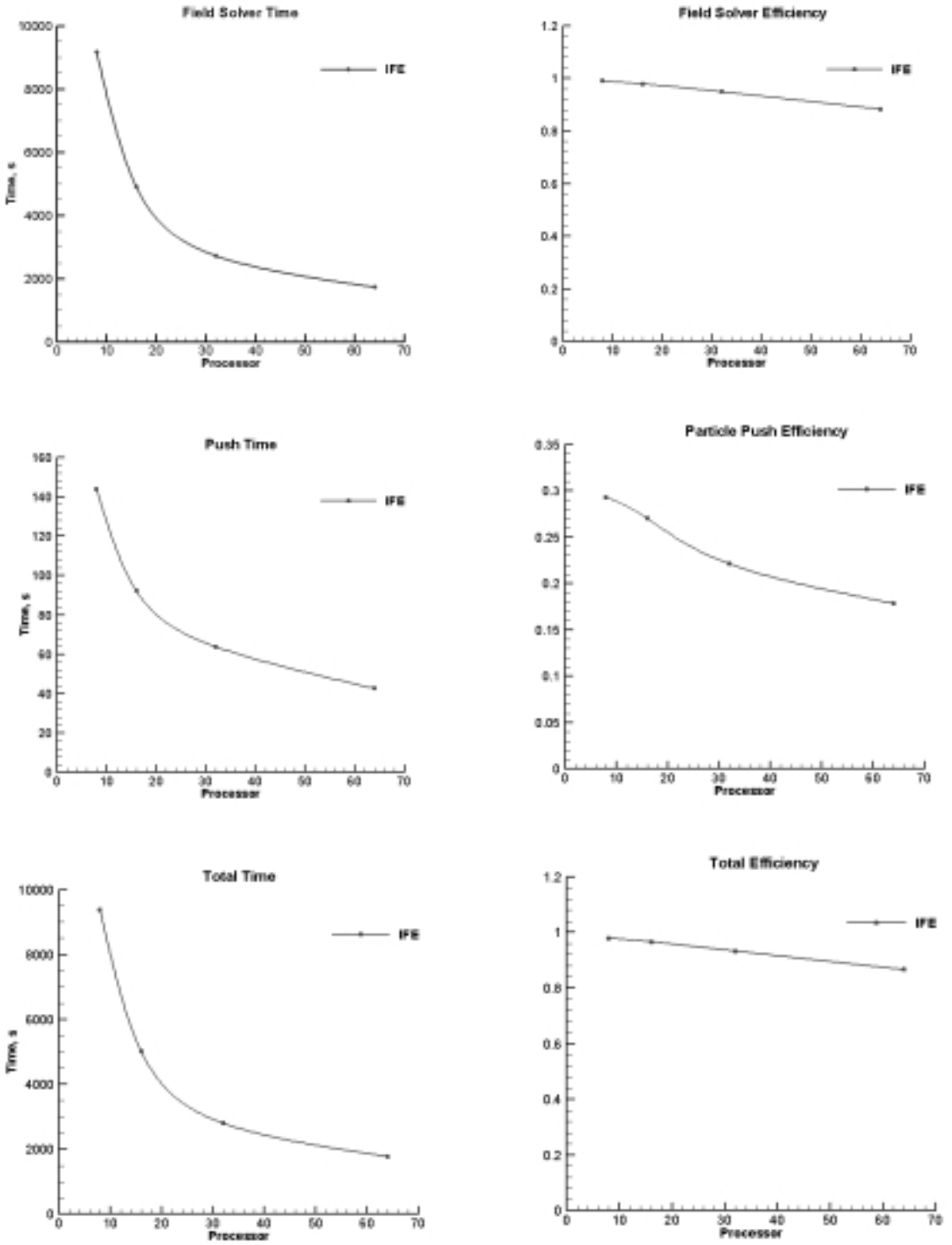


Fig. 9. Performance benchmarks of the paralle IFE-PIC code using 8, 16, 32, and 64 processors of the JPL Dell Xeon cluster. Left column: field solve, particle push, and total time measured for PIC time steps from 151 to 200. Right column: paralled efficiencies of field solve, particle push, and total code performance measured during the same period.

Table 1
Simulation input parameters derived from DS1 data

| | |
|---|---|
| $v_b$ (km/s) | 38.7 |
| $n_{b0}$ ($1/cm^3$) | $3.22 \times 10^9$ |
| $n_{n0}$ ($1/cm^3$) | $0.23 \times 10^{12}$ |
| $\Phi_{p0} - \Phi_{sc}$ (V) | 19 |
| $T_e$ (eV) | 2.09 |

and charge-exchange collision cross section. In our model, $n_{b0}$ and $n_{n0}$ are derived from DS1 in-flight ion engine data. $\Phi_{p0} - \Phi_{scg}$ and $T_e$ are taken to have the values measured by the DS1 in-flight measurement [19]. The spacecraft surface is assumed to have a uniform potential distribution equal to the spacecraft ground $\Phi_{scg}$. The simulation input for the results presented in this section is listed in Table 1.

Due to symmetry, we only need to simulate half of the spacecraft configuration with respect to the x-z plane of the thruster centerline. The simulation presented here is intended to provide a high resolution description of the CEX plasma environment surrounding spacecraft. Hence, the simulation domain is taken to be sufficiently large to enclose the entire solar array panel and the mesh resolution is taken to be sufficiently high to resolve the Debye length of the CEX plasma in the wake region. The size of the entire simulation domain is $9.3 \text{ m} \times 9.3 \text{ m} \times 15.4 \text{ m}$. This domain is decomposed into subdomains along the z direction using 1-D domain decomposition. The PIC mesh is a uniform Cartesian mesh, with cell size taken to be a 6cm cube. The total number of PIC cells in the simulation domain is $155 \times 155 \times 256$ (more than 6.15 million cells). The entire IFE mesh has 30,752,000 tetrahedral elements. The simulation presented here typically uses $\sim 12.5$ million particles. Simulations are performed using 8, 16, 32, and 64 processor of the JPL Dell Xeon cluster.

The steady state potential contours and CEX ion density contours are shown in Fig. 8 on selected 2-dimensional surfaces. Previously, it has been shown that CEX ion backflow is through an expansion process similar to that of a mesothermal plasma expansion into a vacuum [1]. Figure 8 shows the same expansion characteristics. The detailed CEX plume structure is more complex than the single thruster plume. The presence of the antenna dish also enhances plume backflow as the antenna is assumed to have the spacecraft ground potential.

## 4. Performance benchmarks

The Dell Xeon cluster at JPL is used for simulations presented in this paper. This section presents performance benchmarks of this parallel PIC code on this parallel computer. The Dell Xeon cluster has 1024 Intel Xeon processors (3.2 GHz), with 2GB memory per CPU. The total memory of the system is 2TB and the theoretical peak speed is 6.55 TFLOPS. The cluster supports parallel programming with the message passing interface (MPI) and runs the Linux operating system. The performance of the code is measured using "fixed problem size" analysis. We use the high resolution simulation of multiple ion thruster plume discussed in Section 3.2 as an example. We compare the times to run the same size problem on an increasing number of processors.

We measure the total code time per time step loop $T_{\text{tot}}$ as well as the times spent by each major functions of the code. Let us denote $T_{\text{move}}$ , $T_{\text{deposit}}$, $T_{\text{field}}$, as the total time spent by the code on particle move, charge deposit, and field solve respectively. We define the particle push time as

$$T_{\text{push}} = T_{\text{move}} + T_{\text{deposit}} \tag{5}$$

Hence

$$\begin{aligned} T_{\text{tot}} &= T_{\text{move}} + T_{\text{deposit}} + T_{\text{field}} \\ &= T_{\text{push}} + T_{\text{field}} \end{aligned} \tag{6}$$

Since each processor runs the code with slightly different times, the times measured are the maximum processor times among all processors used. Hence, there will be a small difference between the measured $T_{\text{tot}}$ and the value of $T_{\text{move}} + T_{\text{deposit}} + T_{\text{field}}$. Moveover, since the clock calls introduce synchronization, the measured times presented here are slightly longer than the times spent by the code with all subroutine clocks turned off.

The particle move, charge deposit, and field solve functions of the code all require inter-processor communications and related processing. The times spent on these functions represent the overhead for using parallel processing. We denote $T_{\text{push}}^{\text{com}}$ as the times spent on trading particles and exchange guard cells and related inter-processor communication by particle move and charge deposit. We denote $T_{\text{field}}^{\text{com}}$ as the time spent on processing subdomain boundary conditions and related inter-processor communications by field solve. Hence, the total overhead for parallel processing is

$$T_{\text{tot}}^{\text{com}} = T_{\text{push}}^{\text{com}} + T_{\text{field}}^{\text{com}} \tag{7}$$

To measure the communication overhead, we define the parallel efficiency of particle push and field solve as

$$\eta_{\rm push} = \frac{T_{\rm push} - T^{\rm com}_{\rm push}}{T_{\rm push}}, \quad \eta_{\rm field} = \frac{T_{\rm field} - T^{\rm com}_{\rm field}}{T_{\rm field}} \quad (8)$$

and the overall parallel efficiency of the code as

$$\eta = \frac{T_{\rm tot} - T^{\rm com}_{\rm tot}}{T_{\rm tot}} \quad (9)$$

The measured $T_{\rm push}$, $T_{\rm field}$, $T_{\rm tot}$, and $\eta_{\rm push}$, $\eta_{\rm field}$, $\eta$ are shown in Fig. 9. The simulations performed is a full transient to steady state simulation where the number of macro-particles increases gradually as the simulation proceeds and the field is updated at every time step. The times shown are measured during the relative early stage of the simulation, between 151 to 200 PIC steps. (The steady state is reached after 900 steps). During this time period, $T_{\rm field}$ dominates $T_{\rm tot}$ because of the relatively small number of particles inside the simulation domain. The performance of the code and the efficiency of the code improves as more particles are introduced into the simulation domain. We also note that, as the code spends most of its computing time on field solve, load balancing between processors can be achieved simply by assigning equal number of mesh points for subdomains in domain decomposition.

The IFE-PIC runs with high parallel efficiency with $\eta \geqslant 90\%$. The is because the IFE field solver is extremely computational intensive and thus the interprocessor communication time is negligible as compared to the computation time. The performance of the code may be further characterized by particle push time per particle per time step, $t_{\rm push}$, and field solve time per cell per time step, $t_{\rm field}$. On the JPL Dell cluster, once the simulation has reached a steady state (PIC steps 900 through 1000), the IFE-PIC code runs at a speed of $t_{\rm push} \simeq 156$ ns/particle/step and $t_{field} \simeq 4690$ ns/cell/step using 64 processors.

## 5. Summary and conclusions

In summary, a parallel, three-dimensional electrostatic PIC code is developed for large-scale simulations of electric propulion plume spacecraft interactions using parallel supercomputers. This code uses a newly developed IFE-PIC algorithm. The IFE-PIC algorithm is designed to handle complex boundary conditions accurately while maintaining the computational speed of the standard PIC code. To demonstrate the effectiveness of parallel computing, the parallel IFE-PIC is used to perform full particle PIC simulation of near-thruster plume and high-resolution simulation of multiple ion thruster plume interactions with a realistic spacecraft. The performance benchmarks of the parallel IFE-PIC is measured on the Dell Xeon cluster. We find that the IFE-PIC runs with a high parallel efficiency for electric propulsion simulations. For example, a simulation of electric propulsion plume interactions was performed for a large domain enclosing the entire spacecraft and solar array ($9.3\,{\rm m} \times 9.3\,{\rm m} \times 15.4\,{\rm m}$) with a high resolution ($dx \simeq 6$ cm) using more than 6.1 million PIC cells, 30.7 million finite elements, and 12.5 million microparticles. Such a simulation can be completed in a little over 2 hours using 64 processors on the Dell cluster at JPL at a parallel efficiency of 90%. This suggests that large-scale PIC simulations can be applied effectively to solve engineering problems in electric propulsion.

## Acknowledgments

We acknowledge many useful discussions with Charles Norton of JPL and Tao Lin of Virginia Tech. This research is supported by contracts from Jet Propulsion Laboratory and Air Force Research Laboratory. Access to the Dell Xeon parallel cluster is provided by the JPL Supercomputing Project.

## References

[1] J. Wang, D. Brinza and M. Young, Three-dimensional particle simulation modeling of ion propulsion plasma environment for deep space 1, *Journal of Spacecraft and Rockets* **38**(3) (2001), 433–440.

[2] R.I. Samanta Roy, D.E. Hastings and N.A. Gatsonis, Ion-thruster plume modeling for backflow contamination, *Journal of Spacecraft and Rockets* **33**(4) (1996), 525–534.

[3] R.I. Samanta Roy, D.E. Hastings and S. Taylor, Three-dimensional plasma particle-in-cell calculations of ion thruster backflow contamination, *Journal of Computational Physics* **128** (1996), 6–18.

[4] D.B. VanGilder, G.I. Font and I.D. Boyd, Hybrid monte carlo–particle-in-cell simulation of an ion thruster plume, *Journal of Propulsion and Power* **15**(4) (1999), 530–538.

[5] I. Boyd, Review of hall thruster plume modeling, *Journal of Spacecraft and Rockets* **38**(3) (2001), 381–387.

[6] S.D. Fergason, Ion thruster plume simulation using clustered PC workstations, in: *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Tucson, Arizona, July 2005, pp. AIAA 05–966.

[7] M.J. Mandell, I.G. Mikellides, L.K. Johnson and I. Katz, A high power ion thruster plume model, in: *40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Fort Lauderdale, Florida, July 2004, pp. AIAA. 04–3816.

[8] M.J. Mandell, R.A. Kuharski, B.M. Gardner, I. Katz, T. Randolph, R. Dougherty and D.C. Ferguson, Ion engine plume interaction calculations for prototypical prometheus 1, in: *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, Tucson, Arizona, July 2005, pp. AIAA 05–4047.

[9] C.K. Birdsall and A.B. Langdon, *Plasma Physics Via Computer Simulation*. The Institute of Physics, 1991.
[10] P. Liewer and V. Decyk, A general concurrent algorithm for plasma particle-in-cell simulation codes, *Journal of Computational Physics* **85** (1989), 302–322.
[11] J. Eastwood, W. Arter, N. Brealey and R. Hockney, Body-fitted electromagnetic pic software for use on parallel computers, *Computer Physics Communications* **87** (1995), 157–178.
[12] J. Verboncouer, A. Langdon and N. Gladd, An object-oriented electromagnetic pic code, *Computer Physics Communications* **87** (1995), 199–211.
[13] J. Wang, P. Liewer and V. Decyk, 3d electromagnetic plasma particle simulations on a mimd parallel computer, *Computer Physics Communications* **87** (1995), 35–53.
[14] J. Wang, P. Liewer and E. Huang, 3-d particle-in-cell with monte carlo collision simulations on three mimd parallel supercomputers, *Journal of Supercomputing* **10** (1997), 331–348.
[15] J. Wang, D. Kondrashov, P. Liewer and S. Karmesin, 3-d deformable grid electromagnetic particle-in-cell for parallel computers, *Journal of Plasma Physics* **61** (1999), 367–389.
[16] V. Decyk and C. Norton, The ucla parallel pic framework, *Computer Physics Communications* (2004), 164.
[17] R. Kafafy, T. Lin, Y. Lin and J. Wang, Three dimensional immersed finite element methods for electric field simulation in composite materials, *International Journal for Numerical Methods in Engineering* **64** (2005), 940–972.
[18] T. Lin, Y. Lin, R. Rogers and M.L. Ryan, *Advances In Computation: Theory And Practice*. Commack, NY: Nova Science Publishers, Inc., 2001, ch.: A rectangular immersed finite element space for interface problems, 107–114.
[19] J. Wang, D. Brinza, M. Young, J. Nordholt, J. Polk, M. Henry, R. Goldstein, J. Hanley, D. Lawrence and M. Shappirio, Deep space 1 investigations of ion propulsion plasma environment, *Journal of Spacecraft and Rockets* **37**(5) (2000), 545–555.