REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 06-03-2012	**2. REPORT TYPE** PhD Thesis Chapter	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE **Chapter 7 of PhD Thesis – "Multiscale Modeling of Hall Thrusters"**		**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Lubos Brieda		**5d. PROJECT NUMBER** **5f. WORK UNIT NUMBER** 33SP0853
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Research Laboratory (AFMC) AFRL/RZSS 1 Ara Road Edwards AFB CA 93524-7013		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory (AFMC) AFRL/RZS 5 Pollux Drive Edwards AFB CA 93524-7048		**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S NUMBER(S)** AFRL-RZ-ED-TP-2012-062

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited (PA #12134).

13. SUPPLEMENTARY NOTES

For presentation at George Washington University, Washington, D.C.

14. ABSTRACT

The final component of a multiscale modeling approach to Hall thrusters addresses the expansion of the plasma plume..

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Justin W. Koo, Ph.D.
Unclassified	**Unclassified**	**Unclassified**	SAR	23	**19b. TELEPHONE NUMBER** *(include area code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

CHAPTER VII

Plume Modeling

7.1 Introduction

The final component of our multiscale approach addresses the expansion of the plasma plume. Outside the thruster, the magnetic field strength decays rapidly and the plume dynamics is dominated by the electrostatic Lorentz force, $\vec{F} = q\vec{E}$. However, an important factor in the plume is collisional interaction between particles. Of particular importance is the so-called *charge exchange* reaction (CEX). The resonant charge exchange occurs when an ion and neutral of the same parent species come into a close contact to exchange an electron without a significant momentum or energy transfer. Since the velocity of neutrals is much smaller ($\sim$ 1 km/s thermal velocity) than the velocity of the ions ($\sim$ 17 km/s), this interaction

$$Xe^{\circ}_{slow} + Xe^{+}_{fast} \rightarrow Xe^{+}_{slow} + Xe^{\circ}_{fast} \tag{7.1}$$

results in the creation of slow ions and fast neutrals.

The thruster plume always contains a radial $\Delta\phi$ component due to the radial decay in density. Furthermore, due to the geometric expansion of the plume, the $\vec{E} = -\nabla\phi$ field points slightly back towards the thruster. The strength of this field can be estimated by considering typical potential drops and beam radii, $E_r =$ 20V/20cm = 100V/m. Considering the primary beam ions, we can see that this beam merely acts to increase the beam divergence. Assuming the ions were initially moving in axial-only direction and that a uniform field acts over a distance of 0.5 m, the induced radial velocity component is $v_r = \sqrt{2 * 50eV/m_{Xe+}} = 8572$ m/s, giving $\theta = \tan^{-1}(v_r/v_z) = 27°$. A much stronger effect is seen on the CEX ions. Since these ions can be approximated as initially stationary, they will be accelerated by potential gradients in the radial direction. Assuming the initial energy is $\sim 0eV$, the CEX ions exit the plume with $W = e\Delta\phi$ kinetic energy. This effect has been demonstrated numerous times by experimental measurements with sweeps of energy analyzer probes showing a clear population of low energy ions at high angles from the beam centerline[34, 35].

Outside the thruster plume, the charge exchange ions expand into a donut-like structure commonly known as charge exchange wings. These wings have been studied extensively by previous researchers, including Roy [36], Wang [37], and Boyd [35]. These expansion of these ions is based on local electrostatic fields and can result in an ion backflow into regions with no line of sight to the thruster. This introduces several issues. First, the collection of plasma current by spacecraft components could lead to a dielectric charging and potentially disastrous arcing. Fortunately, this effect is generally not pronounced, in fact, since ions will be attracted to negatively charged surfaces, electric propulsion plumes can in fact reduce the spacecraft potential. However, directly related to this effect is an impact on instruments. Many space weather satellites utilize charged particle sensors to study space plasmas and phenomena such such solar storms and magnetic reconnection. Operation of EP devices used for station keeping can modify the local plasma environment around the sensor and lead to a spurious current collection. Furthermore, CEX plume expansion can also result in a direct contamination. The Xenon propellant used in typical EP thrusters is not reacting, however, impacts of CEX ions could potentially lead to low-energy sputtering of condensible material. Similarly, condensible materials sputtered from the thruster can be ionized in the plume and subsequently expanded radially in a manner similar to the standard CEX expansion. These molecular contaminants can subsequently polymerize to the surface and form monolayers that reduce transmission of optical sensors and modify emissive properties of thermal control surfaces[38].

7.2 Draco Plume Simulation Code

Unfortunately, it is difficult if not outright impossible to predict the dynamics of the backflowing ion plume without performing numerical simulations. Such simulations allow the designer to perform trade studies, and find the optimal location for sensitive instruments. Quite a large number of numerical codes have been developed precisely to tackle modeling of EP thruster plumes[37, 39–42].In this work we utilize an electrostatic particle-in-cell code *Draco*. Draco was developed in 2005 at Virginia Tech and was subsequently integrated into the AFRL Coliseum framework[43]. The author of this dissertation was the primary developer of Draco. An early version of the code is described in the author's master's thesis [44] and also in [45]. The code has gone through a large number of changes since then, with several improvements of interest implemented in the course of this dissertation. These new features primarily tackle numerical issues necessary to make the code more efficient and robust. These features are described below.

7.2.1 Mesh Splitting

Draco was initially developed as a serial code operating on a strictly Cartesian mesh $x = x_0 + i\Delta x$. However, the need to model larger simulation domains dictated the necessity to incorporate support for parallel processing [46] and non-Cartesian rectilinear stretched meshes[47]. Mesh stretching allows the user to capture a larger physical domain without increasing the number of mesh nodes (unknowns) that need to be solved for and also stored in memory. The latter item also resulted in an addition of *zones* to describe the topology in each axial direction. This was necessary since it would be difficult to describe general simulation set ups using a single stretched mesh. Often we need to use a fine mesh in a high density region around the thruster, and an expanding mesh in the low density far plume.

Unfortunately, while the implementation of the multi-zone approach decreased the number of computational nodes needed to capture a certain region of interest, it also resulted in an additional overhead on the particle push. The primary strength of Draco had always been its speed. Particle codes such as Draco utilize a large number of computational particles to describe the velocity distribution function of the simulated fluid. At every time step, two sets of interpolations are necessary per particles. First the particle positions are scattered to the grid to compute the electric field. The electric field is then interpolated back onto the particles to update their velocity. The strength of the Cartesian mesh approach is that is that the interpolation becomes trivial - the cell in which the particle is located is easily inverted from position, $i = (x - x_0)/\Delta x$. The cell logical coordinates are also easily obtained from $l_i = i - \text{int}(i)$. In the initial implementation of the multi-zone approach, the zones existed as virtual boundaries overlaid over a single computational mesh. The particles moved in the parent mesh, and in order to compute the logical coordinates, the zone containing the particle had to be first determined. This was done by looping through the zones and comparing particle positions to the zone boundaries. This was a highly inefficient operation as indicated by profiling studies, such as the one shown in Figure 7.1. This plot shows the effect of increasing the number of particles from 200k to 500k (2.5 times) and also of increasing the mesh size 8 fold (2x2x2). Blue bars are

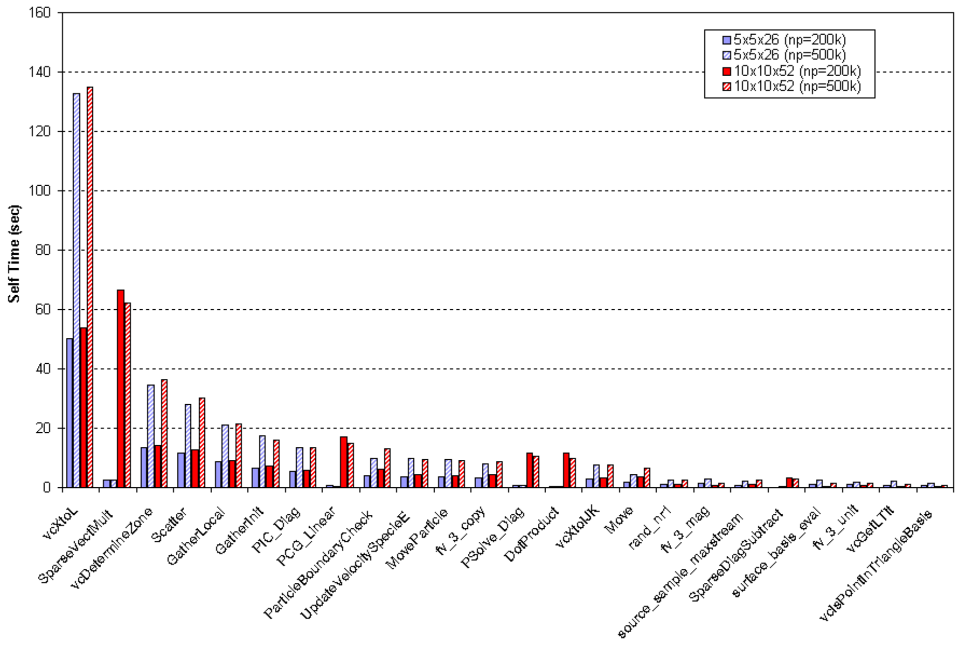
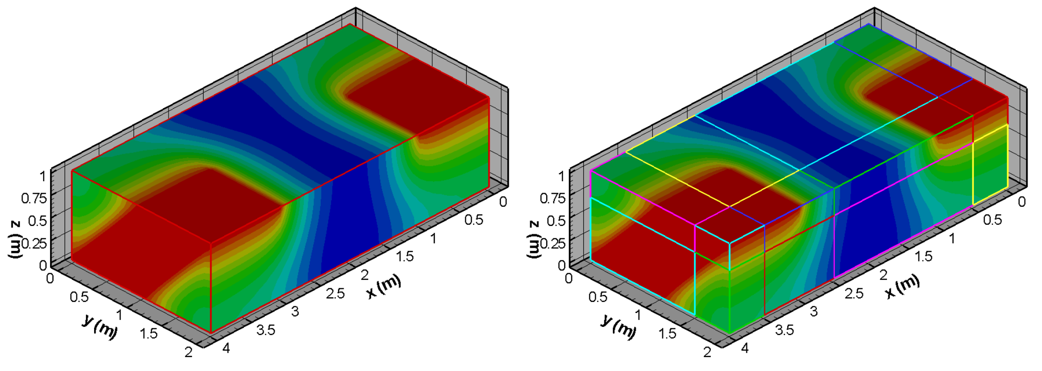


Figure 7.1: Profiling studies from previous version of Draco

for the small mesh size while the red ones are for the larger one. Similarly solid bars indicate the runs with 200k particles, while the dashed one are for runs with 500k particles. In this study, the PCG solver was used to obtain the plasma potential.

Reading from left to right, we can seen that functions utilizing the largest amount of computational time are `vcXtoL`, `SparseVectMult`, `vcDetermineZone`, `Scatter`, `GatherLocal`, and `GatherInit`. Of these, only one, `SparseVectMult` is performing an actual computation. All other functions are responsible for interpolating particle positions to and from the mesh! The function taking up the disproportionately largest chunk of computational resources is `vcXtoL`, which is the function that translates the physical coordinates into logical ones. This is the step that is supposed to be trivial for a Cartesian mesh, yet here we see that it is in fact dominating the code performance. In comparison, the actual particle move implemented in `Move` is trivial. As expected, the time is directly proportional to the number of simulation particles (the matrix-vector multiplication in `SparseVectMult` on the other hand demonstrates dependence on mesh size, as expected). The `vcDetermineZone` function contains the code searching for the particle zone which was added once multiple zones were implemented. Increasing the mesh size is mainly demonstrated as a performance hit in the PCG potential solver. Since most Draco simulations do not use a Poisson solver, this is a non-issue.



(a) (b)

Figure 7.2: Potential solution obtained for a single and multiple (16) zones

Hence, part of the development effort went into optimizing the code and improving the code performance. This was achieved by rewriting the internal mesh representation. Instead of treating the zones as virtual boundaries superimposed over a single contiguous mesh, the mesh was divided into separate chunks. Each chunk took the ownership of particles located in it. Since each zone contains only a single mesh definition, this rewrite completely eliminated the need to search for the containing zone. It also resulted in an improved memory utilization. It is hard to guarantee a large contiguous memory block on modern multi-process operating systems. Consider a computer system with 2 Gb of RAM. Now assume that the bottom 300Mb is used by the operating system. Next, assume the user loads a data analysis program which allocates the next 600 Mb of RAM. Next, the user opens a text editor that uses 10 Mb of data. Finally, the user closes the data analysis program, freeing the 600Mb of RAM, and starts a simulation. Assume the simulation uses a mesh with $250 \times 250 \times 250$ nodes and that the size of data stored on each node is 20 doubles (20*8 bytes). The total memory required to allocate this mesh is 1.19 Gb. Despite the operating system reporting 1.7Gb of free memory, this allocation request will fail. The reason is simple - the system lack a single contiguous memory block of this size. The memory structure in our simple example consists of 300Mb used by O/S, a 600 Mb empty space, 10 Mb block used by the text editor, and finally another empty 1138 MB (1.11 Gb) block. The behavior described in this example is actually fairly typical and previous experience showed that Draco simulations were typically limited to about $200 \times 200 \times 200$ nodes, despite the amount of available RAM. By dividing the mesh into a number of smaller blocks, it is more likely the operating system will be able to accommodate the allocation requests.

The downside of the zone split is that the some performance gain is lost to transferring particles across zone boundaries. However, since such a packing / unpacking operation is also required in the distributed parallel mode, the distributed zone handling approach in fact simplified development and debugging of the parallel code. Besides changing the way the particles move, the split into individual zones also affected the field solver and mesh properties. A boundary buffer was implemented to obtain data from neighboring zones. In the case of a processor boundary, the neighbor data is obtained via MPI send/receive commands. The neighbor data is used to add up properties such as particle densities along zone boundaries, and to compute potential and electric fields. Figure 7.2 shows potential around an arbitrary test geometry computed on a mesh segmented into 16 zones, and on a corresponding mesh consisting of only a single zone. The multi-zone case was in addition run in parallel with 4 zones per processor. As can be seen from this plot, the two solutions are identical.

The division of mesh into multiple zones had another benefit for parallel computations. Previously, due to difference in which zones and processor boundaries were handled, multi-zone meshes were not supported in parallel. In a parallel run, each zone was automatically assigned to a new processor. This imposed a severe limitation on the possible domain decomposition. Consider for instance a computer cluster containing 32 CPUs (CPU counts often come in powers of 2). In order to utilize all 32 CPUs, a $4 \times 4 \times 2$ domain decomposition is required. Such a zone definition is not always ideal. The new implementation completely divorces the number of zones and the number of available CPUs. Zone assignment is now performed in a round-robin case. In the case of a serial run, all zones are given to the first (and only) processor. If multiple computers are present, they are assigned zones in a sequential order: zone 0 goes to CPU 0, zone 1 to CPU 1, zone 2 to CPU 2, zone 3 to CPU 0, zone 4 to CPU 1, etc... This example assumes there are 3 CPUs in the simulation space. Such an assignment results in optimal work balancing for majority of cases. The only limitation is that for obvious reasons the domain contains at least as many zones as CPUs.

Finally, it should be noted that parallel communication across an arbitrary number of processors and zones is a non-trivial task. Care is needed to avoid deadlock, a situation in which processors keep waiting on each other. This is avoided in the new Draco by a assigning each face a unique faceID. Communication is performed by looping over the faceIDs consecutively. Each processor checks if it contains that particular faceID, and if it does, communication is performed. Additional care is also needed in the summation of shared data. It is imperative that the summation is performed one dimension at a time, otherwise, inconsistent values will be obtained along nodes shared by more than two zone.

7.2.2 Mesh Intersection

Second important change is related Draco's helper mesh-generation module *Volcar*. While rectilinear meshes speed up particle push and interpolation, they make it more difficult to capture realistic geometries. In finite-element analysis, the mesh shape contours to the surface, but in a rectilinear mesh a cut cell approach must be taken to capture the surface geometry. Draco uses a limited cut-cell approach in which the surface mesh is used to generate boundaries for particles, but a staircase representation is used by the potential solver. Volcar is responsible for classifying nodes as "internal" (fixed potential, assuming conductive objects) or "external" (free space), and also for collecting surface elements in interface volumes to be used for particle-surface impact checks. Volcar first generates the rectilinear zones according to the user domain definition. It then intersects the volume mesh with a surface mesh describing the geometry. Node location is based on the orientation of the surface normal vectors, with the vectors pointing into the free space.

The original Virginia Tech version of Draco utilized a hybrid structured/tetrahedral mesh. Each simulation cell (a brick) was subdivided into 5 tetrahedra (see, for instance, Figure 2 in [45]). The mesh intersection was performed not with the parent block but with each tet. The motivation for this implementation was that the VT Draco was coupled with an immersed finite element (IFE) solver that utilized such a mesh to solve for potential in presence of interface cuts [48]. Unfortunately, the IFE solver was never fully coupled into Draco, with one of the driving reasons being software language incompatibility. While Draco was developed in C, the IFE solver was written in Fortran 90. Not only did this required two sets of compilers, the difference in memory utilization also dictated the mesh structure to be duplicated in Draco and in IFE, doubling memory requirements. Finally, it was found that the IFE solver was not versatile enough for real-life situation. A basic assumption in the implementation of IFE was that the mesh is fine enough such that each tetrahedron is cut at most one time. Although it is fairly simple to satisfy this requirement for academic test cases consisting of bricks and cylinders, it was found that in practice it was impossible to meet this requirement for CAD-generated geometries. Real world problems typically contain thin features and sharp edges that will result in multiple surfaces cutting through a tetrahedron.

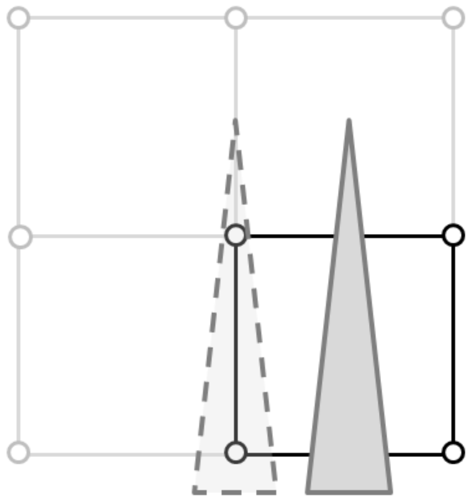
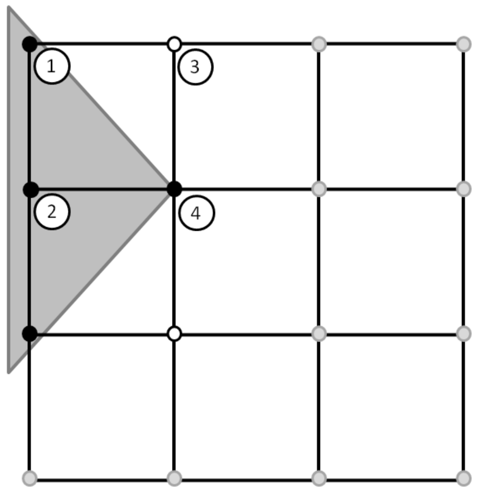


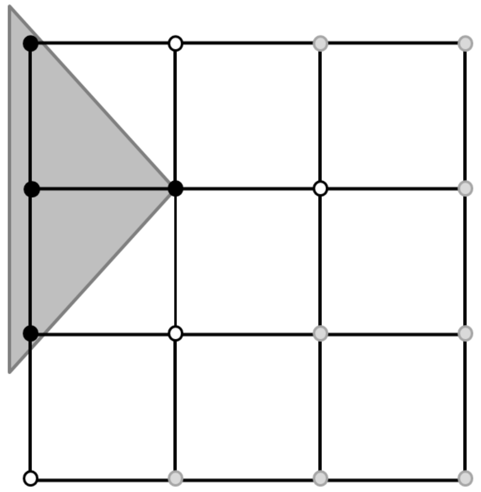
Figure 7.3: A typical thin feature resulting in multiple cell/edge cuts. In order to satisfy the single cut requirement, the wedge would need to be placed as shown by the dashed form.

For these reasons, the IFE solver was officially removed from Draco distribution around 2006. However, much of the mesh generation methodology was left in the code. Most important of these was the fact that Volcar was designed to operate on a cell-by-cell basis. The code looped through the cells and classified each cell as internal, external, or interface. Interface cells were those that had a cut passing through them. Internal and external cells were cells that were completely inside or outside a solid object. This methodology worked well if the single cut requirement was met, however again failed to work for real world cases. To illustrate this, consider a cell containing a thin feature as demonstrated in Figure 7.3. It should be pointed that 2D representation is used in this and subsequent plots for clarity, however, Volcar operates in 3D. This cell is no longer interface in the original sense of the definition, since it does not form an interface between an internal and external region (both left and right side are in free space). However, marking this cell as external would disregard the fact the cell contains surfaces that need to be check for particle interactions.

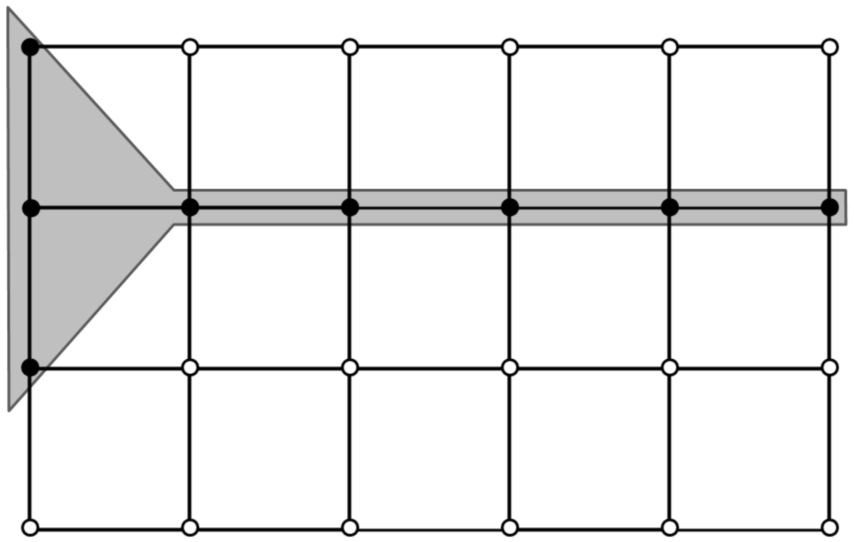
In addition, Volcar uses interface cuts to classify nodes as internal or external. Situation like the one depicted in Figure 7.3 could in some instances result in an incorrect node classification if the incorrect surface triangle is used. Node classification is performed using the first visible surface with visibility determined by checking for no other triangle intersecting the ray from the node to the triangle centroid. In some peculiar instances, due to the cell-based approach, not all surfaces needed to be checked were captured. Another issue arose when surfaces terminated at a cell



(a) Incorrect classification



(b) Correct classification



(c) Floodfill resulting from the incorrect classification

Figure 7.4: This figure shows a typical problem in the earlier version of Volcar. Due to the cell-based approach, external nodes (shown by open circles) would not fully enclose the internal (solid) nodes. Then the subsequent floodfill used to classify remaining unknown (gray) nodes would result in propagation of "spikes". Since internal nodes are used to fix object potential, this error distorted the object shape.

boundary. Consider the example shown in Figure 7.4a. The surface can be seen to "belong" to cells 1 and 2, however, a single vertex is shared with the neighboring cells 3 and 4. Often, due to the cell-by-cell approach and numerical imprecisions, the code would fail to capture the single point located in cells 3 and 4. These cells would then be marked as completely free of surfaces. Classification of node locations in the interface cells 1 and 2 would result in the situation depicted in Figure 7.4a. In this figure solid black markers indicate nodes that have been classified as internal, solid white markers are external nodes, and gray nodes are "unknown" nodes that have not been classified. The unknown nodes are classified by performing a flood fill from the set near-surface nodes. For this step to work correctly, a clear interface must exist around all surfaces. A well defined interface has internal and external nodes set on both sides of the surface, as depicted in Figure 7.4b. The difference is the single externally marked node one cell away from the shared vertex. In order for this node to be set, the cells 3 and 4 had to be marked as interface and contain the correct surface definition. In the absence of this single node being classified, the subsequent flood fill, assuming filling in the +X direction, will result in the situation depicted in 7.4c. The internal nodes will propagate to the mesh boundary or until another previously set node is encountered. Such anomalous "sticks" were a common plague of many older Draco simulations.

Resolving this issue required changing the methodology from a cell-centered to a node-centered approach. Instead of treating a mesh cell as the correct volume to intersect by the surface, a node-centered control volume (NCV) should be used instead. This approach is plotted in Figure 7.5. In the new implementation, the control volume extends by one cell in each dimension. This definition guarantees an overlap between control volumes and reduces the risk of some surfaces not being captured during particle surface intersection check. Although additional testing remains, this new implementation seems to have fully mitigated the issues present in previous Draco simulations.

7.2.3 Time-Dependent Source Model

Hall thrusters are known to be non-steady devices demonstrating a wide range of fundamentals frequencies [13]. The oscillations most easily observed in numerical and experimental studies is a prominent low frequency "breathing mode". This oscillation arises from a prey-predator type depletion and repletion of neutrals and ions. In [13] the frequency range for this type of oscillations is given as 15-22 kHz. Figure 7.6 shows the temporal variation in discharge current from a typical HPHall study of the

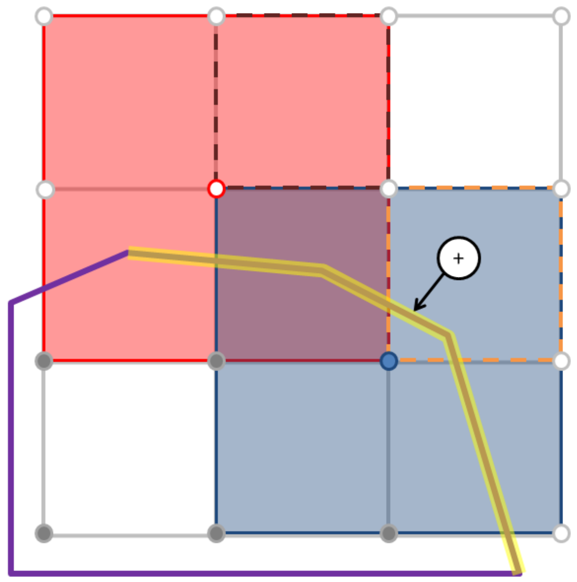


Figure 7.5: The two colored squares show the node control volume (NCV) surrounding each node as well as the overlap between neighboring NCVs. Node classification is performed using surfaces located in the NCV. The dashed boundary shows the cell in the +X,+Y and +Z direction. Particle located in this cell are checked for surface intersections using surfaces located in the NCV (highlighted in yellow).

Princeton Cylindrical Hall thruster. The frequency of the oscillations, as computed by inverting time differences between several prominent peaks, is seen to be in the 17 to 30 kHz range, indicating that this indeed is the breathing mode.

Draco has been coupled previously coupled with HPHall in order to capture the details of the ion beam population [49]. This coupling is accomplished by sampling ions crossing a user defined radial grid line and storing their positions and velocity components into a file. This file thus gives us a radially-varying discretized velocity distribution function. In Draco simulations, particles are injected into the domain by associating *source models* with surface elements. These surfaces then emit particles according to the source-specific distribution functions and according to user defined parameters such as mass flow rate and mean velocity. Standard analytical models such as the Maxwellian and Lambertian (cosine) distribution do not offer a good representation of Hall thruster ions. The reasons are twofold. First, flux is not radially uniform due to a difference in density and velocity as a function of thruster radius. Secondly, ion beam velocity distribution function is also radially varying and non-Gaussian due to the self-induced accelerating forces. This starkly contrasts Hall thrusters from ion thrusters in which a cold ion population is accelerated through grids with a fixed applied potential gradient.

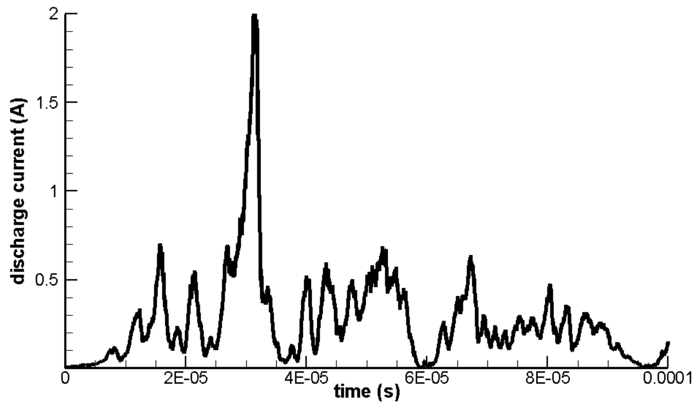


Figure 7.6: Discharge current from a typical HPHall simulation of the CHT showing a strong time dependence.

As described in [49], the HPHall source was indeed able to better reproduce the ion beam population. However, in this previous implementation, no note was paid to the temporal behavior. Instead, the thruster was assumed to operate in a steady state, and ions sampled over several breathing mode oscillations were conglomerated. It is not clear how much importance does the temporal behavior play on plume dynamics. Due to the relaxing nature of collisions, it is possible that the variations in the source will actually be dissipated in the plume. To study this behavior in more detail, the HPHall source was modified to optionally read a time stamp at which the particle data was sampled. The sampling function in HPHall was also modified to export the time, given by `it*DT`, and also the azimuthal velocity, computed in the code as $v_\theta = \hat{h}/r$, where $\hat{h}$ is a mass-normalized angular momentum. To reduce the size of the collected list, a user-specified probability is used to select particles. Since in HPHall ions have a variable specific weight, the probability that a particle will be sampled is obtained by comparing the particle weight to a user specified output weight, $n_{sample} = w_p/(m_{Xe+} * sw_{sample}) + R$. Here w_p is the particle weight, m_{Xe+} is the mass of an actual Xenon ion, sw_{sample} is the desired specific weight, and R is a random number used to reduce round off errors.

If the time information is present, the source bins particles into a user specified number of Δt_b time slots (default is 100). This binning is illustrated in Figure 7.7. At each call to sample a particle, the source determines the bin containing the current simulation time, $j = \text{int}((t - t_0)/\Delta t_b)$. To assure a smooth transition between bins, particle will be sampled from bin j or $j+1$ based on its proximity to the boundary. More specifically, the cell coordinate $f = (t-t_0)/\Delta t_b - j\Delta t_b$ is compared to a random number R. If $f \leq R$, a particle is sampled from bin j, otherwise it sampled from

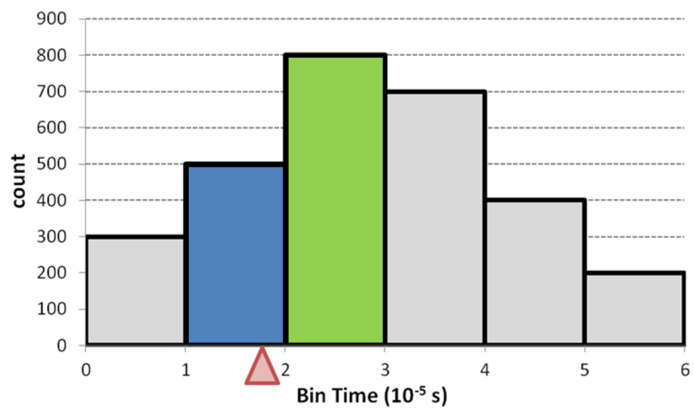


Figure 7.7: At a time indicated by the triangle, a particle will be sampled randomly from one the two colored bins, with a stronger preference given to the green bin.

bin $j + 1$. The sampled particle position Δz and Δr form an offset from the source centroid. The position and the velocity components v_r, v_θ, and v_z are also rotated through a random angle about the thruster centerline. The math used in the rotation is described in more detail in [49] and [47].

The time data is optional, and if it is not present, the source will sample particles from the entire given list. A simple, manually-generated example of input is given below:

```
VARIABLES = z r vz vr vt t
ZONE T=HPHALL_XE+
0 0.50 10000 0 0 0
0 0.50 10000 0 0 0
0 0.50 10000 0 0 0
0 0.50 10000 0 0 0
0 0.50 10000 0 0 0
0 0.25 8000  0 0 2e-4
0 0.25 8000  0 0 2e-4
0 0.25 8000  0 0 2e-4
0 0.00 2000  0 0 4e-4
0 0.25 8000  0 0 6e-4
0 0.25 8000  0 0 6e-4
```

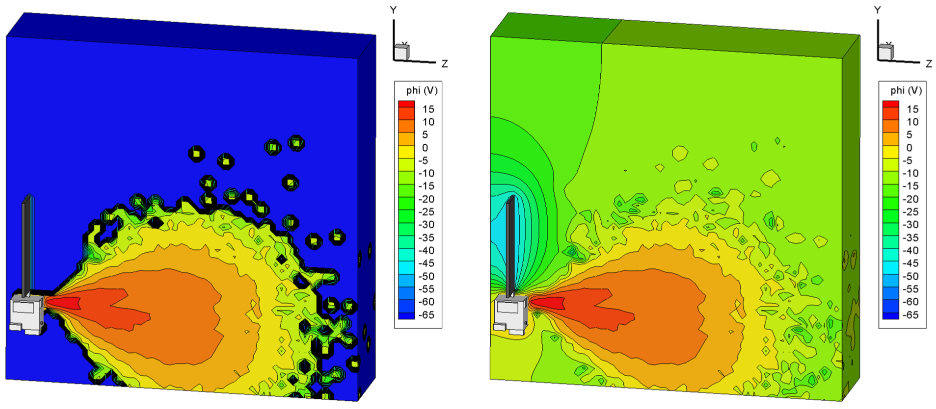
This input shows two important features of the time-variable source. Both position/velocity and also mass flow rate can be a function of time. The variation in mass flow is captured automatically by the number of entries for different time slots.

The mass flow rate at a specific time is obtained from $\dot{m}_j = \dot{m}_0(c_j/\bar{c})$ where c_j is the number of particles in bin j and $\hat{c}$ is the average across all bins. $\dot{m}_0$ is the user specified mass flow rate that will be generated over the entire sampling period. The result is that, after sampling over the entire given time range, the source will produce the defined mass flow rate.

Besides specifying the number of segments and the average mass flow rate, the source also take two additional parameters specifying the initial time t_0 and a cutoff time t_f. The first parameter t_0 can be used to shift the place in the input file where the sampling will begin. This parameter is useful if for instance we are interested at beginning the particle injection at the peak of discharge current, in which case t_0 would be set to the time at which the first peak occurs. The second parameter is used to set the time in the input file at which the sampling will loop to the beginning. Typically, the input data to the source will contain HPHall exit plane data over a few thruster oscillations. However, the plume simulation will be run over a greater time scale, making it necessary to loop through the input data multiple times. The source will perform this looping automatically once the simulation time exceeds the sampling range. However, to avoid anomaly in the plume results, it is necessary to match the end and start data points. The matching can be performed using several ways, one of which is by matching the phases in the discharge current. As such, the recommended procedure is to visualize the discharge current log data from HPHall over the time range exit plane sampling was performed. As an example, these parameters could be used to bracket the data in Figure 7.6 between 4×10^{-5} and 9×10^{-5} seconds. After the Draco simulation completed 5×10^{5} seconds, the sampling would continue from the 4×10^{-5} marker. It should be pointed out that in order to capture all sampled particles, the bin size must be set such that $\Delta t_b \geq \Delta t_{sim}$. If these optional parameters are not specified, no time offset is used and the entire time sequence is used for sampling.

7.2.4 Quasineutral Potential Fix

Finally, a "quasi-neutral" switch has been implemented in the Draco Gauss-Seidel Solver. Implementation in the PCG solver is pending. The switch allows the use of the Poisson solver on meshes too coarse to solve otherwise. A Poisson solver will not converge in the cell spacing is significantly larger than the local Debye length. In the plume core where $n \sim 10^{17}$ m^{-3} and $kT_e \sim 5$ eV, the Debye length $\lambda_D = \sqrt{\epsilon_0 kT_e/ne^2} \sim 5 \times 10^{-5}$ m. Even with the use of a stretched mesh, it becomes computationally infeasible to resolve the Debye length while at the same time



(a) Boltzmann Inversion (b) QN switch

Figure 7.8: Comparison in potential between standard Boltzmann inversion and QN-switched Poisson solver. The sheath structure around the solar wing is clearly absent from the inverted solution.

resolving the plume far-field. As such, the plume potential is typically obtained by assuming quasi-neutrality $n_e = n_i = n$ and inverting the Boltzmann relationship to obtain $\phi = \phi_0 + kT_e \ln(n/n_0)$. Although this approach is valid in the plume, it has the serious handicap in that it does not resolve the non-neutral sheath. As such potential will be correct in the plume, but electric fields surrounding charged objects in the low-density region will not be included. Resolving these fields is on the other hand imperative, since they drive the trajectories of the backflowing ions.

The QN switch was implemented to resolve this limitation. It follows the implementation in Aquila's field solver, [41]. At the start of each Poisson solver call, the local Debye length is calculated at each node using the local value of plasma density and electron temperature. If the node volume is greater than the volume of the Debye sphere, the node is flagged as Dirichlet and potential is fixed on it from the Boltzmann inversion. Poisson equation is solved on the remaining nodes. Hence, the potential in high-density plume is computed directly from the inversion, and the Poisson solver backfills the low-density sheath-dominated region. The Poisson backfill is performed using a different set of reference parameters for the Boltzmann electron relationship, $n_e = n_0 \exp((\phi - \phi_0)/kT_e)$ to capture the properties of the ambient plasma. This feature is demonstrated in Figure 7.8. The first plot shows a potential profile computed using the standard Boltzmann inversion. The second plot shows the solution using the QN switch approach. Both of these plots were generated by loading a particle distribution from a restart file and outputting the initial potential.

To better illustrate the difference, the plume simulation was performed without including collisions. The potential in the plume is seen to be identical, as expected. However, the sheath surrounding the negatively charged solar wing is clearly absent in the Boltzmann inversion solution. As such, the first solver will underpredict the current collected by the wing. The sheath acts to increase the effective volume from which ions are extracted and accelerated towards the surface. In the absence of the sheath, ions are unaware of the existing potential drop until they arrive in a cell adjacent to the surface.

7.3 Results

We demonstrate the plume modeling step using a hypothetical space weather satellite. Small Hall thrusters such as teh Princeton CTH may be an attractive replacement for arcjets commonly used for station keeping. The surface model was shown previously in Figure 7.8. This generic satellite consists of the bus, a solar panel, a cluster of two thrusters, and several instrument boxes. The bottom left box on the exposed face in Figure 7.8 is assumed to contain a space weather instrument designed to measure the flux of low energy ions. We are interested in determining the extent to which the Hall thruster will modify the local plasma environment, and the amount of additional current collected due to the thruster plume backflow (here we are assuming that this instrument does not contain any additional mass-spectrometers that could differentiate the heavy thruster ions from the light solar protons). In addition, we are interested to determine whether the temporal characteristic of the thruster discharge will be captured by the instrument.

We assume the spacecraft is operating in the GEO environment where the baseline ambient plasma environment is almost negligible, $n_i = 5.8 \times 10^5 \text{m}^{-3}$[3]. For simplicity we do not consider the large surface potentials that often arise in GEO satellites due to charging, and set all spacecraft surfaces, including the solar wing, to 0V. The only exception is the space weather instrument, which is assumed to operate at 50V below the ambient plasma to repel low-energy electrons. Ion particles are injected using the data sampled from the thruster simulation corresponding to the discharge current in Figure 7.6. The QN-switched potential solver is used to obtain the plasma potential. The reference parameters used were (ϕ_0, n_0, kTe_0) = 20V, 10^{17} m^{-3}, and 3 eV in the beam plasma, and 0V, 5.8^5 m^{-3}, and 1 eV in the ambient backfill region. Collisions are modeled with the Monte Carlo Collisions (MCC) approach. Only CEX interaction were considered in this work. In addition, neutrals were not modeled directly, instead

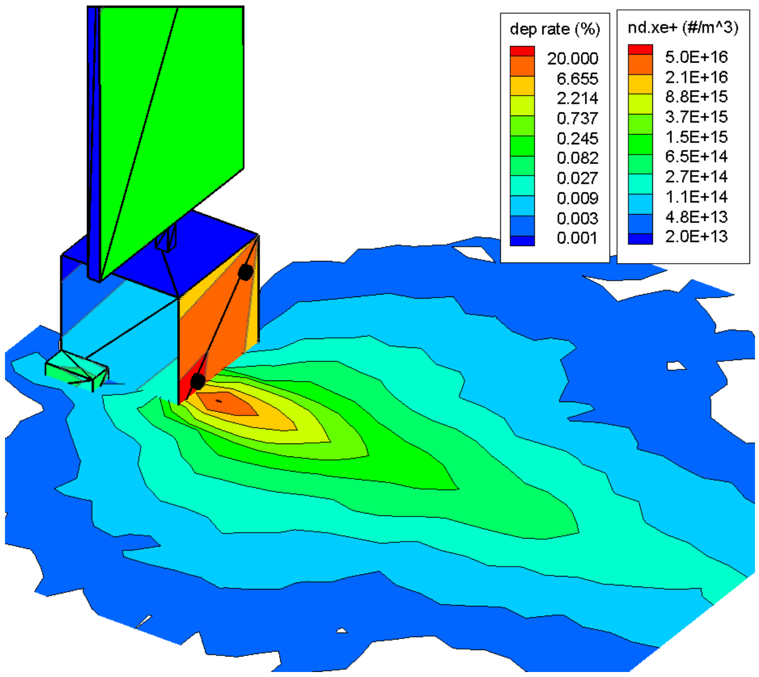


Figure 7.9: Simulation results showing the surface deposition rate, normalized by thruster mass flow rate, and the plume density in m^{-3}.

the neutral plume was represented by a fluid projection.

The simulation was let run for 4000 time steps. Total of 2 million particles were used in the simulation. Such a large number was needed in order to obtain a statistically significant current in the low-density wake region. Results averaged starting at the steady-state are shown in Figure 7.9. This figure contains two sets of contours. The contour levels on the slice through the plume show the ion number densities. The radial expansion of the plume is clearly evident and so is the increased density near the negatively charged instrument. The ion density in the vicinity of the instrument is 5×10^{13} m^{-3}, a four-order reduction from the plume densities, but an O(8) increase from the ambient environment.

The contour levels on the surface correspond to the deposition rate, normalized by the source mass flow rate. As can be seen from this figure, the surface is described by a very coarse mesh (the spacecraft geometry was generated analytically by combining basic building blocks since a CAD package was not available). While this coarse mesh does not provide us with details of the surface collection, it at least allows us to quantify the collection. We can see that majority of backflow occurs in the region behind the thruster, which can be expected. The solar wing also collects a fraction of the backflowing plume. By summing up the exposed faces of the instrument of interest, we can seen that it receives 0.3% of the emitted flux. It should be noted that

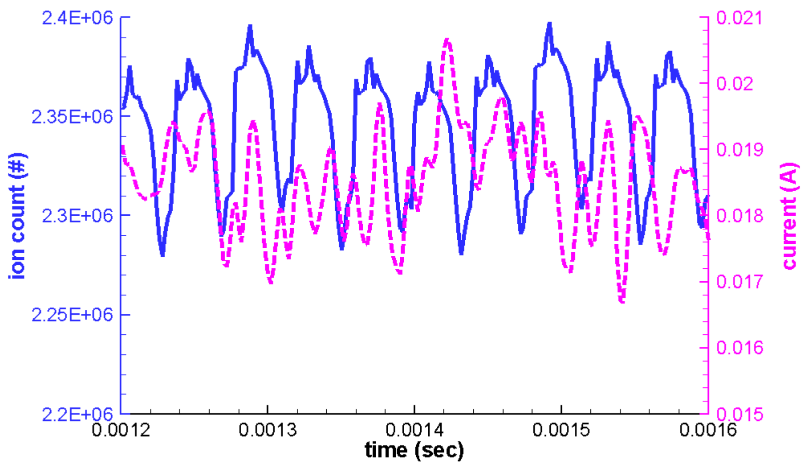


Figure 7.10: Comparison between the simulation particle count at steady state and the instrument collected current.

in a real system, the collection rate would be significantly lower since the instrument aperture represents only a small fraction of the total instruments surface area.

Figure 7.10 shows the temporal response. The blue curve plots the total number of particles in the simulation as a function of time. The count is seen to oscillate due to the time-dependent source. The purple curve shows the current collected by the instrument. The temporal behavior is clearly also transferred to the backflowing plume. However, it appears that the frequency present in the backflowing plume has doubled from the source frequency, since on average two prominent peaks can be seen for each peak in the particle count. The physical reason behind this, as well as the additional detail of the temporal response of the CEX ions remains as future work.

Bibliography

[1] V. Zhurin, H. Kaufman, and R. Robinson, "Physics of closed drift thrusters," *Plasma Sources Science and Technology*, vol. 8, no. R1, 1999.

[2] S. Now, "Pppl hall thruster experiment (htx)." `http://htx.pppl.gov`.

[3] D. Hastings and H. Garrett, *Spacecraft-environment interactions.* Cambridge University Press, Cambridge, UK, 2004.

[4] J. Szabo, *Fully Kinetic Numerical Modeling of a Plasma Thruster.* PhD thesis, Massachusetts Institute of Technology, 2001.

[5] M. Hirakawa and Y. Arakawa, "Particle simulation of plasma phenomena in hall thrusters," in *Proceedings of the 24th International Electric Propulsion Conference (Moscow), IEPC-95-164*, 1995.

[6] Y. Raitses, D. Staack, M. Keidar, and N. J. Fisch, "Electron-wall interaction in hall thrusters," *Physics of Plasmas*, vol. 12, no. 047104, 2005.

[7] A. I. Morozov, Y. V. Esinchuk, G. N. Tilinin, A. Trofimov, Y. A. Sharov, and G. Y. Shchepkin, "Plasma accelerator with closed electron drift and extended acceleration zone," *Soviet Physics – Technical Physics*, vol. 17, no. 1, 1972.

[8] M. Lampe, G. Joyce, W. Manheimer, and S. P. Slinker, "Quasi-neutral particle simulation of magnetized plasma discharges: General formalism and application to ecr discharges," *IEEE Transcations on Plasma Science*, vol. 26, no. 6, 1998.

[9] S. Now, "Patience required as aehf 1 recovery begins new mode." `http://spaceflightnow.com/atlas/av019/101017hct.html`.

[10] D. Sydorenko, A. Smolyakov, I. Kaganovich, and Y. Raitses, "Modification of electron velocity distribution in bounded plasmas by secondary electron emission," *IEEE Transactions on Plasma Science*, vol. 34, no. 3, 2006.

[11] C. Birdsall, "Particle-in-cell charged-particle simulations, plus monte carlo collisions with neutral atoms, pic-mcc," *Plasma Science, IEEE Transactions on*, vol. 19, no. 2, pp. 65–85, 1991.

[12] Wikipedia, "Tridiagonal matrix algorithm," 2011. `http://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm`.

[13] E. Choueiri, “Plasma oscillations in hall thrusters,” *Physics of Plasmas*, vol. 8, p. 1411, 2001.

[14] J. Fox, *Advances in Fully-Kinetic PIC Simulations of a Near-Vacuum Hall Thruster and Other Plasma Systems.* PhD thesis, Massachusetts Institute of Technology, 2007.

[15] J. Boris and R. Lee, “Non-physical self forces in electromagnetic plasma-simulation algorithms,” in *NRL Memorandum Report 2418*, pp. 1–10, 1972.

[16] C. Birdsall and A. Langdon, *Plasma physics via computer simulation.* Institute of Physics Publishing, 2000.

[17] M. Lieberman and A. Lichtenberg, *Principles of plasma discharges and materials processing.* Wiley-interscience, 2005.

[18] A. Dunaevsky, Y. Raitses, and N. Fisch, “Secondary electron emission from dielectric materials of a hall thruster with segmented electrodes,” *Physics of Plasmas*, vol. 10, p. 2574, 2003.

[19] Y. Raitses and N. Fisch, “Parametric investigations of a nonconventional hall thruster,” *Physics of Plasmas*, vol. 8, no. 5, 2001.

[20] L. Brieda, M. Keidar, Y. Raitses, and N. Fisch, “Self-consistent calculation of electron transport in a cylindrical hall thruster,” in *31st International Electric Propulsion Conference, Ann Arbor, MI*, 2009.

[21] Y. Raitses, A. Smirnov, and N. Fisch, “Effects of enhanced cathode electron emission on hall thruster operation,” *Physics of Plasmas*, vol. 16, no. 057106, 2009.

[22] A. Smirnov, Y. Raitses, and N. Fisch, “Plasma measurements in a 100 w cylindrical hall thruster,” *Journal of Applied Physics*, vol. 95, no. 5, 2004.

[23] B. Reid and A. Gallimore, “Plasma potential measurements in the discharge of a 6-kw hall thruster,” in *44th Joint Propulsion Conference, Hartford, CT*, 2008.

[24] A. Fruchtman and A. Cohen-Zur, “Plasma lens and plume divergence in the hall thruster,” *Applied Physics Letters*, vol. 89, no. 111501, 2006.

[25] M. Keidar and I. I. Beilis, “Sheath and boundary conditions for plasma simulations of a hall thruster discharge with magnetic lenses,” *Applied Physics Letters*, vol. 94, no. 191501, 2009.

[26] M. Keidar, I. D. Boyd, and I. I. Beilis, “Plasma flow and plasma-wall transition in hall thruster channel,” *Physics of Plasmas*, vol. 8, no. 12, 2001.

[27] M. Keidar and I. D. Boyd, “On the magnetic mirror effect in hall thrusters,” *Applied Physics Letters*, vol. 87, no. 121501, 2005.

[28] J. Fife and M. Martinez-Sanchez, "Two-dimensional hybrid particle-in-cell (pic) modeling of hall thrusters," in *24th International Electric Propulsion Conference, Moscow, Russia*, pp. 1213–1224, 1995.

[29] F. Parra, E. Ahedo, J. Fife, and M. Martinez-Sanchez, "A two-dimensional hybrid model of the hall thruster discharge," *Journal of Applied Physics*, vol. 100, no. 023304, 2006.

[30] E. Ahedo, R. Santos, and F. Parra, "Fulfillment of the kinetic bohm criterion in a quasineutral particle-in-cell model," *Physics of Plasmas*, vol. 17, p. 073507, 2010.

[31] Y. Garnier, V. V., J. F. Roussell, and J. Bernard, "Anomalous conductivity and secondary electron emission in hall effect thrusters," *Journal of Vacuum Science and Technology A*, vol. 17, p. 3246, 1999.

[32] J. Yim, M. Keidar, and L. Boyd, "A hydrodynamic-based erosion model for hall thrusters," in *29th International Electric Propulsion Conference, Princeton, NJ*, 2005. IEPC–2005–013.

[33] J. Linnell and A. D. Gallimore, "Internal plasma potential measurements of a hall thruster using plasma lens focusing," *Physics of Plasmas*, vol. 13, no. 103504, 2006.

[34] B. Beal and A. Gallimore, "Energy analysis of a hall thruster cluster," in *28th International Electric Propulsion Conference*, 2003. IEPC-2003-055.

[35] I. Boyd and R. A. Dressler, "Far field modeling of the plasma plume of a hall thruster," *Journal of Applied Physics*, vol. 92, no. 4, 2002.

[36] R. I. Samanta Roy, *Numerical Simulation of Ion Thruster Plume Backflow for Spacecraft Contamination Assessment.* PhD thesis, Massachusetts Institute of Technology, 1995.

[37] J. Wang, D. Brinza, and M. Young, "Three-dimensional particle simulations of ion propulsion plasma environment for deep space 1," *Journal of Spacecraft and Rockets*, vol. 38, pp. 433–440, 2001.

[38] A. Tribble, *Fundamentals of contamination control.* SPIE, 2000.

[39] M. Mandell, V. Davis, D. Cooke, A. Wheelock, and C. Roth, "Nascap-2k spacecraft charging code overview," *IEEE Transactions on Plasma Science*, vol. 34, no. 5, 2006.

[40] M. Tajmar, W. Meissl, J. del Amo, B. Foing, H. Laakso, G. Noci, M. Capacci, A. Malkki, W. Schmidt, and F. Darnon, "Charge-exchange plasma contamination on smart-1: First, measurements and model verification," in *40th AIAA Joint-Propulsion-Conference*, (Fort Lauderdale, FL), 2004.

[41] M. Santi, S. Cheng, M. Celik, M.-S. M., and P. J, "Further development and preliminary results of the aquila hall thruster plume model," in *39th Joint Propulsion Conference, Huntsville, Alabama*, 2003.

[42] J. Forest, A. Hilgers, B. Thiébault, L. Eliasson, J. Berthelier, and H. de Feraudy, "An open-source spacecraft plasma interaction simulation code picup3d: tests and validations," *Plasma Science, IEEE Transactions on*, vol. 34, no. 5, pp. 2103–2113, 2006.

[43] M. Gibbons, D. Kirtley, D. VanGilder, and J. Fife, "Flexible three-dimensional modeling of electric thrusters in vacuum chambers," in *39th AIAA Joint Propulsion Conference*, (Huntsville, Al), 2003. AIAA-2003-4872.

[44] L. Brieda, "Development of the DRACO ES-PIC code and fully-kinetic simulation of ion beam neutralization," Master's thesis, Virginia Tech, 2005.

[45] L. Brieda, R. Kafafy, J. Pierru, and J. Wang, "Development of the Draco code for modeling electric propulsion plume interactions," in *Proceedings of the 40th Joint Propulsion Conference, Fort Laudardale, FL, USA*, pp. 1–21, 2004.

[46] J. Pierru, "Development of a parallel electrostatic pic code for modeling electric propulsion," Master's thesis, Virginia Polytechnic Institute and State University, 2005.

[47] R. Spicer, "Validation of the draco particle-in-cell code using busek 200w hall thruster experimental data," Master's thesis, Virginia Polytechnic Institute and State University, 2007.

[48] R. Kafafy, T. Lin, Y. Lin, and J. Wang, "Three-dimensional immersed finite element methods for electric field simulation in composite materials," *International journal for numerical methods in engineering*, vol. 64, no. 7, pp. 940–972, 2005.

[49] M. Nakles, L. Brieda, G. D. Reed, W. A. Hargus, and R. Spicer, "Experimental and numerical examination of the BHT-200 hall thruster plume," in *43rd AIAA Joint Propulsion Conference, Cincinnati, OH*, 2007. AIAA-2007-5305.