40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit
11 - 14 July 2004, Fort Lauderdale, Florida

AIAA 2004-3633

40th Joint Propulsion Conference, July 12-14, 2004, Fort Laudardale, Fl

Development of the DRACO Code for Modeling Electric Propulsion Plume Interactions

Lubos Brieda*, Raed Kafafy†, Julien Pierru‡ and Joseph Wang§
Computational Advanced Propulsion Laboratory, Virginia Tech, Blacksburg, VA, 24060

COLISEUM is a plasma simulation package for modeling electric plume interactions currently being developed by Air Force Research Lab, MIT, and Virginia Tech. One of the major components of COLISEUM is the DRACO code. DRACO, designed to perform first-principle based, high fidelity simulations of plume spacecraft interactions, include several simulation modules. As the purpose of DRACO is to simulate real engineering problems, DRACO reads in spacecraft configurations defined by commercial CAD tools. A unique feature of DRACO is that it incorporates the recently developed immersed finite element particle-in-cell (IFE-PIC) algorithm. This method allows one to use a Cartesian mesh to handle realistic spacecraft geometry 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. DRACO is cross platform and runs on Windows, Linux, and Unix. This paper presents an overview of the DRACO code and presents simulation results for simulation of CEX backflow in a vacuum tank as well as plume/spacecraft interaction in the presence of a charged plume shield.

I. Introduction

The Air Force Research Laboratory (AFRL) is sponsoring the development of a flexible, user-friendly, plasma computational package called COLISEUM. The core library of COLISEUM provides the rudimentary input/output support to several plasma simulation packages. The complexity of the simulation packages ranges from a simple ray-tracing algorith, through a prescribed plume model to several ES-PIC simulation modules. The DRACO module, being developed at Virginia Tech, is described in this paper. Additional information about COLISEUM and its simulation packages is available in Ref. 1-2.

DRACO is a multi-purpose electro-static, particle-in-cell (ES-PIC) plasma simulation package. As shown in Fig. I, DRACO allows the user to choose from several Poisson solvers. The quasi-neutral solver obtains the potential by assuming the Boltzmann distribution for the electron density and a constant electron temperature. The quasi-neutral solver is intended for quick calculations and cannot be used to resolve the plasma sheath. The DADI solver uses a standard finite-difference formulation to solve the electric field. It is designed for problems with relatively simple geometries.

The Immersed Finite Element (IFE) is DRACO's most sophisticated solver. IFE 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 body-fitted mesh, the IFE method uses a structured mesh without consideration of the object surface location. Thus the standard, Cartesian mesh based method for particle-mesh interpolations and pushing particles can be used even in simulations involving complex geometric boundaries. This allows DRACO to retain the computation speed of a standard PIC code. Many of DRACO s subroutines are based on 3-D plasma simulation codes previously developed by J. Wang to simulate ion thruster plume interactions[4] and ion optics plasma flow.[5] The legacy code produced results that were in excellent agreement with data from Deep Space 1 in-flight measurements and the long duration test of the NSTAR thruster.

*Graduate Student, Department of Aerospace and Ocean Engineering, Student Member AIAA, lbrieda@yahoo.com
†Graduate Student, Department of Aerospace and Ocean Engineering, Student Memeber AIAA, rkafafy@vt.edu
‡Graduate Student, Department of Aerospace and Ocean Engineering, Student Member AIAA, jpierru@vt.edu
§Associate Professor, Department of Aerospace and Ocean Engineering, Associate Fellow AIAA, jowang@vt.edu

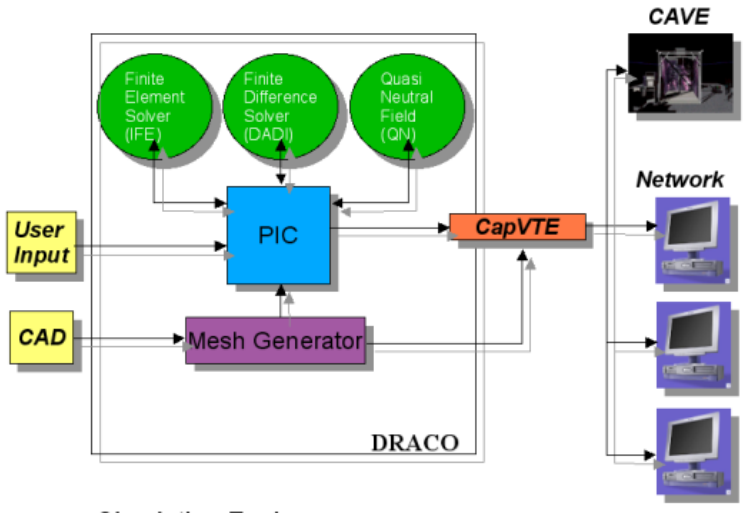


Figure 1. DRACO Code Structure

The simulation progresses according to commands specified in COLISEUM's input file. A typical command sequence first loads in a triangular mesh defining the surface of the test objects. Surface triangles are grouped into component zones. The user defines module-specific parameters on each component, such as the material name, potential, conductivity and/or visibility. Material properties are also loaded and include density, charge, specific weight (for propellant material). In a DRACO simulation, a helper mesh-generation module, VOLCAR, is initiated next. VOLCAR reads an input file specifying the span and discretization of the simulation mesh (or meshes). The generated mesh is intersected with the surface definition. Particle sources are specified by attaching emmision models to surface components. The poisson solver is then initiated and electron reference parameters are specified. Constant background fields can also be loaded and ionization can be modeled using MCC. A new simulation or a restart of a previous run then begins. All input parameters are assumed to be in SI units, unless a non-dimensional input flag is set. In the former case, the code automatically non-dimensionalizes the input according to the mass and density of a reference specie. Field diagnostics as well as PIC diagnostics (such as the number of particles, n_{mp}, or energy) are outputted at specified interval. At the end of the simulation, selected field scalars are outputted to a grid file. The output is formatted for use with Tecplot or the CapVTE[3] immersed virtual reality environment. The code can restore the full domain in a partial simulation by mirroring the solution along the symmetric faces. The solution is also extrapolated onto the surface definition. Particle positions and velocities can be sampled and outputted as well.

This paper describes in detail the mesh generation process as well as the fundamental equations used by the tetrahedral-based PIC code. Two examples are presented. In the first example, the CEX environment inside two vacuum tanks of different sizes is modeled and compared to an "in-space" reference case. The effect of background neutral density is also investigated. The second example studies the influence of a charged plume shield on the backflow of CEX around a satellite. The steady state solution is obtained for a negative shield and surface flux is compared to a grounded shield case.

II. Uniform Tetrahedral Simulation Grid

A. Simulation Grid

A crucial aspect of computer plasma simulation is the generation of a computational grid and making embedded objects "visible." One possibility involves the use of an unstructured mesh. An unstructured mesh consists of nodes scattered throughout the simulation domain and a connectivity list linking the nodes into simulation cells. The mesh is *body-fitted*, and as such will contain only the unknown nodes along with B.C. specified on the external faces. The process of creating a body-fitted unstructured mesh is not trivial, however, multiple commercial packages are available to perform this task.

The lack of structure which makes the unstructured grid so suitable for geometry modeling also imposes an overhead on the remaining aspects of the simulation. In order to perform tasks such as particle-to-cell weighing or a boundary check efficiently, the program needs to actively keep track of the particles residing in each cell. At every "move" operation, the code must flag all particles crossing the cell's walls. These particles are then shipped to the appropriate neighbor. Additional memory is also needed for storage of the node connectivity list.

The weighing is trivial if a structured grid is used, since the position of a particle is mapped to the simulation cell by directly inverting the analytical function used to define the node placement. From the performance standpoint, a structured grid is superior, especially if a large number of macro-particles is used, as is often needed in order to minimize the simulation noise. Unfortunately, a uniform Cartesian mesh cannot resolve complicated geometries, since the geometry detail is limited by the spacing of the grid nodes. Smooth objects will degenerate to a "stair-case" representation. The electric field and surface interaction will not be resolved properly in the vicinity of the objects, which is usually the region of primary intersect in the study of plume/spacecraft. In cases of simple geometries, it is possible to use an analytical representation of the objects to describe the surface boundary condition, but this approach is not practical for large-scale simulations.

Hence, a mesh which can resolve complicated geometries fairly accurately, yet can retain the benefits of a structured mesh is desired. The DRACO simulation package presented in this paper uses such a mesh. A primary uniform Cartesian grid (UCG) is used to perform the particle-to-grid weighing and the calculation of the electric field. Each cell of the UCG is subdivided into five tetrahedra, as shown in Figure 2. The embedded geometry is resolved by planar cuts of the interface tetrahedron. The tetrahedral mesh is then used to calculate the field potential[a], ϕ, as well as to perform the particle-surface interactions.

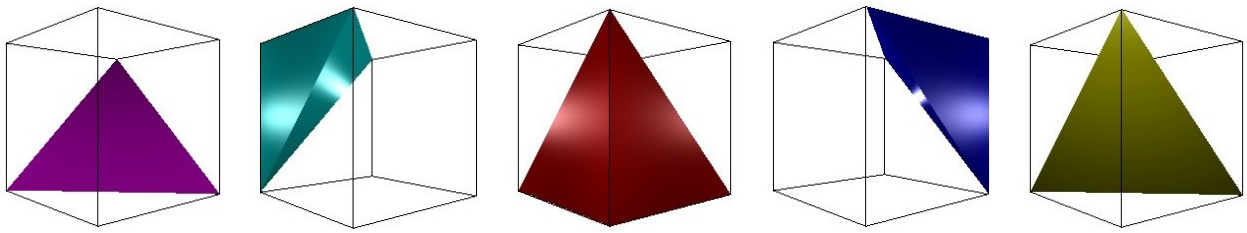


Figure 2. Tetrahedral mesh

The availability of the tetrahedral mesh allows for a much smoother representation of the test objects. The 2D-drawing in Figure 3 illustrates the difference. The node-only representation of a circle is clearly inferior. However, an almost perfect match is possible if the object is specified using planar cuts of the tetrahedral elements. This figure also illustrates that two types of elements will be present in the simulation. *Non-interface* elements are those that are located completely either inside or outside the objects. *Interface* elements, on the other hand, are being cut by the object surfaces. Hence they will contain nodes located both in and outside of the object.

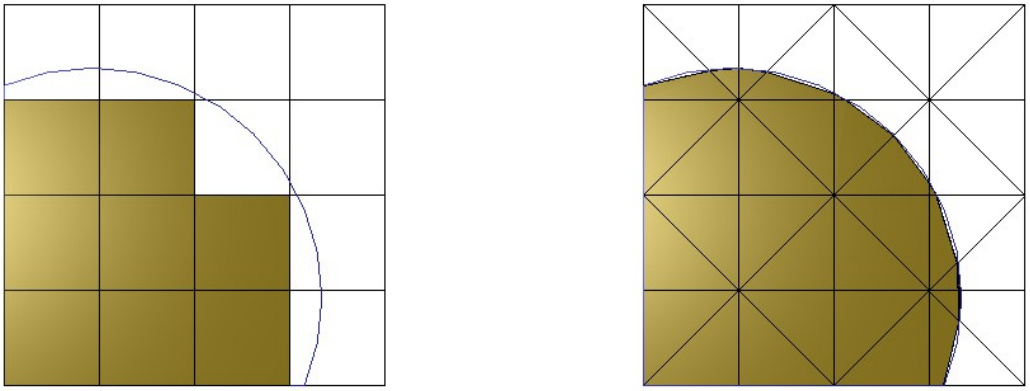
B. Mesh-Surface Intersection

Because only planar cuts through the interface elements are allowed, each interface element must have either three or four if its six edges cut. The number of intersected edges depends on the orientation of the cut, as shown in Figure 4. Since the test objects are represented by a triangular surface mesh, the problem simplifies to the common line-triangle intersection (LTI) algorithm.

First, in order for the intersection to be possible, the two endpoints of the edge must not lie on the same side of the plane to which the surface triangle belongs. The check here is very simple. The simulation volume Ω is cut by the plane Γ into two halves: Ω_+ and Ω_-. Then

$$\vec{x}\cdot\hat{n}+D\begin{cases}=0 & \forall\vec{x}\in\Gamma\\ >0 & \forall\vec{x}\in\Omega_+\\ <0 & \forall\vec{x}\in\Omega_-\end{cases} \tag{1}$$

[a]If the IFE solver is used



(a) Uniform Cartesian Mesh

(b) Overlaid Tetrahedral Mesh

Figure 3. 2D representation of a circle on a uniform Cartesian mesh and on an overlaid Tetrahedral mesh

(a) A three-edge cut

(b) A four-edge cut

Figure 4. Planar intersection of a tetrahedral element

where $\vec{x}$ is some arbitrary point, and Γ is defined by its normal vector $\hat{n}$ and some constant D. If a substitution of both endpoints of the edge into Eq. 1 results in a non-negative RHS, both points lie on the same half of the plane, and an intersection is impossible. A secondary algorithm is used for the coplanar RHS=0 case.

If the an intersection with a plane is possible, the intersection point is found from

$$\vec{p} = \vec{x_1} + t\left(\vec{x_2} - \vec{x_1}\right) \tag{2}$$

where $\vec{x_1}$ and $\vec{x_2}$ are the two endpoints of the edge and t is given by

$$t = \frac{D - \hat{n} \cdot \vec{x_1}}{\hat{n} \cdot \left(\vec{x_2} - \vec{x_1}\right)} \tag{3}$$

Lastly, a check needs to be made to verify that the intersection point is internal to the triangle. Several methods could be utilized. The three FE linear basis function, Ψ_1 to Ψ_3, can be written for the triangle. The functions are evaluated for the triangular coordinates of the point in question. The range of each Ψ_i will be in $[0:1]$ as long as the point is internal to the triangle. However, a different approach is used in the code. At the node $\vec{A}$, an angle α can be formed as $\angle\vec{C}\vec{A}\vec{B}$. Two secondary angles can also be formed, defined by $\alpha_1 = \angle\vec{C}\vec{A}\vec{p}$ and $\alpha_2 = \angle\vec{p}\vec{A}\vec{B}$. If $\alpha_1 > \alpha$ or $\alpha_2 > \alpha$, the intersection point is outside of the triangle, and the intersection does not exist. If the point passes the test at node $\vec{A}$, the process is repeated at nodes $\vec{B}$ and $\vec{C}$.

The intersection algorithm works well for smooth surfaces, but problems arise if edges or corners terminate inside the tetrahedron. In such a case, tets with fewer than three cut edges will result. Similarly, the presence of a sharply concave geometry can result in the number of cuts being higher than four. Presently, such bad intersections are corrected by discarding intersection information if the number of cuts is less than three, and by keeping only the first three intersections for the elements with a cut count higher than four. A more robust correction algorithm needs to be developed.

The problem of bad intersections can be partly eliminated by snapping all sharp edges to the grid. This technique works well for edges parallel to one of the three major grid dimensions, however, it is not applicable for arbitrarily oriented objects, such as diagonal beams. Mesh refinement, which is discussed in section D, allows the user to specify finer meshes around test objects. Since the "sharpness" of an edge is directly related to the size of the tetrahedron, the addition of finer meshes will result in a local smoothing of the geometry.

C. Classification of node/element location

The location classification (LC) algorithm marks all nodes and non-interface elements as either internal or external. The algorithm requires that the node ordering of the surface mesh is such that the normal vectors at each surface triangle point outward. Hence, the location algorithm is conceptually very simple. If $\vec{x}$ is the position of a node, or a tetrahedron's centroid, and $\vec{c}$ is the centroid of a surface triangle *visible* from $\vec{x}$, then a vector $\vec{v} = \vec{c} - \vec{x}$ can be formed. Then, the angle between $\vec{v}$ and the triangle normal $\hat{n}$ is given by

$$cos(\alpha) = \frac{\vec{v} \cdot \hat{n}}{|\vec{v} \cdot \hat{n}|} \tag{4}$$

and the node is external if $\alpha \in \left(-\frac{\pi}{2} : \frac{\pi}{2}\right)$.

The implementation difficulty arises from the need to find a visible surface triangle. Since only centroid information is needed, a triangle is deemed visible if its centroid is visible. For the centroid to be visible, there must not be any other triangle intersecting the ray $\vec{v}$. This calculation can become a computational nightmare if a large number of surface triangles is used. To illustrate this point, lets assume that a continuous surface is specified by n_{el} triangles. Checking the visibility of any particular surface triangle requires up to $n_{el} - 1$ calls to the LTI algorithm. On average, $\frac{n_{el}}{2}$ triangles may need to be checked until a visible triangle is found. Hence, to classify the location of a *single* node, $O\left(n_{el}^2\right)$ calls to the LTI algorithm may be needed. The operation count will be smaller for the average node, however, a typical simulation contains around 10^6 nodes, and five times as many elements, which immediately shows why this approach cannot be used without some optimization.

Two such optimizations are employed in the code. First, the likelihood that a triangle will be visible decreases as the distance to its centroid increases. This is because more triangles will be present in the region between the node and the centroid. The LC algorithm divides all surface triangles into "bins", according to

the distance to their centroid. A visible triangle is first searched in the closest bin, if none is found, the code moves to the second bin, and so on.

The second optimization is based on the fact that all nodes internal to an object will be separated from the external nodes by the interface elements. Before classifying the location according to Eq. 4, the LC algorithm checks whether location can be copied from the left, front or bottom $(i-1, j-1, k-1)$ neighbor. This optimization is highly dependant on a sucesful execution of the intersection algorithm. If too many bad intersections are formed, the interface will contain holes, and the location type will "leak-out".

Element location is classified according to the node location, when possible. If all four vertices of a tetrahedron have the same location type, the tetrahedron must clearly also have the same location type. If mixed vertices exist, and the element is non-interface, as may be the case for a tetrahedron along a surface snapped to the Cartesian grid, the location is determined by applying the described LC algorithm to tetrahedron's centroid.

D. Mesh Refinement

Often, the simulation geometry will contain details smaller than the cell size of the simulation domain. Similarly, the geometry may contain many sharp corners which are not being intersected properly using the larger cells. Decreasing the cell size in the entire domain is usually not the option, since it would require increasing the number of cells such that the span of the domain remains constant. Memory and computational requirements for the larger domain could be prohibitive.

A better alternative is to use mesh refinement. VOLCAR allows for multiple overlapping grids of varying depths to be present in the simulation domain. The meshes are specified by the user in the same format as the main mesh is specified, with the exception that discretization is chosen automatically. VOLCAR currently supports only a 1:2 refinement, in which every other node, beginning with the first node, is overlapping a node on the parent grid. In simulations containing an object in an initially empty domain (such as a satellite in space), the refined mesh can encompass the entire simulation geometry. The parent coarse grid then does not resolve any object boundaries, and can be specified as Cartesian-only. The node location on the refined mesh is be propagated upward to the parent mesh. The nodes not overlapped by any child mesh are set as external. This approach reduces memory requirements for large-scale simulations.

III. ES-PIC Code

A. Fundamental Equations

The DRACO simulation package is based on the electro-static, particle-in-cell (ES-PIC)[7] method. The plasma is represented by a finite set of *macro-particles*, each of which is accelerated according to

$$\vec{F} = m\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = q\left(\vec{E} + \vec{v} \times \frac{\vec{B}}{c}\right) \tag{5}$$

where $\vec{B}$ is a constant background magnetic field. The electric field, $\vec{E}$ is determined from

$$-\nabla\vec{E} = \nabla^2\phi = -\frac{\rho}{\varepsilon_0} \tag{6}$$

where $\varepsilon_0 = 8.854 \times 10^{-12} F/m$.

Computer arithmetics is higly susceptible to round-off errors, especially when operations are applied to operands of different orders of magnitude. DRACO's input parameters are specified in the SI units, but are internally normalized prior the start of the simulation. Length is normalized by the Debye length, λ_d, and velocity is normalized by the plasma frequency, ω_p. Both parameters are computed using the mass and densities of a specified reference specie. The remaining quantitites are normalized accordingly. The dimensional quantites are restored automatically at the end.

B. Particle-surface interaction

Stability of the solution prohibits the particles from traveling through more than one cell during any single time step. This condition can be exploited to efficiently perform the particle-surface interaction check. A

particle interacts with a surface if, during the time step motion, it crosses inside the solid object. But since the minimum thickness that can be resolved by the simulation grid equals the cell size, a particle in a properly defined simulation cannot travel completely through an object during a single step. The only required check is whether the final position of the particle is within a solid object. Since the simulation grid is divided into tetrahedral cells, a particle position check could be performed by placing the particle into a tetrahedron, and, according to its type, performing an appropriate function. However, from the performance standpoint, this approach is not acceptable. During the simulation, the majority of particles will reside in the volume far removed from the test objects, and thus will be located in an external Cartesian cell. Placing the particle into a cell on a uniform Cartesian grid is a trivial, computationally cheap operation. The first test consists of checking whether the cell is either completely external or internal to the object - whether all its nodes have the same location type.

The majority of particles will be classified using the first test. However, if the particle resides in a Cartesian cell being cut by the test objects, a second test will be needed. Here it is necessary to place the particle into the proper tetrahedron. Since each Cartesian cell is subdivided into five tetrahedra, the particle needs to reside inside one of them. Hence, instead of calculating which tetrahedra the point could be in, the code only loops through the five possible choices. The check is performed by subdividing the tet into four sub-tets, each formed by joining three of the four original vertices with the position of the macro-particle. The particle is located inside the tetrahedron if the total volume of the four sub-tets equals the volume of the tetrahedron. In other words, a point is located in a tetrahedron if

$$\left| V_{tet}\left(1,2,3,4\right) - \sum_{i=1}^{4} V_{tet}\left(p, v_{i1}, v_{i2}, v_{i3}\right) \right| \leq \epsilon \tag{7}$$

where a volume of a tetrahedron, V_{tet}, given by vertices $\left(x_i, y_i, z_i\right)\big|_1^4$ is

$$V_{tet}\left(1,2,3,4\right) = \frac{1}{6} \left| \begin{array}{ccc} \left(x_2 - x_1\right) & \left(y_2 - y_1\right) & \left(z_2 - z_1\right) \\ \left(x_3 - x_1\right) & \left(y_3 - y_1\right) & \left(z_3 - z_1\right) \\ \left(x_4 - x_1\right) & \left(y_4 - y_1\right) & \left(z_4 - z_1\right) \end{array} \right| \tag{8}$$

and $\left(x_i, y_i, z_i\right)_p$ denotes the position of the particle. The tolerance term, ϵ, is used to account for the inexactness of computer arithmetic. The three vertices of the original tetrahedron used to construct a sub-tet are given by the circular indexes v_i[b].

If the particle resides inside an interface element, a third test is necessary to determine which side of the cut the particle is in. DRACO checks whether the particle and a single vertex of the interface tetrahedron reside on the same side of the planar cut, as given by Eq. (1). If both points lie on the same side of the cut, the particles location is set to that of the vertex, otherwise it is set to the opposite.

If the particle impacted the surface, several actions are feasible according to the particle's type and its impact velocity. A neutral particle can either reflect or stick to the surface. A charged particle will be neutralized, and, similarly, can either impinge to the surface or bounce back. If the impact energy is high enough, the impact can result in several surface atoms being sputtered off. The simulation presented in this paper tracked only the CEX ions, and impacting ions were removed from the simulation. This is analogous to neutralizing the ion and reflecting the neutral atom, while imposing that the addition of the new neutrals will have only a negligible effect on the neutral density.

C. Immersed Finite Element Potential Solver

The immersed finite element (IFE) solver is based upon a method developed by Lin et.al.[6] Previous works show that such a Cartesian mesh based field solver can resolve the potential near to grid boundaries as accurately as a body-fit unstructured mesh based field solver.

The IFE method applies the "interface" concept which forms the base for the original immersed/imbedded boundary techniques. However, the IFE method is significantly different from other immersed/immbeded boundary techniques because:

1. The IFE method is based upon finite element formulation.

[b] $v_1 = (1,2,3)$, $v_2 = (2,3,1)$,$\cdots$

2. The trial functions used in the IFE method are constructed *merely* using the physics-based jump conditions across the interface.

The essential feature of the immersed finite element method is that its mesh can be formed without consideration of the interface location. Similar to those methods based on the immersed/imbedded boundary techniques, the IFE method needs to handle interface elements with special attention; otherwise the method will lose accuracy in the vicinity of the interface elements where important physics usually happens. The IFE method uses finite element basis functions that can satisfy the interface jump conditions required by physics in the interface elements. Since tetrahedral elements possess favorable topological features compared to a brick element while keeping the total number of mesh nodes the same, the IFE uses a Cartesian-based tetrahedral mesh. In a non-interface element, the standard linear local nodal basis functions can be constructed to span the local finite element space. The interface Γ divides a typical interface element T into T^+ and T^- sub-elements. This partition of T is used to construct four piece-wisely linear local nodal basis functions.

D. Effect of the refined mesh

1. Particle motion

Since the particle-surface interaction check assumes that the distance traveled by any particle during any time step is no greater than the cell length, the time step size needs to be adjusted according to the finest mesh. Hence, the elimination of the memory-storage problem imposes a penalty of longer computations. A possible work-around is to calculate the field potential at the time-scale corresponding to a movement on the coarse grid. Since moving the particles is a relatively inexpensive computational step, this method allows the code to resolve a particle interaction with complex geometries without negatively affecting the performance.

2. Field solution

The division of the computational domain into meshes of varying cell size requires addition of a Poisson solver capable of operating on such a mesh. One aproach is to treat the sub-grids as components of a single larger grid, with coefficients of the Laplacian matrix for each unknown set up according to that unknown's neighbors. For instance, the Laplacian operator associated with the Poisson equation can be represented (in 1-D) using the 2^{nd} order finite difference method as:

$$\nabla^2\phi\,|_i \approx \frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{\Delta^2 h} \tag{9}$$

where ϕ_i is the value of the potential at the position corresponding to the node n_i. This representation is valid as long as the cell spacing Δh is the same for both the left and right neighbors, as is the case for all nodes *internal* to a particular uniform grid. If node n_i represents the boundary between two grids of varying cell size, such that $x_i - x_{i-1} = \Delta h$ and $x_{i+1} - x_i = \Delta h/2$, then the finite difference representation analogous to Eq. (9) is

$$\nabla^2\phi\,|_i \approx \frac{4f_{i-1} - 12f_i + 8f_{i+1}}{3\Delta^2 h} \tag{10}$$

with a similar relationship existing for the reversed refinement order.

This approach will result in a single Laplacian matrix. The associated system is solved using an arbitrary linear solver. However, in some instances, using a single solver is not desirable. Such is the case if the test object is completely immersed in the fine mesh, while the coarse mesh is used to desribe the surrounding space environment. The object will thus be completely resolved on the fine mesh, and the coarse mesh does not need to contain any intersection information. The coarse mesh can then be solved using a finite difference solver. The solution is retained only on the nodes not being overlapped by the child mesh. The potential on the fine mesh is obtained from IFE.

Two issues become immediately apparent. First, due to the elliptic nature of the Poisson eq., the solution of the fine mesh is dependant on the solution of the coarse mesh at the boundary nodes. Second, a continuity of solution needs to be assured. The standard C^1 continuity requires $\vec{E}_{\Gamma_-} = \vec{E}_{\Gamma_+}$, however the current implementation assures only the C^0 continuity. This simplification was chosen for its performance advantages, as well as to allow for the use of non-C^1 solvers[c].

[c]Such as the direct inversion of the Boltzmann equation

The field solution begins by obtaining a solution on the coarse mesh. The solution is interpolated onto the fine mesh, and external nodes are set as Dirichlet. Because the Cartesian mesh cannot resolve the geometry detail accurately, the coarse solution will not be correct in the vicinity of the objects. Hence, enough free space needs to be included in the fine mesh such that these effects disappear. Solution on the fine mesh is then gathered onto the coarse mesh. The nodes of the coarse mesh which overlap *non-boundary* child nodes are set as Dirichlet. New solution is obtained on the coarse mesh. The process can be repeated until the solution along the mesh boundaries stabilizes.

IV. Examples

A. Influence of vacuum chamber walls and background pressure on CEX backflow

1. *Reference "in-space" condition*

The first example looks at the difference in CEX creation and backflow between an "in-space" condition and one that can be found in a vacuum tank. More specifically, the influence of the backround neutral density and the proximity of the chamber walls was studied. The thruster assembly was represented by a 20cm long cylinder with a 32 cm diameter. The thruster exit had a 30cm diameter. The operating conditions were chosen to represent the NSTAR ion engine, operating at mission throttle level (ML) 83, as given in Wang.[4] The parameters are summarized in Table 1 and the thruster geometry is shown in Figure 1. The propellant is xenon.

Table 1. Thruster input parameters

Parameter	Value
n_{b0}, m^{-3}	3.22×10^{15}
n_{n0}, m^{-3}	2.30×10^{17}
V_{b0}, m/s	3.87×10^{4}
σ_{cex}, m^2	3.37×10^{-19}
n_{ele0}, m^{-3}	3.22×10^{15}
n_{i0}, m^{-3}	2.76×10^{7}

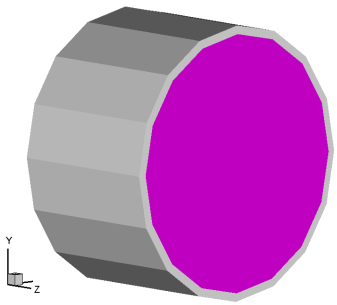


Figure 5. Simulation model of the 30cm ion thruster in a 32cm casing

A rigorous DSMC modeling can be utilized to predict the creation of the CEX ions. However, such a model requires that the code keeps track of the fast moving beam ions as well as the neutrals. Due to their high velocity, the beam ions are not significantly influenced by the electric field and move at their ballistic trajectories. In this simulation, the beam ions are represented by an analytical charge density profile. An analytical model is also used to represent the neutral density, as given by Roy,[8] with a term added to account for a constant background density. This approach frees up computational resources, which can then be used to track a higher number of CEX ions.

The charge exchange production rate at the thruster exit is then determined from

$$\dot{n}_{cex0} = n_{b0} n_{n0} v_b \sigma_{cex} \tag{11}$$

where v_b is the velocity of the beam ions. The collision cross-section area, σ_{cex}, is given by

$$\sigma_{cex} = (k_1 ln v_b + k_2)^2 \times 10^{-20} m^2 \tag{12}$$

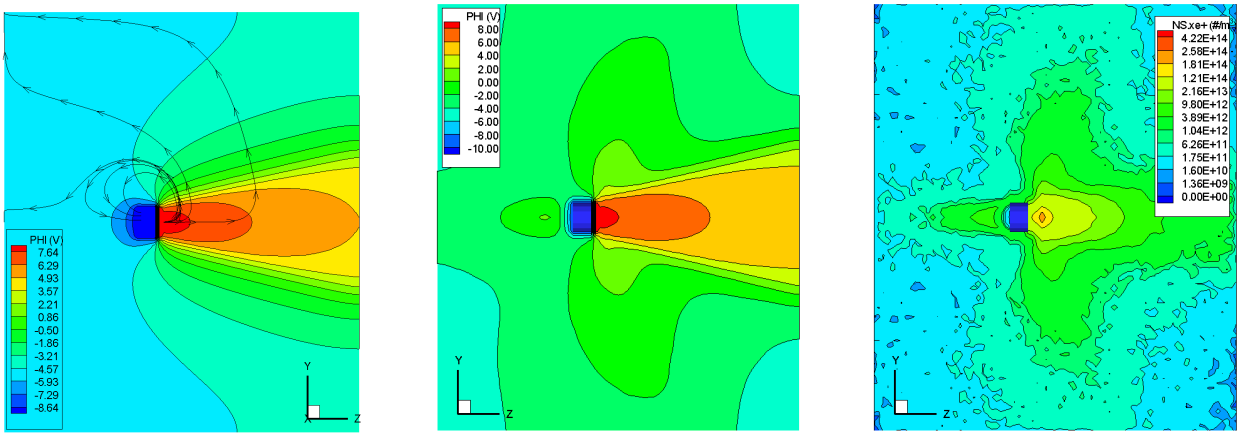
For Xenon, the coefficients k_1 and k_2 are -0.8821 and 15.1262, respectively.

The first simulation run was used to establish the reference, in-space behavior. The thruster exit was located at x=0m, y=0m, z=0.9m. The chamber walls were not present. Charged particles striking the thruster body were removed from the domain without affecting the charge on the thruster. Spacecraft charging was instead approximated by fixing the potential on the thruster at -10V. The potential at the thruster exit was fixed at ϕ_{ref}=9V. Electron reference temperature, Te_{ref}, of 1.25eV was used. Wang[4] indicates that the local Debye length in the backflow region at ML83 is of $O(10cm)$. In this similation, the CEX reference density, n_0, was set to $2.76\times10^{10}\mathrm{m}^{-3}$, which corresponds to λ_D of 5cm. The steady state number density obtained from the simulation agrees with the input.

The cell size was set to the reference λ_D in all three directions. Due to the problem symmetry, only one quarter of domain was simulated. From the computational stand-point, it is not practical for the simulation domain to be large enough such that the outer boundary is essentially CEX-free. The expansion into an ambient plasma was instead modeled by setting the Neumann condition, $\nabla\phi = 0$, on the outer boundaries. Symmetry on the x_{min} and y_{min} face required that particles leaving these faces be reflected elastically back to the domain. Domain length in the x and y directions were 2.25m; length in the z direction was 4m.

The simulation ran until the steady state, given as $\frac{|\Delta n_{mp}|}{n_{mp}} < 10^{-4}$, where Δn_{mp} is the change in the number of macro particles between two consecutive time steps. The steady state was achieved after 450 time steps. The total simulation time[d] was 80 minutes. The IFE Poisson solver was used in all the examples presented in this paper and performed extremely very well. Tolerance was set to 10^{-6}. In most cases, the solver obtained the initial solution in less than 10 iterations of the non-linear solver.

The ϕ due to the primary beam in the absence of CEX ions is shown in Fig. 6a). The streamlines of the $\vec{E}$ field are also shown. Due to their initial slow velocity, the CEX ions initially follow the $\vec{E}$ field and will tend to backflow to the region upstream of the thruster. This is indeed the case as Fig. 6 b) and Fig. 6 c) indicate. The space/velocity phase plot based on an uniform sampling of every 100^th particle is shown in Figure. 7. Phase plot for x vs. u is not shown, since, due to the problem symmetry, it matches y vs. v.



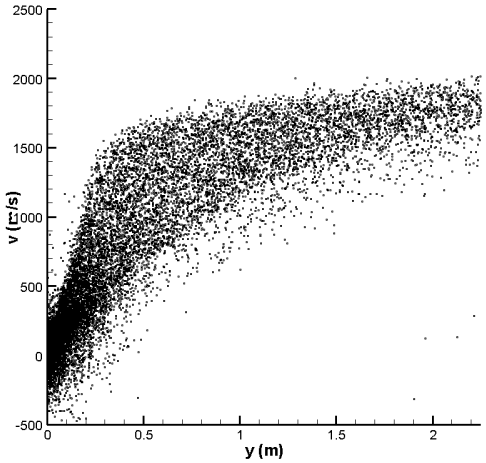
(a) ϕ, V, in the absence of CEX ions

(b) ϕ, V

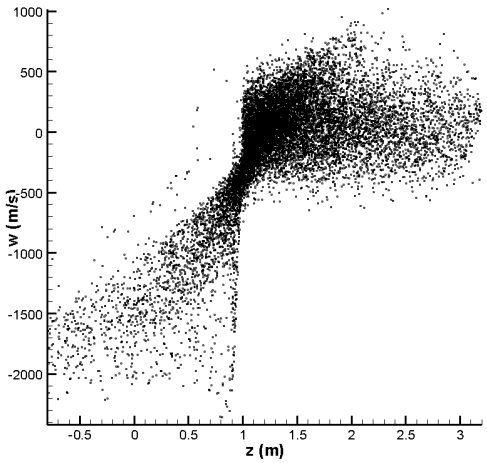
(c) CEX number density, particles/m^3

Figure 6. "In-space" simulation results, yz cut at x=0

[d]Using an off-the-shelf Pentium IV, 2.5GHz laptop with 500Mb of RAM



(a) y vs. v, m and m/s



(b) z vs. w, m and m/s

Figure 7. "In-space" simulation results, space vs. velocity phase plot

2. Large chamber

The dimensions of the first chamber were chosen large enough, such that the chamber walls did not directly obstruct the CEX expansion. Any variation in expansion behavior can be then attributed to the difference in external boundary conditions. Diameter was set to 3m and the length was 6m. The upstream face of the tank was located at z=0. The thruster was aligned with the tank's major axis (x and y=0), with the beam exit located at z=2.45m. The whole simulation setup can be seen in Figure 2. Both the thruster and the tank were grounded with ϕ=0. Beam reference potential, ϕ_{ref}, was set to 19V. The beam input parameters were as described in the previous section.

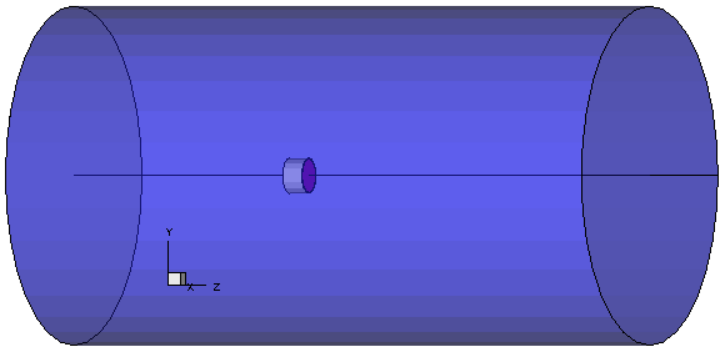
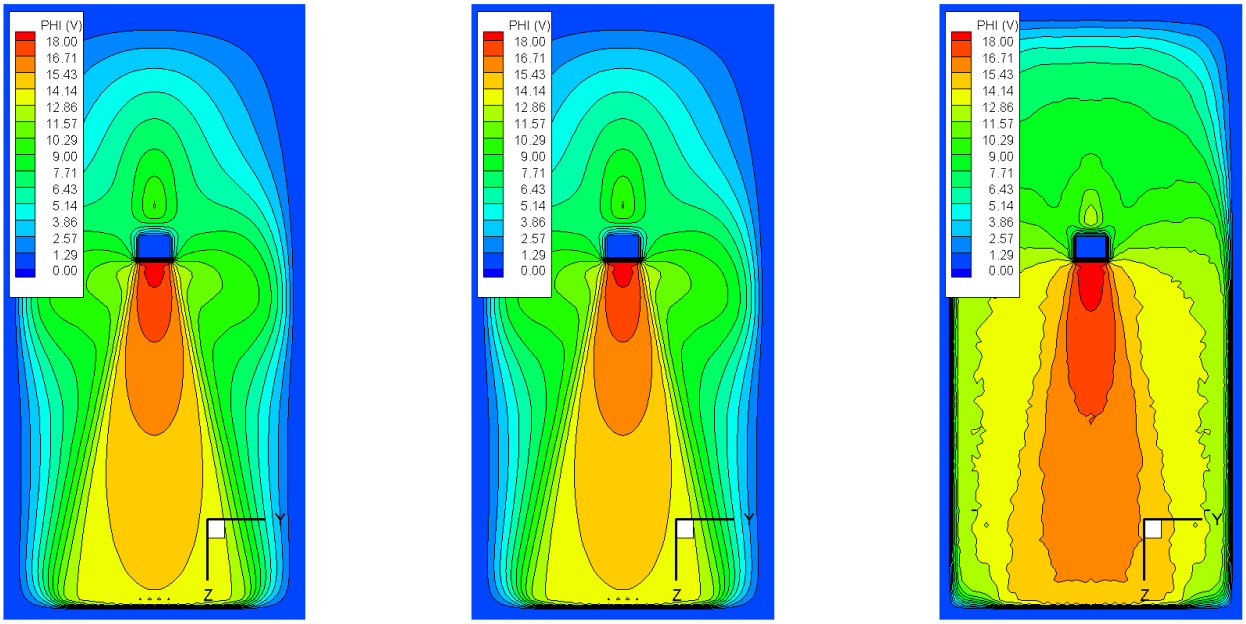


Figure 8. Simulation model of the ion thruster in a large vacuum tank

Total of three runs were performed for this configuration. The background ion, n_{i0}, and neutral, n_{nback}, densities were varied as given in Table 2. In terms of pressure, the background densities correspond to approximately 3×10^{-10} torr for cases 1 and 2 and 3×10^{-6} Torr for case 3, assuming ideal gas. Reaching the steady state solution took the longest for case 3 with approximately 600 time steps and over 5 hours of wall-time. This was partly due to the time step size chosen too small. Figure 9 shows the contour lines of the field potential on a plane of symmetry for the three cases. A similar plot was generated for the number density of the CEX ions, and can be seen in figure 10. Both sets of plots show that the density of the background ions, n_{i0}, does not influence the backflow, at least not for the low values that can be expected in a vacuum tank. The CEX density in the backflow region for cases 1 and 2 correlates with the reference "in-space" results.

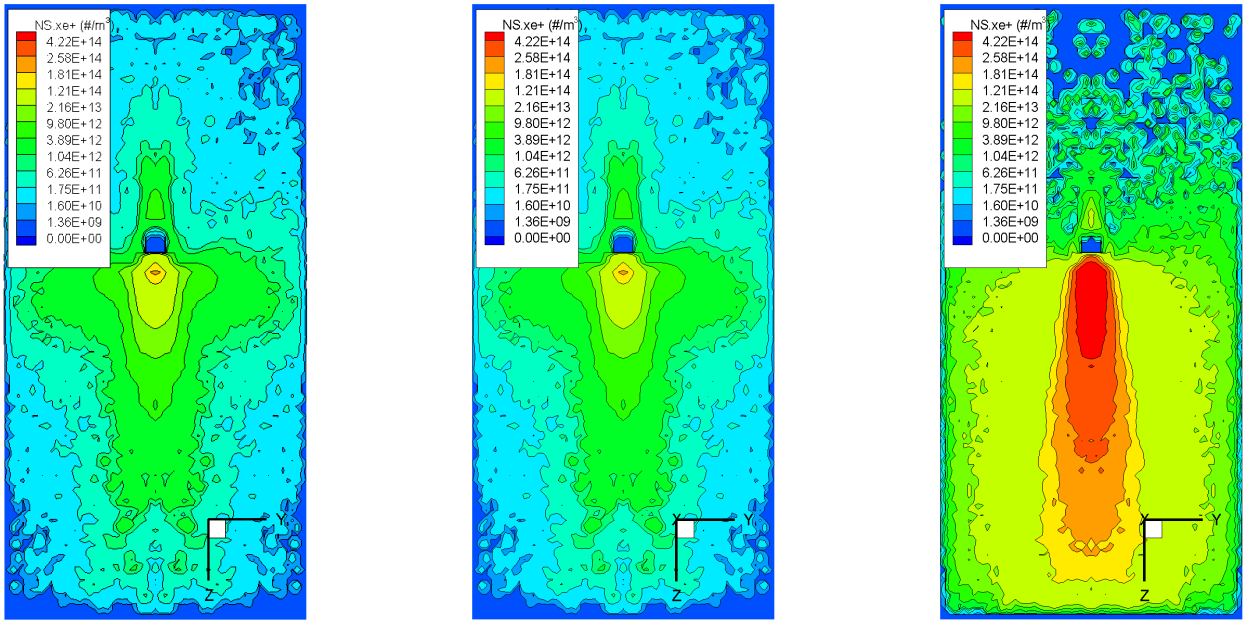
Table 2. Vacuum tank background parameters

Case No.	n_{i0}, m^{-3}	n_{nback}, m^{-3}
1	0	1×10^{13}
2	2.76×10^{7}	1×10^{13}
3	2.76×10^{7}	1×10^{17}



(a) case 1 (b) case 2 (c) case 3

Figure 9. Large tank simulation results, ϕ, V, at steady state, yz cut at x=0



(a) case 1 (b) case 2 (c) case 3

Figure 10. Large tank simulation results, CEX number density at steady state, particles/m^{-3}, yz cut at x=0

The background neutral density used in cases 1 and 2 did not significantly influence the CEX production rates. The same is not true for case 3. The CEX production rate just outside the thruster exit increased by almost 45%. The CEX production zone, which is dominated by a small region outside the thruster exit in the in-space case, now extended through the entire downstream region.

Figures 11 and 12 show the space/velocity phase plots for this configuration. The yv plot shows a steady increase in radial velocity within a 0.5m radius of the beam centerline. The same acceleration can be seen in the reference case. Past this distance, the acceleration decreases dramatically, and the radial velocity stabilizes at around 1500m/s. In the vacuum tank case, the acceleration decrease is also apparent at the 0.5m radial distance. However the acceleration resumes at 1m from the centerline. The secondary expansion is due to the potential gradient created by the grounded tank walls. The effect is more noticable in case 3, since the higher ion density will result in a larger potential drop. A wall-induced acceleration in the axial direction is also noticeable in the zw plot, however, the region of influence is very small. This simulation indicates that a tank of 1.5m radius can reasonably approximate the in-space conditions only within radial distance of about 1m from the centerline.

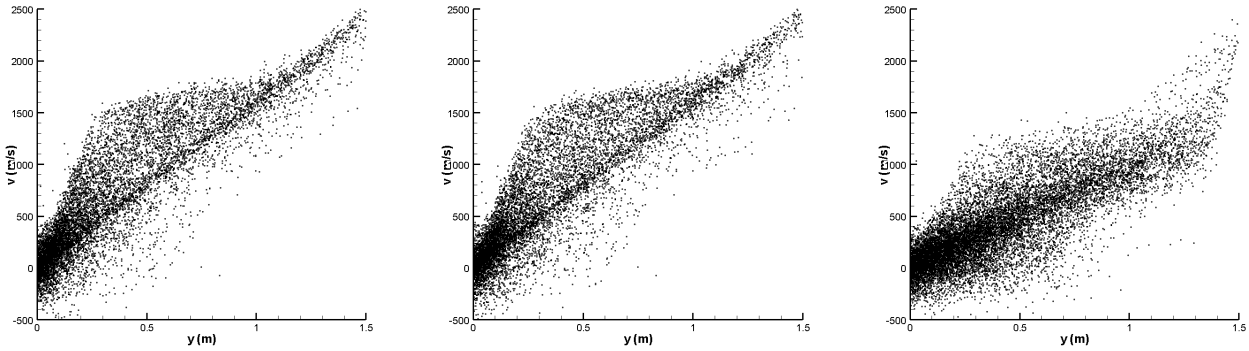
3. Small chamber

The diameter and length of the vacuum chamber were next halved to 1.5 and 3m, respectively. The thruster exit was located at z=0.9m. Three cases were ran, with the same parameters given in the previous section. The influence of the tank walls, as shown in Figure 15, is much stronger. In fact, the region of constant radial CEX velocity is almost completely eliminated. Environment inside a chamber this small is obviously not a good representation of the in-space environment.

B. Effect of plume shield on the CEX backflow

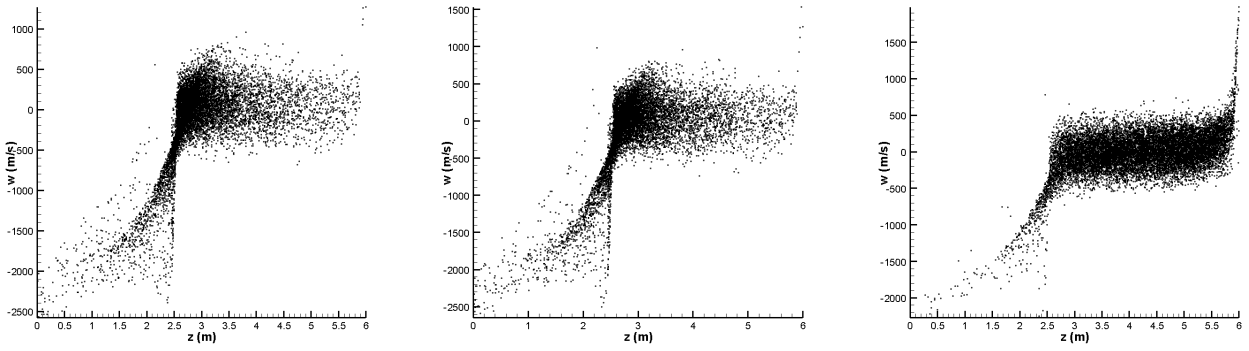
1. Simulation model

Charge-exchange ions created due to imperfect propellant ionization will back-flow onto spacecraft components. Although xenon is not a contaminating specie, the presence of charged particles can affect the readout from sensitive instrumentation, even if the instruments are not in a direct line of sight of the thruster plume. High-energy collisions with the surface will result in sputtering of the surface material. Particularly affected



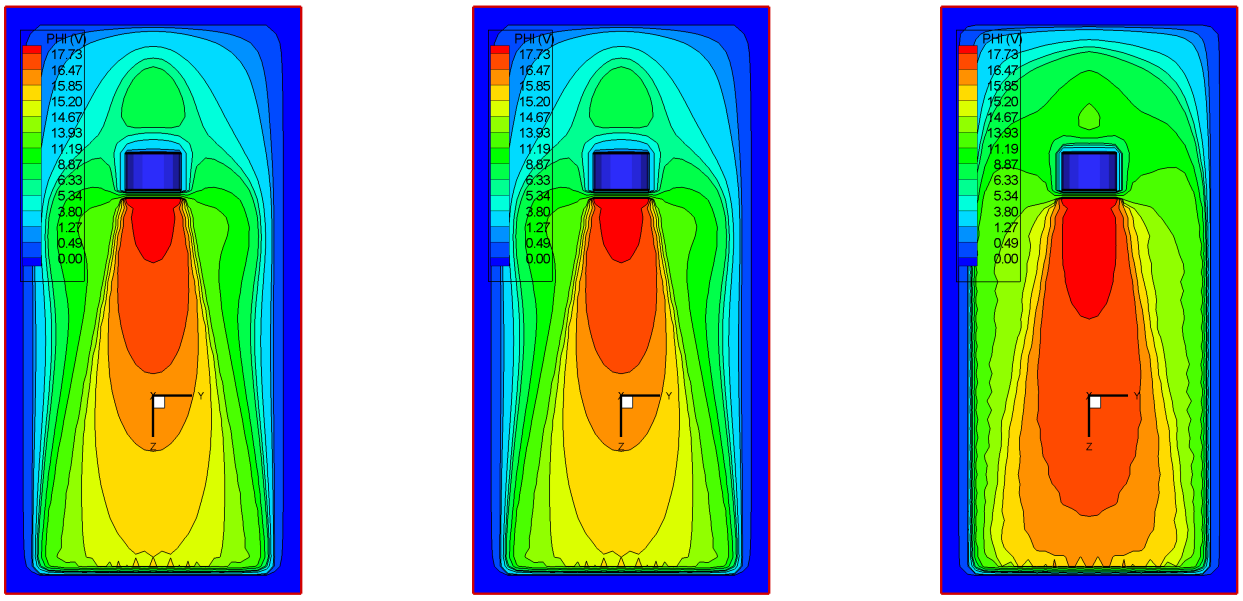
(a) case 1 (b) case 2 (c) case 3

Figure 11. Large tank simulation results, CEX x, m, vs. u, m/s, phase plot at steady state



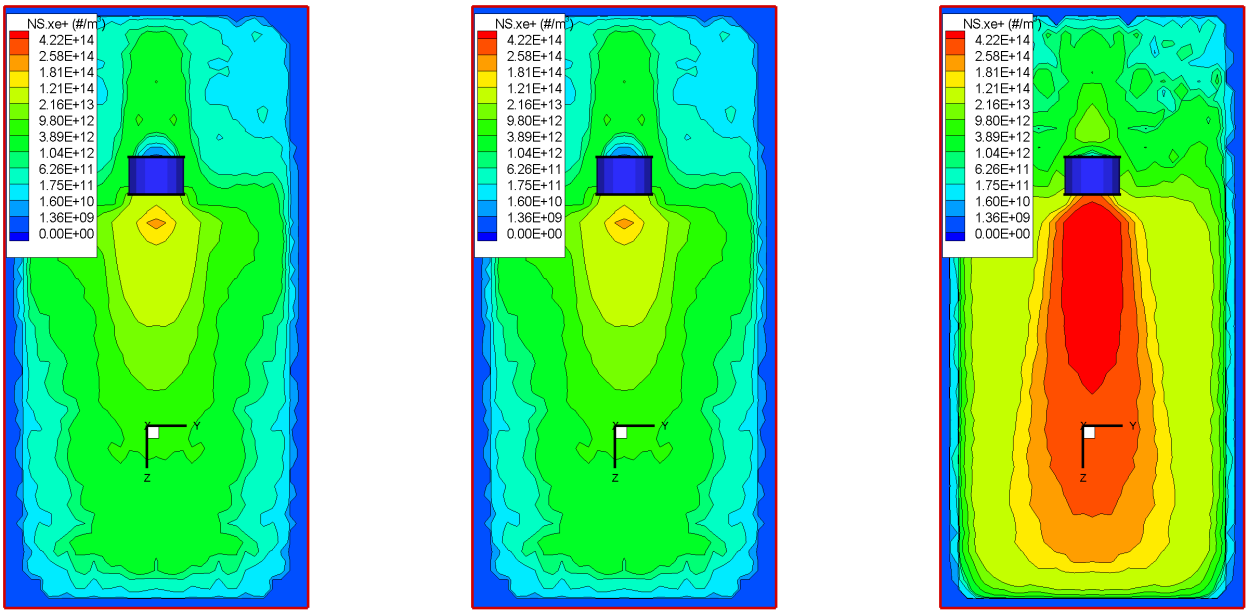
(a) case 1 (b) case 2 (c) case 3

Figure 12. Large tank simulation results, CEX z, m, vs. w, m/s, phase plot at steady state



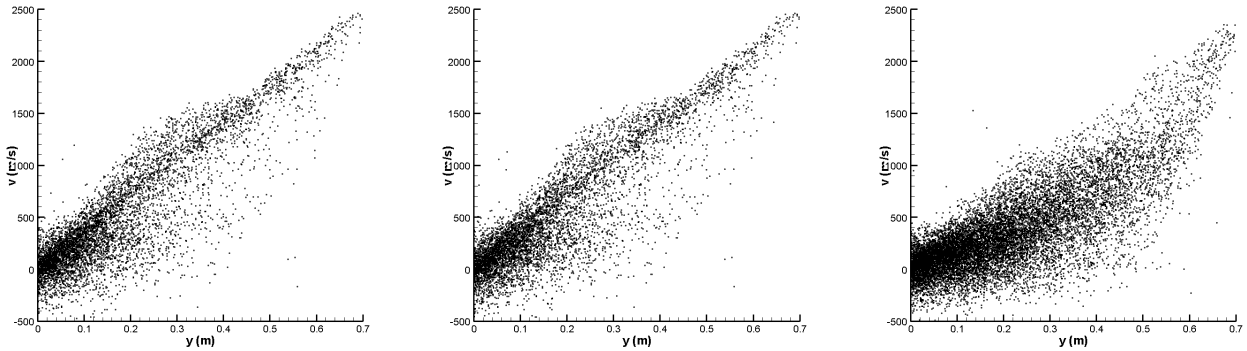
(a) case 1 (b) case 2 (c) case 3

Figure 13. Small tank simulation results, ϕ, V, at steady state, yz cut at x=0



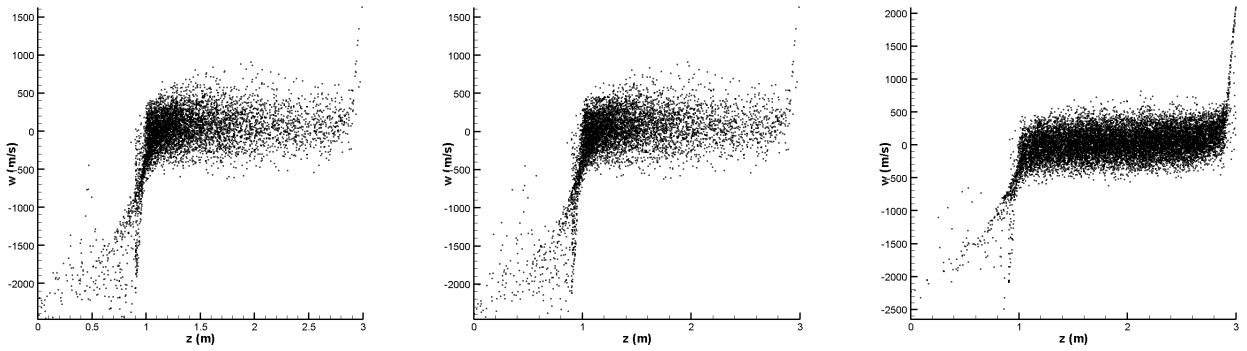
(a) case 1 (b) case 2 (c) case 3

Figure 14. Small tank simulation results, CEX number density at steady state, particles/m^{-3}, yz cut at x=0



(a) case 1 (b) case 2 (c) case 3

Figure 15. Small tank simulation results, CEX x, m, vs. u, m/s, phase plot at steady state



(a) case 1 (b) case 2 (c) case 3

Figure 16. Small tank simulation results, CEX z, m, vs. w, m/s, phase plot at steady state

is the accelerator grid at the thruster exit, usually manufactured using molybdenum. The sputtered off molybdenum will accumulate in the high beam density region and will undergo the same CEX collisions experienced by the neutral xenon propellant, corrected for the different ionization energy. Molybdenum is highly contaminating and can coat sensitive equipment, such as the solar panels or cameras. The erosion of the accelerator grid due to sputtering of molybdenum is also the leading factor limiting the useful life of an ion thruster. Only the backflow of xenon was studied in this example. The backflow behavior for different materials will be influenced by the material's molecular mass. However, the mass of molybdenum, the primary contaminating specie, is close to that of xenon and the backflow behavior will be similar.

The satellite geometry is shown in Figure 1. A 30cm NSTAR thruster, operating at the same conditions outlined in the previous example, was centered on the back face of the satellite bus. The bus was a 1m box with numerous geometry detail added to represent on-board instrumentation. Simulation parameters matched the "in-space" case of the previous example, namely a -10V potential on the bus, with ϕ_{ref}=9V. Electron refence temperature remained at Te_{ref}=1.25eV and CEX reference density of 2.76×10^{10} particles/m^3 was used. The mesh-refinement concept could not be applied to this case, due to a very slow convergence of the DADI solver. Hence, only a quarter of the domain was simulated. While the satellite is not completely symmetric, the lack of symmetry is not large enough to significantly affect the CEX backflow. The quarter domain contained 50x50x90 cells, with cell spacing, dh, equal to 5 cm. Time step was set such that no CEX traveled more than approximately $0.5dh$ per step, which was about 5×10^{-6}s for this particular case.

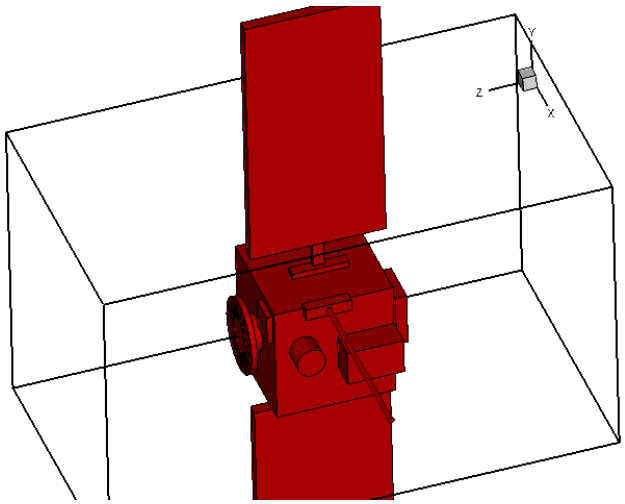
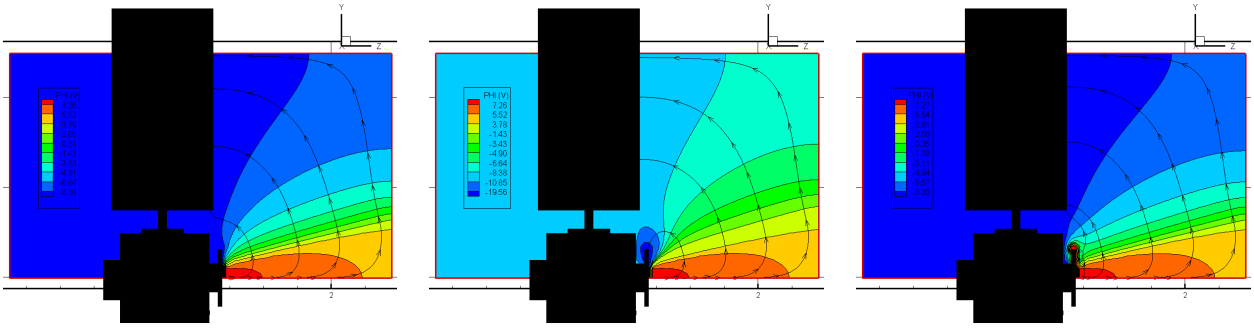


Figure 17. Satellite geometry with the domain bounding box

A thin plate of a 120cm radius was flushed with the thruster exit and served as the plume shield. A larger shield would reduce the backflow by physically blocking the path of the backflowing CEX ions. However, the added mass cost may be too prohibitive. This example examines the influence of a *charged* plume shield. Charging the shield will increase its effective area, since trajectories of ions will be influenced by the potential gradient. A negatively charged shield will attract ions, which will neutralize upon impact. Depending on the incidence angle, the new neutrals can drift back into to the primary beam, and undergo new CEX collisions. This secondary re-ionization is however not studied in this example. The colliding ions are removed from the simulation, without affecting the surrounding neutral density. An alternative approach is to charge the shield positively with the intention of repelling them away from the spacecraft.

2. Preliminary Study

The negatively charged shield shield was simulated by setting $\Delta\phi = \phi_{shield} - \phi_{bus}$ on the shield to -19V. Similarly, +19V $\Delta\phi$ was set for the positive configuration. Countours of the potential in the abscence of CEX for the the grounded, negative and positive configurations are plotted in Fig. 18. Streamlines of the $\vec{E}$ field are also shown. Due to their slow velocities, the CEX ions will tend to follow the $\vec{E}$ field. Immediately a large tendency of ions to backflow onto the solar panel can be observed.



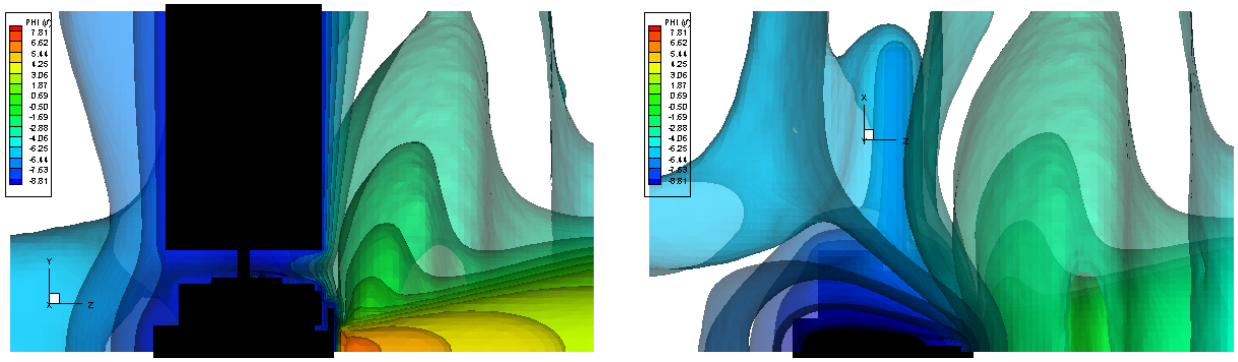
(a) shield $\Delta\phi$=0 (b) shield $\Delta\phi$=-19V (c) shield $\Delta\phi$=+19V

Figure 18. Potential due to the ion beam, with $\vec{E}$-field streamlines, $\Delta\phi = \phi_{shield} - \phi_{bus}$

The negatively charged shield has a strong effect on the ions that would normally backflow onto the bus, but only a small influence on the solar panel contamination can be seen. Specifically, this preliminary run predicts a reduced contamination around the bottom solar panel corner. The positively charged shield, however, does not seem to have a strong influence on the backflow. The added potential is not large enough to overcome the strong drop in the beam radial direction. Higher plate potentials were not tested as they caused problems for the convergence behavior of the Poisson solver. However, the potential would need to be set high enough to deflect the ions completely away from the influence of the spacecraft. Otherwise, the ions normally backflowing onto the bus could be pushed further into the expansion wing from where they could backflow onto the sensitivie solar array.

3. Steady-State solution

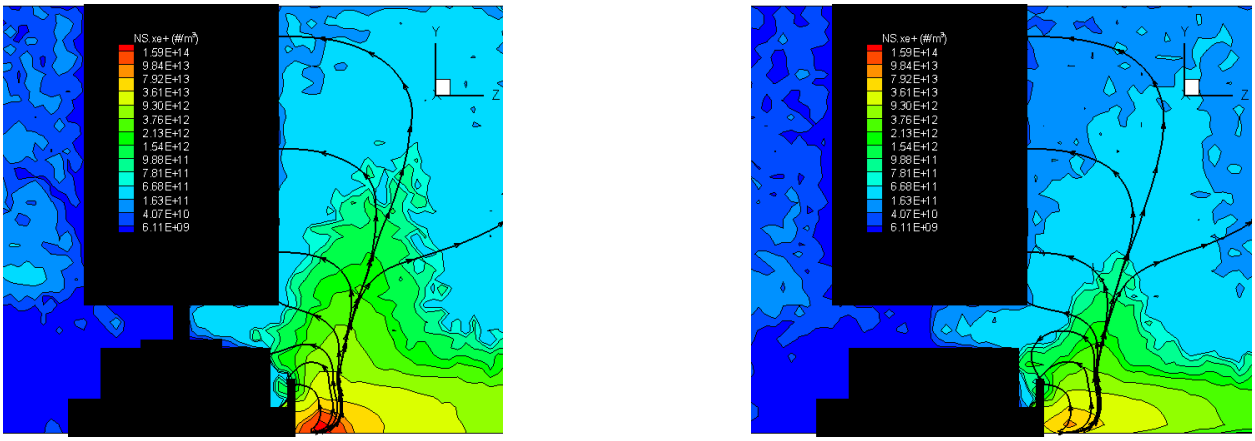
The steady state solution was achieved in about 3.5 hours and 550 steps. At this time, approximately 1.6 million CEX ions were tracked. Figure 19 shows the isosurfaces of ϕ a the steady state for the grounded plate configuration. A sheath surrounding the spacecraft can be seen. Contour slices of ion CEX density are plotted in Figure 20 and 21 for the grounded and negative configurations. The negative shield has an obvious influence on the expansion of the CEX wing, however, it doesn't significantly affect the ion density in the near vicinity of the spacecraft. A secondary effect is the reduction in the maximum CEX density in the beam core which could have a significant impact on the rate of grid erosion.



(a) yz view from y=0 (b) xz view from top of the solar array

Figure 19. Isosurfaces of potential at steady state for the grounded shield

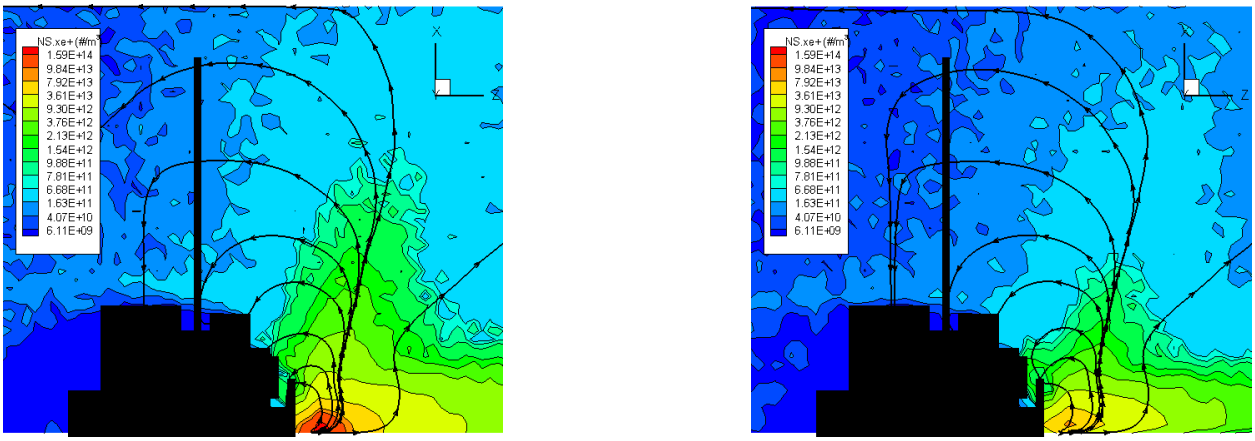
Phase plots for the grounded configuration are shown in Figure 22. A very good correlation with the reference case presented in the previous example can be seen. Further, only minor difference can be seen between the xu and yv plots, indicating that the majority of CEX is contained in the expansion wing.



(a) shield $\Delta\phi$=0

(b) shield $\Delta\phi$=-19V

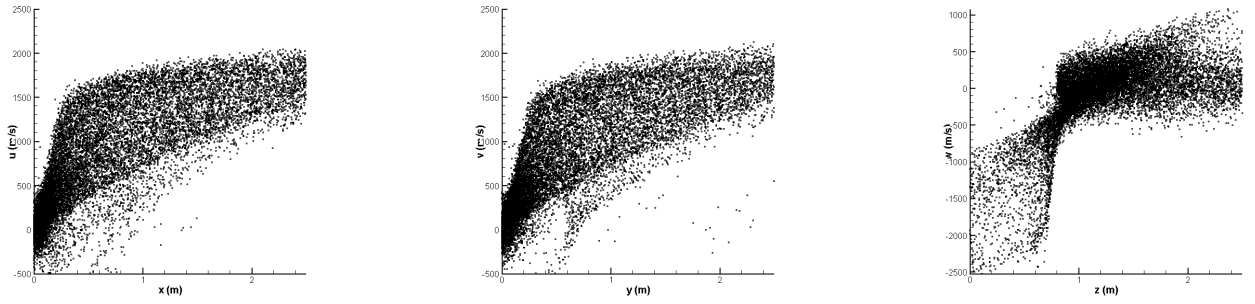
Figure 20. CEX number density at steady state, yz cut at x=0, with $\vec{E}$ field streamlines



(a) shield $\Delta\phi$=0

(b) shield $\Delta\phi$=-19V

Figure 21. CEX number density at steady state, xz cut at y=0, with $\vec{E}$ field streamlines



(a) xu

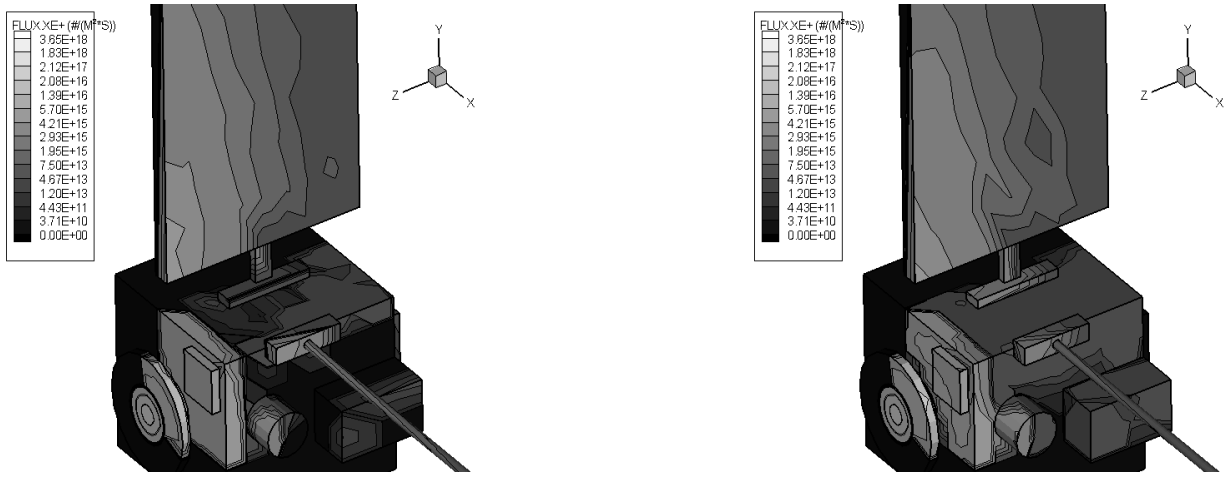
(b) yv

(c) zw

Figure 22. Phase plots of CEX at steady state

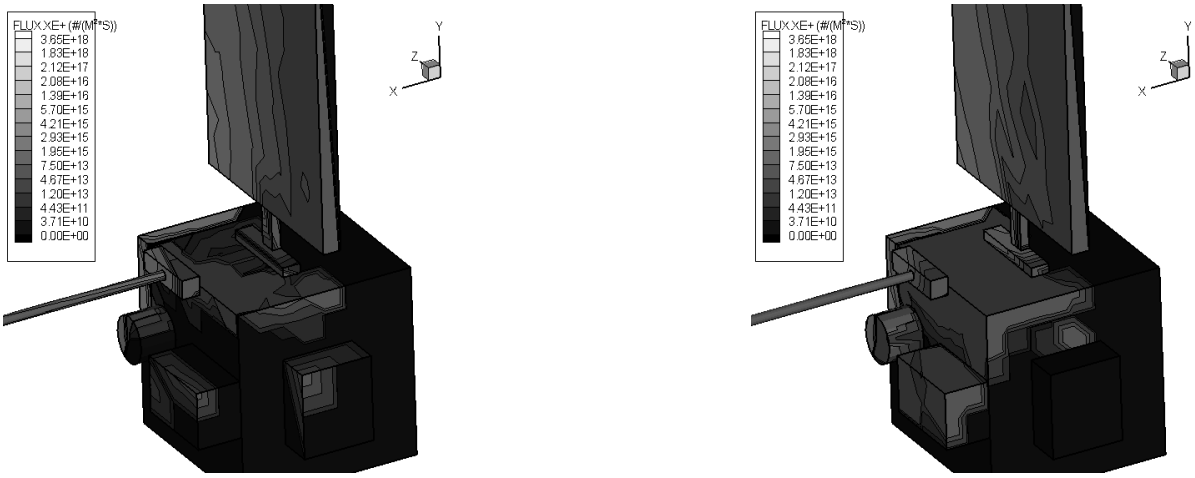
4. Surface Flux

Finally, the simulation was restarted for additional 300 time-steps, at a reduced time-step size. The field potential was solved only once every 25 iterations. Since the simulation was at the steady state, the potential did not vary considerably between iterations. During this time, surface flux was calculated by weighing density of removed particles onto surrounding surface nodes. Contours of the surface flux from two view angles can be seen in Figures 23 and 24. As predicted, the contamination on the lower corner of the solar panel is slightly reduced if the negatively charged shield is used. However, the flux on the bus increased. Similarly, the flux on the surfaces on the opposite side of the bus from the thruster also increased. However, the particle density at steady state in the back region was not high enough to obtain an accurate statistical sample. The particle density used in this example was limited by the lack of a large amount of RAM on the simulation computer.



(a) $\Delta\phi$=0V (b) $\Delta\phi$=-19V

Figure 23. Average surface flux, view angle 1



(a) $\Delta\phi$=0V (b) $\Delta\phi$=-19V

Figure 24. Average surface flux, view angle 2

V. Conclusion

A flexible plume/spacecraft interaction modeling package called DRACO is being developed at Virginia Tech as a part of AFRL's COLISEUM project. DRACO is an ES-PIC code, based on a uniform Cartesian tetrahedral mesh. Such a mesh can accurately resolve complicated surface geometries, while retaining the computation efficiency of a structured grid. This paper describes the algorithms used in the generation of the mesh. Elements of the PIC module are also described. Several examples are presented. The first example looks at the influence of vacuum chamber walls on the backflow of CEX around a 30cm ion thruster. The addition of fixed boundary condition significantly influenced the radial expansion of the CEX plume. The representation of the environment inside a 3m diameter vacuum chamber was valid only to within the internal radius of 1m from the centerline. Plasma within 0.5m from the tank walls was attracted to it due to the potential drop existing between the grounded walls and the positive CEX ion region. A smaller vacuum chamber with diameter of 1.5m could not accurately resolve the radial expansion of the CEX. The effect of increased background pressure was also studied. No influence was noted for background density for $O(10^{13})$ neutrals/m^3. Increasing the density to $O(10^{17})$ resulted in CEX ions created almost uniformly through the entire beam upstream region.

In the second example, a plume shield was placed around the thruster exit on a detailed satellite. Potential due to the beam without the presence of CEX ions was obtained first. A positive shield did not seem to be effective and was not studied further. Steady-state solution was obtained for a grounded shield (physical barrier) and a negative shield (attracting ions). The negative shield reduced the size of the CEX expansion wings, however, surface flux of xenon did not decrease. In fact, the flux on the bus increased slightly. However, the negative shield decreased the maximum density of CEX in the beam core. This reduction could result in lower rates of molybdenum sputtering from the thruster accelerator grid, but this behavior was not accounted for in the simulation.

Acknowledgments

The authors would like to thank Doug VanGilder, Matt Gibbons and Mike Fife for support with DRACO integration into COLISEUM.

References

[1]Gibbons, M. R., Kirtley, D. E., VanGilder, D. B., and Fife, J. M., "Flexible Three-Dimensional Modeling of Electric Thrusters in Vaccum Chambers," AIAA-2003-4872, 2003

[2]Santi, M., Cheng, S., Celik, M., Martinez-Sanchez, M., and Peraire, J., "Further Development and Preliminary Results of the Aquila Hall Thruster Plume Model," AIAA-2003-4873, 2003

[3]Wang, J., Brieda, L., Kafafy, R., Pierru, J., "A Virtual Testing Environment for Electric Propulsion-Spacecraft Interactions," AIAA-2004-0652, 2004

[4]Wang, J., "Three-Dimensional Particle Simulations of Ion Propulsion Plasma Environment for Deep Space 1," *Journal of Spacecraft and Rockets*, Vol. 38, No.3, 2001, pp. 433-440

[5]J. Wang, J. Polk, J. Brophy, and I. Katz,"3-D Particle Simulations of Ion Thruster Optics Plasma Flow and Grid Erosion," *Journal Propulsion and Power*, 2003, Vol. 38, No. 6, pp. 1192-1199

[6]Kafafy, R., Lin, T., Lin, Y., and Wang., J., "3-D Immersed Finite Element Method for Electric Field Simulation in Composite Materials", submitted to Int. Journal for Numerical Methods in Engineering, 2004.

[7]Birdsall, C. K., and Langdon., A. B., *Plasma Physics via Computer Simulations*, 1st ed., Institute for Physics Publishing, Bristol, 2000

[8]Roy, S.R., "Numerical Simulation of Ion Thruster Plume Backflow for Spacecraft Contamination Assesment," Ph.D. Dissertation, Aeronautics and Astronautics Dept., Mass. Institute of Technology, Cambridge, MA, 1985