Ion Propulsion Simulations Using Parallel Supercomputer

IEPC-2005-271

Presented at the 29th International Electric Propulsion Conference, Princeton University October 31 – November 4, 2005

J. Wang*, Y. Cao†, R. Kafafy‡, and J. Pierru§
Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

V. Decyk¶
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. Two algorithms are implemented in the code, a standard finite-difference (FD) PIC and a newly developed immersed-finite-element (IFE) PIC. The IFE-PIC is 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 high-resolution simulation of multiple ion thruster plume interactions for a realistic spacecraft using a domain enclosing the entire solar array panel. The second is a full particle simulation of ion beam neutralizer interactions using real ion to electron mass ratio. The IFE-PIC and FD-PIC run with similar overall speed and the IFE-PIC achieves a high parallel efficiency of $\geq 90\%$.

I. Introduction

There have been significant progress in modelling and simulation studies in support of electric propulsion research activities in recent years. There are two classes of models. The first class of models is system level engineering tools designed to produce quick estimations. The second class of models is first-principle based simulation models which attempts to simulate the detailed physics using fundamental physics laws with few assumptions. Particle-in-Cell (PIC) has emerged as the most appropriate algorithm for first-principle based modelling of many ion propulsion problems. Numerous particle simulation models have been developed.

A PIC code models a plasma as many macro-particles and follows the evolution of the orbits of individual test particles in the self-consistent electromagnetic field[1,5].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. The computational time/cost and computer memory size restricts the time scale, spatial scale, and number of particles that can be used in a simulation. Due to computational limitations, particle simulation studies of electric propulsion are typically performed using relatively small simulation domains with simplified spacecraft or thruster configurations. Hence, existing particle simulation models of electric propulsion are mostly used as research tools rather than engineering tools.

Recent advances in massively parallel supercomputers have provided computational possibilities that were previously not conceivable. While parallel computing is increasingly being adopted in simulation studies, it

*Associate Professor, Department of Aerospace and Ocean Engineering, jowang@vt.edu.
†Postdoctoral Fellow, Computational Advanced Propulsion Lab
‡Postdoctoral Fellow, Computational Advanced Propulsion Lab
§Graduate Research Assistant, Computational Advanced Propulsion Lab
¶Research Scientist, Department of Physics

has rarely been used in electric propulsion simulations due to specific difficulties associated with the parallel implementation.

Most electric propulsion simulation studies concern electrostatic problems with complex internal boundary conditions imposed by the thruster or spacecraft *inside* the simulation domain. A major challenge in applying parallel computing in electric propulsion simulation is to develop a highly efficient electrostatic PIC code involving complex boundary conditions. Previously, various 3-D electrostatic and electromagnetic PIC codes have been implemented on parallel supercomputers. Domain decomposition is typically used to divide the computations among processors. To minimize the overhead of inter-processor communications, almost all parallel PIC code are designed to use a *local*, *non-iterative* method to solve the electric or electromagnetic fields, such as the fast Fourier transform (FFT) method or the finite-difference time-domain (FDTD) method or discrete-volume time-domain method. High parallel efficiencies have been demonstrated for PIC codes using such field solution techniques. For instance, Wang et al have developed parallel 3-D electromagnetic PIC codes using FDTD[11] and discrete-volume time-domain with non-orthogonal grids[13] and achieved a parallel efficiency of $\geq 95\%$ for both codes using 512 processors. However, a *local*, *non-iterative* field solve method cannot be used in electric propulsion simulations because of the elliptical nature of the Poisson's equation and the complex internal boundaries. Rather, one must solve the Poisson's equation using a *global*, *iterative* method. An unstructured mesh is also typically needed to body fit the object boundary. Developing such a field solver with high parallel efficiency has been a difficult issue in the parallel implementation of electrostatic PIC codes.

This paper presents a particle simulation code developed for large-scale, 3-dimensional electric propulsion simulations using parallel supercomputers. The parallel code presented here incorporates a newly developed immersed-finite-element (IFE) algorithm[6]. The IFE method allows one to use a structured mesh, even a Cartesian mesh, to solve the electric field involving arbitrarily shaped boundary conditions at an accuracy comparable to that of the standard finite element method using a body-fit unstructured mesh. This approach retains the computing speed of a standard PIC code without losing accuracy as well as allows an easy implement on parallel computers using domain decomposition. For comparison, a standard finite-difference (FD) algorithm is also implemented in the PIC code. We show that both the FD based and IFE based parallel PIC codes run with a high parallel efficiency.

We present two ion propulsion simulation studies in this paper. The first study is a large-scale 3-D simulation of multiple ion thruster plumes. Computational constraints have limited existing plume simulations to simple spacecraft configuration and small simulation domain. The simulation presented here considers a realistic spacecraft configuration, modelled after the DAWN spacecraft, with three ion thrusters. Simulation is performed using a domain sufficiently large to enclose the entire spacecraft and the solar array panel with a high mesh resolution to resolve the Debye length near the solar array panel. The second study is a preliminary simulation of near-thruster plume and ion beam neutralization. Due to computational constraints, almost all PIC models for ion propulsion use the hybrid approach where only the ions are treated as particles and electrons are simplified using various assumptions. Computational costs have prohibited ion propulsion simulations using the full particle approach with real ion to electron mass ratio. As a result, the interaction between ion beam and neutralizer electrons has not been modelled in detail and the physics underlying the ion beam neutralization process is still not well understood. The simulation presented here is one of the first attempts to resolve the ion beam neutralizer interactions. It treats both ions and electrons as marco-particles, uses the real ion to electron mass ratio, and runs the simulation based on electron time scales.

Section II briefly discusses the algorithm and parallel implementation. Section III presents results from the two simulation studies. Section IV presents performance benchmarks of running the parallel code on a Dell cluster. Section V contains a summary and conclusions.

II. A Parallel PIC Code for Ion Propulsion

A. Algorithm

The basic function of a PIC code is to solve the electrostatic field self-consistently with the boundary condition 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

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

As particle can be located anywhere in the simulation domain while the field quantities are only defined on mesh points, a PIC also involves two particle-mesh interpolation steps, a gather step to interpolate field quantities from mesh points to particle location and a scatter step to deposit particle charge to mesh points. Depending on specific applications, one may choose to use either the full particle approach where both the electrons and ions are treated as macro-particles or a hybrid approach where the electron density n_e is obtained by solving the fluid equation or other simplifications. A commonly used approximation in ion thruster plume simulation model is to use the Boltzmann approximation $n_e \simeq n_o \exp((\Phi - \Phi_o)/T_e)$ for electrons. *Both* the full particle PIC and the hybrid PIC with Boltzmann electrons are implemented in the code presented here.

A major challenge in the application of PIC codes on engineering problems, including electric propulsion, is the boundary condition involving complex plasma-material interface. Complex geometries are usually best handled by a body-fit mesh using tetrahedral cells or unstructured meshes. However, a tetrahedral cell based or unstructured mesh based particle code can be significantly computationally more expensive than a standard orthogonal mesh PIC code. In a standard orthogonal 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 either tetrahedral cells or unstructured meshes, a fairly complex scheme is typically needed to determine a particle's new cell. These added complexities can make large-scale 3-D simulations prohibitively expansive computationally[13]. 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.

Recently, a new PIC algorithm based on the use of an immersed-finite-element (IFE) formulation was developed for PIC simulations involving complex plasma-material interface[6,7] The primary attraction of the IFE formulation is that it allows one to use a numerical mesh without consideration of the interface location. In particular, a Cartesian mesh can be used in the IFE space to solve a problem involving a complicated interface. The IFE-PIC algorithm uses a structured Cartesian-tetrahedral mesh (Figure 1). The Cartesian mesh is the primary mesh used by PIC. Each Cartesian cell is further divided into five tetrahedral elements. The tetrahedral mesh is the secondary mesh used only by the IFE field solver. The IFE-PIC algorithm has been applied in simulation studies of whole subscale ion optics[7] and ion thruster plumes[15]. It is shown that the IFE-PIC approach has the capability to handle complex boundary conditions accurately while maintaining the computational speed of a standard PIC code. *Both* the newly developed IFE-PIC algorithm and the standard finite-difference based PIC algorithm (FD-PIC) are implemented in the code presented here.

B. Parallel Implementation

Both the IFE-PIC and FD-PIC algorithms are parallelized using domain decomposition using the General Concurrent PIC (GCPIC) approach[8] 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 can be performed locally, each processor also stores guard cells (e.g. 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 GCPIC approach has been applied in various large-scale PIC simulations with high parallel efficiency (see, for example, Wang et al[11,12,13] and references therein). 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 FD-PIC. The message passing interface (MPI) is used for interprocessor communications.

For the specific applications considered in this paper, we utilize two legacy sequential PIC models developed by Wang for electric propulsion applications. The first model is a 3-D hybrid PIC model of ion thruster plume developed for the Deep Space 1 mission[14]. This model has shown excellent agreement with in-flight measurements from Deep Space 1. The incorporation of the IFE formulation into the model have significantly increased the model's capability on handling complex spacecraft configuration. The second

model is a 3-D full-particle PIC model developed to study ion and electron beam emissions[9,10].This model setup is modified to study near-thruster plume interactions and ion beam neutralizations in this paper.

1. Parallel IFE Field Solver

The parallel implementation of the IFE field solver is significantly more involved than the finite difference based field solver. In the code, the IFE mesh as well as the mesh-object intersections are generated *locally* for each sub-domain. 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 2 illustrates domain decomposition of the IFE mesh and node classification.

The finite element system is constructed and stored *locally*, i.e. 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 all 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 inter-connectivity 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). Figure 3 illustrates these subdivisions of the stiffness matrix. 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.

2. Parallel PIC

The construction of the parallel code includes the development of a parallel IFE field solver, a parallel FD 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)[4]. 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 software from the UPIC Framework and software specifically written by Wang and students for Electric Propulsion. 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.

III. Ion Propulsion Simulations Using Parallel Computer

The section presents results of two ion propulsion simulations from using the parallel PIC code. The first study concerns the induced plasma environment for a spacecraft with multiple ion thrusters. Simulations are performed for a sophisticated spacecraft configuration and using a large simulation domain enclosing the entire solar array panel with a high mesh resolution. The second study concerns near-thruster plumes and the beam neutralization processes. Simulations are performed using the real ion to electron mass ratio. Both simulations are almost impossible to carry out on regular computers due to the memory and computational time required. Results presented here are obtained from simulations on a scalable distributed-memory parallel processor system, the Dell Xeon cluster at the Jet Propulsion Laboratory. The section presents simulation results.

A. High-Resolution Simulations of Multiple Ion Thruster Plume Interactions

We first consider ion thruster plume interactions for a spacecraft with multiple thrusters. The IFE-PIC version of the code is used in this study to accurately resolve the boundary conditions at the spacecraft surface. The model formulation is the same as that described in Wang et al.[14]. As the focus here is the charge-exchange ion backflow, only the charge-exchange (CEX) ions are treated as particles. The electrons density is modelled by the Boltzmann distribution. The density distribution of the propellant beam ions $n_b(x)$ and the neutrals $n_n(x)$ are modelled by analytical profiles. Charge-exchange ions are introduced into the simulation domain according to n_b, n_n, beam ion velocity v_b, and the charge-exchange collision cross section σ_{cex}:

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

.

The spacecraft is taken to have a configuration similar to the DAWN spacecraft, a cube with a side length of about 1.3m plus a spherical antenna dish, a solar array, and some payloads, and three ion thrusters. The ion thrusters are assumed to be the 30cm diameter NSTAR ion thruster. Similar to [Wang et al., 2001], the input parameters for charge-exchange simulation include the ion beam density and neutral plume density at thruster exit (derived from DS1 ion engine operating condition for ML 83), the potential difference between the plume and spacecraft ground and the electron temperature in the plume (assumed to have the values measured by DS1 IDS instrument), and charge-exchange collision cross section. The spacecraft surface is assumed to have a uniform potential distribution equal to the spacecraft ground. Details of these values are given in Wang et al.[14].

The simulations presented here is intended for a follow up study to model ion thruster plume interactions with the solar array. 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 spacecraft model and the simulation mesh are shown in Figure 4. Due to the symmetry, we only need to simulate of the spacecraft configuration. Domain decomposition is along the z direction. The PIC mesh is a uniform Cartesian mesh of cells. Each PIC cell is further divided into 5 tetrahedral elements in the IFE field solver. The size of the simulation domain is $9.3m \times 9.3m \times 15.4m$. Based on our previous simulations of the NSTAR ion thruster plume, the PIC cell size is 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 12.5 million particles.

We have performed this simulation using 8, 16, 32, and 64 processor of the JPL Dell Xeon cluster. The total computational time required for this simulation depends on the number of processors used and how the simulation is performed. For instance, a full transient to steady state simulation resolving CEX plasma propagation from the start of thruster firing requires field update at every PIC step. Such a simulation takes about 10hrs on 64 processors for 1000 PIC steps (Steady state is reached at about 900 PIC steps). On the hand, if one is only interested in the steady state result, one could choose to perform an accelerated simulation with field update at intervals of PIC steps. Such a simulation could finish in a little more than 2hrs on 64 processors instead. More performance benchmarks will be discussed in the next section.

The steady state potential contours and CEX ion density contours for three-thruster firing are shown in Figure 5. Due to the large ion density variations, the CEX ion density contours are shown only for the range up to $1.5 \times 10^6 cm^{-3}$. It was shown that CEX backflow is through an expansion process similar to that of a mesothermal plasma expansion into a vacuum[14]. Results presented here show the same expansion characteristics although the detailed CEX plume structure is more complex than single thruster firing. Additionally, the presence of the antenna dish enhances the backflow, as the antenna is assumed to have the spacecraft ground potential.

B. Full Particle Simulations of Ion Beam Neutralizer Interactions

We next discuss a preliminary simulate of ion thruster plume in the near-thruster region. As the focus here is the detailed electron dynamics and ion electron mixing process, we shall only include a simplified thruster in the simulation domain. The FD-PIC version of the code is used in this study.

Although ion beam neutralization is readily achieved in experiments, our understanding of the underlying physics has remained mostly at the level of "electrons are attracted to ions by the Coulomb's force". The ion

beam neutralization process not only is an interesting physics problem but also has practical implications in electric propulsion. For instance, an understanding of ion beam neutralization is important in the design of ion thruster clusters with a single shared neutralizer as well as in understanding of the effects of cathode plume model operation. Very few studies have attempted to simulate this problem primarily due to computational constraints[2,3,16]. This is because, in order to simulate the physics correctly, such simulations 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) and using a very large simulation domain to minimize the effects of the simulation boundary.

The simulation setup is shown in Figure 6. Both the thruster firing direction and domain decomposition is along the z direction. The thruster exit has a curved beam emission surface with a divergence angle of $\sim 19^o$. We consider two different simulation cases. In case 1, we assume that the ions and the electrons have already mix properly. The purpose of case 1 is to analyze the electron distributions in a neutralized ion beam. In case 1, both the ions and electrons are injected into the simulation domain from the beam emission surface at thruster exit with a drifting Maxwellian distribution. In case 2, we attempt to simulate the ion and electron mixing process. Hence, in case 2, the electrons are injected into the simulation domain from the neutralizer cathode with a Maxwellian distribution. To minimize the effects of the boundary, the thruster is placed in the middle of the simulation domain with a 0 volt potential. There is no potential difference between the thruster body and the neutralizer. Electron and ion mixing occurs entirely through the electric field of the space charge. Due to symmetry, we only need to simulate 1/4 of the configuration in case 1 and 1/2 of the configuration in case 2.

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 $T_i = 0.1$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-down" problems. The mesh resolution is taken to be the Debye length $\lambda - D$ at the thruster exit. The PIC mesh for case 1 is taken to be $60 \times 60 \times 256$. The PIC mesh for case 2 is taken to be $121 \times 60 \times 256$. 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 NSTAR thruster exit.)

As ion beam neutralizer interaction occurs very close to the thruster exit, we concentrate on the region within one thruster diameter. Results for case 1 are shown in Figures 7 and 8. In Figure 7, we show the 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. For comparison On the right panel of Figure 7, the electron density is plotted on the left half of the panel while the ion density is plotted on the right half of the panel. Results of Figure 7 shows a nicely neutralized ion beam with electrons confined within the ion beam. The electron and ion positions on a z-x plane cutting through the thruster center are shown in the left panel of Figure 8. The electron distribution functions for the velocity component v_x, $f_e(v_x)$ at different downstream locations from the thruster exit are shown on the right panel of Figure 8. The distributions functions are calculated for electrons within every 4 cells along the z direction. Figure 8 shows that the electrons in a neutralized beam maintain their initial distributions. Results for case 2 are shown Figures 9 through 11. As this simulation is extremely time consuming, the results shown is for a very early stage of ion thruster firing (within a transient time period when the beam ions only reach about half the thruster diameter). Figure 9 shows the potential contours and charge density contours (only plotted for positive charge). Figure 10 shows the electron and ion positions where the mixing of electrons and ions is evident. Figure 11 shows the electron distribution functions for the velocity components v_x and v_z at different downstream locations from the thruster exit. Again, the distribution functions are calculated for electrons within every 4 cells along the z direction. Figure 11 clearly shows that electrons undergo acceleration and heating during the neutralization process.

IV. 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, shown in Figure 12, has 1024 Intel Pentium 4 Xenon processors (3.2GHz), 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, where we compare the times to run the same size problem on an increasing number of processors. The multiple thruster plume simulation discussed in Section III-A is used for this analysis. For comparison, we have run the same problem using both the IFE-PIC and FD-PIC versions of the code. The simulations are run using 8, 16, 32, and 64 processors.

To analyze the performance of the code, we have measured the total code time per time step loop T_{tot} as well as the times spent by each major functions of the code. Let us denote T_{move} , $T_{deposit}$, T_{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

$$T_{push} = T_{move} + T_{deposit} \tag{3}$$

Hence

$$T_{tot} = T_{move} + T_{deposit} + T_{field} = T_{push} + T_{field} \tag{4}$$

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_{tot} and the value of $T_{move} + T_{deposit} + T_{field}$. 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^{com}_{push} 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^{com}_{field} 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

$$T^{com}_{tot} = T^{com}_{push} + T^{com}_{field} \tag{5}$$

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

$$\eta_{push} = \frac{T_{push} - T^{com}_{push}}{T_{push}}, \quad \eta_{field} = \frac{T_{field} - T^{com}_{field}}{T_{field}} \tag{6}$$

and the overall parallel efficiency of the code as

$$\eta = \frac{T_{tot} - T^{com}_{tot}}{T_{tot}} \tag{7}$$

The measured T_{push}, T_{field}, and T_{tot} are shown on the left column of Figure 13 and the η_{push}, η_{field}, and η are shown on the right column. The simulations performed is a fully transient plume simulation where the number of macro-particles increases gradually as the simulation progresses 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_{push} dominates T_{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.

Both the IFE-PIC and the FD-PIC runs at similar speed. Comparing the IFE solver and the FD solver, the FD solver spends less time on computations but more time on inter-processor communications. For this particular application, the IFE-PIC runs with high parallel efficiency with $\eta \geq 90\%$. The performance of the code may be further characterized by particle push time per particle per time step, t_{push}, and field solve time per cell per time step, t_{field}. Once the simulation has reached a steady state (PIC steps 900 through 1000), the IFE-PIC code runs at a speed of $t_{push} \simeq 156$ ns/particle/step and $t_{field} \simeq 4690$ns/cell/step using 64 processors.

V. Summary

In summary, a parallel, three-dimensional electrostatic PIC code is developed for large-scale electric propulsion simulations using parallel supercomputers. Two algorithms are implemented in the code, a

standard FD-PIC and a newly developed IFE-PIC. The IFE-PIC is designed to handle complex boundary conditions accurately while maintaining the computational speed of the standard PIC code. Two ion propulsion simulation studies are carried out using the parallel PIC code. In the first simulation, the parallel IFE-PIC is used to study multiple ion thruster plume interactions with a realistic spacecraft. Simulation is performed using a domain sufficiently large to enclose the entire spacecraft and the solar array panel with a high mesh resolution to resolve the Debye length near the solar array panel. In the second simulation, the parallel FD-PIC is used to study ion beam-neutralizer interactions. Full particle simulations are performed using the real ion to electron mass ratio. Performance benchmarks of both the IFE-PIC and FD-PIC are measured on the Dell Xenon cluster. Both the IFE-PIC and FD-PIC runs at similar overall speeds for the same problem. In particular, we observe that the IFE-PIC runs with a high parallel efficiency of $\geq 90\%$, a particle push speed of ~ 156ns/particle/step, and a field solve speed of ~ 4690ns/cell/step on 64 processors on the Dell cluster.

Acknowledgments

We acknowledge many useful discussions with L. Johnson and C. Norton of JPL, L. Brieda and D. VanGilder of AFRL, and T. Lin of Virginia Tech. This research is supported by research grants from Jet Propulsion Laboratory and Air Force Research Laboratory at Edwards, AFB. Access to the Dell Xeon parallel cluster was provided by the JPL Supercomputing Project.

References

[1] C.K. Birdsall and A.B. Langdon, *Plasma Physics via Computer Simulation*, McGraw-Hill, New York, 1985.

[2] L. Brieda and J. Wang, "Modeling of Ion Thruster Beam Neutralization Using a Fully Kinetic ES-PIC Code", AIAA 2005-4045, *41st Joint Propulsion Conference*, Tucson, Arizona, July, 2005.

[3]L. Brieda, D. VanGilder, J. Wang, "Modeling Ion Beam Neutralization and Near-Thruster Plume Interactions", IEPC 2005-270, *29th International Electric Propulsion Conference*, Princeton, NJ, Oct. 2005.

[4]V. Decyk and C. Norton, "The UCLA Parallel PIC Framework", *Comp. Phys. Comm.*, 164, 2004.

[5] R. Hockney and J. Eastwood, *Computer Simulation Using Particles*, McGraw-Hill, New York, 1981.

[6]R. Kafafy, T. Lin, Y. Lin, and J. Wang. "3-D Immersed Finite Element Methods for Electric Field Simulation in Composite Materials". *International Journal for Numerical Methods in Engineering*, 64, 2005, pp940972.

[7]R. Kafafy and J. Wang, "Whole Subscale Ion Optics Simulations Using a Hybrid Grid Immersed-Finite-Element Particle-in-Cell Code", submitted to *Journal of Propulsion & Power*, 2005.

[8]P. Liewer and V. Decyk, "A General Concurrent Algorithm for Plasma Particle-in-Cell Simulation Codes", *J. Computational Physics*, 85, 1989, pp302-322.

[9]C. Marrese, J. Wang, K. Goodfellow, and A. Gallimore, "Space-Charge Limited Emission from Field Emission Array Cathodes for Electric Propulsion and Tether Applications", Chapter 18 in *Micropropulsion for Small Spacecraft*, *Progress in Aeronautics & Astronautics*, V187, AIAA, 2000.

[10] J. Wang and S. Lai, "Virtual Anode in Ion Beam Emissions in Space: Numerical Simulations", *J. Spacecraft & Rockets*, 34(6), 1997, pp829–836.

[11]J. Wang, P. Liewer, and V. Decyk, "3D Electromagnetic Plasma Particle Simulations on a MIMD Parallel Computer", *Comput. Phys. Comm.*, 87, 1995, pp35–53.

[12]J. Wang, P. Liewer, and E. Huang, "3-D Particle-in-Cell with Monte Carlo Collision Simulations on Three MIMD Parallel Supercomputers", *J. Supercomputing*, 10, 1997, pp331–348.

[13]J. Wang, D. Kondrashov, P. Liewer, and S. Karmesin, "3-D Deformable Grid Electromagnetic Particle-in-Cell for Parallel Computers", *J. Plasma Physics*, 61(3), 1999, pp367–389.

[14]J. Wang, D. Brinza, and M. Young, "Three-Dimensional Particle Simulation Modeling of Ion Propulsion Plasma Environment for Deep Space 1", *J. Spacecraft & Rockets*, 38(3), 2001, pp433–440.

[15]J. Wang and R. Kafafy, "A HG-IFE-PIC Model for Ion Thruster Plume Interactions", *9th Spacecraft Charging Technology Conference*, Tsukuba, Japan, 2005.

[16]A. Wheelock, D. Cooke, and N. Gatsonis, "Computational Analusis of Current Coupling of Ion Beam-Neutralizer Interactions", AIAA 2005-3692, *41st Joint Propulsion Conference*, Tucson, Arizona, July, 2005.

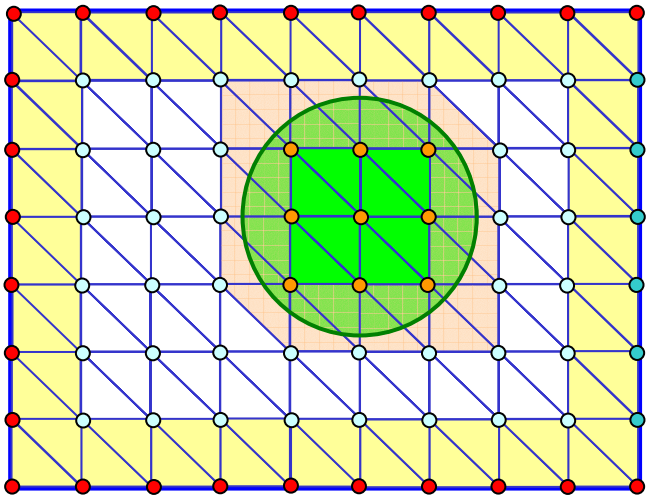


Fig. 1: The Cartesian-tetrahedral mesh used by IFE-PIC

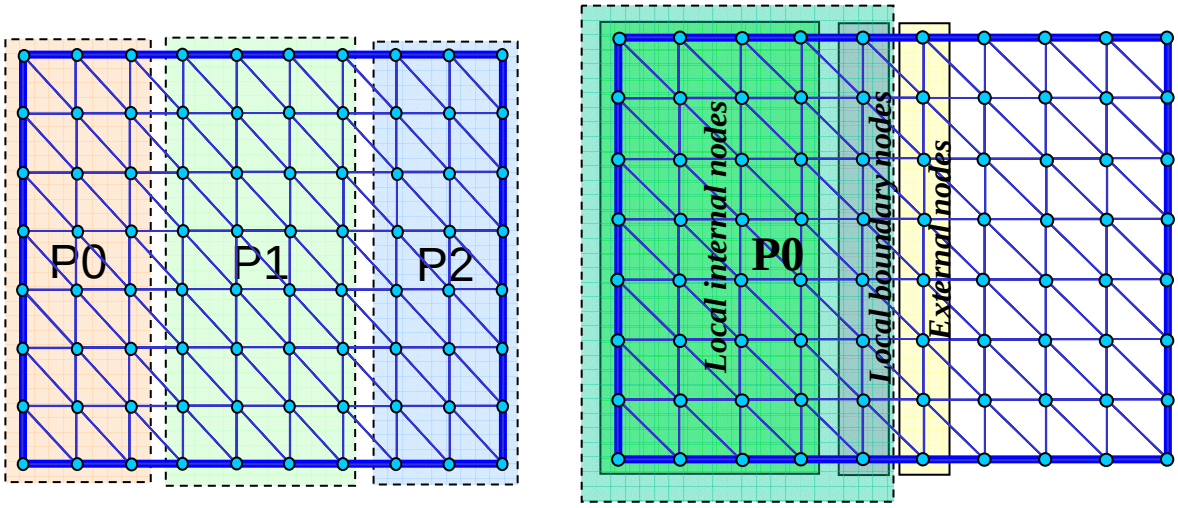


Figure 2: Domain decomposition of the IFE mesh and node classification

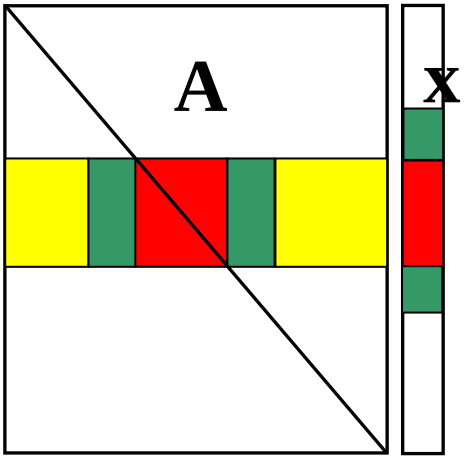


Figure 3: Illustration of subdivisions of the stiff matrix and matrix-vector multiplications

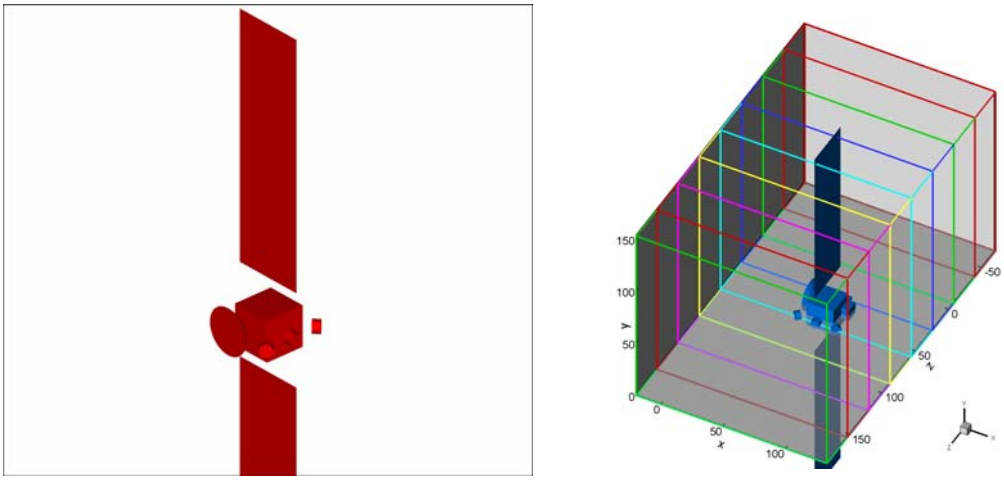


Figure 4: Spacecraft model and domain decomposition for multiple-thruster plume simulations

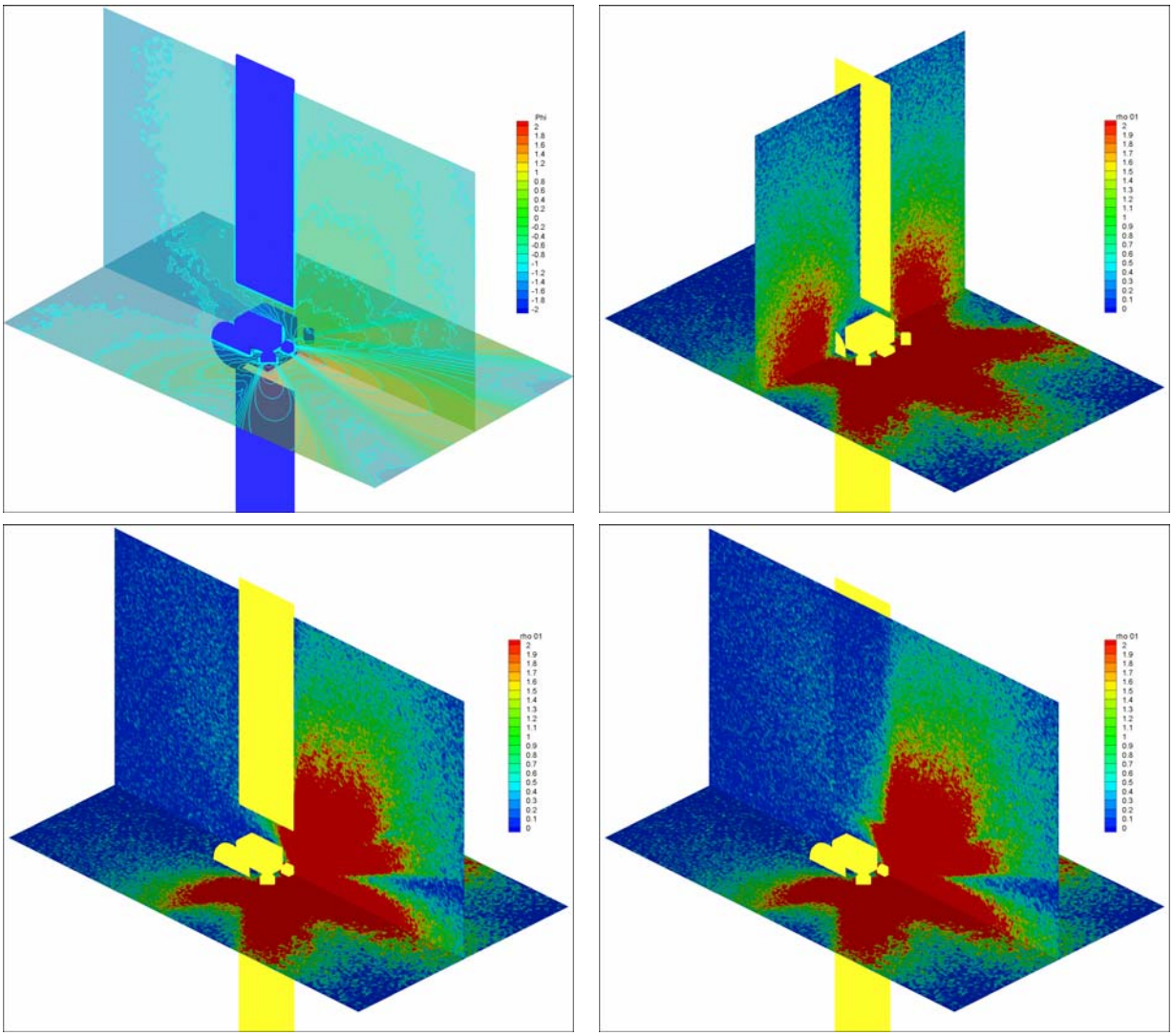


Figure 5: Normalized potential & ion density contours (Three Thruster Firing). The potential is normalized by Te=5V. The ion density is normalized by $n=0.75 \times 10^5$ /cm^3

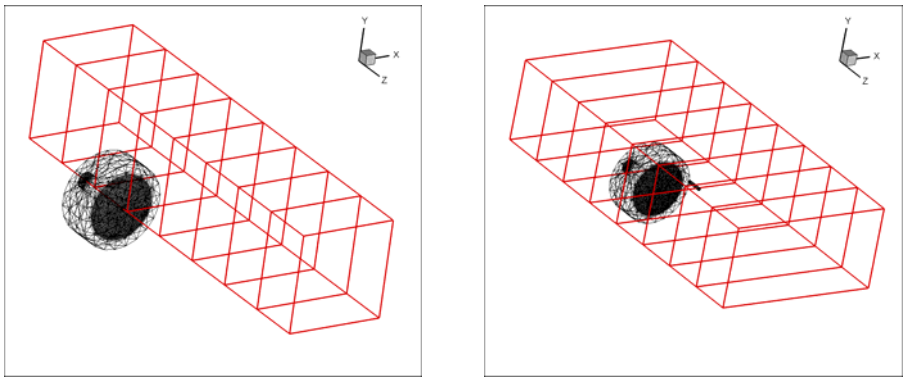


Figure 6: Setup and domain decomposition for near-thruster plume simulation case 1 (left) and case 2 (right)

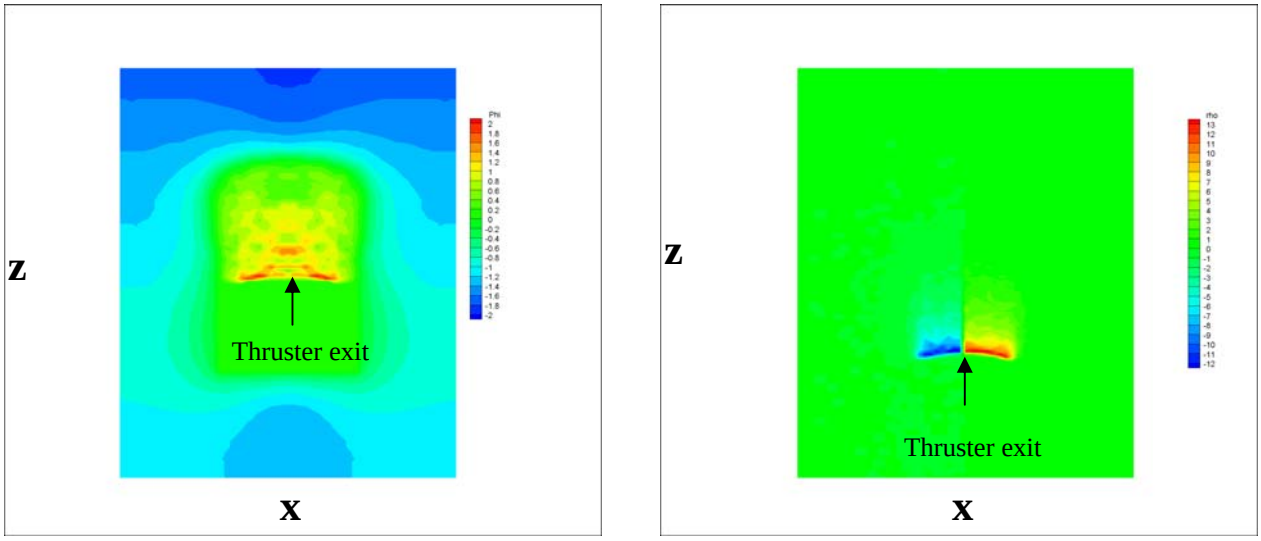


Figure 7: Normalized potential contours (left) & electron and ion density contours (right) for case 1. The potential is normalized by Te=2eV. On the right panel: electron density is shown on the left half and ion density is shown on the right half panel.

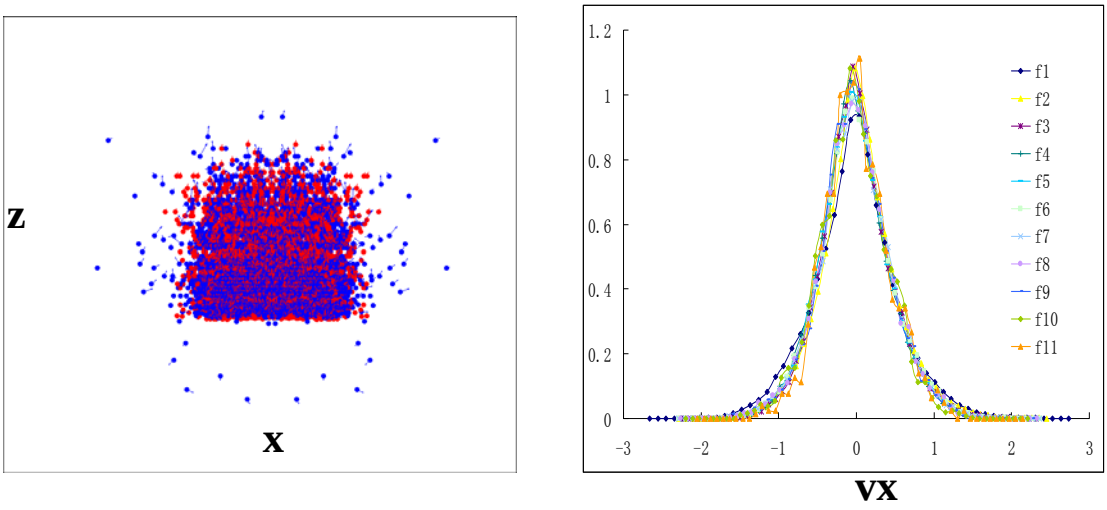


Figure 8: Left: Ion (red) and electron (blue) positions; Right: electron distribution functions for case 1. Vx is normalized by electron thermal speed $(Te/m_e)^{1/2}$

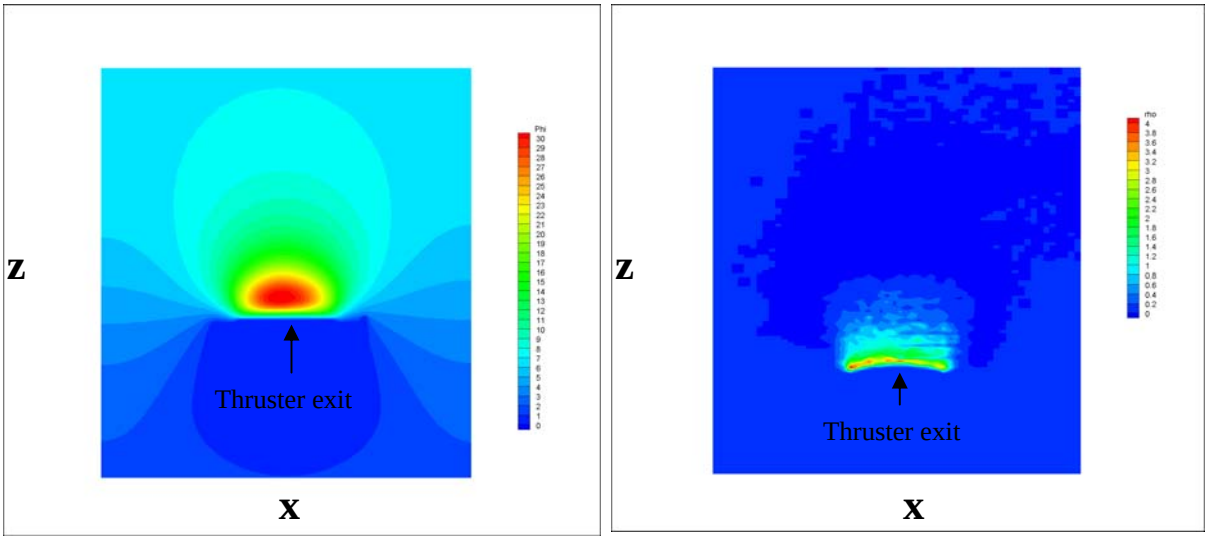


Figure 9: Potential contours (left) & charge density contours (right) for case 2. The potential is normalized by Te=2eV. The charge density on the right panel is only shown for positive values.

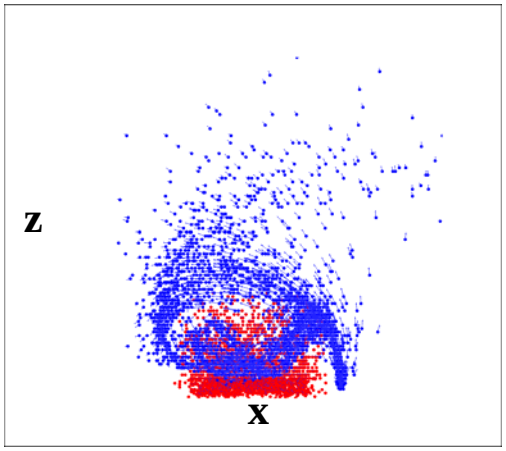
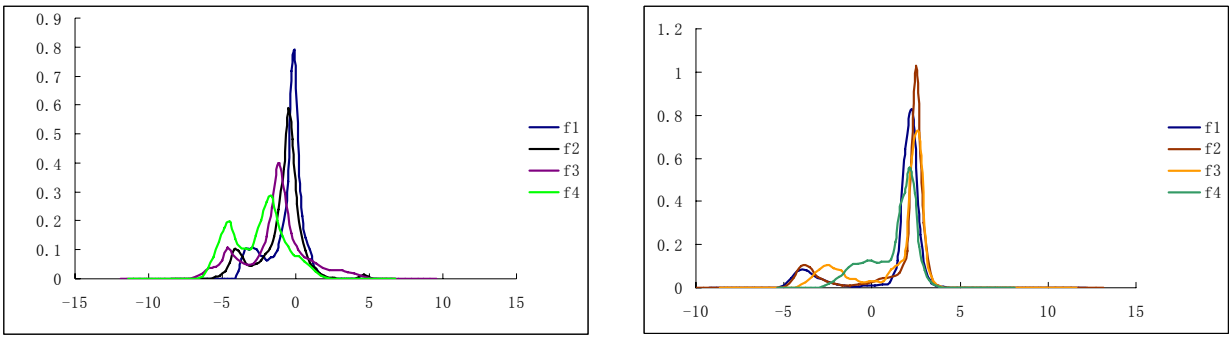


Figure 10: Ion (red) and electron (blue) positions for case 2



VX VZ

Figure 11: Electron distribution functions for case 2. Vx and Vz are normalized by electron thermal speed $(Te/m_e)^{1/2}$

Figure 12: The Dell Xeon cluster at JPL

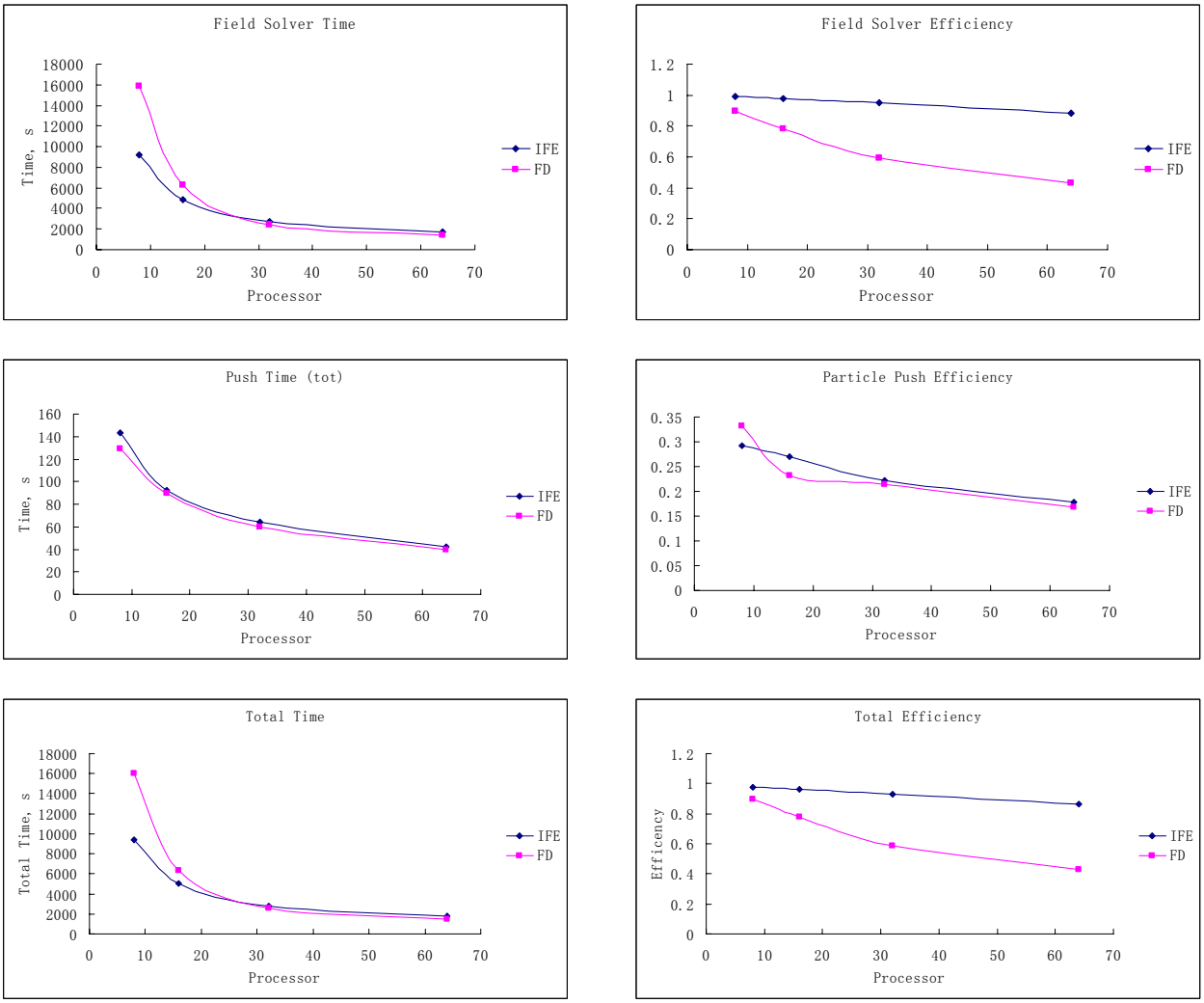


Figure 13: Performance of the parallel IFE-PIC and FD-PIC codes for simulation of the same multiple thruster plume case using 8, 16, 32, and 64 processors of the JPL Dell Xeon cluster. Left column: field solve, particle push, and total time measured for PIC time steps from 151 to 200. Right column: parallel efficiencies of field solve, particle push, and total code performance measured during the same period.