DEVELOPMENT OF A SCALABLE GAS-DYNAMICS SOLVER
WITH ADAPTIVE MESH REFINEMENT

A Dissertation in
Aerospace Engineering
by
Burak Korkut

© 2015 Burak Korkut

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2015
The dissertation of Burak Korkut was reviewed and approved* by the following:

Deborah A. Levin  
Professor of Aerospace Engineering  
Dissertation Adviser  
Co-Chair of Committee

Sven G. Bilén  
Professor of Aerospace Engineering  
Co-Chair of Committee

Kamesh Madduri  
Assistant Professor of Computer Science and Engineering

Kenneth S. Brentner  
Professor of Aerospace Engineering

George A. Lesieutre  
Professor of Aerospace Engineering  
Head of the Department of Aerospace Engineering

*Signatures are on file in the Graduate School.
Abstract

There are various computational physics areas in which Direct Simulation Monte Carlo (DSMC) and Particle in Cell (PIC) methods are being employed. The accuracy of results from such simulations depend on the fidelity of the physical models being used. The computationally demanding nature of these problems make them ideal candidates to make use of modern supercomputers. The software developed to run such simulations also needs special attention so that the maintainability and extendability is considered with the recent numerical methods and programming paradigms. Suited for gas-dynamics problems, a software called SUGAR (Scalable Unstructured Gas dynamics with Adaptive mesh Refinement) has recently been developed and written in C++ and MPI. Physical and numerical models were added to this framework to simulate ion thruster plumes. SUGAR is used to model the charge-exchange (CEX) reactions occurring between the neutral and ion species as well as the induced electric field effect due to ions. Multiple adaptive mesh refinement (AMR) meshes were used in order to capture different physical length scales present in the flow. A multiple-thruster configuration was run to extend the studies to cases for which there is no axial or radial symmetry present that could only be modeled with a three-dimensional simulation capability. The combined plume structure showed interactions between individual thrusters where AMR capability captured this in an automated way. The back flow for ions was found to occur when CEX and momentum-exchange (MEX) collisions are present and strongly enhanced when the induced electric field is considered. The ion energy distributions in
the back flow region were obtained and it was found that the inclusion of the electric field modeling is the most important factor in determining its shape. The plume back flow structure was also examined for a triple-thruster, 3-D geometry case and it was found that the ion velocity in the back flow region appears to be roughly independent of the number of thrusters. The influence of solar cell panel voltage on backflow flux is evaluated using a combined commercial software generated potential with the induced potential and the influence of gas-surface interactions is studied. Finally, ion thruster plumes were simulated for vacuum and chamber configurations to characterize changes in the flow field.
Table of Contents

List of Tables ................................................................. ix

List of Figures ................................................................. x

Nomenclature ................................................................. xvi

Acknowledgments ............................................................ xix

Chapter 1. Introduction .................................................. 1

1.1 Electric Propulsion Devices .......................................... 1

1.2 Numerical Modeling of Electric Propulsion Devices .............. 3

1.3 Importance of high performance computing ...................... 7

1.4 Dissertation Structure ................................................. 8

Chapter 2. Development of a Scalable Gas Dynamics Software .... 9

2.1 Adaptive Mesh Refinement ............................................ 9

2.2 Parallelization .......................................................... 11

2.2.1 Cartesian Mesh ...................................................... 12

2.2.2 Communication via MPI .......................................... 12

2.3 Domain Decomposition ................................................. 13

2.3.1 2D Regular Blocking ............................................. 13

2.3.2 Integration with a Graph Partitioner ......................... 14

2.4 Supplemental Libraries ............................................... 20
2.4.1 Python ........................................... 20
2.4.2 Pugixml .......................................... 20
2.4.3 VTK ............................................. 23
2.4.4 Tecplot .......................................... 26

Chapter 3. Development of Modules for an Ion Thruster Application ........... 27
  3.1 Numerical Techniques .................................. 27
    3.1.1 DSMC ........................................ 27
    3.1.2 PIC ........................................ 29
  3.2 Physical Models ...................................... 34
  3.3 Modeling of the Vacuum Chamber ......................... 41
  3.4 Use of Multiple AMR Meshes ............................ 43

Chapter 4. Comparison of SUGAR with a 2-level DSMC code ......................... 47
  4.1 Comparison of Performance and Results of Neutral Expansion to Vacuum Simulations using a Cartesian Versus an AMR-Octree Grid .... 47
  4.2 Hypersonic Flow over a Double-wedge Configuration ........................ 53

Chapter 5. Application of SUGAR on Ion Thrusters Plumes ........................ 61
  5.1 Single Ion Thruster Cases ................................ 65
  5.2 Three Ion Thruster Cases ................................ 78

Chapter 6. Characterization and Quantification of Backflow Due to Ion Thruster Plumes .................................................. 86
  6.1 Single Thruster Cases ................................... 89
B.2 Installing dependent libraries .................................. 159
B.3 Compilation ......................................................... 159
B.4 Compilers .......................................................... 160
B.5 Scripts .............................................................. 160
C Main modules .......................................................... 161
C.1 General Workflow ................................................... 161
C.2 Screen Output ....................................................... 163
C.3 Performance and Scalability ....................................... 168
C.4 Restart Capability .................................................. 168
C.5 Module List .......................................................... 168
C.6 Python scripts ....................................................... 174
  C.6.1 do.py .......................................................... 174
  C.6.2 post.py .......................................................... 176
D Future Extensions .................................................... 178

References .................................................................. 179
List of Tables

4.1 Comparison of numerical parameters between SMILE and SUGAR . . . 49
4.2 Numerical parameters for the flow of argon over a double-wedge. . . . 56
5.1 Definition of cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
5.2 Exit conditions for the thrusters . . . . . . . . . . . . . . . . . . . . . . . 65
6.1 Selected macro parameters for Xe as a function of angle from the thruster centerline.\textsuperscript{a} . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
6.2 Selected macroparameters for Xe\textsuperscript{+} as a function of angle from the thruster centerline.\textsuperscript{a} . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
6.3 Conditions for upstream and downstream solar panel configurations, Xe 111
7.1 Numerical Parameters for DSMC Simulations . . . . . . . . . . . . . . . 114
7.2 Thruster Plume Exit Conditions . . . . . . . . . . . . . . . . . . . . . . . . 116
7.3 Simulation Domain and Background Conditions for different Pump Sizes\textsuperscript{a} 132
List of Figures

2.1 Schematic description of tree structure .......................... 11
2.2 Speed-up plot for SUGAR with load-balancing study ............. 15
2.3 Processor topology with 2-D blocking algorithm .................. 16
2.4 Processor topology with ParMETIS implementation for the single thruster case .................................................. 21
2.5 Processor topology with patched based approach for the four-thruster case .................................................. 22
2.6 Individual representation of data cells ............................. 24
2.7 Simultaneous representation of data fields .......................... 25
3.1 Schematic of octree cell indexing for solution of electric field. ... 31
3.2 Electrical boundary conditions and geometry ....................... 35
3.3 Convergence study for $E_{BC}$ at $y = 0.5$ m and $z = 1$ m of Fig. 3.2. . . 36
3.4 Comparison of the effect of numerical schemes on neutral velocity . . 40
3.5 Three computational meshes used for Case 3 ....................... 45
3.6 V-Mesh, visualization mesh for a single thruster .................. 46
4.1 Computational domain for SUGAR-SMILE comparison studies . . . 48
4.2 Neutral number density and x-component of velocity contours predicted by SUGAR for a four thruster configuration ....................... 51
4.3 Comparison of neutral number density and velocity profiles along the plume centerline for SUGAR and SMILE ....................... 52
4.4 Schematic of the double-wedge. ................................................. 55
4.5 Collision mesh comparison for SUGAR and SMILE. .................... 57
4.6 Comparison of contours for the flow of argon over a double-wedge. .... 58
4.7 3-D effects for the flow of argon over a double-wedge in SUGAR. .... 59
4.8 Comparison of macroparameters for the flow of argon over a double-wedge. 60

5.1 Computational domain for a single ion thruster ............................. 62
5.2 Computational domain for three ion thruster configuration shown in the
          y-z plane ............................................................................ 63
5.3 Three computational meshes used for Case 3 ................................. 67
5.4 Cell size ratios related to collisions and electric field mesh resolution .. 68
5.5 Number density contours on X-Z plane at y/D = 5 for Case 3 ............ 70
5.6 Velocity contours on X-Z plane at y/D = 5 for Case 3 .................... 71
5.7 Ion velocity contours on X-Z plane at y/D = 5 superimposed with stream-
          lines ................................................................................. 72
5.8 Plasma potential contours .......................................................... 73
5.9 Cross plume variation of ion properties ........................................ 74
5.10 Number density distribution along the centerline of the thruster ...... 76
5.11 Velocity(U) distribution along the centerline of the thruster ............ 77
5.12 Comparison of neutral density profile at the centerline for single and
          three thrusters ....................................................................... 79
5.13 Neutral number density contours on X-Z plane at y/D = 5 ............... 80
5.14 Velocity contours on X-Z plane at y/D = 5 superimposed with streamlines 81
5.15 Number density distribution along the centerline of the top thruster for three thruster configuration and single thruster .................................................. 83
5.16 Velocity(U) distribution along the centerline of the top thruster for three thruster configuration and single thruster .................................................. 84
6.1 Computational domain for a single ion thruster ........................................ 87
6.2 Computational domain for three ion thruster configuration on y-z plane ........ 87
6.3 V-Mesh, visualization mesh for a single thruster ....................................... 90
6.4 Neutral velocity contours in the X – Z plane at y/D = 5 superimposed with streamlines for a single thruster .................................................. 93
6.5 Ion velocity contours in the X – Z plane at y/D = 5 superimposed with streamlines for a single thruster .................................................. 94
6.6 Plasma potential, $E_P$, contours for the single thruster Case 3 conditions. 95
6.7 Upstream number density variation in the back flow region. .................... 97
6.8 Upstream velocity along the x-direction variation in the back flow region. 98
6.9 Ion energy distribution at the backflow plane for a single thruster configuration. ......................................................................................... 102
6.10 V-Mesh, visualization mesh for three thrusters ........................................ 103
6.11 Velocity contours on X – Z plane at y/D = 5 superimposed with streamlines for the three-thruster case .................................................. 105
6.12 Ion profiles for different the solar panel voltages and gas surface interaction models. ......................................................................................... 108
6.13 x-component of velocity contours for the downstream solar panel configuration at $y/D = 5$ plane. 109
6.14 Comparison of ion number density and velocity fields for upstream and downstream configurations at $y/D=5$ plane 110
7.1 Computational domains for single and triple ion thruster configurations 115
7.2 Values of AMR refinement level at $y/D=5$ for a ground chamber (a) versus space (b) 117
7.3 Neutral number density contour lines for the study of the effect of domain boundary conditions on the thruster plume at $y/D=5$ 123
7.4 Ion number density profiles for the study of the effect of domain boundary conditions on the thruster plume at $y/D=5$ 124
7.5 Neutral velocity contours for the study of the effect of domain boundary conditions on the thruster plume at $y/D=5$ 125
7.6 Ion velocity contours for the study of the effect of domain boundary conditions on the thruster plume at $y/D=5$ 126
7.7 CEX ion number density contour lines for the study of the effect of domain boundary conditions on the thruster plume at $y/D=5$ 127
7.8 CEX ion number density contour lines for $y/D=5$ using an overlay approach 128
7.9 CEX ion population comparison, top represents chamber and bottom space-vacuum conditions 129
7.10 Plume data 131
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.11</td>
<td>Neutral number density contour lines for the study of the effect of back-</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td>ground pressure on the chamber simulations at y/D=5</td>
<td></td>
</tr>
<tr>
<td>7.12</td>
<td>CEX ion number density contour lines for the study of the effect of</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>background pressure on the chamber simulations at y/D=5 using an overlay</td>
<td></td>
</tr>
<tr>
<td></td>
<td>approach</td>
<td></td>
</tr>
<tr>
<td>7.13</td>
<td>Neutral density line levels for the study of the effect of domain size on</td>
<td>137</td>
</tr>
<tr>
<td></td>
<td>the chamber simulations at y/D=5</td>
<td></td>
</tr>
<tr>
<td>7.14</td>
<td>Ion number density contours for the study of the effect of domain size on</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td>the chamber simulations at y/D=5</td>
<td></td>
</tr>
<tr>
<td>7.15</td>
<td>CEX ion density line levels for the study of the effect of domain size on</td>
<td>139</td>
</tr>
<tr>
<td></td>
<td>the chamber simulations at y/D=5 using an overlay approach</td>
<td></td>
</tr>
<tr>
<td>7.16</td>
<td>Partial specular simulation results for the study of the effect of wall model</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td>at y/D=5</td>
<td></td>
</tr>
<tr>
<td>7.17</td>
<td>CEX energy distribution comparisons for baseline and partially specular</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>case</td>
<td></td>
</tr>
<tr>
<td>7.18</td>
<td>Neutral number density profiles for triple thruster simulation at y/D=5</td>
<td>147</td>
</tr>
<tr>
<td>7.19</td>
<td>Ion number density profiles for triple thruster simulation at y/D=5</td>
<td>148</td>
</tr>
<tr>
<td>7.20</td>
<td>Velocity contours superimposed with streamlines for triple thruster sim-</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>ulation at y/D=5</td>
<td></td>
</tr>
<tr>
<td>A.1</td>
<td>Visualization with Paraview</td>
<td>164</td>
</tr>
<tr>
<td>A.2</td>
<td>Visualization with Tecplot</td>
<td>165</td>
</tr>
<tr>
<td>A.3</td>
<td>Speed-up</td>
<td>169</td>
</tr>
</tbody>
</table>
A.4 Module Interconnection Diagram . . . . . . . . . . . . . . . . . . . . . . 169
Nomenclature

Roman Symbols

$\Delta t_{\text{ref}}$ Reference time step for each species, s

$\Phi$ Plasma potential, V

$\Phi_0$ Reference plasma potential, V

$\sigma_T$ Total cross section, $m^2$

$\sigma_{\text{CEX}}^{\text{Xe}^-\text{Xe}^+}$ CEX cross section between neutrals and ions, $m^2$

$\sigma_{\text{MEX}}^{\text{Xe}^-\text{Xe}^+}$ MEX cross section between neutrals and ions, $m^2$

$\sigma_{\text{Xe}^-\text{Xe}^+}$ Collision cross section between neutrals, $m^2$

$\tau_i$ Ion species time step, s

$\tau_n$ Neutral species time step, s

$e$ Charge of an electron, $1.602 \times 10^{-19}$ C

$E_{\text{tot}}$ Total electric field, V/m

$E_{\text{BC}}$ Electric field due charged surfaces, V/m

$E_p$ Electric field due to plasma, V/m

$F_{\text{num}}$ The number of real particles each computational particle represents

$k$ Boltzmann constant, $1.381 \times 10^{-23}$ J/K
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n_0)</td>
<td>Reference electron number density, (m^{-3})</td>
</tr>
<tr>
<td>(N_{\text{max}})</td>
<td>Number of possible collisions in a cell per timestep</td>
</tr>
<tr>
<td>(n_e)</td>
<td>Number density of electrons, (m^{-3})</td>
</tr>
<tr>
<td>(n_i)</td>
<td>Number density of ions, (m^{-3})</td>
</tr>
<tr>
<td>(T_e)</td>
<td>Electron temperature, eV</td>
</tr>
<tr>
<td>(v_{\text{rel}})</td>
<td>Relative velocity, m/s</td>
</tr>
<tr>
<td>(V_c)</td>
<td>Volume of cell, m(^3)</td>
</tr>
<tr>
<td>(W_i)</td>
<td>Particle weight of ions</td>
</tr>
<tr>
<td>(W_n)</td>
<td>Particle weight of neutrals</td>
</tr>
<tr>
<td>Xe</td>
<td>Xenon</td>
</tr>
</tbody>
</table>

**Acronyms**

- **AMAR** | Adaptive Mesh and Algorithm Refinement
- **AMR** | Adaptive Mesh Refinement
- **CEX** | Charge Exchange
- **DSMC** | Direct Simulation Monte Carlo
- **FEEP** | Field Emission Electric Propulsion
- **HPC** | High Performance Computing
- **IFE-PIC** | Immersed Finite Element-Particle In Cell
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPL</td>
<td>Jet Propulsion Laboratory</td>
</tr>
<tr>
<td>LDT</td>
<td>Long Duration Test</td>
</tr>
<tr>
<td>MEX</td>
<td>Momentum Exchange</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>NEXT</td>
<td>NASA Evolutionary Xenon Thruster</td>
</tr>
<tr>
<td>NSTAR</td>
<td>NASA three-ring Solar Electric Propulsion Technology Application Readiness Program</td>
</tr>
<tr>
<td>PIC</td>
<td>Particle In Cell</td>
</tr>
<tr>
<td>SMILE</td>
<td>Statistical Modeling in Low-density Environment</td>
</tr>
<tr>
<td>SPARTA</td>
<td>Stochastic Parallel Rarefied gas Time accurate Analyzer</td>
</tr>
<tr>
<td>SUGAR</td>
<td>Scalable Unstructured Gas dynamics with Adaptive Mesh Refinement</td>
</tr>
<tr>
<td>UFS</td>
<td>Unified Flow Solver</td>
</tr>
<tr>
<td>VHS</td>
<td>Variable Hard Sphere</td>
</tr>
<tr>
<td>VTK</td>
<td>Visualization Tool Kit</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
Acknowledgments

I feel many sacrifices were made for the last four years to realize this dissertation, yet I also feel I have learned so much that I could only have imagined at the start. This would not be the case if I had not met the right people at the right time. First of all, I have to express my gratitude to my advisor, Dr. Levin, who has guided me thorough many obstacles along the way. I feel very fortunate for her patience and support and believe that her challenging yet insightful feedback helped me become a much more observant and knowledgable researcher. I am in debt to my teammates in our lab because of the intellectual support they have given me. My committee members have also been instrumental in giving their invaluable feedback to such a multi-disciplinary problem I have worked on. I also want to thank my friends Orcun and Ozgur for their brotherly support during my graduate studies.

I am very grateful for the support provided by the Air Force Office of Scientific Research (AFOSR) Grant No. FA9550-11-1-0158 for the duration of my graduate studies. I have also had the privilege to access and use the computational resource provided on ERDC Garnet, AFRL Spirit, and NCSA Bluewaters without which this work would be impossible.

A perhaps uncommon thanks go to Mr. Kemal Sunal who has passed away but still is one of the dearest people to any Turk on the planet. Some nights, I was only able to keep tackling problems, especially while developing and debugging SUGAR, with his movies playing in the background giving me focus.
I know most of the people think that they have the best parents and I respect that. Specifically, I believe every parent deserves a sincere gratitude who are supporting their children during their graduate studies. Nonetheless, this will not hold me back from saying that had I not have the best parents in the world, I would not be able go this far. Even though they have been half a world away most of the time, their love and support was what kept me going.
Chapter 1

Introduction

1.1 Electric Propulsion Devices

Satellite missions that require formation flying are often carried out with small satellites of sizes ranging from several centimeters to about one meter and masses of around 100 kg or less [1, 2]. While the electronic equipment on-board is sensitive to the ambient (radiation, magnetic field, etc.), the main bottleneck for the lifetime of these satellites, and therefore the entire mission, is highly dependent on the thruster systems. Due to their small size, the onboard power generation capacity is limited so that efficient electrostatic thrusters that can generate a few micro-Newton of thrust are preferred for orbit maintenance. Electric propulsion devices are advantageous compared to conventional chemical thrusters in terms of higher specific impulse [1, 2, 3]. One such type, ion thrusters [3], can supply low thrust with high efficiency and specific impulse and are being considered on a number of international and U.S. spacecraft [4, 5, 6].

Ion thrusters, which are a subset of electric propulsion devices, produce a collimated beam of ions that are accelerated electrostatically by a grid system to velocities typically on the order of 50,000 m/s. Moreover, due to their cost-saving advantages and their established performance, electrostatic thrusters have been used for several applications [4], such as for the use in interplanetary missions (e.g., NASA’s New Millennium Deep Space One and Dawn). In addition to their use on satellites, future spacecraft
are also proposed to employ a highly integrated chemical and electric propulsion system on board to provide thrust both from the chemical thruster electro propulsion (EP) thruster for a variety of operation mode (high thrust-low thrust) [7]. One of the recent ground-breaking efforts for a new generation ion thruster system is NASA’s Evolutionary Xenon Thruster (NEXT) [8, 9, 10, 11, 12] managed by NASA Glenn Research Center. These experimental studies mainly focused on the beam characteristics coming out of the thruster exit and their effect on the performance metrics such as thrust, specific impulse and endurance. Shastry et al. [11] discusses the results obtained from the Long-Duration Test (LDT) which was initiated in 2005. It was shown that while processing close to 800 kg of Xenon and being operated for 42,000 hours, various throttle levels were swept with a minor amount of degradation for the full NEXT system. Another recent performance evaluation was done on the T6 Ion Engine [13] which is baselined for the BepiColombo mission to Mercury where the results have shown that the engine was capable of producing more than 140 mN of thrust with a specific impulse of 4120 s at full power.

Due to collisions in the plume and the induced electric field, the propellant gas as well as the sputtered material causes contamination on highly sensitive flight instruments and solar panels for a range of propulsion devices. Predictive characterization of contamination requires high fidelity approaches for a realistic representation of the flow field surrounding the vehicle. The studies which focus on looking at merely the performance variables such as specific impulse or thrust cannot accomplish such goals in a straight-forward way. Although experiments are an essential and integral part of the field [14, 12], the conditions at which these experiments can be performed often fail
to simulate the actual space environment. In addition, extremely sensitive materials used in such tests increase the cost and effect reusability of such facilities. Despite the fact that the feasibility of ion thrusters for use in attitude control maneuvers has been verified, there are several active studies focused on the interactions between energetic charged particles and spacecraft surfaces causing contamination, erosion, and spacecraft charging [15, 4, 16]. Regarding backflow to the spacecraft surface, Wang et al. [16] has demonstrated that the charge exchange ions are the major species and the characteristics of the interaction between the thruster plume and spacecraft is governed by the charge exchange plasma. An accurate representation and analysis of the plasma environment surrounding the spacecraft requires in-flight data collection as well as simulation results [16].

1.2 Numerical Modeling of Electric Propulsion Devices

A number of investigators have modeled the plume of ion thrusters and its interaction with spacecraft surfaces. Roy et al. [15], being one of the pioneer works on the large scale modeling of ion thruster plume and contamination, analyzed the flow-field and looked at the effect of charge exchange collisions for axisymmetric and three dimensional configurations. A study based on a multiple ion thruster configuration used on the MUSES-C spacecraft also analyzed the backflow effects when all thrusters are used simultaneously [17]. Hyakutake et al. [17] did not model the momentum and energy coupling between ions and neutrals (i.e. two way coupling) emitted by the thruster exits but used an overlay approach where the neutral flow was simulated prior to the ion flow. Another study [18] showed that for an interplanetary spacecraft with a moderate
charging potential, charge exchange ion backflow occurs through an expansion process similar to that of the expansion of a mesothermal plasma into a vacuum. Studies have also shown that material sputtered from the inside of the ion thruster can also cause contamination [19, 20]. Material sputtering can occur when the highly energetic ions react with the grids that are used to accelerate the charged particles across an applied electric field. This sputtered material, such as molybdenum (Mo), can travel along the electric field and cause contamination. In addition to ion thrusters, backflow from other types of electric propulsion devices such as Hall Thrusters [21], Field Emission Electric Propulsion (FEEP) [22] thrusters, and colloid thrusters [23] have also been studied.

Experiments are an integral part of the design, prototype, and testing stage of these thrusters. Performance characterization and endurance tests are conducted in ground-based discharge and vacuum chambers [24, 11]. Although these experiments have contributed significantly to the design methodologies, limitations are present in terms of the testing conditions or configurations that are desired. The first of these limitations is the finite background pressures that cannot be avoided due to the limited pumping capabilities. Although having more and more powerful pumping capabilities helps decrease the background pressures to microTorr levels [25, 26, 27], there are still issues regarding the lifetime of the thrusters that are tested in these ground tests [28] (wall erosion, etc.) as well as the changes in the plume structure [29]. Numerical simulations offer excellent advantages in terms of flexibility of operating conditions, cost, and time. Various studies have been conducted to look at the fundamental physical phenomena occurring in the plume as well as its interaction with the satellites or spacecraft [29, 19,
21, 20, 30]. The results from these simulations took background pressures into account to enable better comparisons with ground-tests.

Van Gilder [29] conducted numerical simulations to compare with plasma measured at different plume locations during UK-10 thruster tests [25]. Simulations focusing on the plasma inside the thruster (discharge chamber) were also conducted in the recent years [31, 32, 28]. Sudhakar et al. [31] used a particle-based model to simulate the plasma inside the discharge chamber of an ion engine while tracking five major species types (neutrals; singly and doubly charged ions; and primary and secondary electrons). The electric and magnetic field calculations were carried out and the charged particles were moved under the effect of these forces. Simulations have shown good agreement with the experimental findings [24] of ion beam and discharge current from NASA’s three-ring Solar Electric Propulsion Technology Application Readiness Program (NSTAR). Jian et al. [32] studied the effect of the background pressure on the characteristics of ion impingement on the accelerator grid by three-dimensional numerical simulations. The simulations using immersed-finite-element particle-in-cell (IFE-PIC) and direct simulation Monte Carlo (DSMC) showed that the accelerator grid impingement current increases with the background pressure and suggested that the background pressure in the vacuum chamber should be less than 1 mPa for reliable lifetime prediction in ground tests.

In addition to ion thrusters, studies on the discharge chamber wall erosion of Hall thrusters were conducted. Hofer et al. [28] developed a chamber erosion sub-model at the Jet Propulsion Laboratory (JPL) that was integrated into the computer code HPHall-2. The simulations found that improvement of the velocity boundary condition at the injector resulted in better agreement with the experiments. Modeling and
simulation of plumes inside a vacuum chamber has also been conducted using kinetic approaches [33, 34]. Bondar et al. [33] looked at plume-spacecraft interactions for a Hall thruster by characterizing the neutral and ion species spatial distribution for vacuum and chamber configurations. Their chamber solution showed that the density of the beam ions dramatically decreases along the plume axis due to the collisions with background particles.

The plasma medium which consists of ions, charged droplets, neutral atoms, and electrons is not amenable to theoretical, analytic studies and is sufficiently rarefied to cause the application of fluid models to be questionable. Therefore a proper analysis of the gas dynamics and the plasma essentially demands a kinetic consideration. The fidelity of the physical models used in the simulations is also important to ensure realistic modeling of momentum exchange (MEX) and charge exchange (CEX) collisions [35]. In addition to this, the multi-scale nature of the problem caused by the expansion of the gas to a vacuum and the presence of both neutral and ion species necessitates the use of computational techniques that can efficiently resolve the flow field physics. In this work, the two well known numerical approaches known as the direct simulation Monte Carlo (DSMC) method used to model collisions between neutrals and neutrals and ions and the particle-in-cell (PIC) method for modeling charged particle motion under an electric field, will be used to the disparate time and length scales associated with ion thruster plume flows.

To address the limitations of the aforementioned examples, which do not even include the additional challenges of simultaneously modeling neutral and charged species rarefied flows, the use of an AMR-Octree approach and demonstration of its superior
capabilities for this class of problems is presented. To date, no work has been done on modeling electric propulsion device plumes in a three dimensional, adaptive mesh refinement framework with a rigorous modeling of CEX of neutral and ion species and an electric field environment. The objective of this work is to characterize and quantify the structure of ion thruster plumes for single and multiple thruster configurations making use of the novel computational AMR-Octree approach within a parallel framework.

1.3 Importance of high performance computing

As mentioned previously, conducting studies through numerical simulations are becoming more and more popular among scientific communities. The use of supercomputers through tailored programming languages makes it much more user friendly compared to two decades ago. Looking at the fastest supercomputers globally, it can be observed that programming languages that exploit distributed memory (Message Passing Interface, MPI [36]), multi-threads (OpenMP [37]) and attached accelerators (GPUs [38], OpenACC [39]) are used actively to achieve computing speeds that help reduce simulation times for computationally intensive simulations. Other programming languages such as Coarray-Fortran [40], UPC [41], Chapel [42], CilkPlus [43] are under development to serve as hybrid approaches for the previous languages which aim to serve as higher level languages. It is obvious that such computing capabilities granted by High Performance Computing (HPC) enable researchers to further the knowledge in their branches which would otherwise be impossible.
1.4 Dissertation Structure

The remainder of the dissertation is organized as follows. In Chapter 2, implementation of multiple AMR/octree grids in a new DSMC code, refinement criteria, and the parallelization strategy and preliminary estimates of performance is discussed with the description of the general framework of the developed code. Chapter 3 presents the physical and mathematical models used for an ion thruster application and the implementation of multiple AMR meshes for modeling collisions and electric field. Following this, a comparison of SUGAR simulations with the two-level SMILE DSMC code [44] is presented for a neutral plume expansion and compression cases in Chapter 4. The results for the ion thruster plume modeling is given Chapter 5 where the focus is given on understanding the various physical mechanisms. As a continuation, In Chapter 6, the backflow phenomena is analyzed. Then, results for the plume inside a test chamber is given in Chapter 7 which presents the differences between a space and chamber application of ion thrusters. Finally, Chapter 8 presents the conclusions for the study and discusses some future research paths.
Chapter 2

Development of a Scalable Gas Dynamics Software

The software developed and used to run the simulations presented in this dissertation is known as the Scalable Unstructured Gas dynamics with Adaptive Mesh Refinement (SUGAR) code. SUGAR uses the Message Passing Interface (MPI) to harness the computational power of modern computing architectures for parallel computing and is developed in C++ to make use of object oriented programming (OOP) paradigms, which minimize the effort needed to develop, maintain, and extend the code capabilities for large-scale efforts. The framework has also been designed to be capable of future integration with multi-threading and accelerator paradigms.

2.1 Adaptive Mesh Refinement

The grid structure used in the present work is Adaptive Mesh Refinement (AMR), which has unique capabilities for obtaining efficient solutions for problems that have multi-scale properties and large gradients in their computational domain. Following the pioneering work of Berger and Colella [45], its implementation is being widely used in many diverse scientific branches from astrophysics [46] to biology [47]. Recently, it has also been implemented in similar DSMC work [48].

One of the drawbacks for using an unstructured computational mesh, which Octree AMR is a subset of, is the necessity to store the connectivity information between
neighboring cells. Structured meshes are able to store connectivity information implicitly by the coordinate indices (i.e., $i,j,k$). To obtain the same precision in the computational domain, the number of cells needed by structured meshes exceeds the number for unstructured meshes by many orders of magnitude, depending on the problem. Tetrahedral meshes accomplish this by explicit storage of node and cell data. The motivation of using the octree storage for cell connectivity lies in its efficient data storage. Using unstructured tetrahedral meshes has advantages and as well as disadvantages. The main advantage is that the domain can be discretized in a flexible way, especially capturing solid boundaries with a body-fitting approach. However, forming such a complex mesh requires human intervention to guarantee that the resultant mesh is free of issues. This requires a considerable amount of time even with recently developed smart software that aids the user in meeting typical rules of thumb that can be used as metrics for high quality meshes.

The motivation of using the octree storage in hexahedral (i.e. polyhedron with six faces) meshes for cell connectivity lies in its efficient data storage as well as straightforward adaptive refinement capabilities. In tree structures, of which octrees are a subset, each node has $2^d$ children where $d$ represents the number of dimensions present in the problem and for this reason binary, quad and octree algorithms are analogous to each other. For three dimensional cases, a node is divided into eight children nodes. Describing the terminology briefly, every node that has children is called a parent. It is apparent that in this tree structure, a node can simultaneously be a parent (a node without a parent is a root) and a child (a node without any children is a leaf). A schematic description is summarized in Fig. 2.1. It can be seen that a parent and its children occupy the
same space in the computational domain, so flags are also introduced for each node to distinguish if it is active or deactive. This guarantees that the computational domain is taken up by unique nodes that only share boundaries or surfaces.

![Computational Mesh and Quadtree Structure](image)

Fig. 2.1. Schematic description of tree structure

### 2.2 Parallelization

SUGAR was designed and developed to be run on modern supercomputers that have distributed memory architecture. In this section, the underlying modules that were developed regarding the domain decomposition, communication, and load-balancing are discussed.
2.2.1 Cartesian Mesh

The parallelization is achieved by using domain decomposition for the computational domain and using MPI for communication between the processors. Each simulation starts by creating a Cartesian mesh of the computational domain for which the user supplies the number of cells in each direction. Inside each of these Cartesian cells, an AMR capability is initiated. The three AMR meshes start from the same root cell; however, since they follow different subdivision criteria, the resultant meshes are different. Each processor stores the underlying Cartesian mesh as a whole, but only stores the portion of the three grids that falls into their region according to the domain decomposition.

2.2.2 Communication via MPI

For the modeling of an ion engine plume, computational particles are released from the thruster exit into the computational domain. At each time step, computational particles are mapped to each of the three computational meshes in order to carry out the computations required to model collisions, field effects in the flow, and calculate macroparameters. Once this mapping stage is finished, computational particles are moved with their local species-dependent timestep, as will be explained in the Section 3. After moving these computational particles, each one is checked to see if it is still on the same processor, has exited the computational domain, or has been transferred to another processor. For the computational particles that are going to be transferred to another processor, each processor creates linked lists to store the corresponding particle lists that need to be transferred. Once each processor has information regarding which
processor to which it must send and from which it must receive computational particles, communication is accomplished in a point-to-point fashion via MPI. In terms of parallel performance to date, Fig. 2.2 shows a basic speed-up plot on a SGI Ice X platform using Intel E5 Sandy Bridge processors with a core speed of 2.6 GHz for the conditions of Case 3 which will be discussed in Chapter 5.

2.3 Domain Decomposition

2.3.1 2D Regular Blocking

Two dimensional blocking with rectangular prisms are used for partitioning the underlying Cartesian mesh, i.e. coarse-grained parallelism is used. The domain is split $N_x, N_y, N_z$ times in each respective direction where $N_x \times N_y \times N_z$ equals to the total number of MPI ranks that are used. This approach is essentially a guided recursive coordinate bisectioning (RCB) method. Although domain decomposition is done manually (i.e. no cell or particle information is used), the knowledge of spatial number density distribution is used to have a decent approximation for partitioning (each block having a similar number of computational particles). A typical domain decomposition with this approach can be seen in Fig. 2.3. The colors represent the identity of each MPI Rank on one of the three meshes to be described in Chapter 3 where the cluster of small V-mesh cells belong to the thruster exit. The resultant speed-up figure for a strong scaling study can be seen Fig. 2.2, in which the solid black line represents a simulation which used approximately 10,000,000 computational particles. Up to 64 MPI ranks, an almost perfect scalability is observed which implies that processors are almost fully loaded. However,
after this point, the speed-up values start to deviate from the ideal curve which results in speed-up values around 400 for 1024 MPI ranks and processors being loaded by an average of 40%.

2.3.2 Integration with a Graph Partitioner

To improve the performance that was achieved by the 2-D blocking algorithm, potential remedies to increase the speed-up values were investigated. Two noteworthy examples from the literature are briefly discussed that have similar features and capabilities to SUGAR which uses fully or partially kinetic approaches. The first effort that will be discussed is the Stochastic Parallel Rarefied gas Time accurate Analyzer (SPARTA) [49]. It is a parallel DSMC code which has 2D and 3D simulation capabilities and was developed in C++ and MPI. Physical objects with triangulated surfaces can be embedded in the computational domain. It uses RCB to decompose the domain among the processors. The collision cells are assigned weights proportional to the number of computational particles they own. It has been stated that RCB is fast and that the major computational cost is the rebalancing for moving and rebuilding data [49]. The second approach was the Unified Flow Solver (UFS) [50] by CFD Research Corporation. UFS employs an Adaptive Mesh and Algorithm Refinement (AMAR) methodology which sits upon the framework of Gerris [51], an open source computing environment for solving partial differential equations with AMR. A major extension was done to Gerris framework so that UFS is able to simulate rarefied, translational and continuum flows by solving the Boltzmann and Navier-Stokes equations.
Fig. 2.2. Speed-up plot for SUGAR with load-balancing study.
Fig. 2.3. Processor topology with 2-D blocking algorithm
Focusing on SUGAR again, the proposed two dimensional blocking algorithm did not pay any attention to possible uneven computational loads among the processors. This kind of an uneven load distribution is especially apparent for multi-scale problems. Looking at Fig. 2.2, although slight deficiencies are apparent after 64 MPI ranks, a rather good speed-up value is observed up until 1024 for the 1 thruster case using 2-D domain decomposition represented by the "SUGAR, 1 Thruster, 2-D Block" curve. Beyond this value, the problem size has been increased to four times the baseline case (i.e. a simulation using approximately 40,000,000 computational particles). It has been observed that as the problem size is increased, the speed-up value at 1024 MPI ranks increased by almost 50% for 4 thruster case using 2-D domain decomposition represented by the "SUGAR, 4 Thruster, 2-D Block" curve. A slight increase in the speed-up value is increased at 2048 compared to 1024 but for 4096 and 8192 MPI ranks, a slow-down is observed.

The strategy to improve the speed-up further was to use the underlying Cartesian mesh to supply granularity for domain decomposition. The first approach was to use a graph partitioner called Parmetis [52]. The initial step was to form a graph that represents the cells of the Cartesian Mesh. Weights which represent the total number of computational particles are assigned to each of these cells. After this assignment, processors make a collective function call to Parmetis. The solution generated by Parmetis is expected to propose a new processor topology that accounts for the load imbalance for highly and lowly weighted nodes. Following this output from Parmetis which necessitates changing the ownership of the Cartesian cells, SUGAR calls a native function
that reshuffles all the cells and computational particles between each processor. A typical processor topology can be seen Fig. 2.4 when Parmetis is used at the steady state of the simulation. The initial effort was to calibrate the input parameters to increase the speedup value for 1024 processor case. Several set of inputs were tried by changing the parameters to the function calls some of which caused Parmetis to find no solution and crash. The solution point which resulted in a greater speed-up compared to the 2-D blocking approach was achieved for 1024 MPI ranks with the default options to the function "ParMETIS_V3_PartKway" with "itr" and "ubvec" parameters equal to 1,000,000 and 1.05, respectively and a lower threshold of 50 particles for a unit amount of weight. The results which also include single thruster simulations using less than 1024 MPI ranks are shown in Fig. 2.2 represented by "SUGAR, 1 Thruster, Parmetis" curve. It can be observed that although the resultant speedup values are approximately half of what was achieved by 2-D blocking algorithm for number of MPI ranks less than 1024, at 1024 MPI ranks, a value twice of the 2-D blocking result was observed. This shows that when a certain input parameter set is supplied, a processor topology can be obtained for a certain number of MPI ranks which can surpass the speedup value achieved by the 2-D blocking algorithm. However, this results in inferior speedup values for smaller number of MPI ranks which implies that there is no automation for any problem size or number of MPI ranks with this approach. It should also be stated that when this approach was used for a four thruster case using 2048 MPI ranks represented by "SUGAR, 4 Thruster, Parmetis" curve, a speedup value roughly half of 2-D blocking result was obtained.

Since it was found that Parmetis could be prone to errors and result in inferior speed-up values (at least for our problem in hand) for the simulations that were run, a
patched based approach was adopted, as shown in Fig. 2.5. This approach was pseudo-manual in the sense that the partitioning is done by using half of the total MPI ranks distributed as a 2-D blocking approach, but the remaining half is assigned to the location where the computational particle and AMR cell density is higher (near the thruster exit). Looking at Fig. 2.2, using this patched based approach represented by "SUGAR, 4 Thruster, Patched" curve, a slight increase in the speed-up is observed (7%) for 2048 MPI ranks. However, the patched based approach is 40% slower than the 2-D blocking approach at 4096 MPI ranks. Both of these results shows that using this patched based approach, the improved load balancing in terms of the computations is not enough to overcome the overhead brought by the change in the communication routines for the resultant processor topologies.

To conclude, it should be stated that the initial approach using 2-D blocking algorithm is decent enough to give good scalability until 2,048 MPI ranks when the processors are loaded by 20-40%, on average. All the other studies with more complex domain decomposition approaches imply that the communication routines in SUGAR should be revisited by better or alternative use of graph partitioners, code profiling tools and modified while still prioritizing the particle nature of the numerical methods being used. In other words, using kinetic approaches dictates using computational particles where these particles are moved in the computational domain based on their location, velocity and acceleration. This necessitates that these computational particles be able to travel to any processor domain which was accomplished by developing generic communication routines. As a final remark, for the range of simulations that have been performed
during this work, the parallelization strategies used allowed us to bring the simulation
times down from a few months to several hours.

2.4 Supplemental Libraries

2.4.1 Python

Python [53] is a high-level programming language which has started to be widely
used in scientific communities since the last decade. In SUGAR framework, a few python
scripts has been developed and used for pre- and post-processing purposes. The pre-
processing script called "do.py" is responsible for creating the executable file. Post-
processing script called "post.py" is responsible for merging individual Tecplot files into
a single one. Both of these files are supplied in the Appendix.

2.4.2 Pugixml

For a hierarchical view and user readability purposes, input file for SUGAR was
selected to be of XML (eXtensible Markup Language) [54] type. Three XML files are
used in a SUGAR simulation which are read at the start to initialize the simulation
parameters supplied by the user. These files are "main.xml", "dsmc.xml" and "pic.xml"
which store computational domain, DSMC and PIC related parameters, respectively.

The open-source parsing library Pugixml [55] has been used to read these input
values from SUGAR. Within the parallel framework, Processor 0 reads in the input data
on behalf of all other Processors. After all the input files are read, Processor 0 broadcasts
all the input data by using "MPI_Broadcast" command.
Fig. 2.4. Processor topology with Parmmetis implementation for the single thruster case

(a) x-z plane at y/D=5

(b) x-y plane at z/D=5
Fig. 2.5. Processor topology with patched based approach for the four-thruster case
2.4.3 VTK

A special capability to output computational results in a state of the art fashion using Visualization Tool Kit (VTK) [56] was developed. I/O can potentially have a considerable impact on scalability if it is done in serial as opposed to being in parallel. A single processor might need to receive the output data on behalf of the rest where a single stream of data writing produces a bottleneck in the performance. Instead, each processor in SUGAR writes out its own portion of the results in VTK form and merging of these results from all processors is carried out during the post-processing step. The additional advantage is also the fact that binary output is used by VTK which decreases the storage requirements as well as transfer time between platforms.

This is illustrated by the following example. A trivial MPI program is written which works with 4 MPI ranks. Each processor creates a separate VTK file and signs it by appending its MPI rank ID to the filename (e.g. Example_3.vtu for third processor). Figure 2.6 shows the individual screenshots of these data files once opened separately. Moreover, Processor 0 (an arbitrary selection) assumes the responsibility to write out a ”.pvtu” file which is needed to merge all the individual files in-situ while visualizing. In other words, since a ”.pvtu” file has the number of total MPI ranks for a specific simulation and the corresponding names, the visualization software (e.g. Paraview, Visit) can open all the files simultaneously. Figure 2.7 shows the screenshot when a ”.pvtu” file was used and the simultaneous representation of the data field. This is noteworthy as this saves the user considerable amount of time by practically skipping a post-processing stage for each run (approximately several minutes).
Fig. 2.6. Individual representation of data cells
Fig. 2.7. Simultaneous representation of data fields
2.4.4 Tecplot

As an alternative output format, Tecplot [57] data files are also supported by SUGAR. Following the data format description in Ref. [57], individual files from each processor are outputted. In order to post process these individual files (i.e., create the merged data file to be read by Tecplot), a Python script is developed. Although VTK files are used and visualized by Paraview/Visit, the majority of the time (during development/debugging process), Tecplot data files were used and visualized by Tecplot in order to produce publication quality results (since it has better support for axis labels, streamlines, etc.)
Chapter 3

Development of Modules for an Ion Thruster Application

3.1 Numerical Techniques

3.1.1 DSMC

Direct Simulation Monte Carlo (DSMC) is a discrete particle simulation method that provides a numerical approximation to the solution of the Boltzmann equation if the simulation numerical parameters are correctly selected [58]. The underlying principle of the DSMC method is the splitting of the continuous motion and collisions of simulated particles, each representing a large number of true atoms or molecules. During a simulation time step there are two sequential stages: free molecular motion and collisions. Due to the uncoupling of free molecular motion and molecular collisions, the time step used in simulations must be less than the mean collision time. Simulated particles move, reflect from the walls or surfaces, and collide with each other according to prescribed interaction laws. Particles are tracked through the computational domain, indexed, and sorted/grouped into cells according to their locations. Probabilistic techniques are used to model collisions and the colliding particles are assigned new velocities and internal energies, so that the energy and momentum are conserved. Boundary conditions are modeled through collisions of the computational particles with the surfaces by suitably
choosing a gas surface interaction parameter. At steady state, macroparameters are fin-
ally computed by the summation over all the particles contained in each cell [58]. Since
the computational cost scales as the cube of the number of computational particles, it
is crucial to implement grid schemes that strategically allocate particles in the required
regions of the flow.

Many DSMC calculations are performed on structured, two-level Cartesian grids
with cell size adaption based on the local mean free path [44]. More recently, Arslanbekov
et al. [48] employed Octree based adaptive mesh refinement similar to the present work
to simulate two dimensional hypersonic flow over a cylinder and a wedge and a three
dimensional flow over the Stardust body. With Octree implementation, the authors
found that the transition from the neighboring cells resulted in much smoother profiles
while maintaining the required number of computational particles at each location in
the domain. Kolobov [59] has also employed the AMR technique for modeling low-
temperature plasmas using fluid and kinetic approaches where immersed solid bodies
were present. Olson and Christlieb [60] developed a gridless octree method for scaling
particles into local clusters so as to model flows with complex geometries present in
the domain. Benchmark simulations of low speed flow over a plate and hypersonic
flow over embedded geometries showed that the approach is indeed superior to fixed grid
approaches. In Nompelis and Schwartzentruber [61], adaptive mesh refinement strategies
were applied to local Cartesian mesh patches created in the computational mesh. This
approach, however, causes problems with controlling the location and size of the patches
and has the drawback of over-refining the grid in order to meet a refinement criteria.
while causing the number of computational particles per cell to drop to values so low as to cause statistically inaccurate results.

3.1.2 PIC

The Particle-in-Cell (PIC) technique [62] is essentially a particle-mesh approach for tracing charged particle motions under applied as well as induced electric and magnetic fields. In this method, ions are treated as macro-particles, as in DSMC, with each macro-particle representing many physical particles, typically on the order of $W_i = 10^9$. The particles’ equations of motion are integrated under the influence of the computed self-consistent electromagnetic fields. For the problem of interest, self induced magnetic fields are considered negligible, and hence we will only consider the electrostatic potential governed by Poisson’s equation.

Due to the ions present in the flow, an electric field is induced and has to be taken into account for the movement of ion species, as was done in previous work [17, 29]. The force due to the field is introduced as an acceleration of the charged particles which is used to update the velocity of the charged particles. This updated velocity is used to move the computational particles during the "move" stage of the DSMC simulation.

The ion thrusters are equipped with an electron gun near the ion thruster exit. This gun emits electrons in order to neutralize the ions in the plume to minimize the backflow of these ions for potential contamination of spacecraft surfaces. Assuming the electrons emitted from the neutralizer are governed by the Boltzmann relation, we have

$$
\Phi = \Phi_o + \frac{kT_e}{e} \ln \left( \frac{n_e}{n_o} \right)
$$

(3.1)
where \( \Phi \) is the plasma potential, \( \Phi_0 \) is the plasma potential at a reference condition, \( k \) is Boltzmann’s constant, \( T_e \) is the electron temperature, \( e \) is the elemental charge, \( n_e \) is the electron number density, and \( n_o \) is the electron number density at the reference condition. Furthermore, if the plume is assumed to be quasi-neutral, it is possible to determine the plasma potential from the ion density \( (n_e \approx n_i) \). The plasma potential at the reference condition, \( \Phi_0 \), is assumed to be 21 V at the thruster exit and \( T_e = 3 \) eV [17]. Hyakutake [17] showed that as the electron temperature is varied between 1 and 10 eV, the electron number density in the core region remains essentially the same. Once the potential is known the electric field can be calculated by

\[
E = -\nabla \Phi
\]  

(3.2)

where \( E \) is the electric field vector and \( \Phi \) is the plasma potential field. In practice, Eq. 3.2 is discretized and the electric field is solved in the domain.

For regular Cartesian meshes, this is a well-established approach, yet in our case, the stencil for discretization must follow the octree mesh structure. In this work, first-order accurate differencing schemes (forward and backward respectively) given by

\[
\frac{\partial \Phi}{\partial x_s} = \frac{\Phi_{s+1} - \Phi_s}{\Delta x_s}
\]

(3.3)

\[
\frac{\partial \Phi}{\partial x_s} = \frac{\Phi_s - \Phi_{s-1}}{\Delta x_s}
\]

(3.4)

are employed in each cell of the E-Mesh which is created to model the induced electric field, where \( s \) is the direction along the gradient operator (i.e., \( s = i, j, k \)). Given
the potential, the electric field is calculated by using a cell-centered finite differencing scheme. Figure 3.1a shows the relationship of cells in terms of their levels. If all the children cells of a parent cell are active, i.e., none of the cells have any other cells occupying their space, as shown in Fig 3.1b, Scenario 1, the children cells use Eqs. 3.3 and 3.4 accordingly. If not, that is if any of the children cells have any children of their own as shown in Fig 3.1c, Scenario 2, a cell centered averaging among the four second level children cells is performed and a value is assigned to the inactive child cell so that Eqs. 3.3 and 3.4 can be used.

Finally, it should be noted that since the distribution of particles inside a cell is made almost uniform by the AMR approach, particle weighting on the nodes was not necessary in contrast to common Cartesian approaches. Therefore, it was adequate to
store the data points at the cell center for each cell where the density was calculated by using $F_{NUM}$, particle count, and the volume of the cell.

One of the major focuses of this work is to develop an approach to account for the electric field both from an external source, such as a solar cell, as well as the induced electric field from the plasma. To achieve this we decompose the total electric field, $E_{tot}$, into the contributions from the charged particles, $\Phi_p$, and the imposed external field, $\Phi_{BC}$, so that the total electric field can be expressed as

$$E_{tot} = -\nabla(\Phi_p + \Phi_{BC}) \quad (3.5)$$
$$E_{tot} = E_p + E_{BC} \quad (3.6)$$

where $E_{BC}$ corresponds to the electric field vector generated by the solar panels. Each contribution to the total electrical field is taken into account separately and weighted on the ions consecutively at each timestep. $E_{BC}$, however, is calculated only once, in contrast to $E_p$, by weighting its nodal values on the location of the ions at each time step. The weighting scheme on the computational particles is described in detail in Birdsall [63] and was adopted in this work.

The solution for $E_{BC}$ is obtained using the commercial software ”ANSYS Workbench” on a uniform Cartesian Mesh [64] where the computational set-up is shown in Fig. 3.2. Consistent with the surface thruster outflow boundary condition, the thruster surfaces do not contribute to the electrical boundary conditions. That is, only the potential on the solar panels are considered as these are the part of the satellite that is
differentially charged and assumed to contribute to the resultant total electric field or influence the general backflow plume structure. Note that Van Gilder. [29] and Hyakutake et al. [17] did not model the plasma sheath for backflow studies as their main focus was also the overall backflow features. In future work, this assumption will be revisited and the plasma sheath formation will be considered. The voltage on the solar panel is taken to be $-3 \text{ V}$ based on the approximation given in Wang et al. [18] where it is assumed that the front face is an insulator with a surface voltage of $-T_e$ or the electron temperature surrounding the solar array/spacecraft. To obtain the ANSYS solution, the steady-state Electric Conduction module was used with the sparse matrix solver on 64,000 cells on $41^3$ grid points. Two solar panels extending radially and having a width and length of 0.1 and 0.4 m, respectively, are located at the $y-z$ plane ($x = 0 \text{ m}$). As mentioned above, ion impact on the solar panels were assumed to result in neutralization and the neutral species were reflected with full accommodation at a solar panel wall temperature at 300 K.

The solution of the electric potential due to the solar cell arrays obtained from ANSYS can be seen in Fig. 3.2c where the $-3 \text{ V}$ potential difference of the solar cell panels relative to the free stream is clear. A grid convergence study was also done to ensure that the resulting solution for $\Phi_{BC}$ was grid converged. Figure 3.3 shows the $x$-component of $E_{BC}$, the dominant component, extracted from a line which starts from $x, y, z = 0.0, 0.5, 1.0 \text{ m}$ and ends at $1.0, 0.5, 1.0 \text{ m}$. Three cases which used $21^3$, $41^3$ and $81^3$ grid points were compared and it can be seen that although the solution obtained with the $21^3$ grid is in good agreement with the other two solutions for $x \geq 0.2 \text{ m}$, the solution near the top solar panel ($x, y, z = 0.1, 0.5, 1.0 \text{ m}$) is poor. It can also
be seen that the $41^3$ and $81^3$ solutions are essentially the same so that the $41^3$ grid is adequate. The electric field solution for $E_{BC}$ obtained on a Cartesian grid was extracted from ANSYS and pre-processed once to be used by SUGAR prior to start of the main simulation. This means that for the simulations which makes use of the ANSYS solution for the solar cell panels, there are three AMR/Octree Meshes (C-, E- and V-Meshes) created by SUGAR and one Cartesian Mesh supplied by ANSYS, making a total of four being used simultaneously. At each time step, the two electric field solutions ($E_p$ on E-Mesh and $E_{BC}$ on the Cartesian Mesh by ANSYS) are used consecutively by simulating the individual effects on the ions to generate the composite electrical force. This also combines the advanced capability of a research code such as SUGAR with a commercial computational tool available for modeling operational space conditions.

3.2 Physical Models

The DSMC-PIC simulations have both neutral and ion species that expand from the exit of the electric propulsion thruster. The three collision processes that are modeled among neutrals and ions are:

$$Xe(p_1) + Xe(p_2) \rightarrow Xe(p'_1) + Xe(p'_2) \quad (3.7)$$

$$Xe^+(p_1) + Xe(p_2) \rightarrow Xe^+(p'_1) + Xe(p'_2) \quad (3.8)$$

$$Xe^+_{fast}(p_1) + Xe_{slow}(p_2) \rightarrow Xe^+_{slow}(p_2) + Xe_{fast}(p_1) \quad (3.9)$$
Fig. 3.2. Electrical boundary conditions and geometry.
where \( p_1 \) and \( p_2 \) are the pre-collisional momentum and \( p'_1 \) and \( p'_2 \) are the post-collisional momentum. The processes represent momentum exchange (MEX) among neutral collisions, MEX among neutral and ions pairs, and charge exchange collisions (CEX) between ions and neutrals, respectively. Note that CEX is a major mechanisms for depositing energy into neutral species from ions. The cross sections for MEX between neutrals\[65]\] and MEX[35] and CEX[66] between neutrals and ions are given by,

\[
\sigma_{\text{Xe–Xe}} = \frac{2.117 \times 10^{-18}}{v_{rel}^{0.24}} \text{m}^2 \quad (3.10)
\]

\[
\sigma_{\text{Xe–Xe}^+}^{\text{MEX}} = (213.04 - 30.94 \times \log_{10}(E)) \times 10^{-20} \text{m}^2 \quad (3.11)
\]

\[
\sigma_{\text{Xe–Xe}^+}^{\text{CEX}} = (87.3 - 13.6 \times \log_{10}(E)) \times 10^{-20} \text{m}^2 \quad (3.12)
\]

where \( v_{rel} \) represents the relative velocity between the selected collision pair and \( E \) represents the energy of the ion in the laboratory frame in eV.

Fig. 3.3. Convergence study for \( E_{BC} \) at \( y = 0.5 \text{ m} \) and \( z = 1 \text{ m} \) of Fig. 3.2.
For typical operating conditions, there is a $O(2)$ magnitude difference between the ion and neutral species in their velocities and densities. This phenomenon would create an inefficient particle movement algorithm if a single timestep were employed for both species since in order to resolve the smallest length scale, a timestep corresponding to the fastest species, ions, would have to be used. To overcome this inefficiency, a modified movement and collision scheme was suggested in Ref. [67]. For two species, having disparate length and density scales, the number of possible collisions, $N_{\text{max}}$, in a cell of volume ($V_c$) during timestep $\Delta t$ is given by

$$N_{\text{max}} = N_i \times N_n \times \text{max}(W_i, W_n) \times (g\sigma_T)_{\text{max}} \times F_{\text{num}} \times \Delta t/V_c$$  \hspace{1cm} (3.13)

where $N_i$ and $N_n$ are the number of computational ion and neutral particles, $W_i$ and $W_n$ are the weighting factors for ions and neutrals, $(g\sigma_T)_{\text{max}}$ is the maximum product of relative velocity between the particles and the total cross-section of each collision type and $F_{\text{num}}$ is the value representing the number of real neutral particles represented by a single computational particle. Weighting factors are employed in order to realistically simulate the interaction between major and trace species. The collision scheme stated in Eq. 7.1 can be used for collisions between neutral-neutral and neutral-ions with small modifications such as changing the product $N_iN_n$ to $\frac{1}{2}N_n(N_n - 1)$ for neutral-neutral collisions where the factor of one-half avoids double counting neutral-neutral pairs.

In each time step, once $N_{\text{max}}$ is calculated, the following steps are repeated $N_{\text{max}}$ times, in each cell.

1. A pair of particles $A^s_i$ and $A^r_n$ ($s = 1, ..., N_i; r = 1, ..., N_n$) are selected at random.
2. Their relative velocity $g_{in}^{sr} = |\mathbf{c}_i^s - \mathbf{c}_n^r|$ is calculated, where $\mathbf{c}$ is the particle’s velocity vector. A random number $R$ (i.e. $0 < R < 1$) is called. If $R > (g_{in}^{sr} \sigma_T / (g\sigma_T)_{max})$, then no collision occurs and one returns to Step 1. If not, proceed to Step 3.

3. Replace $\mathbf{c}_i^s$ by $\mathbf{c}_i^{s'}$ with probability $W_i / \max\{W_i, W_n\} \times \max\{\tau_i, \tau_n\} / \tau_n$, and $\mathbf{c}_n^r$ by $\mathbf{c}_n^{r'}$ with probability $W_i / \max\{W_i, W_n\} \times \max\{\tau_i, \tau_n\} / \tau_i$

where $\tau_i$ and $\tau_n$ are the ion and neutral species time steps which can be different. Note that for the conditions of interest in this work, the first probability is essentially unity and the second one becomes equal to $W_i/W_n \times \tau_n/\tau_i$.

Serikov et al. [67] demonstrated that this algorithm is well suited for computational modeling of low temperature collisional plasmas in magnetron sputtering discharge applications for argon neutral and ion species. However, during the implementation stage of this algorithm in SUGAR, it was found that due to the large difference in directed velocities between neutral and ion species, non-physical behavior was observed in the neutral species velocities. This was attributed to the fact that after a CEX collision, the neutral velocity is so large that using the neutral timestep resulted in neutrals traveling across several computational cells at once. To remedy this, a local timestep algorithm based on a species dependent time step was implemented. Kannenberg [68] proposed that each computational particle should be assigned a time weighting factor, $s$, given by

$$s = \frac{\Delta t}{\Delta t_{ref}}$$

(3.14)
where \( \Delta t \) is the timestep used for the individual computational particle when it is moved and \( \Delta t_{\text{ref}} \) is the reference timestep for the species type that the particle belongs to. In this work, the value of \( s \) is calculated only for the neutral particles that have undergone CEX collision and is equal to \( \frac{W_{i}}{W_{n}} \). The timestep used for each particle is scaled with a weighting factor, \( W_{\text{Par}} \) given by

\[
W_{\text{Par}} = W_{\text{Par,ref}} \times s \tag{3.15}
\]

where \( W_{\text{Par,ref}} \) is the reference weighting factor for the species that the particle belongs to. Similar to the calculation of \( s \), \( W_{\text{par}} \) is only calculated for neutral particles that have undergone CEX collision. Flux conservation must be satisfied when calculating the velocity during macroparameter sampling. This is accomplished by multiplying the calculated \( W_{\text{par}} \) with each of the computational particle’s macroparameters while looping over the particle list for each V-Mesh cell. In summary, the basic idea behind this scheme is to extrapolate the impact of collisions between fast and slow species during the shorter time interval of the faster species to the longer time interval corresponding to the slower one. To demonstrate the improvement seen through the use of species dependent time steps we consider the neutral velocity profile along the center line of the thruster for a simulation that models the three collisional processes in the presence of an electric field (later referred to as Case 3). Figure 3.4 shows a comparison of the two neutral velocity profiles along the centerline of the thruster for the pure Serikov and the combined Serikov and Kannenberg approaches.
It can clearly be seen that Serikov algorithm by itself results in discontinuities in the neutral velocity profile whereas the two approaches together results in a smooth, physical result. It should also be noted that both methods predict the same neutral number density profiles.

The boundary conditions for the computational domain and the surfaces of the thrusters were taken as outflow boundary conditions where once a computational particle reaches these boundaries, it is deleted from the simulation. With respect to the thruster surfaces, during the initial stages of the simulations, it was observed that since so few particles collided with the thruster surface, gas surface interactions have essentially no effect on the general structure of the highly diffuse flow field. Furthermore, since the main focus of this work is understanding the back flow phenomena from particles emanating from the thruster, the simple removal of particles impacting the thruster surface was implemented. However, in future work higher fidelity plasma-surface interactions
will be implemented through the use of cut-cell and ray-tracing algorithms within the AMR/octree mesh approach. For the simulations that model the solar panels, a gas surface interaction model is used that reflects incoming computational particles from the solar-cell surfaces. A simple ray tracing algorithm is used to detect such collisions and a diffuse boundary condition is imposed [44] on the computational particles that is governed by the solar panel temperature. If an ion impacts the solar cell surface, it is assumed to become neutralized.

3.3 Modeling of the Vacuum Chamber

In a typical chamber configuration, a gas is fed into the chamber at the thruster exit with prescribed flow conditions. To simulate a chamber pump capability in the simulations, two rectangular prisms are used in the computational domain. If a computational particle reaches the location of these pumps, shown in Fig. 7.1, it is removed from the simulation. Employing this numerical pump does not cause any additional complexity to the octree implementation since particles are not allowed to reside inside these pumps. Therefore, there is no refinement of the octree meshes at these locations. The numerical pumps are used from the start of the simulations, which means any particle reaching the pumps at any timestep is removed. While these pumps create a low pressure region in their vicinity, changing the cross-sectional area of these pumps serves as a means to control the background pressure throughout the domain. Three pump sizes, which were determined by trial and error, will be considered. They correspond to a pumping rate of 140,000 L/s, which is typical of the values found in chamber facilities [25, 26, 27].
The boundary conditions for the computational domain surface and the outer surfaces of the thrusters were taken as physical walls that reflect any impinging particles based on the momentum accommodation coefficient for diffuse reflection for Maxwell’s model [69]. That is, the reflection is specular if the coefficient, \( \alpha \), is 0, or diffuse if it is 1 and partially diffuse walls can be prescribed with a value between 0 and 1. The simulations uses a globally prescribed \( \alpha \) value. If a neutral particle encounters a surface, the outcome of a specular or diffuse reflection is evaluated by generating a random number, \( R_1 \), between 0 and 1. If \( R_1 \) is less than \( \alpha \), a diffuse reflection is carried out, otherwise the reflection is specular. If the reflecting particle is an ion, a random number, \( R_2 \), between 0 and 1 is generated. If \( R_2 \) is greater than \( \alpha \), a specular reflection of the ion particle is carried out without neutralization. If \( R_2 \) is less than \( \alpha \), an extra random number, \( R_3 \), is generated. If \( R_3 \) is less than \( \frac{W_i}{W_n} \), the ion particle changes its type to a neutral (i.e., neutralization) and leaves the surface with a speed accommodated to the wall temperature. If \( R_3 \) is greater than \( \frac{W_i}{W_n} \), the ion particle is deleted from the simulation. With this treatment for the ions, the mass flux of ions and neutrals are conserved. Moreover, the chamber walls and thruster surfaces are assumed to be electrically grounded, which means that only the electric field caused by the self-induced plasma is modeled.

In terms of the parallel framework implementation, each MPI rank in the simulation stores the location of the chamber walls, numerical pumps, as well as the geometric description of the thruster. Since the size of this geometric information is very small, it essentially has no effect on the processor memory and therefore does not affect the load-balancing of the processors. During the movement stage of the computational particles,
each computational particle is checked to determine if surface impingement will occur. If this is true, the particle is reflected based on the physical model discussed above. This procedure is independent of the particle mapping to the octree mesh because, once the computational particles are moved, they are assigned their new positions and then mapped to the three meshes.

3.4 Use of Multiple AMR Meshes

In this work, three individual AMR/Octree meshes that independently adapt according to the relevant, specific phenomena are used. The first of these meshes is named the collision mesh (C-Mesh), on which DSMC computational particles are mapped to octree cells to model the collisions occurring in the flow via the DSMC method. In the second mesh, the electric field mesh (E-Mesh), DSMC computational particles that represent charged species are mapped to PIC cells to model the induced space charge. The third mesh, referred to as the visualization mesh (V-Mesh), is used to properly render the flow field macroparameters such as number density, velocity, and energy. The simultaneous usage of multiple meshes is a challenge requiring book-keeping of the flow variables, related numerical variables, and spatial information such as computational particles. Use of object-oriented programming makes these trackable since all three meshes share common class declarations. The variables used and computed by each mesh was controlled by higher level program declarations, which in practice meant that each mesh was invisible to the other two and considerable potential bottlenecks were avoided.
The refinement procedure is often controlled by a user-defined criteria. The refining, that is the creation of new children, is done recursively until this criterion is satisfied. One powerful aspect of this definition is that multiple criteria can be employed. A user-defined limit of the maximum level of refinement to be present in the simulation can also be prescribed. Arslanbekov et al. [48] used a refinement criterion based on the mean free path of the flow in their demonstration cases and found that the resultant mesh was able to resolve the flow field with sufficient resolution. The criteria for adapting cells in the C-mesh is that the cell size should be equal or smaller than the local mean free path. For the E-Mesh, the criterion is to have cell sizes that are of on the same order of the local Debye length at each cell location. Finally for the V-Mesh, there is not a physically enforced criterion as such. The selection is usually done so that the V-Mesh cells are coarser than C-Mesh cells [44, 61]. In this work, the number of particles was used as the criterion for subdividing the cells for the three meshes, i.e. if a cell had more particles than the given criterion, the cell was refined. Specifically, cells were subdivided if the number of particles were greater than 80, 500, and 250 for the C-, E-, and V-meshes, respectively. The criterion for each V-Mesh cell was set to be 250 computational particles which was found to give a smooth profile for flow field variables. For example, the cells in the C-, E- and V-meshes just near the thruster exit had 12, 200 and 84 particles respectively at steady-state.

In order to give an overall view, these three meshes are shown in Fig. 3.5 for a single thruster configuration for Case 3 defined in Table 5.1. In addition to this single thruster configuration, V-Mesh for a three thruster configuration when the thrusters are perturbing into the computational domain is given in Fig. 3.6 similarly for Case 3.
Fig. 3.5. Three computational meshes used for Case 3
Fig. 3.6. V-Mesh, visualization mesh for a single thruster
Chapter 4

Comparison of SUGAR with a 2-level DSMC code

4.1 Comparison of Performance and Results of Neutral Expansion to Vacuum Simulations using a Cartesian Versus an AMR-Octree Grid

A comparison study was performed of SUGAR with respect to the well-established two-level Cartesian SMILE [44] code for a four thruster configuration with neutral species only since species specific timesteps are not implemented in SMILE. Moreover, neutral only flows create a good test for AMR/octree schemes since they capture the majority of the large number density variation created in the supersonic expansion to vacuum. Figure 4.1 shows the side and front views of the computational domain where the four thrusters (with diameters of 0.1 m) form a square. This configuration was selected to emphasize a highly three-dimensional problem, as required for real on orbit missions. The neutral species conditions at the thruster exits and the key numerical parameters used in the simulations are given in Tables 4.1 and 5.2, respectively.

Generally, in SMILE, the collision grid system used to select particles for collisions is constructed at the first level with square Cartesian cells of size on the order of the estimated local mean free path. To improve the computational efficiency, in cases with high number density gradients, second-level sub-cells inside each first level cell are created during the simulations to adapt the cell size to the low local mean free path. For
Fig. 4.1. Computational domain for SUGAR-SMILE comparison studies
Table 4.1. Comparison of numerical parameters between SMILE and SUGAR

<table>
<thead>
<tr>
<th></th>
<th>SMILE, Run 1</th>
<th>SMILE, Run 2</th>
<th>SUGAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of computational particles</td>
<td>≈1,000,000</td>
<td>≈2,000,000</td>
<td>≈266,000</td>
</tr>
<tr>
<td>Ratio of true to computed number of particles</td>
<td>$2 \times 10^8$</td>
<td>$1 \times 10^8$</td>
<td>$7.5 \times 10^8$</td>
</tr>
<tr>
<td>Number of computational cells</td>
<td>≈2,000,000</td>
<td>≈2,000,000</td>
<td>≈38,000</td>
</tr>
<tr>
<td>Domain size (m$^3$)</td>
<td>$1.0^3$</td>
<td>$1.0^3$</td>
<td>$1.0^3$</td>
</tr>
<tr>
<td>Total number of time steps$^a$</td>
<td>57,000</td>
<td>57,000</td>
<td>57,000</td>
</tr>
<tr>
<td>Total number of time steps for sampling</td>
<td>40,000</td>
<td>40,000</td>
<td>40,000</td>
</tr>
<tr>
<td>Elapsed real time (s)</td>
<td>1600</td>
<td>2600</td>
<td>1700</td>
</tr>
<tr>
<td>Number of processors</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
</tbody>
</table>

$^a$A time step of $t = 5 \times 10^{-6}$ s is used in all simulations.

the comparison performed here, the SMILE adaptive capability was employed with a maximum cell size refinement of four. The baseline Cartesian mesh had 128 cells in all directions with a cell size of $7.8125 \times 10^{-3}$ m.

The spatial distribution of the neutral number density and x-component of velocity values for the four-thruster geometry in the $x - z$ plane at $y/D = 4$ predicted by SUGAR is shown in Fig. 4.2. The same contours predicted by SMILE (the $F_{NUM} = 1.0 \times 10^8$ case) are in excellent agreement with SUGAR and both simulations predict similar interaction structure between the thruster plumes. In order to have a quantitative comparison between the two solutions, the solutions along a line starting from the exit of the top left thruster ($x/D, y/D, z/D = 0.0, 4.0, 6.0$) to the end of the computational domain ($10.0, 4.0, 6.0$) of Fig. 4.1b were compared. Figures 4.3a and 4.3b show that the number density and velocity profiles are in good agreement for the three $F_{NUM}$ values. The insert in Fig. 4.3b shows that beyond $x/D = 5$, however, the SMILE results begin to fluctuate and are less smooth compared to SUGAR. The nature of this fluctuation is
due to a low number of particles per cell values in these cells in the SMILE runs and although the level does not seem to be large, this would be a problem if we were trying to estimate plume back flow fluxes which are even two orders of magnitude lower than what we find in the core flow.
Fig. 4.2. Neutral number density and x-component of velocity contours predicted by SUGAR for a four thruster configuration.
Fig. 4.3. Comparison of neutral number density and velocity profiles along the plume centerline for SUGAR and SMILE.
4.2 Hypersonic Flow over a Double-wedge Configuration

In this section, the flow over a double-wedge configuration is presented to show the capability of SUGAR for simulating complex three-dimensional geometries with comparison to SMILE [44]. The geometry, shown in Fig. 4.4, has been used in the experiments conducted by Swantek and Austin [70] to analyze the impact of the thermochemical effects on shock wave boundary layer interactions at different stagnation enthalpies. At continuum-like conditions, Edney type of shock-shock interactions are observed which include the attached oblique shock formed by the first wedge, detached bow shock caused by the upper wedge, separation and reattachment shocks resulting from their interactions at the triple point, separation of the boundary layer near the hinge, and three-dimensional effects [71]. However, for this preliminary case, the chosen number density for the simulation is too low to observe the Edney type of shock interactions. While Table 4.2 shows the numerical parameters used in this study, detailed explanation of the geometry implementation and more comparison cases can be found in Sawant et al. [72].

Figure 4.5 shows the collision mesh formed in this simulation at a steady state. For the SUGAR mesh, the zoomed-in section near the surface reveals a fourth level of refinement. Such a high level of refinement would help capture the separation near the hinge, shear layer, and shock-boundary layer interactions for the continuum-like Knudsen number which is our ultimate goal. Note that the SMILE code can achieve two levels of refinement. Therefore, to obtain a similar level of refinement using the SMILE code in the vicinity of the surface, the uniform grid must be created with a cell size equal to that of the third level cell in the SUGAR code. This approach is computationally
disadvantageous because of the additional efforts spent in refining the domain that does not need such a high level of refinement. Also, the grid needs to be set considering the lowest resolution, which is hard to predict in such simulations. Thus, the SUGAR code stands out in such extreme cases where multi-scale phenomena need to be captured efficiently.

Figure 4.6 shows the spatial distribution of translational temperature and the velocity field in the plane passing through the center of the wedge on the V-Mesh for the SUGAR and SMILE codes. Because of the differences in the orientation of the axes used for both codes, the center plane of the wedge in the SUGAR code is at $y=0.060$ m as shown and for the SMILE code it is at $z=0.0$ m. Looking at Fig. 4.7a, the shock thickness and the highest temperature obtained after the bow shock decreases along the span as expected due to the three dimensionality effects. Moreover, Fig. 4.7b shows the streamlines which verifies the spanwise velocity component.

To make a quantitative comparison, data is extracted along the dashed red line shown in Fig. 4.4. Figure 4.8 shows the number density, velocity in the x-direction, and translational temperatures for both codes. The number density and velocity profiles are in a good agreement. Small discrepancies are observed in the translational temperature profile which is likely due to the small differences in the sampling cell sizes and the number of particles per cell between two runs. Moreover, both the codes predict the highest temperature value of around 13,400 K achieved after the bow shock in the center plane of the wedge.
Fig. 4.4. Schematic of the double-wedge.
Table 4.2. Numerical parameters for the flow of argon over a double-wedge.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>SUGAR</th>
<th>SMILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Density [#/m³]</td>
<td>9.33×10¹⁰</td>
<td>9.33×10¹⁰</td>
</tr>
<tr>
<td>$F_{NUM}$</td>
<td>0.25×10¹¹</td>
<td>0.25×10¹¹</td>
</tr>
<tr>
<td>Freestream Temperature [K]</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Freestream Velocity [m/s]</td>
<td>4200</td>
<td>4200</td>
</tr>
<tr>
<td>Mach Number</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Time Step [s]</td>
<td>5.0×10⁻⁸</td>
<td>5.0×10⁻⁸</td>
</tr>
<tr>
<td>$\alpha_l$ &amp; $\alpha_e$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Surface Temperature [K]</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Elastic Collision Model</td>
<td>VHS</td>
<td>VHS</td>
</tr>
<tr>
<td>Viscosity Index</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>Number of Samples</td>
<td>12000</td>
<td>12000</td>
</tr>
<tr>
<td>Smallest Cell Size on the V-Mesh [m]</td>
<td>6.25×10⁻⁴</td>
<td>6.25×10⁻⁴</td>
</tr>
<tr>
<td>Number of Particles</td>
<td>51,990,000</td>
<td>59,850,000</td>
</tr>
<tr>
<td>Number of Processors Used</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Time Required for Sampling (min)</td>
<td>330</td>
<td>93</td>
</tr>
</tbody>
</table>
Fig. 4.5. Collision mesh comparison for SUGAR and SMILE.
(a) Translational temperature

(b) Velocity in X-direction

Fig. 4.6. Comparison of contours for the flow of argon over a double-wedge.
Fig. 4.7. 3-D effects for the flow of argon over a double-wedge in SUGAR.

(a) Temperature reduction in span-wise direction

(b) Streamlines over the double-wedge
Fig. 4.8. Comparison of macroparameters for the flow of argon over a double-wedge.
Chapter 5

Application of SUGAR on Ion Thrusters Plumes

Two main cases are set up to demonstrate the capability of SUGAR for an ion thruster plume simulation. Figure 5.1 shows the computational domain for modeling a single ion thruster plume with both neutral and ion species. The thruster exit is located at the center of the domain (0.0 m, 0.5 m, 0.5 m) with a thruster radius of 0.05 m. Note that all the dimensions in the subsequent figures are normalized by the thruster diameter (D=0.1 m). A second computational domain is shown in Fig. 5.2 for a three thruster configuration to investigate the interactions between the thrusters. The gaseous species of Xe and Xe\(^+\) are used in all simulations. The boundary conditions, except for the thruster exits, are taken as outflow boundary conditions and once a computational particle reaches these boundaries, it is deleted from the simulation. Moreover, since the quasi-neutrality assumption is used to solve for the electric field in the domain no electrical boundary conditions are required.

In order to observe the effect of the interaction between species on the plume structure and velocity fields, collisions between ions and neutrals and the presence of the electric field are turned on sequentially. While the definition of the cases are given in Table 5.1, only Cases 1, 2 and 3 are used for this section and Case 4 is left to be discussed in the next section. Case 3 is the most realistic simulation among the three cases because both MEX and CEX collisions and the electric field are modeled. No
Fig. 5.1. Computational domain for a single ion thruster
Computational Domain

Ion Thrusters

Fig. 5.2. Computational domain for three ion thruster configuration shown in the y-z plane.
collisions indicate that there are no collisions between neutral-neutral and neutral-ion species. The operating conditions for the ion thruster for these three cases which involve both neutrals and ions are summarized in Table 5.2 which were selected based on thruster conditions given in Refs. [17] and [29].

<table>
<thead>
<tr>
<th>Case</th>
<th>MEX &amp; CEX</th>
<th>Electric field due to $\Phi_p$</th>
<th>Electric field due to $\Phi_{BC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Case 2</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Case 3</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Case 4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

In terms of the numerical parameters related to the simulations presented in this chapter, both single and multiple thruster cases used a timestep of $\Delta t_{ref} = 4.88 \times 10^{-6}$ for neutrals and $\Delta t_{ref} = 2.44 \times 10^{-8}$ for ions. The weighting factors for neutrals and ions were $W_n = 1.0$ and $W_i = 0.005$ respectively and the simulated particle to real particle ratio was $F_{num} = 0.225 \times 10^9$. The simulations were run for a total of 120,000 steps and sampled for 100,000 steps. The meshes were adapted for a total of ten times between timesteps of 10,000 and 20,000. The single thruster cases required approximately 30,000 cells for the V-Mesh, 44,000 for the C-Mesh and 64,000 for the E-Mesh at the steady state. The total number of computational particles in the domain was approximately 11,000,000. The total simulation time was around 15 hours using 256 AMD Interlagos Opteron processors with a core speed of 2.5 GHz. The multiple thruster cases ended up
Table 5.2. Exit conditions for the thrusters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0$, K</td>
<td>300</td>
</tr>
<tr>
<td>$p_0$, mPa</td>
<td>5.054</td>
</tr>
<tr>
<td>$n(m^{-3})$, Xe</td>
<td>$4.6 \times 10^{16}$</td>
</tr>
<tr>
<td>$n(m^{-3})$, Xe$^+$</td>
<td>$2.3 \times 10^{15}$</td>
</tr>
<tr>
<td>$U$(m/s), Xe</td>
<td>200</td>
</tr>
<tr>
<td>$U$(m/s), Xe$^+$</td>
<td>40,000</td>
</tr>
</tbody>
</table>

using approximately 75,000 cells for the V-Mesh, 116,000 for the C-Mesh and 147,000 for the E-Mesh. For multi-thruster cases, the number of computational particles in the domain was approximately 34,000,000 and the total simulation time was around 34 hours using the same type of processors as in the single thruster case.

5.1 Single Ion Thruster Cases

First we present the C-Mesh, E-Mesh and V-Mesh for the Case 3 simulation obtained for a single thruster configuration in Fig. 5.3. For Cases 1 and 2, the three meshes look similar to Case 3, as the flow field is not dramatically changed by inclusion of the MEX and CEX collisions and electric field in the domain. Since these three meshes have different cell refinement criteria as discussed in Chapter 3, it can clearly be seen that the resultant meshes look different. Another important computational metric which is dependent on the C- and E-meshes, respectively, is the local Knudsen number and the ratio of the Debye length to the cell size. Figure 5.4a shows the spatial distribution of the local Knudsen number, the ratio of the mean free path to the cell size from the C-mesh, at the center plane of the domain. The mean values are around 1,000 indicates that the
cell sizes are very small compared to local mean free path. However, the relative velocity between the neutrals and ions can be as high as 40,000 m/s which increases the collision frequency, sharpens the flow gradients, and therefore requires cells to be much smaller than the usual DSMC requirement.

Even though a very small number of collisions are taking place in each timestep, since the outcome of a CEX collision is a large change in terms of velocity for each species, it is important to model these low frequency collisions properly [29]. Similarly, the Debye length to cell size ratio in the E-Mesh is shown in Fig. 5.4b. It can be seen that the cell sizes are on the order of the Debye length in the core of the plume but are significantly larger than one in the region of backflow. This is a typical requirement for ion thruster plume simulations [17, 29, 18, 16].

Starting with the spatial distribution of macroparameters, Fig. 5.5a, shows a zoomed-in view of the neutral number density contours that illustrate the rapid decay in magnitude due to comparable thermal and directed velocity components. The ion number density contours are shown in Fig. 5.5b where it can be seen that, in contrast to the neutrals, the ions exhibit a collimated profile due to the fact that their velocities along the $x$-axis are large. The figures demonstrate that the AMR mesh adaption and species specific timesteps capture the flow gradients well. Figure 5.6a shows the neutral velocity contours where the expansion shows a smooth profile with no distinct core region typical of supersonic expansions to vacuum. For the ion counterpart, one can observe from Fig. 5.6b that there is a distinct core region for the ion expansion. This is caused by the fact that the directed velocity along the $x$-axis (40,000 m/s) is so large that ion thermal effects are small. It can also be observed there is an ion presence outside the
Fig. 5.3. Three computational meshes used for Case 3
Fig. 5.4. Cell size ratios related to collisions and electric field mesh resolution.
core region \((2 < \frac{x}{D} < 6, 7 < \frac{z}{D} < 8)\) mainly through the CEX collisions which diverts the path of some ions towards these regions outside the core.

In order to have a better understanding of the various mechanisms modeled in the simulations, superimposed streamlines with velocity contours are generated for each case listed at Table 5.1. Neutral streamlines, which are shown only for Case 3 in Fig. 5.6a for brevity, resulted in a negligible difference as they are not much affected by the collisions and not at all by the induced electric field. Looking at the ion streamlines, Case 1 (Fig. 5.7a) shows that the ions remain inside the core region of expansion. However, with the inclusion of collision modeling for Case 2 (Fig. 5.7b) and Case 3 (Fig. 5.6b), one can observe ions that travel outside the core region. Inclusion of the electric field in Case 3 changes the contour values compared to Case 2, yet, the streamline structure does not considerably change. Figure 5.8 shows that the potential values decrease from the core region radially outwards and this in turn results in ions gaining energy from the electric field to populate the off-core regions. To study the cross-wise variation of ion properties outside of the core region for the different cases, the ion number densities and velocities are shown along a line starting from \((x/D, y/D, z/D = 7.5, 5.0, 5.0)\) to \((7.5, 5.0, 10.0)\) in Fig. 5.9.

Figure 5.9a shows that Case 3 has the smallest ion number density at the center location \((z/D = 5.0)\). However, outside the core location \((z/D = 7.5)\), it has roughly an order of magnitude higher value due to the presence of the electric field. Similarly, the velocity along the \(x\) direction is also increased roughly four times for Case 3 compared to Case 2, as shown in Fig. 5.9b. These plots show that the ions are electrostatically
Fig. 5.5. Number density contours on X-Z plane at y/D = 5 for Case 3

(a) Neutrals

(b) Ions
Fig. 5.6. Velocity contours on X-Z plane at y/D = 5 for Case 3
Fig. 5.7. Ion velocity contours on X-Z plane at y/D = 5 superimposed with streamlines
Fig. 5.8. Plasma potential contours
Fig. 5.9. Cross plume variation of ion properties
diverted towards the off-core regions for Case 3, which results in widening the plume compared to the cases which did not model the self-induced electric field.

In terms of the centerline thruster properties for the three cases, the neutral number density from the thruster exit \((x/D, y/D, z/D = 0.0, 5.0, 5.0)\) to the end of the domain \((10.0, 5.0, 5.0)\) is shown in Fig. 5.10a. It can be seen that for all the cases, the neutral variation is close implying that neither MEX or CEX collisions or the electric field has an apparent effect on the neutral number density. However, in Fig. 5.10b, for Cases 2 and 3, which both model MEX and CEX collisions, it can be seen that the ion number density is up to 10% different and also the Case 3 solution seems to be in better agreement with Case 1. This is due to the fact that the two phenomena (MEX and CEX collisions versus the electric field) have an opposite effect where the first one increases the ion number density level via collisions and the second decreases it by accelerating ions. However, this discrepancy is reduced in further downstream regions since both MEX and CEX collisions are not as frequent and the potential gradient is much smaller.

With respect to species velocities, the neutral velocity variation is presented in Fig. 5.11a. For Cases 2 and 3, an increase in velocity magnitude is observed from the start compared to Case 1 with a difference of up to 15% far downstream. An additional simulation was performed that only included neutral-neutral collisions without an electric field (i.e. a variation of Case 1). This gave an identical velocity profile to Case 1 of Fig. 5.11a. Therefore, the difference of 15% observed in the downstream for the neutral velocity profiles of Cases 2 and 3 is primarily caused by neutral-ion collisions. This result is noteworthy since the ions actually affect the neutral field, contrary to the common assumption that is usually used that the coupling between the neutral and ions is only one
Fig. 5.10. Number density distribution along the centerline of the thruster
Fig. 5.11. Velocity ($U$) distribution along the centerline of the thruster
way (neutrals affect ions, but not vice versa) [17, 73]. Figure 5.11b shows the ion velocity variation along the expansion direction. For Case 2, one can observe the deceleration of ions caused by CEX collisions, especially near \( x/D = 1 \) location. Downstream of this location, the velocity magnitude seems to recover but does not reach the original velocity, falling off approximately 2%. However, for Case 3, the ion velocity does not decrease as much compared to Case 2. The lowest velocity magnitude is again near the \( x/D = 1 \) location but further downstream the velocity recovers to the original value. This behavior is consistent with the predicted ion density, i.e., the potential field decreases along the ion flow direction thereby accelerating the ions.

### 5.2 Three Ion Thruster Cases

The geometry shown in Fig. 5.2 for the Case 3 conditions is now considered here. First we present a comparison of our results with those of Hyakutake [17] for the neutral species. In their work [17], an overlay approach was used where the neutral flow fields were first obtained without any ions. In a second simulation, beam ions were introduced into the computational domain and CEX ions were created by using the steady state neutral number density profiles and local CEX rates computed in the first simulation. Figure 5.12 shows a comparison of neutral density profiles at the centerline of the domain for the single thruster case and the centerline for the top thruster in Fig. 5.2 of the three thruster configuration. Good agreement is found which implies that the overlay approach is adequate for modeling the neutral number density variation. However, as shown previously, up to 15% error is observed for the neutral velocity profile when two-way coupling is not used. It should also be emphasized that two results predict
the interaction of the thrusters to occur at the same location around two diameters
downstream of the exit of the top thruster in Fig. 5.2.

Moving to SUGAR two-way coupled results, Fig. 5.13a shows the neutral number
density for the top thruster in Fig. 5.2 at the center plane (y/D=5). Similar to the
single thruster cases, a large drop in number density can be observed. In addition, from
Fig. 5.13b which is a zoomed-in image of ion density contours, the interaction of the
off-plane plumes can be seen. The AMR algorithm smoothly captures this interaction
and refines the grids accordingly to the increase in density near the x/D=3 and z/D=5
regions.
Fig. 5.13. Neutral number density contours on X-Z plane at y/D = 5
Fig. 5.14. Velocity contours on X-Z plane at $y/D = 5$ superimposed with streamlines
Similarly, at the center plane (y/D=5), focusing on the top thruster in Fig. 5.2, Fig. 5.14a shows that the neutral velocity contours exhibit a slight asymmetry around the x-axis. Such asymmetry for the expansion of the flow is much more obvious for the ion velocity contours as seen in Fig. 5.14b. The reason for this is the fact that, the ion expansion is quite collimated up until x/D=4, but after this point the interaction between the plumes is quite strong. Moreover, investigating the expansion through the use of streamlines, Fig. 5.14a shows that the neutral velocity is indeed slightly asymmetric downstream of the thruster. Figure 5.14b shows that the ions stay collimated in the core region of the expansion with a slight deflection induced by the off-plane thruster. Outside the core region, there are streamlines present pointing radially outwards which are caused by dominantly by the CEX collisions. These outside-core streamlines also are asymmetric which is due to the ion plume interactions.

In order to make quantitative comparisons between single and three thruster cases, the flow field variables were plotted along a line parallel to the flow direction. A line that starts from the exit of the top thruster (x/D=0,y/D=5,z/D=6) and ends at the domain boundary (x/D=10,y/D=5,z/D=6) is selected to analyze the changes between the single and three thruster cases in terms of density and velocity profiles and is designated as "centerline". Fig. 5.15a shows the neutral density variation along the centerline. For neutrals, single and three thruster case shows a similar expansion profile up until x/D=1, but downstream of this location, the three thruster case shows a higher neutral density presence with up to four times difference at x/D=10. From Fig. 5.15b, one can observe that both ion results show a very similar profile regardless of the number of thrusters up until x/D=4 location. After this point, the three thruster case shows a clear interaction
Fig. 5.15. Number density distribution along the centerline of the top thruster for three thruster configuration and single thruster.
Fig. 5.16. Velocity($U$) distribution along the centerline of the top thruster for three thruster configuration and single thruster
from the off-plane thruster and approximately three times increase in the ion number density is observed until the end of the domain.

From Fig. 5.16a, the neutral velocity profiles for both cases can be seen. Except for the initial part of the expansion, single and three thruster cases exhibit a different acceleration profile. Downstream of the $x/D=0.75$ location, the three thruster case shows a slower acceleration due to the plume interaction but slightly overshoots at the very end compared to single thruster case. The ion velocity profiles given in Fig. 5.16b show a distinctly different profile for single and three thruster cases. The single thruster case shows a rather smooth transition from the deceleration to the acceleration region ($x/D = 1$), whereas the three thruster cases exhibit a fluctuating profile between $x/D=2$ and $x/D=8$ locations. To check if this behavior is due to the selection of numerical parameters, the number of samples during the sampling stage was increased from the baseline value of 100,000 timesteps to 400,000. Also another more refined visualization mesh was created to check if the mesh that was mainly adapted to the ion distribution in the domain had any role in the fluctuating profile. Likewise, the results did not change, thus leading us to conclude that this behavior is indeed caused by the interaction of the three plumes.
Chapter 6

Characterization and Quantification of Backflow Due to Ion Thruster Plumes

Similar to Chapter 5, two set of cases were set up to investigate the backflow phenomena in the plume. The first set of cases corresponds to the single thruster configuration whereas the second set corresponds to an array configuration of three thrusters. Figure 6.1 shows the computational domain for modeling a single thruster case from the side and front views with the center point of the thruster exit located at (0.1875 m, 0.5 m, 0.5 m). The computational domain corresponding to the three thruster configuration is shown in Fig. 6.2 which is selected to investigate the interaction of multiple plumes and their effect on the backflow. For all of the simulations, gaseous species of Xe and Xe$^+$ are used and are introduced into the computational domain at the thruster exit at each time step. At this inflow boundary condition, the neutral species are assumed to have a uniform profile in density and the ions have a Gaussian distribution in number density versus radial distance from the center of the thruster. The exit conditions used in the simulations for the thrusters are tabulated in Table 5.2 and are the same as in Chapter 5.

In terms of the numerical parameters related to the simulations, both single and multiple thruster cases used a timestep of $\Delta t_{ref} = 4.88 \times 10^{-6}$ and $2.44 \times 10^{-8}$ for neutrals and ions, respectively. The weighting factors for neutrals and ions were $W_n = 1.0$ and $W_i = 0.005$, respectively, and the simulated particle to real particle ratio was
Fig. 6.1. Computational domain for a single ion thruster

Fig. 6.2. Computational domain for three ion thruster configuration on y-z plane
$F_{NUM} = 0.225 \times 10^9$. The simulations were run for a total of 120,000 steps and sampled for 100,000 steps. The meshes were adapted for a total of ten times between timesteps of 10,000 and 20,000. The single thruster cases required approximately 30,000 cells for the V-Mesh, 38,000 for the C-Mesh and 64,000 for the E-Mesh at the steady state. The total number of computational particles in the domain was approximately 10,000,000. The total simulation time was around 15 hours using 256 AMD Interlagos Opteron processors with a core speed of 2.5 GHz. The multiple thruster cases required approximately 75,000 cells for the V-Mesh, 116,000 for the C-Mesh and 147,000 for the E-Mesh. For multi-thruster cases, the number of computational particles in the domain was approximately 32,000,000 and the total simulation time was around 27 hours using 256 AMD Interlagos Opteron processors with a core speed of 2.5 GHz.

It should be noted that the numerical parameters are similar to those of Chapter 5, however, the computational domain in the present work is shifted along the $-x$ direction by 0.1875 m the length of the thruster to model the back flow region. This domain shift reduced the number of computational particles from earlier work (11 to 10 million) because the back flow region has a lower density than the downstream region of the plume. In addition, it will be shown that the AMR/octree mesh strategy is able to capture the local flow features near the thruster exit using approximately 100 particles per collision cell as well as the back flow region with 10-20 particles per collision cell which will be shown to be adequate to resolve the many orders of magnitude difference in the number densities and fluxes.

In order to investigate the effect of various physical mechanisms on the back flow characteristics four cases are studied by sequentially including MEX/CEX collisions and
the influence of the electric field, as defined in Table 5.1. Note that Case 3 is the most realistic simulation compared to Cases 1 and 2 because both MEX/CEX collisions and the electric field are modeled whereas Case 1 has no collisions at all between neutral-neutral or neutral-ions. As can be seen from the Table 5.1, Case 4 studies the effect an additional electric field due to the presence of solar cell arrays charged at a nominal and elevated voltage value. As the downstream flow field was analyzed in Chapter 5, in this chapter, the results of backflow as well as the effects of the charged solar panels will be analyzed in detail.

6.1 Single Thruster Cases

In this Section, Cases 1 through 3 listed in Table 5.1 were simulated. Figure 6.3 shows the V-mesh derived from the simulations of Case 3 that will be used to display the flow field results. Similar to the simulations in Chapter 5, Cases 1 and 2 have mesh structures similar to Case 3 since the general flow field structure is not dramatically changed by inclusion of the MEX and CEX collisions or the electric field in the domain. The C- and E-Meshes were generated based on the ratios of local mean free paths and Debye lengths to the cell lengths. During the E-Mesh generation the cells are subdivided so that they are on the order of the Debye length for ion thruster plume simulations. It was found that everywhere in the computational domain except for the core region the ratio of the Debye length to cell size is significantly larger than the one shown in Chapter 5.

The nature of expansion of the neutrals and ions to vacuum is well-known [29, 30]. The neutral densities decay in magnitude rapidly due to the thermal velocity being
Fig. 6.3. V-Mesh, visualization mesh for a single thruster
comparable to the directed velocity. The ions also decay in magnitude quite fast but in contrast to the neutrals exhibit a collimated profile due to the fact that the ion velocity is quite large and directed along the x-axis.

The velocity profiles, on the other hand, are more sensitive to physical mechanisms in the flow field and are worthy of detailed study. Figures 6.4 and 6.5 show the spatial dependence of the neutral and ion species velocity fields and their associated streamlines in the $x - z$ plane for $y/D = 5$ for Cases 1, 2 and 3. In Fig. 6.4, which shows the neutral velocity contours, one can observe for Case 1, with no collisions or an electric field modeled, that all streamlines have a positive velocity along the x-axis and there is no back flow for either neutrals or ionic species. The presence of neutral particle streamlines in the back flow region for Case 3, shown in Fig. 6.4b, is due to collisions which induce a back flow towards the thruster surface. Furthermore, this implies that the effect of the electric field on the neutral velocity field is almost non-existent, as expected. Since Cases 2 and 3 predict essentially the same streamline structures, only Case 3 is shown. It should also be noted that for all the three cases, the neutral expansions show a smooth profile with no distinct core region typical of supersonic expansions to vacuum.

For the ion counterpart, one can observe from Fig. 6.5 that there is a distinct core region for the ion expansion for all three cases. Looking at the ion streamlines in Fig. 6.5a, Case 1, we see that the ions remain inside the core region of expansion. However, with the inclusion of collisions (Cases 2 and 3, Figs. 6.5b and c), one can observe ions that travel upstream of the thruster exit towards the backflow region. The inclusion of electric field modeling in Case 3, Fig. 6.5c, predicts curved streamlines toward the backflow region as well an increase in the ion velocities just outside the core region of the plume. Figure 6.6
shows that the potential, $\Phi_p$, decreases from the thruster exit to the backflow region and this in turn results in CEX ions gaining energy due to the electric field. The ion number density profiles are essentially the same for all the three cases in the downstream region and are within 10% of each other. One significant outcome of using a two-way coupling between neutrals and ions is that for Cases 2 and 3, the neutral velocity profiles differ by as much as 15% [30].

In order to have a better understanding of the conditions that are present in the backflow region, a cross line that starts near the center region of the $y - z$ plane for $x/D = 0$ and terminates at the end of the domain boundary is selected which starts from $x/D, y/D, z/D = 0.3, 5.0, 6.5$ and ends at $(0.3, 5.0, 10.0)$. For the single thruster cases, one can observe in Fig. 6.7a, a roughly constant neutral number density. Small variations are present between Cases 2 and 3.

In Fig. 6.7b, the ion number density can be seen and in contrast to the neutral number density profiles, the single thruster solution for Case 3 (with $\Phi_p$) results in almost three orders of magnitude increase in ion number density. This result indicates that the electric field has a strong effect on the backflow phenomena and predicts similar the ion number density levels compared to Hyakutake [17] which also modeled collisions and electric field.

Figure 6.8 shows the velocity profiles on the cross-line for neutrals and ions. In Fig. 6.8a, for single thruster cases, a similar velocity profile constant around -0.5 m/s is observed. The ion velocity profile in Fig. 6.8b shows that the velocity of ions for the single thruster configuration for Case 2 in the backflow region is around -25 m/s. However, once the electric field effect due to $\Phi_p$ is accounted for (Case 3), an approximately two
Fig. 6.4. Neutral velocity contours in the $X - Z$ plane at $y/D = 5$ superimposed with streamlines for a single thruster.
Fig. 6.5. Ion velocity contours in the $X-Z$ plane at $y/D = 5$ superimposed with streamlines for a single thruster.
Fig. 6.6. Plasma potential, $E_P$, contours for the single thruster Case 3 conditions.
orders of magnitude increase in ion velocity magnitude is observed. The potential field turns and deposits energy into the CEX ions resulting in a few thousand m/s speeds which are much higher compared to the post-collisional speeds for CEX collisions. This velocity profile also captures the trend seen in Hyakutake [17].

6.1.1 Angular Flux Study

A study was conducted to look at the angular variation of density, velocity and flux of both neutrals and ions. At a five diameter distance from the center of the thruster exit, four points are selected, as shown in Fig. 6.4b. Table 6.1 shows the number density, velocity, and flux present at these four points for the neutral species. Point 1 shows the highest density and velocity levels as expected since it is directly facing the thruster. At an angle of 30° (Point 2), the number density level is almost halved and the flux level is reduced by a factor of three. At Point 4, which is perpendicular to the core flow, the number density level is decreased several orders of magnitude, but, although small in magnitude, a backflow is observed.

Similarly, Table 6.2 shows the results for ions at the same four points shown in Fig. 6.4b. Although the number density level is similar to neutrals, the ion flux at Point 1 is approximately fifty times higher. Due to the collimated behavior of ions, the number density and flux levels drop sharply at the higher angles. At Point 4, compared to neutrals, ions have approximately four orders of magnitude higher flux towards the backflow region.
Fig. 6.7. Upstream number density variation in the back flow region.
Fig. 6.8. Upstream velocity along the $x$-direction variation in the back flow region.
Table 6.1. Selected macro parameters for Xe as a function of angle from the thruster centerline.\textsuperscript{a}

<table>
<thead>
<tr>
<th>Point #</th>
<th>Angle</th>
<th>$n$ (m$^{-3}$)</th>
<th>$u$ (m/s)</th>
<th>Flux$^a$ (m$^{-2}$/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$7.74 \times 10^{14}$</td>
<td>344.29</td>
<td>$2.66 \times 10^{17}$</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>$3.68 \times 10^{14}$</td>
<td>255.17</td>
<td>$1.10 \times 10^{17}$</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>$6.79 \times 10^{13}$</td>
<td>126.39</td>
<td>$1.77 \times 10^{16}$</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>$3.83 \times 10^{9}$</td>
<td>-3.13</td>
<td>$1.35 \times 10^{10}$</td>
</tr>
</tbody>
</table>

\textsuperscript{a}With respect to the surface facing the thruster.

Table 6.2. Selected macroparameters for Xe$^{+}$ as a function of angle from the thruster centerline.\textsuperscript{b}

<table>
<thead>
<tr>
<th>Point #</th>
<th>Angle</th>
<th>$n$ (m$^{-3}$)</th>
<th>$u$ (m/s)</th>
<th>Flux$^b$ (m$^{-2}$/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$2.36 \times 10^{14}$</td>
<td>40014.6</td>
<td>$9.44 \times 10^{18}$</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>$8.98 \times 10^{10}$</td>
<td>15488.3</td>
<td>$1.92 \times 10^{15}$</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>$1.21 \times 10^{11}$</td>
<td>3029.5</td>
<td>$1.23 \times 10^{15}$</td>
</tr>
<tr>
<td>4</td>
<td>90</td>
<td>$7.10 \times 10^{10}$</td>
<td>-1198.3</td>
<td>$3.72 \times 10^{14}$</td>
</tr>
</tbody>
</table>

\textsuperscript{b}With respect to the surface facing the thruster.
6.1.2 Ion Energy Distribution

Another study to analyze the backflow was to look at the ion energy distributions in the backflow region for Case 3 of the single thruster configuration. The region where the energy distribution will be analyzed, the collecting region, is selected to be a thin rectangular prism with a hole (the thruster) at the center. The surface is a y-z surface with a thickness of 6 mm (from x=0 mm to x=6 mm). During the sampling stage of the simulations, the velocities of computational ion particles that fall into this region were collected to generate energy distributions at steady state.

Figure 6.9a shows the energy distributions for the all ion particles that fell into the collecting region as well as a Maxwellian distribution at a temperature of 13 eV which was found to be the best fit to the actual distribution except that a deviation for energies greater than 20 eV towards the tail of the distribution is apparent. An additional theoretical distribution with a 3 eV shift on top of the 13 eV temperature was more successful at representing the mid-energy scale of the spectrum. The local potential value where each CEX ion was created plays an important role in such distributions as there is a continuous production of CEX ions with the highest rate being just near the thruster exit. A second energy distribution for two different shells selected to investigate the spatial distribution of $2 < r/D < 3$ and $4 < r/D < 5$ showed that the energy distributions remained essentially the same as those shown in Fig. 6.9a. Although there are small differences among the profiles, it can be stated that there is almost no spatial inhomogeneity in the distributions in the back flow region. Finally, a separate case was simulated where MEX collisions were turned off for Case 3. This was done since the outcome of MEX
collisions can cause post-collisional velocities which can point to the backflow region in three dimensions (i.e., having negative velocity component in x-direction). Figure 6.9b shows that the ion energy distribution in the backflow region is almost single-handedly caused by the electric field with MEX providing a small contribution to the population of ions at energies around 10 eV.

6.2 Three-Thruster Cases

The main motivation of developing a three dimensional capability for SUGAR was to be able to simulate an array configuration of thrusters where no symmetry is present such as shown in Fig. 6.2. The physical and computational parameters are the same as in Case 3 listed in Table 5.1 except for the two additional thrusters. Similar to the single thruster case, the visualization mesh can be seen for this multiple thruster configuration in Fig. 6.10. It can be observed that AMR capability captures the interaction smoothly and eliminates the need for creating computational meshes prior to the simulation for such arbitrary configurations.

It has been shown that, while features similar to single thruster cases are observed as decaying in magnitude quite rapidly in terms of number densities, the interaction of the plumes with each other can be captured with the AMR algorithm smoothly and the corresponding increase in density near the vicinity of the thrusters resulted in further refinement. At the center plane \((y/D = 5)\), focusing on the top thruster in Fig 6.2, Fig. 6.11a shows that the neutral velocity contours exhibit a slight asymmetry around x-axis which can also be seen in the streamlines. Such asymmetry for the expansion of the flow is much more obvious for the ion velocity contours as seen in Fig. 6.11b. The
Fig. 6.9. Ion energy distribution at the backflow plane for a single thruster configuration.
Fig. 6.10. V-Mesh, visualization mesh for three thrusters
reason for this is the fact that the ion expansion is quite collimated up until $x/D = 4$, but after this point the interaction between the plumes is quite strong. Similar to single thruster cases, it can be observed that there is a backflow of ions towards the backflow region.

Turning to the back flow region, in Fig. 6.7a, it can be seen that the neutral number density is about three to four times larger compared to the single thruster cases, and has more variation compared to the single thruster cases. Examination of Fig. 6.7b for the multiple thruster case shows that the ion number density level is three to four times higher than the single thruster case which implies there is roughly a linear relationship between the number of thrusters and ion number density level in the backflow region. Figure 6.8a shows that the velocity for neutrals is a bit higher than for the single thruster cases, around -1.75 m/s, and, for ions, the velocity magnitude shown in Fig. 6.8b, is about the same as that of the single thruster case, Case 3. In summary, neutral and ion number density levels exhibit a linear trend with respect to the number of thrusters but ion velocity is roughly constant regardless of the number of thrusters.

6.3 Solar Panel Effects

Following the approach discussed in Sec. 3.1.2, additional simulations were run for Case 4 conditions listed in Table 5.1 for a single thruster. The motivation of this case with a charged surface is to investigate the effects on the flow field for a configuration realistic for a generic spacecraft model.

Looking at the results, the overall flow features are almost the same with the results for Case 3 shown in Figs. 6.4b and 6.5c. However, if we examine the ion number
Fig. 6.11. Velocity contours on $X-Z$ plane at $y/D = 5$ superimposed with streamlines for the three-thruster case.
density and velocities along the orange dashed line in Fig. 3.2a which corresponds to the top of the solar panel, we can see important effects due to the presence of the solar cells. Figure 6.12 shows a comparison of the ion number density and velocities for different gas-surface models. The results designated as “w/o surface” mean that if an ion or neutral species impacts the solar cell array (which is at the very edge of the domain) it is removed from the computation. The simulations designated as “w gas surface” correspond to a gas surface interaction where an ion is neutralized and released with a wall temperature of 300 K and full accommodation is assumed for neutral species at the same wall temperature. The “no solar panel” case is Case 3 shown earlier in Figs. 6.7b and 6.8b. For the no gas-surface interaction case, a distinct increase in the ion number density as well as ion velocity magnitude along the x-direction is observed for the 3 V potential case. This result is expected as the electric field caused by the charged solar panels contributes to the energy deposition to the ions and results in a larger backflow. Similarly, as the magnitude of the potential is increased to 10 V, the monotonic increase in the density and velocity magnitude is also observed. However, once the gas surface interaction is modeled between the flow and the solar panels (blue curves), a two to three times decrease in the ion number density is observed due to the neutralization of ions as they impact the solar cell panels. It is interesting to note that the ion number density decreases almost in a linear trend as the distance from the ion thruster exit whereas the velocity magnitude remains fairly constant at about 1,000 m/s. It should also be stated that the change in the potential of the solar panel results in almost no change in the number density and velocity magnitude values which implies that gas-surface interactions dominate over the charging effects for the range of surface voltages considered.
An additional configuration was created so that the solar panels were placed downstream of the ion thruster. This configuration has the importance from the perspective of constellation type of missions that employ several satellites in arbitrary formation. During such a scenario, the solar panel of a satellite could be inadvertently bombarded by an ion thruster of a neighbor satellite. We keep the configuration of the problem unchanged by moving the solar panels from the upstream to downstream location. In other words, the configuration of the solar panels on y-z plane (shown in Fig. 3.2a) are moved from the $x/D = 0$ to the $x/D = 10$ location. The first change that can be observed is in the neutral velocity profile structure shown in Fig. 6.13a. The presence of the solar cell panel creates a weak compression region, slowing the neutral atoms down and the AMR-Octree grid automatically refines near the $x/D = 10, z/D = 5$ location to capture this effect. For the ions (Fig. 6.13b), one can observe very small changes due to the neutralization of ions on the solar panels compared to the case which had solar panels at the upstream or the case which had no solar panels. In Fig. 6.14, top and bottom half of the results correspond to downstream and upstream solar panel configurations, consecutively. Figure 6.14a shows the ion number density profiles where a considerable similarity can be observed except for the region that is far away from the ion thruster diagonally where the upstream configuration predicts almost an order of magnitude smaller densities. For the velocity contours (Fig. 6.14b), the similarity is again observed where slight changes are observed at upstream locations.

Even though the flow structure does not change dramatically between the two cases, it is apparent that the conditions near the solar panels for both cases will be very different. Table 6.3 summarizes the conditions for the two cases. Two set of points
Fig. 6.12. Ion profiles for different solar panel voltages and gas surface interaction models.
Fig. 6.13. x-component of velocity contours for the downstream solar panel configuration at $y/D = 5$ plane.
Fig. 6.14. Comparison of ion number density and velocity fields for upstream and downstream configurations at y/D=5 plane
are selected that are 1 cm away from the solar panel surfaces. For points at $z/D = 7$, the upstream solar panel has two to three orders of magnitude higher values in number density and velocity. For points at $z/D = 9$, number densities are at the same order of magnitude where the velocity magnitude is almost ten times higher. These results show that for a constellation type mission, such a direct exposure from an ion thruster from another satellite will have a few to several orders of magnitude higher flux values which will affect the degradation of solar panels much more than an indirect but consistent backflow of charged species.

Table 6.3. Conditions for upstream and downstream solar panel configurations, Xe

<table>
<thead>
<tr>
<th>Solar Panel Location</th>
<th>$x / D$</th>
<th>$z / D$</th>
<th>$n$ ($m^{-3}$)</th>
<th>$u$ (m/s)</th>
<th>Magnitude of Flux ($m^{-2}/s$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upstream</td>
<td>0.1</td>
<td>7</td>
<td>$2.61 \times 10^{10}$</td>
<td>-1.211</td>
<td>$3.16 \times 10^{13}$</td>
</tr>
<tr>
<td>Downstream</td>
<td>9.9</td>
<td>7</td>
<td>$2.95 \times 10^{13}$</td>
<td>39.804</td>
<td>$1.17 \times 10^{15}$</td>
</tr>
<tr>
<td>Upstream</td>
<td>0.1</td>
<td>9</td>
<td>$1.10 \times 10^{10}$</td>
<td>-734</td>
<td>$8.07 \times 10^{12}$</td>
</tr>
<tr>
<td>Downstream</td>
<td>9.9</td>
<td>9</td>
<td>$5.55 \times 10^{10}$</td>
<td>8,733</td>
<td>$4.85 \times 10^{14}$</td>
</tr>
</tbody>
</table>

Finally, it should be stated that these additional two cases which included the solar panel modeling took approximately 6.5 hours using 1024 MPI ranks while all the other numerical parameters were kept constant as described at the beginning of this section. Based on this timing, it can be said that the computational size of the problem
increased by approximately 25% with the inclusion of the extra calculations required for model the a three-dimensional spacecraft charged surface.
Chapter 7

Chamber Cases

7.1 Comparison of Single Ion Thruster Plumes in Chamber versus Space Conditions

In this section, simulations using a chamber and vacuum configuration are run. Figures 7.1a and b show the computational domain for modeling a single thruster plume from the side and front views with the center point of the thruster exit located at (0.1875 m, 0.5 m, 0.5 m). In subsequent figures, the dimensions are normalized by the thruster diameter \( D = 0.1 \) m. For all of the simulations, gaseous \( Xe \) and \( Xe^+ \) species are used and are introduced into the computational domain at the thruster exit at each time step. The temperature of the chamber walls and thruster exterior walls was set to 300 K. In terms of the numerical parameters related to the simulations, a timestep of \( \Delta t_{\text{ref}} = 4.88 \times 10^{-6} \) and \( 2.44 \times 10^{-8} \) for neutrals and ions was used, respectively. The weighting factors for neutrals and ions were \( W_n = 1.0 \) and \( W_i = 0.005 \), respectively. The simulations were run for a total of 200,000 timesteps and were sampled for 120,000 timesteps. The meshes were allowed to adapt for a total of six times between timesteps of 20,000 and 80,000 rather than for each timestep. This was deemed to be adequate because essentially no change was observed for more frequent adaptations. Table 7.1 lists the total computational particles, total collision cells, and the simulated particle to real particle ratios used for all the cases. Since the focus of this work is to characterize the trends
caused by the changes in operating a thruster in a chamber versus space, background density level, domain size as well as the number of thrusters operating simultaneously, the following subsections analyze the outcomes on the plume structures. Note that all simulations were run using 1024 to 2048 AMD Interlagos Opteron processors with a core speed of 2.5 GHz with simulation times totalling to 5-20 hours.

Table 7.1. Numerical Parameters for DSMC Simulations

<table>
<thead>
<tr>
<th></th>
<th>Number of Computational Particles</th>
<th>Number of Collision Cells</th>
<th>$F_{num}^a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>$2.0 \times 10^6$</td>
<td>292,727</td>
<td>$0.275 \times 10^{10}$</td>
</tr>
<tr>
<td>Vacuum</td>
<td>$1.5 \times 10^6$</td>
<td>26,314</td>
<td>$0.275 \times 10^{10}$</td>
</tr>
<tr>
<td>Larger Pump</td>
<td>$9.0 \times 10^6$</td>
<td>258,854</td>
<td>$0.275 \times 10^{10}$</td>
</tr>
<tr>
<td>Smaller Pump</td>
<td>$42.0 \times 10^6$</td>
<td>2,084,580</td>
<td>$0.275 \times 10^{10}$</td>
</tr>
<tr>
<td>Smaller Chamber</td>
<td>$4.6 \times 10^6$</td>
<td>243,195</td>
<td>$0.275 \times 10^{10}$</td>
</tr>
<tr>
<td>Triple Thruster</td>
<td>$61.0 \times 10^6$</td>
<td>2,072,134</td>
<td>$0.275 \times 10^{10}$</td>
</tr>
<tr>
<td>T6 Engine</td>
<td>$23.0 \times 10^6$</td>
<td>323,000</td>
<td>$0.875 \times 10^{10}$</td>
</tr>
<tr>
<td>UK-10 Engine</td>
<td>$8.0 \times 10^6$</td>
<td>121,000</td>
<td>$0.120 \times 10^{10}$</td>
</tr>
</tbody>
</table>

*a* Simulated particle to real particle ratio

We start by studying the effect of domain boundary condition type on the ion thruster plume. Table 7.2 presents two cases where the baseline case uses a chamber boundary condition and the vacuum case uses vacuum boundary conditions which effectively simulates the space environment. In the vacuum case, a particle that reaches the ends of the domain or the thruster surface is removed from the simulation. The boundary
Fig. 7.1. Computational domains for single and triple ion thruster configurations
condition has an effect on the simulations particularly in terms of the resultant computational meshes for the two cases. Since the cell refinement criterion is proportional to the number of computational particles, the vacuum case has larger cells and therefore lower refinement levels overall except near the thruster exit as can be seen from Figs. 7.2a and b. In other words, since the vacuum case has approximately ten times fewer computational particles compared to the baseline case, its computational mesh has cells with lower levels of refinement. Therefore, the computational meshes corresponding to each case reflect the nature of the flow, an expansion versus a confined flow, respectively.

Table 7.2. Thruster Plume Exit Conditions

<table>
<thead>
<tr>
<th></th>
<th>Thruster Baseline$^a$</th>
<th>T6$^b$</th>
<th>UK-10$^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ (m$^{-3}$), Xe, at thruster exit</td>
<td>4.6×10$^{17}$</td>
<td>1.217×10$^{17}$</td>
<td>8.22×10$^{17}$</td>
</tr>
<tr>
<td>$n$ (m$^{-3}$), Xe$^+$, at thruster exit</td>
<td>2.3×10$^{15}$</td>
<td>9.307×10$^{15}$</td>
<td>6.54×10$^{15}$</td>
</tr>
<tr>
<td>$U$ (m/s), Xe, at thruster exit</td>
<td>200</td>
<td>250</td>
<td>199</td>
</tr>
<tr>
<td>$U$ (m/s), Xe$^+$, at thruster exit</td>
<td>40,000</td>
<td>52,024</td>
<td>40,200</td>
</tr>
</tbody>
</table>

$^a$ The pump cross-section used is (1.0/16.0)$^2$ m$^2$

$^b$ The pump cross-section used is (2.0/16.0)$^2$ m$^2$

Figures 7.3a and b show the neutral number density contours for the baseline and vacuum configurations, respectively. Looking at Fig. 7.3a, it can be seen that while small in magnitude, there is indeed a spatial variation of neutral density with an average background level of approximately $6.0 \times 10^{16}$ m$^{-3}$. The $5.0 \times 10^{16}$ m$^{-3}$ isoline is close to the pump locations (near $x/D=9$, $z/D=1$ and $x/D=9$, $z/D=9$) for the baseline case,
Fig. 7.2. Values of AMR refinement level at y/D=5 for a ground chamber (a) versus space (b)
however as seen in Fig. 7.3b, the same isoline is closer to the thruster exit in the vacuum case. Hence, the neutral density levels drop sharply for regions farther away and the rest of isolines are located much closer to each other near the thruster exit. Moreover, the profiles along the thruster centerline presented in Fig. 7.3c show a clear difference as can be seen in the far field region where the neutral density levels are uniformly higher for the baseline compared to the vacuum case. This difference in the neutral density between chamber and space is similar to the background pressures reported in experiments [25, 26, 27].

The total ion number density contours were found to be essentially the same for the baseline and vacuum cases and therefore are not shown. Figure 7.4 shows the total ion density profile along the centerline of the thruster which is typical of ion thruster expansions. In contrast to the neutral species, it can be seen that the ions remain more columnated along the direction of the jet. This outcome is expected as the ions have a large momentum directed in the x-direction where the majority of them travel unhindered from the collisions.

Looking at the velocity profiles shown in Figs. 7.5a and b, the neutral velocity contours for the vacuum case show a typical supersonic expansion. The streamlines are unevenly distributed due the fact that the thermal speeds are comparable to the directed velocity so that the expansion angle for the neutrals is large. However, in the test chamber, Fig. 7.5a, the result is different and it can be seen that after the thruster exit, a deceleration occurs as opposed to the vacuum case because the flow is essentially confined by the chamber walls and stagnates near the boundaries. The small velocity
induced near the pumps can be seen in the streamlines in the right upper and lower corners of Fig. 7.5a.

Figures 7.6a and b show the ion velocity contours for the baseline and vacuum cases, respectively. Although there is a slight difference in the velocity magnitudes outside the core region of the ion plume, there is no distinctive difference in the shape of the plume or in the streamline structures. One thing to notice is that for the baseline case, there is a deceleration zone near the x/D=10, z/D=5 location. This is due to the fact that near the wall region there are more neutrals that the ions can collide with and undergo CEX.

Examination of the spatial distribution of the CEX ions in Figs. 7.7a and b however shows a significant difference for the two cases. In the vacuum case, one can observe that the CEX ion population is greatest near the thruster exit, reaching a value of approximately 10% of the total ion number density. This is expected since the densities of neutrals and ions and therefore the collision frequency is highest in this region. As the distance from the thruster exit increases, the CEX ion population decreases rapidly. For the baseline case, the presence of the neutral background density increases the CEX ion population as seen in the increase of green and blue regions in Fig. 7.7a. This result suggests that due to the increase in the neutral background density levels, the CEX ion production is enlarged in a test chamber compared to what it would be in space, a phenomena which is also reported in Van Gilder [29] and Bondar [33]. Moreover, the CEX spatial distribution resembles the CEX wing structures reported in previous work [15, 18, 21] where CEX ions were observed to expand into donut-like shapes.
Although the fluctuations seen in Fig. 7.7a near x/D=6 and z/D=2 are very small compared to the overall neutral and ion density values, especially near the thruster exit, it is important to understand their origin to resolve the true spatial distribution. Since the CEX ions are the outcome of neutral and ion collisions, the most obvious cause for these fluctuations is related to fluctuations in the number of collisions in the plume. The number of possible collisions, \( N_{max} \), in a cell of volume \( V_c \) during timestep \( \Delta t \) for neutral and ion species is given by

\[
N_{max} = N_i \times N_n \times max(W_i, W_n) \times (g\sigma_T)_{max} \times F_{num} \times \Delta t/V_c \quad (7.1)
\]

where \( N_i \) and \( N_n \) are the number of computational ion and neutral particles, \( W_i \) and \( W_n \) are the weighting factors for ions and neutrals, \( (g\sigma_T)_{max} \) is the maximum product of relative velocity between the particles and the total cross-section of each collision type, and \( F_{num} \) is the value representing the number of real neutral particles represented by a single computational particle. If the additional structure seen in Fig. 7.7a is noise in the CEX ion production rate, it can be reduced by using averaged neutral and ion number densities which modifies Eq. 7.1 as,

\[
N_{max} = \overline{N_i} \times \overline{N_n} \times max(W_i, W_n) \times (g\sigma_T)_{max} \times F_{num} \times \Delta t/V_c \quad (7.2)
\]

where \( \overline{N_i} \) and \( \overline{N_n} \) represents the averaged number of computational particles present in the corresponding cell. The averages are obtained by summing over 120,000 timesteps during sampling which is the same value used for the baseline case. Once \( N_{max} \) is
calculated, a pair of neutral and ion particles is selected to test if a collision would actually occur using the acceptance-rejection method based on the ratio \( \frac{g \sigma_T}{(g \sigma_T)_{max}} \) where the numerator represents the product of the current pair’s relative velocity and total cross-section and the denominator represents the maximum value of this product encountered throughout the simulation for each cell. If it is accepted, then the probability that the outcome of the collision is MEX or CEX is evaluated by generating a random number between 0 and 1. If the random number is less than or equal to the probability of an MEX collision, only the momentum (and energy) is transferred between the charged and neutral particles. Otherwise, a CEX process is assumed to occur wherein the velocities of the two colliding particles do not change but their identity (charge to neutral and vice versa) occurs. The selection of pairs of particles described above as well as the evaluation of MEX and CEX processes is repeated \( N_{max} \) times in each cell per timestep. Using this modified approach in the collision routine, the CEX spatial distributions are presented in Fig. 7.8 for the baseline and vacuum cases. The noise that appeared in Fig. 7.7a outside the core region is seen to be removed in Fig. 7.8a however Fig. 7.8b shows that the vacuum solution does not change significantly. The reason for this is due to the fact that the vacuum case has sufficiently low neutral and ion number density at the locations where noise might occur which results in essentially no CEX ion production. For the baseline case, although the ion number density is still small, the neutral number density is large enough to result in CEX ion production caused by the confinement of the gas due to the chamber. These two additional simulations show however that the larger scale structures present in the chamber and vacuum solutions persist. Based on this finding, the subsequent figures which show the CEX ion density results will use Eq. 7.2,
a quasi-overlay approach. A smaller pump, partially specular and triple thruster cases which will be discussed in the next section were also run using Eq. 7.2 but for 360,000 further timesteps to better resolve the spatial distribution of CEX ions.

A comparison was made with respect to the UK-10 engine [25] in order to anchor our approach and make further chamber predictions in the next section. Table 7.1 lists the total computational particles, total collision cells and the simulated particle to real particle ratio used for this simulation, where the rest of the numerical parameters such as timestep and weighting factors as well as the computational domain are the same as given at the beginning of Sec. 7.1. The operating conditions are also given in Table 7.2. Based on the plasma conditions and the measured background pressure, the spatial variance of CEX ions in the chamber was simulated. An extra simulation was also run in a vacuum without any background pressure to observe the difference between the two conditions. Van Gilder [29] also performed numerical simulations to compare the ion thruster plumes at these two conditions. Good agreement in the CEX ion distribution in the domain can be seen in Fig. 7.9 where the top and bottom portions correspond to the chamber and vacuum conditions, respectively. Both solutions show that the CEX number density is essentially the same in the vicinity of the thruster exit, however the chamber solutions predict larger CEX ion presence at locations further away from the thruster exit, a trend which is also seen in the previous results of this section.

7.2 Sensitivity of test chamber ion plumes to operational factors

We begin this section by first validating the implementation of the thruster exit condition used in SUGAR and then focus on the effect of the background neutral density
Fig. 7.3. Neutral number density contour lines for the study of the effect of domain boundary conditions on the thruster plume at y/D=5
Fig. 7.4. Ion number density profiles for the study of the effect of domain boundary conditions on the thruster plume at y/D=5
Fig. 7.5. Neutral velocity contours for the study of the effect of domain boundary conditions on the thruster plume at y/D=5
Fig. 7.6. Ion velocity contours for the study of the effect of domain boundary conditions on the thruster plume at y/D=5.
Fig. 7.7. CEX ion number density contour lines for the study of the effect of domain boundary conditions on the thruster plume at y/D=5

(a) Baseline

(b) Vacuum
Fig. 7.8. CEX ion number density contour lines for $y/D=5$ using an overlay approach
Fig. 7.9. CEX ion population comparison, top represents chamber and bottom space-vacuum conditions
levels, chamber size, wall models and number of thrusters on the plume structure within a test chamber.

### 7.2.1 Modeling of T6 Engine Inlet

Snyder et al. [13] studied the performance of the T6 engine in terms of beam current, thrust, as well as secondary effects such as beam divergence and thrust vector orientation. Table 7.1 lists the total computational particles, total collision cells and the simulated particle to real particle ratio used for this simulation where the rest of the numerical parameters as well as the computational domain are the same as given at the beginning of Sec. 7.1. The plasma conditions for throttle level 3 were used with the exit conditions summarized in Table 7.2. The beam current and thrust variance along the x-axis are shown in Fig. 7.10 where a variation of 3% and 16% can be observed, respectively. The beam current and thrust values are calculated by adding up the contributions from the V-mesh cells at steady-state. Since the thruster exit is at x/D=1.875, as seen in Fig 7.1, the calculated beam current and thrust of 1.85 A and 123 mN at this exit location shown in Fig. 7.10 agree well with the experimental findings of 1.84A and 123 mN.

### 7.2.2 Background Neutral Level

We start with the effect of the neutral background level which is varied by changing the pump cross section with respect to the baseline case for the conditions given in Table 7.3. The larger pump case is simulated by increasing the pump cross-section by four compared to the baseline case whereas the smaller pump case is similarly simulated
Fig. 7.10. Plume data
by decreasing the pump cross-section by four compared to the baseline case. Figure 7.11a shows that the increase in the pump size effectively decreases the background neutral levels compared to the baseline case (Fig. 7.3a). Moreover, Fig. 7.11b shows that the decrease in the pump cross-section results in an increase in the neutral background level which verifies that the pump cross-section works as a numerical means of control for the background level in the test chamber. It can be seen that the $5.0 \times 10^{16} \text{ m}^{-3}$ isoline of Fig. 7.11a is located much nearer to the thruster exit for the larger pump case as compared to Fig. 7.3a. This result shows that the background neutral level is decreased compared to the baseline case and also has a closer resemblance to the vacuum case shown in Fig. 7.3b. For the smaller pump case shown in Fig. 7.11b, it can be observed that the background neutral level increased to approximately $1.5 \times 10^{17} \text{ m}^{-3}$. One can also see that the yellow region seen in Fig. 7.3a for the baseline case has expanded for the smaller pump case.

Table 7.3. Simulation Domain and Background Conditions for different Pump Sizes$^a$

<table>
<thead>
<tr>
<th>Pump cross-section ($\text{m}^2$)</th>
<th>Baseline</th>
<th>Larger Pump</th>
<th>Smaller Pump</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n(\text{m}^{-3})$, Xe, at thruster exit</td>
<td>$4.6 \times 10^{17}$</td>
<td>$4.6 \times 10^{17}$</td>
<td>$4.6 \times 10^{17}$</td>
</tr>
<tr>
<td>$n(\text{m}^{-3})$, Xe$^+$, at thruster exit</td>
<td>$2.3 \times 10^{15}$</td>
<td>$2.3 \times 10^{15}$</td>
<td>$2.3 \times 10^{15}$</td>
</tr>
<tr>
<td>$n(\text{m}^{-3})$, Xe$_{\text{back}}$</td>
<td>$6.0 \times 10^{16}$</td>
<td>$2.0 \times 10^{16}$</td>
<td>$1.5 \times 10^{17}$</td>
</tr>
</tbody>
</table>

$^a$ Domain Size is 1.0 m$^3$
It is expected that an increase in the background neutral level should result in a higher CEX ion population since the collision rate is enhanced. The CEX ion population for the larger pump case seen in Fig. 7.12a shows that the spatial distribution is similar in the vicinity of the thruster exit (i.e. \( x/D=3, z/D=5 \)) to the baseline case result in Fig. 7.7a. The main difference between the two results is the decrease in the population near \( x/D=7, z/D=5 \) region in Fig. 7.12a due to the decrease in the background neutral density for the larger pump case. Also, the CEX ion population near the chamber wall at \( x/D=10, z/D=5 \) is not changed significantly which implies that it is dominated by the presence of the chamber wall. For the comparison with the smaller pump case, as discussed previously, the average background neutral density is two-three times higher than the baseline case. For this reason, the CEX ion population for the smaller pump case shown in Fig. 7.12b is overall greater than baseline case. Similar to the larger pump case, the CEX ion population is essentially the same in the vicinity of the thruster exit compared to the baseline case. An important difference that can be observed is that the green region which represents the \( 2.0 \times 10^{12} \text{ m}^{-3} \) level is much more expanded especially towards the lateral chamber walls \((z/D=0 \text{ and } z/D=10)\). Moreover, a local peak, represented by the yellow isoline, is observed for smaller pump case in the vicinity of the wall near \( x/D=10, z/D=5 \) location. This result implies that the CEX ion production is enhanced near the wall as discussed previously.

### 7.2.3 Effect of Chamber Size

Next we focus on the effect of the chamber size where a smaller chamber domain of \((0.5 \text{ m})^3\) using the baseline pump size is modeled. The motivation to perform such
Fig. 7.11. Neutral number density contour lines for the study of the effect of background pressure on the chamber simulations at $y/D=5$
Fig. 7.12. CEX ion number density contour lines for the study of the effect of background pressure on the chamber simulations at $y/D=5$ using an overlay approach.
a study was to observe the effect of smaller test chambers which are usually the most cost-effective options for universities or small sized institutions.

Figures 7.13a and b show the neutral number density contours for the baseline and smaller chamber cases. The main difference in the results is the larger neutral density level for the smaller chamber case farther away from the thruster exit. The green region represented by the $2.0 \times 10^{17} \text{ m}^{-3}$ isoline is much larger and expands close to the pump locations for the smaller chamber case compared to the baseline case. With the pump cross-sections the same, the flow is more confined for a smaller chamber.

Looking at the ion number density contours in Fig. 7.14, one can observe that the results are essentially the same for the baseline and smaller chamber cases. The comparison helps to make the point that the total ion number density profile is not susceptible to change when the background neutral density level or the test chamber size is changed. On the other hand, examination of the CEX ion population in Fig. 7.15 shows an obvious difference near the wall ($x/D=5$, $z/D=5$) where the CEX ion population is higher for the smaller chamber case. This increase is also apparent in the local higher level of cell refinement. The enlargement of the green regions in Fig. 7.15b compared to Fig. 7.15a is also consistent with the trend seen in Fig. 7.13.

### 7.2.4 Effect of Wall Model

This subsection studies the effect of the gas-surface interaction model for particle-wall collisions using the baseline pump size. While the baseline case used an accommodation coefficient of $\alpha = 1.0$, an additional simulation was run using $\alpha = 0.9$ where the implementation details are given in Sec. 3.3. The selection of taking $\alpha = 0.9$ is based on
Fig. 7.13. Neutral density line levels for the study of the effect of domain size on the chamber simulations at $y/D=5$. 

(a) Baseline 
(b) Smaller chamber
Fig. 7.14. Ion number density contours for the study of the effect of domain size on the chamber simulations at $y/D=5$
Fig. 7.15. CEX ion density line levels for the study of the effect of domain size on the chamber simulations at $y/D=5$ using an overlay approach.
Hofer [28] where it has been discussed that although a fully diffuse model is adequate for engineering problems, the particle-wall interaction physics is sufficiently complex that a value as low as $\alpha = 0.8$ should be used to check the sensitivity of the results.

A comparison of neutral density and velocity between the two cases shows practically no difference as expected, however, for the ion species, some considerable changes are observed. Figure 7.16a shows the ion velocity contours superimposed with streamlines for the partially specular case. The most obvious difference between the two cases (Fig. 7.6a and Fig. 7.16a) is in the velocity contours seen in Fig. 7.16a below the $z/D=2$ and to the right of the $x/D=8$ line. Effectively with the introduction of the specular wall model, a fraction of reflected ions conserve their pre-collisional speeds after reflection from the wall. Moreover, near the wall which faces the plume directly (the rectangle bounded by $8<x/D<10$ and $0<z/D<10$), the streamlines of the local ion velocity field are observed to change. In addition to the velocity field, changes are also observed in the CEX ion population, as shown in Fig. 7.16b caused by the difference in the wall model accommodation coefficient. Due to this change, an increase in the CEX ion population is observed for the partial specular case in the regions near $x/D=7$, $z/D=1$ and $x/D=7$, $z/D=9$ compared to the baseline case as shown in Fig.7.8a. Note that in the baseline case, Fig.7.8a, the yellow region of enhanced CEX ion production is not present.

To understand the difference in the spatial CEX ion number densities discussed above we further examined the origins and history of how the CEX ions were created in the partially specular case. The spatial distribution of the average energy of CEX ions are shown in Figs. 7.17a and b for baseline and partially specular cases, respectively. The CEX ion energy distributions and streamlines can be seen to be essentially the same in
Fig. 7.16. Partial specular simulation results for the study of the effect of wall model at y/D=5
the vicinity of the thruster exits for both cases but a difference can be discerned near the chamber walls at $x/D = 4, z/D = 9.5$ and $x/D = 4, z/D = 0.5$. In order to study this further, a rectangle bounded by $3.6 < x/D < 4.0$ and $9.3 < z/D < 9.7$ was selected (and is denoted by the pink box shown in Fig. 7.17a) to collect ions that have undergone CEX collisions during the sampling portion of the DSMC simulation, i.e., at steady state, for the full and partially accommodated cases. Figure 7.17c shows the energy distributions for the two cases where the energies are normalized by 7 eV and approximately 200,000 computational particles were used to compute the distributions. The energy distributions are in agreement with the spatial energy distributions shown in Figs. 7.17a and b where the baseline case is observed to be at a slightly larger average energy compared to the partially specular case. At this point, we hypothesized that this difference in the average energy was not caused by a local phenomenon but rather was related to the possibility that there are two distinct mechanisms for CEX ion formation in the partially specular case. To test this idea, the same rectangular box was used to collect ions that had not undergone CEX collisions. The examination of this second set of results showed that the partially specular case has particles that have specularly reflected from the chamber wall near $x/D = 10, z/D = 7$ and $x/D = 10, z/D = 3$ locations that then continue to travel to the location of the collection box region, whereas, the baseline case has no such particles. The pathline of two such computational particles from the partially specular case are shown in Fig. 7.17b denoted by pink dashed lines. It should be noted that these specularly reflected beam ions that reach the collecting region represent a very small fraction of the original thruster exit ions of approximately one out of 10,000 particles. However, they do represent 30% of the ions in the collecting region box which have not undergone
CEX collisions. Therefore, we conclude that these ions, which subsequently collide with the local neutrals in the collecting box region, are responsible for the difference in the spatial CEX number density and energy distribution for the partially specular reflecting case. Moreover, the AMR octree computational technique is able to resolve these subtle, although potentially important, differences between the two cases.

7.2.5 Effect of Multiple Thruster

The configuration of the triple thrusters can be seen in Fig. 7.1c while the rest of the computational parameters such as timesteps, weighting factors, sampling times and physical conditions such as number density and velocity at the thruster exit plane are kept the same as the single thruster case. Starting with the density fields, Fig. 7.18a shows the neutral number density contours. The most obvious change compared to single thruster case is the asymmetry of the flow which is caused by the interaction of the plumes with each other with respect to baseline case results shown in Fig. 7.3a. One can also observe the expansion of the yellow region to the majority of the plume compared to a much smaller size for the baseline case. This result also implies that for the same domain and pump size, the background neutral level has increased due to the addition of two thrusters. Figure 7.18b shows the comparison of the neutral density levels along the centerline of the single thruster and the top thruster in the triple thruster configuration shown in Fig. 7.1c inside the chamber as well as the triple thruster configuration in vacuum. It can be seen that up until one diameter away from the thruster exit, the difference between the three profiles are within 10-20% of each other. However beyond
Fig. 7.17. CEX energy distribution comparisons for baseline and partially specular case
this location, the difference between single and triple thruster solutions in the chamber approaches a factor of three.

The spatial distribution of the ion number density is shown in Fig. 7.19. The total ion number density field can be seen in Fig. 7.19a which shows a much more distinct plume interaction region compared to the neutrals. Figure 7.19b shows a comparison along the centerlines of the single (chamber) and the top thrusters (chamber and vacuum) where it can be seen that upstream of one diameter from the thruster exit, there is essentially no difference. Downstream of this point, the chamber triple thruster solution starts to deviate from the other two solutions because of the increase in the background neutral level in the chamber which causes a higher CEX collision frequency. Therefore, the total ion number density increases even without taking into account the effect of the plume interaction. However, downstream of the x/D=5.5 location, a sharp increase in both of the vacuum and chamber triple thruster solutions is observed due to the plume interaction. In addition to this, differences of up to three times can be observed in the far downstream region between triple thruster solutions for both chamber and vacuum cases and single thruster solutions. An interesting difference can be observed in the CEX ion distribution between the single and triple thruster cases. Compared to Fig. 7.7a for a single thruster, a large change can be observed in Fig. 7.19c in the spatial region where the $2.0 \times 10^{14} \text{ m}^{-3}$ level has significantly expanded. This result is in agreement with the trend shown in Fig. 7.18b which is directly caused by the increase in the background neutral density. Moreover, a comparison between chamber and vacuum solutions shows that although an asymmetry is present for the vacuum CEX ion distribution due to plume interactions as shown in Fig. 7.19d, the magnitude of the CEX ion population for
the chamber case is approximately an order of magnitude higher for the majority of the domain except in the near vicinity of the thruster exit.

Finally, the velocity fields within the chamber for the triple thruster configuration is shown. The neutral velocity field shown in Fig. 7.20a is asymmetric compared to the symmetric result of the single thruster case in Fig. 7.5a while the streamlines are also pointing towards the pump locations. For the ions, Fig. 7.20b shows that the plume interaction is also evident and the streamlines illustrate that there is a backflow of ions in an asymmetric way.
Fig. 7.18. Neutral number density profiles for triple thruster simulation at y/D=5
Fig. 7.19. Ion number density profiles for triple thruster simulation at $y/D=5$
Fig. 7.20. Velocity contours superimposed with streamlines for triple thruster simulation at $y/D=5$. 

(a) Neutrals 

(b) Ions
Chapter 8

Conclusions

The application of combined DSMC/PIC simulations based on adaptive mesh refinement with octree grids to ion thruster plumes has been demonstrated using a newly developed software called SUGAR. A timestep algorithm based on the combined algorithms of Serikov [67] and Kannenberg [68] was implemented to effectively model two-way coupling between neutral and ion species caused primarily by charge exchange reactions. Parallelization of DSMC/PIC calculations on these three grids is more complex than a single-grid implementation, but necessary because of the large difference in length scales for collisions and electric field. Nevertheless, the use of three AMR/octree grids for collisions, electric field, and visualization demonstrated that the AMR/octree approach could more than adequately resolve the local Knudsen and Debye lengths. Parallelization was achieved by using two-dimensional blocking during domain decomposition and found to give approximately 100 times speed-up when 256 MPI ranks were used. A comparison study for a three-dimensional, multi-plume configuration of neutral species obtained with the AMR/octree simulation versus a two-level Cartesian grid was done. Up to several diameters away from the exit of the plumes, both results agreed well. Downstream of this location, due to having fewer number of particles per cell, the two-level Cartesian result started to fluctuate. Moreover, AMR/octree simulation used much smaller number of computational particles and computational cells where the simulation ran faster.
Results for single- and triple-thruster cases are presented and evaluated for the relative importance of elastic and CEX collisions and the effect of the electric field on the ion plume structure. In general, it was found that neutrals diverge from the bulk flow due to thermal effects and ions, in contrast, maintain a more collimated structure due to their initial high momentum along the thruster axis. The CEX collisions were shown to produce back flow in both single- and triple-thruster configurations and increase the ion number density up to 10% near the thruster exit. The two-way coupling causes the neutral velocity profile to vary by up to 15% difference compared to the overlay assumption and a strong deceleration zone near the thruster exit was observed in the ion velocity profile where the collision frequency is a maximum. When collisions and electric field are modeled simultaneously, the neutral and ion number densities of three-thruster configuration are up to four times different compared to the single-thruster case at downstream.

It was found that the back flow of ions is strongly enhanced when the induced electric field is considered and there is almost three orders of magnitude increase in the ion number density compared to the case when only MEX and CEX collisions are present. Ion energy distributions in the back flow region were obtained and were found to match a shifted Maxwellian distribution. A study of the angular distribution of neutral and ion fluxes five diameters from the thruster exit was performed and it was found that the flux of ions in the back flow region was almost five orders of magnitude lower than the flux directly facing the thruster. In addition to the single-thruster configuration, a triple-thruster case was simulated and the AMR capability was shown to smoothly capture the interaction between the thruster plumes. The neutral and ion number density levels
exhibit a linear trend with respect to the number of thrusters, whereas the ion velocity is roughly constant in the back flow region. The effect of the differentially charged solar panels with respect to the ambient space environment was modeled to investigate its effect on the back flow. A commercial software was used to obtain the solution for the boundary condition problem combined with the induced electric field due to the plasma for a single-thruster configuration. It was found that the back flow of ions increases both in terms of number density and velocity up to a few times in the back flow region when gas surface interactions are not considered. When gas surface interactions are modeled with neutralization of ions that impact the solar panels and accommodation to the solar panel temperature, it is observed that the ion number density and velocity of ions is dominated by these interactions regardless of the solar panel voltage for the range of values that was studied.

Finally, ion thruster plumes were simulated for a vacuum and chamber configurations. The proposed wall and pump model resulted in a realistic representation of the background pressures reported in the experiments. The simulations conducted in chambers were used to characterize the changes in the neutral and CEX ion distributions compared to the vacuum simulations. Both vacuum and chamber simulations showed that CEX ions are approximately 10% of total ions for both cases in the vicinity of the thruster exit. The velocity profiles that were superimposed with streamlines showed that the neutral velocity distribution for the chamber simulation has changed immensely compared to vacuum where a much stagnant gas is observed. The ion velocity profiles are much more similar, whereas a deceleration zone for the chamber solution is observed
at one diameter away from the chamber wall. By changing the effective pump cross-section as a means to control the background pressure level in the chamber, the CEX ion distribution was shown to change drastically approximately two diameters away from the thruster exit. The maintained background pressures are shown to match the reported values from experiments, which are a few to several microTorr. The use of a smaller chamber size resulted in a change in CEX ion number density near the plume-facing wall where AMR capability was shown to capture this trend. Finally, a triple-thruster configuration was used to investigate the changes on the flow field compared to a single-thruster configuration. Plume interactions were observed for both neutrals and ions. A strong interaction in the form of a sharp increase in the ion number density at four diameters away is predicted for the thrusters for both chamber and vacuum solutions. Moreover, the CEX ion number density distribution significantly changed for the triple-thruster case compared to the single-thruster case when the chamber configuration was used. Based on the methodologies proposed for simulations of ground chamber configurations, the quantification of the background levels can be used to correlate the results to the actual vacuum conditions, not only for single-thruster configurations, but also multiple array configurations.

The developed software and the discussion in this dissertation are a contribution to modeling of rarefied flows with the usage of HPC. Although, physical models used in this work are based on recent work, enhancements are always possible such as the physical modeling of the collisions and plasma. The collision models should be revisited in order to focus on secondary effects by including a larger chemistry set that would take into account the electron collisions or other trace species. The plasma modeling
can also be extended so that the quasi-neutrality assumption can be relaxed as well as modeling of electrons explicitly as computational particles. Moreover, the immersed-body modeling capability of SUGAR can help to study other applications such as flow over re-entry vehicles, MEMS devices, or plasma flows where the DSMC-PIC methods are applicable. Although the use of high performance computing was shown to tackle computational problems that were assumed to be intractable a decade or more ago, the advent of exa-scale computing is expected to bring additional considerations to current approaches. The heterogeneity of the modern hardware on current and future supercomputers will definitely leave room for exploitation of the distributed memory implementation while a simultaneous implementation of shared memory approaches or node-attached accelerators. The usage of modern software development approaches such as using object-oriented programming or making use of external libraries are vital to maintain and extend the capabilities of such software so that the life-cycle is increased. The ability to conduct numerical simulations in motivation to support or even partially replace the costly experiments has become increasingly important; this dissertation is an important starting point for some applications that are vital to aerospace field.
Appendix A

Appendix
SUGAR Software

**Scalable Unstructured Gas-dynamics with Adaptive Refinement**

**Version 1.0**

**Developed by**

Burak Korkut
Deborah A. Levin
Zheng Li
Saurabh Sawant
Ozgur Tumuklu

**At**

The Pennsylvania State University

University of Illinois at Urbana-Champaign
A Introduction

A.1 What is SUGAR?

SUGAR (Scalable Unstructured Gas-dynamics with Adaptive Refinement) is a Direct Simulation Monte Carlo (DSMC) - Particle in Cell (PIC) code that models gases using collisions, chemistry, electric field effects and gas-surface interactions. It uses an octree adaptive refinement mesh (AMR) to track, group and list computational particles and output flow variables in three dimensions. Physical surfaces embedded in the computational domain are represented by triangulated surfaces.

- Runs with a single processor or in parallel using MPI on distributed memory platforms.
- Domain decomposition of the computational domain
- Written in C++ with OOP
- Easy to maintain and extend with new functionality
- Runs from input XML files
- Python scripts for the automation of the simulation workflow
- Uses open-source libraries
- Three dimensional representation of the computational domain
- Employs octree AMR meshes
- Specular of diffuse gas-surface interaction
• Species weighting factors

• Static and dynamic load-balancing of the computational domain

• Support for VTK and Tecplot output formats

### A.2 Conventions

A right handed coordinate system is being used in SUGAR. An underlying cartesian mesh is used to accomplish MPI communication over the MPI ranks where cell numbering starts from z axis then y axis then to x axis. In addition, there is no normalization being used in SUGAR. The units for all variables being printed out or outputted are in International System of Units (SI).

### A.3 Problem Reporting

To report any problems/bugs or ask questions, please contact Dr. Deborah Levin (deblevin@illinois.edu) or Mr. Burak Korkut (brkorkut@yahoo.com).
B Framework

B.1 Platforms

SUGAR has been used developed, debugged and ran on four UNIX/LINUX clusters to date, namely PSU Lion-X, AFRL Spirit, ERDC Garnet and NCSA BlueWaters. There is no foreseen problem that will prevent compiling and running SUGAR on other distributed memory platforms.

B.2 Installing dependent libraries

SUGAR makes use of several external libraries. First one is Pugixml which is used to parse and read the XML input files. Pugixml source files are compiled from source files while compiling SUGAR’s main executable. Therefore, there is no need for installation. Second library is VTK which is used for outputting result files to be read by Paraview and VisIt visualization software. VTK (Version 6) needs to be installed on the platform that is being worked on. The library path should be specified in the Makefile.

Third library is Parmetis which is used as a graph partitioner to predict a load-balanced processor topology. Similar to VTK, Parmetis (Version 4) and Metis (Version 5) needs to be installed on the platform that is being worked on. The library path should be specified in the Makefile.

B.3 Compilation

The executable that is obtained by compiling the source files is called "maestro.x". The compilation of roughly 20 source files are accomplished via a Makefile. However,
in order to automate the compilation procedure which involves creating the executable with the user defined compilation flags and moving the executable to the proper folder, a python script was developed. This python script is called "do.py" and located in the binary folder.

B.4 Compilers

On the platforms that SUGAR has been used, Intel and GNU compilers were used.

B.5 Scripts

There are two main scripts being used in SUGAR which are developed in Python. First one is called "do.py" which used to create new cases and compile the source code and create the main executable "maestro.x". The second one is called "post.py" which is used to create a final tecplot file from the partitioned files so that Tecplot can be used to visualize the results. The detailed explanation of these scripts is given in the next section.
C Main modules

C.1 General Workflow

After downloading/copying to the working folder, user must verify that there is a python file named "do.py" and the folder "src" which holds the source files. A full procedure of a simulation in SUGAR can be described as follows:

1. For a specific case/simulation, the user should first select a proper case name (for example "rocket") which should consists of only lower case letters and no special characters like ",", ";", ";", etc.").

2. User should make sure the directory level is the same as do.py and src. Then, the command

   $ python do.py -c rocket

   should be given on the terminal. This will create three folders named "bin_rocket", "in_rocket" and "out_rocket" for the executable, input files and output files, respectively.

3. User should then switch the directory to "in_rocket" which is the input folder.

   There are three input files to be set;

   - in_rocket/main.xml is the file where user sets the compilation flags, computational domain size and the number of cells in each direction for the underlying Cartesian mesh.
• in_rocket/kinetic/dsmc.xml is the file where the main computational parameters as well as DSMC parameters are defined (e.g. FNUM, timestep, flow conditions etc.)

• in_rocket/pic.xml is the file where PIC related parameters are defined.

4. After setting the input values in the input folder, user should switch to the directory “bin_rocket”. The executable ”maestro.x” is ready to be created at this point. The command

$ python do.py -m

should be given on the terminal. This command will invoke a script that creates the executable ”maestro.x”. Specifically, this command will look for the compilation flags from ”in_rocket/main.xml”, execute ”src/Makefile” with these flags and move the executable ”maestro.x” to the current directory.

5. Since ”maestro.x” is now created, the simulation is ready to be started. User can now submit ”maestro.x” with the preferred pbs job script. A typical run command could be

$ aprun -n 2048 ./maestro.x > my_prog.out

which is included in the pbs script.

6. Once the simulation finishes, the results file could be located in ”out_rocket/Results”.

7. In order to visualize the results, user have two options;
• Visualizing with Paraview/VisIt (Fig. A.1); Copy the *vtu files to the desired directory where Paraview or VisIt will be used. For example, for a simulation result at 100,000 timestep and with 1024 processors, user should copy 1024 .vtu files named "SUGAR_Output_100000_i.vtu" where "i" stands for the processor rank and "SUGAR_Output_100000.pvtu" file which is used to merge these files on the fly. User should just open "SUGAR_Output_100000.pvtu" file in Paraview/VisIt and then do the post-processing accordingly.

• Visualizing with Tecplot (Fig. A.2); Tecplot files needs an additional post-processing step. The partitioned tecplot files generated by each processor must be merged using a python script called "post.py" which is located in "out_rocket/Results". For example, for a simulation result at 100,000 timestep and with 1024 processors, user should invoke the command

$ python post.py keep 100000 1024

which will create the file "SUGAR_Tecplot_Output_100000.tec". At this point, user should open the merged file which is called "SUGAR_Tecplot_Output_100000.tec" in Tecplot and do post-processing accordingly.

C.2 Screen Output

While the simulation is running, a typical console output seen by the user is given as follows;

Communication is alive.
Fig. A.1. Visualization with Paraview
Fig. A.2. Visualization with Tecplot
SUGAR (Scalable Unstructured Gas-Dynamics with Adaptive Refinement) - Master process:

The number of processes is 2048

Read casename is str

All input files are read and broadcasted.

Arrays are allocated.

MPI grid is generated.

Verification 1 left 0 remaining 11494 total 11494 ghosts 0
Verification 2 left 0 remaining 22988 total 22988 ghosts 0
Verification 3 left 0 remaining 34482 total 34482 ghosts 0
Verification 4 left 0 remaining 45976 total 45976 ghosts 0
Verification 5 left 0 remaining 57470 total 57470 ghosts 0
Verification 6 left 0 remaining 68964 total 68964 ghosts 0
Verification 7 left 0 remaining 80458 total 80458 ghosts 0
Verification 8 left 0 remaining 91952 total 91952 ghosts 0
Verification 9 left 0 remaining 103446 total 103446 ghosts 0

........................................

Verification 14995 left 161116419 remaining 11236111 total 172352530 ghosts 0
Verification 14996 left 161128222 remaining 11235802 total 172364024 ghosts 0
Verification 14997 left 161139962 remaining 11235556 total 172375518 ghosts 0
Verification 14998 left 161151405 remaining 11235607 total 172387012 ghosts 0
Verification 14999 left 161162697 remaining 11235809 total 172398506 ghosts 0
BEGINNING of STEP 15000
Verification 15000 left 161174025 remaining 11235975 total 172410000 ghosts 0
Arrays are deallocated.
Communication is dead.

SUGAR:

Normal end of execution.

29 March 2015 02:06:58 PM

Elapsed wall clock time since program start = 2066.413508 seconds.
Elapsed wall clock time since adaptation started = 1628.218862 seconds.
Elapsed wall clock time since sampling started = 937.124563 seconds.

User is given the information of how many MPI ranks are being used, at each timestep how many computational particles are in the computational domain. At the end of the simulation, three timings which correspond to the time since the simulation started, since the adaptation of the mesh started and since the sampling started are outputted, respectively.
C.3 Performance and Scalability

To date, SUGAR has been mostly used with 1000 MPI ranks for production runs. Fig. A.3 shows the speed-up results for a strong scaling type study.

C.4 Restart Capability

In order to use the restart capability of SUGAR, a simulation must be started with the variable ”restartCapa” defined in ”in_rocket/kinetic/dsmc.xml” should be equal to 1. Assuming an error occurred (platform restart, power issues etc.), the simulation can be continued from the checkpoints in the sampling stage. In order to operate in restart mode user should set the variable ”restartMode” defined in ”in_rocket/kinetic/dsmc.xml” to 1 and the ”restartStartStep” defined in ”in_rocket/kinetic/dsmc.xml” to the last checkpoint timestep.

C.5 Module List

At this point, a convention followed in SUGAR should be explained. There are roughly 15 folders that accommodate relevant source files (.cpp and .h) to the folder name. Each folder has a corresponding .h and .cpp file that is used to interface with the files from the other folders. For example ”allocation” folder has ”allocation.cpp” and ”allocation.h” files. Moreover, all the source files whose names start with ”def” hold class and function definitions and used by the implementation files. A schematic of how the modules are interconnected with each other is shown in Fig. A.4.

- main.cpp and main.h => Higher level function definitions such as ”allocate arrays”, ”start simulation” are defined in these files.
Speed-up

Fig. A.3. Speed-up

Module Interconnection Diagram

Fig. A.4. Module Interconnection Diagram
• **allocation** => Global arrays that are required by the simulation are defined, allocated and deallocated in this module.

  - allocation.cpp
  - allocation.h

• **bc** => Boundary conditions routines are defined in this module.

  - bc.cpp
  - def_bc.cpp
  - def_bc.h

• **chemistry** => Collisions for the computational particles are defined in this module. Collision Cells are passed to the main function and collision routines are accomplished.

  - def_dsmc.cpp
  - def_dsmc.h
  - dsmc.cpp
  - dsmc.h

• **communication** => MPI framework is defined in this module. Communication of particles, partitioning of Cartesian cells, load-balancing are accomplished.

  - comm.cpp
  - comm.h
  - def_comm.cpp
- def comm.h
- def list.cpp
- def list.h
- def partition.cpp
- def partition.h
- def processor.cpp
- def processor.h

- **electromagnetics** => The electric field cells are passed to this routine, where PIC routines are accomplished and the electrostatical forces are calculated on the charged particles.
  - def electromagnetics.cpp
  - def electromagnetics.h
  - electromagnetics.cpp
  - electromagnetics.h

- **geometry** => Cell, Mesh definitions, refinement procedures, particle mapping routines are described in this module.
  - def cell.cpp
  - def cell.h
  - def grid.cpp
  - def grid.h
- geometry.cpp
- geometry.h

- global
  - global.cpp
  - global.h

- initializer
  - def_initializer.cpp
  - def_initializer.h
  - initializer.cpp
  - initializer.h

- input
  - def_input.cpp
  - def_input.h
  - input.cpp
  - input.h
  - pugixml.cpp
  - pugixml.hpp
  - pugiconfig.hpp
• lagrangian => This module holds the particle definitions.
  
  - def_particle.cpp
  - def_particle.h
  - species.h

• Makefile => The makefile that hold the compilation and dependency directives for the main executable "maestro.x"

• output => This module has the definitions for the output generation classes. This classes support .vtk and .tec file generation.
  
  - def_output.cpp
  - def_output.h
  - output.cpp
  - output.h
  - liberOut.cpp
  - liberOut.h
  - liberVTK.cpp
  - liberVTK.h

• solver => This module holds the higher level definitions of a kinetic simulation such particle movement, time loop etc. Moreover, higher level calls to particle mapping, output file generation are initiated in this module.
  
  - def_solver.cpp
- `def_solver.h`
- `solver.cpp`
- `solver.h`

- **templates** => This module includes the default input and executable related files that is initialized with the creation of cases.

### C.6 Python scripts

#### C.6.1 do.py

The python script "do.py" which is used to create cases and executable is as follows;

```python
import os, time, sys
from os import system, remove, path

# Option to create a case
if(len(sys.argv)<2):
    print 'Please use option -h to see all available options'
```

```python
# Python 2.5 : Batteries included
try:
    import xml.etree.ElementTree as ET
except ImportError:
    # Python <2.4
    import elementtree.ElementTree as ET
except ImportError:
    raise ImportError, "Element Tree is not installed."
```
return

if(sys.argv[1]=='-c'):
    if(len(sys.argv)<3):
        print 'Please give a case name'
        return
    print 'Case %s is being created.' %sys.argv[2]
    create_case1 = 'mkdir in_%s' %sys.argv[2]
    create_case2 = 'mkdir bin_%s' %sys.argv[2]
    create_case3 = 'mkdir out_%s' %sys.argv[2]
    case1 = 'in_%s' %sys.argv[2]
    case2 = 'bin_%s' %sys.argv[2]
    case3 = 'out_%s' %sys.argv[2]
    CopySetupToBin = 'cp do.py bin_%s/.' %sys.argv[2]
    CopySetupToOut = 'cp do.py out_%s/.' %sys.argv[2]
    CopyTemplatesToIn = 'cp -r src/templates/in_/* in_%s/.' %sys.argv[2]
    CopyTemplatesToBin = 'cp -r src/templates/bin_/* bin_%s/.' %sys.argv[2]
    CopyTemplatesToOut = 'cp -r src/templates/out_/* out_%s/.' %sys.argv[2]
    os.getcwd()
    if os.path.exists(case1):
        print 'Path already exists.'
    else :
        system(create_case1)
        system(create_case2)
        system(create_case3)
        system(CopySetupToBin)
        system(CopyTemplatesToIn)
        system(CopyTemplatesToBin)
        system(CopyTemplatesToOut)
        print 'Case created successfully.'

# Option to print help options
elif(sys.argv[1]=='-h'):
    print 'help option is invoked.'
    print 'LiberLocus multiscale multiphysics solver version 1.0.'
    print 'To create a case use -c'
    print 'To run a case use -r'

# Option to create exe with make
### post.py

The python script "post.py" which is used to post-process tecplot files is as follows:

```python
C.6.2 post.py

The python script "post.py" which is used to post-process tecplot files is as follows;
```
import os, time, sys
from os import system, remove, path

def main():
    # Option to create a case
    
    if(len(sys.argv)<4):
        print 'Please use option -h to see all available options'
        return

    if(len(sys.argv)==4):
        timestep = '%s' %sys.argv[2]
        MPIRanks = '%s' %sys.argv[3]
        print sys.argv[1]
        print 'SUGAR post-processing tool operating...
        print 'At %s th timestep for %s MPI ranks' %(timestep, MPIRanks)
        bfn = 'SUGAR_Tecplot_Output_' #baselineFileName
        myRange = range(0, int(MPIRanks))
        removeMainFile = 'rm %s%s.tec' %(bfn,timestep)
        system(removeMainFile)
        for i in myRange:
            appendCommand = 'cat %s%s_%s.tec >> %s%s.tec'
            %(bfn,timestep,i,bfn,timestep)
            system(appendCommand)
            print 'File #%i is appended to %s%s.tec' %(i,bfn,timestep)

        if(sys.argv[1]=='keep'):
            print 'Partitioned files are kept since KEEP option was used'
        elif(sys.argv[1]=='remove'):
            print 'Partitioned files are removed since REMOVE option was used'
        else:
            print 'Unknown argument used, partitioned files are kept for safety'

        elif(sys.argv[1]=='-h'):
            print 'help option is invoked.'
            print 'LiberLocus multiscale multiphysics solver version 1.0.'
            print 'To post-process a case use 8000 32, the timestep
            and total number of MPI ranks'
    if __name__ == '__main__':
        main()
D Future Extensions

Since 2012, major focus was on the development of the MPI framework of SUGAR. Transforming SUGAR into an exa-scale computing tool will definitely require hybrid use of other parallelism approaches such as shared memory parallelism (e.g. OpenMP) or accelerator attached to computing nodes (e.g. GPUs). Moreover, for higher fidelity chemistry modeling (larger chemistry set, multiple chemical species), the routines in chemistry should be revisited. Finally, the quasi-neutrality assumption used in PIC routines (electromagnetics routine) should be replaced by a higher fidelity approach (e.g. solving a full PDE set) in order to address problems such plasma sheath modeling.
References


[64] *ANSYS Academic Version 15.0*, ANSYS Inc.


Vita

EDUCATION
M.S. Aerospace Engineering, Georgia Institute of Technology, 2011
B.S. Aerospace Engineering, Middle East Technical University, 2009

EMPLOYMENT
Aug. 2011 – present, Graduate Research Assistant, Aerospace Engineering, Penn State University
Aug. 2009 – Jun. 2011, Graduate Research Assistant, Aerospace Engineering, Georgia Institute of Technology

PUBLICATIONS AND TECHNICAL PRESENTATIONS