DEVELOPMENT OF A VIRTUAL TESTING LABORATORY FOR SPACECRAFT-PLASMA INTERACTIONS

J. Wang
Department of Aerospace and Ocean Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0203
Phone: 540-231-8114
E-mail: jowang@vt.edu

L. Brieda
R. Kafafy
J. Pierru
Virginia Polytechnic Institute and State University

Introduction

Modeling and simulation are playing an ever more important role in space environments and effects research activities. In recent years, the sophistication of the models and the capability of state-of-the-art supercomputers have reached such a level that in many cases one can produce simulation results that are in good quantitative agreement with experimental data. For example, the 3-D ion thruster plume model described in Wang et al.[2001] produced results that are in excellent agreement with data taken from Deep Space 1 in-flight measurements. The recent advance in simulation capabilities suggests that "virtual" experiments using first-principle based models may be used to replace real experiments to quantify spacecraft environment interactions for many applications.

This paper discusses a prototype virtual testing laboratory for spacecraft plasma interactions and electric propulsion. As illustrated in Fig. 1, this virtual laboratory consists of a simulation engine and a virtual testing environment. A user provides physical parameters and spacecraft configurations to the simulation engine. The virtual laboratory simulates the physical processes using a set of particle simulation codes and then displays the results in a virtual testing environment using immersed and/or collaborative visualization.

In order to develop a virtual laboratory based on first principle based simulations, one must overcome at least two major challenges. First, one needs to be able to build up a code that is sophisticated enough so the complex geometry associated with a satellite can be modeled properly and yet computationally efficient enough so large-scale 3-D particle simulations can be performed routinely. Second, one needs to be able to quickly transform "data rich" simulation results to "information rich" for engineering applications. Two research activities at Virginia Tech were carried out to address these two issues. First, a new particle simulation code, DRACO, was developed for plasma simulations involving complex boundary conditions. A unique feature of DRACO is that incorporates the recently developed immersed finite element particle-in-cell (IFE-PIC) algorithm. This method allows one to use a Cartesian mesh to handle complex geometric or time-varying interface between plasma and object without scarifying the accuracy in electric field solutions. The computational speed of an IFE-PIC simulation is about

the same as that of standard PIC simulation. Second, a new visualization and data analysis tool, capVTE, was developed for visualizations using both virtual reality environments and regular desktop/laptop machines. capVTE offers immersed visualization for users with access to virtual reality environments as well as a shared, collaborative environment that allows remotely connected users to interact with each other over the same data objects. Both DRACO and capVTE are cross platform. DRACO and capVTE form the foundation for a virtual testing laboratory.

This paper is organized as follows: section II discusses the development of DRACO; section III discusses the design of capVTE; section IV presents examples of spacecraft-electric propulsion plume interactions visualized through the desktop interface of capVTE (examples of immersed visualization are shown in the presentation); Section V contains a conclusion.

3-D Particle Simulation Engine: DRACO

Overview

The simulation engine of the virtual testing laboratory is a multi-purpose particle simulation software, DRACO. DRACO is designed to perform first-principle based, high fidelity simulations of spacecraft-plasma interactions. DRACO runs on UNIX, Linux, and Window. Many of DRACO's subroutines are based on 3-D plasma particle simulation codes previously developed by J. Wang to simulate ion thruster plume interactions [Wang et al., 2001] and ion optics plasma flow [Wang et al., 2003a]. The plume simulation model by Wang et al.[2001] produced results that are in excellent agreement with data from Deep Space 1 in-flight measurements and the ion optics model by Wang et al.[2003a] produced results that are in excellent agreement with data taken during the long duration tests of the NSTAR thruster. Additionally, DRACO also includes a new particle simulation algorithm based on the immersed finite element (IFE) formulation [Wang and Lin, 2003; Wang et al., 2003b], IFE-PIC.

As shown in Fig.1, DRACO allows a user to choose from three simulation modules. In the QN-PIC module, the plasma is assumed to be quasi-neutral and the electric field is obtained by assuming the electron density follows the Boltzmann distribution. The QN-PIC module is intended for quick calculations and cannot be used to resolve the plasma sheath. The DADI-PIC module is a standard PIC code using a finite-difference formulation to solve the electric field. DADI-PIC is designed for problems with relatively simple geometric conditions. Out of considerations for computational efficiency for large-scale simulations, DADI-PIC uses a Cartesian mesh. To resolve the geometry associated with a curved surface, a method of sub-grid scale placement of boundaries is used. This method explicitly includes the location of object surface in relation to the computational mesh so that the placement of the object boundary is not restricted to the mesh points. The electric field is obtained by solving the Poisson's equation using a dynamic alternating direction implicit (DADI) method [Doss et al.,1979; Hewett et al.,1992] with a defect correction using the Douglass-Gunn operator splitting [Douglas and Gunn, 1964]. This DADI method was chosen over other algorithms for its increased stability properties over fully explicit methods and its relatively simple tridiagonal system of equations produced by the partially implicit nature of the method. This DADI-PIC model was discussed in [Wang et al, 2001].

The IFE-PIC is DRACO's most sophisticated module. The IFE-PIC is based on a finite element formulation, and is designed to perform simulations accurately for problems involving complex geometric and material boundary conditions. Instead of using a complex mesh to body-fit the body surface, the IFE method uses a structured mesh without consideration of the object surface location. Hence, one may use the standard, Cartesian mesh based method for particle-mesh interpolations and pushing particles even in simulations involving complex geometric boundaries. This allows IFE-PIC to retaining the computation speed of a standard particle-in-cell code [Wang et al., 2003b]. The next section presents a brief overview of he IFE-PIC model.

As the purpose of DRACO is to solve real engineering problems, DRACO reads in spacecraft configurations defined by commercial CAD tools. A Mesh Generator module is developed as the interface between CAD tools and the PIC code. The outputs from CAD tools are stored in an ANSYS file. The Mesh Generator then performs the "intersection" operation to identify the intersections by an object of arbitrary geometrical shape onto the computational mesh.

Immersed Finite Element Particle-in-Cell (IFE-PIC)

A PIC code typically spends a significant portion of its computing time performing particle-mesh interpolations and pushing particles. Out of consideration for speed, standard PIC codes are based on the use of structured, Cartesian mesh so the location of memory of quantities defined in neighboring cells can be found trivially via indexing. However, a Cartesian mesh based field solver is susceptible of losing accuracy when a irregular boundary is involved. To solve the electric field accurately for complex geometries, one would typically need to use an unstructured mesh to body-fit the boundary surface. However, an unstructured grid based PIC code can be significantly more expensive computationally because it requires additional memory references (e.g. lookups in a table) to find neighboring cells and a complex scheme to push particles [Westermann, 1992; Wang et al., 1999]. Hence, accuracy and computing speed often represent conflicting requirements for a PIC code.

We recently developed a new, 3-dimensional PIC algorithm using the recently developed immersed finite element (IFE) method [Ewing et al., 1999; Lin et al., 2000] to solve the electric field. Rather than treating the object surface as a boundary condition, this method includes the object material as part of the solution domain and solves the original boundary value problem as an "interface" problem. For instance, consider the electric field in a composite domain consisting of two sub-domains each occupied by a different type of material (Fig. 2). These two sub-domains are separated by a curved surface. The electric potential is described by a boundary value problem:

$$-\nabla \cdot (\sigma \nabla \Phi) = f(\Phi)$$

together with the boundary conditions and the jump conditions on the interface. The media property is described by a material dependent coefficient $\sigma(X)$ (i.e. permittivity). Mathematically, $\sigma(X)$ is a piecewise constant function defined by:

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

To solve the interface problem above, one may treat Γ either as a boundary to the sub-domains Ω^+ and Ω^-, or as an interface inside the entire domain Ω. The IFE method treats Γ as an interface and uses the finite element formulation to solve $\Phi(X)$ over domain Ω.

The essence of the IFE method is that its mesh can be formed without consideration of the interface location. While this "interface" concept is the same as that used in the immersed/imbedded boundary techniques developed for CFD, the IFE method itself is significantly different because a) the IFE method is based on the finite element formulation and b) the trial functions used in the IFE method are constructed only using physics based jump conditions at the interface.

The IFE-PIC model uses a structured Cartesian-tetrahedral mesh. The Cartesian mesh is the primary mesh used by PIC. Each Cartesian cell is further divided into five tetrahedral elements as shown in Fig. 3. The tetrahedral mesh is the secondary mesh used only by the IFE field solver. When a curved object surface is present, the IFE mesh will include both interface cells (those cells that have at least one edge whose interior intersects with the interface) and non-interface cells. In a non-interface cell, the standard linear local nodal basis functions can be used to span the local finite element space. In an interface cell, the physical jump conditions at interface are used to determine the basis function. For instance, consider the interface cell shown Fig.3. The interface divides a typical interface tetrahedron T, with vertices $A_i, (i = 1,2,3,4)$, into T^+ and T^-. This partition of T can be used to introduce four piece-wise linear local nodal basis functions $\Psi_i(x)$:

$$\Psi_i(\vec{x}) = \begin{cases} \Psi_i^+(\vec{x}) = a_0 + a_1 x + a_2 y + a_3 z, \vec{x} \in T^+, \\ \Psi_i^-(\vec{x}) = b_0 + b_1 x + b_2 y + b_3 z, \vec{x} \in T^-. \end{cases}$$

The coefficients in the linear basis functions are determined by the following physical jump conditions to be satisfied by the solution:

1. Continuity on the plane *EFGH:*

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

2. Flux continuity across the plane *EFGH*:

$$\int_{EFG} (\sigma^+ \frac{\partial \Psi_i^+}{\partial n} - \sigma^- \frac{\partial \Psi_i^-}{\partial n}) dS = 0, i = 1,2,3,4$$

Satisfying these conditions provides eight equations for each local basis function which are enough to uniquely determine that basis function. Further details of the IFE method are discussed in [Wang and Lin, 2003; Wang et al., 2003b], where it is also shown that the IFE field solver possesses a second order convergence.

The Virtual Testing Environment: capVTE

Overview

Large-scale, 3-D simulation models generate a wealth of data output. However, in order to apply these models as engineering design tools, one must be able to quickly transform "data rich" simulation results to "information rich" results. To achieve this goal, one needs a data analysis environment which would satisfy at the minimum the following requirements: 1)allow easy visualization and interpretation of complex multi-dimensional data generated by particle or CFD codes; 2)enable interactive access to information from different geographic locations and online collaborations; and 3)be machine independent. A data analysis tool that satisfies these requirements in essence becomes a "virtual" testing environment.

The CAP Lab Virtual Testing Environment (capVTE) is a cross-platform visualization and data analysis tool [Brieda et al., 2003]. Visual analysis can be performed on a wide variety of hardware platforms, ranging from high-end PCs, through SUN and SGI workstations, to virtual reality facilities such as CAVE. In addition to standard graphics capability visualizing multi-dimensional data set, capVTE offers immersed visualization for users with access to virtual reality environments as well as a shared, collaborative environment which allows remotely connected users to interact with each other over the same data objects.

The source code of capVTE was designed to be machine independent so that it may be run from any computing platforms. The graphic engine used by capVTE is the open-source Visualization Toolkit (VTK) library by Kitware and QT by Trolltech. Both VTK and QT are platform independent. VTK interacts with the OpenGL package, and versions of VTK are available for Windows, UNIX/Linux and Mac OS X. QT is the basis of GUI, which is used as the user interface for capVTE. QT is used to handle issues associated with cross-platform development, such as interaction with windows, user input, and the creation of timers. Therefore, deployment of capVTE for a particular platform involves only linking the source code with the VTK and QT libraries made for that specific platform.

The user interface of capVTE, as shown in Fig. 4, is a multi-window GUI application. The user navigates through each data set in real-time using the mouse. Menus and toolbars are used to select the active tools and to change their properties. The data input consists of a single ASCII file which contains the appropriate header, followed by program commands and data blocks. However, a data set is allowed to span over several physical files, in a fashion similar to that of C/C++'s *#include* directive. The main input file can be a just a short collection of statements pointing at a geometry file created from an output generated by 3D object modeling tools, and the grid and particle files, generated by particle codes or CFD codes.

CapVTE allows a data export in three formats: JPEG, VRML (Virtual Reality Markup Language), and Inventor. JPEG is used to save images from static snapshots. Both VRML and Inventor are used for interactive visualization of 3-dimensional images. As VRML is commonly used to publish 3D data on the Internet and plugins are available for the popular browsers, capVTE uses the VRML format for online collaborative visualization through network. Inventor is compatible with DIVERSE (Device Independent Virtual Environment Reconfigurable

Scalable Extensible), a collection of Application Programming Interfaces (APIs) developed at Virginia Tech to run the virtual reality facility CAVE. CapVTE uses the Inventor format for immersive visualization.

Immersive visualization

Immersed visualization currently is performed using the DIVERSE software to drive a variety of immersed visualization equipment such as the CAVE (Cave Automatic Virtual Environment), Immersa Desk, Head Mounted Display (HMD), etc. DIVERSE can also be used to drive visualization on conventional desktop or laptop computers.

The CAVE is a multi-person, room-sized, high-resolution, 3D video and audio environment used as a "virtual reality theater". The CAVE facility at Virginia Tech, shown in Fig. 5, has a size of 73.5x84 inch footprint and a 104-inch high ceiling. The three side faces along with the floor act as screens, on which an image is projected from four Electrohome Marquis 8000 projectors. The false-color images are shifted such that, when viewed using the CrystalEyes goggles, they create a perception of the visualized object floating in the center of the cube.

Inside the CAVE, one "walks" into an image. Navigation is through a tracking headset. Its position and view angle is tracked by an array of sensors. The central computer then adjusts the projections to create a realistic image. Further control can be accomplished using a hand-held wand. Its translation and rotation are both tracked, and thus the wand can act as a virtual pointer. A small joystick located on the wand allows the user to travel greater distances than that allowed by the limited physical size of the CAVE.

Collaborative visualization

The real-time data immersion capability of capVTE allows the creation of a collaborative virtual environment on the internet. capVTE includes a network module, based on a client/server and TCP/IP architecture. The network module simplifies data sharing and allows interactive visualization by remotely connected users. The user specifies whether a new server should be established, or whether the program should connect as a client to an already running collaboration. Each user specifies an "avatar", a small graphical object, which is located and oriented according to that user's viewpoint. Thus, every member of the online collaboration can see what part of the dataset that the others are looking at, as illustrated in Figure 4. The network is also partially synchronized. When one client toggles cutting planes, for instance, cutting planes will be toggled for all other clients as well. Thus, all members share the same virtual space with the only difference being the position from which the view is made.

Simulation Examples

To demonstrate the capability of the virtual testing laboratory, this section shows one simulation example of ion thruster plume spacecraft plasma interaction. Other simulation examples are shown in the presentation. We consider a model spacecraft shown in Fig. 6. The spacecraft geometry is composed of a box bus, a thruster, an antenna dish, and a flat solar array orientated at an angle with respect to the thrust direction. Fig. 7 shows the primary Cartesian

mesh used by DRACO and the intersection of spacecraft with the computation mesh. The thruster is taken to be the 30cm NSTAR ion thruster used on Deep Space 1. Simulation setup and thruster operating conditions are the same as that discussed in Wang et al.[2001]. We assume that the solar array surface, spacecraft bus, and antenna are at the same potential.

Simulation results using the DADI-PIC and IFE-PIC module are shown in Figs 8 and 9, respectively. To further demonstrate the effect of solar array on charge-exchange ion backflow, the normal direction of the solar array in the DADI-PIC run is taken to be –45 degree with respect to z while that in the IFE-PIC run is taken to be +45 degree. The DADI-PIC run uses a computation mesh of 91x41x71 with a mesh resolution of 5cm. The number of micro-particles used is more than 5 million. The primary Cartesian mesh used in IFE-PIC run is 60x26x46 with a mesh resolution of 7.5cm. The number of micro-particles used is more than 2.3 million. Comparing Figures 8 and 9, one finds that both runs show similar results in the downstream region of the thruster. However, the upstream region shows significant difference due to the different solar array orientation angle. It is interesting to observe that, in the DADI-PIC run, the solar panel accelerates the backflow charge exchange ions and produces a significant plasma wake behind the antenna dish.

Summary and Conclusions

In conclusion, we have developed a prototype virtual testing laboratory for spacecraft plasma interactions. This virtual laboratory uses a simulation engine based on a set of plasma particle simulation codes, DRACO, and a virtual testing environment, capVTE. A major feature of DRACO is that it includes a new particle simulation algorithm IFE-PIC. The IFE-PIC allows one to use directly the standard Cartesian mesh based algorithms for particle push regardless of the geometry of object boundary. Hence, simulations involving complex boundary conditions can still be performed at a computational speed close to that of a standard PIC code. The interface formulation used by the IFE method to solve the electric field allows material properties to be included explicitly and maintains the desired physics at a given material interface. capVTE enables a quick transform of "data rich" simulation results to "information rich" ones through immersive and/or collaborative visualization. The immersed and shared visualization environment driven by capVTE also provides an virtual experimental platform. Both DRACO and capVTE are cross platform. Currently, the virtual laboratory only concerns spacecraft-plasma interactions. Future work will expand the virtual testing laboratory into other areas of spacecraft-environmental interactions.

Acknowledgments

The development of the DRACO code is supported by Air Force Research Laboratory at Edwards AFB through a grant from ERC Inc.

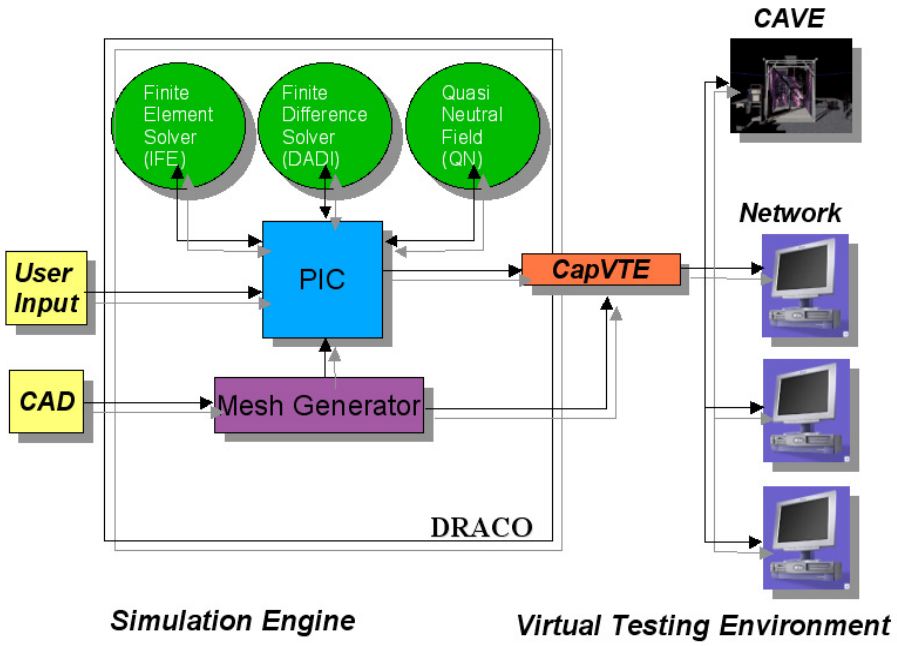

Figure 1. Virtual Testing Laboratory block diagram

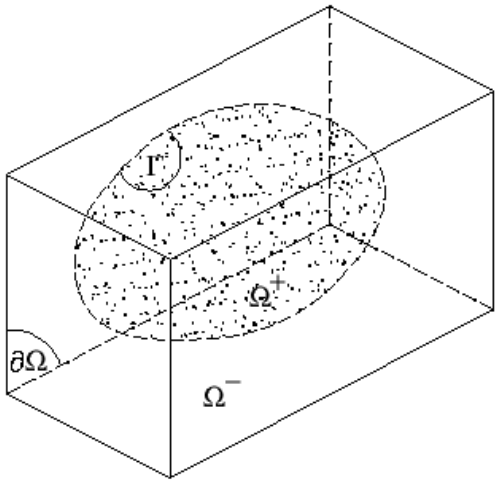

Figure 2. The 3-D interface problem domain

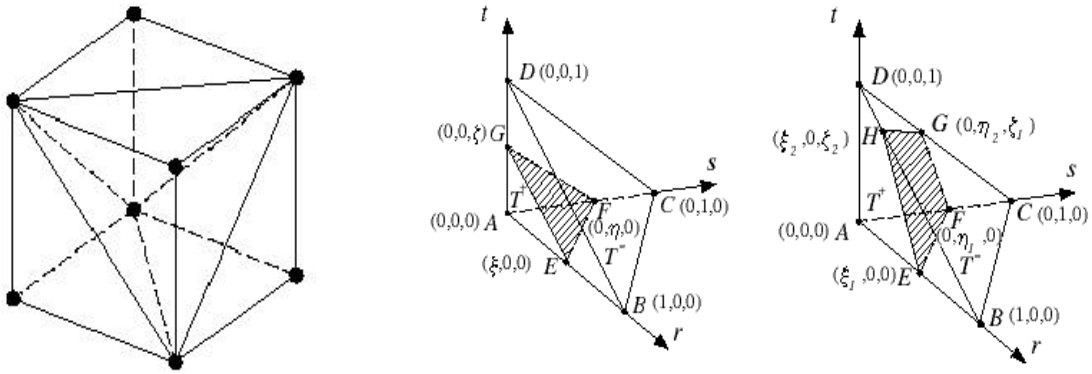

Figure 3. The Cartesian-tetrahedral cell used by IFE-PIC and intersection topologies of a tetrahedral element

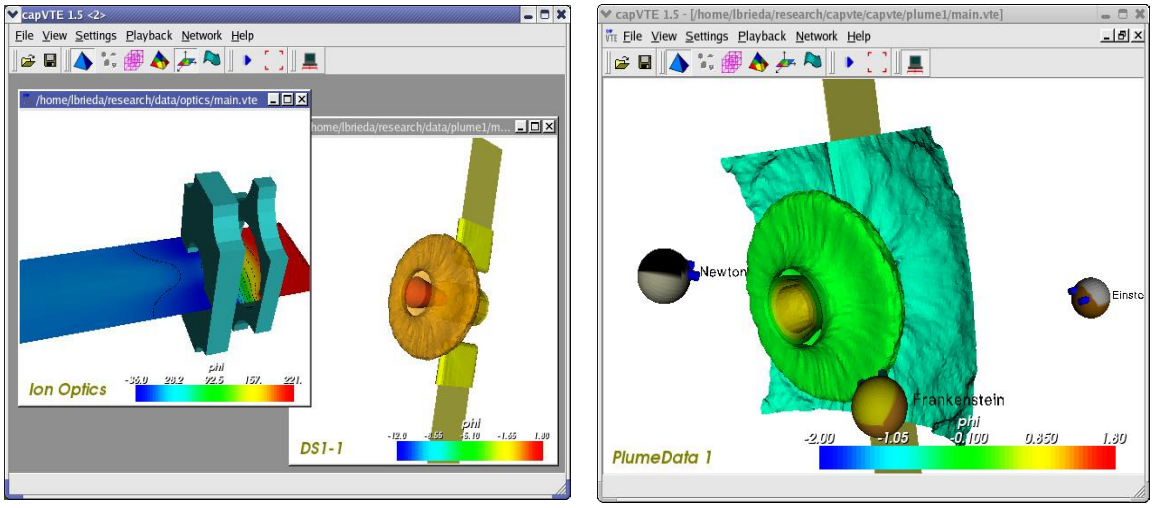

Figure 4. CapVTE desktop interface. Left: single user viewing multiple data objects. Right: multiple users connected through network viewing the same data object.

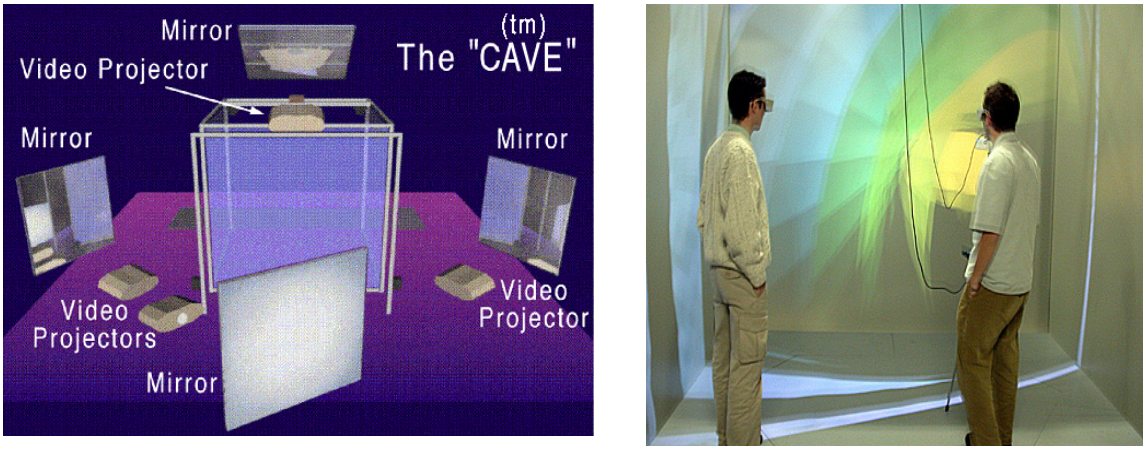

Figure 5. The Virginia Tech CAVE facility(left) and immersed visualization inside CAVE (right).

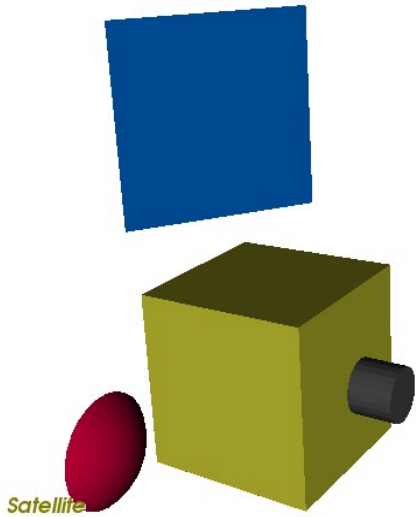

Figure 6. A model satellite configuration generated by CAD as simulation input.

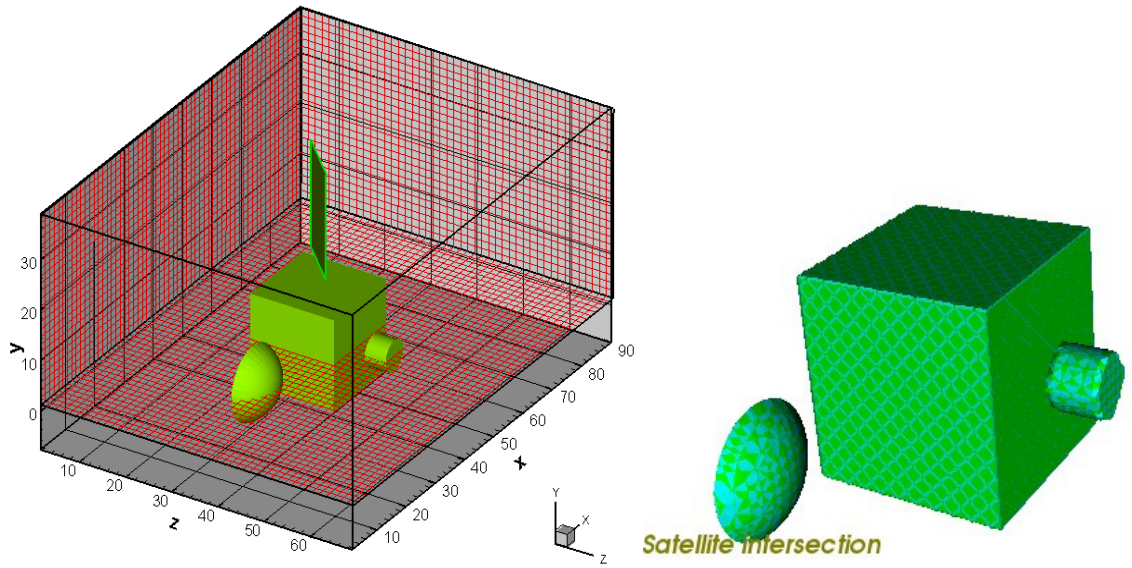

Figure 7. Left: simulation domain with Cartesian primary mesh. Right: interface cells from satellite surface intersection with the Cartesian mesh.

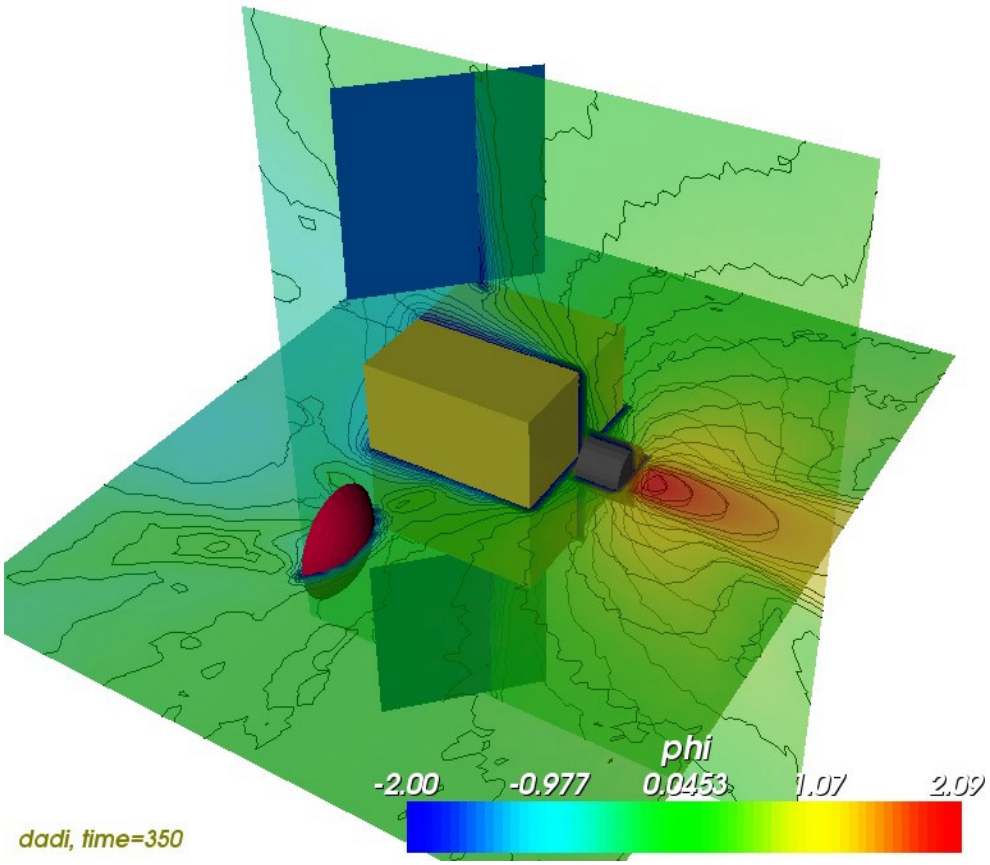

Figure 8. 3-D ion thruster plume simulation by the DADI-PIC module: potential contours

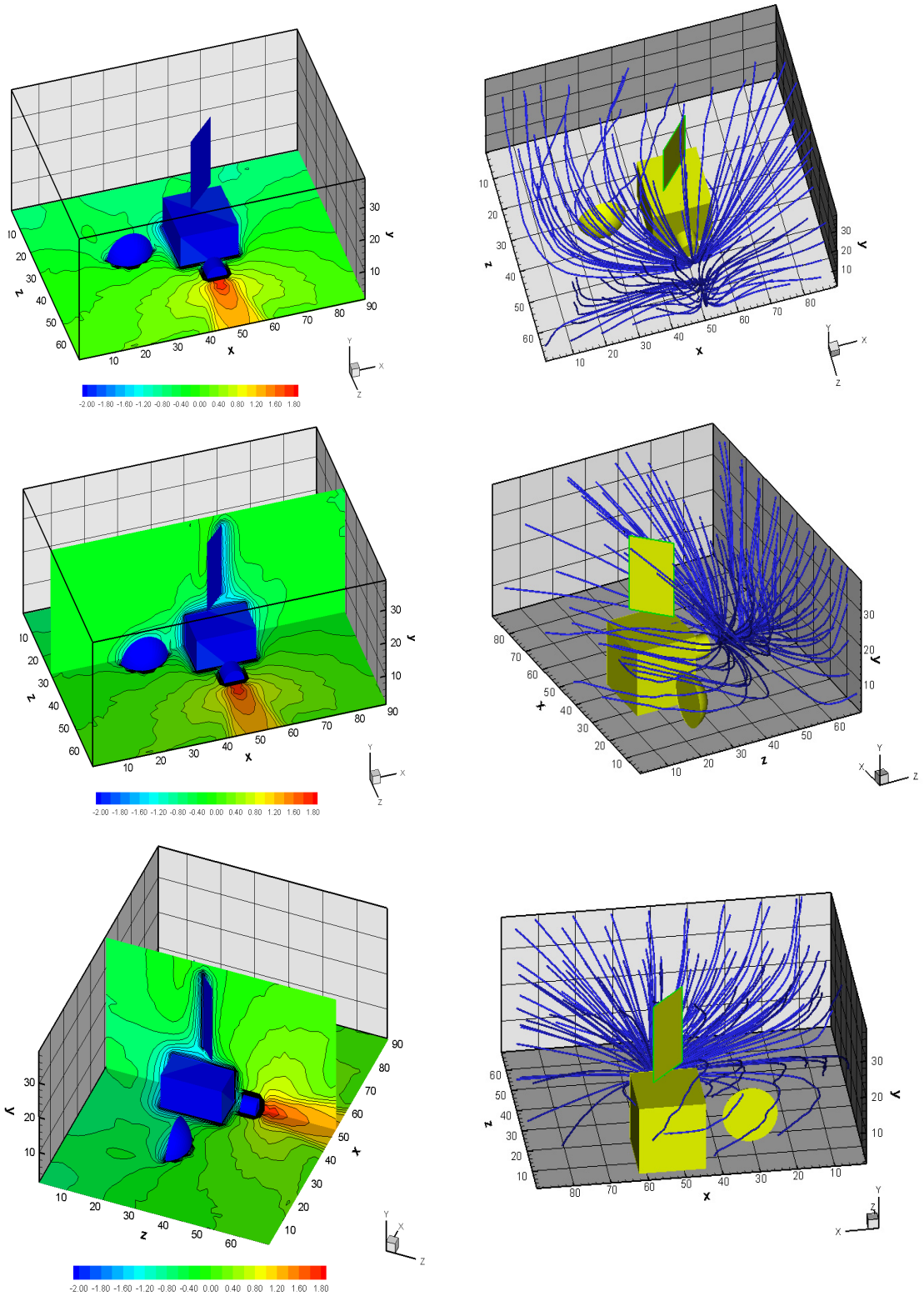

Figure 9. 3-D ion thruster plume simulation by the IFE-PIC module. Left: potential contours. Right: charge-exchange ion trajectories.

References

1. L. Brieda, J. Pierru, R. Stillwater, and J. Wang, "A Virtual Testing Environment for Electric Propulsion", AIAA 2003-5020, 2003.

2. S. Doss,S. and K. Miller, "Dynamic ADI Methods for Elliptic Equations", Siam J. Numer. Anal., 16(5), pp837-855, 1979.

3. J. Douglas and J. Gunn, "A General Formulation of Alternating Direction Methods: Part I. Parabolic and Hyperbolic Problems", Numerische Mathematik, 6, pp428-453, 1964.

4. R. Ewing, Z. Li, T. Lin, and Y. Lin, "The immersed finite volume element method for the elliptic interface problems", Mathematics and Computers in Simulation, 1999.

5. D. Hewett, W. Larson, and S. Doss, "Solution of Simultaneous Partial Differential Equations using Dynamic ADI: Solution of the Streamlined Darwin Field Equations", J. Computational Physics, 101, pp11-24, 1992.

6. T. Lin, Y. Lin, R.C. Rogers, and L.M. Ryan, "A rectangular immersed finite element method for interface problems", In Proceeding of the 2nd International Workshop on Scientific Computing and Applications, Kananaskis, Canada, May 28-June 1, 2000.

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

8. 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), pp433-440, 2001

9. J. Wang, J. Polk, J. Brophy, and I. Katz, "3-D Particle Simulations of Ion Thruster Optics Plasma Flow and Grid Erosion", J. Propulsion & Power, 2003a (in press).

10. J. Wang and T. Lin, "The Immersed Finite Element Method for Plasma Particle Simulation", AIAA 2003-0842, 2003.

11. J. Wang, R. Kafafy, and L. Brieda, "An IFE-PIC Simulation Model for Plume-Spacecraft Interactions", AIAA 2003-4874, 2003b.

12. T. Westermann, "Localization schemes in 2D boundary-fitted grids", J. Computational Physics, 101, p307, 1992