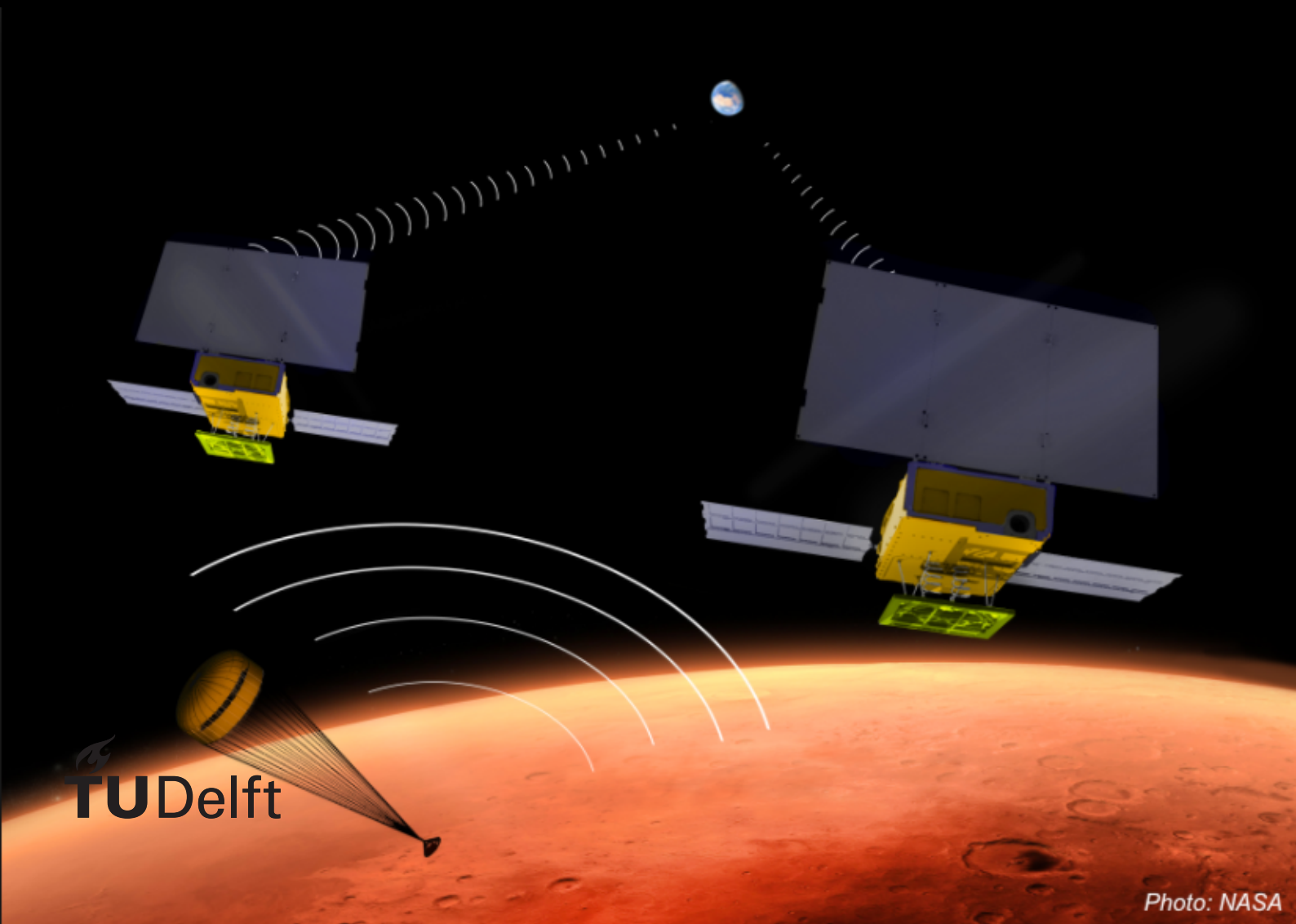
# Comparative Study Between MEMS and Conventional Thrusters for Small Spacecraft Micropropulsion

Karim Khamis

**Master of Science Thesis**



TUDelft

Photo: NASA

# Comparative Study Between MEMS and Conventional Thrusters for Small Spacecraft Micropropulsion

by

Karim Khamis

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday May 29, 2019 at 1:00 PM.

Student number: 4712129
Project duration: October 1, 2018 – May 29, 2019
Thesis committee:

| | |
|---|---|
| Prof. dr. ir. Bendiks Jan Boersma (Chairman), Department of Process & Energy (Energy Technology) Chairman, | Mechanical Engineering, TU Delft - Supervisor Faculty of Mechanical, Maritime, and Materials Engineering |
| Dr. Angelo Cervone, Department of Space Engineering (Space Systems Engineering), | Aerospace Engineering, TU Delft - Supervisor Faculty of Aerospace Engineering |
| Dr. ir. Mathieu J. B. M. Pourquie, Department of Process & Energy (Fluid Mechanics), | Mechanical Engineering, TU Delft Faculty of Mechanical, Maritime, and Materials Engineering |



# Abstract

Miniaturization of spacecraft has been gaining wide interest in the space industry, given its potential for reducing space missions' costs and providing a novel approach to enhancing and facilitating spaceflight, such as allowing efficient distributed space systems. Its focused robotized use is revolutionizing space technology. Recently, a lot of research has been successfully put into this field along with the advancements that make it more feasible, though a major obstacle to achieving the new generation of spacecraft is the technical challenge of fitting a suitable propulsion system. Useful chemical propellants are usually corrosive, flammable, and/or toxic, so alternatives need to be found. The aerospace industry is shifting towards green and nontoxic propulsion systems, so water could be used as an effective propellant, considering its relatively high mass density and low molecular mass. New microelectromechanical systems (MEMS) technologies show promising opportunities for the integration of miniaturized propulsion systems, due to their versatility and robustness. The propulsion system can then properly fit alongside microsensors, microactuators, microelectronics, and other technologies on small spacecraft, though its functionality is to be researched and improved. Certain numerical analysis methods can be implemented to study its thruster's applicability. In this thesis, a comparative study of nozzle flow, heat transfer, and thermodynamics in two different thrusters is conducted. One thruster is based on MEMS, with a typically quasi-2D geometry, while the second thruster is based on more conventional technologies and manufacturing techniques, with an axially symmetrical 3D shape.

After briefly introducing micropropulsion and discussing the propellant selection and nozzle fabrication along with the background theory related to micropropulsion as well as the analytical and OpenFOAM numerical (DSMC, continuum, and a hybrid approach containing both to accommodate to the variation in Knudsen number throughout the computational domain) modeling methods, the used methodology is based on using OpenFOAM's DSMC solver (dsmcFoam+) following the mesh creation using blockMesh and snappyHexMesh and developed analytical model (using MATLAB and CoolProp) along with an additional VLM ANSYS Fluent CFD model prepared in advance at TU Delft, where their (steady state as well as transient with very quick backward forming shock diamonds detected from the throat for DSMC) results (including the same conventional and MEMS nozzles) are processed and discussed. The nozzles are simulated for inlet pressures of 5 and 7 bar at inlet temperatures of 550 and 773 K for a total of four cases for each nozzle. To note, many of dsmcFoam+'s functionalities (mass flow rate measurements, inlet pressure boundary condition, axisymmetric capabilities, statistical error measurements, and dynamic load balancing) are implemented and described along with the full methodology, as Blender (with add-ons) and ParaView with a Python script to extract averaged data (along the nozzle and plume region) along with sampleDict are also used in pre and post-processing respectively and the simulations are carried out on a computer cluster. Furthermore, a quite interesting theoretical project on the side has been independently worked on in parallel. It started as a noticed idea that was decided to be explored using equations, which led to extended continuum/kinetic dimensionless numbers for diffusivity ($DN$) and rarefaction intensity relative to the studied object's timescale ($VDN$). $DN$ represents the continuum advective transport rate to intrinsic kinetic diffusive transport rate ratio of an object/particle in its fluid medium (ideal gas) defining how efficiently an object or particle with a constant interface can blend or diffuse into the fluid medium and between the fluid's own molecules at the instant of evaluation, as a larger and faster or smaller and slower object/particle will experience relatively greater resistance as determined by the fluid medium, which seeks optimal balance with its own properties. $VDN$ is analogous to the Mach number with an average molecular speed term instead of speed of sound (as found within $DN$ along with the Knudsen number), where a faster object/particle will observe a relatively slower flow medium average molecular speed leading to a greater rarefaction intensity and vice versa. See Appendix A for the theory derivation and its general implementation along with the explanation and evaluation. It is also tested in the present study and shows promising results, including that $DN$ is capable of approximately detecting the flow regime within its assumptions and could be more helpful when thermodynamic data (such as dynamic viscosity) for calculating the Reynolds number is difficult to obtain and $VDN$'s approach becomes rather different compared to the Mach number in flows such as the initially vacuum plume region, as the Mach number could vary sig-

nificantly (increase), which allows for potential applications for $VDN$ and its understanding in highly rarefied flows.

Concerning the conventional and MEMS nozzle comparisons, it is quite clear that the conical 3D conventional nozzle (simulated as a wedge with single cell thickness using axial symmetry) is superior in performance realistically, due to the quasi-2D MEMS nozzle's (simulated as 3D) significant boundary layer considering the viscous dissipation of flow kinetic energy from shear on the walls, especially after the throat in the diverging section. However, it reaffirms that the MEMS nozzle's geometry provides easier heat transfer with proper exterior insulation mitigating undesired heat rejection, which could be an advantage with the propellant heating involved, as the heat for these thrusters is not coming from chemical reactions, but from resistive microheaters instead, along with the possible uses for (regenerative along with potential film, curtain, transpiration, and radiation) cooling to avoid melting (in different conditions considering that the (stored) inlet microresistojet temperature considered is 283.16 K if it were to be used [27]), or decreasing viscosity, as gas viscosity generally increases as temperature increases due to the gas molecular collisions increase, contrary to the liquid viscosity decrease with a temperature increase, considering that it decreases the dominant cohesive force between the liquid molecules. Ultimately, there are tradeoffs to choosing either thruster, where it is impractical to fault one nozzle for not performing better than the other, as it comes back to the desired features and nature of the mission each is undertaking, where compromises have to be made.

# Preface

I would like to sincerely thank Dr. Cervone for this thesis opportunity and his continuous support and constructive criticism. As his master's student, our meetings have always been enlightening. I truly admire his professionalism, while his care and motivation made this experience all the better. Professor Boersma's kind and helpful advice and positivity throughout the project are tremendously appreciated. I am also grateful for Dr. Pourquie's wholehearted assistance and guidance, especially from a numerical standpoint and with accessing and utilizing the Faculty of Mechanical, Maritime, and Materials Engineering computer clusters for the simulations.

If there is anything that I kept with me throughout my education, it is the engineering method, which focuses on the process of not only optimally solving problems, but also inventing solutions. The environment (though not exactly when it comes to the weather) at Delft University of Technology has always been what drove me here. The level of expertise and general attitude towards science is unsurpassed. TU Delft is a powerhouse at the frontier of producing prime research and education with a limitless noble mindset. Hopefully as strived for, the work is chiefly comprehensive and finds and serves a constructive purpose for fellow interested people and following related research.

In genuine gratefulness and warmest regards, I dedicate this work to my parents (Hani and Barea), siblings (Sami and Nour), and girlfriend (Ellen).

It has been an honor.

*Karim Khamis*
*Delft, May 2019*

# Contents

# List of Figures

# List of Tables

# Nomenclature

| | |
|---|---|
| $\alpha$ | Thermal diffusivity, diverging half angle, or variable soft sphere (VSS) molecular model parameter as applicable |
| $\bar{v}$ | Average molecular speed |
| $\beta$ | Converging half angle |
| $\Delta p$ | Pressure drop |
| $\Delta t$ | Time step size |
| $\Delta v$ | Delta-$v$ |
| $\delta_{ij}$ | Kronecker delta |
| $\dot{N}$ | Particle flux |
| $\epsilon_{tr}$ | Overall translational energy |
| $\Gamma$ | Vandenkerckhove function |
| $\gamma$ | Specific heat ratio |
| $\hat{\mathbf{v}}$ | Momentum density |
| $\hat{E}$ | Total energy density |
| $\lambda$ | Molecules' mean free path |
| $\mathbf{c}$ | Molecule velocity vector |
| $\mathbf{F}$ | Force |
| $\mathbf{n}$ | Normal unit vector |
| $\mathbf{q}$ | Heat flux |
| $\mathbf{r}$ | Molecule position vector |
| $\mathbf{S}_f$ | Outwards vector normal to owner cell face surface with magnitude denoting the face's area |
| $\mathbf{V}$ | Velocity vector |
| $\mathrm{I}$ | Unit tensor |
| $\mu$ | Viscosity |
| $\Omega$ | Solid collision angle |
| $\omega$ | Temperature coefficient of viscosity |
| $\Phi$ | Flux |
| $\phi$ | Azimuthal angle |
| $\psi$ | Fluid compressibility |
| $\rho$ | Density |
| $\sigma$ | Stefan-Boltzmann constant |

| | |
|---|---|
| $\sigma_T$ | Molecular cross section |
| $\tau$ | Viscous stress tensor or time period as applicable |
| $\theta$ | Elevation angle or angle to surface normal as applicable |
| $\Theta_t$ | Angular position limit |
| $\tilde{c}$ | Molecular thermal speed |
| $\underline{\mathbf{D}}$ | Deformation gradient tensor |
| $\vec{v}$ | Velocity vector |
| $\xi$ | Molecular velocity |
| $^*$ | For post-collision properties |
| $A$ | Area |
| $a$ | Accommodation coefficient or local speed of sound as applicable |
| $B$ | Equilibrium breakdown parameter |
| $C$ | Specific heat |
| $c$ | Molecular speed, specific heat, or speed of sound as applicable |
| $c'_m$ | Most probable molecular speed |
| $c_r$ | Two colliding molecules' relative speed |
| $CFL$ | Courant-Friedrichs-Lewy number |
| $D$ | Diameter or mass diffusion coefficient as applicable |
| $d$ | Thickness |
| $D_h$ | Hydraulic diameter |
| $d_{ref}$ | Reference molecular diameter |
| $DN$ | Dimensionless number |
| $E$ | Total nonchemical energy or fractional statistical error as applicable |
| $e$ | Internal energy |
| $e_s$ | Sensible energy |
| $F$ | Force |
| $f$ | Velocity distribution function, Darcy friction factor, or direction as applicable |
| $F_N$ | Corresponding number of actual atoms or molecules represented by individual DSMC particles |
| $g$ | Gravitational acceleration |
| $H$ | Enthalpy |
| $h$ | Heat transfer coefficient or height as applicable |
| $h_s$ | Sensible enthalpy |
| $I$ | Impulse or current as applicable |
| $I_{bit}$ | Minimum impulse from thruster firing |

| | |
|---|---|
| $I_{sp}$ | Specific impulse |
| $I_{ssp}$ | System-specific impulse |
| $J$ | Nonlinear integral describing collisions of two molecules or moment of inertia as applicable |
| $K$ | Thermal conductivity or scattering kernel as applicable |
| $k$ | Boltzmann constant |
| $Kn$ | Knudsen number |
| $L$ | Representative physical length scale or length as applicable |
| $l$ | Moment arm, roughness height, or characteristic length as applicable |
| $L_v$ | Latent heat of vaporization |
| $L_{max}$ | Maximum present level of parallel load imbalance |
| $M$ | Mach number, molar mass, or number of independent samples as applicable |
| $m$ | Mass |
| $m_n$ | Normalized mass flow rate |
| $M_w$ | Molecular mass |
| $Ma$ | Mach number |
| $N$ | Instantaneous number of DSMC particles |
| $n$ | Number density or number of moles as applicable |
| $N_A$ | Avogadro's number/constant |
| $n_{proc}$ | Number of activated simulation processors |
| $P$ | Probability, power, stress tensor, or perimeter as applicable |
| $p$ | Pressure or momentum as applicable |
| $Pe$ | Péclet number |
| $Q$ | Heat or volumetric flow rate as applicable |
| $q_i$ | Monoatomic gas heat flux vector |
| $R$ | Resistance or gas constant as applicable |
| $r$ | Radius |
| $R_A$ | Universal gas constant |
| $R_f$ | Uniform random number picked between 0 and 1 |
| $Re$ | Reynolds number |
| $RWF$ | Radial weighting factor |
| $S$ | Molecular Mach number or dimensionless pressure drop as applicable |
| $sgn$ | Sign function |
| $T$ | Temperature |
| $t$ | Time |

| | |
|---|---|
| $t_s$ | Residence time |
| $u$ | Local velocity or stream-wise velocity as applicable |
| $V$ | Volume or voltage as applicable |
| $v$ | Velocity or tangential velocity as applicable |
| $v_{rms}$ | Root mean square speed |
| $VDN$ | Velocity-based dimensionless number |
| $w$ | Depth or width as applicable |
| $Z_{rot}$ | Rotational relaxation probability |

# 1 Introduction

## 1.1. The Need for Small Spacecraft

The technological spacecraft set to embark humankind's journey beyond Earth made history in the mid-1900s. On October 4, 1957, the first artificial satellite, the USSR's SPUTNIK 1, was launched into low Earth orbit (LEO), sparking the Space Age [20]. Within the first few months of 1958, the United States launched its first two satellites, Explorer 1 and Vanguard 1 with masses of 15 kg and 1.5 kg respectively [11]. The Soviet Union followed by sending a 1.5-ton "flying laboratory" into orbit to counter the first true small satellites launched by the United States, which marked the dawn of the Space Race [20].

In addition to the spike in larger multifunctional satellite developments over the following years with advances in space technology, satellite pointing requirements and orbital precision were achieved using on-board propulsion systems to counter aerodynamic drag, solar pressure disturbances, and gravity-well distortions caused by the Earth's oblateness [11]. The field of micropropulsion was initiated for smaller satellites, which were suitable for the launch vehicle capacity [11].

In 1959, a famous lecture "There's Plenty of Room at the Bottom" was presented by Richard Feynman to acknowledge miniaturized systems, but it was not until the 1990s when micropropulsion was truly realized using microelectromechanical systems (MEMS), which were developed in the 1960s for watches using quartz crystal [12]. The small satellites of today are typically under 500 kg and are categorized based on their mass as shown in Table 1.1, though the main focus will be placed on small satellites under 100 kg to a minimally applicable limit, which concerns the groups typically associated with micropropulsion.

Table 1.1: Satellite classification [32]

| Group | Mass (kg) |
|---|---|
| Large satellite | >1000 |
| Medium satellite | 500 to 1000 |
| Mini satellite | 100 to 500 |
| Micro satellite | 10 to 100 |
| Nano satellite | 1 to 10 |
| Pico satellite | 0.1 to 1 |
| Femto satellite | <0.1 |

Although small satellites are not yet developed to become individually multifunctional, similar to much larger satellites, their focused automatic implementation might offer great opportunities. For example, cost-effective efficient distributed space systems could replace the general idea of larger spacecraft and open a novel field for accessible space missions. They could also be flying in very low-altitude orbits. Nowadays, the small satellite field of research and development has grown tremendously and across many nations. Small unmanned spacecraft, such as satellites and probes, are being launched in hundreds at once, creating universal communication, navigation, observation, among other feats a feasible reality within a shorter production time [34]. There has been an approximately 40% annual

rate rise in nano- and microsatellite launches since 2011 and it is projected to continue, as there were 101 nano- and microsatellite global launches in 2013 compared to 320-460 launches in 2016 [34]. For example, SpaceX plans to achieve a communication system for global internet by launching above four thousand microsatellites into predefined orbits [34].

CubeSats are U-class spacecraft, which means they can be combined by dimensions of single units (U), built of multiple 10 $cm^3$ unit cubes with a maximum mass of 1.33 kg per unit, while PocketQubes, which are similarly P-class of units (P), are around 5 $cm^3$ and 250 grams at most. Figure 1.1 shows a typical CubeSat and small satellites' launch history and projection and Figure 1.2 shows nanosallite launches with projections.

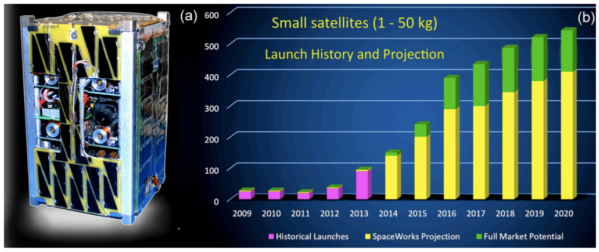



Figure 1.1: CubeSats and their launches: (a) A 2U 2.5 kg electrically-propelled CubeSat. (b) Launch history and projection of small satellites (1-50 kg) from 2014, proving relatively accurate results as explained for 2016 [34].

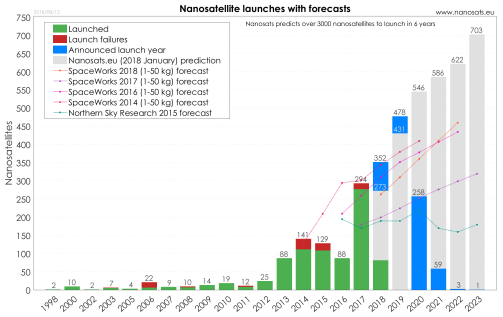



Figure 1.2: Nanosatellite launches with forecasts from 2018 [2]

Small satellites allow the commercial and academic sectors to join the well-established space sector players to possibly become vastly common among them. It was only the major space organizations that were capable to produce, enhance, insure, deploy, and operate spacecraft with costs typically over a hundred million USD. With the advent of small satellites, the general cost of producing a communications satellite starts at twenty-five thousand USD, with an estimated outlay of one million USD. In addition, dedicated small launch vehicles and other newer rocket families with larger payload capacity such as SpaceX Falcon or Arianespace Ariane make space more accessible for small satellites, even as additional payload [34]. Small spacecraft are usually deployed as piggyback launches from launch vehicles, such as carrier rockets or aircraft for air launch to orbit. Some difficulties arise with current capabilities on managing positioning, autonomous operation, and final decommissioning in regions with wide temperature differences, trapped and transient radiation, and bombardment by dust and debris [34].

However, the need for small spacecraft is mainly limited by the need for micropropulsion. Generally, the efficiency and reliability of propulsion systems for small spacecraft need improvements to last for longer periods of deployment. In short, miniaturization of propulsion systems might not be as direct as downscaling the larger ones used, but might require inventing new concepts and integration techniques.

## 1.2. State-Of-The-Art Micropropulsion

Smaller, lighter, cheaper, and often more functional space technologies have been becoming more possible with the great progress in the fields of microelectronics and miniaturized space robotics [34]. Generally within the solar system closer to the Sun, spacecraft rely on photovoltaic (PV) solar panels as the main electric power source, but as the Sun gets further away, radioisotope thermoelectric generators (RTGs) become more effective [7]. Figure 1.3 considers the different propulsion systems for their desired application. Delta-v ($\Delta v$) is summed linearly for several maneuvers, which provides an indication for propulsion system selection as shown in Figure 1.3 along with the desired thrust. It is important to note that the desired maximum thrust is accompanied by the desired minimum thrust to allow for precise maneuvers. Additional pioneering propulsion technologies are to follow in this section.

Figure 1.4 shows the four prominent types of electric propulsion thrusters used today [34]. The solid and liquid propellant rocket engines in the center of the figure, which use chemical propulsion systems, have a significant thrust-to-weight ratio up to 200 and maximum exhaust velocity of around $5000\frac{m}{s}$ using optimal fuels, such as liquid hydrogen and liquid oxygen [34]. On the other hand, electric propulsion systems provide larger exhaust velocities up to $10^4\frac{m}{s}$ without physical limitation for improvement below

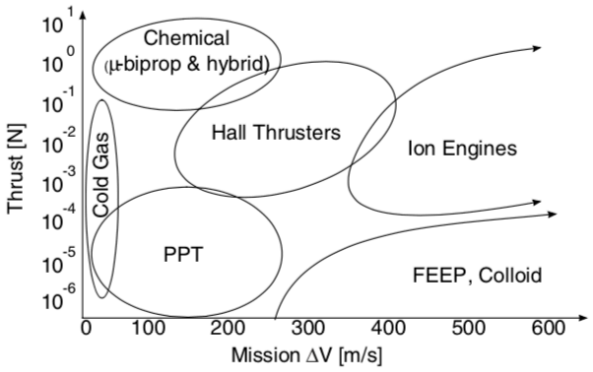



Figure 1.3: Propulsion modes with different properties for various applications [11]. Electric propulsion appears to be more suitable for higher $\Delta v$, but this does not account for all of the innovative propulsion systems of today.

the speed of light, though the thrust is much lower with thrust-to-weight ratio up to 0.01. The energy efficiency achieved by gridded ion and Hall thrusters, which are electrostatic thrusters, is considered highest of all at 75%. Plasmadynamic thrusters are continuously operated and allow for larger thrust-to-weight ratio, while pulsed plasmadynamic thrusters prove highly efficient with low thrust pulses for increased maneuvering precision [34]. Electric propulsion systems' lifetime and device power efficiency are limited due to their reliance on electric energy to speed up the ionized propellant, though they provide high low-thrust efficiency and specific impulse [34]. Chemical propulsion systems are usually energy limited [16]. To note, there are many other impressive candidates for electric propulsion systems in [34], but only some are discussed for the sake of introduction to miniaturized electric space propulsion systems.

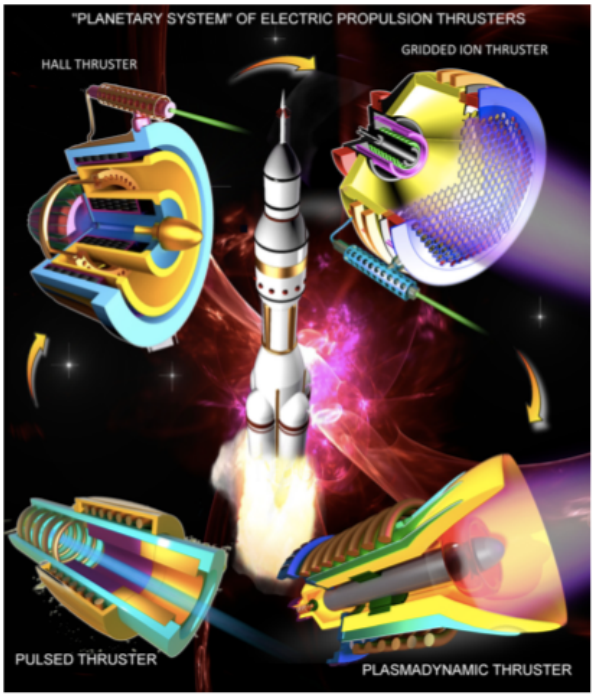



Figure 1.4: Main state-of-the-art electric propulsion systems [34]

Table 1.2 shows a comparison of different notable small satellite propulsion systems based on current capabilities which might be enhanced, though the list is far from being comprehensive considering the popularity of developing the optimal small spacecraft propulsion system [42]. Each criteria is given a weight which is multiplied by the assigned expected comparative value for the criteria from zero to ten, where higher is better.

Table 1.2: Small satellite propulsion systems comparison table [42]

| Criterion (Weight) / Propulsion System | Mass (7) | TRL (6) | Power (8) | Manufacturability (7) | Safety (8) | Cost (8) | Storability (7) | Total | Rank |
|---|---|---|---|---|---|---|---|---|---|
| **Cold Gas** | 1 | 6 | 8 | 5 | 7 | 6 | 7 | 295 | 1 |
| **Resistojet** | 2 | 5 | 5 | 6 | 7 | 7 | 7 | 287 | 2 |
| **Monopropellant Chemical** | 3 | 4 | 9 | 4 | 5 | 6 | 4 | 271 | 3 |
| **Arcjet** | 4 | 5 | 3 | 4 | 7 | 7 | 7 | 261 | 4 |
| **Solid Chemical** | 1 | 5 | 8 | 2 | 4 | 5 | 8 | 243 | 5 |
| **Solar Thermal** | 2 | 4 | 8 | 1 | 7 | 3 | 7 | 238 | 6 |
| **Bipropellant Chemical** | 1 | 4 | 9 | 2 | 4 | 5 | 5 | 224 | 7 |

Although Table 1.2 shows that the first place goes to cold gas propulsion systems, which are the most experimented with and flight qualified, they typically provide low specific impulse, which negatively affects their volume and mass, as safety complications arise due to their propellant pressurization. The latest satellites are tending towards the more volume and $\Delta v$ efficient resistojets, which are electrothermal and ranked second, and electric propulsion systems, even though their technology readiness level (TRL) is still low [42]. The typical liquid propellant of a resistojet needs more energy for vaporization compared to a single phase cold/hot gas thruster, but resistojets offer lower pressure and lighter storage tanks and considering the valve quality and propellant amount, resistojets can be operated discontinuously compared to solid propellant thrusters [15]. Table 1.3 provides a summary for the International Organization for Standardization (ISO) TRL summary [1]. The purpose of mission plays a significant role in choosing the desirable propulsion system, as the performance, thrust, and $\Delta v$ need to be considered directly in Table 1.2.

Table 1.3: ISO technology readiness levels and their description [1]

| TRL | Level Description |
|---|---|
| 1 | Basic principles observed and reported |
| 2 | Technology concept and/or application formulated |
| 3 | Analytical and experimental critical function and/or characteristic proof-of-concept |
| 4 | Component and/or breadboard functional verification in laboratory environment |
| 5 | Component and/or breadboard critical function verification in relevant environment |
| 6 | Model demonstrating the critical functions of the element in a relevant environment |
| 7 | Model demonstrating the element performance for the operational environment |
| 8 | Actual system completed and accepted for flight (”flight qualified”) |
| 9 | Actual system ”flight proven” through successful mission operations |

Table 1.4 indicates the typical characteristics of different small satellite propulsion systems [42]. Just like Table 1.2, it does not contain a comprehensive list of all propulsion systems. Electric propulsion provides high specific impulse and $\Delta v$, but low thrust due to its power limitations. Chemical propulsion provides high thrust, but has temperature limitations. Cold gas, microresistojet, and monopropellant propulsion provide high thrust, though they have a relatively low specific impulse. Microresistojet propulsion systems ultimately show their relative superiority to cold gas propulsion systems in power, thrust-to-power ratio, temperature, simplicity, scalability, and specific impulse [42].

Table 1.4: Small satellite propulsion systems' characteristics [42]

| Characteristic / Propulsion System | Thrust (mN) | Specific Impulse (s) | Application | Power (W) |
|---|---|---|---|---|
| **Cold and Warm Gas** | 0.0001-100 | 30-100 | Station Keeping and Attitude Control (Suitable for Small $\Delta v$: 4-5 $\frac{m}{s}$) | 2-6 |
| **Microresistojet**<br>**TU Delft's VLM**<br>**TU Delft's LPM** | 0.1-10<br>1.7-4.2<br>0.5-3.0 | 80-200<br>110-130<br>80-120 | Station Keeping and Attitude Control ($\Delta v > 10$ $\frac{m}{s}$) | 4-10<br>0-2 |
| **Pulse Plasma Thrusters** | 0.0001-0.001 | 300-3000 | Station Keeping, Attitude Control, and Slow Orbit Transfer | 2-10 |
| **Electrospray** | 0.005-0.05 | 1000-5000 | Accurate Orbit Control and Slow Orbital Transfer | 1-15 |
| **Chemical** | 100-1000 | 150-350 | Orbit Transfer | 10-20 |
| **Water Electrolysis** | 50-500 | 110-300 | Accurate Orbit Control (However, Difficult in Controllability) | - |
| **Ion/Hall Effect** | 0.1-5 | 1000-4000 | Precise Pointing, Slow Orbit Transfer and Maneuvers | 10-30 |

With the different mission requirements and restrictions considered, thrust level, specific impulse, and minimum impulse bit are typically the most important parameters for performance [16]. When

variously different methods of propulsion are considered, the system specific impulse can provide better insight into the suitable micropropulsion system. Additionally, current CubeSat technologies reach a minimum of 5 mm misalignment between the thrust vector and satellite center of mass, which results in a torque that needs to be adjusted for using attitude control, so attitude control maximum disturbance torque level requirements need to be complied with using a more limited thrust level [16].

In addition to propulsion systems that boost by shooting out particles, Newton's third law, which states that every action has an equal and oppose reaction, can be applied in different ways. Other compelling propulsion systems that also require further development include solar sails, which use radiation pressure, as photons are reflected on their large and highly reflective sail, to move similar to using the wind for a sailboat. The radiation could be obtained from the Sun, or more interestingly, by shooting guided lasers at the spacecraft, providing a potentially effective means for deep space travel.

## 1.3. Micropropulsion at Delft University of Technology

The main contemporary micropropulsion technologies at Delft University of Technology consist of a second generation vaporizing liquid microresistojet (VLM) and low-pressure microresistojet (LPM) thrusters [42]. The first generation microresistojet developments began around 2010 [42]. Both thrusters are based on microelectromechanical technologies, where the performance can be improved at lower volume and mass, and currently use water as propellant. Microresistojets have been realized to be promising for small spacecraft applications, considering their fast response, low required volume and mass, high thrust-to-mass ratio and reliability, and ease of integration, especially in a thruster array [27]. This thesis focuses on the VLM, where the propellant is fed through an inlet channel for vaporization in a heating chamber to be followed by acceleration to supersonic velocities in an inplane de Laval nozzle [49]. The heating element's geometry and material are essential for optimization, as it is the main section with heat transfer and the energy transfer is not typically efficient [49]. The VLM's liquid propellant needs to be slightly pressurized by a gas, which in some cases could be a constraint considering the elaborate two phase mixing caused by microgravity, along with that the LPM then contains slightly less wet mass and equivalent initial propellant mass due to its lower operational pressure [16], though that also poses challenges in the valve and tank designs [49]. As the thrust level is typically high for the required minimum impulse bits, the propulsion system might need to be rapidly operated for short periods of time with a fast response time around one or two orders of magnitude less than a second, which might cause complications taking into account the heat transfer processes [16]. It is important to note that the performance of both thruster systems is generally similar, but can be compared on an operational basis, such as the required temperature and pressure conditions for propellant storage [16]. Considering the VLM and LPM's complementary operating conditions of temperature and pressure, a dual thruster model could be used, where both thrusters share a propellant tank, feed system, and sensing and control electronics. Table 1.5 includes a rough approximation for the micropropulsion system mass considering the effective masses of its components.

Table 1.5: Approximate mass of TU Delft micropropulsion system payload

| Items | Mass (g) | Minimum # of Components |
|---|---|---|
| Thruster and Housing (VLM and LPM) | 20 | 2 |
| Feed System (Valves, Connectors, and Tubing) | 20 | 2 |
| Storage Tank and Sensors | 10 | 1 |
| Electronic Board with Microcontroller | 13 | 1 |
| Pressurant/Propellant | 12 | N/A |
| **Total** | **75** | |

Figure 1.5 shows a block diagram of TU Delft's micropropulsion system model, while Figure 1.7 displays its components' placement inside a small satellite. The propulsion system is desirably installed in the middle for precise and slow attitude control and drag compensation. As shown, the propellant tank is based on rolled tubes for maximum volume usage and preventing propellant sloshing, which might undesirably cause the liquid and gas phases to mix. Also, Figures 1.6 and 1.8 represent the concept VLM and LPM thrusters respectively.

The main idea of the model could be reshaped, where each of the thrusters can be used individually. Then, the VLM operates at pressures of 1-5 bar under two-phase flow, with a liquid flow at the inlet

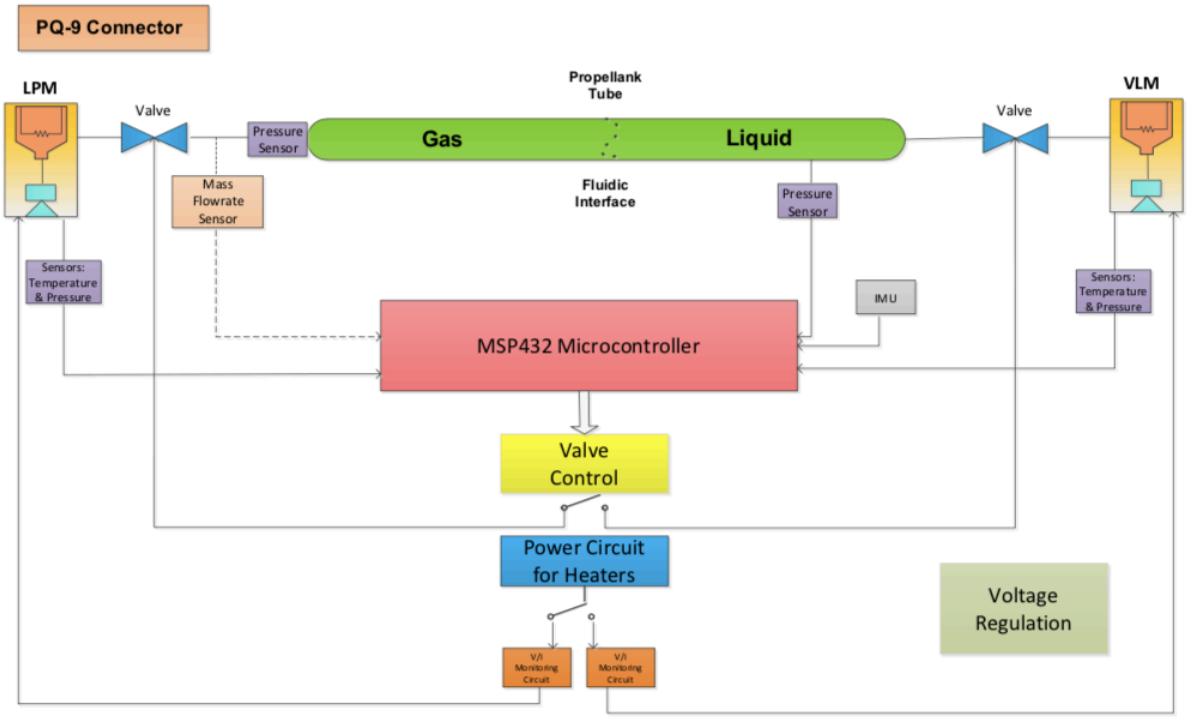



Figure 1.5: TU Delft micropropulsion system schematic [34]

and gaseous flow at the nozzle, and the LPM operates at pressures of 0.01-1 mbar using the gaseous flow from stored sublimating ice at low pressure below the triple point, where the molecules of the ice sublimate to maintain the tank's vapor pressure. TU Delft complies with the propulsion system's set design constraints, such as on mass, volume, and power. Table 1.6 shows TU Delft's CubeSat VLM and LPM micropropulsion target performance using an equal amount of propellant and subject to similar operational conditions [6].

Table 1.6: TU Delft CubeSat VLM and LPM micropropulsion target performance at comparable operational conditions and equivalent propellant amount [6]

| Parameter | Chamber/Channel Temperature (K) | Chamber/Plenum Pressure (Pa) | Propellant Mass Flow Rate (mg/s) | Power transferred to water (W) | Thrust (vacuum) (mN) | Specific impulse (vacuum) (s) | Total impulse (Ns) | Propellant mass (g) | Propellant mass (g) |
|---|---|---|---|---|---|---|---|---|---|
| **VLM** | 550 | $5 \times 10^5$ | 1.63 | 5.25 | 1.52 | 94.9 | 46.4 | 50 | <360 |
| **LPM** | 573 | 150 | 1.32 | 4.51 | 1.14 | 88.1 | 43.2 | 50 | <330 |

The research group at TU Delft is also working on a solar thermal micropropulsion concept, where electric/thermal power and propulsion can be achieved concurrently using solar concentrators transferring energy to a high-temperature organic Rankine cycle (ORC) turbogenerator and thruster(s) [6].

### 1.3.1. Propellant Selection

The choice of propellant has a significant impact on the propulsion system's performance. The aerospace industry is leaning towards green and nontoxic propellants, which are commonly very active chemicals resulting in being corrosive, flammable, or toxic [6]. Inert gases could be used, but they require large storage tanks or undesirably high storage pressures. On the other hand, water ($H_2O$), in its liquid and solid ice forms, has high mass density and low molecular mass, which is suitable for micropropulsion applications, especially the VLM and LPM.

The reasoning behind this decision is based on a TU Delft study on 95 different fluids with diverse properties for microresistojet green propellant selection and characterization considering the necessary safety requirements on pressure, corrosivity, flammability, and toxicity [27]. The selection methodology used consists of subsequently collecting data on the fluids, assessing their feasibility for storage and usage, considering an analytic hierarchy process (AHP) with a decision-matrix Pugh method to rank the fluids based on safety, performance, and system density, and finally making a selection considering the optimal thermal properties, propulsive performance, and safety from the third step to be applied to

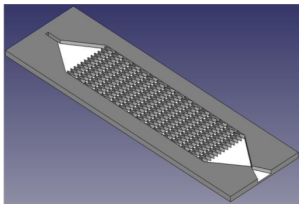


Figure 1.6: VLM thruster concept (flow direction is towards de Laval nozzle)

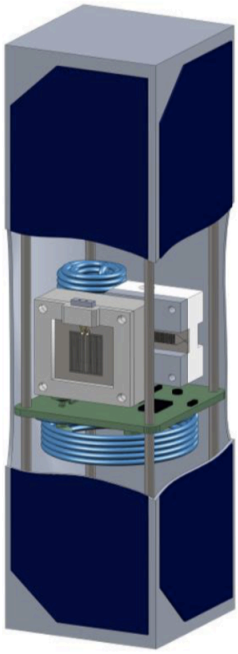


Figure 1.7: TU Delft micropropulsion system concept [34]

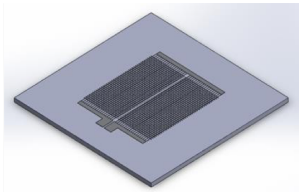


Figure 1.8: LPM thruster concept (flow direction is perpendicularly up)

VLM and LPM [27].

To summarize the work done after the first step, the feasibility check step includes filtering the fluids based on criteria for the required storage pressure and propellant mass, which are considered to be liquids and solids, excluding gases due to their lower density [27]. Step three's Pugh matrix classifies the criteria into first (FL) and second (SL) levels. FL includes safety and design, while SL for safety includes flammability, instability, and health hazard and SL for design includes performance and system density [27]. They are set a weighting factor ($WF$) based on a pairwise comparison for ranking their increasing importance from 1 to 6 and the weight ratio (WR) from the AHP [27]. The criteria were characterized and scored by +, -, or 0 indicating positive, negative, or neutral respectively. The final results for each fluid included an average score and standard deviation, as the weighting factors are based on the evaluations of five experts from TU Delft [27]. The main factors considered to accomplish the final step are the thrust, specific impulse, and required propellant heating power. To calculate the thrust and specific impulse, the ideal rocket theory is applied [27]. Only 63 fluids were selected from the feasibility assessment and the resultant Pugh matrix is shown in Figure 1.9 as a boxplot [27].

It is noticeably that the nine most suitable fluids were acetone, ammonia, butane, cyclopropane, ethanol, isobutane, methanol, propene, and water, where their score ranges are exclusive from the score ranges of other fluids. After comparing their properties at operational conditions, ammonia and water shows the optimal performance with higher specific impulse, though ammonia requires less power as water would undergo phase change for VLM and similarly for LPM, in which case they would show similar required power to other fluids [27]. However, performance is not solely the determining factor, as safety and cost play a significant role, especially for academic or commercial applications, where water becomes the fluid of choice compared to ammonia, because it is slightly flammable and highly hazardous, which produces handling complications. From data in [27], acetone and butane present the highest thrust, while ammonia and water present the lowest thrust for VLM, while the fluid thrusts are equivalent for LPM, though this is generally less important than specific impulse and required power. Also, water, which is typically the most abundant fluid, presents the highest density under storage conditions resulting in the optimal case for maximal volumetric efficiency. Using Equation 2.97, the rocket equation linearized approximation for $\Delta v$ is calculated under certain applicable conditions and water proves to favorably provide the highest $\Delta v$ per volume. Certainly, different propellants could be potential candidates for different missions, but water shows generally promising aspects, though ammonia is a less safe competitor for VLM, as it shows higher $\Delta v$ and lower power consumption. Wa-

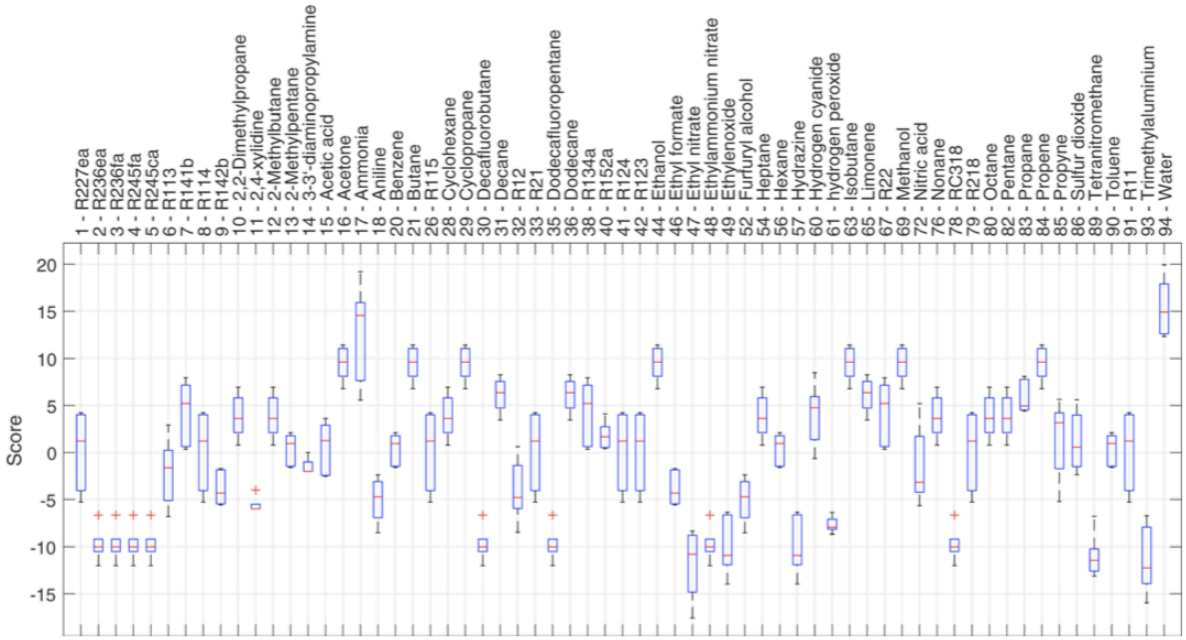



Figure 1.9: Pugh matrix results of propellant choice by score, where higher is better (for every box: internal line represents the median, upper and lower box borders represent the upper and lower quartiles respectively, top and bottom external lines represent the maximum and minimum values, and crosses represent outliers). [27]

ter's main limitations as a propellant are due to its high specific heat and latent heat of vaporization [16]. Specific impulse is simply approximated to be inversely proportional to the square root of the propellant's molecular mass, but a molecular mass that is too low would need a greater storage volume due to its lower density [16]. At 18 $\frac{g}{mol}$, water's molecular mass proves to be just right to fulfill both requirements [16]. Water is currently the most suitable candidate, considering that both VLM and LPM are often considered jointly in the propulsion system. Also, note that nitrogen could be easier to use experimentally than water, which is a reason for its popularity.

### 1.3.2. Fabrication

Some applications for MEMS components include ink cartridges and car accelerometers. They are typically fabricated simultaneously in batches and allow for the integration of multiple components in one module [12], which does not apply to conventionally manufactured components. However, it is common to use the rapidly-increasing available commercially off-the-shelf (COTS) products, which are suitable for spaceflight applications, especially for academic or commercial purposes, because they lead to relatively shorter development times and lower costs [27]. In the future, some micropropulsion systems might become available as COTS components.

Lithography is used to define the structure on a wafer to be etched out. These structures can then be sandwiched to create fluid channels, for example in the case of the two silicon or ceramic wafers of a generic VLM, where only one contains the channels, nozzle, and heater at the present time [49]. Generally, MEMS gas channels have a width smaller than 0.1 mm [12]. The materials used to manufacture the resistive microheaters are either silicon carbide or molybdenum. The only COTS component in the TU Delft's thrusters is the valve [16]. Currently, the microvalves used are COTS solenoid electromagnetic-operated valves with quick response time and can withstand the required application pressure. As shown in Table 1.7, the 1960s bulk manufacturing technologies uses wet or dry etching of typically silicon substrate, the more intricate surface manufacturing forms structures by deposition and etching of sacrificial and structural thin films, and the more expensive and effective LIGA forms 3D structures by the sequential process of mold fabrication and then injection molding or electroplating are the MEMS fabrication technologies to be compared to conventional machining, which might use drilling, milling, forging, welding, etc. [39], though the axisymmetric nozzles could also be constructed using more advanced methods such as semi-isotopic etching with an array mask using microloading and reactive ion etching lag as in [43] or microsystem technology (MST/MEMS)

such as DRIE, and more promisingly, femtosecond laser machining (FLM) or a powder blasting and heat treatment combination as in [36], but their resulting dimensions are still different from the quasi-2D (extruded) MEMS nozzles, especially due to its possible relatively smaller throttle diameter.

Table 1.7: Comparison of MEMS fabrication technologies and conventional machining [39]

| Capability | LIGA | Bulk Micromachining | Surface Micromachining | Conventional Machining |
|---|---|---|---|---|
| **Feature size** | Around 3 to 5 $\mu$m | Around 3 to 5 $\mu$m | 1 $\mu$m | Around 10 to 20 $\mu$m |
| **Device thickness** | $>$ 1 mm | $>$ 1 mm | 13 $\mu$m | Very Large |
| **Lateral dimension** | $>$ 2 mm | $>$ 2 mm | 2 mm | $>$ 10 m |
| **Relative tolerance** | Around $10^{-2}$ | Around $10^{-2}$ | Around $10^{-1}$ | $> 10^{-3}$ |
| **Materials** | Electroplated Metals or Injection Molded Plastics | Very Limited Material Suite | Very Limited Material Suite | Extremely Large Material Suite |
| **Assembly Requirements** | Assembly Required | Assembly Required | Assembled as Fabricated | Assembly Required |
| **Scalability** | Limited | Limited | Yes | Yes |
| **Microelectronic Integratability** | No | Yes for Silicon-On-Insulator (SOI) Bulk Processes | Yes | No |
| **Device Geometry** | Two-Dimensional High Aspect Ratio | Two-Dimensional High Aspect Ratio | Multi-Layer Two-Dimensional | Very Flexible Three-Dimensional |
| **Processing** | Parallel Processing at the Wafer Level | Parallel Processing at the Wafer Level | Parallel Processing at the Wafer Level | Serial Processing |

VLM and LPM are manufactured in a collaboration with TU Delft's Else Kooi Laboratory, formerly known as Delft Institute for Microsystems and Nanoelectronics (DIMES) [16]. The LPM contains a storage tank, plenum chamber, and expansion slots, which sometimes act as heated microchannels. The VLM contains the inlet, heater, and nozzle. The VLM is modular, where each of the inlet section, heating chamber, and nozzle can be swapped for applying a multitude of experiments [6]. Figures 1.10 and 1.11 show the scales of TU Delft's VLM and LPM thrusters respectively, as well as their physical images.

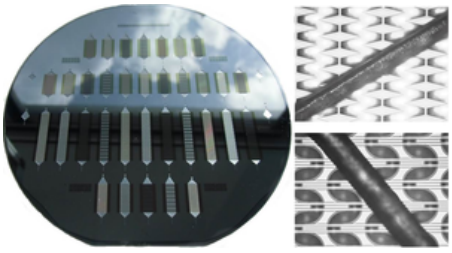


Figure 1.10: Left: MEMS wafer with various VLM design concepts. Right: Two VLM heating sections' SEM-microscope details compared to a human hair (straight dark stripe). [6]

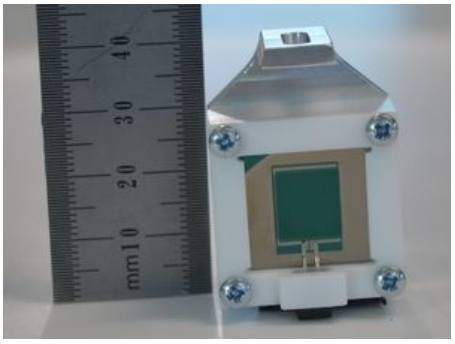



Figure 1.11: LPM prototype with the circular (not visible) expansion slots in the green area [6]

The heating chamber's design requires the consideration of a variety of factors concerning its geometry and internal features, which are limited by the fabrication precision [16]. The main considered options include multiple different shapes and structures for pillars, fins, and plain channels. The resistive heaters are centered in the chamber's channel to reduce the undesired heat transfer to the environment [16]. The etched wafer channels are mirrored with the heating layer in the middle [16]. The heating chamber length significantly influences the design, considering that the flow velocity is low leading to a highly laminar flow, so to achieve complete fluid vaporization, the modular design provides the ability to extend the length of the heating chamber as needed in case the heating efficiency is lower than initially designed for [16]. It is also recommended that the geometry insures that the stochastic water vaporization is complete, otherwise liquid water could make it to the nozzle.

Geometry, boundary layers, and surface roughness greatly influence micronozzles, where flow separation can reduce the effective throat area, which in turn leads to lower actual performance [16]. Water condensation at the nozzle exit is to be prevented. A constant current mode is used due to its lower required power compared to a constant voltage mode, as the increasing temperatures leads to the decreasing resistance of silicon carbide (SiC), which is the selected material for the resistive heating elements because of its inertness, relatively low density, and high strength and maximum operating temperature [16].

For simplicity and cheapness, the channels are etched in silicon, although low temperature co-fired ceramic (LTCC) could also be simpler and cheaper [49], with less than 10% channel depth inaccuracy [16]. An anodically bonded layer of glass for insulating conduction confines both of the thruster's sides, with a second layer of reflecting metal to serve as a radiation shield, and a final layer of around 1 cm consisting of a protective encasing resin compound for extra rigidity and insulation [16]. A silicon dioxide layer is also deposited on the silicon wafer, below and above the heating layer, for electrically insulating between the heating layer and conductive silicon [16]. Etching is applied for the heating structure first, where an anisotropic deep reactive-ion etching (DRIE) is carried out followed by isotropic DRIE without passivation in all directions to suspend the heating elements by etching the structure below them [16]. This procedure is to be applied for both faces of the thruster, but with a pocket where the heating elements should be in the other face, before their silicon fusion bonding. For power connections, 1 by 1 $mm^2$ holes are etched on the silicon wafer's topside until reaching the SiC heating layer in the middle [16]. A thin aluminum layer is deposited on the exposed SiC topside for bonding the electrical wires to it due to their incompatibility with direct SiC bonding [16]. For testing and operation, a custom printed circuit board (PCB) is bonded to the thruster to be followed by electrically connecting it to the thruster's bond pads using golden bond wires, where the bond cavities are then hermetically sealed using a sealant with low viscosity [16].

Molybdenum has also been used as a heating element, considering its endurance to temperature with its 2693 °C melting point and linearly proportional resistivity up to 700 °C for accurate measurements of temperature [50]. The fabrication using molybdenum at TU Delft consists of slightly different steps. Figure 1.12 presents the fabrication process, which is briefly discussed as:

a) Low pressure chemical vapor deposition (LPCVD) silicon nitride (SiN) deposited on both faces for isolation between the heating elements and the substrate [50]

b, c, and d) Molybdenum (Mo) is deposited by sputtering on the top side to be masked with photoresist and plasma-enhanced chemical vapor deposition tetraethoxysilane (PECVD TEOS) before being patterned [50]

e and f) Wafer with heaters is cleaned and masked with silicon dioxide ($SiO_2$) [50]

g and h) DRIE etching of cavities anisotropically and then isotropically [50]

i) Wafer is masked with silicon dioxide and photoresist [50]

j) Anisotropic etching of inlet hole [50]

k and l) Wafer is cleaned and bonded to glass wafer for convenient visualization [50]

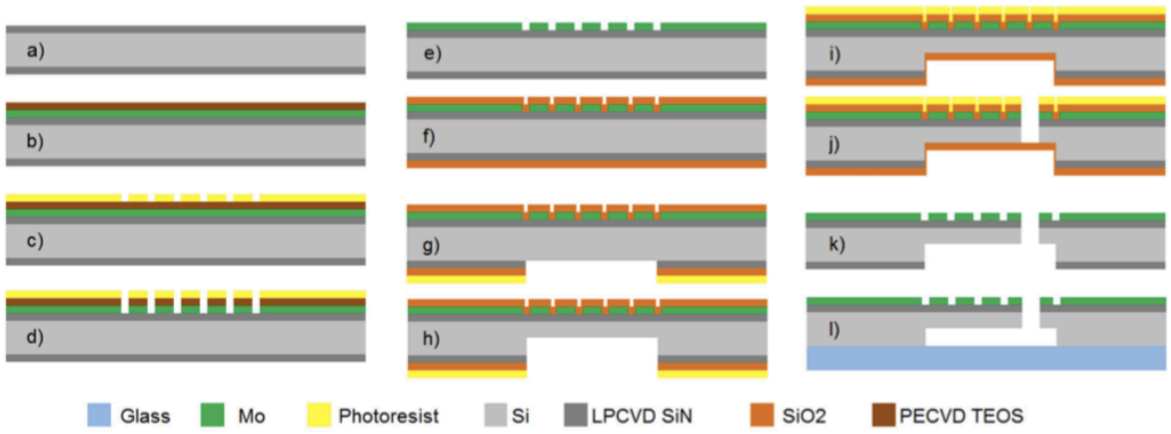



Figure 1.12: VLM fabrication process, where the thrusters become ready for testing after dicing the wafer [50]

## 1.4. Thesis Scope

The main scope of this (mostly analytical/numerical) thesis is to evaluate the advantages and drawbacks of using a microelectromechanical systems (MEMS) solution compared to a conventional solution for two micropropulsion thrusters considering the Department of Space Engineering at Delft University of Technology's (TU Delft) set performance targets and requirements. The MEMS thruster has a typically quasi-2D geometry and the conventionally developed and manufactured thruster has an axially symmetrical 3D shape. Since a more rarefied flow is expected in certain regions, such as in the diverging section of the nozzle and especially due to expansion into vacuum, resulting in a higher Knudsen number, where a molecule's mean free path is significant compared to the flow's representative physical length scale, Direct Simulation Monte Carlo (DSMC) is needed as compared to continuum flow modeling.

The comparative study involves the two thrusters' nozzle flows, heat transfer, and thermodynamics. After a comprehensive literature review related to the project, the modeling process is conducted and optimized for both thrusters, to be followed by the results and discussion as well as the final section with conclusions and recommendations. The open-source C++ toolbox for numerical solvers and pre/post-processing utilities that incorporates computational fluid dynamics (CFD), OpenFOAM (Open-source Field Operation and Manipulation), includes a DSMC solver, dsmcFoam(+), which is used for the required flow simulations.

The MEMS thruster studied is a TU Delft VLM thruster, while the conventional thruster studied is based on the technological possibilities for taking on the role of a MEMS VLM thruster. The main differences between MEMS and conventional thrusters is in the manufacturing and geometry. The manufacturing process for conventional thrusters uses conventional machining techniques, with a minimum throttle diameter size larger than MEMS nozzle miniaturization capabilities. MEMS thrusters will be checked for feasibility and applicability compared to conventional thrusters. The second difference is in the geometry, as explained above, and detailed in this work. The nozzles are simulated for inlet pressures of 5 and 7 bar at inlet temperatures of 550 and 773 K for a total of four cases for each nozzle. The models will be checked outside the nozzle in the plume regions as well, as the pressure immediately after the nozzle exit in the plume region could affect the nozzle flow. Particle dissociation will be considered negligible at the expected speeds. The Reynolds number is expected to remain lower for the MEMS thruster, while higher for the conventional thruster, with the MEMS and conventional nozzles based on geometries with comparable performances. To put this into perspective, the flow gets closer for a rectangular cross section than a circular one for the same area and this could have significant impacts on the performance. Generally, a more unconventional de Laval nozzle flow is expected for the MEMS thruster versus a more conventional de Laval nozzle flow for the conventional thruster. Simply flat/straight sides will be used throughout the nozzle, even though it is not ideally optimal compared to the more elaborate (at such small scale) parabolic shapes.

# 2

# Theory

## 2.1. Scaling Analysis for Micropropulsion

The mass reduction leads to faster dynamic and thermal response times, which lowers the minimum impulse bit [11]. The square-cube law can define performance, where the area-dependent properties decreases slower than volume-dependent properties [11]. Their ratio might be seen proportional to different properties such as the power-to-mass and thrust-to-mass ratios, though the thrust-to-mass ratio might be proportional to the inverse of the tank pressure instead of being scale-dependent, as the engine mass is small compared to the propulsion system including the stored propellant [11]. The Reynolds number ($Re$) can define the frictional losses:

$$Re = \frac{\rho v D}{\mu} = \frac{\dot{m}}{\mu h_o}, \tag{2.1}$$

where $\rho$ is the density of the chamber, which is proportional to the chamber pressure, $v$ is velocity, $D$ is throat width, $\mu$ is viscosity, $h_o$ is the height, in case of an extruded nozzle, considering the evaluation of these values at the throat [11]. Thrust is proportional to the product of the mass flow rate and exit velocity, which is ideally determined from the nozzle expansion ratio, but is reduced by $Re$ as viscous effects rise [11]. Exit pressure is also a function of the nozzle expansion ratio, so it is not considered for thrust scaling. Therefore, by using the proportionality relationship 2.1, and substituting the mass flow rate into the thrust proportionality:

$$F_t \propto \dot{m} v_{exit} \propto Re h_o \mu_{throat} v_{exit} \tag{2.2}$$

$Re$ is found to desirably increase with decreasing scale for constant thrust, which reduces frictional losses and increases specific impulse, so higher thrust-to-mass ratio can be achieved with scale reduction, as less propellant is needed because of the increase in specific impulse even though the tank mass does not directly scale [11].

However, with these downscaling advantages comes drawbacks such as the reduction of the particle residence time in the channel resulting in the flow energy's partial equilibration [11]. While the relaxation time is greater than the residence time, the flow's thermal energy undergoes incomplete conversion to kinetic energy leading to lower performance and it is termed frozen losses, which might also refer to the energy for breaking molecular bonds, where in this case it would not be converted to fluid kinetic energy as molecular recombination time is insufficient in the nozzle [11]. Over the nozzle, the flow is then at a lower energy state fixed/frozen composition [11]. As the collision frequency depends on the gas density and residence time or scale, the mentioned losses depend on $Re$ [11]. As heat transfer is enhanced from the heating elements, the flow is just as likely to transfer the heat at another colder location downstream [11]. Fabrication complications also arise, as the local surface roughness becomes relatively larger than for larger geometries [11], which adds insight into the differences between MEMS and conventional thrusters.

Considering the greater influence of wall roughness, it should be considered to avoid inducing shock waves, which lead to a pressure spike followed by less momentum [11]. By generalizing shock formation through viscosity:

$$\mu \sim \rho c \lambda, \tag{2.3}$$

where $c$ is the molecular speed [11]. Then, nondimensionalize by roughness height, $l$:

$$\frac{\mu}{\rho u l} \sim \frac{c}{u}\frac{\lambda}{l} \quad \text{or} \quad Ma \sim ReKn, \tag{2.4}$$

where $M$ is the Mach number, $u$ is the local velocity, and $\rho$ is the local density. The Mach number needs to be kept less than one to avoid shock formation, which would also require the roughness height not to enter the flow's supersonic region and remain inside the subsonic boundary layer [11]. Heat transfer also becomes more significant with the larger surface area to volume ratio at this small scale. Therefore, due to the thermal, viscous, and rarefied effects, directly miniaturizing macroscale propulsion systems would be rather inexpedient.

## 2.2. Rarefied Flow Modeling

To find the degree of rarefaction, the dimensionless Knudsen number needs to be evaluated:

$$Kn = \frac{\lambda}{L} = \lambda\frac{|\nabla\rho|}{\rho} = \sqrt{\frac{\gamma\pi}{2}}\frac{M}{Re}, \tag{2.5}$$

where $\lambda$ is the molecules' mean free path, $L$ is the flow's representative physical length scale, which is often considered the thermal or mass transport gap length, $\gamma$ is the specific heat ratio, $M$ is the Mach number, $Re$ is the Reynolds number, and $\rho$ is density and is often used to clarify the characteristic dimension, though density can be proven to be related to velocity and temperature and they are also less commonly used [17, 23]. The mean free path is calculated as:

$$\lambda = \frac{1}{\sqrt{2}\pi D_c^2 N_d} = \frac{R_A T}{\sqrt{2}\pi D_c^2 N_A p} = \frac{\mu}{p}\sqrt{\frac{\pi k T}{2M_w}}, \tag{2.6}$$

where $D_c$ is the gas molecular diameter, $N_d$ is the number density [11], $\mu$ is the viscosity, $p$ is the pressure, $T$ is the temperature, $M_w$ is the molecular mass, $k$ is the Boltzmann constant, $R_A$ is the universal gas constant, and $N_A$ is Avogadro's number. Along with using the Knudsen number as a breakdown parameter, continuum approach applicability could evaluated using its breakdown parameter ($B$), which is considered critical at a value of 0.05 though wall-slip and proper boundary conditions could still be needed below that, as it is derivable from the relation of Boltzmann and Navier-Stokes equations:

$$B = M\sqrt{\frac{\pi\gamma}{8}}\frac{\lambda}{\rho}\left|\frac{d\rho}{ds}\right|, \tag{2.7}$$

where $M$ is the Mach number, $\rho$ is the density, $\lambda$ is the local mean free path, $s$ is the distance along the streamline, and $\gamma$ is the ratio of specific heats at constant pressure ($C_p$) and volume ($C_v$) respectively [29]:

$$\gamma = \frac{C_p}{C_v} \tag{2.8}$$

The maximum value for the magnitudes of normalized stress tensor ($\tau_{ij}^*$) and heat flux ($q_i^*$) can also be considered for the equilibrium breakdown parameter, which is considered to occur at 0.1, evaluated using:

$$B = \max(|q_i^*||\tau_{ij}^*|), \tag{2.9}$$

$$q_i^* = \left(\frac{K}{p}\right)\left(\frac{2m}{kT}\right)^{0.5}\nabla T, \tag{2.10}$$

$$\tau_{ij}^* = \left(\frac{\mu}{p}\right)(\mathbf{V}_{i,j} + \mathbf{V}_{j,i} - \mathbf{V}_{k,k}\delta_{ij}), \tag{2.11}$$

where, the indices $i$, $j$, and $k$ represent the three velocity components, $k$ is the Boltzmann constant, $\mu$ is the dynamic viscosity, $T$ is the temperature, $m$ is the mass, $\delta_{ij}$ is the Kronecker delta, and $\mathbf{V}$ is the velocity [18]. A typical value for considerable flow rarefaction is at $Kn \sim 0.01$, which is when the conventional Navier-Stokes equations become noticeably less accurate [47]. The Boltzmann equation can be used fundamentally, though its implementation is generally time-consuming and complex. Figure 2.1 demonstrates the Knudsen number's effect on the flow regime [52].

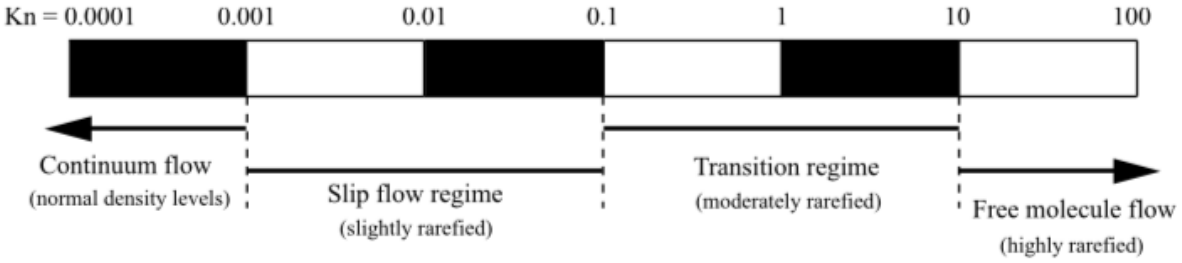



Figure 2.1: Flow regimes rarefaction by Knudsen number [52]

The flow is considered to be inviscid and in thermodynamic equilibrium around $Kn \sim 0$, where intermolecular collisions are dominant. The degree of rarefaction increases with increasing Knudsen number due to fewer molecular collisions. At high Knudsen numbers, namely free molecule flow regime, molecular collisions become negligible, which along with molecule-surface interactions lead to non-equilibrium. Note that velocity slip, temperature jump, and thermal diffusion/transpiration are considerable nonequilibrium effects along with noticed internal flow nonlinear pressure gradients [54].

Euler fluid equations are the most suitable numerical method for the inviscid flow regime, Navier-Stokes-Fourier (NSF) fluid equations for the continuum and slip (as well as velocity-slip and temperature-jump boundary conditions) flow regimes, and Boltzmann equation or particle methods (DSMC) for transition and free molecule (collisionless) flow regimes [52].

Continuum and Inviscid Flow Regime

The NSF equations for continuum-fluids prove to be highly accurate near thermodynamic equilibrium, as local macro-properties are assumed to be averaged over portions that are considered large compared to the fluid's microscopic structure and small for macroscopic phenomena allowing for the application of differential calculus. In other words, the macroscopic time scale is considerably greater than the time of dominant intermolecular collisions, which lead to local thermodynamic equilibrium. Around $Kn \sim 0$, the NSF equations' molecular diffusion is considered negligible along with the continuum momentum and energy equations' transport terms [52].

Slip Flow Regime

Non-equilibrium regions emerge near surfaces as their interactions with molecules decrease with an increasing Knudsen number. Macroscopically, the gas and surface's velocities and temperatures do not become equal, which translates to velocity-slip and temperature-jump respectively [52]. Certain methods can be used to study this non-equilibrium, with extended NSF equations using Maxwell's velocity-slip and Von Smoluchowski's temperature-jump boundary conditions [52].

Transition and Free Molecule Flow Regimes

At higher Knudsen numbers in transition and free molecule flow regimes, the linear constitutive relations of NSF equations become invalid due to the inaccurate assumptions of continuum and local equilibrium. Therefore, the need for solving the Boltzmann equation arises. For a single-species monatomic non-reacting gas, the Boltzmann equation is as follows:

$$\frac{\partial(nf)}{\partial t} + \mathbf{c}\frac{\partial(nf)}{\partial \mathbf{r}} + \mathbf{F}\frac{\partial(nf)}{\partial \mathbf{c}} = J(f, f^*), \tag{2.12}$$

where $n$ is the number density, $f$ is the normalized molecular velocity distribution function, $\mathbf{r}$ is a molecule's position vector, $\mathbf{c}$ is a molecule's velocity vector, $\mathbf{F}$ is the external force, $J(f, f^*)$ describes binary collisions as a nonlinear integral, and $^*$ stands for post-collision properties [52]. In addition, the collision integral is:

$$J(f, f^*) = \int_{-\infty}^{+\infty} \int_0^{4\pi} \pi^2 (f^* f_1^* - f f_1) c_r \sigma_T \, d\Omega \, d\mathbf{c}_1, \tag{2.13}$$

where $f$ and $f_1$ are the velocity distribution functions at $\mathbf{c}$ and $\mathbf{c}_1$ respectively, $c_r$ is two colliding molecules' relative speed, $\sigma_T$ is the molecular cross section, and $\Omega$ is the solid collision angle [52]. To note, numerical modeling of the transition regime is most challenging, as the mean free path is considerable compared to the characteristic length scale and molecular collisions are still highly applicable [54].

## 2.3. From Boltzmann to Navier-Stokes

Without time-averaged equations like Reynolds-averaged Navier-Stokes equations (RANS) and turbulence models, it is difficult to numerically solve the Navier-Stokes equations for turbulent flows considering the large variation in turbulent mixing-length scales, which would need an unfeasibly fine mesh. Based on kinetic theory and molecular chaos, DSMC is a powerful tool that can simulate molecular to hydrodynamic length scales including turbulence and its decay [21]. Turbulence is a time-dependent chaotic phenomenon with the idea that it happens due to the inertial forces dominating the viscous forces in the Reynolds number, where time-dependent and convective acceleration is significant, and considering the expected dominance of the viscous forces for the flow at this small scale micropropulsion application, the flow regime is expected to be laminar. The previous information is provided as an example of DSMC capabilities. To further elaborate on kinetic theory and its applicability for the flow in microthrusters, the Navier-Stokes equations can be derived from the Boltzmann equation in an approach unlike the Navier-Stokes derivations usually studied in engineering fluid mechanics courses. The derivation will explain the reasoning and conditions behind the estimations of the Boltzmann equation using Navier-Stokes equations.

The Boltzmann equation for an ideal monatomic gas assumedly dilute enough for the dominance of binary collisions and without external forces is:

$$\frac{\partial (nf)}{\partial t} + c_k \frac{\partial (nf)}{\partial x_k} = \left[\frac{\partial (nf)}{\partial t}\right]_{\text{coll}}, \tag{2.14}$$

where $n$ is the number density, $f$ is the velocity distribution function, $c_k$ is the molecular velocity in an inertial frame, the subscript $k$ represents a sum, and the subscript coll represents the collision integral [8]. The Boltzmann equation is multiplied by any function of molecular velocity $Q(c_i)$ and integrated over velocity space to find the moment equations:

$$\frac{\partial \, (n < Q >)}{\partial t} + \frac{\partial \, (n < c_k Q >)}{\partial x_k} = \Delta[Q], \tag{2.15}$$

where $< Q >$ and $\Delta[Q]$ are operators determined by:

$$< Q >= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} Q f \, \mathrm{d}c_1 \mathrm{d}c_2 \mathrm{d}c_3 \tag{2.16}$$

$$\Delta[Q] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} Q \left[\frac{\partial (nf)}{\partial t}\right]_{coll} \mathrm{d}c_1 \mathrm{d}c_2 \mathrm{d}c_3 \tag{2.17}$$

By selecting one of the five collisional invariants:

$$Q^{INV} = m\{1, c_i, c^2/2\}, \tag{2.18}$$

where $m$ is the molecular mass and $c^2$ is the velocity magnitude squared, as the arbitrary function of the molecular velocity $Q(c_i)$, the relative moment of the collision integral is similarly equal to zero, as $\Delta[Q] = 0$, which is applicable to any distribution function and molecular interaction law [8]. The gas dynamics conservation laws can then be determined:

$$\frac{\partial \left(n < Q^{INV} >\right)}{\partial t} + \frac{\partial \left(n < c_k Q^{INV} >\right)}{\partial x_k} = 0 \tag{2.19}$$

Then, with the collisional invariants from Equation 2.18:

$$\frac{\partial(\rho)}{\partial t} + \frac{\partial(\rho < c_k >)}{\partial x_k} = 0, \tag{2.20}$$

$$\frac{\partial(\rho < c_i >)}{\partial t} + \frac{\partial(\rho < c_k c_i >)}{\partial x_k} = 0, \tag{2.21}$$

$$\frac{\partial\left(\rho < c^2/2 >\right)}{\partial t} + \frac{\partial\left(\rho < c_k c^2/2 >\right)}{\partial x_k} = 0, \tag{2.22}$$

where $\rho$ is the mass density given as:

$$\rho = mn \tag{2.23}$$

For the thermal velocity components:

$$C_i = (c_i - u_i), \tag{2.24}$$

where the mean or fluid velocity is:

$$u_i =< c_i > \tag{2.25}$$

The central moments are then:

$$P_{ij} = \rho < C_i C_j >, \tag{2.26}$$

$$p = P_{kk}/3, \tag{2.27}$$

$$\tau_{ij} = -P_{ij} + p\delta_{ij}, \tag{2.28}$$

$$e =< C^2/2 >, \tag{2.29}$$

$$q_i = \rho < C_i C^2/2 >, \tag{2.30}$$

where $P_{ij}$ is the stress tensor, $p$ is the pressure, $\tau_{ij}$ is the viscous stress tensor, $\delta_{ij}$ is the Kronecker delta, which is equal to one for $i = j$ and zero for $i \neq j$, and for a monoatomic gas, $e$ and $q_i$ are the internal energy (translational) and heat flux vector respectively [8]. Then for Equations 2.20 to 2.22, substitute Equations 2.26 to 2.30 to obtain the gas dynamics conservation laws:

$$\frac{\partial(\rho)}{\partial t} + \frac{\partial(\rho u_k)}{\partial x_k} = 0, \tag{2.31}$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_k u_i - \tau_{ki} + p\delta_{ki})}{\partial x_k} = 0, \tag{2.32}$$

$$\frac{\partial\left[\rho\left(e + \frac{u^2}{2}\right)\right]}{\partial t} + \frac{\partial\left[\rho u_k\left(e + \frac{u^2}{2}\right) - \tau_{ki}u_i + p\delta_{ki}u_i + q_k\right]}{\partial x_k} = 0, \tag{2.33}$$

Furthermore, Equations 2.31 to 2.33 can be reformulated for axisymmetric flows in cylindrical coordinators $(r, \theta, z)$ and considering $u_\theta = \frac{\partial}{\partial\theta} = 0$:

$$\frac{\partial(\rho)}{\partial t} + \frac{\partial(\rho u_r)}{\partial r} + \frac{\rho u_r}{r} + \frac{\partial(\rho u_z)}{\partial z} = 0 \tag{2.34}$$

$$\frac{\partial(\rho u_r)}{\partial t} + \frac{\partial\left(\rho u_r^2 + p - \tau_{rr}\right)}{\partial r} + \frac{\rho u_r^2 - \tau_{rr}}{r} + \frac{\partial(\rho u_r u_z - \tau_{rz})}{\partial z} + \frac{\tau_{\theta\theta}}{r} = 0 \tag{2.35}$$

$$\frac{\partial\left(\rho u_z\right)}{\partial t}+\frac{\partial\left(\rho u_r u_z-\tau_{rz}\right)}{\partial r}+\frac{\rho u_r u_z-\tau_{rz}}{r}+\frac{\partial(\rho u_z^2+p-\tau_{zz})}{\partial z}=0 \quad 2.36$$

$$\frac{\partial\left[\rho\left(e+\frac{u^2}{2}\right)\right]}{\partial t}+\frac{\partial\left[\rho u_r\left(e+\frac{u^2}{2}\right)+pu_r-\tau_{rr}u_r-\tau_{rz}u_z+q_r\right]}{\partial r}+\frac{\left[\rho u_r\left(e+\frac{u^2}{2}\right)+pu_r-\tau_{rr}u_r-\tau_{rz}u_z+q_r\right]}{r}+\frac{\partial\left[\rho u_z\left(e+\frac{u^2}{2}\right)+pu_z-\tau_{rz}u_r-\tau_{zz}u_z+q_z\right]}{\partial z}=0 \quad 2.37$$

The model presented above needs to be adjusted before being applied to polyatomic gases, which requires changing Equation 2.14 with an applicable substitute [8]. The collisional invariants also changes as the $mc^2/2$ energy term becomes inaccurate for a particle that has an internal structure:

$$Q^{INV}=\{m,mc_i,(mc^2/2+\epsilon)\}, \quad 2.38$$

where $\epsilon$ is the additional internal energy, over which another integral is needed when applying Equation 2.14, which is assumed to remain valid for the extended distribution function $f(c_i,\epsilon)$, to Equations 2.16 and 2.17 [8]. Therefore, Equation 2.19 is still applicable considering that in a collision, Equation 2.38's quantities are still conserved, which results in an equivalent value of zero for Equation 2.17. As the integration over $\epsilon$ is independent of the $c_i$ integration, the quantities that only include $c_i$ yield the same results from Equation 2.17 to Equations 2.20 and 2.21 and their subsequent Equations 2.31 and 2.32. In a similar way as applied previously, Equation 2.22 can be found:

$$\frac{\partial\left(\rho<c^2/2>+n<\epsilon>\right)}{\partial t}+\frac{\partial\left(\rho<c_k c^2/2>+n<c_k\epsilon>\right)}{\partial x_k}=0 \quad 2.39$$

The unknown additional internal energy is designated as $me_{int}=<\epsilon>$, where $e_{int}=0$ for a monoatomic gas [8]. As a relatively straightforward method, the internal molecular energy modes are all assumed to be in equilibrium internally and with the translational degrees of freedom. $e_{int}$ is then represented by the equilibrium relation using the translational temperature $T$:

$$e_{int}=\frac{1}{2}\left(\frac{5-3\gamma}{\gamma-1}\right)RT, \quad 2.40$$

where $R$ is the gas constant and $\gamma$ is the ratio of specific heats, through which the additional internal energy is considered [8]. Substituting Equations 2.26 to 2.30 into Equation 2.19 results in Equations 2.31 and 2.32 and:

$$\frac{\partial\left[\rho\left(e+e_{int}+\frac{u^2}{2}\right)\right]}{\partial t}+\frac{\partial\left[\rho u_k\left(e+e_{int}+\frac{u^2}{2}\right)+P_{ki}u_i+q_k+(n<C_k\epsilon>)\right]}{\partial x_k}=0, \quad 2.41$$

which could have resulted in Equation 2.33 if the these two equations substituted Equations 2.29 and 2.30 respectively:

$$e=(<C^2/2>+e_{int}), \quad 2.42$$

$$q_i=\rho<C_i C^2/2>+n<C_i\epsilon>, \quad 2.43$$

which means that for a gas with internal structure and state of equilibrium between the internal modes and translation degrees of freedom, Equations 2.31 to 2.33 are applicable when Equations 2.42 and 2.43 are respectively substituted in the respective central moments [8].

The kinetic theory derivation approach is to be further used, as the more general conservation Equations 2.31 to 2.33 are phenomenologically derivable for any basic fluid. Considering the case

of an ideal gas flow, kinetic theory is needed, as it proves that the resulting equations apply for any degree of translational nonequilibrium, such as any translational velocity distribution function [8]. For an equilibrium distribution such as the Maxwellian distribution $f^{Max}$[2; 42], the set of equations results in the Euler equations, while for a Chapman-Enskog distribution, which is an approximate Boltzmann equation solution for a simple gas, $f^{CE}$[39; 43; 44], the set of equations results in the Navier-Stokes equations. The CE distribution is represented by a local Maxwellian and thermal velocity components $C_i$ polynomial function:

$$f^{CE} = f^{Max}(1 + \phi_1 + \phi_2), \tag{2.44}$$

where

$$f^{Max} = (2\pi RT)^{-3/2} \exp(-C^2/2RT), \tag{2.45}$$

$$\phi_1 = -\left(\frac{\rho}{p^2}\right)\left(K^{(1)}\frac{\partial T}{\partial x_k}\right)C_k(C^2/5RT - 1), \tag{2.46}$$

$$\phi_2 = -\left(\frac{\rho}{p^2}\right)\left(\mu^{(1)}\frac{\partial u_j}{\partial x_k}\right)\left(C_jC_k - \frac{C^2\delta_{jk}}{3}\right), \tag{2.47}$$

where $\delta_{jk}$ is the Kronecker delta, which is equal to one for $j = k$ and zero for $j \neq k$, $K^{(1)}$ is the thermal conductivity coefficient, and $\mu^{(1)}$ is the viscosity coefficient found from the first-order Champan-Enskog procedure ($C^2 = C_kC_k$) [8]. The stress and heat flux can be determined by their relative Chapman-Enskog expressions upon the selection of $f^{CE}$:

$$q_i^{CE} = -K^{(1)}\frac{\partial T}{\partial x_i}, \tag{2.48}$$

$$\tau_{ij}^{CE} = \mu^{(1)}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu^{(1)}\left(\frac{\partial u_k}{\partial x_k}\right)\delta_{ij}, \tag{2.49}$$

where Equation 2.49 can be reformulated for axisymmetric flows in cylindrical coordinators $(r, \theta, z)$:

$$\tau_{rr} = \mu^{(1)}\left[2\frac{\partial u_r}{\partial r} - \frac{2}{3}\left(\frac{\partial u_r}{\partial r} + \frac{u_r}{r} + \frac{\partial u_z}{\partial z}\right)\right] \tag{2.50}$$

$$\tau_{\theta\theta} = \mu^{(1)}\left[2\frac{u_r}{r} - \frac{2}{3}\left(\frac{\partial u_r}{\partial r} + \frac{u_r}{r} + \frac{\partial u_z}{\partial z}\right)\right] \tag{2.51}$$

$$\tau_{zz} = \mu^{(1)}\left[2\frac{\partial u_z}{\partial z} - \frac{2}{3}\left(\frac{\partial u_r}{\partial r} + \frac{u_r}{r} + \frac{\partial u_z}{\partial z}\right)\right] \tag{2.52}$$

$$\tau_{rz} = \mu^{(1)}\left(\frac{\partial u_z}{\partial r} + \frac{\partial u_r}{\partial z}\right) \tag{2.53}$$

For a fully specified $f$, any translational velocity distribution function in Equation 2.19, Equations 2.20 to 2.22, and Equations 2.31 to 2.33 can be selected due to set closure [8]. For a general $f$, $\tau_{ij}$ and $q_i$ become unknown quantities due to a problem with closure. The obtained Navier-Stokes equations are only valid for low Knudsen numbers, as $f^{CE}$ is an $\mathcal{O}(Kn)$ expansion of $f$'s solution. At higher $Kn$, the Navier-Stokes equations are incapable of explaining real gas behavior, as they do not show wall-slip and wall-temperature jumps.

## 2.4. Direct Simulation Monte Carlo (DSMC)

In the 1960s, Bird initially developed today's main method to solve for rarefied flow numerically, Direct Simulation Monte Carlo, to avoid the Boltzmann equation's direct application with a probabilistic approach [52]. Bird's 1994 and 2014 monographs clearly explain the DSMC method, as in 1992, the DSMC method was proven to mathematically solve the Boltzmann equation for a monatomic gas limited by an infinite particle number [52]. It is considered a particle method, where many real molecules or atoms are represented by fewer simulation particles. Conventional computational grid cells are generated for the studied geometry. The particles' motion is uncoupled from their collisions boosting the computational efficiency for capturing the flow physics. Following the evaluation of their free motion considering their independent velocities and the local time step, the collisions are then probabilistically simulated per grid cell.

### 2.4.1. DSMC Models Overview

A brief overview of different DSMC models is presented in this subsection, as intermolecular collision simulations' accuracy is highly dependent on the collision model used. The hard-sphere model is the simplest molecular model, but the variable hard-sphere, variable soft-sphere, generalized hard-sphere, generalized soft-sphere and variable sphere models were more elaborate models also developed to enhance the simulation accuracy [45]. Additionally, the statistical inelastic cross section (SICS) models for continuous and discrete internal energy, dynamic molecular collision (DMC) model, and Larsen-Borgnakke (LB) model were introduced for polyatomic molecules, where energy transfer can occur over translational, rotational, and vibrational degrees of freedom [45]. None of the mentioned models are effectively universal, as models are application-specific and their beneficial accuracy could be countered by a larger computational load and limitations [45]. Considering their lone translational degree of freedom, monoatomic flows are relatively easier for DSMC to simulate, applying useful molecular models such as variable hard-sphere and variable soft-sphere [45]. Phenomenological energy exchange models, such as Larsen-Borgnakke, are useful for polyatomic molecules [45].

The models are introduced in order based on their relative level of intricacy. The hard-sphere model's simplicity is drawn back by its unrealistic representation [45]. Based on the inverse power law (IPL) potential, where the intermolecular repulsive force is solely accounted for, the variable hard-sphere model involves a variable collision cross section, though it abides by the hard-sphere model's isotropic scattering law [45]. The variable soft-sphere model, which is also based on the IPL potential, was developed to make up for the variable hard-sphere's disadvantages [45]. At lower temperatures, the intermolecular attractive force is very important, though it is ignored due to its added computational and mathematical complexity [45].

Therefore, more accurate models including both intermolecular attractive force and intermolecular repulsive force were developed based on intermolecular potentials [45]. The generalized hard-sphere model, which is based on the Lennard-Jones (LJ) attractive-repulsive potential in a similar way to how the conventional variable hard-sphere and variable soft-sphere relate to the IPL potential, uses the hard-sphere model's scattering law, though the total cross section variation as a function of relative translational energy is represented by approximated parameters by the measured real gas transport properties' least-square curve data fitting [45]. The generalized soft-sphere model, which is closest to a universal collision model, computes the collision cross section as in the generalized hard-sphere model, though it uses a soft-sphere model's scattering law and is based on the Stockmayer potential, which is based on the LJ potential, but can handle polar molecules' polarization in addition to the nonpolar molecules in rarefied flows [45]. The generalized hard-sphere model and generalized soft-sphere model involve a significant computational load, so they are not too commonly applied, but more computationally efficient modifications exist and could be promising with development [45]. The variable sphere model, which has been proven efficient for rarefied flow DSMC, employs a generally different strategy assuming that instead of every molecular collision's scattering law, the diffusion and viscosity cross sections influence the macroscopic transport phenomena and properties [45]. The variable sphere model is also more general, as the total collision cross section and deflection angle expressions could in theory be derived by using any realistic intermolecular potential [45].

For polyatomic molecules, the most common DSMC energy exchange scheme is of the LB model, as it has been further developed to work with the discrete vibrational levels and electronic excitations using a quantum approach [45]. The macroscopic chemistry method has been developed to become

the most used method in chemical-reaction modeling for DSMC rarefied flows with chemistry in near equilibrium and nonequilibrium [45].

The above methods have been used in DSMC simulations with success, though the variable hard-sphere and variable soft-sphere models in addition to the LB model for energy exchange are the main models implemented, as they provide relatively convenient numerical computations with practical accuracy [45]. For the scope of work considered, the DSMC model used, as part of dsmcFoam, is explained in Subsection 2.4.2.

### 2.4.2. OpenFOAM dsmcFoam(+) Solver

OpenFOAM is an open-source C++ toolbox for numerical solvers and pre/post-processing utilities that incorporates computational fluid dynamics (CFD). For achieving the comparison between MEMS and conventional thrusters for small spacecraft micropropulsion, dsmcFoam, which is a popular DSMC solver designed in object-oriented C++ for OpenFOAM, will be used and modified as needed, though other DSMC software is also available. It is a solver that can handle transient multi-species flows. The main unique features of this explicit time-stepping stochastic particle-based solver include its complementary open-source nature, practical ability to run unlimited parallel simulations, and freedom of 2D and 3D geometrical application. The dsmcFoam solver is based on adjustments from a relatively more computationally expensive pre-existing deterministic molecular dynamics (MD) OpenFOAM solver for the individual particles' classical (Newton's) equations of motion and works similarly to other solvers despite the uncommon Lagrangian nature of DSMC in OpenFOAM. In addition to the MD solver's main capabilities, arbitrary geometries can be used for particle initialization and unstructured arbitrary polyhedral meshes for particle tracking [47]. Its explicit time-stepping model including stochastic molecular collisions is suitable for rarefied flows [52]. The solver's code is structured hierarchically with the coding block ability for derived classes to directly inherit base classes. It can also be combined with other solvers. Some of dsmcFoam's features include:

- Steady and transient models [47].
- Unlimited parallel simulations [47].
- Arbitrary 2D and 3D geometries [47].
- Sub-cell generation promoting nearest neighbor collisions automatically [47].
- Arbitrary gas species number [47].
- Variable hard sphere (VHS) collision and Larsen–Borgnakke internal energy redistribution models [47].
- Freestream flow, diffuse/specular wall reflection, and periodic boundaries [47].

dsmcFoam+ (dsmcFoamPlus), a newer version of the same dsmcFoam solver, but is not included in the standard updated OpenFOAM package and needs to be installed additionally, includes all the functionalities of the original one plus subsonic pressure boundary conditions, chemical reactions, molecular vibrational and electronic energy modes, arbitrary axisymmetric geometries, gravitational force controller, mass flow rate measurement, simulation quality reports, and dynamic load balancing, which is controlled in [case]/system using balanceParDict and loadBalanceDict and allowing the computation of the maximum present level of parallel load imbalance $L_{max}$ [52]. To clarify, dsmcFoam has been available with the official release of OpenFOAM for a while (since OpenFOAM version 1.7), but additional and dsmcFoam related developers (Micro & Nano Flows Group) separately improved the solver under the name of dsmcFoam+ and did not include it in any of the official OpenFOAM releases yet. The scope of this thesis focuses on generally sufficient features already available and attainable in dsmcFoam, though dsmcFoam+, which is fundamentally based on dsmcFoam, allows using the latest and most useful techniques to achieve higher quality results, so it is ultimately used.

The directory structure of dsmcFoam with the submodel source code locations is shown in Figure 2.2 [47]. Concerning the general solution, the mesh is generated for the given geometry. From experience with DSMC, the grid cell ($\Delta x_{cell}$) and time step ($\Delta t_{step}$) sizes should generally be fractions of the mean free path and mean collision time respectively [47]. Their approximations could be found as:

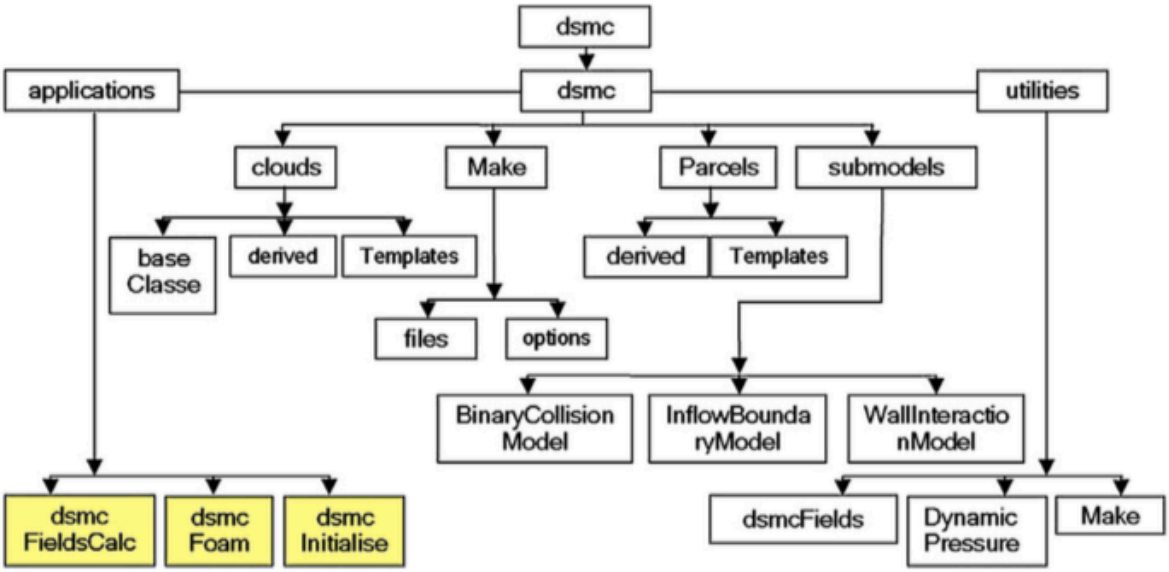



Figure 2.2: dsmcFoam directory structure [47]

$$\Delta x_{cell} \leq \frac{\lambda}{3}, \tag{2.54}$$

$$\Delta t_{step} = \frac{\xi \lambda}{\bar{c}}, \tag{2.55}$$

where $\xi$ is a fraction for the time step size to be a fraction of the calculated mean free time, which is the mean time between collisions, at mean stream conditions and $\bar{c}$ is the mean thermal speed [26]. To obtain the mean free path ($\lambda$) for the variable hard sphere (VHS) molecules used, Equation 2.56 is used, while Equation 2.57 is used for the mean thermal speed.

$$\lambda = \frac{2(5-2\omega)(7-2\omega)}{15}\left(\frac{m}{2\pi k T}\right)^{\frac{1}{2}}\left(\frac{\mu}{\rho}\right) \tag{2.56}$$

$$\bar{c} = \sqrt{\frac{8kT}{\pi m}} \tag{2.57}$$

Here, $\omega$ is the temperature coefficient of viscosity, $m$ is the atomic mass given by dividing the molar mass ($M$) by Avogadro's number ($N_A$), $k$ is the Boltzmann constant, $T$ is the temperature, $\mu$ is the dynamic viscosity, and $\rho$ is the density [47]. For proper movement and collision steps decoupling, the time step size should be smaller than the mean collision time, which is the successive particle collision time on average [54]. The time step size must not allow the DSMC particles to skip grid cells at the most probable molecular speed for sufficient interaction with other particles. The Courant-Friedrichs-Lewy number ($CFL$) can be defined in DSMC to physically provide a time step size allowing particles to remain within their grid cell ensuring sufficient interaction for accuracy rather than CFD's general usage for stability [35]:

$$CFL = \frac{c'_m \Delta t_{step}}{\Delta x_{cell}} < 1, \tag{2.58}$$

where $c'_m$ is the most probable molecular speed given as:

$$c'_m = \sqrt{\frac{2R_A T}{M_w}}, \tag{2.59}$$

where $R_A$ is the universal gas constant and $M_w$ is the molecular mass. Then, the boundary conditions and flow properties are set. The utilities for dsmcFoam are the pre-processing tool dsmcInitialise generating the initial DSMC particle configurations in the geometry, DSMC solver dsmcFoam, and post-processing tool dsmcFieldsCalc that evaluates the intensive and extensive fields, which can also be run in parallel [47].

Initialization
First, the pre-processing and setting the geometries, particle velocities and types, such as species, mass, and internal energy parameters is performed. Then, the macroscopic parameters, temperature, velocity, and density, are defined with the possibility to assign different values along a nonuniform mesh. The dsmcFoam solver then places pre-located particles with energies and positions that macroscopically return the user's set parameters [52].

DSMC Model Algorithm

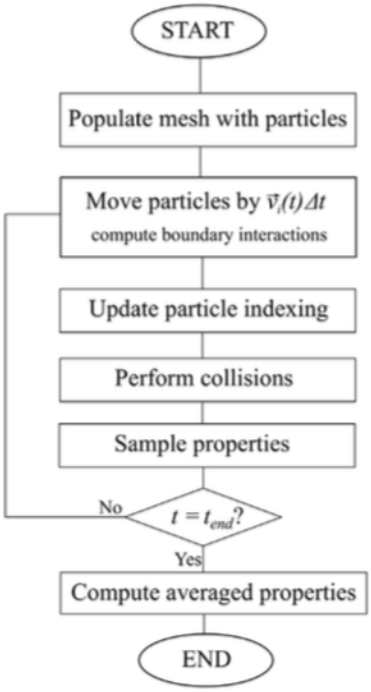



Figure 2.3: DSMC time-integration scheme flow chart [52]

Using an explicit time-stepping scheme, the particles position can be computed over time. For a $t \rightarrow t + \Delta t$ single time-step, all DSMC solvers use the algorithm shown in Figure 2.3 and explained below [52]:

1. OpenFOAM's built-in particle tracking algorithm, which manages the particle movements over the mesh faces and boundaries, is used to update the positions for all $N$ particles in the domain, where the move step for the $i$th particle is determined from its equation:

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\Delta t = \vec{r}_i(t) + \Delta\vec{r}_i, \tag{2.60}$$

   where $t$ is time and $\vec{v}$ is the velocity vector.

2. Every computational cell's list of particles is updated in preparation for the collision computations.

3. Calculate the number of collisions to apply for every computational cell.

4. Using the set binary collision model, the collisions are executed.

5. Particle positions, velocities, internal energies, etc. are sampled to provide the set macroscopic parameters.

6. Iterate from the first step using the new $t = t + \Delta t$ until $t_{end}$ is achieved.

Particle Tracking
OpenFOAM's built-in particle tracking algorithm determines when the discrete particles transfer between the mesh's cells and manages their interactions with the boundaries, securing that particles stay within the domain and allowing different elaborate boundaries. Instead of using $\Delta\vec{r}_i$ on particles immediately, the mesh's number of faces that are probably traversed is computed first. Afterwards, the particle tracking is applied to every face and cell index in order, where the properties of every face are evaluated after every intersection for checking the subsequent step. The particle moves on to the following position with no extra steps in the case of an internal face, but an additional action is done midstream of the move step if the face is a section of a boundary. For instance, the particle's normal velocity component direction is flipped if it touches a specular wall boundary and then it proceeds on its path [52].

Binary Collisions
Following the particle motion computations and their updated tracked positions, they are reindexed due to the collision and sampling routines' informational dependence on every cell's present occupancy. Next, the collisions are probabilistically evaluated. For maintaining an accurate number of collisions, one of the available methods mainly used in dsmcFoam is the no-time-counter (NTC) scheme developed by Bird [52]. The sub-cell techniques including newer transient-adaptive methods are utilized for stimulating near-neighbor collisions. The following equation for a given cell determines the probability ($P_{\text{collision}}$) of a particle $i$ to collide with particle $j$:

$$P_{\text{collision}}[i,j] = \frac{\left|\mathbf{c}_i - \mathbf{c}_j\right|}{\sum_{m=1}^{N_{cell}} \sum_{n=1}^{m-1} \left|\mathbf{c}_m - \mathbf{c}_n\right|}, \tag{2.61}$$

where $N_{cell}$ is the instantaneous number of DSMC particles in a cell [52]. However, an acceptance-rejection scheme is utilized to pick the colliding pairs, as computing it for every pair is inefficient . The NTC scheme chooses pairs in a cell at a given time step as:

$$\frac{1}{2V_C} F_N N(N-1)(\sigma_T c_r)_{\text{max}} \Delta t, \tag{2.62}$$

where $V_C$ is the volume of the cell, $_{\text{max}}$ represents the maximum value of all potential particle pairs in a cell, $\Delta t$ is the time step size, and $F_N$ is the corresponding number of actual atoms or molecules represented by individual DSMC particles [52]. For establishing near-neighbor collisions, particle $i$ is picked randomly from all of the cell's particles and $j$ is picked from the corresponding sub-cell. The acceptance-rejection method is then applied to every collision pair $ij$ to check if their collision is approved under the requirement:

$$\frac{(\sigma_T c_r)_{ij}}{(\sigma_T c_r)_{\text{max}}} > R_f, \tag{2.63}$$

where $R_f$ is a uniform random number picked between 0 and 1 [52]. When the colliding particle pair is determined, the collision takes place. Possible reactions are checked for the determined colliding particle pair in a chemical reaction model. The quantum-kinetic (QK) chemical reaction framework is performed for dsmcFoam+, permitting the occurrence of dissociation, exchange, and ionization reactions. Both velocities of the colliding particles in the pair are reset to simulate the collisions at their unchanged position. Considering elastic collisions, where particles do not transfer rotational or vibrational energy, the conservation of linear momentum occurs with a constant center of mass velocity, $\mathbf{c}_{cm}$:

$$\mathbf{c}_{cm} = \frac{m_i \mathbf{c}_i + m_j \mathbf{c}_j}{m_i + m_j} = \frac{m_i \mathbf{c}_i^* + m_j \mathbf{c}_j^*}{m_i + m_j} = \mathbf{c}_{cm}^*, \tag{2.64}$$

where $m$ stands for mass [52]. The conservation of energy occurs with retaining a constant relative velocity magnitude:

$$c_r = \left|\mathbf{c}_i - \mathbf{c}_j\right| = \left|\mathbf{c}_i^* - \mathbf{c}_j^*\right| = c_r^* \tag{2.65}$$

$\mathbf{c}_r^*$ could be solved for using a scattering angles equation along with Equations 2.64 and 2.65. The scattering angles $\theta$ and $\phi$ are uniformly distributed over a unit sphere in a VHS gas. The elevation angle $\theta$ is uniformly distributed within an interval of [-1,1] and can be found using the following equations:

$$\cos(\theta) = 2R_f - 1 \quad \text{and} \quad \sin(\theta) = \sqrt{1 - \cos^2(\theta)} \tag{2.66}$$

The azimuthal angle $\phi$ is uniformly distributed between 0 and $2\pi$ and can be found using the following equation:

$$\phi = 2\pi R_f \tag{2.67}$$

The post-collision relative velocity's three components can be calculated using:

$$\mathbf{c}_r^* = c_r^* \left[(\cos(\theta))\hat{\mathbf{x}} + (\sin(\theta)\cos(\phi))\hat{\mathbf{y}} + (\sin(\theta)\sin(\phi))\hat{\mathbf{z}}\right] \tag{2.68}$$

Therefore, the post-collision velocities can be found:

$$\mathbf{c}_i^* = \mathbf{c}_{cm}^* + \left(\frac{m_j}{m_i + m_j}\right)\mathbf{c}_r^* \quad \text{and} \quad \mathbf{c}_j^* = \mathbf{c}_{cm}^* - \left(\frac{m_i}{m_i + m_j}\right)\mathbf{c}_r^* \tag{2.69}$$

Inelastic collisions should be considered as well for the energy exchange between translational and rotational modes in diatomic molecules with rotational energy. The phenomenological Larsen-Borgnakke model is prevalent in DSMC for rotational energy exchange. A particular fraction of collisions is assessed inelastically to find the rotational relaxation rate. Specifically, dsmcFoam+ allows a user-defined rotationalRelaxation-CollisionNumber in [case]/constant/dsmcProperties as a constant rotational relaxation probability, $Z_{rot}$, where the rotational relaxation is checked as a collision is to occur and is accepted and a new rotational energy is set for the particle under the following condition [52]:

$$\frac{1}{Z_{rot}} > R_f \tag{2.70}$$

For energy conservation, a relative decrease in the collision pair's overall translational energy is performed and the post-collision relative speed $c_r^*$ is updated using the equation:

$$c_r^* = \sqrt{\frac{2\epsilon_{tr}}{m_r}}, \tag{2.71}$$

where $\epsilon_{tr}$ is the particle pair's overall translational energy following the rotational relaxation modification. The rest of the collision continues similarly from Equation 2.66 [52]. Additionally for energy exchange between vibrational and electronic to translational modes, dsmcFoam+ employs the quantum Larsen-Borgnakke model [52].

Sampling

After evaluating the collisions, the particle properties are sampled for time-averaging to then find the macroscopic flow properties, which is generally the main purpose of DSMC simulations. The computational cell number density (n) is evaluated as:

$$n = \frac{F_N \overline{N}}{V_C}, \tag{2.72}$$

where $\overline{N}$ is the time-averaged DSMC particle number in the cell within the measurement interval and $V_C$ is the volume of the cell. For a more elaborate explanation of finding the macroscopic fields from specific measurements, refer to Bird's 2013 book [13]. For a steady flow, a steady state simulation is permitted with properties evaluated across a sufficiently large sample size to reach an acceptable smaller statistical error, which is achievable using approximations from relations found in [28]. However, repeating the simulation is required until a sufficiently large sample size is reached for a transient flow, so that ensemble averaging can be done to obtain the results [52].

## 2.5. OpenFOAM Continuum Compressible Flow Solvers

This section is intended to provide a brief overview of applicable OpenFOAM continuum compressible flow solvers in case using dsmcFoam for the full model requires considerable improvement in computational time.

### 2.5.1. Governing Equations

For the continuum compressible flow solvers concerned, which differ in their numerical approach, the general governing fluid equations solved in an Eulerian frame of reference include mass conservation:

$$\frac{\partial(\rho)}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0, \tag{2.73}$$

Conservation of momentum (neglecting body forces):

$$\frac{\partial(\rho \mathbf{V})}{\partial t} + \nabla \cdot (\rho \mathbf{V}\mathbf{V}) = \nabla p + \nabla \cdot \underline{\tau}, \tag{2.74}$$

where $\rho$ is the mass density, $\mathbf{V}$ is the fluid velocity, $p$ is the pressure, and $\underline{\tau}$ is the viscous stress tensor from Boussinesq:

$$\underline{\tau} = 2\mu \mathrm{dev}(\underline{\mathbf{D}}), \tag{2.75}$$

where $\mu$ is the dynamic viscosity, $\underline{\mathbf{D}}$ is the deformation gradient tensor, and $\mathrm{dev}(\underline{\mathbf{D}})$ is its deviator component:

$$\underline{\mathbf{D}} = \frac{1}{2}[\nabla V + (\nabla V)^T], \tag{2.76}$$

$$\mathrm{dev}(\underline{\mathbf{D}}) = \underline{\mathbf{D}} - \frac{1}{3}\mathrm{tr}(\underline{\mathbf{D}})\mathrm{I}, \tag{2.77}$$

where $\mathrm{I}$ is the unit tensor [37]. The conservation of energy equations include balance equations using the sensible energy ($e_s$) and total nonchemical energy ($E$):

$$e_s = h_s - \frac{p}{\rho} = \int_{T_0}^{T} C_v dT - RT_0/W, \tag{2.78}$$

$$E = H - \frac{p}{\rho} = e_s + \frac{1}{2}\mathbf{V} \cdot \mathbf{V}, \tag{2.79}$$

where $h_s$ is the sensible enthalpy ($h_s = \int_{T_0}^{T} C_p dT$) and $H$ is the total enthalpy ($H = h_s + \frac{1}{2}\mathbf{V} \cdot \mathbf{V}$), so the energy balance equations are:

$$\frac{\partial(\rho e_s)}{\partial t} + \nabla \cdot [\mathbf{V}(\rho e_s)] + \nabla \cdot \mathbf{q} + (p\mathrm{I} - \underline{\tau})\nabla \cdot \mathbf{V} = 0 \tag{2.80}$$

$$\frac{\partial(\rho E)}{\partial t} + \nabla \cdot [\mathbf{V}(\rho E)] + \nabla \cdot \mathbf{q} + \nabla \cdot [(p\mathrm{I} - \underline{\tau})\mathbf{V}] = 0 \tag{2.81}$$

The working gas is assumed to act as a one component (frozen mixture composition) calorically perfect gas, which allows using the equations:

$$e_s = \left(C_p - \frac{R_A}{W}\right)T = C_v T, \tag{2.82}$$

$$T = \frac{1}{C_v}\left[\frac{(\rho R)}{\rho} - \frac{1}{2}\mathbf{V} \cdot \mathbf{V},\right], \tag{2.83}$$

$$p = \rho \frac{R_A}{W} T = \frac{\rho}{\psi}, \tag{2.84}$$

where $\psi$ is the fluid compressibility, $R_A$ is the universal gas constant, and $C_p$ and $C_v$ are the specific heats at constant pressure and volume respectively [37]. Fourier's law can be used for the heat flux ($\mathbf{q}$):

$$\mathbf{q} = -KT, \tag{2.85}$$

where $K$ is the conductivity and $T$ is the temperature. The conservation equations reduce to Euler's equations when the flow is inviscid for $\underline{\tau} = \mathbf{q} = 0$ [37].

In compressible flow Reynolds-averaged Navier-Stokes (RANS) equations, Favre's mass weighted averages are utilized for mean quantity conservation equations, where any dependent variable $f$ can be divided into mean and fluctuating components as $f = \tilde{f} + f''$. Favre specifically averages the product as $\widetilde{\rho f''} = 0$ instead of $f'' = 0$, resulting in a mean value of:

$$\tilde{f} = \frac{\overline{\rho f}}{\overline{\rho}} \tag{2.86}$$

Therefore, this averaging method can be used for the instantaneous conservation equations. The mass and momentum conservation equations respectively become:

$$\frac{\partial(\overline{\rho})}{\partial t} + \nabla \cdot (\overline{\rho}\tilde{\mathbf{V}}) = 0, \tag{2.87}$$

$$\frac{\partial(\overline{\rho}\tilde{\mathbf{V}})}{\partial t} + \nabla \cdot [(\tilde{\mathbf{V}}(\overline{\rho}\tilde{\mathbf{V}})] + \nabla\overline{p} - \nabla \cdot (\underline{\tau} + \underline{\tau_t}) = 0, \tag{2.88}$$

where $\underline{\tau_t} = 2\mu_t \mathrm{dev}(\underline{\tilde{\mathbf{D}}})$ [37]. Also, $\mu_{eff} = \mu + \mu_t$ can be used when the mean deformation gradient tensor $\tilde{\mathbf{D}}$ is used for the molecular viscous stress tensor along with the turbulent dynamic viscosity ($\mu_t$) for building the turbulent Reynolds stresses [37]. The conservation of energy equations neglecting molecular viscous heating become:

$$\frac{\partial(\overline{\rho}\tilde{e}_s)}{\partial t} + \nabla \cdot [\tilde{\mathbf{V}}(\overline{\rho}\tilde{e}_s)] - \nabla \cdot (\alpha_{eff}\nabla\tilde{h}_s) + \overline{p}\nabla \cdot \tilde{\mathbf{V}} = 0, \tag{2.89}$$

$$\frac{\partial(\overline{\rho}\tilde{E})}{\partial t} + \nabla \cdot [\tilde{\mathbf{V}}(\overline{\rho}\tilde{E})] - \nabla \cdot (\alpha_{eff}\nabla\tilde{h}_s) + \nabla \cdot (\overline{p}\tilde{\mathbf{V}}) = 0 \tag{2.90}$$

where $\alpha_{eff} = \alpha + \alpha_t$ is the effective thermal diffusivity considering its the turbulent and local mean molecular quantities [37]. The sensible enthalpy is $\tilde{h}_s = \tilde{e}_s + \frac{\overline{p}}{\overline{\rho}} = \tilde{E} + \frac{\overline{p}}{\overline{\rho}} - \frac{1}{2}\tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}$ [37].

## 2.5.2. Solver Selection

OpenFOAM is also chosen due to its versatility. In OpenFOAM, pressure (e.g. sonicFoam) or density (e.g. rhoCentralFoam) based solvers with segregated or coupled solutions for their governing equations are applied for compressible flow finite volume numerical solutions [37]. OpenFOAM applies coupled equations through a segregated method, which is an applicable method as long as the component coupling is not significant, as every dependent variable has an equation that is consecutively solved, so scalar equations are used for every component in solving vector equations which allows for convergence iterations [37].

sonicFoam, which is a transient solver that can handle transonic/supersonic turbulent compressible gas flow, uses a non-iterative approach for implicitly discretized time-dependent flow equations coupling [37]. The pressure implicit with splitting of operators (PISO) method, which works for incompressible and compressible forms of the transport equations, considers pressure and velocity as dependent variables [37]. The PISO method essentially divides the solution process into a series of steps with the decoupling of pressure and velocity operations, which would lead to equation sets that can be solved by standard methods. Following every PISO step, the fields become more accurate estimates of the difference equations' actual solution based on a general order of accuracy considering the operation-splitting number [37]. As the errors decrease and the general scheme's stability is only slightly reduced by the splitting procedure, iterations could be lowered, yet the implicit differencing advantage of allowing greater time steps is maintained [37]. Numerical methods for high-speed compressible flow should be

able to evaluate discontinuities like shocks and contact surface without misleading oscillations, which is applied by many effective methods that employ Riemann solvers, characteristic decomposition, and Jacobian evaluation for numerical flux determination, though it is challenging on a mesh of polyhedral cells with an arbitrary number of faces [37]. Therefore, central schemes, such as the central-upwind scheme in the rhoCentralFoam transient solver which can also handle turbulence, heat transfer, and compressible flow, are used for accurate solutions without Riemann solvers. A finite volume discretization is involved in rhoCentralFoam, which uses semidiscrete nonstaggered central schemes for colocated (defined at the same set of discrete locations, such as the cell centers) variables prescribed on a mesh of polyhedral cells with an arbitrary number of faces [24].

rhoCentralFoam proved better than sonicFoam in supersonic flow simulations in [37], considering its simplicity, up to three times fewer computational grid cells, lower computational time, and adequacy in interpolating flux for compressible flows where wave-transported properties are well accounted for along with velocity-transported properties. rhoCentralFoam's Minmod and van Albada limiters were outperformed by the van Leer limiter, which allowed for optimal balance for oscillation-free fields, shock capture, and needed computational grid cell number and cost. In the PISO loop, sonicFoam, which is developed from an initially incompressible approach, uses the equation of state to couple pressure and density leading to the need for the energy equation and applies decoupling of pressure and velocity, though supersonic flows are dominated by the local quantities' interactions [37]. However, these density based solvers become less accurate for more incompressible flows at $M < 0.3$, where the extended PISO method could become more accurate, even with compressibility effects from quick temperature distribution changes and large density variations possibly from chemical reactions [37].

Therefore, considering that the decision to use another solver is to reduce the computational load and since the modeled flow is mostly expected in the compressible subsonic regime mainly in the treated region from the inlet to the throat of the nozzle, the density based solver, rhoCentralFoam, would be chosen.

The rhoCentralFoam compressible flow Navier-Stokes solver algorithm is presented below, where $\mathbf{v}$ is the fluid velocity, $\hat{\mathbf{v}} = \rho\mathbf{v}$ is the momentum density, $E$ is the total energy, $\hat{E} = \rho E$ is the total energy density, $\mathbf{S}_f$ is a vector perpendicular to the face surface pointing outwards of the owner cell and its magnitude denotes the face's area with $f+$ and $f-$ being the directions of $+\mathbf{S}_f$ and $-\mathbf{S}_f$ respectively, $K$ is the conductivity, $p$ is the pressure, $T$ is the temperature, $\gamma$ is the specific heat ratio, and $\tau_{exp}$ is the viscous stress tensor, where explicit treatment is applied for its terms with intercomponent coupling $\left(\tau_{exp} = \mu\left[(\nabla\mathbf{u})^T - \left(\frac{2}{3}\right)\mathrm{tr}(\nabla\mathbf{v})I\right]\right)$ [24]. The overall computational method is more elaborately described in [24].

rhoCentralFoam Compressible Flow Solver Algorithm [24]

while $t < t_{end}$ do
  Set $t := t + \Delta t$
  Evaluate $\rho_{f\pm}$, $\hat{\mathbf{v}}_{f\pm}$, and $T_{f\pm}$ from $\rho$, $\hat{\mathbf{v}}$, and $T$ using van Leer limiter
  Calculate: $\mathbf{v}_{f\pm} = \hat{\mathbf{v}}_{f\pm}/\rho_{f\pm}$; $p_{f\pm} = \rho_{f\pm}RT_{f\pm}$; $\phi_{f\pm} = \mathbf{S}_f \cdot v_{f\pm}$; $c_{f\pm} = \sqrt{\gamma RT_{f\pm}}$
  Calculate convective derivatives and $\nabla p$ from $f\pm$ interpolates
  Update $\tau_{exp}$, $\mu$, and $K$
  Solve for $\rho$ using density equation
  Solve for $\hat{\mathbf{v}}$ using inviscid momentum prediction
  Update $\mathbf{v}$ from $\hat{\mathbf{v}}$ and $\rho$
  Solve for $\mathbf{v}$ using diffusive velocity correction
  Solve for $\hat{E}$ using inviscid energy prediction
  Update $T$ using $\hat{E}$, $\mathbf{v}$, and $\rho$
  Solve for $T$ using diffusive temperature correction
  Update $p$ by $p = \rho RT$
end while

## 2.6. Analytical Model

General parameter manipulations from the equations provided in this section can provide further basic insight into propulsion and micropropulsion theory and the effects of different parameter alterations.

### 2.6.1. Rocketry

The micropropulsion system analysis follows the same principles of larger propulsion systems, though the assumptions might differ, such as in the inability to neglect friction forces [50]. The propellant used for each thruster firing for maintaining attitude about a deadband, $\dot{W}_p$, in the case of a uniaxial undisturbed rotation of a constant mass vehicle is:

$$\dot{W}_p = \frac{l\,(\Delta I_{bit})^2}{4JI_{sp}\Theta_t}, \tag{2.91}$$

where $l$ is the moment arm, $I_{bit}$ is the minimum impulse from thruster firing, $J$ is the moment of inertia, and $\Theta_t$ is the angular position limit [11]. For attitude control, it is recommended to focus more on decreasing $I_{bit}$ than increasing the specific impulse, $I_{sp}$, where higher is better, as it measures the propellant usage effectiveness in units of seconds, though it is a thrust per propellant unit weight measurement and not time [50]:

$$I_{sp} = \frac{F_T}{\dot{m}g_0} = \frac{I}{mg_0} = \sqrt{\frac{2\gamma RT_c}{g^2(k-1)}}, \tag{2.92}$$

where $F_T$ is the thrust, $\dot{m}$ is the fuel consumption, $g_0$ is the surface gravitational acceleration [11], $\gamma$ is the specific heat ratio, $T_c$ is the chamber temperature [17], and $I$ is the total impulse:

$$I = \int_{t_1}^{t_2} F dt = \Delta p = mv_2 - mv_1, \tag{2.93}$$

where $p$ is momentum. Often, the system-specific impulse, $I_{ssp}$, is considered for evaluating propulsion performance:

$$I_{ssp} = \frac{I}{m}, \tag{2.94}$$

where $m$ is the system mass. The Tsiolkovsky rocket equation is used for self-propelled vehicles that make use of the conservation of momentum to apply thrust by jetting part of its mass:

$$\Delta v = v_e \ln\left(\frac{m_i}{m_f}\right), \tag{2.95}$$

where delta-$v$ ($\Delta v$) is a measure of impulse to perform a maneuver in the form of the maximum velocity change under the absence of other acting external forces, $m_i$ is the wet mass (initial mass including propellant), $m_f$ is the dry mass (final mass without propellant), and $v_e$ is the effective exhaust velocity:

$$v_e = I_{sp}g_0, \tag{2.96}$$

When the consumed propellant mass is outstandingly smaller than the spacecraft's overall mass $\left(\frac{m_p}{m_i} << 1\right)$, $\Delta v$ can be evaluated through the ideal rocket equation's (Equation 2.95) linearized approximation [27]:

$$\Delta v = g_0 I_{sp}\frac{m_p}{m_i}, \tag{2.97}$$

where $m_i$ is the spacecraft's initial mass and $m_p$ is the propellant mass.

The basic rocket thrust equation is:

$$F_T = \dot{m}v_e + (p_e - p_0)A_e, \tag{2.98}$$

where $\dot{m}$ is the mass flow rate, $v_e$ is the exit velocity, $p_e$ is the exit pressure, $p_0$ is the external pressure (assumed zero for vacuum in space), and $A_e$ is the exit area. The Mach number ($M$) can be evaluated as:

$$M = \frac{v}{c}, \tag{2.99}$$

where $v$ is the velocity and $c$ is the speed of sound in the medium:

$$c = \sqrt{\frac{\gamma p}{\rho}} = \sqrt{\frac{\gamma RT}{M}}, \tag{2.100}$$

where $M$ is the molar mass. For a uniform supersonic stream expansion around a corner to determine the flow characteristics, it is also possible to use continuum equations to calculate the angle after which vacuum occurs, as it results in an infinite Mach number for the region with vacuum. A molecular Mach number $S$ can also be defined as:

$$S = \frac{v}{\tilde{c}}, \tag{2.101}$$

where $v$ is the flow speed and $\tilde{c}$ is the random molecular motion's approximated thermal speed [23].

### 2.6.2. Continuum Flow Equations

The following equations mainly apply to the continuum flow regime. The classical relationships for continuum flow regime, energy conservation, and ideal gas consider the mass flow rate as:

$$\dot{m} = \frac{p_c A^*}{\sqrt{\frac{R_A}{M_w} T_c}} \Gamma = \rho v A, \tag{2.102}$$

where $p_c$ is the pressure of the chamber, $A^*$ is the area of the nozzle throat, $R_A$ is the universal gas constant, $M_w$ is the molecular mass, $T_c$ is the temperature of the chamber, and $\Gamma$ is the Vandenkerckhove function of the specific heat ratio ($\gamma$) [27]:

$$\Gamma = \sqrt{\gamma \left(\frac{1+\gamma}{2}\right)^{\frac{1+\gamma}{1-\gamma}}} \tag{2.103}$$

The nozzle expansion ratio, which is the exit area ($A_e$) to throat area ratio, is related to the pressure ratio, which is the exit pressure ($p_e$) to chamber pressure ratio [27]:

$$\frac{A_e}{A^*} = \frac{\Gamma}{\sqrt{\frac{2\gamma}{\gamma-1}\left(\frac{p_e}{p_c}\right)^{\frac{2}{\gamma}}\left[1-\left(\frac{p_e}{p_c}\right)^{\frac{\gamma-1}{\gamma}}\right]}} \tag{2.104}$$

The nozzle exit jet velocity ($v_e$) is [27]:

$$v_e = \sqrt{\frac{2\gamma}{\gamma-1}\frac{R_A}{M_W} T_C \left[1-\left(\frac{p_e}{p_c}\right)^{\frac{\gamma-1}{\gamma}}\right]} \tag{2.105}$$

The exhaust velocity can also be calculated as:

$$v_e = M_{exit}\sqrt{\gamma RT_{chamber}}, \tag{2.106}$$

where $\gamma$ equals [50];

$$\gamma = \frac{c_P}{c_V}, \tag{2.107}$$

where $c$ is the specific heat capacity at constant pressure and volume with $P$ and $V$ as subscripts respectively. Some of the parameters and their relative locations chosen might be based on assumptions and are not ideal, so it would be best to reach a more accurate solution using the most relevant parameters possible as considered in the calculations to follow in the next chapters.

The expansion ratio can be used to determine the Mach number at the exit:

$$\frac{A_{exit}}{A_{throat}} = \left(\frac{\gamma+1}{2}\right)^{-\frac{\gamma+1}{2(\gamma-1)}} M_{exit}^{-1}\left(1+\frac{\gamma-1}{2}M_{exit}^{2}\right)^{\frac{\gamma+1}{2(\gamma-1)}}, \quad 2.108$$

which can then be used to compute the temperature, pressure, and density at the exit or throughout the nozzle [50]:

$$T_{exit} = T_{chamber}\left(1+\frac{(\gamma-1)}{2}M_{exit}^{2}\right)^{-1} \quad 2.109$$

$$p_{exit} = p_{chamber}\left(1+\frac{(\gamma-1)}{2}M_{exit}^{2}\right)^{\frac{-\gamma}{\gamma-1}} \quad 2.110$$

$$\rho_{exit} = \rho_{chamber}\left(1+\frac{(\gamma-1)}{2}M_{exit}^{2}\right)^{\frac{-1}{\gamma-1}} \quad 2.111$$

If the fully expanded Mach number ($M_j$) is equal to $M_{exit}$, then the exit pressure equals ambient pressure, otherwise the performance will be decreased due to shocks in the exhaust or nozzle [43]:

$$M_j^2 = \frac{2}{\gamma-1}\left[\left(\frac{p_{stagnation}}{p_{ambient}}\right)^{\frac{\gamma-1}{\gamma}} - 1\right] \quad 2.112$$

The residence time ($t_s$) can be found as:

$$t_s = \frac{V}{Q}, \quad 2.113$$

where $V$ is the chamber's volume and $Q$ is the volumetric flow rate [33]:

$$Q = vA \quad 2.114$$

The power needed to vaporize the water is:

$$P = \dot{m}\Delta H, \quad 2.115$$

where $P$ is power, $H$ is enthalpy, and the change in enthalpy needed is equivalent to:

$$\Delta H = H_V - H_L, \quad 2.116$$

where $H_V$ and $H_L$ refer to the enthalpies of water at boiling temperature for vapor and initial temperature for liquid respectively [50].

In a more elaborate approach, assuming that all the supplied energy heats the propellant, the total energy ($Q_T$) is:

$$Q_T = Q_0 + Q_1 + Q_2 + Q_3, \quad 2.117$$

where $Q_1$ is the needed energy to reach vaporization temperature and $Q_2$ is the needed energy for vaporization upon reaching vaporization temperature considering the liquid propellant with mass $m$ [38]:

$$Q_1 = mc\Delta T = mc(T_c - T_i), \quad 2.118$$

$$Q_2 = mL_v, \quad 2.119$$

where $T_c$ is the vaporization temperature in the chamber, $T_i$ is the initial temperature of propellant at inlet, $c$ is the liquid's specific heat, and $L_v$ is the latent heat of vaporization [38]. Considering the nonuniform temperature distribution in the chamber, it is challenging to analytically determine $Q_0$, which is the needed energy for heating the remaining liquid in the chamber ($\rho V_c - m$) to less than the vaporization

temperature, where $\rho$ is the liquid's density and $V_c$ is the volume of the chamber [38]. Finally, $Q_3$ is the heat for raising the vapor's temperature after the liquid propellant is completely vaporized:

$$Q_3 = \rho V_c C_v (T_c - T_c'), \tag{2.120}$$

where $C_v$ is the specific heat of gas at constant volume and $T_c'$ is the chamber temperature at complete vaporization [38]. The water entering the chamber's temperature slowly rises when the heater power is low, so $Q_0$ will be high compared to $Q_1$ and $Q_2$ along with $Q_3 = 0$ [38]. A larger amount of liquid is vaporized for medium heater power, as $Q_1$ and $Q_2$ increase, $Q_0$ decreases, and $Q_3$ is still zero. At the vaporization point of the chamber's liquid ($m = \rho V_c$), $Q_0$ becomes zero and $Q_3$ remains as zero until the liquid undergoes complete vaporization, after which it increases for higher heater power, which would result in constant $Q_1$ and $Q_2$ [38]. Considering $\dot{m}$ as the chamber's liquid mass flow rate that also becomes vapor and exits the nozzle, the fluid leaves over time $\tau$:

$$\tau = \frac{\rho V_c}{\dot{m}}, \tag{2.121}$$

which leads to the average power ($P_{avg.}$):

$$P_{avg.} = \frac{Q_T}{\tau} = \frac{m}{\rho V_c}\dot{m}(c\Delta T + L_v) + \frac{\dot{m}}{\rho V_c}(Q_0 + Q_3), \tag{2.122}$$

However, realistic undesired heat transfer which does not heat the propellant has not been accounted for. For the conduction rates ($q$) on the top ($t$) and bottom ($b$) sides of the thruster considering 1D heat flow:

$$q_{ct} = \frac{K_m A_{avt}}{d_t}(T_c - T_t), \tag{2.123}$$

$$q_{cb} = \frac{K_m A_{avb}}{d_b}(T_c - T_b), \tag{2.124}$$

where $A_{av}$ is the average conduction cross sectional area, $d$ is the thickness, $T_s$ is the thruster's outside surface temperature, and $K_m$ is the thermal conductivity of the thruster material [38]. Convection, which would possibly be negligible in space, and radiation from the outside of the thruster ($to$) are expressed as:

$$q_{conv.} = h_c A(T_{to} - T_{fluid}), \tag{2.125}$$

$$q_{rad.} = \sigma A(T_{to}^4 - T_{env.}^4), \tag{2.126}$$

where $h_c$ is the average convective heat transfer coefficient, $\sigma$ is the Stefan-Boltzmann constant, and $A$ is the effective convection and radiation surface area [38]. For steady state:

$$q_c = q_{conv.} + q_{rad.} \tag{2.127}$$

The heater power ($P$) is:

$$P = VI = I^2 R = \frac{V^2}{R}, \tag{2.128}$$

where $V$ is the applied voltage, $I$ is the current, and $R$ is the resistance [38]. Considering the rejected heat and Equation 2.122, the remaining power to vaporize the liquid is:

$$P_{avg.} = P - q_{ct} - q_{cb}, \tag{2.129}$$

Furthermore, Equation 2.129 can be reformulated as:

$$\frac{P - q_{ct} - q_{cb}}{\dot{m}} = \frac{m}{\rho V_c}[c\Delta T + L_v] + C_v(T_c - T_c') + \frac{1}{\rho V_c}Q_0, \tag{2.130}$$

where $H$ and $\dot{m}$ are controllable and $q_{ct}$ and $q_{cb}$ are dependent on the chamber temperature $T_c$ on the left hand side (LHS). On the right hand side (RHS), $m$ and $Q_0$ are unknown. $\frac{m}{\rho V_c}$ increases as

more liquid is vaporized. It becomes equal to one and $\Delta T$ saturates at $\Delta T_{max} = T_c' - T_i$, as complete vaporization occurs at an input power level [38]. When the power is higher, $Q_3$ becomes greater than zero and all variables in Equation 2.130 become known to find $T_c$, which by using thermodynamic principles leads to the chamber pressure $P_c$ [38]. This means that the thrust can be approximated using the heater power and liquid flow rate through Equation 2.98.

Compared to macroscale flow, the fluidic resistance ($R$) is larger for microchannel flow because of the channels' small cross sectional area [33]. The resistance ($R$) for a low aspect ratio rectangular microchannel, where the width ($w$) is comparable to the height ($h$), is:

$$R = \frac{12\mu L}{wh^3}\left[1 - \frac{h}{w}\left(\frac{192}{\pi^5}\sum_{n=1,3,5}^{\infty}\frac{1}{n^5}\tanh\left(\frac{n\pi w}{2h}\right)\right)\right]^{-1}, \tag{2.131}$$

where $L$ is the length of the microchannel [33]. The fluid resistance considering a high aspect ratio rectangular microchannel is estimated as:

$$R = \frac{12\mu L}{wh^3} \tag{2.132}$$

Additionally, the channel's surface roughness should also be taken into consideration, as it relates to $Re$ through the Darcy friction factor ($f$), which would be relatively large due to the low $Re$ in this small hydraulic diameter ($D_h$) microfluidic flow resulting in large fluidic resistance [33], where $A$ is cross sectional area, and $P$ is perimeter:

$$f = \frac{64}{Re} \tag{2.133}$$

$$D_h = \frac{4A}{P} \tag{2.134}$$

Therefore, the pressure drop could be lowered by adjusting the geometry as well. It is observable by plotting $f$ against $D_h$ that $f$ asymptotically becomes low enough beyond $D_h = 500\mu$m and grows quickly below this value [33]. The Darcy-Weisbach equation can be used to evaluate the pressure drop ($p_{drop}$):

$$p_{drop} = \lambda\left(\frac{L}{D_h}\right)\left(\frac{\rho\mu^2}{2}\right), \tag{2.135}$$

where $\lambda$ is the friction coefficient [33]. Furthermore, a function based on molar mass dependency of Knudsen-dependent correcting function coefficients has been developed in [22] to correct the pressure drop of continuum flow numerical models with no slip boundary conditions in the slip flow and transition regimes along the nozzle for their different studied gases (Xenon, Krypton, Argon, Neon) using four gas-independent accommodation coefficients:

$$S_{corr}^{*} = \frac{S_{sim}}{1 + \left(\frac{M}{0.278\text{kg/kmol}}\right)^{0.243} Kn_{sim}\left[1 + \left(\frac{M}{236.492\text{kg/kmol}}\right)^{-0.939} Kn_{sim}\right]} \tag{2.136}$$

where $Kn_{sim}$ is the numerical Knudsen number, $M$ is the propellant gas molar mass, and $S_{corr}^{*}$ and $S_{sim}$ refer to the gas-independent coefficient based corrected and numerical dimensionless pressure drops based on a dimensionless number respectively:

$$S = \frac{\Delta p w h^3}{Q\mu L}, \tag{2.137}$$

where $\Delta p$ is the nozzle pressure drop, $Q$ is the volumetric flow rate, $w$ is the depth, $h$ is the height, $\mu$ is the dynamic viscosity, and $L$ is the nozzle length. When the collision cross section, which is the area surrounding a particle in which the presence of another particle's center would cause a collision, of the gas is small, it leads to a greater deviation from the experimental results and can be better corrected using a second degree polynomial and the function becomes more linear with large collision cross sections [22]. Gas-dependent coefficients can also be determined for more accurate results and its

procedure is explained in [22], though it basically involves an experimental reference for plotting $f(Kn)$ as a function of $Kn_{sim}$ and fitting the data using polynomial function coefficients ($A_2$ and $A_1$), where:

$$f(Kn) = \frac{S_{sim}}{S_{exp}} - 1, \tag{2.138}$$

$$f(Kn) = A_2 Kn^2 + A_1 Kn, \tag{2.139}$$

Then, Knudsen function coefficients ($C_1$ and $C_2$) are determined to solve for $S_{corr}$:

$$C_1 = \frac{1}{A_1} \quad \text{and} \quad C_2 = \frac{A_2}{A_1}, \tag{2.140}$$

$$S_{corr} = \frac{S_{sim}}{1 + \frac{Kn_{sim}}{C_1}(1 + C_2 Kn_{sim})}, \tag{2.141}$$

To reach the gas-independent coefficients, a power function of the form $C = \left(\frac{M}{M^*}\right)^{-\beta}$ is used for $C_1$ and $C_2$ by plotting them against the molar mass and fitting the data to determine the constants $M_1^*$, $M_2^*$, $\beta_1$, and $\beta_2$, so Equation 2.136 becomes:

$$S_{corr}^* = \frac{S_{sim}}{1 + \left(\frac{M}{M_1^*}\right)^{\beta_1} Kn_{sim}\left[1 + \left(\frac{M}{M_2^*}\right)^{-\beta_2} Kn_{sim}\right]} \tag{2.142}$$

# 3

# Methodology

## 3.1. General Modeling Properties and Procedure

Blender, the open-source 3D multiuse computer graphics creation suite software, is used to design the model and create the geometry for OpenFOAM, as it allows for exporting as ASCII or binary and Stereolithography (STL) file format input for meshing with STL regions, naming, refinement, and quality, such as in skewness, triangulation creation, and overlapping edges. The models are done using the DSMC solver, dsmcFoam+, within OpenFOAM, which is an open-source C++ toolbox for numerical solvers and pre/post-processing utilities that incorporates computational fluid dynamics (CFD). Note that OpenFOAM uses lower camel (Dromedary) case with an initial lowercase letter and capitalized first letters of subsequent attached words. It is recommended to use dsmcFoam+ on Linux (Ubuntu) for a more direct installation and native usage, as has been done in this work using the latest dsmcFoam+ version, which was downloaded and rather straightforwardly built on a supercomputer and personal workstation following the links provided in [52]. Note that dsmcFoam+ slightly differs in its case setup from OpenFOAM 6's dsmcFoam. The converging section of the nozzle is generally expected to be in the continuum flow regime, so rhoCentralFoam, which is a Navier-Stokes (NS) solver, could be employed for that section to decrease the computational load which would incur from exclusively applying a DSMC solution, which would be more useful for the rarefied flow sections, though this option is found to be unnecessary and a full DSMC simulation is used to provide more comprehensive results. Data obtained from the numerical simulation including pressure, velocity, and temperature fields is to be used to calculate the model's properties, such as thrust and related performance efficiency terms, among other parameters in Chapter 4. ParaView (from the additionally installed newer OpenFOAM 6 from the OpenFOAM Foundation and not dsmcFoam+'s OpenFOAM 2.4.0), the open-source program for interactive and scientific visualization and data analysis qualitatively and quantitatively, is used as OpenFOAM's main post-processing tool. The methodology including the pre/post-processing work is intended to be mainly comprehensive.

To reference, a Dirichlet-Dirichlet boundary condition state-based coupling would be used to share the properties between both continuum and kinetic based models. Instead of using the continuum based model for the converging section of the nozzle and kinetic based model for the rest of the model as usually expected, equilibrium breakdown parameters (Equations 2.7 and 2.9) including the Knudsen number would be considered to realize when continuum based models become no longer accurate. If the interface considered between the models is chosen further upstream of its optimal conditions, it might significantly increase the computational load, whereas it might not give the most accurate solutions further downstream. A small overlap region solved using both solvers might also be applied to split the models for a safer result. First, the Navier-Stokes model would be applied to the whole model, which would result in less accurate results in the nonequilibrium regions. Second, the breakdown parameters would be evaluated to determine the location of the interface dividing the DSMC and Navier-Stokes regions. Third, the DSMC solver would be applied to the determined rarefied flow and possible overlap regions and the Navier-Stokes to the continuum flow and possible overlap regions. Fourth, an iteration would be used by evaluating the breakdown parameter in both regions and possibly adjusting the interface multiple times to determine the final solution. A similar approach is taken in [18].

The model studied is based on TU Delft's VLM. The design will follow the referenced results of TU Delft VLM models in Subsection 4.2.2, which would become especially helpful for further comparison, as the referenced models are full continuum simulations done using ANSYS Fluent with an SST $k-\omega$ model and low Reynolds numbers and compressibility effects corrections. The thrusters are typically made of seven heating sections in series that are 1.28 mm long in the direction of the flow, 3 mm wide along the thruster's cross section, and with 1 W of power each [16]. The satellite bus supplies 5 V of voltage, so a current of 200 mA can provide the required heater power [16]. The heater might be activated to reach the desired temperature, though that lowers its efficiency, or it could be reaching the desired temperature during firing, which lowers the specific impulse [11]. Firing at a particular time during heating might provide an optimal outcome, though it will add complexity to the design. It is desirable to increase the velocity throughout the nozzle, so a de Laval nozzle is used, where the subsonic gas speeds up in the converging section until reaching sonic speed at the throat, which it continues accelerating from with supersonic speed in the diverging section. This requires careful design considerations to obtain the desired flow. If a bend is included in the geometry (for the inlet), it might result in an intrinsically modeled backflow (could also be due to rarefaction), which would need to be further analyzed. The nozzles' throats will have a sharp corner, which would lead to a wider boundary layer decreasing thrust, though the results will remain comparable since it will be applied to both MEMS and conventional thrusters. An angle of 30° is most common for the converging section of a conventional nozzle, though it is not as important as the diverging section's angle for performance [36]. Along with the diverging half angle, throat and exit diameters mainly constitute the most important parameters of de Laval nozzles, as the mass flow rate can be set by the throat area and the ratio of throat and exit areas can set the outlet velocity and pressure [36]. A choked flow with $M = 1$ is desired at the throat. The throat's radius of curvature is usually set as the throat's diameter [36]. Figure 3.1 shows a generic de Laval nozzle and some of its properties [36].

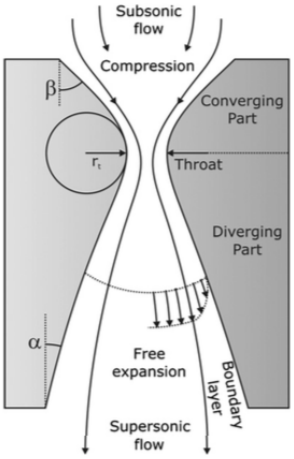



Figure 3.1: Generic de Laval nozzle ($\beta$ and $\alpha$ are the converging and diverging half angles respectively and $r_t$ is the throat's radius of curvature) [36]

As the MEMS VLM is quasi-2D, it has to be simulated as 3D to capture significant flow features that would not be captured in a 2D simulation, while the axial symmetry will be benefitted from to reduce the simulations' running time for the conventional convergent-divergent (CD) conical 3D nozzle to be simulated as a 5° wedge. dsmcFoam+ simulations are set in a 3D Cartesian coordinate system [52]. Since OpenFOAM uses the finite volume method for 3D simulations and needs special treatment for lower dimensions, the conical de Laval nozzle is modeled as a wedge that can be rotated during post-processing, as the dsmcFoam+ solver's axisymmetric capabilities incorporate the unique considerations for axisymmetric DSMC simulation particles in wedges with single cell thickness (in a similar way to 2D) due to the increasing number of molecules along the normal to the axis of symmetry. Considering the relatively small angle of the 5° wedge, its outer curvature could be negligible, as it will be rotated to 360°. The transient phase in these nozzles occurs very rapidly, though both transient and steady-state simulations would be useful to consider. Since this project mainly concerns the nozzle, the following Table 3.1 summarizes the dimensional features of the nozzle models. It also includes the

dimensions for the additional relative rectangular region outwards of the nozzle exit to study the plume, where one of the rectangle's shorter sides and the nozzle exit's centers coincide. The nozzles are simulated for inlet pressures of 5 and 7 bar at inlet temperatures of 550 and 773 K for a total of four cases for each nozzle. Notice that the ratio of higher pressure/temperature to lower pressure/temperature is around 1.4.

Table 3.1: Modeling dimensions for nozzles including plume region

| Nozzle | Inlet Diameter (mm) | Converging Half Angle (°) | Throat Diameter (mm) | Diverging Half Angle (°) | Outlet Diameter (mm) | Rectangle Length to Nozzle Length Ratio from Outlet Center | Rectangle Height/Width to Nozzle Exit Height/Width Ratio from Outlet Center |
|---|---|---|---|---|---|---|---|
| **MEMS (Quasi-2D, Simulated as 3D With Rectangular Section Depth of 0.1 mm)** | 2 | 45 | 0.025 | 30 | 0.8 | 3 | 5 |
| **Conventional (Conical 3D, Simulated as Wedge Using Axial Symmetry)** | 0.3 | 15 | 0.06 | 20 | 0.3 | 3 | 5 |

Using the properties from Table 3.1, the geometries of both MEMS and conventional nozzles and their plume regions have been created in Blender and shown in Figures 3.2 to 3.4 and Figures 3.5 to 3.7 respectively. A perspective view is additionally presented in Figures 3.4 and 3.7 considering the 3D MEMS and conventional nozzles and their plume region geometries, while the rest of the geometries in the mentioned figures are shown in orthographic view. To note, as Blender is not generally used for CFD modeling and may need tricks to attain some specifically desired results, a set of add-ons are used to help create the used models and integrate practical modeling functionalities. MeasureIt is used to present the mentioned rendered figures and their details. Mesh Align Plus is used to help in accurately rotating the conventional nozzle's wedge model. Snap Utilities Line is used to ease making lines and snapping them. tinyCAD Mesh tools is used to allow the creation of vertices at lines' points of intersection. Rheologic STL export is used to allow exporting all objects as separate STL files with readily desired file content.

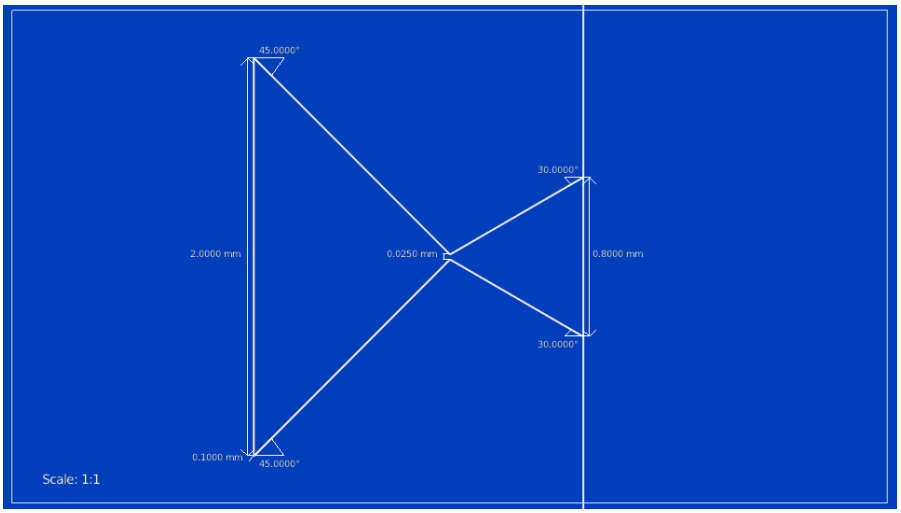



Figure 3.2: MEMS nozzle geometry

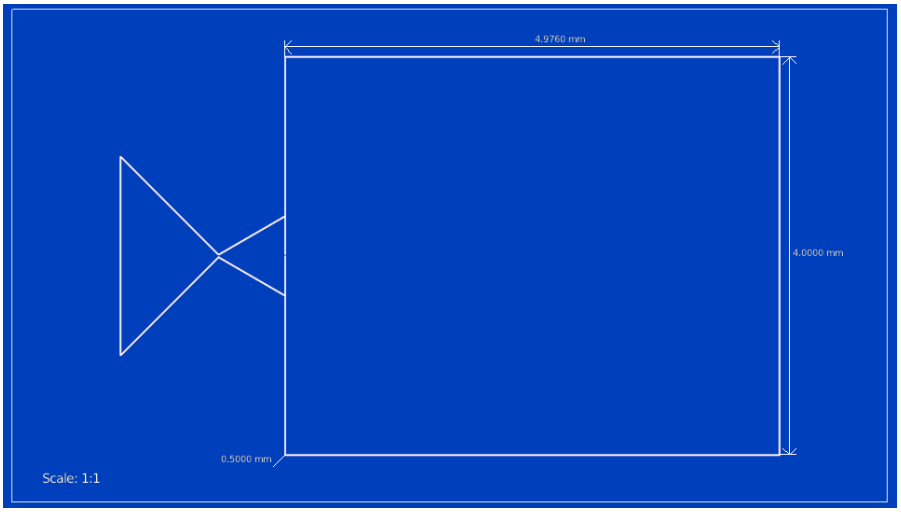



Figure 3.3: MEMS nozzle and plume region geometry

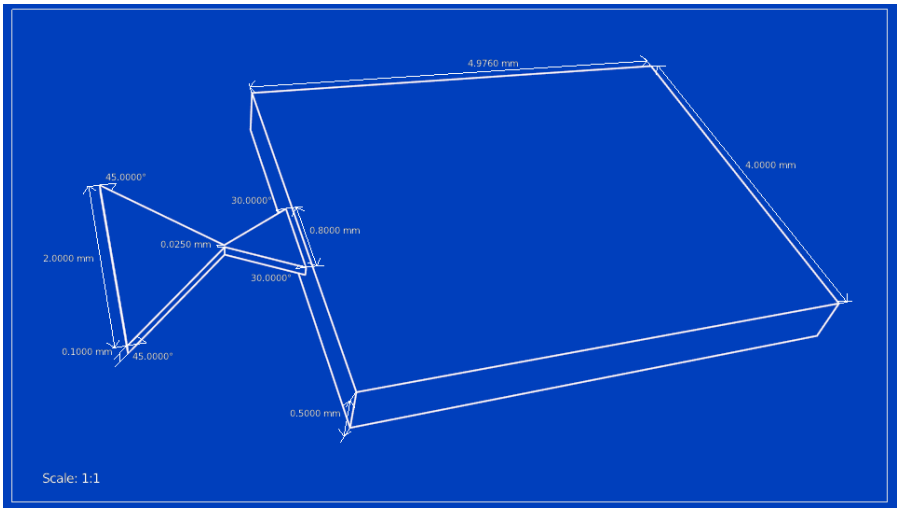



Figure 3.4: Perspective view of MEMS nozzle and plume region geometry

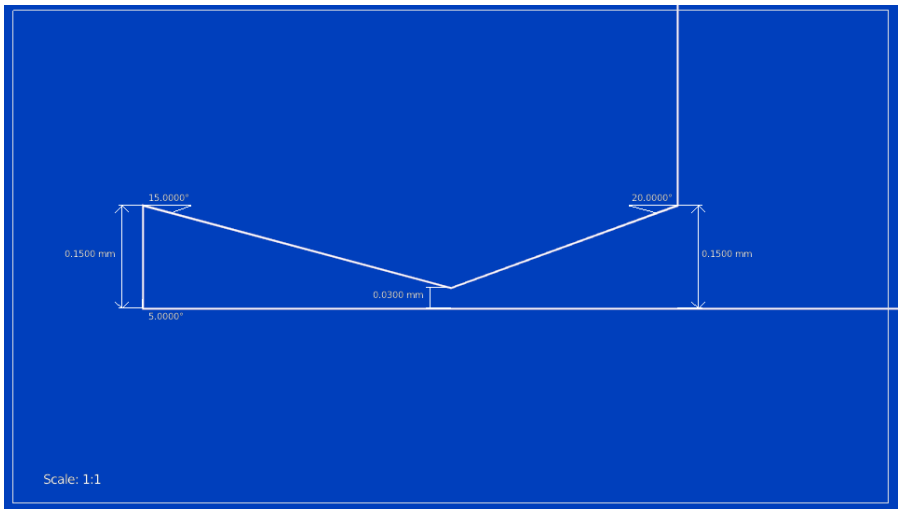



Figure 3.5: Conventional nozzle geometry

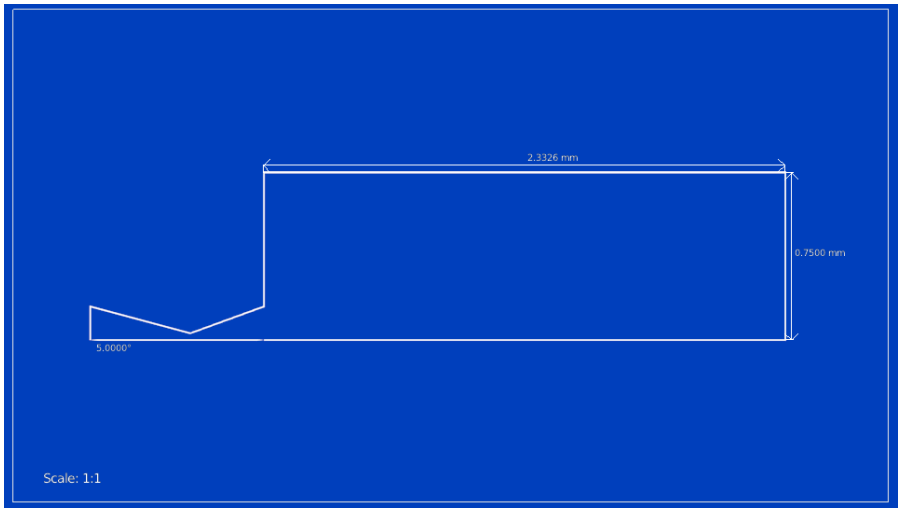



Figure 3.6: Conventional nozzle and plume region geometry

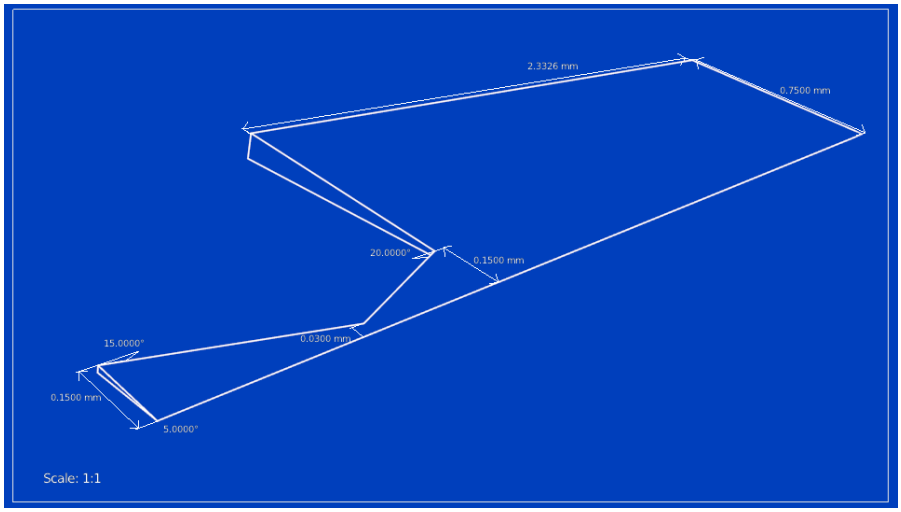



Figure 3.7: Perspective view of conventional nozzle and plume region geometry

## 3.2. OpenFOAM/dsmcFoam+

As commonly done in OpenFOAM, dsmcFoam+ usage starts with a new DSMC case, which has a standard file structure with a case directory containing the system and constant directories, which consist of most of the running parameters and physical domain data controlling dictionaries respectively [52]. dsmcFoam+ problems are often considered N-body simulations, where a number of objects and their interactions are simulated over time [52]. Usually, problems have less data to store for every discrete system body, such as particle position, though the DSMC particle data amount could be higher and composed non-homogeneously, as it might include velocity and rotational, vibrational, and electronic energy among others possibly depending on whether the particles considered are for atoms or molecules [52]. An individual dsmcCloud class extended from Cloud, the base OpenFOAM Lagrangian class, contains most of the memory data stored in dsmcFoam+, where its instantiation creates a doubly-linked list with stored pointers to parcels (particle collection), which are mutable class instantiations extended from Particle, the base OpenFOAM class [52]. In fact, most of dsmcFoam+'s classes are derived from existing (especially Lagrangian) OpenFOAM classes with specialized classes developed for certain functionalities [52].

### 3.2.1. blockMesh

The grid generation follows the geometry creation. First, a background mesh needs to be created, which is achieved using blockMesh. blockMesh is a basic OpenFOAM mesh generator similar to GAMBIT, though it generates the grid from its blockMeshDict dictionary in constant/polyMesh and not natively using a graphical user interface (GUI). To note, the current version of dsmcFoam+ uses OpenFOAM 2.4.0 and not OpenFOAM 6, which is why blockMeshDict is not in the system folder. blockMesh decomposes the domain into a user specified number of 3D hexahedral blocks with possible grading, straight edges, arcs, or splines [3]. Along with selecting the number of grid cells in each direction, hexahedral blocks of 8 corner vertices each need to be specified for the domain. Vertices can be collapsed to create the wedge blocks with under 8 vertices needed for the conventional nozzle's geometry by repeating them in the block vertices. To explain the upcoming code sections related to the mesh generation, the points specified are simply on the right-handed local coordinate system ($x_1$,$x_2$,$x_3$). It is advised to be consistent in numbering as applied due to the importance of orientation in the block section, so sketching and planning it beforehand would be helpful. The convertToMeters option is set to 0.001 to scale from meters to millimeters for consistency with the work in Blender.

Starting with the conventional nozzle, the vertices (obtainable from the Blender STL ASCII files) and their corresponding commented (//) numbers from 0 to 19 are (-0.447846 0 0) //0, (2.66233 0 0) //1, (2.66233 0 0.065375) //2>1, (-0.447846 0.75 0) //3, (-0.447846 0 0.065375) //4>0, (-0.447846 0.747146 0.065375) //5, (2.66233 0.75 0) //6, (2.66233 0.747146 0.065375) //7, (-0.05 0 0) //8, (-0.05 0 0.065375) //9>8, (-0.05 0.75 0) //10, (-0.05 0.747146 0.065375) //11, (0.329697 0 0) //12, (0.329697 0 0.065375) //13>12, (0.329697 0.75 0) //14, (0.329697 0.747146 0.065375) //15, (0.05 0 0) //16, (0.05 0 0.065375) //17>16, (0.05 0.75 0) //18, and (0.05 0.747146 0.065375) //19. OpenFOAM's C++ convention applies as counting starts from 0. Note that vertices 2, 4, 9, 13, and 17 are replaced with their repeated collapsed vertices 1, 0, 8, 12, and 16 respectively. The created wedge blocks are hex (0 8 10 3 0 8 11 5) (80 180 1) simpleGrading (0.5 1.5 1), hex (8 16 18 10 8 16 19 11) (30 180 1) simpleGrading (1 1.5 1), hex (16 12 14 18 16 12 15 19) (50 180 1) simpleGrading (2 1.5 1), and hex (12 1 6 14 12 1 7 15) (220 180 1) simpleGrading (1.2 1.5 1). To elaborate, the vertices defining the blocks and its bottom and top planes follow the word hex (hexahedra), while considering the axes as specified in ($x_1$,$x_2$,$x_3$), which in this case have the axis of symmetry $x$ in the axial direction, $y$ in the radial direction, and $z$ in the circumferential direction. For reference, the $xy$ plane is the bottom side of the wedge, though the conventional nozzle's geometry could also be rotated so that the $xy$ plane symmetrically splits the geometry, where the positive and negative $z$ coordinate values are each related to 2.5° of the geometry. In addition to initially creating the background mesh to exactly fit the extremities of the geometry, the $z$ maxima were very slightly increased from 0.065367 to 0.065375 to allow better snapping using snappyHexMesh, as that increase provides applicable space for snapping to both the top and radial faces of the wedge. There are four blocks, which in order are the nozzle's split converging, throat, and diverging sections along with the plume region. The second entry of the structured grid created includes a varying number of grid cells in the $x$ direction concentrated in the nozzle region and specifically around the throat. The number of grid cells in the $y$ (180) and $z$ (1) directions is constant,

as the structured grid is one cell thick in the circumferential direction for axisymmetric application. Additionally, to improve the quality of the simulations while being mindful of the computational time, cell expansion ratios in the form of simple grading are used, where refinement is applied in their respectively defined directions to preserve the number of grid cells specified in the second entry while setting the direction length ratio between the last and first grid cells. The axial grading is applied in the converging and diverging blocks with an expansion ratio progressively halving the cell's length towards the throat along with the plume region's weaker grading towards the nozzle. Additionally, radial grading is used to boost the number of grid cells at the nozzle and $x$ flow axis, considering that the plume regions has been preemptively made larger than expected necessary and the throat region's smaller size needs the extra refinement. Mainly, a smaller mean free path should have a respectively smaller grid cell size.

The MEMS nozzle's blockMeshDict follows a very similar approach with vertices (obtainable from the Blender STL ASCII files) and their corresponding commented (//) numbers of (-1 -2.1 -0.3) //0, (5.7 -2.1 -0.3) //1, (5.7 -2.1 0.4) //2, (-1 2.1 -0.3) //3, (-1 -2.1 0.4) //4, (-1 2.1 0.4) //5, (5.7 2.1 -0.3) //6, (5.7 2.1 0.4) //7, (-0.05 -2.1 -0.3) //8, (-0.05 -2.1 0.4) //9, (-0.05 2.1 -0.3) //10, (-0.05 2.1 0.4) //11, (0.67117 -2.1 -0.3) //12, (0.67117 -2.1 0.4) //13, (0.67117 2.1 -0.3) //14, (0.67117 2.1 0.4) //15, (0.05 -2.1 -0.3) //16, (0.05 -2.1 0.4) //17, (0.05 2.1 -0.3) //18, and (0.05 2.1 0.4) //19. The created blocks are hex (0 8 10 3 4 9 11 5) (15 34 13) simpleGrading (0.5 ((0.5 0.5 0.32)(0.5 0.5 3.125)) 1), hex (8 16 18 10 9 17 19 11) (2 34 13) simpleGrading (1 ((0.5 0.5 0.32)(0.5 0.5 3.125)) 1), hex (16 12 14 18 17 13 15 19) (9 34 13) simpleGrading (2 ((0.5 0.5 0.32)(0.5 0.5 3.125)) 1), and hex (12 1 6 14 13 2 7 15) (42 34 13) simpleGrading (1.2 ((0.5 0.5 0.32)(0.5 0.5 3.125)) 1). The axes as specified in ($x_1$,$x_2$,$x_3$) in this case similarly have $x$ in the flow direction, $y$ towards the converging and diverging sides of the nozzle, and $z$ towards the flat top and bottom sides of the nozzle. For reference, the $xy$ plane is the bottom side of the nozzle and the $x$ axis is centered, though the $xy$ plane could also be raised to intuitively symmetrically split the geometry. The background mesh is a slightly larger fit than the extremities of the geometry. Just like for the conventional nozzle, there are four blocks, which in order are the nozzle's split converging, throat, and diverging sections along with the plume region. The second entry of the structured grid created includes a varying number of grid cells in the $x$ direction concentrated in the nozzle region and specifically around the throat. The structured grid's number of grid cells in the $y$ (34) and $z$ (13) directions is constant. Furthermore, simple and multi-grading are used, where multi-grading, which can be found nested within simple grading, can comparably preserve the number of grid cells specified in the second entry while setting the direction length ratio between the last and first grid cells applied in their respectively defined split regions and multiple directions, as the first number represents the automatically normalized scale (fraction/percentage/absolute length) of the desired region, second number represents the region's automatically normalized scale (fraction/percentage) of grid cells, and third number represents the region's expansion ratio. The $x$ axis grading is applied in the converging and diverging sections' blocks with an expansion ratio progressively halving the cell's length towards the throat along with the plume region's weaker grading towards the nozzle. Due to the significantly small size of the throat, the ability to have multiple grid cells along its width becomes challenging, so the multi-grading tool is used to increase the number of grid cells in that region by splitting at the $xz$ plane with an identical number of grid cells on each side and grading in the $y$ direction towards it. The reason why the section expansion ratios 0.32 and 3.125 are used is because they are reciprocals and attain the desired grading identically on both sides.

To note, both conventional and MEMS nozzles' $x$ and $y$ axes gradings applied towards the throat desirably lead to more uniform squarish rectangles at the throat and extended sides. Since snappyHexMesh is used, one way to proceed is through specifying the boundary section outer patch faces with vertices externally counterclockwise (right-hand rule), though then they will simply need to be erased from the constant/polyMesh/boundary file along with similarly decreasing the number of boundaries by the number of erased boundaries before running the simulation. Note that for the geometry used, it might have been more time efficient to only use blockMesh for the simulations, though it is ultimately done using the methodology mentioned as that would be easier and more flexible to use for more elaborate geometries in possible future optimization work. As for much of this chapter, further information can be found in the OpenFOAM User Guide, which is easily obtainable online at [3]. It would also be more convenient to use or modify reference, tutorial, or master dictionary files instead of writing them from scratch. Following running blockMesh, the mesh can be visualized in ParaView (as explained in Subsection 3.4). Respectively, Figures 3.8 and 3.9 show the conventional nozzle's blockMesh from the front and back. Similarly, Figures 3.10 and 3.11 do the same for the MEMS nozzle.

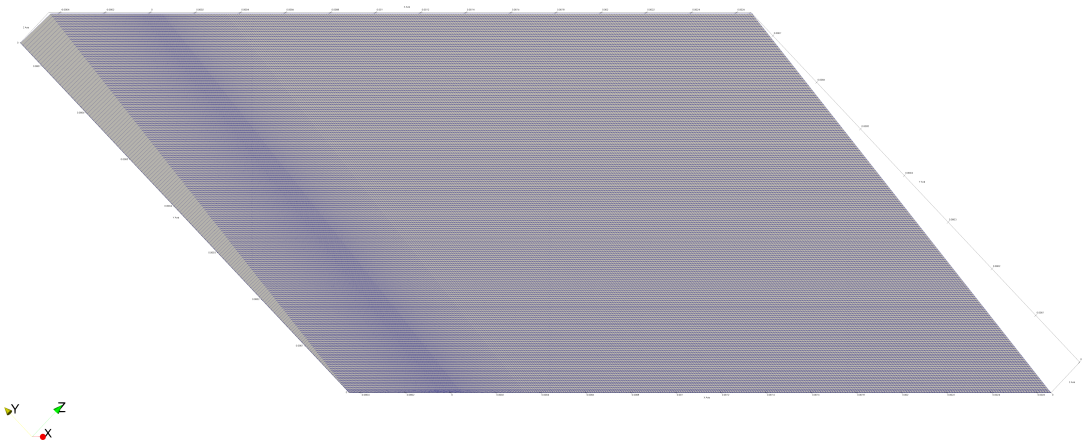



Figure 3.8: Conventional nozzle blockMesh (front)

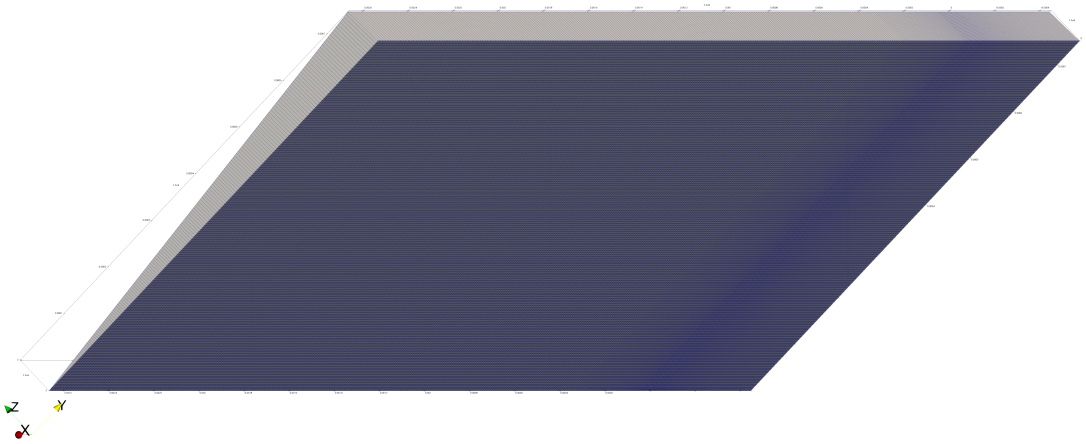



Figure 3.9: Conventional nozzle blockMesh (back)

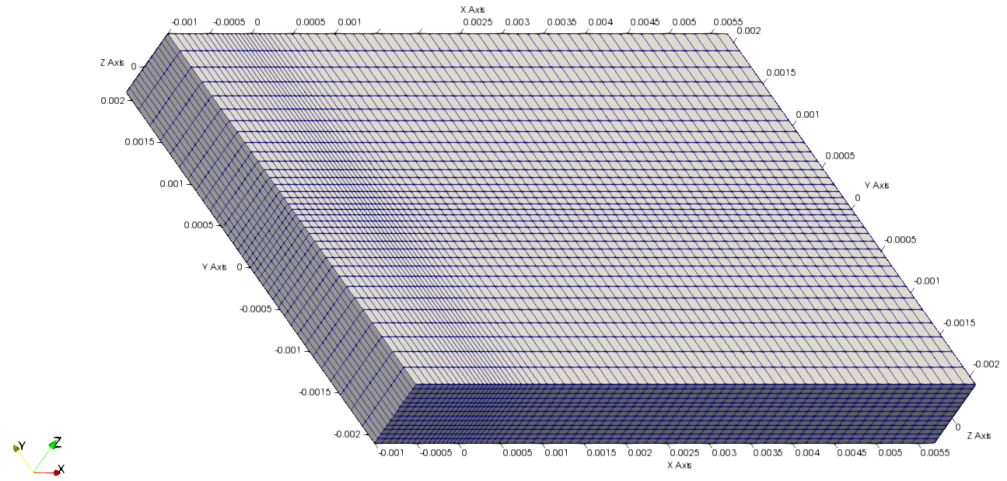



Figure 3.10: MEMS nozzle blockMesh (front)

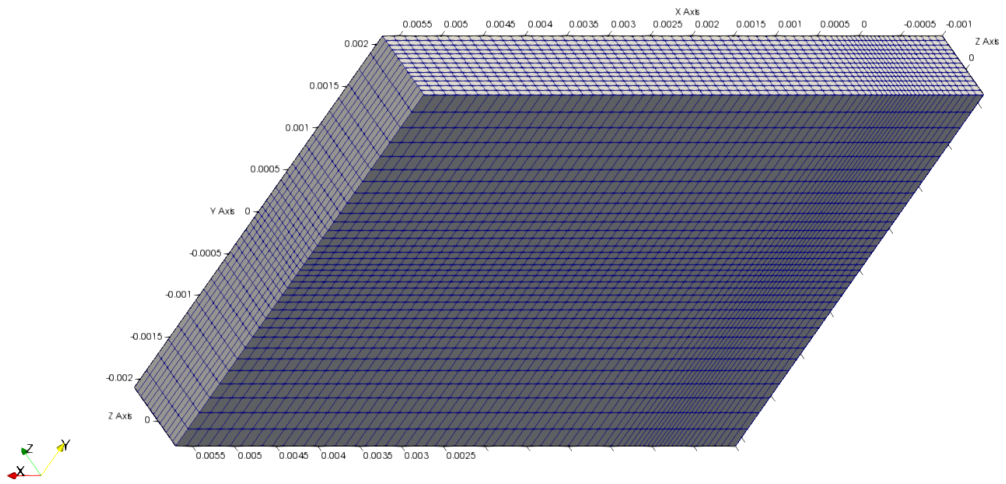



Figure 3.11: MEMS nozzle blockMesh (back)

### 3.2.2. snappyHexMesh (with surfaceFeatureExtract)

snappyHexMesh, in comparison with blockMesh and its the structured grids, is a 3D hybrid mesh generator with hexahedra and split-hexahedra automatically created from STL or Wavefront Object (OBJ) files' triangulated surface geometries (tri-surfaces) [3]. The grid roughly fits to the surface through iterations starting from a background mesh. Cell layers can also be added, though they were not used in this work, due to satisfactory resolution for capturing the expected turbulence phenomena and the potentially added computational time. It can also run in parallel with iterating load balancing. After the creation of the background base mesh using blockMesh and using the separate STL files for the geometry previously created using Blender by placing them in the case directory's constant/triSurface sub-directory, snappyHexMesh is used to snap to the geometry and refine the MEMS nozzle's grid, as the conventional nozzle's grid refinement using snappyHexMesh would cause an undesired refinement in the $z$ direction, which is why it is preliminarily more refined in its base level mesh of blockMeshDict compared to the MEMS nozzle's grid. It is important to note that before running snappyHexMesh -overwrite (to overwrite existing mesh/results files), surfaceFeatureExtract is run, where it extracts sharp edges from STL file surfaces and is read from its dictionary at system/surfaceFeatureExtractDict. After entering all the STL file names, both nozzles use extractionMethod, which defines how raw features are extracted, of extractFromSurface and writeObj (OBJ write format options) set to yes, though the extractFromSurfaceCoeffs, which marks features from edges with adjacent surface normals at an angle below the set includedAngle, for the conventional nozzle have an includedAngle of 180, which selects all edges, while 100 is used for the MEMS nozzle.

To explain the process for snappyHexMesh, the castellatedMesh and snap options are set to true while addLayers is set to false in the system/snappyHexMeshDict dictionary file, followed by listing the STL files with a triSurfaceMesh type in the geometry sub-dictionary. For the conventional nozzle, they are inlet, converging, diverging, exterior (outlet), exteriorSides (extended from the nozzle exit), backWedge, and frontWedge, while for the MEMS nozzle, inlet, converging, throat, diverging, exterior, and exteriorSides (extended from the nozzle exit) are used. For Blender's STL files, the MEMS nozzle is also split with a separate throat region as the extra snappyHexMesh refinement is only done for the MEMS nozzle's grid in addition to the added need for refinement at the MEMS nozzle's throat region due to its smaller width, while the conventional nozzle's throat region is not separated to maintain a single cell thick mesh in the circumferential direction without snappyHexMesh refinement. A simply castellated mesh will result in a Lego-like mesh, where an expected oblique line is decomposed into jagged blocks like stairs rather than a ramp, before being smoothened by the snap option. The castellatedMeshControls sub-dictionary for both are standardly set, though the nCellsBetweenLevels, which is the different refinement levels' number of connecting buffer layers, is set to 1 for the conventional nozzle in contrast with 4 for the MEMS nozzle, which is mainly and simply due to the lesser need for it in the conventional nozzle's grid. As a recommended precaution, the approximate overall cell limit, maxGlobalCells, can be modestly set to prevent filling up the random-access memory (RAM) by stopping the process if the refinement is overkill. No explicit feature edge refinements, which define the refinement level for intersecting edge cells, are set, though the eMesh files of each of the geometry sub-dictionary STL files are listed. This section also concerns the splitting of cells based on the set refinement levels starting with the specified edge features. The surface based refinement's first level number describes the minimum level of refinement for every cell intersecting a surface, while the second maximum level is for multiple surface intersecting cells with an angle greater than the resolveFeatureAngle, which resolves sharp angles and is standardly set at 30 for both nozzles. For the conventional nozzle, the levels are all set to 0 to avoid refining in the circumferential direction as mentioned, considering that the refinement of a 3D cell divides it in each direction into 8 smaller cells, with the inlet and exterior's patchInfo types set to patch, converging, diverging, and exteriorSides' patchInfo types set to wall, considering the potential outside of the spacecraft for evaluation, and backWedge and frontWedge patchInfo types set to symmetry (specular walls or wedge types are also equivalent). However, for the MEMS nozzle, the inlet with patchInfo type patch has level (2 2), converging and diverging with patchInfo types wall have level (2 2), throat with patchInfo wall has (3 3) on the other hand, to boost the number of cells and resolution along the small width of the throat, exterior with patchInfo type patch has level (0 0), and exteriorSides with patchInfo type wall, considering the potential outside of the spacecraft for evaluation, has level (0 0). Also, it is noticeable that the nozzle region is desired to be more refined than the plume region, as implemented, to focus on the important regions for accuracy along with that the Lagrangian nature of DSMC simulations which recommend a minimum number of particles to achieve for each grid

cell and having a refined larger plume region will make it more difficult to accomplish that while keeping the simulations achievable within a feasible time. Afterwards, the mesh selection entry includes a locationInMesh point to specify which part of the mesh to keep after refinement while the rest of the cells (with around less than 50% of their volume in the region) are removed, as in inside or outside the refinementSurfaces. Since the locationInMesh point should be inside a cell and not on a face, it has been calculated for the nozzles in meters where it is placed at the center of the throat's geometry for the MEMS nozzle (0 0 0.00005) due to its smaller scale and in the middle of the conventional nozzle's geometry by dividing its dimensions by 2 to result in (0.0015550875 0.000375 0.0000326835), though other points could be chosen as well.

After the cell removal, the surface snapping is initiated, where cell vertex points are transferred to the STL surface geometry to remove the castellation followed by internal mesh relaxation solving, determining if any new vertices create mesh quality complications based on the set parameters to then have their displacement decreased with a repeating process until the mesh quality is accepted. Interestingly, the snapping settings (snapControls sub-dictionary) using implicit wrapping to preserve features for these particular geometries with sharp edges prove to be much more effective in snapping using a smaller number of iterations, where for the conventional nozzle, the nSmoothPatch, which is the patch's number of smoothing iterations to be followed by the projection onto the surface, tolerance, which is the points' relative distance to be attracted by the surface feature point/edge using the true distance as its factor multiplied by the local maximum edge length, nSolveIter, which is the interior snapped mesh displacement relaxation iterations number, nRelaxIter, which is the maximum snapping relaxation iterations number limiting the scaling back iterations for reducing errors to stop prior to the attaining correct mesh, and feature snapping nFeatureSnapIter, which is the eMesh files' feature edge snapping iterations number, are all set to 1. For the MEMS nozzle, nSmoothPatch of 2, identical tolerance of 1.0, nSolveIter of 25, nRelaxIter of 3, and nFeatureSnapIter of 10 are set. Both nozzles' feature snapping settings have implicitFeatureSnap, which finds geometric features through surface sampling, set to false, since the explicitFeatureSnap and surfaceFeatureExtract are being used, and explicitFeatureSnap, which as the name implies enables explicit feature snapping, and multiRegionFeatureSnap, which finds points on more than one surface to be used for explicitFeatureSnap, are set to true.

The mesh quality settings (meshQualityControls sub-dictionary), which specify when to undo to a former accepted version, as the mesh quality is continuously checked during the snappyHexMesh run, are set to generally standard values with maximum allowed non-orthogonality, which is the normalized dot product of the face area vector with the connecting cell to adjacent cell centroids vector [30], maxNonOrtho of 90, maximum allowed skewness, which is found as the distance from the face center to the cell centers' face intersection point normalized by the connecting cell to adjacent cell centroids distance [30], maxBoundarySkewness of 20 and maxInternalSkewness of 4, maximum allowed concaveness, which inspects the interior face angles [30], maxConcave of 80, minimum pyramid volume, which is an absolute pyramid cell volume and typically defined as a fraction of the expected smallest cell volume as it is found from the face area vector and cell to face centers vector dot product [30], minVol of 1e-13, minimum tetrahedral cells quality, which is obtained from cell decomposition determined from the face center, cell center, and variable base point minimum decomposition triangles as it is found using the circumferential radius and tetrahedral volume [30], minTetQuality of 1e-30, which is set as positive for applicable tracking such as for streamlines, minimum face area minArea of -1 (disabled), minimum face twist, in which the faces are split from the face center using triangles and it is found as the cell to adjacent cell centers vector and triangular face area vector normal dot product [30], minTwist of 0.02, minimum normalized cell determinant, which is found as the determinant of the tensor of the evaluated face area vectors [30], minDeterminant of 0.001, face weight metric, which is evaluated as the lesser of the projected owner or neighbor cell to face centers lengths divided by their sum [30], minFaceWeight of 0.02, minimum face volume ratio metric, which is evaluated as the owner and neighbor volumes' ratio of their minimum divided by their maximum [30], minVolRatio of 0.01, and face triangle twist, in which the faces are split from the face center using triangles and it is found as the neighboring triangular element unit normals' dot product [30], minTriangleTwist of -1. In its advanced settings, the displacement scaling smoothing error distribution iteration number nSmoothScale is set at 4 and error point scaling back displacement amount errorReduction of 0.75. The mergeTolerance, which is the starting mesh's total bounding box fraction, is set at 1e-6, as the write tolerance needs to exceed its value as set by the ASCII data files' writeFormat written to a writePrecision (write floating point precision) of 7 significant figures in controlDict (Subsection 3.2.7). To note, especially with a

starter's lower sensibility of the algorithms and due to the edges' sharpness, a lot of trial and error is expected to be involved in this section.

After execution and as explained using ParaView (Subsection 3.4), the mesh can be visualized. Figures 3.12 and 3.13 show the conventional nozzle's final mesh from the front and back respectively, while Figure 3.14 is a magnified shot of the nozzle section's mesh. Furthermore, the patch names used as colored differently are shown in Figure 3.15 and the Extract Cells By Region Filter is used to show the (Y Normal (outside extraction side with extract intersected cells) at the geometrical throat radius midpoint) cross section of the mesh and single cell thickness as shown in Figure 3.16. Respectively in order, Figures 3.19, 3.20, 3.21, 3.22, and 3.23 show the same for the MEMS nozzle, though the Extract Cells By Region Filters use the Y along with the Z Normal (inside extraction side with extract intersected cells) at the respective throat dimensions' midpoints to show the cross sections of the mesh. Also, the conventional nozzle's Figures 3.17 and 3.18 respectively show its rotated final mesh from the front and back with the Angular Periodic Filter using a 5° Rotational Angle on Axis X.

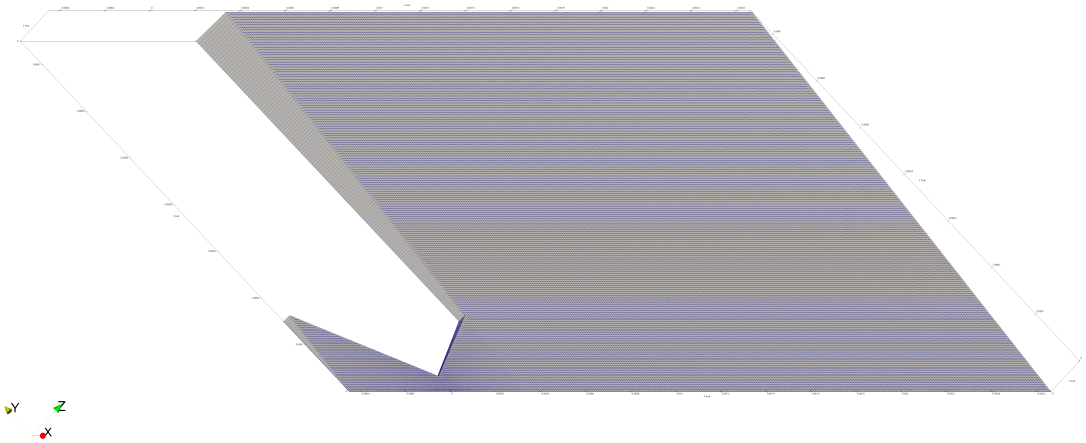



Figure 3.12: Conventional nozzle final mesh (front)

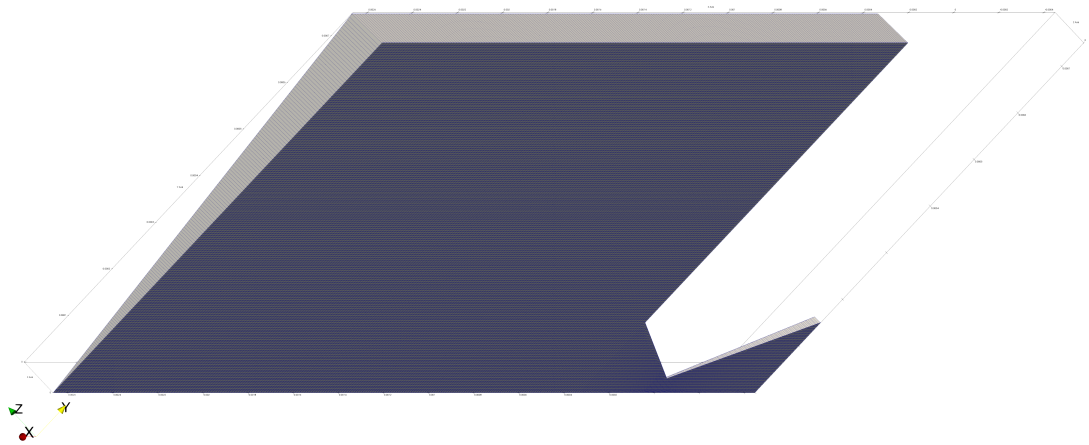



Figure 3.13: Conventional nozzle final mesh (back)

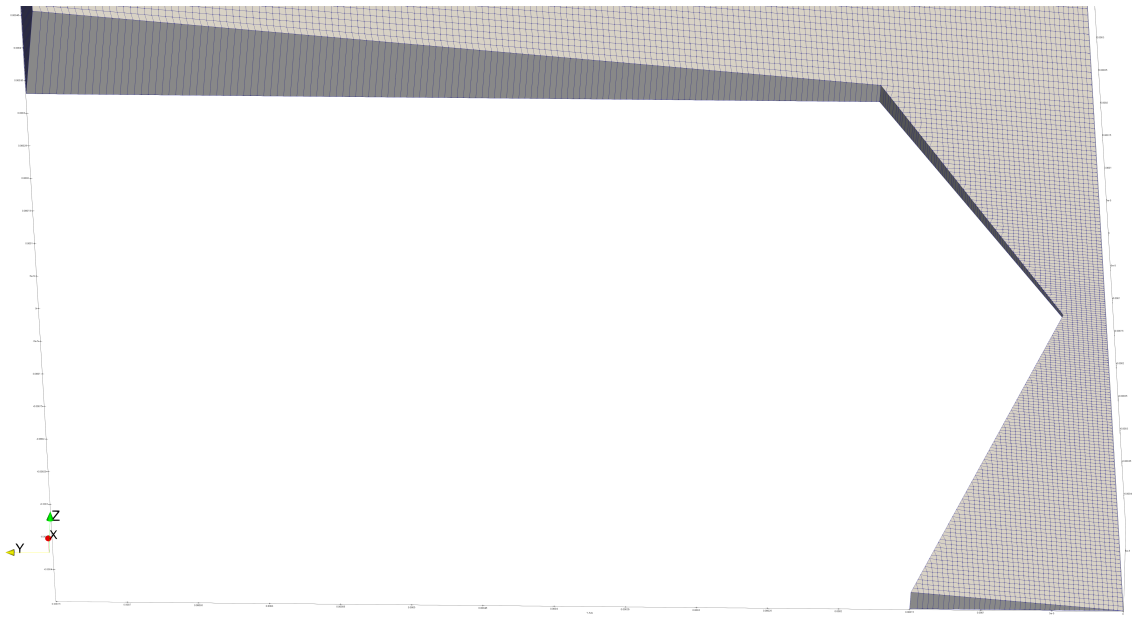



Figure 3.14: Conventional nozzle final mesh with magnified nozzle section

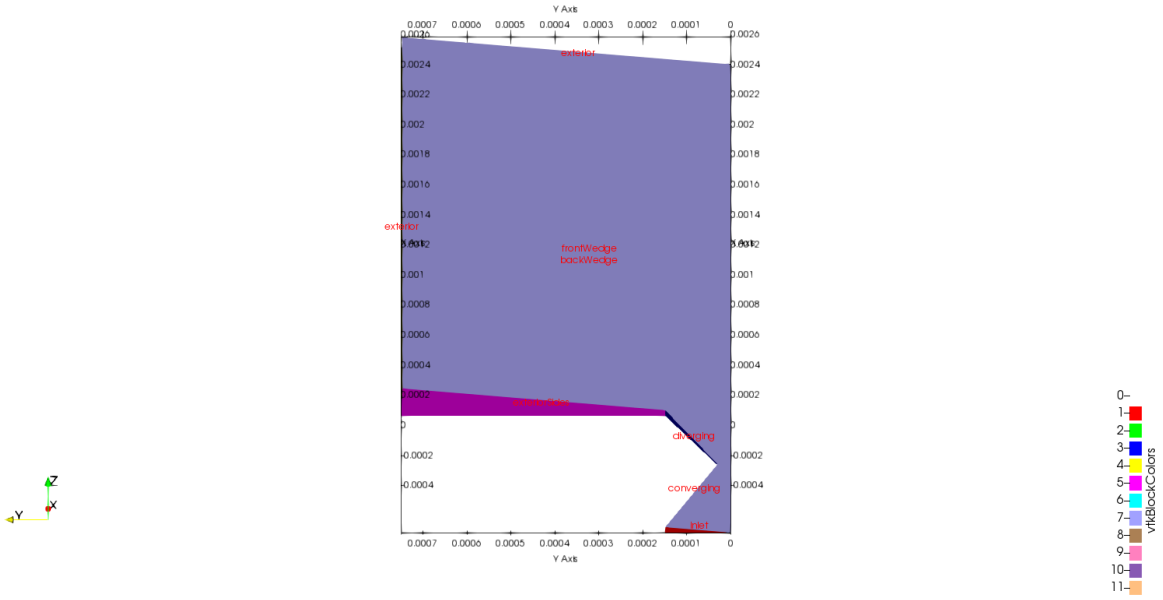



Figure 3.15: Conventional nozzle final mesh with patch names (correspondingly differently colored)

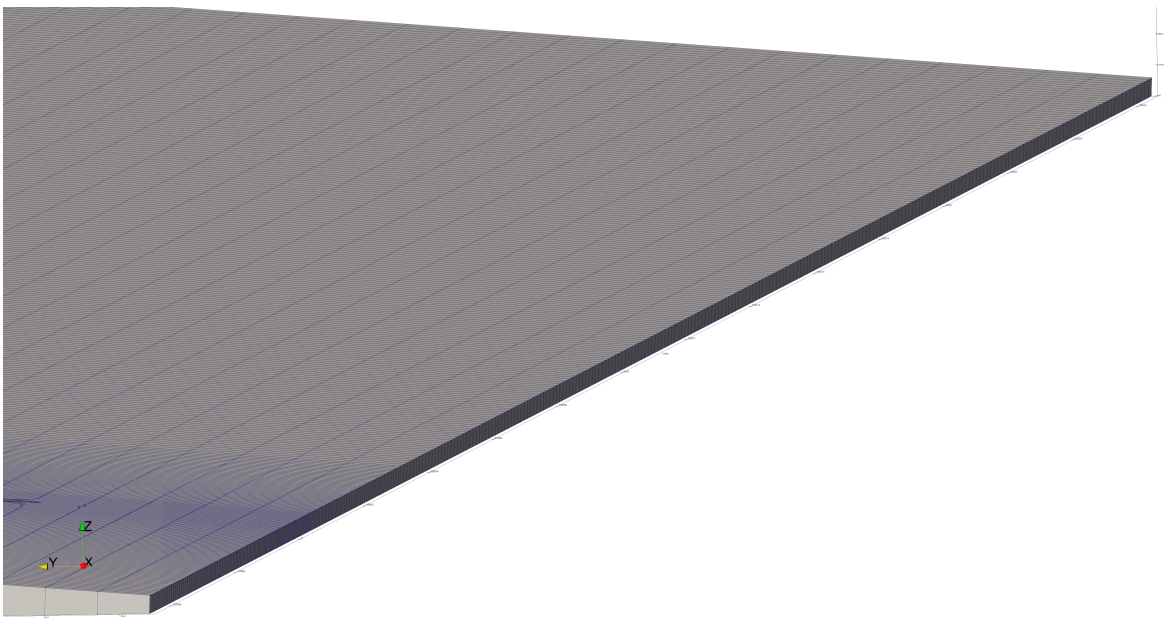


Figure 3.16: Conventional nozzle final mesh with the Extract Cells By Region Filter used to show the (Y Normal (outside extraction side with extract intersected cells) at the geometrical throat radius midpoint) cross section of the mesh and single cell thickness

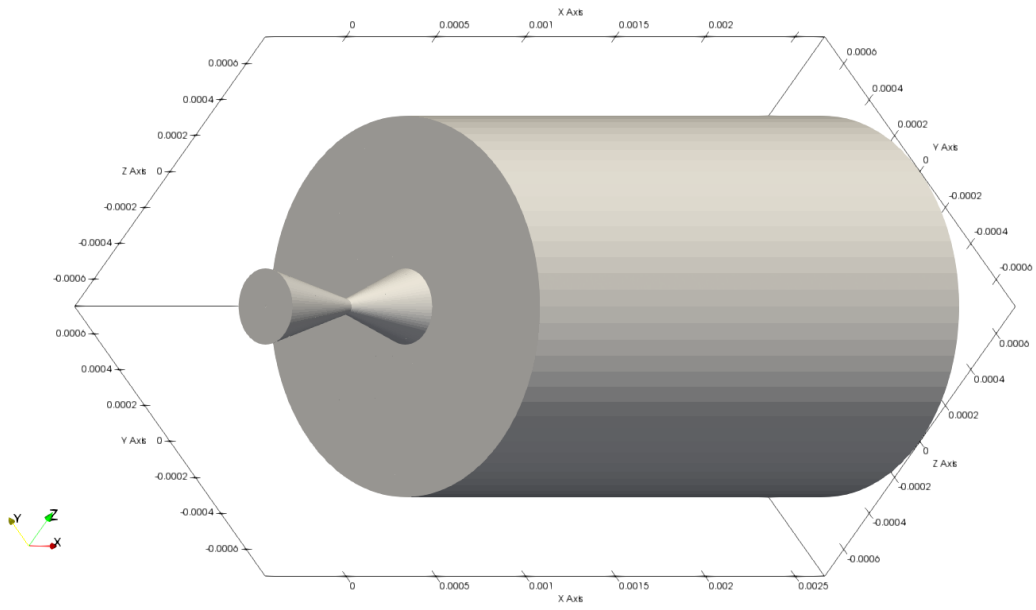



Figure 3.17: Conventional nozzle rotated final mesh with the Angular Periodic Filter using a 5° Rotational Angle on Axis X (front)

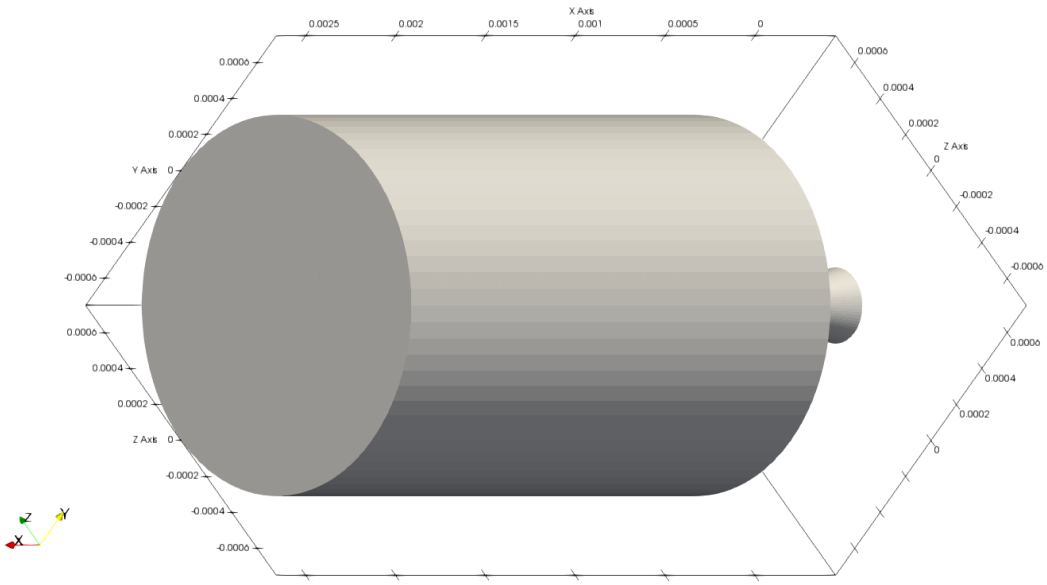



Figure 3.18: Conventional nozzle rotated final mesh with the Angular Periodic Filter using a 5° Rotational Angle on Axis X (back)

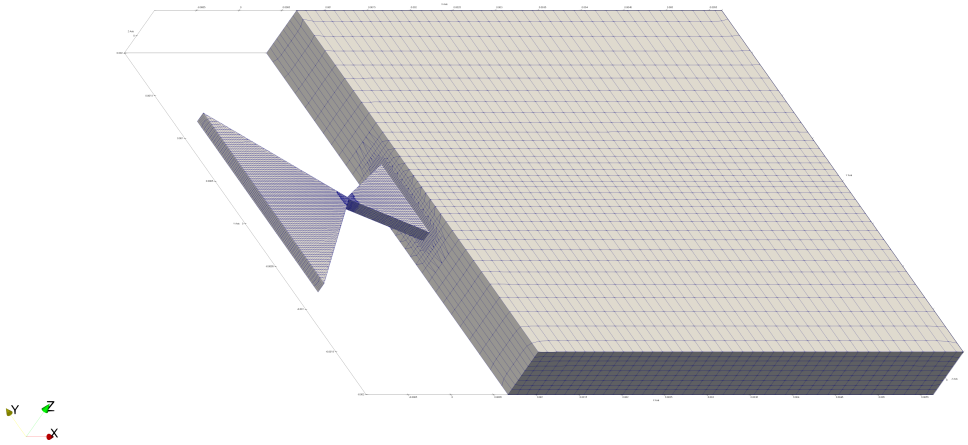


Figure 3.19: MEMS nozzle final mesh (front)

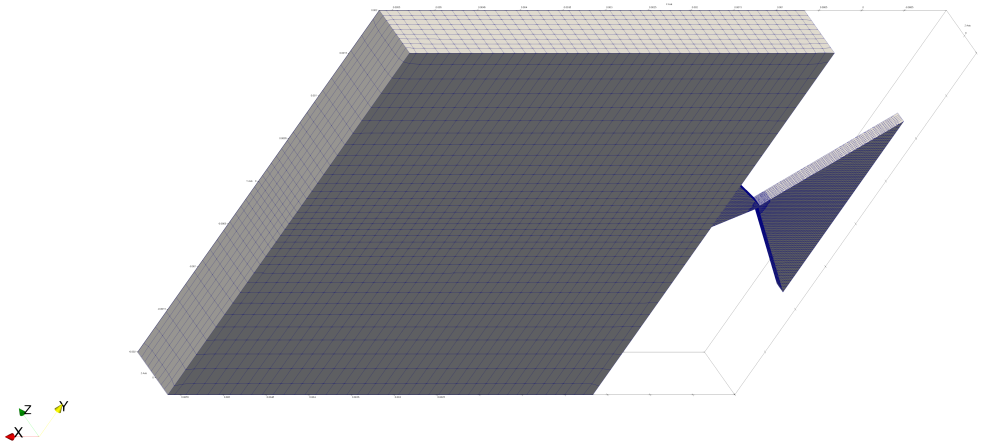


Figure 3.20: MEMS nozzle final mesh (back)

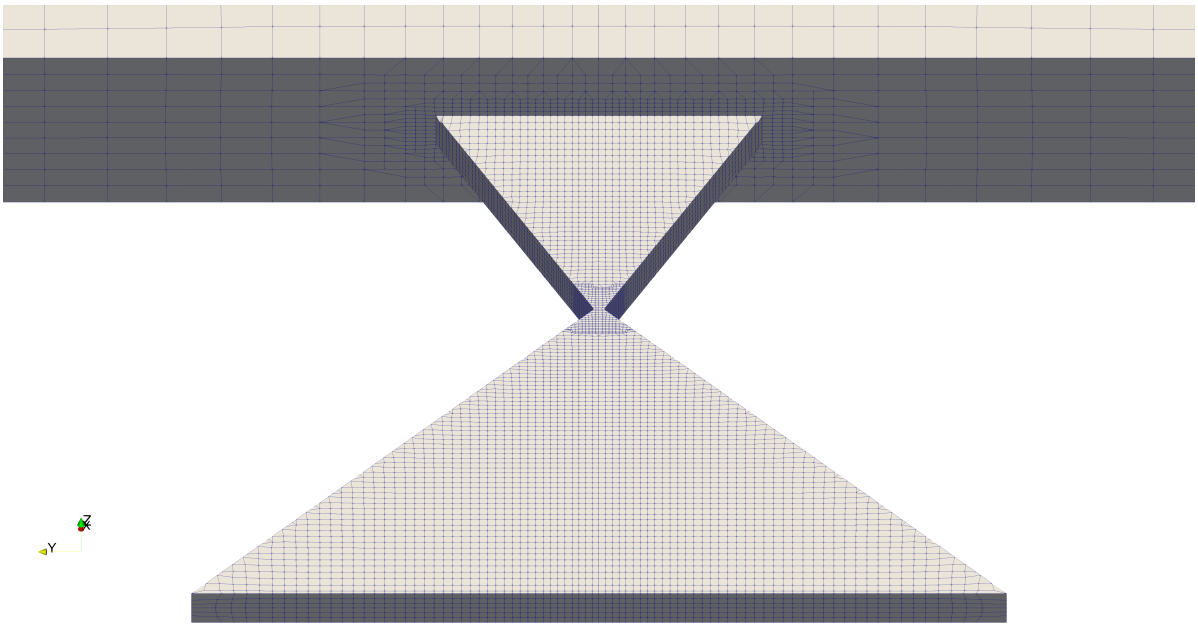


Figure 3.21: MEMS nozzle final mesh with magnified nozzle section

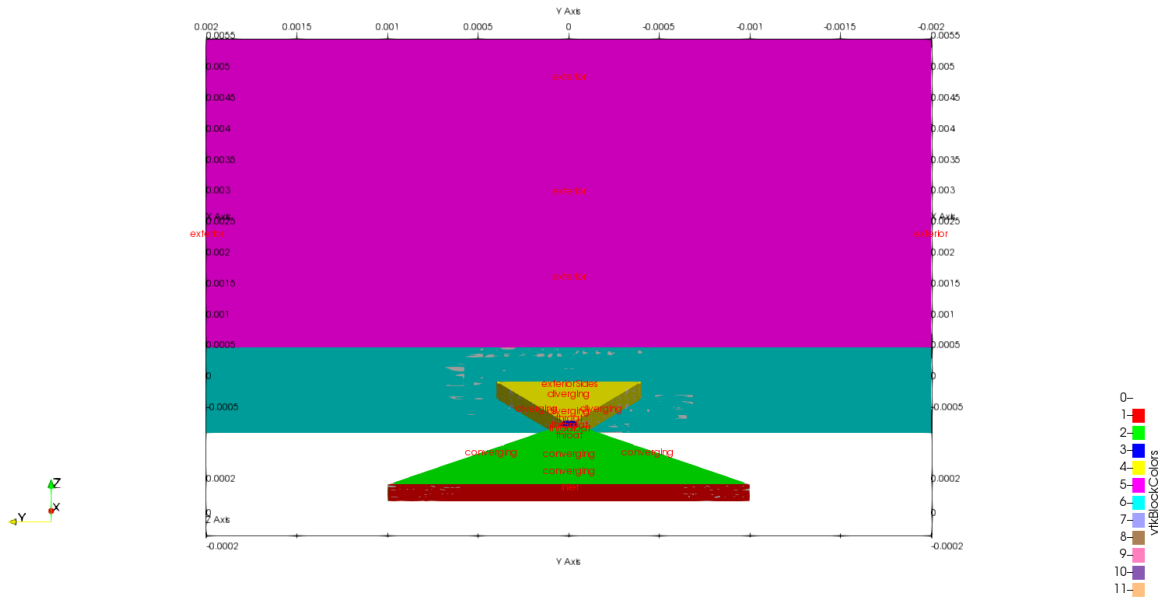



Figure 3.22: MEMS nozzle final mesh with patch names (correspondingly differently colored)

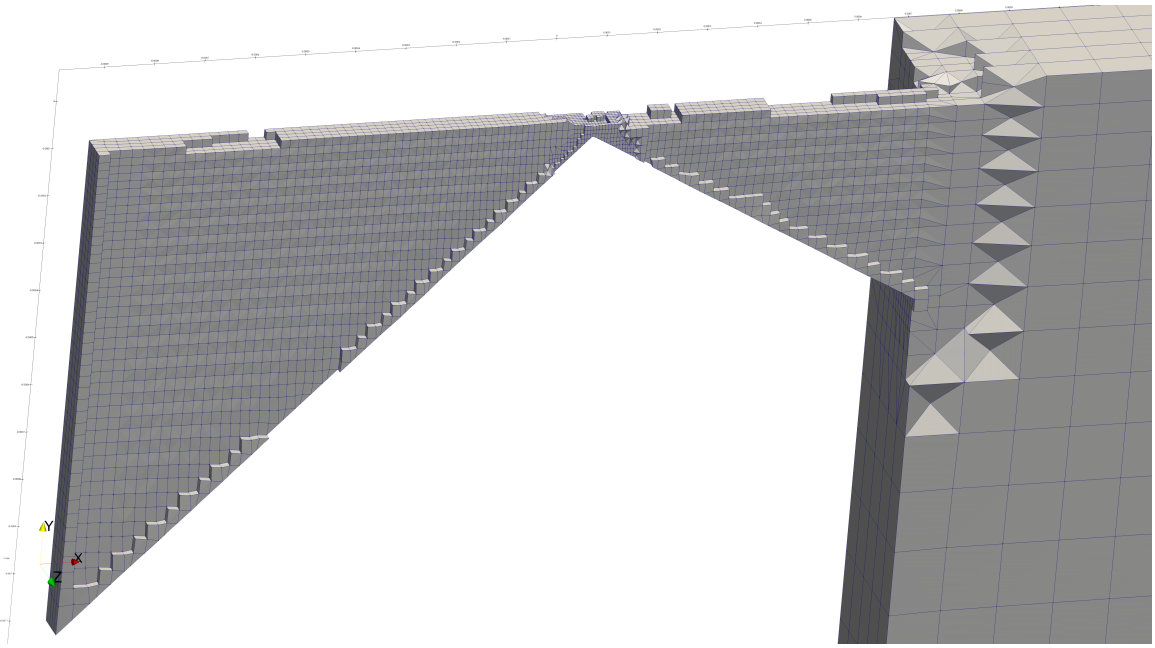



Figure 3.23: MEMS nozzle final mesh with the Extract Cells By Region Filters used to show the Y along with the Z Normal (inside extraction side with extract intersected cells) at the respective throat dimensions' midpoints to show the cross sections of the mesh

### 3.2.3. createCellZones and createFaceZones

From the microScaleTestCase tutorial in dsmcFoam+ (which is the initial source of many case files), zonesDict can be obtained to run createCellZones and createFaceZones after creating the mesh. The cellZones are used to set zoneNames for dsmcInitialise and fieldPropertiesDict's dsmcMassFluxSurface fieldModel, which are discussed in Subsections 3.2.8 and 3.2.9 respectively. The faceZones are used for dsmcFoam+'s valuable mass flow rate measurements functionality and fieldPropertiesDict, which are more statistically accurate than calculating the mass flow rate using macroscopic velocity and density values due to the significant number of particles crossing their faces [52]. It computes the macroscopic fluxes for mesh faces from the particles passing them per time step, where OpenFOAM's particle tracking algorithm is utilized for this purpose [52]. To elaborate, consider the $P$ (owner) and $Q$ (neighbor) to be cells sharing face $f$, where $\mathbf{n}_f$ is directed positive from $P$ to $Q$. For every particle $i$ crossing any face $f$, an evaluation using $sgn(\mathbf{c}_i \cdot \mathbf{n}_f)$, where $sgn$ is a sign function as shown in Equation 3.1, is done to compute the direction of movement as positive (1), negative (-1), or neutral (0) referring to movement along the relative face. With summing for face $f$, the average mass flow rate $\left(\langle \dot{m}_f \rangle\right)$ can be calculated (Equation 3.2). The mass flux is evaluated by dividing $\left(\langle \dot{m}_f \rangle\right)$ by $A_f$, the face $f$'s area.

$$sgn(\mathbf{c}_i \cdot \mathbf{n}_f) = \begin{cases} -1 & \text{if } \mathbf{c}_i \cdot \mathbf{n}_f < 0 \\ 0 & \text{if } \mathbf{c}_i \cdot \mathbf{n}_f = 0 \\ 1 & \text{if } \mathbf{c}_i \cdot \mathbf{n}_f > 1 \end{cases}, \tag{3.1}$$

$$\langle \dot{m}_f \rangle = \frac{1}{t_{av}} \sum_{i}^{\Delta N_f(t \to t_n)} F_N m_i sgn(\mathbf{c}_i \cdot \mathbf{n}_f), \tag{3.2}$$

where $t_{av}$ is the mass flow rate averaging's physical time, $F_N$ is every representative DSMC particle's real number of atoms or molecules, $m_i$ is the molecular mass, and $\Delta N_f(t \to t_n)$ is the overall number of computational particles passing face $f$ from time $t$ to $t + t_{av}$ [52]. Note that face flux files for the number of particles and their mass are generated along with the face mass flow rate measurements.

Both nozzles have similar layouts of cellZones and faceZones in their respective zone entries, where four regions are created, region1, region 2, region 3, and regionAll representing the converging, diverging, plume, and combined sections respectively using option boundBox with each of their geometrical extremities written in startPoint and endPoint as minimums and maximums respectively. For the conventional nozzle, region1's startPoint is (-0.447846e-3 0 0) and endPoint is (0 0.75e-3 0.065367e-3), region2's startPoint is (0 0 0) and endPoint is (0.329697e-3 0.75e-3 0.065367e-3), region3's startPoint is (0.329697e-3 0 0) and endPoint is (2.66233e-3 0.75e-3 0.065367e-3), regionAll's startPoint is (-0.447846e-3 0 0) and endPoint is (2.66233e-3 0.75e-3 0.065367e-3). The MEMS nozzle's region1 startPoint is (-0.9875e-3 -2e-3 -0.2e-3) and endPoint is (0 2e-3 0.3e-3), region2 startPoint is (0 -2e-3 -0.2e-3) and endPoint is (0.67117e-3 2e-3 0.3e-3), region3 startPoint is (0.67117e-3 -2e-3 -0.2e-3) and endPoint is (5.64717e-3 2e-3 0.3e-3), regionAll startPoint is (-0.9875e-3 -2e-3 -0.2e-3) and endPoint is (5.64717e-3 2e-3 0.3e-3). For the faceZones, three faces are created for each of the two nozzles, where face1 represents the outlet face of the first inlet cells, since the inlet face would result in null data, face2 represents the face at the throat, and face3 represents the face at the nozzle's outlet. The option pointToPoint is used for both face1 and face2, though boundBox is needed for face3, as it needs to be limited to the nozzle's outlet and not extending to the exteriorSides, which would occur when pointToPoint's planar method is used. Concerning the conventional nozzle, face1's startPoint is (-0.441e-3 0 0 ) and endPoint is (-0.44e-3 0 0 ), face2's startPoint is (-0.0001e-03 0 0 ) and endPoint is (0.0001e-03 0 0 ), and face3's startPoint is (0.329685e-3 -0.001e-03 -0.001e-03 ) and endPoint is (0.32985e-3 0.1501e-03 0.01307301e-03 ). Furthermore, the MEMS nozzle's face1 startPoint is (-0.9645e-03 0 0 ) and endPoint is (-0.953e-03 0 0 ), face2 startPoint is (-0.001e-03 0 0 ) and endPoint is (0.001e-03 0 0 ), and face3 startPoint is (0.665e-03 -0.4001e-03 -0.001e-03 ) and endPoint is (0.675e-03 0.4001e-03 0.101e-03 ). Note that the values are in meters. The numbers, which could be obtained manually and iteratively, are based on values that would exactly capture the desired regions or faces.

The visualization can be done in ParaView (Subsection 3.4) after their respective command executions. Respectively for the conventional and MEMS nozzles, the cellSet, cellZone, and faceZone are overlapping for the regions (Figures 3.24 and 3.26), as the faceSet and faceZones are for the faces (Figures 3.25 and 3.27). Note that regionAll corresponds to region1, region2, and region3 combined.

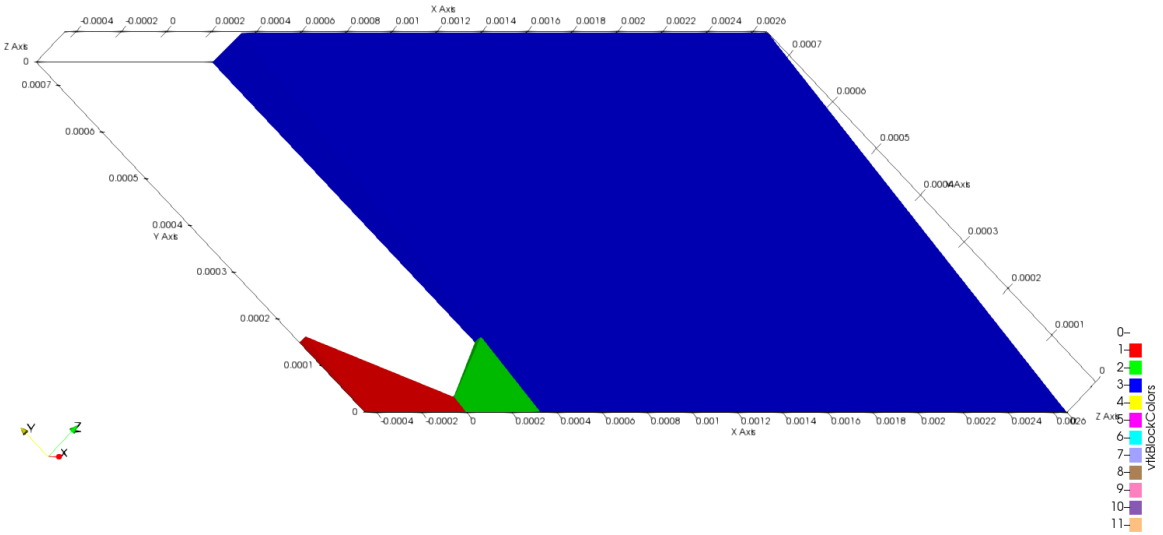



Figure 3.24: Conventional nozzle final mesh created cellZones (region1, region2, and region3 correspond to red, green, and blue respectively)

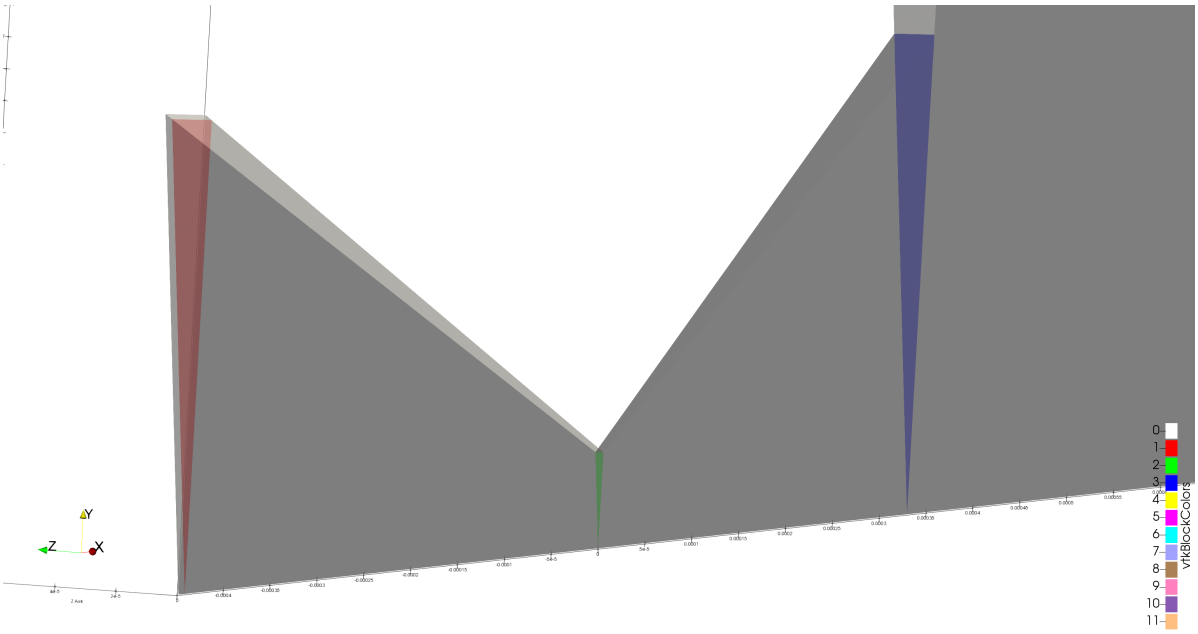



Figure 3.25: Conventional nozzle final mesh created faceZones (face1, face2, and face3 correspond to red, green, and blue respectively)

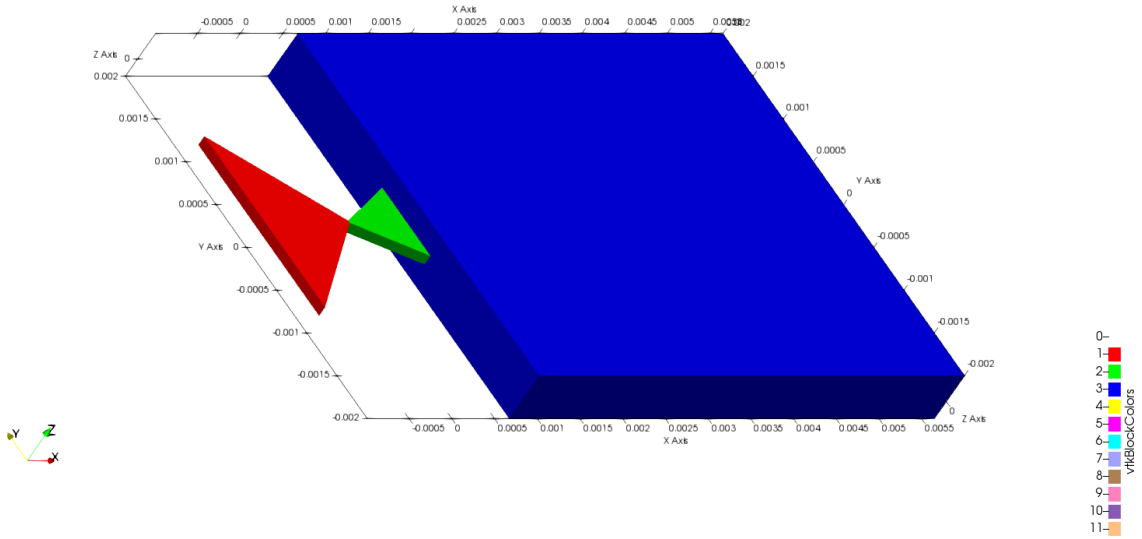



Figure 3.26: MEMS nozzle final mesh created cellZones (region1, region2, and region3 correspond to red, green, and blue respectively)

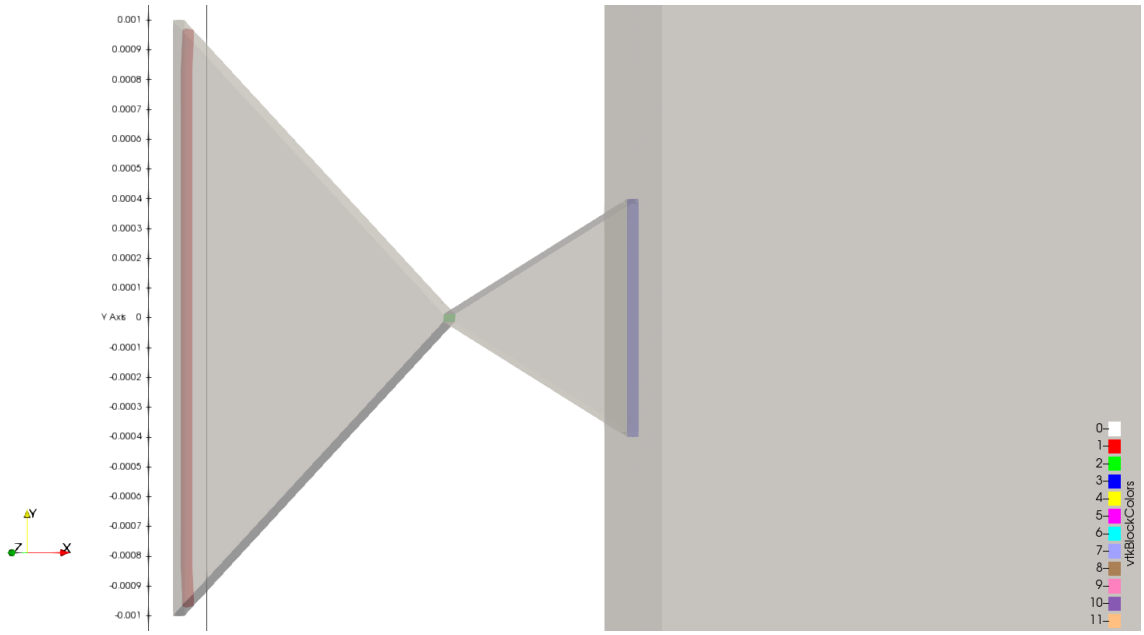



Figure 3.27: MEMS nozzle final mesh created faceZones (face1, face2, and face3 correspond to red, green, and blue respectively)

### 3.2.4. checkMesh

Executing checkMesh creates a general report about the mesh's quality and properties, as its metrics' details are explained in the mesh quality settings of the snappyHexMesh (with surfaceFeatureExtract) Subsection 3.2.2. To generally note, a failure in mesh, multiply connected surfaces patch topology, boundary openness, non-orthogonality, minimum volume, and face pyramids failures are more detrimental than skewness, aspect ratio, and minimum face area failures [30]. Cell and face sets worthy of attention are written to constant/polyMesh/cells. To provide a more detailed validity report, checkMesh -allGeometry -allTopology is additionally run where all (including non-finite-volume specific) geometry and addressing are checked due to the respectively added options. Executing checkMesh in parallel is also possible. First, checkMesh -allGeometry -allTopology is discussed as it is more comprehensive. The conventional and MEMS nozzles' mesh statistics are presented in Table 3.2.

Table 3.2: checkMesh statistics for conventional and MEMS nozzles, where $j : i : k$ refers to a vector of numbers from $j$ to $k$ with constant increments of $i$

| Mesh Statistics | Points | Internal Points | Edges | Internal Edges | Faces | Internal Faces | Cells | Faces per Cell | Overall Number of Cell Types |
|---|---|---|---|---|---|---|---|---|---|
| **Conventional Nozzle** | 87491 | 0 | 217729 | 42750 (All Using Two Boundary Points) | 173581 | 86091 | 43342 | 5.991233 | 42962 (Hexahedra)<br>380 (Prisms) |
| **MEMS Nozzle** | 51343 | N/A | N/A | N/A | 130465 | 114188 | 39702 | 6.162234 | 38309 (Hexahedra)<br>1393 (Polyhedra with 6:3:18 Faces in Generally Decreasing Order and Maximum at 12 Faces) |

It is may be unexpected to have both axisymmetric conventional wedge and 3D MEMS nozzles' grid cell numbers to be of the same order of magnitude, but it is important to note that the Lagrangian nature of DSMC simulations lays great emphasis on the computationally expensive number of simulation particles. A high level of optimization is performed to determine the best feasible simulation settings. Basically, 20 or more particles are recommended for the no-time-counter method to allow the recovery of more precise collision statistics and possibly less are needed when the Knudsen number is sufficiently high and collisions become less relevant [52]. This certainly presents a greater challenge for both nozzles, and especially the MEMS nozzle, where greater rarefaction is expected beyond the throat generally due to the geometry and constricting size of the throat. Therefore, this calls for a much higher number of simulation particles along with relatively and acceptably larger grid cells past the throat as necessarily attempted to be able to accurately represent the flow, the former of which leads to a larger computational load. Considering this work's goals, the numbers of grid cells used following many iterations starting from well crafted finer meshes has proven to be adequate. Nevertheless, the single cell thick mesh of the axisymmetric conventional nozzle allows for a significant particle number decrease compared to when simulating its conical 3D shape.

In the topology section, both of the conventional and MEMS nozzle meshes' metrics are OK with one region. To note, the boundary definition evaluates the boundary mesh for uniform patch face addressing, point usage checks for points that are unused, upper triangular ordering checks are done for the internal faces, and face vertices checks tell whether the face vertices are unique and within point range [30]. Note that 71 cells with two non-boundary faces written in the twoInternalFacesCells set (located at the converging/diverging sides and mesh corners) for the conventional nozzle's mesh. In the patch topology for multiply connected surfaces, the used patches have ok surface topology and are non-closed singly connected. In the geometry check, the values obtained for the conventional nozzle's mesh are a boundary openness of (-2.003388e-18 -9.382614e-17 -3.096731e-15), maximum cell openness of 2.5483e-16, maximum aspect ratio of 78.86849, minimum face area of 4.975064e-13, maximum face area of 7.593534e-10, minimum cell volume of 1.658353e-18, maximum cell volume of 4.166844e-15, and total cell volume of 5.750106e-11, mesh non-orthogonality with maximum of 20.01674 and average of 0.6380222, maximum skewness of 2.420073, coupled point location match average of 0, minimum/maximum edge lengths of 2.947834e-07/6.543724e-05, face flatness, where 1 is flat and 0 is butterfly, with minimum of 0.9684396 and average of 0.9999998, face interpolation weight with minimum of 0.3113614 and average of 0.497019, and face volume ratio with minimum of 0.3316159 and average of 0.9767763. All mesh metrics are OK, except for 2 failed mesh checks concerning the cell determinant (wellposedness) with a minimum of 1.092612e-22 and average of 1.261654e-07 to result in 43340 underdeterminedCells with small determinant (< 0.001) found (relatively similar to checkMesh -allGeometry -allTopology on blockMesh (Subsection 3.2.1) for the conventional nozzle along with a few

twoInternalFacesCells on the mesh corners, as it and checkMesh return Mesh OK for both conventional and MEMS nozzle meshes otherwise) and concave cells (using face planes) totaling 38 concaveCells (located at converging/diverging sides). The data is given as is, considering its self-explanatory units.

For the MEMS nozzle's mesh, a boundary openness of (-3.577964e-17 1.369116e-17 -3.558038e-17), maximum cell openness of 3.582136e-16, maximum aspect ratio of 12.49687, minimum face area of 6.446744e-12, maximum face area of 2.240224e-08, minimum cell volume of 7.683789e-17, maximum cell volume of 1.283997e-12, and total cell volume of 1.007964e-08, mesh non-orthogonality with maximum of 50.4801 and average of 9.949445, maximum skewness of 3.663984, coupled point location match average of 0, minimum/maximum edge lengths of 1.972777e-07/0.000189699, face flatness, where 1 is flat and 0 is butterfly, with minimum of 0.8899717 and average of 0.9998181, cell determinant (wellposedness) with minimum of 0.01403129 and average of 8.707453, face interpolation weight with minimum of 0.1407596 and average of 0.4798149, and face volume ratio with minimum of 0.06509031 and average of 0.8949648 are found. All mesh metrics are OK, except for 1 failed mesh check concerning the concave cells (using face planes) totaling 1948 concaveCells (located mainly in the nozzle section). Additionally, with only one asterisk for attention instead of three, there are 88 faces with concave angles between consecutive edges with a maximum concave angle of 68.99542 degrees written to set concaveFaces (located at converging/diverging sides and around the throat section's connection). The conventional and MEMS nozzles at this point have non-constrained vector mesh (non-empty, non-wedge) directions (1 1 1) and mesh (non-empty) directions (1 1 1). None of the given errors resulted in an unfixable crashing simulation. A word of advice, a mesh check error in face tets (low quality or negative volume decomposition tets (tetrahedrons) written as lowQualityTetFaces set and checkable with checkMesh -allGeometry -allTopology) compared to OK face tets might result in these Lagrangian simulations to indefinitely hang/freeze along with warnings/errors when running dsmcInitialise. Also, running checkMesh without the additional options shows no failures or written sets, as all of the mesh quality metrics for both conventional and MEMS nozzle meshes are adequate. With the proper judgement and attention to detail, ParaView should secondarily be used to check the mesh with the rule of thumb that, if the mesh also looks good, then it probably is (to start with), noting that it also has a Mesh Quality Filter for measuring geometrical cell fitness. Generally speaking, the meanings of the mesh quality metrics and their preferred values for the grid are describable by each of their own term's self-explanatory relation with more intuitively balanced cells. Also, note that the evolution of the mesh should also be preferably smooth.

For evaluation, the grid cell ($\Delta x_{cell}$) and time step ($\Delta t_{step}$) sizes are calculated in Table 3.3 for the four considered cases of pressure and temperature combinations of both conventional and MEMS nozzles. The pressure and temperature values are chosen for comparison with ANSYS Fluent results in Chapter 4. As done in the Analytical Model Subsection 4.2.1, water's density and dynamic viscosity values are obtained at the selected inlet pressure and temperature from the US National Institute of Standards and Technology's (NIST) publicly accessible database of thermodynamic data. Then, Equation 2.56 is used to calculate the mean free path with a temperature coefficient of viscosity of 1.0855 obtained from a report by Sandia National Laboratories [10] to be discussed under dsmcProperties Subsection 3.2.6, mass of a single water molecule as $2.991 \cdot 10^{-26}$kg found by dividing the molar mass $\left(M = 18.015 \frac{\text{g}}{\text{mol}}\right)$ by the Avogadro constant $\left(N_A = 6.022140857 \cdot 10^{23}\text{mol}^{-1}\right)$, and Boltzmann constant $\left(k = 1.38064852 \cdot 10^{-23}\text{J} \cdot \text{K}^{-1} = 1.38064852 \cdot 10^{-23}\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1}\right)$ followed by the maximum of the inequality Equation 2.54 for the maximum recommended grid cell size. Afterwards, a rough method to compute the used average grid cell size for the conventional and MEMS nozzles is implemented by calculating the cube root (to obtain the meters dimension instead of volumetric cubic meters assuming that the grid cell is a cube) of the total cell volume obtained from checkMesh divided by the total number of cells in the mesh. Using this conservative method, the grid cell sizes used are a few orders of magnitude larger than recommended, though this does not account for the greater concentration of grid cells around the more important nozzle region and that larger grid cells are needed past the throat, where the flow is more rarefied and the pressure is lower, to maintain a feasible simulation runtime. Next, the mean thermal speed is calculated using Equation 2.57 to be used by the recommended time step size calculation (Equation 2.55) along with the mean free path and a $\xi$, which is a fraction for the time step size to be a fraction of the calculated mean free time, that is the average time between collisions at mean stream conditions, of 1 for relative scaling. Note that the values of the used parameters are repeated as used for Equation 2.56. In a different approach and perspective

concerning the used grids, the mean free path can be recalculated from the used average grid cell size for the conventional and MEMS nozzle meshes as the maximum grid cell size in the inequality Equation 2.54 and then used in Equation 2.55 for the expected time step size. However, note that the grid is nonuniform with a finer mesh at the nozzle region. The used time step size could be considered close to the recalculated time step size multiplied by a fraction $\xi$ and an order of magnitude larger than the recommended time step size before multiplying it by the fraction $\xi$. Its application and usage are explained in the boundariesDict Subsection 3.2.5 and controlDict Subsection 3.2.7, as it has to do with the stability of the simulation using a pressure boundary condition, especially when the grid cell size used is not the recommended grid cell size. Also, the flow is expected to become cooler and more rarefied along the nozzle. Note that the time step size should be smaller than the mean collision time, which is the successive particle collision time on average, for proper movement and collision steps decoupling [54]. Furthermore, the grid cell and time step size calculations' results for all four cases with different pressures/temperatures are relatively comparable for the intended purposes in this section. The Courant-Friedrichs-Lewy number ($CFL$) number is also calculated using Equation 2.58 following the most probable molecular speed ($c'_m$) calculation (Equation 2.59) using the universal gas constant ($R_A = 8.3144598\text{J} \cdot \text{K}^{-1} \cdot \text{mol}^{-1} = 8.3144598\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1} \cdot \text{mol}^{-1}$) and water molecular mass ($M_w = 0.018\text{kg} \cdot \text{mol}^{-1}$). The $CFL$ number proves to be acceptable below one for the used average grid cell and time step sizes compared to the maximum recommended grid cell and time step sizes due to the relative smallness of the time step size, which means that the time step size can keep the DSMC particles in their grid cells for a sufficient period at the most probable molecular speed for accuracy in their physical interaction with other particles rather than CFD's general usage for stability.

### 3.2.5. boundariesDict

Next, the setup for the cases after creating the meshes and before running the simulations is explained. In system/boundariesDict, the dsmcPatchBoundaries entry, which concerns immediate actions for patch boundary collisions, for both conventional and MEMS nozzle cases includes boundary for the walls, where patchBoundaryProperties and its patchName used applies for converging, diverging, and exteriorSides along with the throat for the MEMS nozzle, due to the separated section used, as it is correlated to mesh creation, and boundaryModel of dsmcDiffuseWallPatch is used, where the wall interaction uses the Maxwellian distribution for a defined temperature to set a random microscopic velocity. The impermeable solid surface wall boundary makes molecules rebound, though it is not ideal, such as a pool ball bouncing off the rail cushion as in specular scattering. The transfer of momentum and energy between a gas flow and solid surface is attributed to collisions of which's process is named gas-surface interaction [40]. To elaborate on this gas-surface interaction's theoretical approach and its implementation as a boundary condition in kinetic theory analysis, its modeling concepts of interaction parameters and scattering kernel are described. The interaction parameters for gas-surface interaction models greater in scale than a nanometer are applicable macroscopically as accommodation coefficients, which numerically explains the gas flow's degree of kinetic or thermal accommodation with an interacting solid surface and given for a molecular property ($Q$) (e.g. total energy or normal/tangential momentum) as:

$$a_Q = \frac{\Phi_i^Q - \Phi_r^Q}{\Phi_i^Q - \Phi_w^Q}, \tag{3.3}$$

where $\Phi_i^Q$, $\Phi_r^Q$, and $\Phi_w^Q$ refer to the incident, reflected, and full accommodation reflected fluxes of $Q$ respectively [40].

Furthermore, the scattering kernel is described for the gas-surface interaction modeling concept of which its formulation is a formal mathematical construct [40]. Consider that $K(\vec{\xi}_i, \vec{\xi}_r)$ is a scattering kernel for the probability density that an incident molecule with velocity $\vec{\xi}_i$ reflects with velocity $\vec{\xi}_r$ under the same circumstances, as it links the incident ($f_i(\vec{\xi}_i)$) and reflected ($f_r(\vec{\xi}_r)$) velocity distribution functions as shown in the integral transform Equation 3.4 with independent gas-surface interactions and a temporal evolution of $f$ relatively greater than the average interaction time [40].

$$\xi_{n,r} f_r(\vec{\xi}_r) = \int_{\xi_{n,i}<0} |\xi_i| f_i(\vec{\xi}_i) K(\vec{\xi}_i, \vec{\xi}_r) d\vec{\xi}_i, \tag{3.4}$$

Table 3.3: Grid cell ($\Delta x_{cell}$) and time step ($\Delta t_{step}$) size calculations

| $P$ **(bar)** | 5 | 5 | 7 | 7 |
|---|---|---|---|---|
| $T$ **(K)** | 550 | 773 | 550 | 773 |
| $\mu$ **(Pa · s)** | 0.000019269 | 0.000028573 | 0.00001924 | 0.000028576 |
| $\rho\left(\frac{\mathbf{kg}}{\mathbf{m^3}}\right)$ | 1.9984 | 1.4069 | 2.8145 | 1.9726 |
| $\lambda$ **(m)** | 1.3906E-08 | 2.47063E-08 | 9.85889E-09 | 1.76229E-08 |
| **Maximum Recommended Grid Cell Size ($\Delta x_{cell}$) (m)** | 4.63532E-09 | 8.23545E-09 | 3.2863E-09 | 5.87431E-09 |
| **Total Volume from checkMesh Geometry (Conventional Nozzle) (m$^3$)** | 5.75E-11 | 5.75E-11 | 5.75E-11 | 5.75E-11 |
| **Total Number of Cells (Conventional Nozzle)** | 43342 | 43342 | 43342 | 43342 |
| **Used Average Grid Cell Size ($\Delta x_{cell}$) (Conventional Nozzle) (m)** | 1.10E-05 | 1.10E-05 | 1.10E-05 | 1.10E-05 |
| **Total Volume from checkMesh Geometry (MEMS Nozzle) (m$^3$)** | 1.00796E-08 | 1.00796E-08 | 1.00796E-08 | 1.00796E-08 |
| **Total Number of Cells (MEMS Nozzle)** | 39702 | 39702 | 39702 | 39702 |
| **Used Average Grid Cell Size ($\Delta x_{cell}$) (MEMS Nozzle) (m)** | 6.33E-05 | 6.33E-05 | 6.33E-05 | 6.33E-05 |
| $\bar{c}\left(\frac{\mathbf{m}}{\mathbf{s}}\right)$ | 804.0531565 | 953.2200384 | 804.0531565 | 953.2200384 |
| **Recommended $\Delta t_{step}$ Using $\lambda$ and $\xi$ of 1 (s)** | 1.72948E-11 | 2.59188E-11 | 1.22615E-11 | 1.84878E-11 |
| **$\lambda$ Recalculated from Used Average Grid Cell Size ($\Delta x_{cell}$) (Conventional Nozzle) (m)** | 3.30E-05 | 3.30E-05 | 3.30E-05 | 3.30E-05 |
| **$\Delta t_{step}$ Using Recalculated $\lambda$ from Used Average Grid Cell Size ($\Delta x_{cell}$) and $\xi$ of 1 (Conventional Nozzle) (s)** | 4.10E-08 | 3.46E-08 | 4.10E-08 | 3.46E-08 |
| **Used $\Delta t_{step}$ (Conventional Nozzle) (s)** | 5E-10 | 5E-10 | 5E-10 | 5E-10 |
| **$\lambda$ Recalculated from Used Average Grid Cell Size ($\Delta x_{cell}$) (MEMS Nozzle) (m)** | 1.90E-04 | 1.90E-04 | 1.90E-04 | 1.90E-04 |
| **$\Delta t_{step}$ Using Recalculated $\lambda$ from Used Average Grid Cell Size ($\Delta x_{cell}$) and $\xi$ of 1 (MEMS Nozzle) (s)** | 2.36E-07 | 1.99E-07 | 2.36E-07 | 1.99E-07 |
| **Used $\Delta t_{step}$ (MEMS Nozzle) (s)** | 5E-10 | 5E-10 | 5E-10 | 5E-10 |
| $c'_m\left(\frac{\mathbf{m}}{\mathbf{s}}\right)$ | 712.8154577 | 845.0560419 | 712.8154577 | 845.0560419 |
| **$CFL$ Using Maximum Recommended Grid Cell ($\Delta x_{cell}$) and Time Step ($\Delta t_{step}$) Sizes** | 2.66 | 2.66 | 2.66 | 2.66 |
| **$CFL$ from Used Average Grid Cell ($\Delta x_{cell}$) and Time Step ($\Delta t_{step}$) Sizes (Conventional Nozzle)** | 3.24E-02 | 3.85E-02 | 3.24E-02 | 3.85E-02 |
| **$CFL$ from Used Average Grid Cell ($\Delta x_{cell}$) and Time Step ($\Delta t_{step}$) Sizes (MEMS Nozzle)** | 5.63E-03 | 6.67E-03 | 5.63E-03 | 6.67E-03 |

Respectively, the conditions of positivity, normalization, and reciprocity are to be validated by the formulation, where the reciprocity criterion represents the gas flow and solid surface interaction equilibrium condition to be satisfied:

$$K(\vec{\xi}_i, \vec{\xi}_r) \geq 0, \tag{3.5}$$

$$\int_{\xi_{n,r}>0} K(\vec{\xi}_i, \vec{\xi}_r) d\vec{\xi}_r = 1, \tag{3.6}$$

$$|\vec{\xi}_{n,i}| f_M(\vec{\xi}_i) K(\vec{\xi}_i, \vec{\xi}_r) = |\vec{\xi}_{n,r}| f_M(\vec{\xi}_r) K(-\vec{\xi}_i, -\vec{\xi}_r), \tag{3.7}$$

where $f_M$ is the Maxwellian velocity distribution at solid surface equilibrium, similar to Equation 2.45, as $f(\vec{\xi})d\vec{\xi}$ is a molecule's probability of having velocity $\vec{\xi}$ inside the bounds of $d\vec{\xi}$:

$$f_M(\vec{\xi}) d\vec{\xi} = (2\pi R T_{wall})^{-\frac{3}{2}} \exp\left(-\frac{\xi^2}{2RT_{wall}}\right) d\vec{\xi}, \tag{3.8}$$

The Maxwell model is the oldest and most popular kinetic theory gas-surface interaction model used for DSMC and it applies to specular and diffuse interactions resulting from molecular collisions with ideal reflecting and accommodating surfaces respectively [40, 41]. Scattering distributions are probability plots representing the gas-surface reflection or scattering angle between the surface and reflected molecular velocity [41]. Figure 3.28 shows the Maxwell model scattering distribution polar plots for a molecular beam with a set surface incident angle, where specular and diffuse reflections appear as a relatively more protruding sharp oval and circle respectively, which means that the composite distribution has a circular shape with an extended peak [41]. At identical beam molecular velocities, the specular reflection angle peak becomes a line [41]. To note, other models also exist, as Maxwell's model, which became more prominent during the Space Age, becomes less accurate for high speed molecular beam experiments, which result in uncentered petal shaped scattering distributions [41].

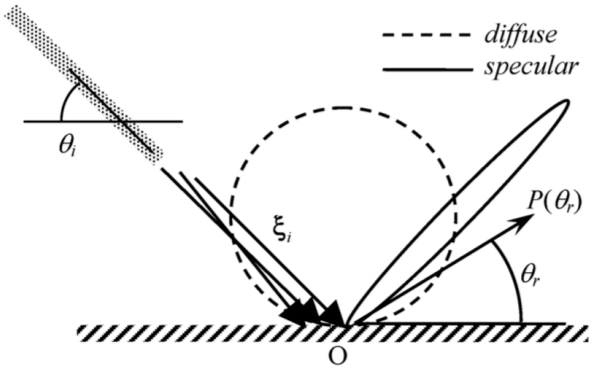



Figure 3.28: Molecular beam with a set surface incident angle Maxwell model scattering distribution [41]

Incident molecules reflect elastically from an ideally flat solid surface's molecular structure for specular reflections, as it is assumed to happen for collisions of gas molecules, as rigid elastic spheres, and surface peaks, with the solid molecules as rigid elastic spheres, where the molecule's velocity surface normal and tangential components are inverted and unaltered respectively, angles of reflection and incidence are equal, and molecule thermal energy is assumedly kept unchanged [40]. Specular reflections can provide information about the normal stress on the surface, but not the heat flux and shear stress. For diffuse reflections, the incident molecule reflects/evaporates from the solid surface's molecular structure based on the local surface temperature Maxwellian velocity distribution after reaching thermal equilibrium [40, 41]. For a short time, the surface absorbs an incident molecules' fraction ($a_M$), which can describe the diffuse reflection probability and does not describe a flux ratio, to diffusely reflect them afterwards, while the rest of the incident molecules interact through specular reflections [40]. The criteria of positivity, normalization, and reciprocity are satisfied by its scattering kernel:

$$K_M(\vec{\xi}_i, \vec{\xi}_r) = (1 - a_M)\delta(\vec{\xi}_i - \vec{\xi}_{r,specular}) + a_M f_M(\vec{\xi}_r)|\vec{\xi}_r \cdot \mathbf{n}|, \tag{3.9}$$

where $\vec{\xi}_{r,specular}$ is the specular reflection molecular velocity and $\mathbf{n}$ is the solid surface's local normal unit vector [40].

Note that the following logic is valid for a variable initial wall/possible (regenerative along with potential film, curtain, transpiration, and radiation) cooling (to avoid melting in different conditions considering that the (stored) inlet microresistojet temperature could be assumed as 283.16 K [27]) temperature along with decreasing viscosity (before the throat, while still being increased beyond it considering the de Laval nozzle temperature behavior), as gas viscosity generally increases as temperature increases due to the gas molecular collisions increase, contrary to the liquid viscosity decrease with a temperature increase, considering that it decreases the dominant cohesive force between the liquid molecules, though in this case it is also used for analyzing the heat transfer as discussed (with alternatives and next step recommendations) in Chapter 4's Pressure and Temperature Subsection 4.1.4. The dsmcDiffuseWallPatchProperties have a set velocity of (0 0 0) and temperature of 300 (K), which is in reference to an approximate average of the varying surface temperatures of the International Space Station (ISS) in low Earth orbit due to similar operating conditions, where the Sun-facing side gets as hot as 121 °C/394 K and drops to -157 °C/116 K on the dark side in the absence of thermal controls

along with partial considerations for the thrusters' possibly expected warmer operating conditions due to the vessel [46]. Figure 3.29 shows the significant variation in surface temperatures on the ISS (-250 °F/-157 °C/116 K to 250 °F/121 °C/394 K) compared to a normal house on Earth (-15 °F/-26 °C/247 K to 110 °F/43 °C/316 K) [46].

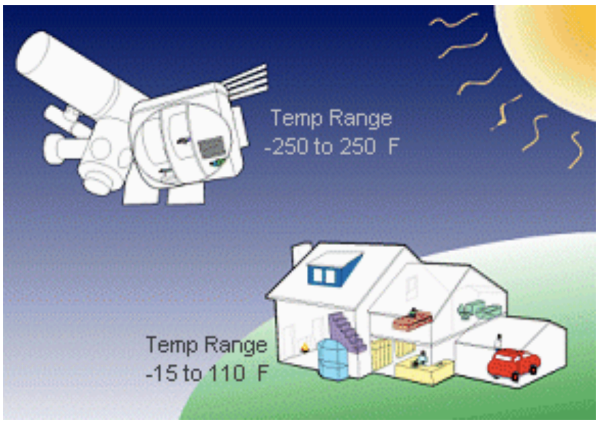



Figure 3.29: International Space Station's varying surface temperature range due to the Sun's view factor compared to a normal house on Earth [46]

Moreover, using patchName inlet and exterior in patchBoundaryProperties while separately set as boundary for both conventional and MEMS nozzles, their boundaryModel is set as dsmcDeletionPatch with dsmcDeletionPatchProperties as true for allSpecies. The dsmcDeletionPatch can act as an immediate full deletion boundary patch for dsmcParcels (collection of particles) colliding with a boundary face. It is applicable as a vacuum boundary condition as used for the exterior. As an additional note, selectively erasing certain species while others undergo specular reflection can be done, so this boundary can be used for different purposes as well. The flow for the outlet is assumed to exit only due to its high speed (external particle thermal velocities are predictably not sufficiently high for reentering the computational domain) and otherwise vacuum conditions, while the stochastic flow at the inlet can realistically go both ways, so for the boundary in the dsmcGeneralBoundaries entry, which concerns boundary molecules time dependent actions, the generalBoundaryProperties patchName is inlet, the boundary model is dsmcWangPressureInlet, and the dsmcWangPressureInletProperties include the typeIds of (H2O), which is discussed in the dsmcProperties Subsection 3.2.6, moleFractions with H2O of 1.0, and inletPressure and inletTemperature each independently set for all four combinations of 500000 (Pa) or 700000 (Pa) and 550 (K) or 773 (K) respectively.

Subsonic flow pressure driven implicit boundary conditions are available and applicable for rarefied micro gas flows [53]. Of the various developed forms, the initially common particle flux conservation's large statistical noise causes inaccuracies [53]. Recently, the theory of characteristics based boundary condition forms, which is considered the most popular and efficient method, employs time averaged macroscopic properties derived from boundary cells [53]. Note that the boundary conditions are adapted for DSMC, but originate from continuum CFD boundary conditions, which do not account for possible rarefied gas flow phenomena unobserved in continuum flows [53]. Usually, conventional DSMC boundaries could be hypersonic and for typical MEMS flows, the particle thermal velocities are expected to be greater than the stream velocity considering the boundary condition's location, which leads to the possibility of particles entering and leaving the boundary [53]. The macroscopic velocity, temperature, and density must be known in DSMC to find the accurate boundary number flux, as the equilibrium Maxwell-Boltzmann distribution (Equation 3.10), along with using Equations 3.11 and 3.12, is used to compute the imposed particle flux ($\dot{N}$) at a surface normal angle ($\theta$) for a boundary area ($A$):

$$\frac{\dot{N}}{A} = \frac{nV_{mp}\{\exp(-q^2) + \sqrt{\pi}q[1 + \mathrm{erf}(q)]\}}{2\sqrt{\pi}}, \tag{3.10}$$

$$q = \frac{V}{V_{mp}}\cos\theta, \tag{3.11}$$

$$V_{mp} = \sqrt{\frac{2kT}{m}}, \tag{3.12}$$

where $V$ is the local stream velocity, $V_{mp}$ is the local most probable thermal velocity, $n$ is the boundary number density, $T$ is the boundary macroscopic temperature, $m$ is the molecular mass, and $k$ is the Boltzmann constant [53]. Note that the Maxwellian form boundary condition is assumed due to the difficulty of determining the boundary velocity distribution function form, as it may not be Maxwellian [54]. Additionally, the overall temperature generally derives the added particles' translational and kinetic energies, as the pressure can be found from the translational temperature, considering its exclusive translational kinetic particle motion dependence, where ideal thermodynamic equilibrium, which could be more difficult to achieve throughout the nozzle in this higher inlet to outlet pressure ratio and lower aspect ratio compressible case, suggests that the gas has an overall temperature that is equivalent to the translational and rotational temperatures [53]. Note that for typical MEMS applications, a pressure driven gas flow results in expansion, cooling, and density drop, where the drop in thermal energy first happens in the translational mode to be balanced with the rotational mode by intermolecular collisions following a relative relaxation time, which is not high as a rotational relaxation [53]. However, the decrease of molecular collisions in more rarefied flows may cause the translational and rotational energies to remain different, which would lead to a local translational energy and pressure drop relative to an equilibrium gas with an identical density and total energy [53]. The used boundary condition assumes rotational equilibrium at the inlet [53].

The upstream inlet boundary is based on theory of characteristics, where the gas pressure ($P_{in}$) and temperature ($T_{in}$) are set and the ideal gas law is employed for computing the number density at the inlet ($n_{in}$):

$$n_{in} = \frac{P_{in}}{kT_{in}} \tag{3.13}$$

The boundary face ($f$) at the inlet's stream-wise ($u_{in}$) and tangential ($v_{in}$) velocities are computed using the theory of characteristics:

$$(u_{in})_f = u_j + \frac{P_{in} - P_j}{\rho_j a_j}, \tag{3.14}$$

$$(v_{in})_f = v_j, \tag{3.15}$$

where $u_j$ and $v_j$ are the associated boundary face cells' first order extrapolations of stream-wise and tangential velocities respectively, $\rho$ is the mass density, $a$ is the local speed of sound, and boundary cell values are designated by the subscript $j$ [53].

Therefore, considering that the subsonic flow at the inlet is expected to be in the continuum regime in contrast to the flow downstream, the pressure boundary condition for internal flows is used, which is especially useful in microfluidics, where pressure boundary conditions at the inlet and outlet are usually provided, along with the temperature, considering them being easier to measure experimentally compared to the typical DSMC velocity and determinable number density Dirichlet boundary conditions usually employed for external flows using a farther outlet. To note, both boundary conditions provide relatively comparable results when preliminarily tested, as the number density can be obtained from the ideal gas law (Table 3.4 using Equation 3.13, where the Boltzmann constant $k = 1.38064852 \cdot 10^{-23}\text{J} \cdot \text{K}^{-1} = 1.38064852 \cdot 10^{-23}\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1}$) and velocity from Table 4.5 as calculated in the Analytical Model Subsection 4.2.1, though the pressure boundary condition is ultimately used for the sake of possible accuracy and novelty. As an example of the dsmcFreeStreamInflowPatch boundary model, it uses the typeIds of (H2O) and translationalTemperature, rotationalTemperature, vibrationalTemperature, and electronicTemperature possibly as the inletTemperature or 0 (K) as applicable in order, and the $x$ velocity and numberDensities (for H2O) to be determined as mentioned and set. Note that the inlet number density in Table 3.4 shows an increase with higher pressure and decrease with higher temperature as expected.

Table 3.4: Calculated inlet number density ($n_{in}$) for the applied simulation cases using Equation 3.13

| **Inlet Temperature ($T_{in}$) (K)** \ **Inlet Pressure ($P_{in}$) (bar)** | 5 | 7 |
|---|---|---|
| 550 | $n_{in} = 6.58452 \cdot 10^{25} \frac{\text{molecules}}{m^3}$ | $n_{in} = 9.21833 \cdot 10^{25} \frac{\text{molecules}}{m^3}$ |
| 773 | $n_{in} = 4.68498 \cdot 10^{25} \frac{\text{molecules}}{m^3}$ | $n_{in} = 6.55897 \cdot 10^{25} \frac{\text{molecules}}{m^3}$ |

### 3.2.6. dsmcProperties

The dsmcProperties dictionary can be found in the constant directory. Its general properties section includes setting a value for nEquivalentParticles, which defines the number of real gas atoms/molecules represented by each DSMC simulation particle. Typically, a value is set to reach an expected minimum of 20 particles per grid cell as explained in the checkMesh Subsection 3.2.4 [52]. In some cases with generally uniform Knudsen number (rarefaction) and grid cell sizes, the average grid cell size as found in Table 3.3 from the checkMesh Subsection 3.2.4 and molecule number density found from the ideal gas law (Equation 3.13 for the simulation domain) can predict the maximum nEquivalentParticles number by multiplying the average grid cell size in cubic meters per grid cell by the molecule number density in molecules per cubic meter and then dividing the resulting molecules per grid cell value by nEquivalentParticles in molecules per particle to obtain the final particle per grid cell value to be set as 20 or above [52]. However, in the case of the simulated de Laval nozzles, their grid cell sizes vary along with the Knudsen number (rarefaction) due to the accelerating flow and throat section, which additionally results in the diverging section having less particles per grid cell compared to the converging section, along with the initially vacuum and larger plume region. Therefore, nEquivalentParticles is ultimately set based on iterations with 3e4 and 1e9 for the conventional and MEMS nozzles respectively. The general guidelines followed are through iteratively checking the grid cells' DSMC particle count after the throat in the diverging section, which makes it one of the major challenges for decreasing computational time considering the Knudsen number increase along the nozzle. Additionally, the chargedParticles and adsorption settings are set to false. Considering the axisymmetric properties, the axisymmetricSimulation flag is set to false for the MEMS nozzle, while it is true for the conventional nozzle along with a radialExtentOfDomain, which is the maximum radial extremity of the wedge geometry, of 0.00075 (m) and maxRadialWeightingFactor of 10000.0. As DSMC particles represent a number of real atoms/molecules, radial weighting factors based on the radial position of grid cell centers are used for the conventional nozzle's axisymmetric simulation, where particles moving radially away from or towards the radial center could probabilistically (depending on the old and new weighting factor ratio) be discarded (due to the new larger weighting factor) or cloned (due to the new smaller weighting factor) respectively. Particle duplication is relatively less straightforward in application than particle deletion [14]. Typically for a steady flow, a global flow field duplication buffer and its internally randomly located duplicate molecule appearance time delay are used to make it less probable for the coincident duplicated molecules to collide with each other, where the copied molecule is put in the duplication buffer and another molecule initially at its address is added to the flow following a random delay based on the buffer size's average value [14]. Since the weighting factors are cell-based, which depend on the particle's respective grid cell radius, a molecule traveling parallel to the axis of symmetry can enter a grid cell that has a different height leading to a different associated weighting factor along with that its usage has shown complications in retaining normal to the axis of symmetry flow gradients that are smooth [14]. In addition to the typical scatter effects of axially symmetric flows and recommended higher reference to innermost cell radii ratio, the generally recommended molecule-based weighting factors, which depend on the molecule's radius, might inefficiently need more intricate collision coding, so that the grid cell's model molecules constitute a varying real atom/molecule number [14]. Nevertheless, an average grid cell weighting factor for its respective molecules has shown a negligible flow effect concerning the collision routines, where the number of molecular choices is multiplied by the average weighting factor, though every collision is considered as the number of selfsame collisions equivalent to the average weighting factor without needing to alter the collision procedures [14]. Locally, the radial weighting factor ($RWF$) is determined as:

$$RWF = 1.0 + maxRadialWeightingFactor\left(\frac{r}{radialExtentOfDomain}\right), \tag{3.16}$$

where $r$ is the respective grid cell center radius from the axis of symmetry. Therefore, a higher

maxRadialWeightingFactor or lower radialExtentOfDomain leads to a higher $RWF$ resulting in a greater radial particle variance. Particles very close to the axis of symmetry ($r \sim 0$) have an $RWF$ of approximately one and as $r$ increases, $RWF$ increases and fewer DSMC particles represent the real atoms/-molecules. This is employed to rebalance the grid cell size's particle per grid cell number difference, where the wedge's grid cells increase in size (volume) along the radial direction leading to more particles per grid cell away from the axis of symmetry, while a minimum of 20 particles per grid cell number is required (as explained in checkMesh Subsection 3.2.4), which would result in a much more computationally expensive simulation due to the significantly higher number of particles needed if done without the particle radial weighting factors [52]. As the radialExtentOfDomain option is based on the geometry, the maxRadialWeightingFactor needs to be calibrated to achieve a relatively uniform radial particle number density by realistically checking the radial DSMC particle number density (dsmcRhoN) along the geometry iteratively, where a higher maxRadialWeightingFactor is set if there are fewer particles close to the axis of symmetry and vice versa. Note that for the conventional nozzle's mesh, there is a greater number of grid cells in the radial direction, which means that the smallest and largest grid cells close to and away from the axis of symmetry becomes relatively smaller and larger with more grid cells respectively, as the smallest and largest grid cells' larger and smaller sections are respectively allocated to other grid cells leading to a relatively greater size ratio between them and the need for a greater maxRadialWeightingFactor. Also, the plume region's larger size (radialExtentOfDomain) leads to a smaller effect for $RWF$ inside the more important nozzle region, as shown in Equation 3.16, and relatively larger grid cells in the radial direction, which are additional motives for setting a relatively higher maxRadialWeightingFactor. Furthermore, the number of grid cells at the axis of symmetry is increased by radial grading, considering the preemptively larger than expected necessary plume region created and that the additional refinement is useful for the throat region's smaller size, which leads to relatively smaller grid cells near the axis of symmetry and larger needed maxRadialWeightingFactor.

Then, the suitable and widely used BinaryCollisionModel selected for accepted dual particle collisions is LarsenBorgnakkeVariableHardSphere using generally standard LarsenBorgnakkeVariableHardSphereCoeffs with Tref of 300 (K) (as related to moleculeProperties), rotationalRelaxationCollisionNumber of 5.0, and electronicRelaxationCollisionNumber of 500.0. The Larsen-Borgnakke (LB) and variable hard sphere (VHS) models are introduced in the DSMC Models Overview Subsection 2.4.1 and generally discussed along with the noTimeCounter collisionPartnerSelectionModel used, which can determine the collision attempt number per computational cell, in the OpenFOAM dsmcFoam(+) Solver Subsection 2.4.2. To note, the LarsenBorgnakkeVariableHardSphere BinaryCollisionModel has LB internal energy redistribution and originates from the INELRS subroutine in Bird's DSMC0R.FOR program. The overall methods are more elaborately described in [14].

In addition, the thrusters' water propellant molecular species are introduced with typeIdList (H2O) using (DSMC) data from a Sandia National Laboratories report's Chemical Species Database [10]. The moleculeProperties entry for H2O has a set mass ($m$) of 0.2991e-25 (kg) (also obtainable by dividing water's molar mass $\left(M = 18.015 \frac{\text{g}}{\text{mol}}\right)$ by the Avogadro constant $\left(N_A = 6.022140857 \cdot 10^{23} \text{mol}^{-1}\right)$), diameter (reference molecular diameter ($d_{ref}$)) of 0.4387e-9 (m), rotationalDegreesOfFreedom of 0, vibrationalModes of 0, omega (viscosity index ($\omega$)) of 1.0855 (viscosity coefficient fit using default at 273-500 K with (coefficient of viscosity) $\mu \sim T^{\omega}$), alpha (variable soft sphere (VSS) molecular model parameter ($\alpha$)) of 1.0 (for VHS molecular model), and an extremely high ionisationTemperature for preventing ionization, though it is inapplicable, as there are not any defined reactions, charge of 0, numberofElectronicLevels of 1, electronicEnergyList of (0), and degeneracyList of (1) along with the rest of the data for characteristicVibrationalTemperature, charDissQuantumLevel, dissociationTemperature, Zref (relaxation vibrational number at reference temperature), and referenceTempForZref left as () due to their extraneousness. From [10], the calculated effective diameter's reference temperature ($T_{ref}$) is 300 K and can be calculated as [14]:

$$d_{ref} = \left( \frac{5(\alpha+1)(\alpha+2)\left(\frac{mkT_{ref}}{\pi}\right)^{\frac{1}{2}}}{4\alpha(5-2\omega)(7-2\omega)\mu_{ref}} \right)^{\frac{1}{2}} \tag{3.17}$$

Here, $\mu_{ref}$ is a viscosity coefficient with temperature exponent $\omega$, which can be inversely calculated using the Boltzmann constant $\left(k = 1.38064852 \cdot 10^{-23} \text{J} \cdot \text{K}^{-1} = 1.38064852 \cdot 10^{-23} \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1}\right)$

to result in $0.000017913\text{N} \cdot \text{s} \cdot \text{m}^{-2}$.

### 3.2.7. controlDict

The controlDict dictionary in the system directory is fundamental in setting the input parameters for the OpenFOAM solver database, which manage the input (I)/output (O), where output data may be desired during the requisitely set runtime over intervals [3]. In all cases for the conventional and MEMS nozzles, the application set is dsmcFoamPlus. Then, the startFrom starting time chosen is latestTime, which relates to the time directories' newest time step including 0, so that the simulation could be directly rerun to continue from where it stops possibly due to the simulation being killed due to the high-performance computing (HPC) cluster time limit or other viable simulation changes. The dsmcFoam+ nTerminalOutputs, which at its set interval writes out information to the terminal, is standardly set at 1, though it could be increased, as it does take time. The simulation's startTime is 0 (s), which is the value that dsmcInitialise generates an initial time folder for. The simulation stopAt time is endTime, where endTime in this case is simply set as a predicted higher value knowing that the thruster firing transient phase normally happens very quickly, as the simulations are monitored and stopped manually after reaching steady state and time averaging using the discussed method in Running and Managing Simulations (Section 3.3) and fieldPropertiesDict (Subsection 3.2.9).

The time step size (deltaT) specified is 5e-10 (s). The right time step choice is imperative here, as a smaller time step size increases computational time, while a larger time step size could create unrealistic collision routine mass, momentum, and energy transfer and move function particle travel distances along with molecules entering the domain in pulses rather than relatively continuously [52]. To explain, the pressure boundary condition is extremely sensitive to the appropriate time step size (high or low) eventually determined iteratively following the calculations in Table 3.3 from the checkMesh Subsection 3.2.4, as it might otherwise provide unphysical results, which might also be affected by dynamic load balancing as explained in the decomposeParDict, balanceParDict, and loadBalanceDict Subsection 3.2.10. As an interesting mention, the generally less stable inlet pressure boundary conditions have been compared to train locomotion with its engine pushing it rather than pulling, where upstream obstacles have a greater likelihood of causing a crash. The writeControl for controlling the output to file write timing is set to runTime, where data is written per writeInterval specified at 5e-8 (s), so that the data can still be feasibly (considering time and memory) stored with the usage of run-time load balancing, as it basically keeps the information for every time step it is activated at, and using a purgeWrite of 3, which defines the saved time directories' number limit by cyclically overwriting the oldest time directory. To add to the writeInterval and purgeWrite considerations, the commonly desired simulation data are time averages instead of instantaneous particle data [52]. Note that the startTime, endTime, deltaT, and writeInterval are chosen such that deltaT constitutes an integer factor of time steps from startTime to endTime and writeInterval constitutes an integer factor of deltaT time steps and write intervals of time from startTime to endTime. One way to ensure this is by dividing the endTime by deltaT, writeInterval by deltaT, endTime by writeInterval to obtain an integer considering that the startTime is 0 (s), otherwise endTime would be replaced by the time from startTime to endTime. Here, the writeInterval divided by deltaT is 100 time steps per write interval.

As exported in Blender, the default data files' writeFormat is set as ascii, which is written to 7 significant figures as set by writePrecision (write floating point precision) to be more than the mergeTolerance of 1e-6 in snappyHexMeshDict (Subsection 3.2.2). The file gzip compression switch (writeCompression) is set to off. The timeFormat is general (default), where the time directories are named using the scientific format with $\pm m.dddddd\text{e} \pm xx$ for an exponent below -4 or higher than or equal to 6 (default, specified for the timePrecision), which is the number of $d$s, otherwise it is fixed with $\pm m.dddddd$. The runTimeModifiable switch is true, so that dictionaries (especially fieldPropertiesDict discussed in Subsection 3.2.9) are reread at the start of every time step during runtime for possible modifications. Also, the adjustTimeStep switch, which could modify the time step during runtime, is set to no. The dictionary of functions loaded at runtime is left unspecified as ().

### 3.2.8. dsmcInitialiseDict

The dsmcInitialiseDict dictionary in the system directory can set pre-processing initialization data within specified zones to evolve the simulation from by running the dsmcInitialise executable. In the considered conventional and MEMS nozzles' cases, the configurations entry's configuration includes the type set to the dsmcZoneFill initialization algorithm class, which recovers the set density, temperature, and

velocity on average by using properties from particles the zones are filled with (dsmcMeshFill without zoneName could be used for the whole computational domain too), followed by the zoneName defined separately for region1, region2, and region3, referring to the converging, diverging, and plume sections respectively (or a single regionAll for their combined sections) as explained in the createCellZones and createFaceZones Subsection 3.2.3. The simulations start from vacuum, so the numberDensities is set to 0 $\left(\frac{\text{molecules}}{m^3}\right)$ for H2O (from dsmcProperties (Subsection 3.2.6)). The translationalTemperature, rotationalTemperature, and vibrationalTemperature are negligibly set to 1 (K) to exhibit collisions along with the electronicTemperature and velocity set to 0 (K) and (0 0 0) $\left(\frac{\text{m}}{\text{s}}\right)$ respectively.

As explained in the dsmcProperties Subsection 3.2.6, note that at times with mainly constant rarefaction and grid cell sizes, multiplying the average grid cell size (obtainable from Table 3.3 in the checkMesh Subsection 3.2.4) in cubic meters per grid cell by the molecule number density (obtainable from the ideal gas law Equation 3.13 for the simulation domain and set in dsmcInitialiseDict) in molecules per cubic meter and then dividing the resulting molecules per grid cell value by nEquivalentParticles in molecules per particle can obtain the final particle per grid cell value to be 20 or above (random fractions are used to result in consecutively lower or higher integers in case the value is a decimal number) as recommended to allow the recovery of more precise collision statistics for the no-time-counter method, though possibly less are needed for a sufficiently high Knudsen number due to less relevant collisions [52].

Usually, mesh generation and more importantly, particle initialization, are the main problems to more complex DSMC simulations [52]. A potential strategy for creating a larger mesh could involve generating a coarse mesh first typically using OpenFOAM's blockMesh capability to then be possibly refined in parallel with the snappyHexMesh utility. Also, refineMesh, among others, is another option for mesh refinement, though parallel operation might be needed over various nodes for sufficient memory access. For a large number of particles, such as over hundreds of millions, the initialization should be done in parallel using the available processors and the command: mpirun -np #ofProcessors dsmcInitialise -parallel [52]. Prior to populating the domain with particles using dsmcInitialise in parallel, the decomposePar command must be used to decompose the domain.

### 3.2.9. fieldPropertiesDict

The fieldPropertiesDict dictionary located in the system folder holds the dsmcFields for computation. Differences between the conventional and MEMS nozzle inputs are noted, though the dictionaries are identical otherwise. One of the fields set contains the dsmcBinsMethod set as a fieldModel and timeProperties timeOption of write and resetAtOutput initially set to on (explained below) along with the dsmcBinsMethodProperties with fieldName of H2O in reference to the dsmcProperties Subsection 3.2.6, zoneName set to regionAll as described in the createCellZones and createFaceZones Subsection 3.2.3, typeIds of (H2O) (similar to fieldName), averagingAcrossManyRuns set to no, binModel of uniformBins, and uniformBinsProperties with startPoint and endPoint which in this case are set as starting with the point in the middle (or perhaps centroid for a wedge) of the inlet face to the orthogonally projected point at the outlet face. The number of bins (nBins) is set to 300, which is based on an acceptable number of $x$ axis averaged slice planes to obtain sufficient representative flow data along the nozzles. Furthermore, the area is based on the cross sectional areas of the conventional and MEMS nozzles' plume regions, which are the greatest cross sectional areas in the simulation domain. Now, this method is not ultimately used, as the bins are uniform and the actual cross sectional area is not. Therefore, it will result in some values being related to the larger area of the plume region rather than the variable area of the nozzle. Besides working on the source code, one way to solve this is by writing a (MATLAB) code with an if statement for the converging, diverging, or plume region based on the $x$ coordinate. It would work by multiplying the wrongly divided data by the plume region cross sectional area used and then redividing it by a computed cross sectional area based on geometric and trigonometric relations. This is an inefficient quick-and-dirty coding solution, as it is basically inextensible and would only work for the specific problem at hand. Therefore, another flexible method using the Python shell in ParaView is employed and explained in its Section 3.4.

Furthermore, the dsmcVolFields fieldModel fills for another field entry. Its timeProperties include a timeOption of write and resetAtOutput set to on at the start (explained below). dsmcVolFields is a measurement class for returning macroscopic fields of the molecular species as volume scalar fields to be visualized using ParaView, as it also identifies the boundary faces' surface normal vector and

finds pressure and shear stress components from the evaluated force density [52]. The dsmcVolFieldsProperties contain the fieldName H2O and typeIds of (H2O) related to dsmcProperties (Subsection 3.2.6), measureMeanFreePath set to true with a mean free path reference temperature (mfpReferenceTemperature) of 300 (K) in reference to moleculeProperties in dsmcProperties Subsection 3.2.6, and measureErrors set to true. The statistical errors are approximated from relationships in [28], which can be appropriately reduced for a steady flow with a property measurement using a great enough sample size or a transient flow with ensemble average results through repeated simulations providing an acceptable sample [52]. The finite sampling statistical errors in molecular simulation algorithms with thermal fluctuations are predicted [28]. As done in [28] and the source code, The volume-averaged quantities of velocity, density, temperature, and pressure use approximations based on a number of independent samples ($M$) referring to sequential time steps for steady state simulations [28]. The fluid velocity ($u$), density ($\rho$), temperature ($T$), and pressure ($p$) fractional errors ($E$) are calculated using the following equations:

$$E_u = \frac{1}{\sqrt{MN_0}} \frac{1}{Ma\sqrt{\gamma}}, \tag{3.18}$$

$$E_\rho = \frac{1}{\sqrt{MN_0}}, \tag{3.19}$$

$$E_T = \frac{1}{\sqrt{MN_0}} \sqrt{\frac{k}{c_v}}, \tag{3.20}$$

$$E_p = \frac{\sqrt{\gamma}}{\sqrt{MN_0}}, \tag{3.21}$$

where $N_0$ is average statistical cell particle number, $\gamma$ is the specific heat ratio, $k$ is the Boltzmann constant, $Ma$ is the local Mach number, and $c_v$ is the particle heat capacity at constant volume [28]. Although water vapor has strong intermolecular forces that may result in a considerable deviation from an ideal gas, ideal gas conditions are assumed for measureErrors, considering that the fluid's intermolecular forces are expected to become less relevant and the ideal gas relations become more applicable at higher rarefaction (relatively lower pressure and higher temperature). Also, the required number of time steps ($M$) to attain a certain fractional error percentage can be inversely calculated [28]. An increased Mach number would require less independent samples for a fixed estimated fluid velocity fractional error [28]. Moreover, a higher particle heat capacity at constant volume leads to a lower estimated temperature fractional error due to smaller fluctuations [28]. To note, the temperature related different degrees of freedom (translational, rotational, vibrational) can be independently defined and computed [28]. For additional related theoretical derivation, refer to [28]. This work's section is intended to provide an idea about the statistical particle simulation errors.

The next three fields have a fieldModel of dsmcMassFluxSurface. As elaborately explained in the createCellZones and createFaceZones Subsection 3.2.3, the dsmcMassFluxSurface class is a mass flux measurement tool. The time properties have a timeOption of write and resetAtOutput of on at first (explained below). The dsmcMassFluxSurfaceProperties include the H2O fieldName and (H2O) typeIds in reference to dsmcProperties (Subsection 3.2.6). The only variable in the three fields is the faceZoneName, which in three separate fields is set to face1, face2, or face3 referring to the faces at the outlet of the first inlet cells (as the inlet face results in null data), throat, and nozzle outlet respectively. Further information on this is also available in the createCellZones and createFaceZones Subsection 3.2.3. The flow's fluxDirection is set as (1 0 0).

When steady state is reached, time averaging should be activated, considering the run-time modifiable capability by setting resetAtOutput to off, which will halt new write intervals from resetting their averaged accumulated information [52]. It is certainly preferable to average for numerous time steps. Steady state solution is assumed when the overall DSMC particle population and system's average linear kinetic energy are considered relatively constant over time (or the inlet and outlet mass flow rates are relatively equal) [47].

### 3.2.10. decomposeParDict, balanceParDict, and loadBalanceDict

The subsection concerns the implemented parallel computing methodology with its dictionaries in the system directory. With access to two of TU Delft's computer clusters at the Faculty of Mechanical, Maritime, and Materials Engineering (3mE), the simulations are run on the newer Reynolds HPC cluster running a Linux operating system (OS). For the purposes of this work, it consists of many Slurm compute nodes with 28 cores (2x Intel Xeon E5-2680 v4) and 64 GB of memory each along with an interactive (no queue) node with the same number of central processing units (CPUs) and 256 GB of memory and a head node. Slurm Workload Manager is the job scheduler used with a FairShare limited first-come first-served priority algorithm, where this work is completed using one node per job with four jobs running simultaneously and maximum wall clock time of 24 hours before rerunning if needed. Therefore, the conventional and MEMS nozzles with four simulations each are run separately at a time. To note, hyper threading technology is not benefitted from in OpenFOAM, which mainly uses it for multitasking enhancement, so it would be faster to run on physical cores only than combined with virtual cores (additional threads per core) [25].

OpenFOAM uses domain decomposition for parallel computing, where the geometry and associated fields are divided and allocated to be solved by different processors in parallel [3]. The parallel environment Message Passing Interface (MPI) for distributed-memory machines from the public domain openMPI is used as standard for running applications in parallel [3]. Often, CFD parallel computations are based on the number of grid cells and their fields being evenly and maximally distributed (minimum processor workload) with their processor domain boundary numbers and sizes minimized to decrease the need for communication/data sharing among them. As a very rough scaling, each core in a general parallelized CFD program should handle above 50000 grid cells and needs 1000 bytes for every grid cell. Therefore, an increase in computational time (slowdown) would be seen with an excessive number of CPUs for example contrary to a typical prediction of the speedup reaching an asymptote using Amdahl's law. However, DSMC simulations are also especially dependent on the number of particles per grid cell, which would make needing a smaller overall number of particles through fewer grid cells than in general CFD simulations more preferable for decreasing computational time due to DSMC's greater computational load, as explained in the checkMesh Subsection 3.2.4. An iterative solution with appropriate expectations is used to optimally choose the number of distributed processors for running the conventional and MEMS nozzle simulations in parallel. Note that it might seem as though less processors are better at the beginning of the simulation, yet more processors become a superior option as time goes on and more particles enter the domain.

Before starting the simulation in parallel, the mesh and fields are decomposed using the decomposePar (pre-processing) utility, which reads its dictionary, decomposeParDict. The numberOfSubdomains the mesh is decomposed to corresponds to the number of cores the simulation will run on. For each of the conventional nozzle simulations, a total numberOfSubdomains of 14 is considerately chosen, so that only half a node for each simulation and two nodes in total are used at a time, while 28 is chosen for each of the more computationally expensive MEMS nozzle simulations resulting in one node per simulation and a total of four nodes used at a time. The method of domain decomposition chosen is scotch, as it does not need manual geometric input, which makes using its algorithm for minimizing the number of processor boundaries convenient for automatic implementation for many processors, iterations, and less straightforward geometries and fields. Further information about scotch's static (done exclusively before running) mapping (minimizing a parallel program with communicating processes and machine's execution time by combinatorial optimization) through source process and target architecture graphs dual recursive bipartitioning can be found in [44]. Note that since the processors are identical, processor decomposition weighting concerning cell allocation (processorWeights) is not used. After running decomposePar, a subdirectories set each named as processor followed by its respective number is generated in the case directory containing a time directory (in processor0) and constant/polyMesh with the decomposed field and mesh descriptions respectively along with initialization/simulation generated directories [3].

The recommended grid cell and time step sizes were prepared for, though following numerous iterations were determined unfeasible in providing results within a reasonable time on the computer cluster while restricted to a single node per simulation and having multiple simulations to run, so they were changed and dynamic load balancing is additionally used, which is another dsmcFoam+ exclusive feature that proves to be very effective considering the evolving density gradients, as it rebalances each processor's computational load by accounting for the number of particles it handles over the simu-

lation's duration. Its decomposition settings are in balanceParDict, where the numberOfSubdomains and method for both conventional and MEMS nozzles are identical to balanceParDict for consistency and straightforward application. There is an added weightField entry set as dsmcRhoN_H2O, which considers the water DSMC particles number density as a weighting factor to decompose the domain in a better balanced way accounting for DSMC's computational load dependency on the number of particles per grid cell. Also, loadBalanceDict is the dictionary where enableBalancing is set to true for both conventional and MEMS nozzles. The dictionary files for run-time load balancing can be found in the system directory. However, the maximumAllowableImbalance is set to relatively normal value of 0.1 (10%) for the MEMS nozzle, while it is set to 0.001 (0.1%) for the conventional nozzle, as it appeared to help in stability when the decomposition occurs more often for the conventional nozzle simulations. Again, the maximumAllowableImbalance is optimally determined iteratively.

The computational domain mesh is divided into the number of available processor cores that will run the simulation. OpenFOAM's inbuilt parallel computing process includes domain decomposition. As dsmcFoam's move function for particle movements from Particle Tracking in Subsection 2.4.2 causes the greatest computational load, a similar number of particles, $N_{proc}$, is recommended on each core for efficiently balancing the overhead. Ideally, $N_{proc}$ would be:

$$N_{procIdeal} = \frac{N}{n_{proc}}, \tag{3.22}$$

where $N$ is the total DSMC simulation particle number and $n_{proc}$ is the number of activated simulation processors. In dsmcFoam+, every processor's number of particles, $N_{proc}$, is checked with $N_{procIdeal}$ at every write interval within $\pm$ of a customizable tolerance, which exceeding causes the simulation to be paused for automatic load rebalancing by applying domain decomposition again through the reconstruction of mesh and field data generating the usual full domain and fields using reconstructPar -latestTime, which merges the (latest) time directory sets of each processor directory into one time directory set to be followed by the decomposition using decomposeDSMCLoadBalancePar -force (to remove all processor directories before decomposition) before continuing using a bash script for the user to adjust and match the final time directory with the one in [case]/system/controlDict as shown in Section 3.3 [52]. Figure 3.30 demonstrates an example of final particle distributions colored per associated processor at the end of the simulation, as using dynamic load balancing can be seen to detect the flow's stagnation region next to the cylinder, where processor power is focused lowering computational time including recomposition and decomposition time [52].

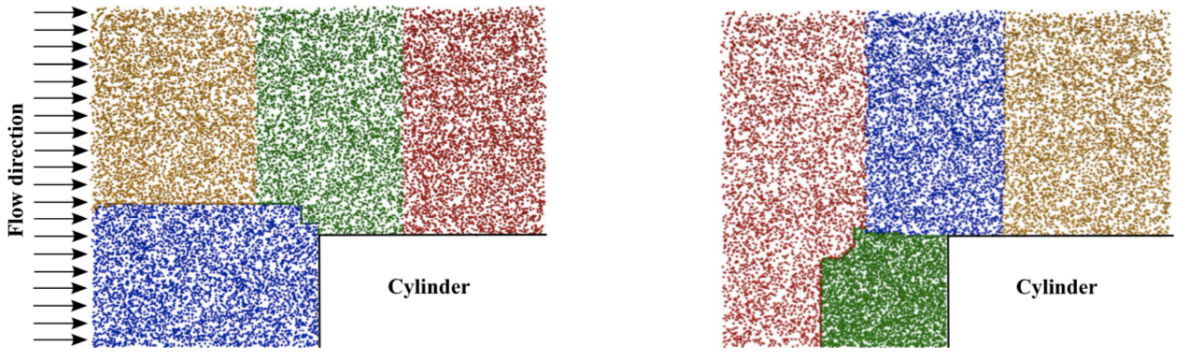



Figure 3.30: Left: Final particle distributions without dynamic load balancing. Right: Final particle distributions with dynamic load balancing. Note that processor numbers are colored. [52]

The maximum imbalance in parallel load is calculated at every writeInterval, which means that once its limit is exceeded, it will remove all but the latest time directory before continuing the simulation. Setting a too high parallel load maximum imbalance value might result in insufficient transient data, as time directories might not be saved often enough. It is ultimately dependent on balancing the simulation settings to obtain the desired data, considering that it would also depend on the writeInterval set in the controlDict Subsection 3.2.7. It is important to note that a handy code is additionally created to save the transient mass flow rate/flux (for particle numbers and their masses) measurements along with the transient solution, as the transient data per time step (fieldMeasurements in the processor0 directory)

was originally being deleted whenever load balancing was activated. This is done in consideration of the accuracy needed for space technology micropropulsion and to transiently study how the propellant flows when the thruster fires. The code is run before the dynamic load balancing script, as shown in Section 3.3. To elaborate, it contains a while true indefinite loop doing a watch command, which detects changes, set to run with -n every 1 second. Note that -d is set to highlight the changes between iterations, -t turns the header off, and -g importantly exits when output from command changes. The watch command is applied for ls, which is used to list information about the directory, with -l for long listing format containing data relevant to the changes applied as calculated and written per time step, such as size and last-modified date, and -R for recursively listing subdirectories (all files). The ls command acts on the generated processor0/fieldMeasurements/ directory containing the desired data that is deleted upon load balancing activation due to decomposeDSMCLoadBalancePar -force. The && logical and boolean operator is used to proceed with executing commands/shell functions when the previous command's exit status is true. Then, when the first command is successful, a cat (concatenate) command is used for each face's mass flow rate and particle number and mass fluxes files referenced starting from processor0/fieldMeasurements/. Using a pipe (|), the cat command's read output data is transferred to the input of the tail command, which typically shows the last 10 lines of a file, where -n 1 is used for tail to output only the last line. Afterwards, the append operator (») is used to append (write at the bottom of the designated existing file avoiding unintended erasure) the new time step data to respective initially empty files in the [case]/fM (fieldMeasurements) directory before finishing the while loop with done and the & operator for running the process in the background. Note that the fM (fieldMeasurements) directory and its initially empty files are created before running the code. In short, every newly written time step data is transferred to another file in a directory that is not deleted due to load balancing.

In case of duplicated time steps, use uniq (File) > (File), where the file's duplicate lines are merged to the first occurrence (uniq) using the redirection operator (>) to make a new file or overwrite the file data if it is existent. Furthermore, cat (File) or * (for concatenating/stringing together all files inside fM) | cut -d 'CTRL+V Tab' -f 1 | uniq -D could be used to check if there still are any duplicated time steps (regardless of the computed data) with cat's read output data piped (|) as the input of cut, which is used for extracting sections from text lines, with the field separator/delimiter -d of Tab, which could be entered after pressing CTRL+V, before piping (|) the value in the first field (-f 1) to the input of uniq, which will print all duplicate lines (-D). Note that uniq filters adjacent lines, which proves to be convenient for this application.

### 3.2.11. fvSchemes, fvSolution, controllersDict, and chemReactDict

Note that standard dsmcFoam+ fvSchemes, fvSolution, controllersDict, and chemReactDict dictionaries should also be in the system directory. They are identical for the conventional and MEMS nozzles. For typical OpenFOAM usage, the fvSchemes dictionary defines the simulation's numerical schemes [3]. In this case for dsmcFoam+, its ddtSchemes (time schemes), gradSchemes (gradient schemes), divSchemes (divergence schemes), laplacianSchemes (Laplacian schemes), interpolationSchemes (interpolation schemes), and snGradSchemes (surface normal gradient schemes) entries are all set to default none along with the fluxRequired entry as default no. Generally in OpenFOAM, the fvSolution dictionary sets the simulation's solution and algorithm control with equation solvers, tolerances, and algorithms [3]. Like fvSchemes, for this case the other dictionaries are basically empty, as fvSolution only contains solvers {}, chemReactDict (chemical reactions dictionary) has reactions (), and controllersDict is made up of dsmcStateControllers () and dsmcFluxControllers ().

### 3.2.12. sampleDict

For post-processing, the sampleDict dictionary is set up in the system directory and called in the case directory using sample (-latestTime for the latest time step), where it writes into a new postProcessing directory containing its time step data sets. For both conventional and MEMS nozzles, the interpolationScheme is set to the mixed linear weighted/cell-face cellPointFace (each polyhedral cell is decomposed into tetrahedra for interpolation using their vertices including the polyhedron cell center inheriting the field value at the cell center along with face vertices with values from cell center interpolations, where a vertex is also coincident with a face center inheriting field values through conventional interpolation schemes from the intersected face cell center values) using cell center, vertex (from neighboring cell center values), and face (from present face interpolation scheme such as linear or gamma) values [4]. The setFormat is csv to output data as CSV (comma-separated values). The surfaceFormat is left as

raw. The (1D line-sampled field domain locations) sets are inlet, converging, throat, diverging, and outlet, though they are separate as axis (set write ordinate) y (coordinate only) and axis (set write ordinate) z (coordinate only) for the MEMS nozzle due to it not being axisymmetric. For the sets, the sampling definition type chosen is uniform as evenly distributed points on a line. The axis (set write ordinate) for the conventional nozzle is set as distance (from the start) with the start and end points respectively being ( -0.447846e-3 increased to -0.447790e-3 0 0 ) and ( -0.447846e-3 increased to -0.447790e-3 0.75e-03 0.0326835e-03 ), ( -0.223925e-3 0 0 ) and ( -0.223925e-3 0.75e-03 0.0326835e-03 ), ( 0 0 0 ) and ( 0 0.75e-03 0.0326835e-03 ), ( 0.16485e-03 0 0 ) and ( 0.16485e-03 0.75e-03 0.0326835e-03 ), and ( 0.32970e-03 decreased to 0.32964e-03 0 0 ) and ( 0.32970e-03 decreased to 0.32964e-03 0.75e-03 0.0326835e-03 ). For the MEMS nozzle, the y axis (middle of $z$-axis) start and end points respectively are ( -0.98750e-03 increased to -0.98749e-03 -1e-03 0.05e-03 ) and ( -0.98750e-03 increased to -0.98749e-03 1e-03 0.05e-03 ), ( -0.49375e-03 -1e-03 0.05e-03 ) and ( -0.49375e-03 1e-03 0.05e-03 ), ( 0 -1e-03 0.05e-03 ) and ( 0 1e-03 0.05e-03 ), ( 0.335585e-03 -1e-03 0.05e-03 ) and ( 0.335585e-03 1e-03 0.05e-03 ), and ( 0.67117e-03 decreased to 0.67116e-03 -1e-03 0.05e-03 ) and ( 0.67117e-03 decreased to 0.67116e-03 1e-03 0.05e-03 ). Note that the MEMS nozzle's z axis sets have 0 for the second entry (middle of $y$-axis) and start point of the third entry, though its end point is set as 0.1e-03. Also, as described during mesh creation (Methodology Chapter 3), the $xy$ plane is the bottom of the nozzle/wedge for both MEMS and conventional nozzles, so the MEMS nozzle $z$ axis values' center is 0.05e-03 between 0 and 0.1e-03. The number of points (nPoints) is set at 20000 (overkill for some sets) for sufficient representation of all considered lengths, considering that the same $y$ start and end points are used for all sets of the respective conventional or MEMS nozzles. The (2D surface-sampled field domain locations) surfaces are left as (), though the sampled fields (scalar) could contain (UMean_H2O overallT_H2O p_H2O rhoN_H2O) to obtain the velocity, temperature, pressure, and particle number density data respectively for the desired set. The velocity (vector) magnitude is calculated afterwards as the square root of the sum of its individual $x$, $y$, and $z$ components squared. Since the conventional nozzle is simulated as a wedge, its data is duplicated, flipped, and prepended with negated distances to resemble its fully rotated conical shape in plots. The inlet (considering the Blender STL ASCII files) and outlet start and end point $x$ values are slightly perturbed to fit within the nozzle, as their respective converging and diverging section lengths are halved to determine the $x$ locations for the converging and diverging section sets as obtained from Table 4.1. The $y$ extrema values of the respective computational domains (MEMS nozzle region only) are used (Table 3.1), as the data is extracted where applicable and the conventional nozzle is simulated as a wedge with single cell thickness. The maximum $z$ value is given for the MEMS nozzle (Table 3.1), though it is calculated as the maximum $z$ (considering the Blender STL ASCII files) value of the computational domain divided by two for the conventional nozzle. The minimum $z$ value is 0 for both conventional and MEMS nozzles.

## 3.3. Running and Managing the Simulations

In continuation from the OpenFOAM/dsmcFoam+ Section 3.2 after preparing the dictionaries and the mesh creation and case initialization are done for pre-processing, a serial run as a single process can be immediately started by calling dsmcFoamPlus (or runApplication 'getApplication' & from a file like Allrun (containing the executables to run) to directly run all commands from the case directory base after loading dsmcFoam+'s OpenFOAM alias setting the paths, where the & operator is used to run the process in the background). However, regarding the significant computational load, the simulation is run in parallel on many processors. To run in parallel without dynamic load balancing, mpirun -np #ofProcessors dsmcFoamPlus -parallel can be used. The domain needs to be decomposed beforehand. The scripts actually used to run the simulations with dynamic load balancing are shown below. Note that using runApplication before the OpenFOAM command automatically writes log.(Application) files in the case directory rather than displaying the output in the terminal, though another way is to use > log.(Application) on the same line after the OpenFOAM command. Also, ./(File) is used for calling from the current working directory. The conventional and MEMS nozzle simulations are run separately at a time each with four simulations. Further information about the commands and topics can be found in their respective subsections of the OpenFOAM/dsmcFoam+ Section 3.2.

Before submitting the Slurm job script as explained in Subsection 3.2.10, the following pre-processing and domain decomposition script is run (the first shebang (#!) interpreter directive line executes the file using the Bourne or compatible shell with /bin/sh as an absolute path/interpreter):

```
1  #!/bin/sh
2  cd ${0%/*} || exit 1    # run from this directory
3
4  # Source tutorial run functions
5  . $WM_PROJECT_DIR/bin/tools/RunFunctions
6
7  runApplication surfaceFeatureExtract
8  runApplication blockMesh
9  runApplication snappyHexMesh -overwrite
10 runApplication createCellZones
11 runApplication createFaceZones
12 runApplication checkMesh
13 checkMesh -allGeometry -allTopology > log.checkMeshAll
14 runApplication dsmcInitialise
15 runApplication decomposePar
```

The following Allrun script is in the current working directory with its first section containing the code for saving the every newly written time step data for each face's mass flow rate and particle number and mass fluxes to another file in a directory that is not deleted due to the second section's dynamic load balancing code as explained in Subsection 3.2.10. Also, notice that np #ofProcessors is not needed for mpirun via Slurm. Lines with options preceded by #SBATCH are Slurm directives and echo displays the strings passed as arguments. In bash, fi is used to end if statements and the script in this case exits at the end with 0 for success. An important note is that mpirun works in this case when called as /usr/bin/X11/mpirun on the Reynolds computer cluster.

```
1  #!/bin/sh
2  cd ${0%/*} || exit 1    # run from this directory
3
4  while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceFlux_face1_H2O_M.xy | tail -n 1
     >> fM/faceFlux_face1_H2O_M.xy; done &
5  while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceFlux_face1_H2O_N.xy | tail -n 1
     >> fM/faceFlux_face1_H2O_N.xy; done &
6  while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceFlux_face2_H2O_M.xy | tail -n 1
     >> fM/faceFlux_face2_H2O_M.xy; done &
7  while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceFlux_face2_H2O_N.xy | tail -n 1
     >> fM/faceFlux_face2_H2O_N.xy; done &
8  while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceFlux_face3_H2O_M.xy | tail -n 1
     >> fM/faceFlux_face3_H2O_M.xy; done &
9  while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceFlux_face3_H2O_N.xy | tail -n 1
     >> fM/faceFlux_face3_H2O_N.xy; done &
10 while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceMassFlowRate_face1_H2O.xy | tail
     -n 1 >> fM/faceMassFlowRate_face1_H2O.xy; done &
11 while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceMassFlowRate_face2_H2O.xy | tail
     -n 1 >> fM/faceMassFlowRate_face2_H2O.xy; done &
12 while true; do watch -n 1 -d -t -g ls -lR processor0/fieldMeasurements/ &&
       cat processor0/fieldMeasurements/faceMassFlowRate_face3_H2O.xy | tail
     -n 1 >> fM/faceMassFlowRate_face3_H2O.xy; done &
```

```
13
14 while :
15 do
16 ### Check that the final time directory does not exist ###
17 if [ ! -d "processor0/1" ]
18 then
19 echo "Directory processor0/1 DOES NOT exist, "\
20 "restart from the latest time."
21 /usr/bin/X11/mpirun dsmcFoamPlus -parallel
22 else
23 echo "Directory processor0/1 DOES exist, "\
24 "killing the script."
25 break ### exit the loop
26 fi
27 done
28 exit 0
```

Then, with a customized Slurm job script (jobscript file) also in the current working directory, it is submitted to call ./Allrun using `sbatch jobscript` (the first shebang (#!) interpreter directive line executes the file using the Bash shell with /bin/sh as an absolute path/interpreter):

```
1  #!/bin/bash
2  #SBATCH -D ./   # "." is the working folder submitted from on the Reynolds
       computer cluster
3  #SBATCH -J my_job   # Job name
4  #SBATCH --mem=32000 or 64000   # Job needs (an expected) 32000 or 64000 MB
       of real memory for the conventional or MEMS nozzle simulations
       respectively (one node has 64 GB of memory)
5  #SBATCH --time=1-0:0:0   # Job runs for 1 day, after which Slurms kills it
6  #SBATCH -n 14 or 28   # Number of cores requested (14 or 28 for the
       conventional or MEMS nozzle simulations respectively) (one node has 28
       cores)
7  #SBATCH -N 1   # Number of nodes requested
8  #SBATCH -o slurm-%N-%j.out   # Output file in working directory (%N is the
       node## and %j is the associated job ID)
9  #SBATCH -e slurm-%N-%j.err   # File with error messages in working
       directory (%N is the node## and %j is the associated job ID #)
10 ./Allrun   # Submitted job in working directory
```

The command-line graphing program, gnuplot, that can produce 2D and 3D plots of functions, data, and data fits is used to recurrently monitor the simulations in realtime through a customized monitor file (also available with dsmcFoam+ tutorials and called using `gnuplot monitor`) in the case directory and generally containing the following contents:

```
1  set ytics nomirror
2  set y2tics nomirror
3
4  set xlabel "Time Step"
5  set ylabel "Number of DSMC Particles"
6  set y2label "Average Linear Kinetic Energy/[J]"
7
8  set grid
9
10 plot \
11 "< cat slurm-%N-%j.out | grep 'Number of DSMC particles' | cut -d '=' -f 2
       "  w l axis x1y1 t "Number of DSMC Particles", \
```

```
12 "< cat slurm-%N-%j.out | grep 'Average linear kinetic energy' | cut -d '='
       -f 2"  w l axis x1y2 t "Average Linear Kinetic Energy"
13 pause 10 # Seconds
14 reread
```

To note, nomirror simply does not mirror the axes tick marks (tics). Also for the plotting, < is used to pipe the datafile through a shell command starting with cat (concatenate) to view the files with its output piped (|) as the input of grep, which finds its succeeding phrase (pattern) with its line output to be piped (|) to the input of cut, which extracts sections from text lines, with the field separator/delimiter -d of = following grep's mentioned pattern (variable name) outputting the value in the second field (-f 2). The plots are with lines (w l) using the same $x$-axis but consecutive $y$-axes (x1y1 for bottom and left axes and x1y2 for bottom and right axes) with a respective key/legend title (t) of the variable.

Note that steady state solution is assumed for turning off resetAtOutput to activate time averaging (stopping new write intervals from resetting their averaged accumulated information) in fieldPropertiesDict (Subsection 3.2.9) when the number of DSMC particles and average linear kinetic energy become relatively constant over time (or the inlet and outlet mass flow rates are comparatively equal) [47]. When rerunning the simulations, additional slurm-%N-%j.out files are generated, so one way to directly monitor them is by listing the newer file names after the older file names for concatenation (stringing together) resulting in a single plot for all files. The further the simulation's time step is from the last write time step based on the writeInterval set in controlDict (Subsection 3.2.7), the more time steps will overlap if the simulation is stopped and the simulation is rerun from the latest write time step. Therefore, only the newer of the overlapping time steps data should ultimately be used to generate the final results. One way to do that (even for the (final) file to remove unwritten time step data) is by using vi (editor) on the file, command (:) using the search backwards (?) from the first line (1) for the last written time step ending with its Maximum imbalance calculation using :1?Maximum and then deleting the following lines using dG before saving and quitting (:wq).

To cleanly stop the process without killing it, stopAt in controlDict (Subsection 3.2.7) can be changed to writeNow, which is upon finishing with the time step being simulated, though since it does not work with dynamic load balancing, simply scancel job ID # can be used to cancel the job along with removing it from the queue (call squeue for the running and waiting jobs list, sinfo for checking the availability for running jobs, and sacct to view information about submitted jobs). Locally however, top can be used to interactively (unlike the snapshots of ps (process status)) monitor the processes and kill them by typing k (compared to kill process identification number (PID) for ps), while q can be used to quit. Furthermore, the default tail command can be used to show the last 10 lines, though -f log.(Application) can in realtime follow the output of the log.(Application) file bottom's changes, while CTRL+C can be used to exit.

As a few additionally helpful notes for guidance, ssh (secure shell) can be used to login into the HPC computer systems and navigate between their nodes. If desired, remember to use -X for X11 forwarding of graphical clients. To transfer and retrieve data between different (local/remote) hosts, scp -r (secure copy protocol recursively) can be used. Also, to empty the swap space into the random-access memory (RAM) if available during testing by restarting, `sudo swapoff -a && sudo swapon -a` could be used with sudo, which allows executing commands as superuser, the && logical and boolean operator for continuing with executing commands/shell functions subsequent to a true exit status for the previous command, and -a for all.

Lossless archiving and compressing could be helpful to transfer or store such large amounts of data. On the side, some desktop computer benchmarking for a sample folder that is around 29.1 GB in size shows that using .zip (ZIP) takes 30 minutes to archive and compress it to about 8.2 GB, while it can be extracted in 14 minutes. Using .tar.gz (tape archive gzip) takes around 34 minutes for a similar size result. Comparably resulting in around 6.8 GB of data, .tar.bz2 (tape archive bzip2) takes around 77 minutes, while .tar.xz (tape archive xz for Linux and macOS) and .7z (tape archive 7-Zip needs installation on Microsoft Windows and macOS) take slightly over 5 hours. Also, note that .tar (tape archive) is used to archive followed by .compressor and the numbers provided are mainly used for a rough comparison. For native compatibility with all operating systems and time convenience, .zip is a competitive option.

Moreover, it might be helpful to check if all the files are transferred. Since the simulation folders contain some symbolic links, use `find -L . | wc -l` to help in searching (find) and returning the files in the working directory (.) including following the symbolic links (-L) with its line output to be piped

(|) to the input of wc (word count), which prints the newline count (-l) returning the working directory's number of items. For file space usage estimation of the working directory (./), `du -shlL ./` can be used, with -s to summarize using a total for each argument, l allowing for hard link size counting, L for dereferencing (following) all symbolic links, and optionally h for printing the size in a simpler (human-readable) format.

## 3.4. ParaView

The open-source application for qualitative and quantitative interactive and scientific visualization and data analysis, ParaView, is mainly used to post-process OpenFOAM's cases. The additionally installed newer OpenFOAM Foundation's OpenFOAM 6 ParaView is used after loading its alias setting the paths, rather than dsmcFoam+'s OpenFOAM 2.4.0. OpenFOAM's helpful post-processing and result visualization tools include paraFoam, which is a wrapper around/script launching ParaView (a visualization environment) to automatically prepare the applicable OpenFOAM data using its reader module. It can even be used to visualize the mesh and initialization (dsmcInitialise in Subsection 3.2.8) data for pre-processing and is recommended to do so [52]. For post-processing a case run in parallel, the domain is divided between the processor folders as explained in the decomposeParDict, balanceParDict, and loadBalanceDict Subsection 3.2.10, so every decomposed domain segment can be post-processed separately and automatically with every processor directory being treated as an individual case using ParaView (.foam) (with Case Type set as Decomposed Case instead of Reconstructed Case under Properties) or the mesh and field data can be reconstructed to generate the usual full domain and fields using reconstructPar, which merges the time directory sets of each processor directory into one time directory set [3].

Using paraFoam creates a (Case).OpenFOAM to run from compared to paraFoam -builtin (for VTK builtin OpenFOAM reader) with a (Case).foam or simply execute touch (File).foam to create an empty file to run paraview (File).foam. The open-source Visualization Toolkit (VTK) is used for manipulating and visualizing data. Typically, the files created in the case directory using paraFoam are temporary. For checking the mesh, paraFoam is used, as it allows easier access to Include Sets and Include Zones under Properties to visualize sets related to checkMesh (Subsection 3.2.4) and createCellZones and createFaceZones (Subsection 3.2.3) along with zones. To note, .foam is used to load lagrangian/dsmc from Mesh Regions in Properties, though in some cases it might be desirable to exclude it due to its demanding computational requirements, so the generally faster .OpenFOAM could be used instead, such as when rotating the conventional nozzle wedge mesh or extensively post-processing the MEMS nozzle (end) results. Note that .foam and .OpenFOAM without changing their settings at launch might have visual differences, especially regarding waterproofing, though they could often be ignorable within reason.

As a reference to some of the ParaView Figures shown in different parts of this report, the White Background palette from Color Palette under Settings is loaded. Camera Parallel Projection and Axes Grid with Include Sets and Include Zones under Properties are used. Show Plane is eventually switched off, when applicable. Solid Color or vtkBlockColors (for showing different Mesh Parts) along with Surface or Surface With Edges (Mesh) could be used. The Adjust Camera can be utilized to achieve the desired viewpoint. Zoom to Data could also be used to focus on the object before further magnification. Opacity could be lowered to make the object translucent between 1 for solid and 0 for invisible. Show Patch Names is also a possible option. Notice that there is a gearwheel to toggle advanced properties, which could be helpful for some settings. To discuss some applicable filters:

- The Extract Cells By Region Filter uses its input dataset to extract cells fully (or optionally not fully) inside or outside the set region (implicit function) for an unstructured grid [5].
- The Angular Periodic Filter is used to rotate the conventional nozzle's 5° wedge mesh (similar to a 3D pie slice) around an axis of rotation, as it generates a periodic multiblock dataset [5].
- The Slice Filter applies on input dataset with any type to result in an extracted specified plane with polygonal data output, similar to a contour, where surfaces and lines are made from volumes and surfaces respectively [5]. The Crinkle slice (PreserveInputCells) is set for extracting the complete (sliced) cells or a triangulated surface of the region, while Triangulate slice can be set to produce triangles in the output [5].

- The Integrate Variables Filter integrates cell and point attributes along with calculating the line length, surface area, or volume [5]. The Divide Cell Data By Volume is a parameter to control whether the cell data output is divided by the integrated cells' calculated surface area/volume.

- The Clip Filter works by cutting part of the input data (sets of all types) with an implicit function/description for unstructured grid data [5]. Crinkle clip is used for either extracting the complete (clipped) cells or clip them to remain on one side of the clipping plane [5]. Invert selects the part to be clipped.

- The Contour Filter calculates isolines/isosurfaces from any input data set type using a point-centered scalar (single-component) array with polygonal data output [5]. To note, Generate Triangles can be set to produce triangles in the output compared to non-triangular polygons to have better compatibility with some filters [5]. For 3D case modules, the constant value is represented by a 2D surface set, though it can be used in combination with the Slice Filter to create the cutting plane first resulting in contour plot lines across a plane if desired [3].

- The Calculator Filter uses existing (scalar or vector) arrays to calculate extra attribute (data or point) arrays as their function [5]. Point-centered and cell-centered arrays remain in the same format, though point coordinate functions result in three-component vector functions [5]. The Calculator is like a scientific calculator and works on any input data set type with a scalar or vector array resulting in a similar output data set type [5]. To note, the scalar menu shows the scalar array names along with the vector array component names of data that is either point-centered or cell-centered, while the vector menu shows the point-centered or cell-centered vector array names [5].

- For data (set of any type) attribute sampling on a line of points, the Plot Over Line Filter can be used to result in graphs, where the line's point-centered variable values are shown in an XY Plot, as it uses interpolation to output polygonal data (line) [5].

- The Stream Tracer Filter could be used to generate streamlines as it integrates vector field streamlines using tracer lines with seed points, where the streamline stops upon crossing the input data set's exterior boundary or due to its set parameters (some might need to be determined iteratively) or initialization/computational issues. It works for any data set type with point-centered vectors producing polygonal data output with polylines. Compute Vorticity is also a possible option.

The filters could be combined to achieve desired results with selectable graphics enabling/disabling of modules in the Pipeline Browser. For some filters, a range of values can be set (with steps) under Properties. To note, as the data range might not automatically update to the maximum and minimum field limits, the color map can be rescaled over time by setting the Transfer Function Reset Mode under Color/Opacity Map Range Options in General (Settings) (or simply Automatic Rescale Range Mode in Color Map Editor) to update at every time step, though it is generally undesirable/misleading. A specified or automatically determined (even for all time steps) color range can be set using the Rescale options under Coloring in Properties (or Color Map Editor). The color map can also be changed or modified using Edit Color Map. Often, standard colors are changeable.

Images can be output to file using Save Screenshot. The images' pixel resolutions (with a possibly locked aspect ratio) could be raised for better visualization. The same can be done for animation output using Save Animation. When creating animations, the number of frames per time step could be set. Although choosing 1 frame per time step initially sounds most intuitive, it might be desirable to artificially add frames obtaining a slower and more controllable video while playing [3]. The named file root "image" and its associated image number along with the chosen file format extension are saved in a series before the image set can be converted into a movie. To note, the executable foamCreateVideo can use a PNG (Portable Network Graphics) image sequence to create an MPEG-4 compressed (.mp4) video file, where options exist and the video codec is high resolution with 10 frames per second (adjustable (typically lowered) using -f #fps for frame rate) by default. Also, avconv or MEncoder for converting (transcoding) different (video/audio) formats need to be installed (for OpenFOAM 6). Furthermore, Save and Load State could be used to return back to a ParaView state file. Note that other ParaView options are used as standard in general.

### 3.4.1. Python Shell

As the saying goes, if you are going to do something more than ten times, then you might as well code it. Due to the possible inflexibility of using dsmcBinsMethod as explained in the fieldPropertiesDict Subsection 3.2.9, the Python Shell is used to provide averaged data instead of centerline data using the Plot Over Line Filter. The Python Shell in ParaView is used to achieve incremental slicing (Slice Filter) along the X Normal (changing X Origin) and integrating the data (Integrate Variables Filter) using the Divide Cell Data By Volume option before the Cell Data (Attribute) is exported as a CSV (comma-separated values) file into a pre-made empty (Python) folder inside the working directory for each filter combination separately and consecutively as shown in the Python script below. Note that the Start Trace tool in ParaView is used at startup to record the application's work in Python and serve as a guideline for the Python script. Notice that integers are used for the range, though the values are not divided by an integer (by adding .0) to result in the desired floating point number values and not expect an integer result. The range numbers start and stop respectively with the minimum and maximum $x$-axis extremities (obtainable from the Blender STL ASCII files, though they are increased (or decreased respectively) when $x$ is undesirably at the boundaries without data) using a step to obtain sufficient representative flow data along the domains, especially the converging and diverging nozzle sections. Also, note that interpolation errors are introduced.

Each CSV file contains a header row before the data row and it is desirable to concatenate them. So, `find ./ | sort -V | xargs cat > all.csv ; sed -i '3~2d' all.csv` is used for searching (find) and returning the files in the working (Python) directory (./) with its line output to be piped (|) as the input of sort to be numerically sorted by their filenames' text with a natural sort of (version/-V) numbers and its line output to be piped (|) to the input of xargs (converting standard input to arguments) for cat (concatenation/stringing together) before using the redirection operator (>) to output the data in a new CSV file. This results in the CSV files being sorted based on their (X Normal/Origin) values and concatenated, though the identical file headers would still be repeated. Therefore, ; is used to run another following command (sed, which parses and transforms text) using in place file editing (i) to delete every other line starting from the third line ('3~2d') of the concatenated file to retain the first header along with the data values.

It is mainly used for the chosen final time step (Last Frame) with Apply under Properties. Note that for the conventional nozzle wedge, the extensive properties could be multiplied by 360° (full rotation) / 5° (wedge) as applicable to result in values for the whole conventional nozzle. A column representing the range for X Origin can be manually input into the CSV files by sequentially adding the step to the starting X per iteration. Other parameters could be studied as well. As the area is given in the data, the representative physical length scale can be determined from depth (given as 0.1 mm for nozzle) multiplied by the desired width (typical representative physical length scale as hydraulic diameter equaling four times the cross sectional area divided by the cross sectional wetted perimeter, considering that it reduces to an equivalent diameter for a circular cross section) value equaling the area for the MEMS nozzle, while the conventional nozzle's wedge area could be multiplied by 360° (full rotation) / 5° (wedge) to equal the rotated area ($\pi r^2$, where $r$ is the radius and $2r$ is the diameter ($D$)), from which the radius can be obtained to find the desired diameter (representative physical length scale) value.

The following Python script, which could be adjusted for other uses as well, is run using the Python Shell in ParaView with the respective (possibly increased $x$) range for the conventional and MEMS nozzles:

```
1  #### import the simple module from the paraview
2  from paraview.simple import *
3
4  # get active source.
5  foamfoam = GetActiveSource()
6
7  i = 1
8
9  for x in range(-447846 increased to -447840,2662330,8000) or range(-987500
       increased to -987490,5647170,17000):
10
11         # create a new 'Slice'
```

```
12          slice1 = Slice(Input=foamfoam)
13          slice1.SliceType = 'Plane'
14          slice1.SliceOffsetValues = [0.0]
15
16          # init the 'Plane' selected for 'SliceType'
17          # Properties modified on slice1.SliceType
18          slice1.SliceType.Origin = [x/1000000000.0, 0.0, 0.0]
19          slice1.UpdatePipeline()
20
21          # create a new 'Integrate Variables'
22          integrateVariables1 = IntegrateVariables(Input=slice1)
23
24          # Properties modified on integrateVariables1
25          integrateVariables1.DivideCellDataByVolume = 1
26          integrateVariables1.UpdatePipeline()
27
28          writer = CreateWriter("Python/" + str(i) + ".csv",
                integrateVariables1)
29          writer.FieldAssociation = "Cells"
30          writer.UpdatePipeline()
31          del writer
32
33          # Increase i by 1
34          i += 1
```

# 4

# Results and Discussion

In this chapter, the results of the DSMC simulations along with an analytical model and its methodology for the same nozzle cases are presented and discussed. In addition to comparing the conventional and MEMS thrusters using the DSMC simulation data, DSMC/continuum model comparisons are made generally using the same nozzle cases in the developed analytical model and an additional VLM CFD model prepared in advance at TU Delft. A comprehensive collection of literature papers with DSMC/continuum numerical and experimental results on relevant conventional and MEMS de Laval micronozzles has been compiled for potential help.

## 4.1. MEMS vs. Conventional Thrusters (DSMC)

Each of the conventional and MEMS nozzles is simulated for all four inlet pressure and temperature combinations with 5 bar and 550 K, 5 bar and 773 K, 7 bar and 550 K, and 7 bar and 773 K. For possible referencing abbreviations, the cases are ordered by number from 1 to 4 respectively and written after the first letter of their respective nozzle. The simulations are run following the Running and Managing the Simulations Section 3.3, where the conventional nozzle simulations took 24 hours (1 day) to complete with assumed steady state time averaging activated (resetAtOutput off in fieldPropertiesDict) after 21 hours, while the MEMS nozzle simulations took 96 hours (4 days) to complete with assumed steady state time averaging activated (resetAtOutput off in fieldPropertiesDict) after 84 hours (3.5 days) to result in around 181 GB and 185 GB of data for the conventional and MEMS nozzles respectively (C1: 54 GB, C2: 60 GB, C3: 29 GB, C4: 38 GB, M1: 34 GB, M2: 36 GB, M3: 58 GB, M4: 57 GB, Total: 366 GB). Note that more data is saved for the conventional nozzle simulations mainly due to the lower dynamic load balancing maximumAllowableImbalance (decomposeParDict, balanceParDict, and loadBalanceDict Section 3.2.10).

When rerunning a simulation (as explained in Running and Managing the Simulations Section 3.3 and decomposeParDict, balanceParDict, and loadBalanceDict Section 3.2.10) it is usually directly rerun using the Slurm job script, though in some cases the simulation might end on the write time step, so it would be safer to first delete the time directory/directories using rm -rf processor*/(TimeStep), which removes (rm) the potentially incomplete time step directories possibly from all (*) processor directories recursively (-r) to remove the contents by force (f), where nonexistent files/arguments are neglected, as confirmation prompts are overridden. It might be helpful to note that decomposeDSMCLoadBalancePar -force > log.(Application) could be used to decompose the latest reconstructed time directory before rerunning too, when it is desirable to remove all processor directories before decomposition or the latest time directory has been reconstructed. Also, purgeWrite in controlDict (Subsection 3.2.7) does not delete the old decomposed time directories when rerunning the simulation. Another issue related to rerunning with the implicit boundary condition is missing the addition of particles for the first time step, while particles can still leave the domain. This effect is amplified with dynamic load balancing due to the frequent rerunning after the recomposition and decomposition. Therefore, it would be useful to have a time step size that is reasonably smaller than the greater writeInterval, as set in controlDict (Subsection 3.2.7) (also due to the lesser significance of a smaller time step size considering the fewer missed entering particles), and use a higher dynamic load balancing maximumAllowableImbalance

(decomposeParDict, balanceParDict, and loadBalanceDict Section 3.2.10).

To note, the MEMS nozzle simulations were backed up after the third 24 hour run using reconstructPar -latestTime > log.(Application). Also, reconstructPar -newTimes > log.(Application) was run to reconstruct the time steps that have not been reconstructed (-newTimes) at the end of all simulations, as applicable.

The scatter plots in many of the subsections below originate from methodology in Chapter 3. The variable names from ParaView are left as is on the $y$-axis for clarity. A logarithmic scale is used when it is necessary to show largely varying data. The Python script is used for approximate data at all locations along the simulation domains, such as at the nozzle throat and outlet, as the resolution is sufficient to extract data from the cells themselves or ones very close to (right before) them.

Notice that in terms of both converging and diverging section lengths, lateral surface areas, and volumes, the MEMS nozzle is longer/larger than the conventional nozzle with a greater total lateral surface area to total volume ratio resulting in possibly greater losses and enhanced heat transfer. Most of the data presented is generally for the more important nozzle section, though the plume region is also included and noted in some cases, as it is also noticeable from the $x$-axis dimension. Both conventional and MEMS nozzle throats are at $x$ of 0 m. The following geometric and trigonometric relations can be used to determine the length of the converging and diverging sections in the nozzles, where for the conventional nozzle, the converging section length is ((((Inlet Diameter of 0.0003 m) - (Throat Diameter of 0.00006 m)) / 2) / tan(Converging Half Angle of 15°)) and diverging section length is ((((Outlet Diameter of 0.0003 m) - (Throat Diameter of 0.00006 m)) / 2) / tan(Diverging Half Angle of 20°)), while for the MEMS nozzle, the converging section length is ((((Inlet Diameter of 0.002 m) - (Throat Diameter of 0.000025 m)) / 2) / tan(Converging Half Angle of 45°)) and diverging section length is ((((Outlet Diameter of 0.0008 m) - (Throat Diameter of 0.000025 m)) / 2) / tan(Diverging Half Angle of 30°)). Table 4.1 shows the lengths, surface areas, and volumes of the conventional and MEMS nozzles' converging and diverging sections. The lateral surface areas of the converging and diverging sections are calculated using the conical frustum lateral surface area formula (Equation 4.2) for the conical 3D conventional nozzle, while the quasi-2D MEMS nozzle's respective trapezoid area sections are calculated using Equation 4.3 before being multiplied by 2 (accounting for both sides of each of the converging and diverging sections) and adding both sides of the converging or diverging lateral areas obtained as the given height of 0.1 mm multiplied by the slant line length obtained from the Pythagorean theorem as the square root of ((((Inlet (2 mm) or Outlet (0.8 mm) Diameter) - (Throat (0.025 mm) Diameter))/ 2)$^2$ + (Respective Calculated Converging or Diverging Length)$^2$). To calculate the volume of the conical 3D conventional nozzle's converging and diverging frustum sections, Equation 4.1 is used, while the quasi-2D MEMS nozzle's respective trapezoid area sections are found using Equation 4.3 before being multiplied by their extrusion of 0.1 mm to result in the volumes. Note that $V$ is the volume, $L$ is the respectively calculated length, $r$ is the radius at the inlet, throat (repeated for both converging and diverging sections), or outlet, $A$ is the area, and $D$ is the diameter at the inlet, throat (repeated for both converging and diverging sections), or outlet.

$$V = \frac{\pi}{3} L(r_1^2 + r_2^2 + r_1 r_2) \tag{4.1}$$

$$A = \pi(r_1 + r_2)\sqrt{(r_1 - r_2)^2 + L^2} \tag{4.2}$$

$$A_{\text{trapezoid}} = \frac{D_1 + D_2}{2} L \tag{4.3}$$

Also, it might be necessary to readily extract thermodynamic properties using a tool such as the MATLAB wrapper for CoolProp, which is a C++ library for thermodynamic data, as in obtaining the dynamic viscosity for $Re$ with the temperature and pressure data preloaded as respective columns in tables into MATLAB. The following MATLAB code could be used, while noting that try and catch are used for executing the statements and catching errors overriding the default error handling to continue after the thermodynamic data becomes unavailable for one variable due to the temperature(/pressure) drop:

```
1 try
2     for a=1:length((Table).(Column))
```

Table 4.1: Conventional and MEMS nozzles' converging and diverging section lengths, lateral surface areas, and volumes

| | Conventional Nozzle | MEMS Nozzle |
|---|---|---|
| **Converging Section/Frustum Length (mm)** | 0.44785 | 0.98750 |
| **Diverging Section/Frustum Length (mm)** | 0.32970 | 0.67117 |
| **Total Length (mm)** | 0.77755 | 1.65867 |
| **Converging Section/Frustum Lateral Surface Area ($mm^2$)** | 0.26219 | 2.27899 |
| **Diverging Section/Frustum Lateral Surface Area ($mm^2$)** | 0.19841 | 0.70872 |
| **Total Lateral Surface Area ($mm^2$)** | 0.4606 | 2.98771 |
| **Converging Section/Frustum Volume ($mm^3$)** | 0.013085 | 0.099984 |
| **Diverging Section/Frustum Volume ($mm^3$)** | 0.0096328 | 0.027686 |
| **Total Volume ($mm^3$)** | 0.022718 | 0.12767 |
| **Total Lateral Surface Area ($mm^2$) to Total Volume ($mm^3$) Ratio** | 20.27467 | 23.40182 |

```
3            (Variable)(a) = CoolProp.PropsSI('V','T', (Table).overallT_H2O(a),
                 'P', (Table).p_H2O(a), 'Water'); %Pa.s - Dynamic Viscosity
4        end
5    catch
6    end
7    (Variable) = transpose((Variable));
```

### 4.1.1. Steady State Convergence and General Final Simulation Data

Considering the simulation times including steady state time averaging as explained above and in Chapter 3 and using gnuplot to monitor the simulations as shown in the Running and Managing the Simulations Section 3.3, the final steady state convergence plots for all conventional and MEMS nozzle simulations with fixed none overlapping data due to rerunning are shown in Figures 4.1 to 4.8. They show the number of DSMC particles and average linear kinetic energy vs. time step.

Some simulations appear to have been run longer while converged than others due to the conventional and MEMS nozzle simulations being run separately with each of their four simulations having the same runtime until all running simulations converge (first) and time average sufficiently before concluding the simulations. As explained earlier in the fieldPropertiesDict Subsection 3.2.9 and Running and Managing the Simulations Section 3.3, the convergence line (horizontal asymptote) is reached when the number of DSMC particles and average linear kinetic energy become relatively constant over time (or the mass flow rates at the inlet and outlet are comparatively equal) [47].

Table 4.2 summarizes the general final simulation data obtained, especially from the simulation output file. To note, the total time steps is calculated by simply dividing the final time step by the time step size of 5e-10 s (for all simulations) set in controlDict (Subsection 3.2.7). Also, the average energy is the respective energy divided by the number of molecules (final number of DSMC/free particles multiplied by nEquivalentParticles (from dsmcProperties Subsection 3.2.6) for the number of real gas atoms/molecules represented by each DSMC simulation particle), while total energy is the sum of the applicable energies. The assumed (roughly) estimated steady state convergence time step data are respectively obtained visually from Figures 4.1 to 4.8. The equivalent time for assumed estimated steady state convergence time step is obtained by multiplying the time step size of 5e-10 s set in controlDict (Subsection 3.2.7) by the assumed estimated steady state convergence time step.

It is interesting to note that the simulations with lower final number of DSMC/free particles (lower pressures with constant temperature and higher temperatures with constant pressure) appear to run faster. The final time step inserted parcels is determined from the inlet pressure boundary condition, which is dependent on many factors as shown in the boundariesDict Subsection 3.2.5 along with nEquivalentParticles in the dsmcProperties Subsection 3.2.6. The final collisions relatively scale with the final number of DSMC/free particles in the computational domain. For the conventional nozzle, the final average linear kinetic energy increases with increasing temperature at constant pressure and decreases with increasing pressure at constant temperature, though the MEMS nozzle's final average linear kinetic energy slightly increases with increasing pressure at constant temperature, yet also increases with increasing temperature at constant pressure. Furthermore, the conventional nozzle's final total energy increases with increasing temperature at constant pressure and increases with increasing pressure at constant temperature, though the MEMS nozzle's final total energy decreases with

increasing temperature at constant pressure, yet also increases with increasing pressure at constant temperature, which starting from the final average linear kinetic energy is due to the varying number of molecules (final number of DSMC/free particles multiplied by nEquivalentParticles (from dsmcProperties Subsection 3.2.6) for the number of real gas atoms/molecules represented by each DSMC simulation particle) as mentioned in its definition. Much of the mentioned trends can be extracted from the ideal gas law relations, with $p$ as pressure, $V$ as volume (might be taken as 1 $m^3$ for comparison), $n$ is the number of moles in the gas, $R_A$ the ideal gas constant, $T$ the temperature, $N$ the number of molecules, and $k$ the Boltzmann constant:

$$pV = nR_AT = NkT \tag{4.4}$$

The conventional and MEMS nozzles seem to reach steady state at around the same physical time with higher temperatures at constant pressures and lower pressures at constant temperatures taking longer due to the number of molecules being directly proportional to the pressure and inversely proportional to temperature. The equivalent time for the assumed estimated steady state convergence time step is determined to be generally in the order of some microseconds. However, the typical response time of these thrusters is in milliseconds, so the fact that steady state is achieved within microseconds makes the transient solution less significant.

For a full conventional nozzle as compared to the 5° wedge, the extensive values could be multiplied by 360° (full rotation) / 5° (wedge), though it would not be ideal to compare the conventional and MEMS nozzles from this aspect here due to the different plume regions and their lesser importance.

Table 4.2: General final simulation data

| | Total Time Steps | Final Time Step (s) | Final Time Step Inserted Parcels | Final Collisions | Final Number of DSMC/Free Particles | Final Average Linear Kinetic Energy/[J] | Final Total Energy/[J] | Assumed Estimated Steady State Convergence Time Step | Equivalent Time for Assumed Estimated Steady State Convergence Time Step (s) |
|---|---|---|---|---|---|---|---|---|---|
| **C1** | 49400 | 2.47e-05 | 225 | 7398479 | 780145 | 9.096855e-21 | 2.12906e-10 | 15000 | 7.5e-06 |
| **M1** | 43200 | 2.16e-05 | 1360 | 163606403 | 11465895 | 6.52231e-21 | 7.478412e-05 | 15000 | 7.5e-06 |
| **C2** | 67800 | 3.39e-05 | 199 | 4838243 | 623769 | 1.147856e-20 | 2.14799e-10 | 25000 | 1.25e-05 |
| **M2** | 45900 | 2.295e-05 | 1151 | 155624401 | 11133070 | 6.682069e-21 | 7.439194e-05 | 20000 | 1e-05 |
| **C3** | 17800 | 8.9e-06 | 322 | 16587249 | 1170928 | 8.417093e-21 | 2.956743e-10 | 12000 | 6e-06 |
| **M3** | 22900 | 1.145e-05 | 1902 | 319410940 | 16020465 | 6.538048e-21 | 0.0001047426 | 14000 | 7e-06 |
| **C4** | 30200 | 1.51e-05 | 276 | 9789462 | 888932 | 1.117359e-20 | 2.979768e-10 | 15000 | 7.5e-06 |
| **M4** | 26300 | 1.315e-05 | 1633 | 303906828 | 15557852 | 6.696452e-21 | 0.0001041824 | 15000 | 7.5e-06 |

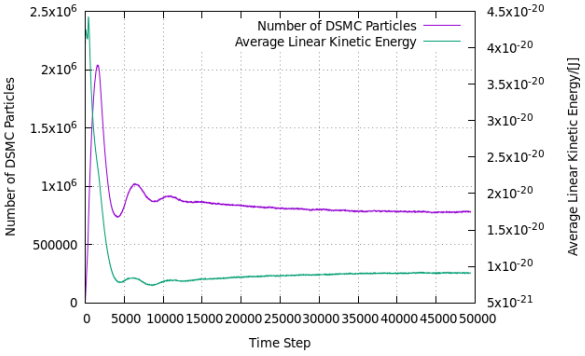



Figure 4.1: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for C1

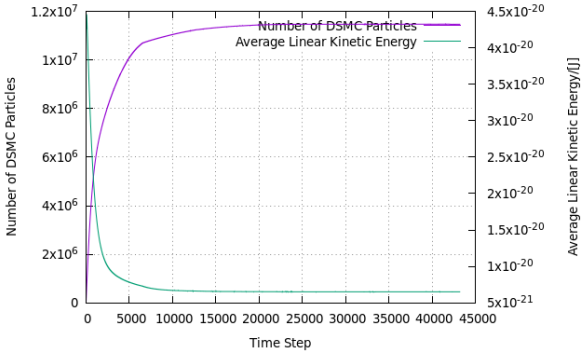



Figure 4.2: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for M1

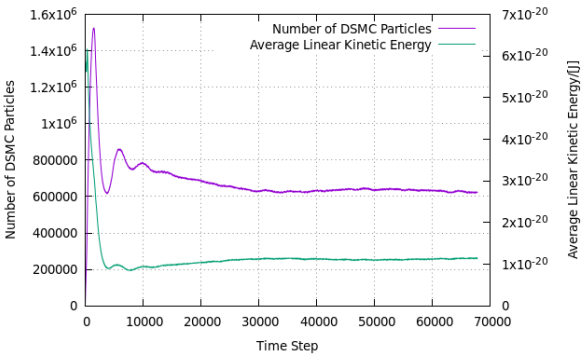



Figure 4.3: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for C2

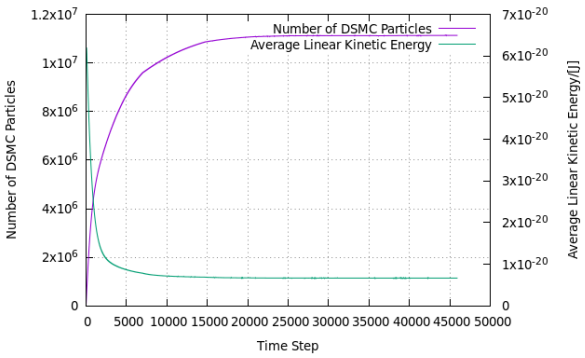



Figure 4.4: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for M2

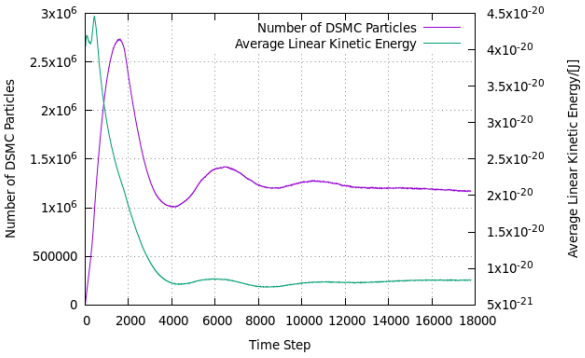



Figure 4.5: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for C3

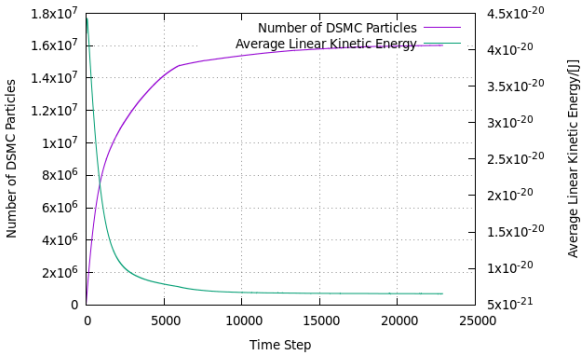



Figure 4.6: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for M3

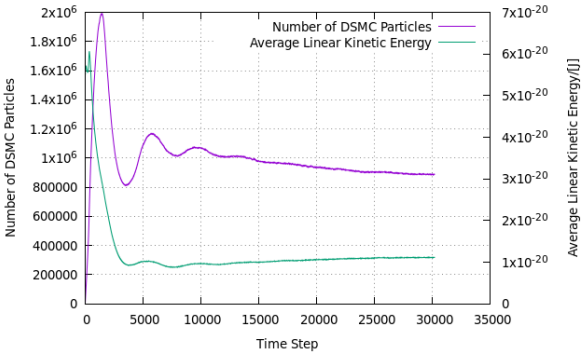



Figure 4.7: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for C4

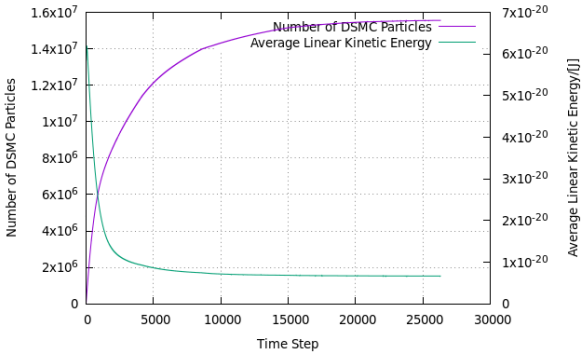



Figure 4.8: Number of DSMC particles and average linear kinetic energy vs. time step steady state convergence plot for M4

### 4.1.2. Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio

Referencing Chapter 3 and the checkMesh Subsection 3.2.4, the reason for the difficulty of maintaining 20 or more particles per grid cell as recommended for the no-time-counter method to allow the recovery of more precise collision statistics with possibly less particles per grid cell needed when the Knudsen number is sufficiently high and collisions become less relevant can be seen in Figure 4.9, where the nozzle mean DSMC particle number density per grid cell is plotted vs. $x$ for all simulations [52]. The drop from the significantly high number of particles per grid cell before the throat to a much lower and relatively acceptable number of particles per grid cell after the throat is clear and challenging to feasibly achieve for these DSMC simulations. To add, the comparably plotted nozzle mass density (Figure 4.10) and particle number density (Figure 4.11) also follow similar expected trends with generally higher densities (particles) for higher pressures at constant temperature and lower temperatures at constant pressures and an initially relative increase before the nosedive at the throat along with a relative decrease before and after the throat. The MEMS nozzle's larger dip past the throat is relatively amplified due to the slightly extended throat section's smaller grid cell size, as explained in the Methodology Chapter 3's mesh creation. Although the conventional nozzle appears to follow the expected trends throughout, the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a relative sharp increase at the inlet, as it can be seen to make the temperature difference at constant pressure very small towards the throat resulting in very similar flows past the throat along with its following increasing to decreasing behavior due to the temperature arch as explained in the Pressure and Temperature Subsection 4.1.4 along with the geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) of the MEMS nozzle (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13) leading to its flow beyond the throat becoming relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). The mentioned densities are generally relatively higher for the MEMS nozzle simulations compared to the conventional nozzle simulations. To note, the particle number density can be found as the mass density multiplied by the Avogadro constant $\left(N_A = 6.022140857 \cdot 10^{23}\text{mol}^{-1}\right)$ divided by the molar mass for water $\left(M = 18.015 \frac{\text{g}}{\text{mol}}\right)$.

Even though none of the nozzle cases exceed the slip flow regime (spoiler from the Knudsen Number Subsection 4.1.10), in a 2D micro channel flow DSMC accuracy study, the slip flow regime is determined to need at least 20 particles per cell for a comparably acceptable accuracy with sufficient intermolecular collisions compared to 10 particles per cell for the transition flow regime due to its molecular motion mechanism with a smaller intermolecular collision rate [48]. However, the real to simulated molecules' scaling factor (nEquivalentParticles) as considerately set in dsmcProperties (Subsection 3.2.6) could be acceptably higher for the slip flow regime $(10^{10})$ compared to the transition flow regime $(10^{9})$ with a more significant individual molecular motion [48].

For the conventional and MEMS nozzles' geometric analysis, the aspect ratio (width/height) vs. $x$ is shown in Figure 4.12 along with the perimeter to cross sectional area ratio vs. $x$ in Figure 4.13. Note that the conventional nozzle's area is multiplied by 360° (full rotation) / 5° (wedge) to obtain the full conventional nozzle values. The aspect ratio is constant at a value of 1 throughout the conventional nozzle due to its conical shape, though the MEMS nozzle's aspect ratio drops sharply from around 20 at the inlet to approximately 0.25 at the throat before increasing back to around 8, which is likely disruptive, as the rectangular horizontal flow becomes a rectangular vertical flow before returning to a previously higher aspect ratio. Furthermore, considering the quasi-2D shape of the MEMS nozzle with the larger total lateral surface area to total volume ratio (Table 4.1) in this case compared to the conical 3D shape of the conventional nozzle, the perimeter to cross sectional area ratio, which peaks at the throat for both conventional and MEMS nozzles, is considerably higher for the MEMS nozzle in general, which leads to effects including higher friction losses and enhanced heat transfer. Note that the widths of the conventional and MEMS nozzle are determined from their mentioned methods in the

Python Script Subsection 3.4.1, while the height for the MEMS nozzle is given. The MEMS nozzle's perimeter is found as the sum of the twice width and height, while the conventional nozzle's perimeter is $2\pi r$. Also, the fully rotated cross sectional area of the conventional nozzle is used.

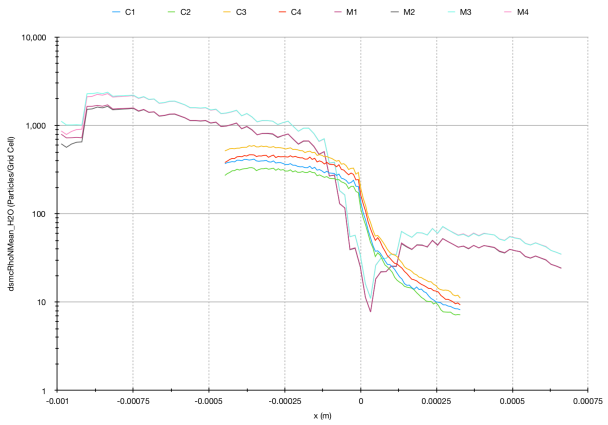



Figure 4.9: Nozzle mean DSMC particle number density per grid cell vs. $x$ for all simulations

### 4.1.3. Mean Collision Time, Mean Collision Rate, Mean Collision Separation, (Local) Variable Hard Sphere Mean Free Path, Separation of Free Paths, and Courant-Friedrichs-Lewy Number

The nozzle mean collision time vs. $x$ for all simulations is shown in Figure 4.14 and the ratio found by dividing the mean collision time by the time step size (5e-10 s from controlDict Subsection 3.2.7) is shown in Figure 4.15. Higher temperatures at constant pressures and lower pressures at constant temperatures lead to a higher mean collision time, with the plots showing an initially relative decrease before the rise at the throat along with a relative increase before and after the throat. Notably, as mentioned in Chapter 3 (checkMesh Subsection 3.2.4 and controlDict Subsection 3.2.7) and OpenFOAM dsmcFoam(+) Solver Subsection 2.4.2, the time step size should be a fraction of the mean collision time, which is the successive particle collision time on average, for more accurate movement and collision steps decoupling, as DSMC particles must not be allowed to skip grid cells at the most probable molecular speed for sufficient interaction with other particles. This generally becomes more difficult to apply before the throat considering the sharp increase in mean collision time following it. The values scale with the calculations made and explained in the checkMesh Subsection 3.2.4. In consideration of simulation feasibility as explained in Chapter 3, the mean collision time to time step size ratio is applicable with its acceptable results. The nozzle mean collision rate (collision frequency) vs. $x$ for all simulations plotted in Figure 4.16 shows higher mean collision rates for higher pressures at constant temperature and lower temperatures at constant pressures and an initially relative increase before the nosedive at the throat along with a relative decrease before and after the throat. Although the conventional nozzle appears to follow the expected trends throughout, the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a relative sharp increase in mean collision rate or decrease in mean collision time at the inlet, as it can be seen to make the temperature difference at constant pressure very small towards the throat resulting in very similar flows past the throat as explained in the Pressure and Temperature Subsection 4.1.4. The mean collision time is relatively higher for the conventional nozzle simulations compared to the MEMS nozzle simulations, which have a relatively higher mean collision rate.

Figure 4.18 shows the nozzle variable hard sphere mean free path vs. $x$ for all simulations. The trends show higher mean free path values for higher temperatures at constant pressures and lower pressures at constant temperatures along with a general decrease towards the throat before a sharp and then gradual increase. Again, the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a relative sharp decrease at the inlet, as it can be seen to make the temperature difference at constant pressure very small towards the throat resulting in very similar flows past the throat. The variable hard sphere mean free path is relatively slightly higher for the conventional nozzle compared to the MEMS nozzle, which is discussed in more depth in the

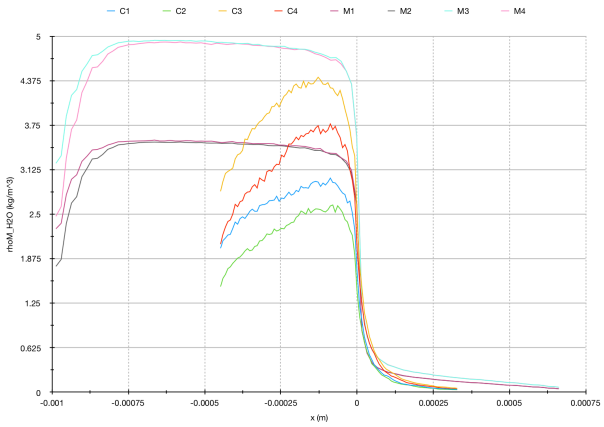



Figure 4.10: Nozzle mass density vs. $x$ for all simulations

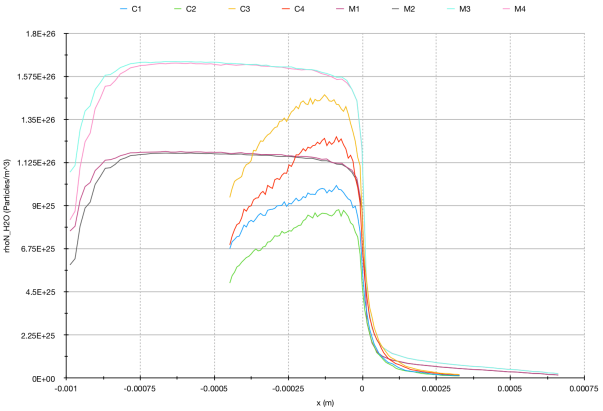



Figure 4.11: Nozzle particle number density vs. $x$ for all simulations

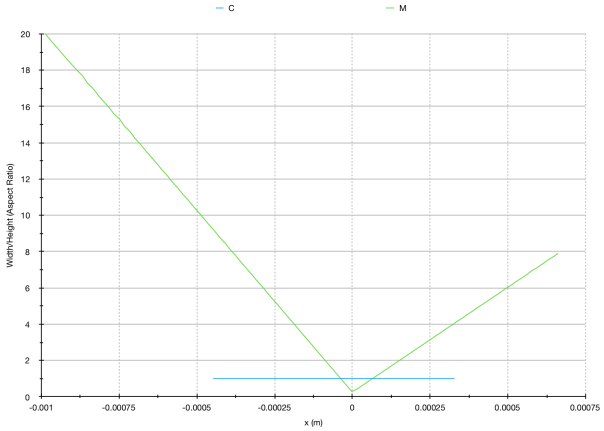



Figure 4.12: Conventional and MEMS nozzles' width/height (aspect ratio) vs. $x$

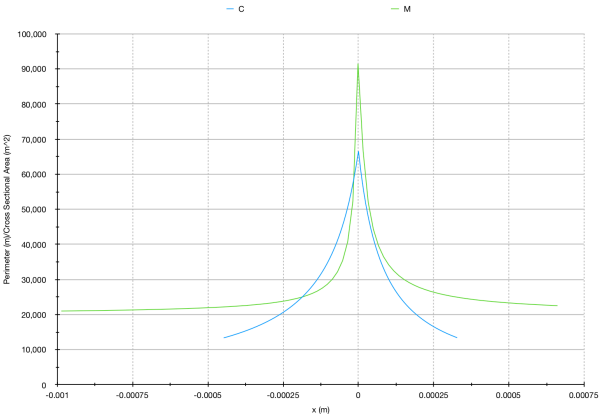



Figure 4.13: Conventional and MEMS nozzles' perimeter to cross sectional area ratio vs. $x$

Knudsen Number Subsection 4.1.10. The same behavior is seen in Figure 4.20 showing the variable hard sphere mean free path/largest cell dimension vs. $x$ for all simulations, where it can be seen that the mean free path is constantly smaller than the largest cell dimension, as has been previously calculated in the checkMesh Subsection 3.2.4. While it is certainly desirable to follow Equation 2.54 for the grid cell size to be at least three times smaller than the mean free path, the simulations' feasibility remaining applicable with its acceptable results has been a priority. The MEMS nozzle's larger bump past the throat is relatively amplified due to the slightly extended throat section's smaller grid cell size, as explained in the Methodology Chapter 3's mesh creation.

Figure 4.17 shows the nozzle mean collision separation vs. $x$ for all simulations and it appears to have an anomaly (high value) for C2, which will be assumedly treated as a plume region value in the analysis. It is still slightly visible, as noticed in the data, that the mean collision separation values dip almost identically for the conventional or MEMS nozzle cases at the throat, where the lower pressure at constant temperature values results become slightly higher than their higher pressure at constant temperature counterparts. Then, the nozzle separation of free paths representing the respective ratio of mean collision separation to variable hard sphere mean free path vs. $x$ for all simulations is plotted in Figure 4.19. Except for the carried anomaly from the mean collision separation data, the results show higher separation of free paths values for lower temperatures at constant pressure and higher pressure at constant temperature with a general increase towards the throat before a sharper and then gradual decrease. Note that the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a relative sharp increase at the inlet, as it can be seen to make the temperature difference at constant pressure very small towards the throat resulting in very similar flows past the throat. The mean collision separation and separation of free paths are higher for the MEMS nozzle compared to the conventional nozzle.

As explained in the OpenFOAM dsmcFoam(+) Solver Subsection 2.4.2 with the Courant-Friedrichs-

Lewy number ($CFL$) being defined in DSMC to physically provide a time step size allowing particles to remain within their grid cell guaranteeing sufficient interaction for accuracy rather than CFD's general usage for stability [35], the nozzle Courant-Friedrichs-Lewy number ($CFL$) vs. $x$ for all simulations is calculated (Equation 2.58) and plotted in Figure 4.21. To obtain the grid cell size along $x$, the Python script (Python Shell Subsection 3.4.1) data is used, where the variable hard sphere mean free path is divided by the variable hard sphere mean free path to largest cell dimension ratio resulting in the used largest cell dimension. Furthermore, the most probable molecular speed is calculated using Equation 2.59 with the universal gas constant ($R_A = 8.3144598\text{J} \cdot \text{K}^{-1} \cdot \text{mol}^{-1} = 8.3144598\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1} \cdot \text{mol}^{-1}$) and water molecular mass ($M_w = 0.018\text{kg} \cdot \text{mol}^{-1}$). As similarly determined in the checkMesh Subsection 3.2.4 calculations, the $CFL$ number proves to be acceptable below one (lower for MEMS nozzle than conventional nozzle) for the used average grid cell and time step sizes due to the relative smallness of the time step size compared to the grid cell size, which means that the time step size can maintain the DSMC particles in their grid cells for a sufficient period at the most probable molecular speed for accuracy in their physical interaction with other particles rather than stability as in CFD's general usage. It can be noted that the values peak at the throat with the higher temperature at constant pressure and lower pressure at constant temperature values maintaining a higher $CFL$ before the throat, though they become rather similar beyond it. Also, the MEMS nozzle's throat peak is more prominent mostly due to the relatively finer throat region grid cell sizes, as explained in the Methodology Chapter 3's mesh creation.

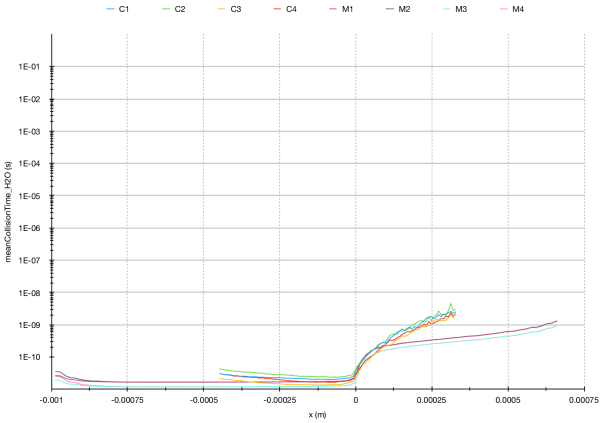



Figure 4.14: Nozzle mean collision time vs. $x$ for all simulations

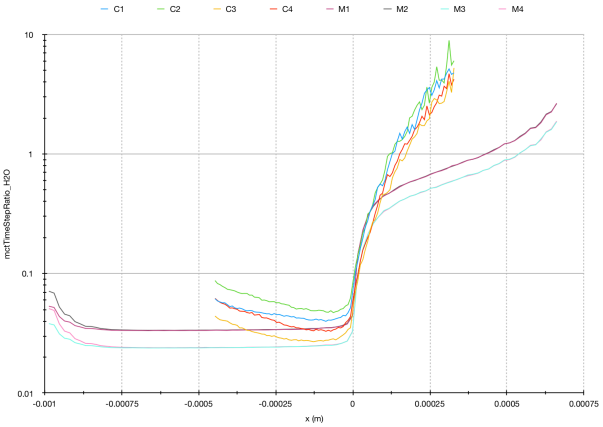



Figure 4.15: Nozzle mean collision time/time step size vs. $x$ for all simulations

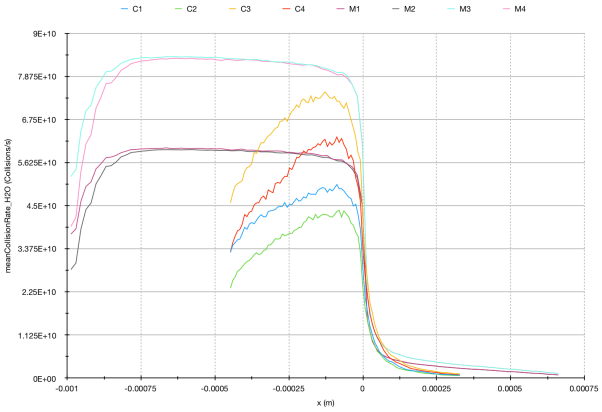



Figure 4.16: Nozzle mean collision rate vs. $x$ for all simulations

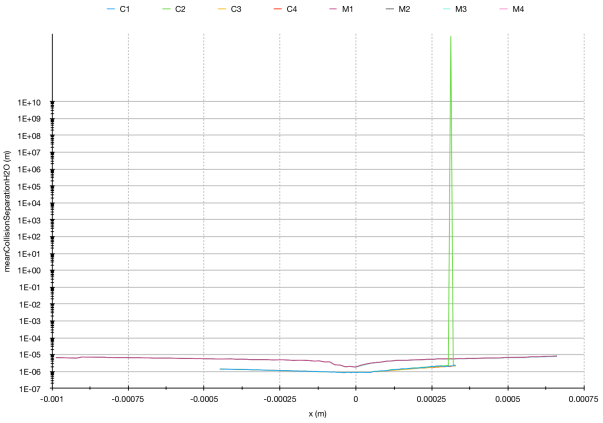



Figure 4.17: Nozzle mean collision separation vs. $x$ for all simulations

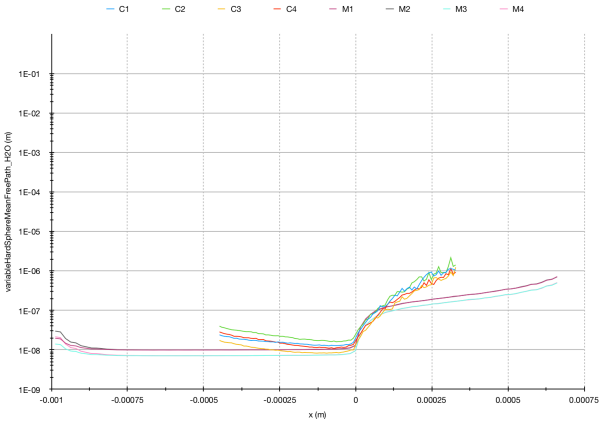



Figure 4.18: Nozzle variable hard sphere mean free path vs. $x$ for all simulations

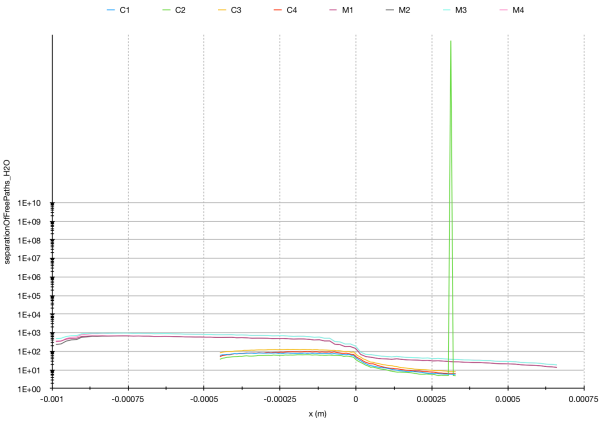



Figure 4.19: Nozzle separation of free paths vs. $x$ for all simulations

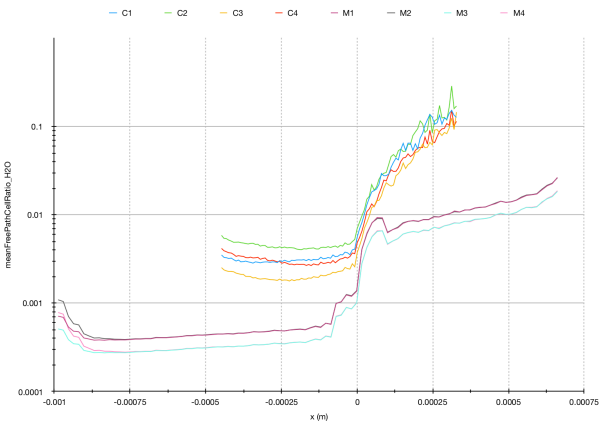



Figure 4.20: Nozzle variable hard sphere mean free path/largest cell dimension vs. $x$ for all simulations

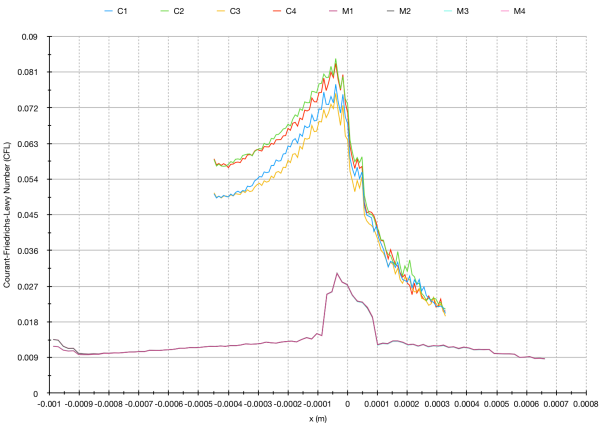



Figure 4.21: Nozzle Courant-Friedrichs-Lewy number ($CFL$) vs. $x$ for all simulations

### 4.1.4. Pressure and Temperature

Figures 4.22 and 4.23 respectively show the nozzle pressure and temperature vs. $x$ for all simulations. Beginning with the pressure, both conventional and MEMS nozzles follow the expected conventional de Laval nozzle pressure drop at the throat, though the conventional nozzle's higher pressure's drop is greater compared to its MEMS nozzle counterpart, while the MEMS nozzle's lower pressure's drop is smaller compared to its conventional nozzle's counterpart. The pressures then plateau with little difference between the higher and lower pressures past the throat. To add, there are some requirements on the design of the propulsion system and what is typically required for such satellites, for example related to operational parameters, where the pressure should not exceed 10 bar.

On the other hand, the temperature is arguably the source for the least realistic results, especially for the MEMS nozzle, though it identifies the significance of heat transfer in the comparison between the conventional and MEMS nozzles. First, the temperature before the throat is higher for higher temperatures at constant pressures and lower pressures at constant temperatures before the temperature difference becomes minimal after the pressure drop at the throat, though the MEMS nozzle has a minimal difference due to pressures at constant temperatures before and after the throat. It becomes immediately noticeable that the MEMS nozzle's temperature is strongly influenced by the wall heat transfer compared to the conventional nozzle, which generally follows a conventional de Laval nozzle temperature drop, even though it is also affected by the wall temperatures. The walls are set at 300 K in the boundariesDict Subsection 3.2.5, which would ideally be realistic for a variable initial wall temperature, possible (regenerative along with potential film, curtain, transpiration, and radiation) cooling to avoid melting (in different conditions considering that the (stored) inlet microresistojet temperature is 283.16 K if it were to be used [27]), or decreasing viscosity (before the throat, while still being increased beyond it (especially in the increasing section of the temperature arch)), as gas viscosity generally increases as temperature increases due to the gas molecular collisions increase, contrary to the liquid viscosity decrease with a temperature increase, considering that it decreases the dominant cohesive force between the liquid molecules. Although the conventional nozzle's temperature drop remains rather realistic at steady state, the MEMS nozzle's temperature drops quickly at the inlet before reaching a temperature stagnation region all the way leading to the throat. This is largely due to the enhanced heat transfer in the MEMS nozzle due to its greater total lateral surface area to total volume ratio (as calculated in Table 4.1) and perimeter to cross sectional area ratio (Figure 4.13). Due to the smaller area (especially the width) of the MEMS nozzle's throat (0.0025 $mm^2$ compared to $0.0028274^2$ mm for the conventional nozzle) along with that the perimeter to cross sectional area ratio is much higher at the throat and its quasi-2D shape has a less than 1 throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2 and Boundary Layers and Rarefaction Phenomena Subsection 4.1.13), the throat (geometry) chokes the flow enough to develop a temperature stagnation region towards the inlet. Also, since the wall temperatures are lower than the set inlet temperatures, the flows actually start from a lower inlet temperature, where it is actually amplified for the MEMS nozzle due to its enhanced heat transfer geometrical properties. Then, the MEMS nozzle's temperature beyond the throat becomes arch shaped (and remains higher than the conventional nozzle's temperature), where it increases slightly due to the relatively higher wall temperature (and still relatively higher perimeter to cross sectional area ratio) before decreasing due to the expansion (relatively lower perimeter to cross sectional area ratio, since its flow beyond the throat becomes relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5) (refer to Boundary Layers and Rarefaction Phenomena Subsection 4.1.13)). To mention, the MEMS nozzle velocity arch along with the temperature arch from the shear on the walls are also seen in [9] (and [51] for the velocity arch).

Therefore, this comparison is quite important in determining the impact of the geometrical characteristics of the conventional and MEMS nozzles on heat transfer. Although the temperatures at the throat and outlet have been calculated in the Analytical Model Subsection 4.2.1 and shown in its MATLAB Workspace Results Subsection B.1.2, an advantage for using a single temperature along the

conventional and MEMS nozzles is that it shows how they handle a heat transfer wall with an identical temperature (indirect insight for studying the heat transfer mechanism along with shape optimization in the used micro heat exchanger for these thrusters and possible consideration of nozzle heat transfer as done for the low-pressure microresistojet) and provides a suitable starting point for further research on more specific conventional vs. MEMS nozzles' wall heat transfer models along with that a single temperature for all simulations allows a clear comparison of the effects of wall heat transfer on the conventional and MEMS nozzles (converging and diverging sections) at different operating conditions, while partially accounting for the nozzle material's thermal conduction in the diverging section.

At first glance, this might seem like a disadvantage for the MEMS nozzle. However, it reaffirms that the MEMS nozzle's geometry provides easier heat transfer with proper exterior insulation mitigating undesired heat rejection, which could be an advantage with the propellant heating involved (even with a lower mass flow rate), as the heat for these thrusters is not coming from chemical reactions, but from resistive microheaters instead. If the spacecraft happens to become warmer (or cooler), especially at the (possibly more exposed) diverging section of the nozzle, for any reason during its mission, the MEMS nozzle will be significantly affected by this temperature change compared to the conventional nozzle. The methodology used highlights the importance of insulation for the MEMS nozzle's thrust, as a material with a lower thermal conductivity (higher thermal resistivity) is desired for the MEMS nozzle, so that the heat before the throat and converging section (from the heater) does not easily conduct heat towards the outlet. This also depends on the resistive heaters' placement, considering that their unobtrusive placement in the center could provide the conventional nozzle an advantage.

In reality, the wall temperatures would vary significantly from the firing time (assumed 300 K) to steady state. Using Table 4.3 (with data from the Analytical Model Subsection 4.2.1's MATLAB Workspace Results Subsection B.1.2) for future simulations of conventional and MEMS nozzles, converging and diverging sections' different approximated wall temperatures could be set based on Equation 2.109 (assuming that the analytical model's expected flow temperature at steady state equals the wall temperature), where the Mach number is 1 at the throat and could be calculated from Equation 2.108 at the exit as done in the Analytical Model Subsection 4.2.1 along with an assumed constant $\gamma$ from the inlet, as it is difficult to find thermodynamic data for the resulting exit temperature and the specific heat ratio is not expected to vary significantly for the considered analysis, as proven by the difference between inlet and throat specific heat ratios in the Analytical Model Subsection 4.2.1, even though their difference with the exit specific heat ratio is expected to be greater, the specific heat ratio at the inlet is still used. Also, instead of just setting the respective approximated average temperatures at the converging and diverging (and throat for the MEMS nozzle since it has been separated during mesh creation as explained in the Methodology Chapter 3), the geometry could easily be subdivided into sufficient subdivisions using Blender (with the suggested add-ons in the General Modeling Properties and Procedure Section 3.1) and prepared for the simulations again using further temperature averaging estimates, which would ideally lead to a more accurate representation of the wall temperature gradient, though this method (inaccurately for de Laval nozzles) assumes that the temperature gradient is linear, which would generally be acceptable. Otherwise, a fixed zero (adiabatic) wall heat flux (using different wall boundary models) could be ideally set or the wall temperature could automatically be adjusted for using Equation 2.109 (ignoring wall radiation in assumed vacuum and material thermal conduction) based on the Mach number and $\gamma$ in the source code for the cells adjacent to the walls' boundary faces, so that there is potentially no need to account for material thermal conductivity. Some additional simulations run using respectively allocated temperatures confirm the expectations that the MEMS nozzle's wall heat transfer remains highly effective and that even when the temperature is set to a considerably lower relative temperature in the diverging section, its velocity still behaves in the same way, while the temperature drops following expected de Laval nozzle behavior with the lower set temperature. This is observed better in the Boundary Layers and Rarefaction Phenomena Subsection 4.1.13, as it relates to the geometrical features of the MEMS nozzle, where the flow beyond the throat (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) becomes relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from

shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). As Table 4.3 shows the conventional and MEMS nozzles' analytical model expected flow temperatures at the inlet, throat, and outlet, where they are to be assumed equal to the wall temperature at steady state, only the lower inlet temperatures' averaged values in the diverging region (not far from the center) could make 300 K more representative of the realistic flow. It might initially seem that the diverging section's rise in temperature after the throat is unintuitive considering that the set wall temperature is generally lower than the wall temperature from Table 4.3, but the converging section's previous temperature drop influences the increase in temperature following the throat's temperature drop that leads to a relatively lower temperature compared to the wall temperature from Table 4.3's throat temperature drop. Generally, the wall temperature should be respectively higher for the converging section and varying from higher to lower in the diverging section.

While it would be more realistic to accommodate for the wall temperature along the nozzle, it must be noted that even with different inlet temperatures for the MEMS nozzle, the temperature is and would have still been significantly influenced by the enhanced wall heat transfer, especially if the average value from Table 4.3 is directly assigned for the converging section. Generally, the conventional nozzle's temperature profile is more applicable as a typical de Laval nozzle, while the MEMS nozzle is quite affected by the wall heat transfer. Note that using multiply averaged converging and diverging sections' wall temperature values (starting from Table 4.3) would make the direct comparison of heat transfer more difficult, while using the averaged converging and diverging sections' wall temperature values might yield relatively more accurate results compared to the ones presented, though with similar plateauing behavior.

By comparing these simulations to an Analytical Model (Subsection 4.2.1) and ANSYS Fluent results from a TU Delft VLM CFD Continuum Model (Subsection 4.2.2) using the Thrust, Specific Impulse, Specific Impulse Quality, Effective Exhaust Velocity, and Discharge Coefficient (Subsection 4.1.12) results in the Discussion Subsection 4.2.3, it turns out that the DSMC models follow similar trends that are comparably correct, although with negatively impacted performance. Therefore, the comparison between the conventional and MEMS nozzles is still valid. Also, notice that some varying temperature and pressure dependent variables could relatively start at the same point in the plots before different behavior is observed since the ratio of higher pressure/temperature to lower pressure/temperature is around 1.4.

Table 4.3: Conventional and MEMS nozzles' analytical model temperatures at the inlet, throat, and outlet

| | **Inlet Temperature (K)** | **Throat Temperature (K)** | **Outlet Temperature (K)** | **Average of Inlet and Throat Temperatures (K) (Converging Section)** | **Average of Throat and Outlet Temperatures (K) (Diverging Section)** |
|---|---|---|---|---|---|
| **C1** | 550 | 474.0893 | 128.5235 | 512.04465 | 301.3064 |
| **M1** | 550 | 474.0893 | 118.2996 | 512.04465 | 296.1945 |
| **C2** | 773 | 677.6507 | 213.5426 | 725.32535 | 445.5967 |
| **M2** | 773 | 677.6507 | 198.4676 | 725.32535 | 438.0592 |
| **C3** | 550 | 472.5628 | 124.4658 | 511.2814 | 298.5143 |
| **M3** | 550 | 472.5628 | 114.3515 | 511.2814 | 293.4572 |
| **C4** | 773 | 677.0628 | 211.7167 | 725.0314 | 444.3898 |
| **M4** | 773 | 677.0628 | 196.6731 | 725.0314 | 436.8680 |

### 4.1.5. Velocity, Root Mean Square Speed, and Most Probable Speed

The nozzle velocity vs. $x$ for all simulations is shown in Figure 4.24. The velocity magnitude is calculated as the square root of the sum of its individual $x$, $y$, and $z$ components squared. The velocity is higher for higher temperatures at constant pressures, though the difference is minimal due to varying pressures at constant temperatures. It gradually increases before the throat, where it increases rapidly. The conventional nozzle's velocity proceeds to increase after the throat, though the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a decreasing to increasing behavior due to the temperature arch along with the geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular

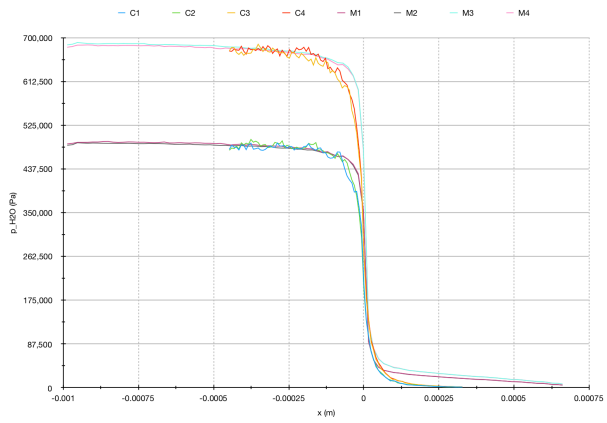



Figure 4.22: Nozzle pressure vs. $x$ for all simulations

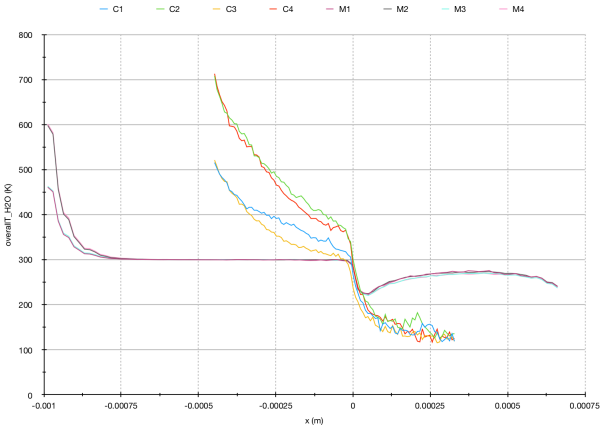



Figure 4.23: Nozzle temperature vs. $x$ for all simulations

horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) of the MEMS nozzle (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13) leading to its flow beyond the throat becoming relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). The velocity is relatively higher for the conventional nozzle simulations compared to the MEMS nozzle simulations.

Figures 4.25 and 4.26 show the nozzle and plume region root mean square speed and nozzle most probable speed vs. $x$ for all simulations respectively. The most probable speed is calculated using Equation 2.59, while the relatively higher root mean square speed is calculated using the following equation using the temperature ($T$) (with the universal gas constant ($R_A = 8.3144598\text{J} \cdot \text{K}^{-1} \cdot \text{mol}^{-1} = 8.3144598\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1} \cdot \text{mol}^{-1}$) and water molecular mass ($M_w = 0.018\text{kg} \cdot \text{mol}^{-1}$)):

$$v_{rms} = \sqrt{\frac{3R_A T}{M_w}}, \tag{4.5}$$

The trends in Figures 4.25 and 4.26 expectedly follow the temperature trends, with higher molecular speeds for higher temperatures at constant pressures. The molecular speeds are slightly higher with higher pressures at constant temperatures before the throat. They gradually decrease before the throat for the conventional nozzle, while there is a plateauing sharp decrease at the inlet for the MEMS nozzle leading to the throat due to the temperature stagnation region from the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4). At the throat, the molecular speeds drop rapidly. The conventional nozzle's molecular speeds proceed to decrease after the throat, though the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes an increasing to decreasing behavior due to the temperature arch. The molecular speeds are relatively higher for the conventional nozzle simulations compared to the MEMS nozzle simulations before the throat and relatively lower after the throat. The root mean square speed drops at the outlet before a gradual decrease in the plume region.

### 4.1.6. Mass Flow Rate, Mass Flux, and Particle Flux

The nozzle mass flow rate (with the fully rotated conventional nozzle) as plotted vs. $x$ for all simulations in Figure 4.27 is calculated using the latter part of Equation 2.102 with data from the Python script (Python Shell Subsection 3.4.1). Note that the conventional nozzle's mass flow rate is multiplied by 360° (full rotation) / 5° (wedge) to obtain the full conventional nozzle values. Although the mass flow rates are relatively constant, a general slight decrease can be seen throughout the nozzle (especially

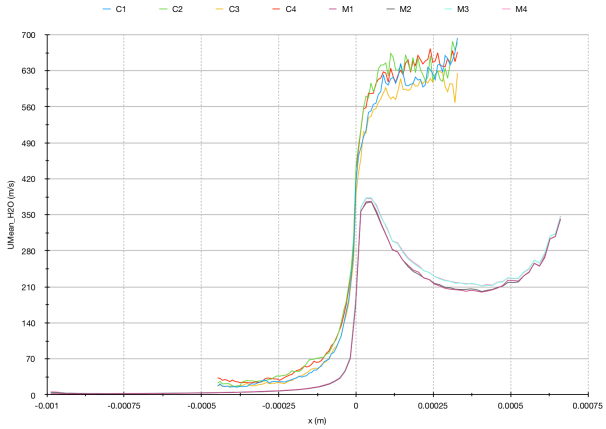



Figure 4.24: Nozzle velocity vs. $x$ for all simulations

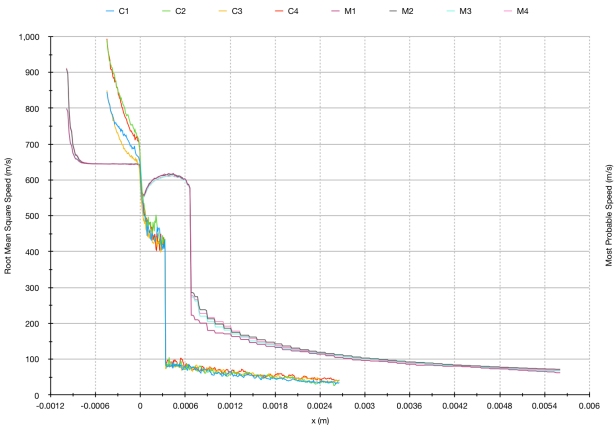



Figure 4.25: Nozzle and plume region root mean square speed vs. $x$ for all simulations

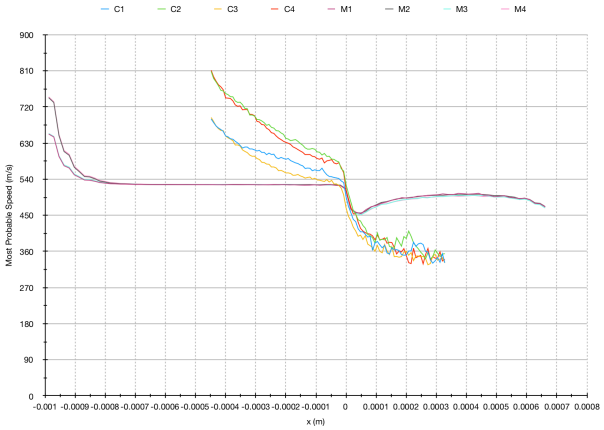



Figure 4.26: Nozzle most probable speed vs. $x$ for all simulations

at the inlet) most likely due to numerical artifacts. Mainly, cases with higher pressures at constant temperatures have higher mass flow rate with the conventional nozzle simulations' mass flow rates being greater than their counterparts in the MEMS nozzle simulations.

The next figures will be listed for their respective inlet, throat, or outlet faces as explained in createCellZones and createFaceZones Subsection 3.2.3 and fieldPropertiesDict Subsection 3.2.9. The data is obtained using the additional steps from the decomposeParDict, balanceParDict, and loadBalanceDict Subsection 3.2.10 and Running and Managing the Simulations Section 3.3. Figures 4.28, 4.29, and 4.30 show the nozzle inlet, throat, and outlet mass flow rates (with full conventional nozzle) respectively vs. time for all simulations. It can be seen that the greater starting pressure ratio allows for a larger mass flow rate at the inlet, though it decreases gradually to a steady state asymptote over time. The nozzle throat and outlet mass flow rates start with a lower mass flow rate rapidly increasing and overshooting before decreasing to a steady state asymptote over time. The (positive) overshoot also occurs due to the higher starting pressure ratio, which is balanced over time. To note, the MEMS nozzle's does not exactly overshoot at the outlet as expected possibly due to the geometry and throat choking leading to an earlier effective balancing. Eventually, the conventional nozzle simulations have a higher mass flow rate than their counterparts in the MEMS nozzle simulations along with that higher pressures at constant temperatures result in higher mass flow rates as seen in Figure 4.27. Figures 4.31, 4.33, and 4.35 show the nozzle inlet, throat, and outlet mass fluxes respectively vs. time for all simulations, while Figures 4.32, 4.34, and 4.36 show the nozzle inlet, throat, and outlet particle fluxes respectively vs. time for all simulations. They follow the same behavior as their respective mass flow rates as the mass flux multiplied by the respective cross sectional area or the particle flux multiplied by the respective cross sectional area and the molar mass for water $\left(M = 18.015 \frac{\text{g}}{\text{mol}}\right)$ divided by the

Avogadro constant $\left(N_A = 6.022140857 \cdot 10^{23}\text{mol}^{-1}\right)$ lead to the mass flow rate.

The mass flow rate and mass and particle fluxes over time could be helpful in extracting time-sensitive data in consideration of the accuracy needed for space technology micropropulsion and to transiently study how the propellant flows when the thruster fires. For consistency, the mass flow rates from the Python script (using macroscopic velocity and density values) (Python Shell Subsection 3.4.1) are mainly used for calculations, as they respectively appear to be acceptably close to the more statistically accurate (due to the significant number of particles crossing their faces) face data.

Knudsen Paradox

The following is inapplicable to the studied cases, as they do not exceed the slip flow regime (spoiler from the Knudsen Number Subsection 4.1.10). Nevertheless, it is still useful information for different conditions.

The Knudsen Paradox represents the phenomenon arising in channels with varying width or pressures, where a minimum mass flux occurs at around $Kn = 0.8$, as can be seen when the channel's normalized mass flux is plotted against $Kn$ with the channel width as the characteristic length scale [31]. It seems unusual since the Navier-Stokes equations would tell that a higher $Kn$ would lead to a lower mass flux, but it can be better explained by considering two Knudsen numbers, where one is significantly high and the other is significantly low [31]. The significantly low $Kn$ would have negligible viscosity, where infinite flux would be found for a fully developed steady state channel flow, while the significantly high $Kn$ would result in negligible particle interaction and infinite flux would also be found as constant acceleration is developed from the external force [31]. This analysis can provide insight into the physical accuracies of the models. A possible normalized mass flow rate ($m_n$) could be evaluated as:

$$m_n = \frac{L\sqrt{2RT}}{(p_{in} - p_{out})h_{in}^2 w}\dot{m}, \tag{4.6}$$

where $L$ is the channel length, $h$ is the channel height, and $w$ is the channel depth, assumed to be one for 2D geometries [19].

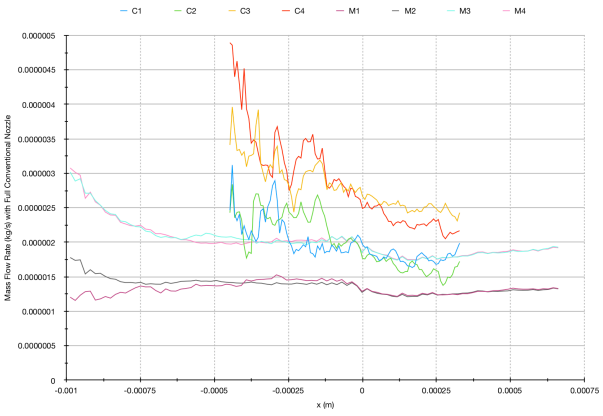



Figure 4.27: Nozzle mass flow rate (with full conventional nozzle) vs. $x$ for all simulations

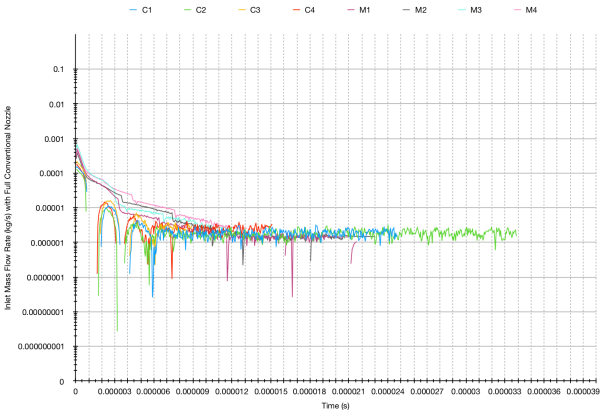



Figure 4.28: Nozzle inlet mass flow rate (with full conventional nozzle) vs. time for all simulations

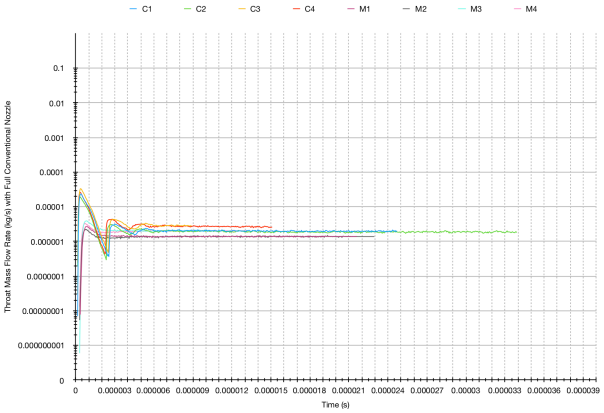



Figure 4.29: Nozzle throat mass flow rate (with full conventional nozzle) vs. time for all simulations

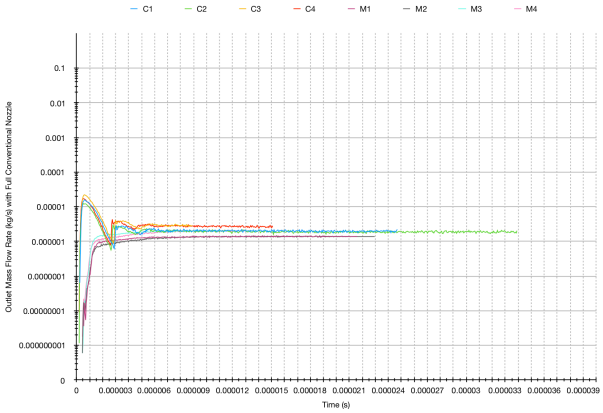



Figure 4.30: Nozzle outlet mass flow rate (with full conventional nozzle) vs. time for all simulations

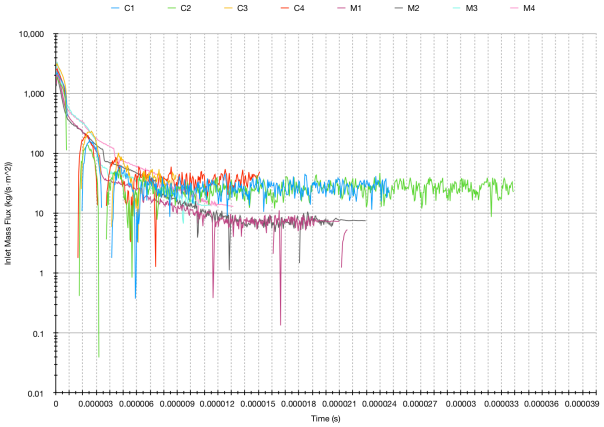



Figure 4.31: Nozzle inlet mass flux vs. time for all simulations

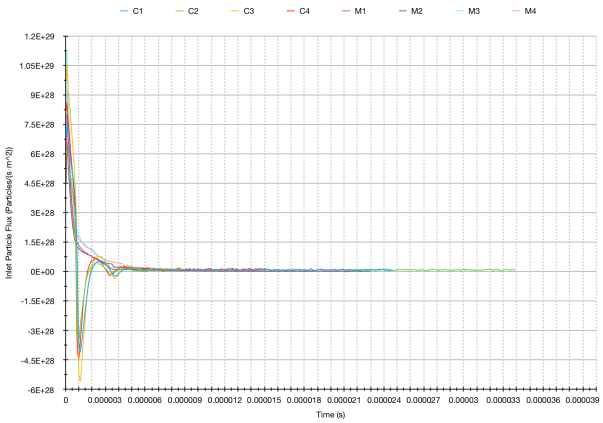



Figure 4.32: Nozzle inlet particle flux vs. time for all simulations

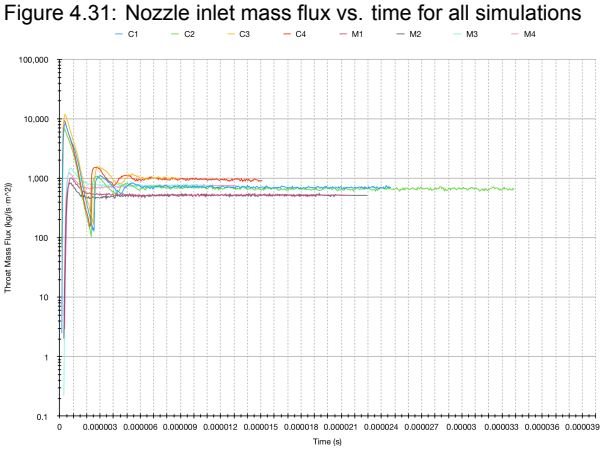



Figure 4.33: Nozzle throat mass flux vs. time for all simulations

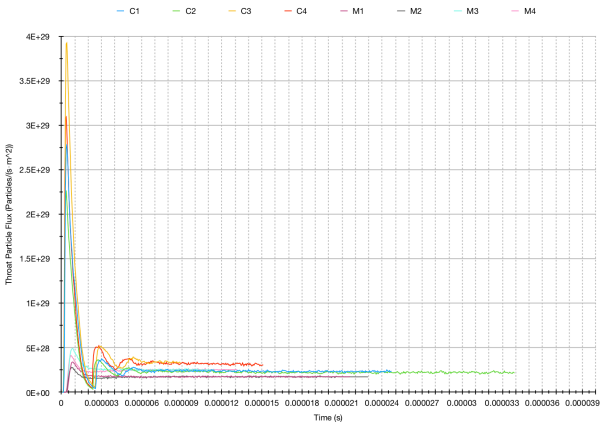



Figure 4.34: Nozzle throat particle flux vs. time for all simulations

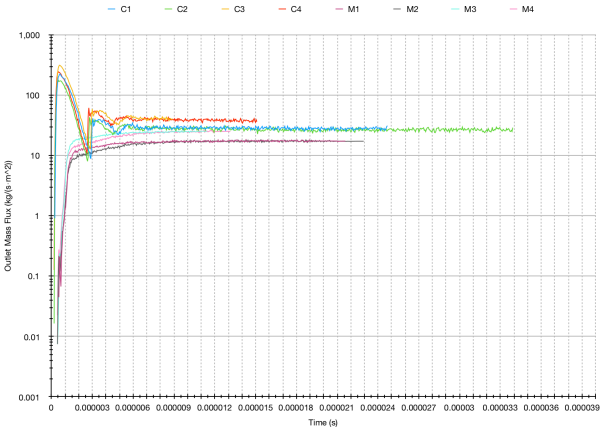



Figure 4.35: Nozzle outlet mass flux vs. time for all simulations

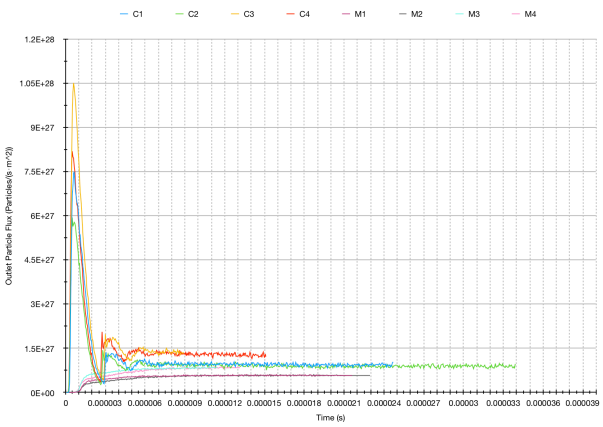



Figure 4.36: Nozzle outlet particle flux vs. time for all simulations

### 4.1.7. $DN$ and $VDN$

The following analysis assumes basic prior understanding of the introduced Extended Continuum/Kinetic Dimensionless Numbers for Diffusivity and Rarefaction Intensity in Appendix A. Equation A.12 is used to calculate $VDN$ in the nozzle and plume region and plot it against $x$ for all simulations in Figure 4.37. The root mean square speed multiplied by the variable hard sphere mean free path result in the intrinsic diffusive transport rate, which is plotted for the nozzle and plume region against $x$ for all simulations in Figure 4.38. Next, the variables depend on a representative physical length scale, which is the hydraulic diameter equaling four times the cross sectional area divided by the cross sectional wetted perimeter, considering that it reduces to an equivalent diameter for a circular cross section, for the MEMS nozzle and diameter for the conventional nozzle (Python Shell Subsection 3.4.1). Therefore, the nozzle advective transport rate ($vL$) vs. $x$ for all simulation is plotted in Figure 4.39. The nozzle $DN$ as calculated from Equation A.10 vs. $x$ for all simulations is also plotted in Figure 4.40.

Figure 4.37 shows that $VDN$ increases from below to above one. The conventional nozzle's $VDN$ is constantly higher than the MEMS nozzle's $VDN$. $VDN$ is generally rising from the inlet towards the throat, where it rapidly ascents and gradually plateaus afterwards in the diverging section for the conventional nozzle, while the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a decreasing to increasing behavior due to the temperature arch along with the geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) of the MEMS nozzle (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13) leading to its flow beyond the throat to become relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). There is a relatively smaller drop in $VDN$ right at the outlet, where it continues to gradually increase and plateau afterwards. The rarefaction intensity rises above 1 for the diverging section and plateauing plume region in the conventional nozzle along with the plateauing plume region of the MEMS nozzle, while it is less than 1 for all other respective $x$ values. Also, $VDN$ is generally slightly higher for cases with higher pressures at constant temperatures. The rarefaction intensity's lowness at the sections below 1 mean that the flow experiences relatively more collisions (higher collision frequency/mean collision rate and lower relative rarefaction) relative to the continuum timescale at the instant of evaluation, while it generally increases in some sections afterwards due to the enhanced individual molecular motion, where the respective flow velocity to root mean square molecular speed ratio increases due to a relatively more directed flow with less probable collisions in the continuum timescale (lower collision frequency/mean collision rate and higher relative rarefaction). The nozzle mean collision rate vs. $x$ for all simulations can also be seen in Figure 4.16. The relative collisions are compared to neutral rarefaction intensity of 1, where the (bulk) flow velocity is the same as the root mean square molecular speed, such as for the case of the probabilistically average fluid molecule irrespective of direction.

Figure 4.38's intrinsic diffusive transport rate relatively decreases at the inlet, before a sharp and following gradual increase around the throat. Note that there is a plateauing sharper decrease at the inlet for the MEMS nozzle leading to the throat due to the temperature stagnation region from the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4). At the outlet, the intrinsic diffusive transport rate is significantly higher, while it gradually decreases plateauing, though the MEMS nozzle follows greater variation with a comparable trend. The conventional nozzle's intrinsic diffusive transport rates are relatively higher than their MEMS nozzle counterpart. Lower pressures at constant temperatures and higher temperatures at constant pressures allow for a higher intrinsic diffusive transport rate. As the intrinsic diffusive transport rate increases, the adaptability with relatively higher and lower advective transport rates increases and decreases respectively, and vice versa.

Speaking of the advective transport rate plots, the conventional nozzle shows higher advective transport rate generally all throughout compared to the MEMS nozzle. The advective transport rate slightly decreases before increasing around the throat, where it grows steeply. For the conventional

nozzle, the advective transport rate continues to gradually increase beyond the throat. To note, a plateauing sharper decrease at the inlet is seen for the MEMS nozzle leading to the throat due to the temperature stagnation region from the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4). It also causes a slightly decreasing to increasing behavior in the diverging section of the MEMS nozzle due to the temperature arch as explained in the Pressure and Temperature Subsection 4.1.4 along with the geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) of the MEMS nozzle (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13) leading to its flow beyond the throat to become relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). As the advective transport rate increases, the adaptability with relatively higher and lower intrinsic diffusive transport rate increases and decreases respectively, and vice versa.

From the previously calculated advective transport rate and intrinsic diffusive transport rate, $DN$ could be calculated using Equation A.10. The $DN$ plot shows a higher $DN$ value for higher pressures at constant temperatures along with lower temperatures at constant pressures (mainly for the conventional nozzle), where there is a considerable peak at the throat, reminiscent of $DN$'s direct relation to $Re$. The conventional nozzle's peak is relatively higher compared to the MEMS nozzle, along with that it starts (increases from the throat) and ends (decreases from the throat) at respectively higher $DN$ values. The outlet $DN$ is slightly greater than at the inlet $DN$. The $DN$ values are consistently greater than 1, which indicates that the advective transport rate or $VDN$ is relatively greater than the intrinsic diffusive transport rate or $Kn$ respectively. Therefore, the flow is relatively faster than it should be relative to its greater representative physical length scale and its rarefaction intensity is greater than its rarefaction, where the respective values of $VDN$ and the advective transport rate are greater than $Kn$ and the intrinsic diffusive transport rate, also considering Equation A.13.

### 4.1.8. Reynolds Number

Using the first part of Equation 2.1 with a representative physical length scale of the hydraulic diameter equaling four times the cross sectional area divided by the cross sectional wetted perimeter, considering that it reduces to an equivalent diameter for a circular cross section, for the MEMS nozzle and diameter for the conventional nozzle (Python Shell Subsection 3.4.1) along with a constant respective inlet dynamic viscosity due to the difficulty of finding thermodynamic data throughout the nozzle for the resulting temperatures, Figure 4.41 shows the nozzle $Re$ (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations. Furthermore, Equation A.9 is used to calculate the nozzle $Re$ using $DN$ (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations in Figure 4.42. Like $DN$, plots show a higher $Re$ value for higher pressures at constant temperatures, though a clearly higher $Re$ can also be seen for lower temperatures at constant pressures for both conventional and MEMS nozzles. The gradual increase from the inlet leads to a sharp increase at the throat, before a sharp then gradual decrease towards the outlet, where $Re$ is found to be slightly greater than at the inlet. The conventional nozzle's peak is relatively higher compared to the MEMS nozzle (which is a better sign in de Laval nozzles), along with that it starts (increases from the throat) and ends (decreases from the throat) at respectively higher $DN$ values. Although the $Re$ using $DN$ plot follows the same trends, its values are generally higher, especially after the inlet, along with that the greater $Re$ due to lower temperatures at constant pressures is difficult to recognize for the MEMS nozzle compared to the conventionally determined $Re$, considering that the difference between the different temperatures at constant pressures becomes relatively smaller. To note, $DN$ becomes relatively closer to $Re$ moving away from the inlet, also considering that the $Re$ calculation uses a constant respective inlet dynamic viscosity.

Therefore, $DN$ is capable of approximately detecting the flow regime within its assumptions for this application and could be more helpful when thermodynamic data for calculating $Re$ is difficult to obtain, such as here with the dynamic viscosity. The throat flow is transient (transitional flow between laminar

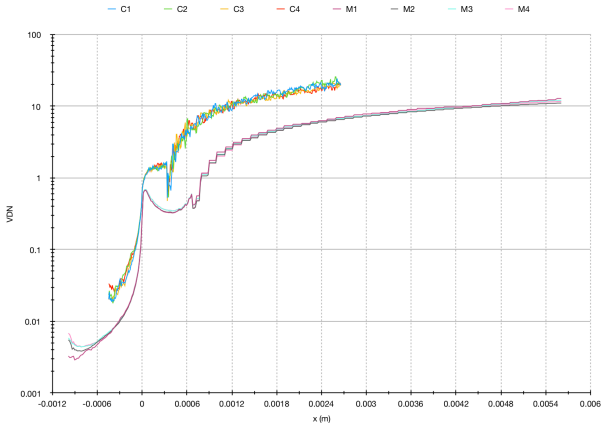



Figure 4.37: Nozzle and plume region $VDN$ vs. $x$ for all simulations

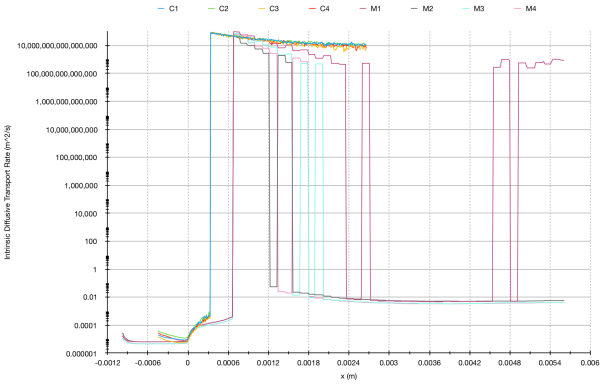



Figure 4.38: Nozzle and plume region intrinsic diffusive transport rate ($\tilde{\nu}\lambda$) vs. $x$ for all simulations

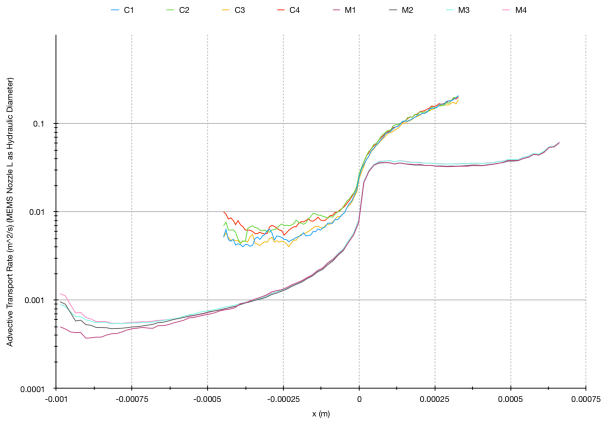



Figure 4.39: Nozzle advective transport rate ($\nu L$) (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations

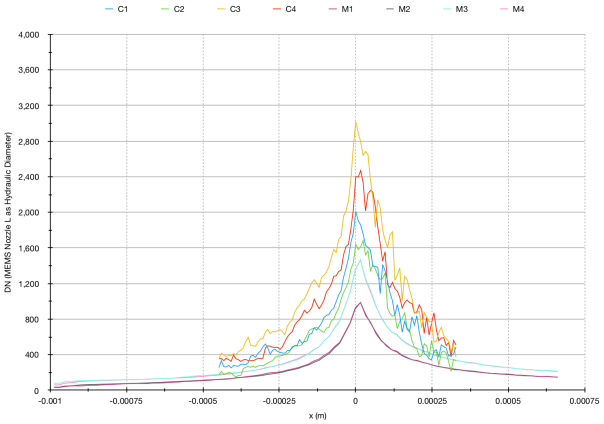



Figure 4.40: Nozzle $DN$ (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations

and turbulent with $2300 < Re < 4000$) for the conventional nozzle at 550 K and both pressures, while laminar ($Re < 2300$) for the other cases. To explain, the viscosity of the gas generally increases with increasing temperature, because of the increase in molecular collisions in the gas, contrary to the decrease of a liquid viscosity with an increase in temperature, which decreases the dominant cohesive force between the liquid molecules. Furthermore, the transient flow in the mentioned cases is also influenced by the density increase with lower temperature. Viscous losses are greater in MEMS nozzles compared to conventional nozzles generally and higher temperature and lower pressure nozzles specifically, as the Reynolds numbers are lower in these cases indicating viscous force dominance. As seen in [11], lower Reynolds numbers mainly in quasi-2D nozzle flows (MEMS nozzle) could cause mentionable inaccuracies using 2D CFD simulations due to the 3D endwall (flat walls capping the side flow channel walls) effects.

### 4.1.9. Mach Number

The nozzle $Ma$ and $Ma$ using $VDN$ (Equation A.15 using a constant inlet $\gamma$ due to the difficulty in finding thermodynamic data along the nozzle for the resulting temperatures and considering that the specific heat ratio is not expected to vary significantly for the considered analysis) vs. $x$ for all simulations are plotted in Figures 4.43 and 4.44 respectively. The conventional nozzle's $Ma$ is consistently higher than the MEMS nozzle's $Ma$, where a higher pressure at constant temperature leads to a slightly higher $Ma$ in general. The $Ma$ gradually increases from the inlet towards the throat, where it rapidly rises to a value of 1 (sonic) and gradually continues to rise afterwards for the (supersonic) conventional nozzle, while the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a decreasing to increasing dip due to the temperature arch as explained in the Pressure and Temperature Subsection 4.1.4 along with the geometrical features (quasi-2D with smaller throat

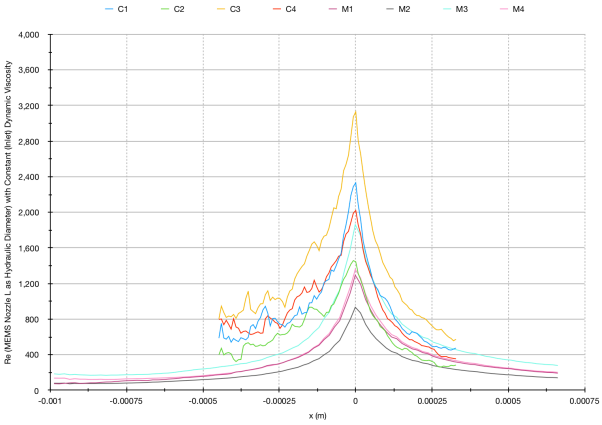



Figure 4.41: Nozzle $Re$ (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations

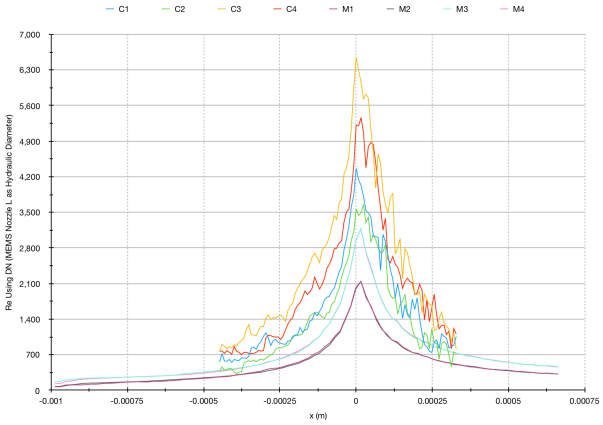



Figure 4.42: Nozzle $Re$ using $DN$ (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations

and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) of the MEMS nozzle (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13) leading to its flow beyond the throat to become relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). One difference to mention about using $VDN$ to calculate $Ma$ is at the outlet, where the conventional nozzle's conventionally calculated $Ma$ appears to be higher. Therefore, $Ma$ using $VDN$ is quite an accurate representation of the conventionally calculated $Ma$. The Mach number could vary significantly (increase) in the initially vacuum plume region, while $VDN$'s approach becomes rather different in such flows ($DN$ and $VDN$ Subsection 4.1.7), which allows for potential applications for $VDN$ and its understanding in highly rarefied flows.

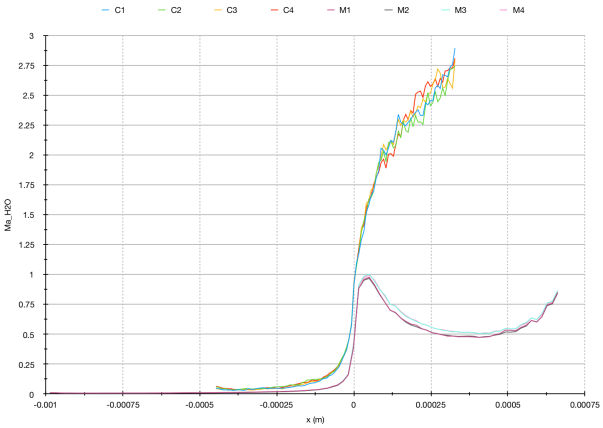



Figure 4.43: Nozzle $Ma$ vs. $x$ for all simulations

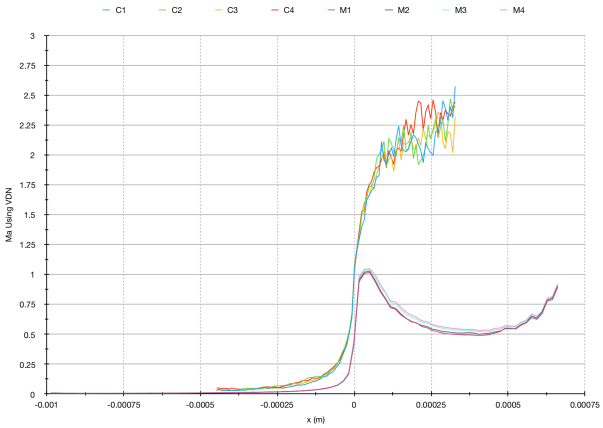



Figure 4.44: Nozzle $Ma$ using $VDN$ vs. $x$ for all simulations

## 4.1.10. Knudsen Number

Figure 4.45 shows the nozzle $Kn$ (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations. The $Kn$ is calculated using Equation 2.5 with a representative physical length scale of the hydraulic diameter equaling four times the cross sectional area divided by the cross sectional wetted perimeter, considering that it reduces to an equivalent diameter for a circular cross section, for the MEMS nozzle

and diameter for the conventional nozzle (Python Shell Subsection 3.4.1). A decreasing to increasing $Kn$ can be seen at the inlet, though the decrease is more rapid for the MEMS nozzle due to the temperature stagnation region from the wall temperature considering its enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) A sharp increase can be seen at the throat, before the gradual plateauing. The Knudsen number remains slightly rarefied within the slip flow regime at the outlet (Figure 2.1), while at continuum flow with normal density levels at the inlet (Figure 2.1), which presents a considerable challenge for DSMC simulations, as the number of particles before and after the throat needs to be optimized. Although the converging section $Kn$ is generally higher for the conventional nozzle, the inlet and outlet $Kn$ are comparable with the MEMS nozzle's $Kn$. This is contrary to expectations, where the MEMS nozzle's $Kn$ is expected to be greater, but that is attributed to the MEMS nozzle's diverging section slightly increasing to decreasing temperature arch as explained in the Pressure and Temperature Subsection 4.1.4 considering its enhanced wall heat transfer along with the geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) of the MEMS nozzle (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13) leading to its flow beyond the throat becoming relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). It also slightly increases the variable hard sphere mean free path and $Kn$ dipping after the throat for the MEMS nozzle. Furthermore, lower pressures at constant temperatures along with higher temperatures at constant pressures (mainly before the throat) lead to higher $Kn$.

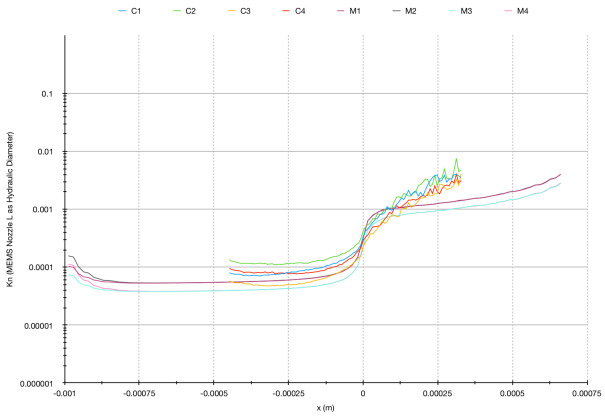



Figure 4.45: Nozzle $Kn$ (with MEMS nozzle $L$ as hydraulic diameter) vs. $x$ for all simulations

### 4.1.11. Statistical Errors

Figures 4.46, 4.47, 4.48, and 4.49 show the nozzle and plume region density, pressure, temperature, and velocity fractional errors respectively vs. $x$ for all simulations (see Subsection 3.2.9). The fractional errors are generally higher for the conventional nozzle compared to the MEMS nozzle. The fractional errors decrease (increase for velocity) at the beginning of the conventional nozzle's converging section, while there is a sharper drop (rise for velocity) at the the inlet of the MEMS nozzle due to the temperature stagnation region from the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4). Towards the throat, the fractional errors increase rapidly (decrease rapidly for velocity). The conventional nozzle's fractional errors proceed to increase (decrease for velocity) in the diverging section, though the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a decreasing to increasing (increasing to decreasing for velocity) behavior due to the temperature arch along with the geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal

flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) of the MEMS nozzle (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13) leading to its flow beyond the throat becoming relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). Also, a drop followed by a plateauing rise is seen for the conventional nozzle after the outlet, while a rise (drop for velocity) followed by a plateauing drop is seen for the MEMS nozzle. The sudden variances are mainly due to the average statistical cell particle number term in the statistical error calculations considering the mesh creation (Methodology Chapter 3). Since the fractional errors generally do not generally exceed a value considerably higher than 0.1 (especially in the more important nozzle section for the density, pressure, and temperature fractional errors), they are considered acceptable to maintain the simulations' feasibility with acceptable results.

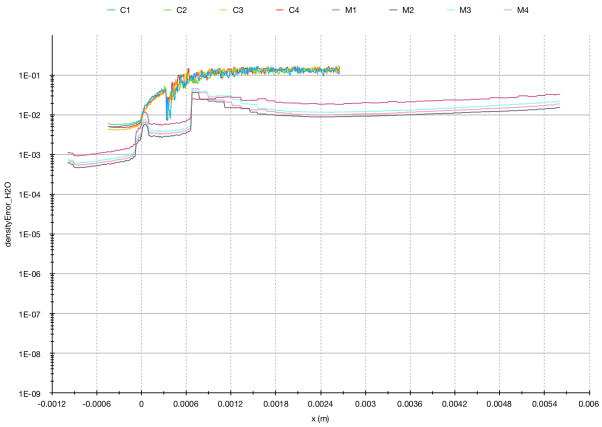



Figure 4.46: Nozzle and plume region density fractional error vs. $x$ for all simulations

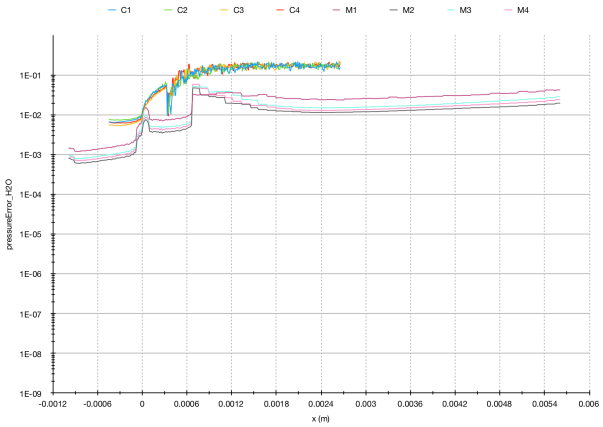



Figure 4.47: Nozzle and plume region pressure fractional error vs. $x$ for all simulations

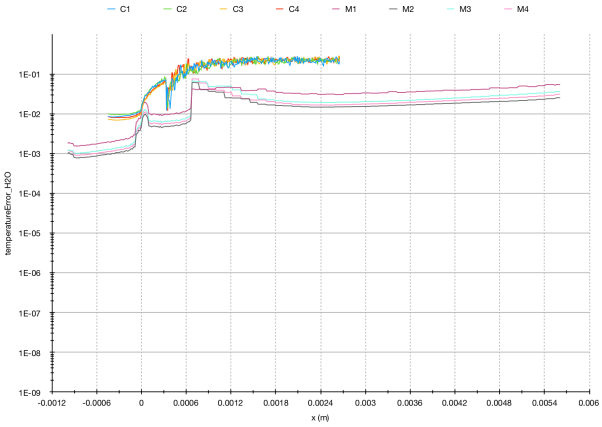



Figure 4.48: Nozzle and plume region temperature fractional error vs. $x$ for all simulations

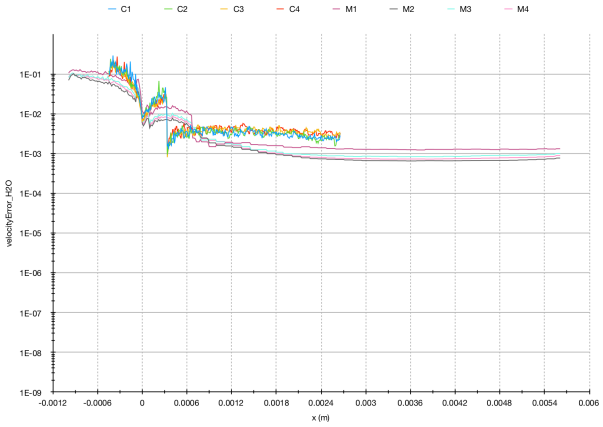



Figure 4.49: Nozzle and plume region velocity fractional error vs. $x$ for all simulations

### 4.1.12. Thrust, Specific Impulse, Specific Impulse Quality, Effective Exhaust Velocity, and Discharge Coefficient

The DSMC model general results at the exit using data from the Python script (Python Shell Subsection 3.4.1) are shown in Table 4.4. The thrust is calculated using Equation 2.98 with the predetermined (outlet) mass flow rate (Subsection 4.1.6 with full conventional nozzle) and X (0) component of (axial)

velocity (Subsection 4.1.5) and assuming vacuum ambient pressure. At the outlet, the axial velocity is only slightly lower than the velocity magnitude calculated as the square root of the sum of its individual $x$, $y$, and $z$ components squared and they are more similar for the MEMS nozzle compared to the conventional nozzle, as the MEMS nozzle's flow is more directed due to the smaller throat size (geometry) along with its lower Reynolds number towards the outlet. Note that the conventional nozzle's area is multiplied by 360° (full rotation) / 5° (wedge) to obtain the full conventional nozzle value. Afterwards, the specific impulse can be found using the Equation 2.92 to be followed by the specific impulse quality, which is the specific impulse divided by the ideal specific impulse obtained from the Analytical Model Subsection 4.2.1 and multiplied by 100 for percentage, where 100% is considered ideal. Then, Equation 2.96 is used to compute the effective exhaust velocity. Note that Earth's gravity is used as 9.81 $\frac{m}{s^2}$. The outlet mass flow rate is then divided by the ideal mass flow rate from the Analytical Model Subsection 4.2.1 to determine the discharge coefficient, where 1 is considered ideal. Notice that the conventional nozzle's thrust and specific impulse (specific impulse quality and effective exhaust velocity) are higher (better) in general. Although the performance of the MEMS nozzle could be more affected by the set wall temperature considering its enhanced heat transfer (Pressure and Temperature Subsection 4.1.4), the conventional nozzle's generally better performance is also expected for assumed adiabatic nozzle walls in both conventional and MEMS nozzles, considering that it is also (less) impacted by the wall heat transfer. Also, the outlet mass flow rate is higher for the conventional nozzle, though the discharge coefficient is closer to 1 for the conventional nozzle at lower temperatures at constant pressures and the MEMS nozzle at higher temperatures at constant pressure, as the mass flow rate is smaller due to the enhanced wall heat transfer and boundary layers. To note, the discharge coefficient of the conventional nozzle at lower pressure is above 1, while it is below 1 for the rest of the cases. Furthermore, higher pressures at constant temperatures and lower temperatures at constant pressures increase the thrust, though the opposite occurs for specific impulse and effective exhaust velocity. The specific impulse quality decreases as pressures at constant temperatures and temperatures at constant pressures increase. The outlet mass flow rate is higher for higher pressures at constant temperatures and lower temperatures at constant pressures. A greater discharge coefficient is determined for higher temperatures at constant pressures for both conventional and MEMS nozzles along with lower pressures at constant temperatures for the conventional nozzle and its inverse for the MEMS nozzle due to the the enhanced wall heat transfer effects on the MEMS nozzle resulting in a less variant mass flow rate due to temperature.

Table 4.4: DSMC model general results

| *Chamber Pressure = 5 bar Chamber Temperature = 550 K* | **Thrust (mN)** | **Specific Impulse (s)** | **Ideal Specific Impulse (s)** | **Specific Impulse Quality (%) (100% Is Ideal)** | **Effective Exhaust Velocity (m/s)** | **Outlet Mass Flow Rate (mg/s) (Full Conventional Nozzle)** | **Ideal Mass Flow Rate (mg/s)** | **Discharge Coefficient (1 Is Ideal)** |
|---|---|---|---|---|---|---|---|---|
| **MEMS Nozzle** | 0.8747 | 67.0694 | 134.9964 | 49.6823 | 657.9506 | 1.3294 | 1.6645 | 0.7987 |
| **Conventional Nozzle** | 1.4418 | 73.9577 | 133.8721 | 55.2451 | 725.5254 | 1.9872 | 1.8825 | 1.0556 |
| *Chamber Pressure = 5 bar Chamber Temperature = 773 K* | | | | | | | | |
| **MEMS Nozzle** | 0.8745 | 67.2568 | 164.4151 | 40.9067 | 659.7897 | 1.3255 | 1.3894 | 0.9540 |
| **Conventional Nozzle** | 1.2513 | 74.0582 | 162.8660 | 45.4719 | 726.5108 | 1.7223 | 1.5713 | 1.0961 |
| *Chamber Pressure = 7 bar Chamber Temperature = 550 K* | | | | | | | | |
| **MEMS Nozzle** | 1.2604 | 66.6355 | 134.3234 | 49.6083 | 653.6945 | 1.9282 | 2.3349 | 0.8258 |
| **Conventional Nozzle** | 1.6013 | 67.2088 | 133.2313 | 50.4452 | 659.3180 | 2.4287 | 2.6408 | 0.9197 |
| *Chamber Pressure = 7 bar Chamber Temperature = 773 K* | | | | | | | | |
| **MEMS Nozzle** | 1.2598 | 66.8433 | 164.1816 | 40.7130 | 655.7325 | 1.9212 | 1.9462 | 0.9871 |
| **Conventional Nozzle** | 1.5340 | 72.1548 | 162.6444 | 44.3636 | 707.8390 | 2.1671 | 2.2011 | 0.9846 |

### 4.1.13. Boundary Layers and Rarefaction Phenomena

The figures presented are obtained using methodology in sampleDict Subsection 3.2.12. Figures 4.50 to 4.59, 4.60 to 4.69, 4.70 to 4.79, and 4.80 to 4.89 represent the nozzle inlet, converging section, throat, diverging section, and outlet (in twos) vs. distance (with full conventional nozzle) or y or z (of MEMS nozzle in every other figure) vs. temperature, velocity, pressure, and particle number density respectively for all simulations. Note that some insignificant anomalies are disregarded in the analysis, where the general trends are mainly considered.

The temperature is higher for the cases at higher temperatures at constant pressures. At the inlet,

the gradient is larger for the conventional nozzle leading to higher maxima, with the longer dimension cases (y compared to distance and distance compared to z) having a flatter front due to the geometry, which is still to be more impacted by the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) considering its quasi-2D geometry. In the temperature stagnation region of the converging section caused by the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4), the MEMS nozzle's temperature is rather uniform, compared to a consistently lower temperature at the sides along with greater maxima for the conventional nozzle. At the throat, the shape is inverted for the MEMS nozzle, with higher temperatures at the sides of the geometry, though the conventional nozzle maintains a peak at the front, even with a decreasing to increasing temperature profile from the sides towards the center. In the diverging section, the conventional and MEMS nozzles have the fully inverted profile with higher temperatures at the sides, with the conventional nozzle having generally lower temperatures. At the outlet, the conventional nozzle maintains the higher temperatures at the sides and lower temperatures compared to the MEMS nozzle, though the MEMS nozzle exhibits a flatter profile on the y axis, but an increasing to decreasing temperature profile at the sides on the z axis.

The velocity is higher for the cases at higher temperatures at constant pressures. At the inlet, the gradient is larger for the conventional nozzle leading to higher maxima, with flatter fronts for the MEMS nozzle due to the geometry (though the z axis exhibits a relative decreasing to increasing velocity profile), which is still to be more impacted by the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) considering its quasi-2D geometry. In the temperature stagnation region of the converging section caused by the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4), the MEMS nozzle's velocity is still rather uniform on the y axis, compared to a consistently lower velocity at the sides along with greater maxima for the conventional nozzle along with the MEMS nozzle on the z axis to a lesser extent. At the throat, the profiles are developed with higher velocities at the sides of the geometry for both conventional and MEMS nozzles. In the diverging section, the conventional and MEMS nozzles profiles remain consistent with their respective throat profiles. At the outlet, the conventional nozzle along with the MEMS nozzle on the z axis maintain the higher velocity at the sides and lower velocities compared to the MEMS nozzle, though the MEMS nozzle exhibits a flatter profile on the y axis. Note that this reiterates the importance of using the Python script (Python Shell Subsection 3.4.1) for a more global representation along the nozzle, considering that parameters like the velocity (which the Mach number trends mainly follow) could considerably vary between the lateral sides and center while using the local Plot Over Line Filter (ParaView Section 3.4).

The pressure is higher for the cases at higher pressures at constant temperatures. Initially at the inlet and converging section, the conventional and MEMS nozzle pressures are comparable and relatively flat. At the throat, the conventional nozzle profile holds lower pressures at the sides and higher pressures at the front, though the y axis MEMS nozzle profile exhibits reverse features and the z axis MEMS nozzle profile remains comparably flat. In the diverging section, the MEMS nozzle pressures are higher with a sharper front on the y axis compared to the conventional nozzle pressures, though the z axis MEMS nozzle pressures remain flat. At the outlet with greater rarefaction, the MEMS nozzle holds lower pressures at the sides and higher pressures at the centers compared to the conventional nozzle, which also exhibits such features to a lesser extent with lower pressures.

The particle number density is higher for the cases at higher pressures at constant temperatures and lower temperatures at constant pressures. At the inlet, the gradient is relatively larger for the conventional nozzle leading to lower minima, with the longer dimension cases (y compared to distance and distance compared to z) having a flatter front due to the geometry, which is still to be more impacted by the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) considering its quasi-2D geometry. In the temperature stagnation region of the converging section caused by the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4), the MEMS nozzle's particle number density is rather uniform, compared to a consistently higher particle number density at the sides along with lower minima for the conventional nozzle. At the throat, the profiles start inverting from the sides towards a higher particle number density at the center and lower particle number density at the sides, as the conventional nozzle maintains a smaller lower particle number density peak at the front. In the diverging section, both conventional and MEMS nozzles have the fully inverted profile with lower particle number density at the sides and higher particle number density in the center, with the conventional nozzle generally having

a smaller particle number density. At the outlet, the shapes of the profiles are maintained from the diverging section, though the conventional nozzle now relatively exceeds the MEMS nozzle's particle number density due to the nozzles' geometrical shapes and the MEMS nozzle's front particle number density flattening. Note that the DSMC particles at the conventional nozzle's axis of symmetry would appear to be greater in number when visualized in ParaView due to the DSMC axisymmetric simulation properties (dsmcProperties Subsection 3.2.6).

Boundary layer growth depends on the nozzle length or expansion divergence angle. Boundary layer thickness decreases with increasing expansion divergence angle resulting in lower blockage and improved performance, but to a limit, as increased angles also lead to lower useful thrust due to the increased momentum component perpendicular to the thrust axis [11]. The boundary layer thickness' prominence is also seen as the flow becomes more rarefied towards the outlet. Considering the boundary layer, the temperature near the diverging section wall is higher compared to at the center considering the wall boundary layer and temperature (Pressure and Temperature Subsection 4.1.4), as the flow at the sides is also slower than at the center. Regarding the velocity profile at the outlet, it is clear that the MEMS nozzle's performance is significantly more affected by its geometry's boundary layers with the flatter (dominant flat top and bottom wall boundary layers also considering its generally higher perimeter to cross sectional area ratio peaking at the throat (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2) and total lateral surface area to total volume ratio (Table 4.1)) and slower outlet front compared to the conventional nozzle that has a conical 3D shape. Simulating the quasi-2D MEMS nozzle as 3D shows its importance in capturing the 3D flow features compared to a 2D simulation.

Notice that the particles accumulate (especially for the conventional nozzle, as the MEMS nozzle's throat choking diminishes this effect) at the converging lateral sides towards the throat, after which the MEMS nozzle's flow with different geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) becomes relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5). Furthermore, the MEMS nozzle's aspect ratio inversion (rectangular horizontal flow to rectangular vertical flow at the throat back to a rectangular horizontal flow) at the throat appears to mainly affect the flow beyond the throat, contributing to the front center tips becoming generally suddenly sharper relative to the profile compared to the conventional nozzle. Additionally, the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) is also relatively greater at the lateral sides due to the closeness with the walls, which also increases this effect. Also, the pressure at the throat lateral sides (and beyond) is higher for the MEMS nozzle, which raises the need for material strength considerations. To mention, the MEMS nozzle velocity arch along with the temperature arch from the shear on the walls are also seen in [9] (and [51] for the velocity arch).

After the flow develops, the temperature of the fluid at the wall becomes more similar to the wall temperature in the converging section along with having closer to a no-slip condition, though the temperature jump and velocity slip become more noticeable as the flow becomes more rarefied towards the outlet, while considering that the $Kn$ is expected to have been greater for the MEMS nozzle, which is impacted by its diverging section's slightly increasing to decreasing temperature arch as explained in the Pressure and Temperature Subsection 4.1.4 considering its enhanced wall heat transfer and velocity arch. Knudsen layers (kinetic boundary layers) with thickness of the order of mean free path also occur in rarefied flows.

Now, paraFoam is used with the conventional (X Normal with plane Origin at 0.00032969) and MEMS (X Normal with plane Origin at 0.00067116) nozzles clipped using the Clip Filter right before the outlet (using data from Table 4.1) along with Z and Y Normal clippings for the MEMS nozzle to visualize inside the domain (ParaView Section 3.4). The internalMesh is chosen from the Mesh Parts along with variableHardSphereMeanFreePath_H2O from the Volume Fields. Feature Edges is set for the clipped nozzle to visualize its boundaries. It is also set for the conventional nozzle's contours, as it has been simulated as a 5° wedge with single cell thickness and the contour Surface becomes less

noticeable as it thins towards the axis of symmetry, though the domain could have also been symmetrically sliced using the Slice Filter beforehand to create contour plot lines. At the final time step, the Contour Filter with Contour By variableHardSphereMeanFreePath_H2O and Compute Scalars (color by variableHardSphereMeanFreePath_H2O points) is used with an added value range with 350 Steps for the MEMS nozzle and 100 Steps for the conventional nozzle with Use Logarithmic Scale checked to show the largely varying data with enough contours along the nozzle after removing all previous entries. In the Color Map Editor using rescale to custom range, the maximum value is set to 3.0e-07 for the desired visualization. Refer to ParaView Section 3.4 for more information on the settings. Then, Figures 4.90 to 4.97 are generated with nozzle variable hard sphere mean free path contour plots for C1, M1, C2, M2, C3, M3, C4, and M4 respectively.

There is a recurring difference between the variable hard sphere mean free path contour plots of the conventional and MEMS nozzles. The conventional nozzle's variable hard sphere mean free path contours generally extend all the way to the outlet from the throat maintaining the shape of the nozzle with greater values at the diverging section's sides, while the MEMS nozzle's variable hard sphere mean free path contours have more of a straight shape from the throat to the outlet (before further expansion), which leads to relatively greater variable hard sphere mean free paths at the sides along with the front, considering the velocity arch and its causes. More information to compliment this subsection is available in the Pressure and Temperature Subsection 4.1.4, Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5, Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, Perimeter to Cross Sectional Area Ratio Subsection 4.1.2, Mean Collision Time, Mean Collision Rate, Mean Collision Separation, (Local) Variable Hard Sphere Mean Free Path, Separation of Free Paths, and Courant-Friedrichs-Lewy Number Subsection 4.1.3, and Transient Solution (Animation Videos) Subsection 4.1.15.

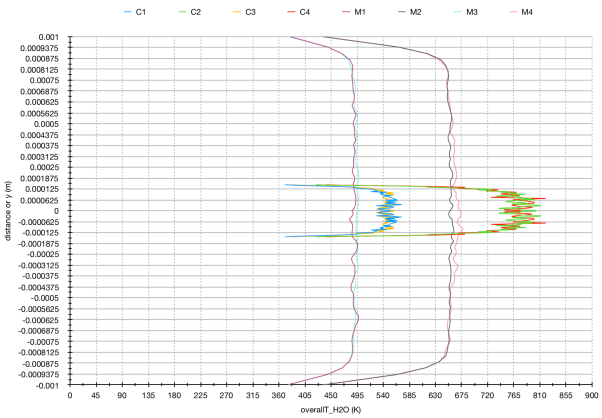



Figure 4.50: Nozzle inlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. temperature for all simulations

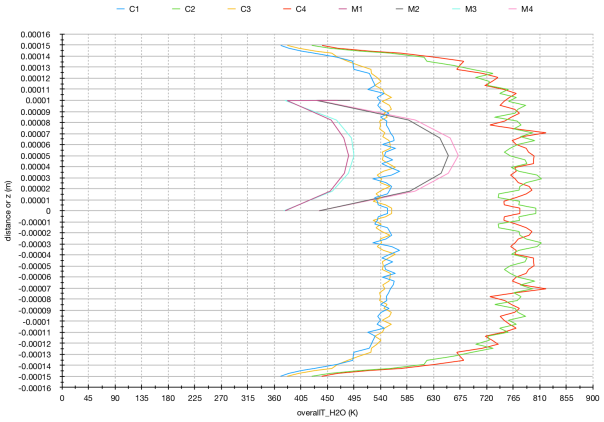



Figure 4.51: Nozzle inlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. temperature for all simulations

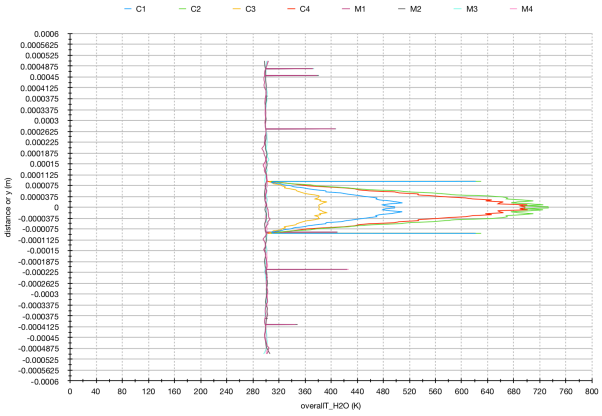



Figure 4.52: Nozzle (middle of) converging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. temperature for all simulations

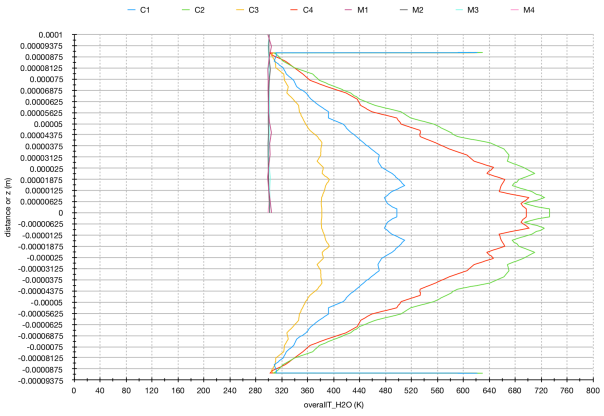



Figure 4.53: Nozzle (middle of) converging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. temperature for all simulations

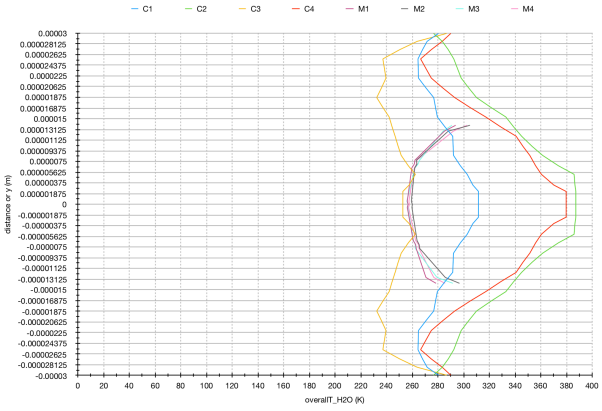



Figure 4.54: Nozzle throat distance (with full conventional nozzle) or y (of MEMS nozzle) vs. temperature for all simulations

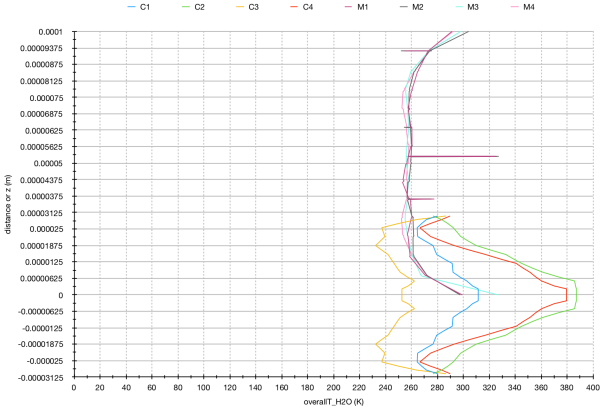



Figure 4.55: Nozzle throat distance (with full conventional nozzle) or z (of MEMS nozzle) vs. temperature for all simulations

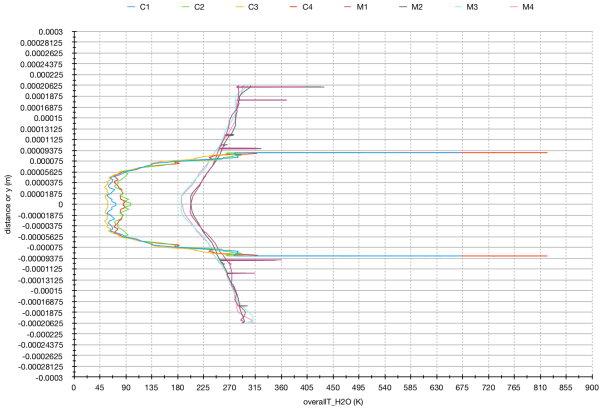



Figure 4.56: Nozzle (middle of) diverging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. temperature for all simulations

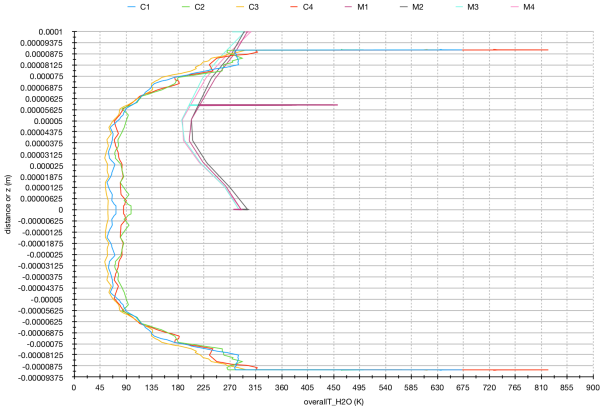



Figure 4.57: Nozzle (middle of) diverging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. temperature for all simulations

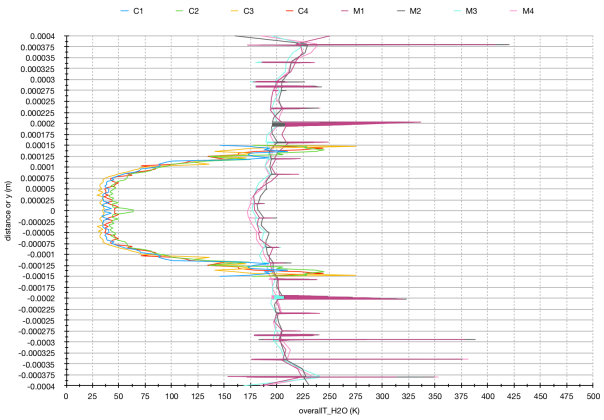



Figure 4.58: Nozzle outlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. temperature for all simulations

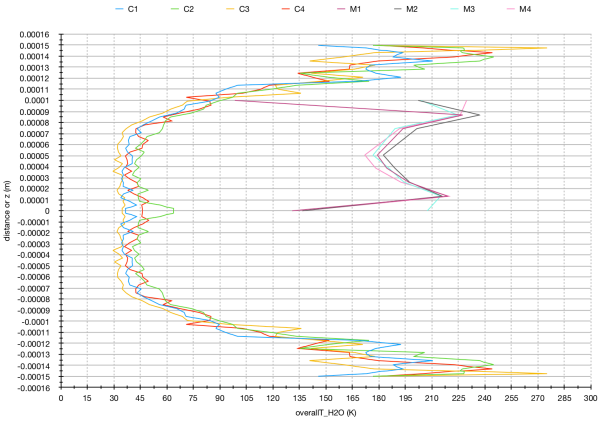



Figure 4.59: Nozzle outlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. temperature for all simulations

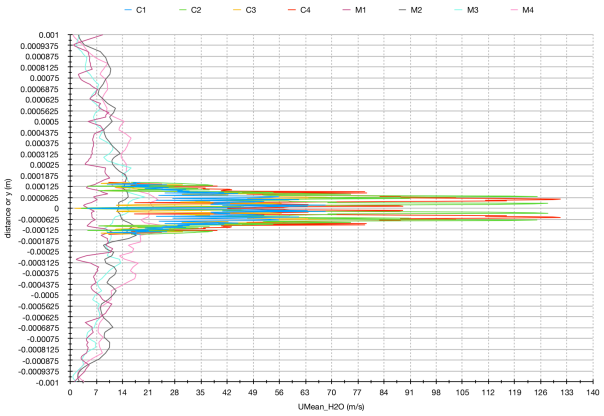



Figure 4.60: Nozzle inlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. velocity for all simulations

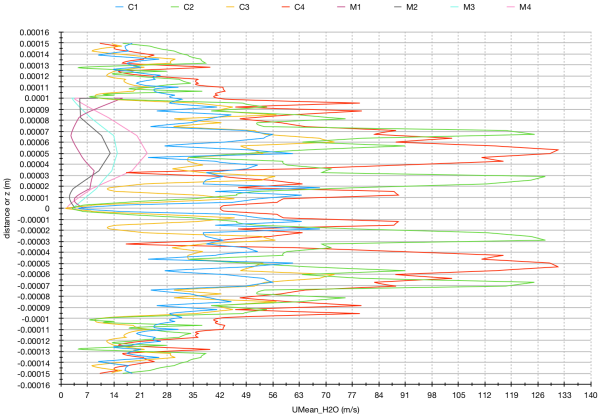



Figure 4.61: Nozzle inlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. velocity for all simulations

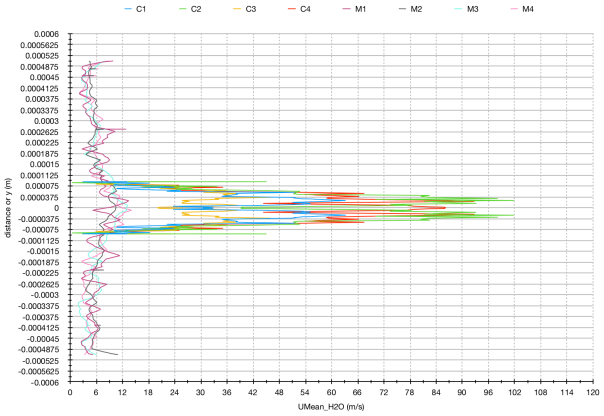



Figure 4.62: Nozzle (middle of) converging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. velocity for all simulations

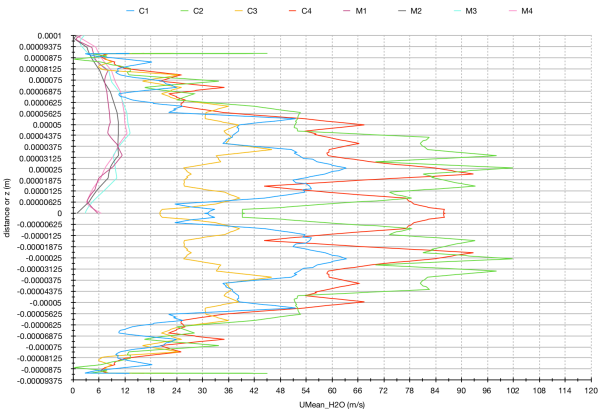



Figure 4.63: Nozzle (middle of) converging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. velocity for all simulations

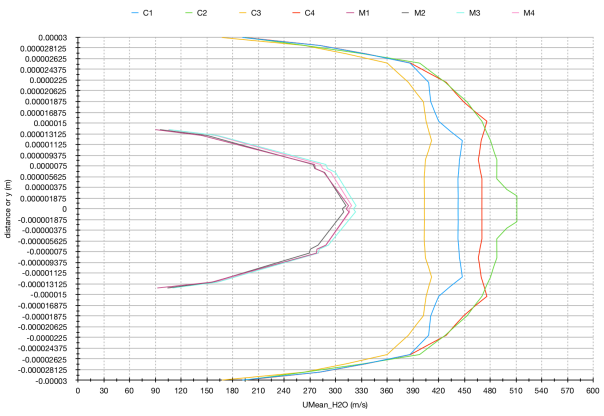



Figure 4.64: Nozzle throat distance (with full conventional nozzle) or y (of MEMS nozzle) vs. velocity for all simulations

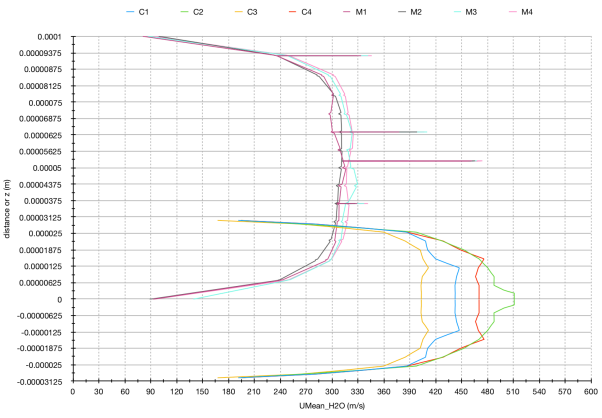



Figure 4.65: Nozzle throat distance (with full conventional nozzle) or z (of MEMS nozzle) vs. velocity for all simulations

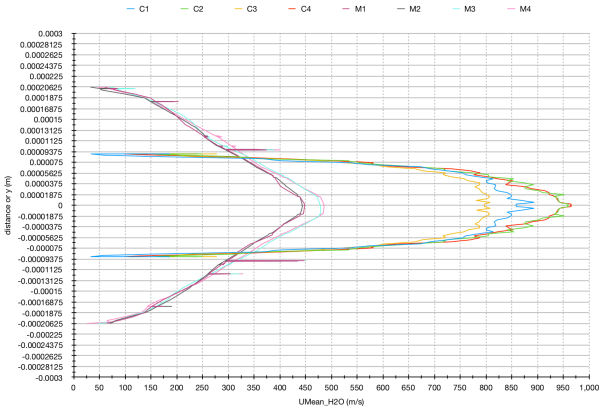



Figure 4.66: Nozzle (middle of) diverging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. velocity for all simulations

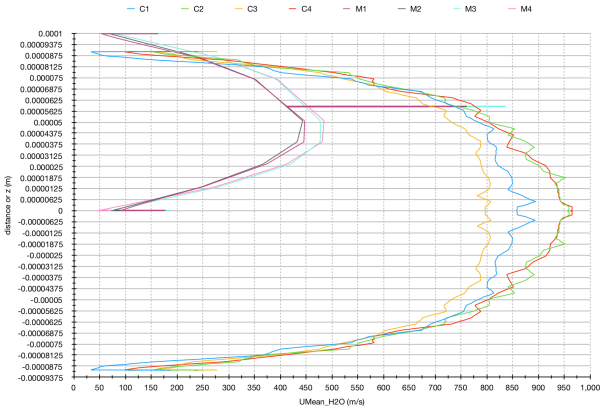



Figure 4.67: Nozzle (middle of) diverging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. velocity for all simulations

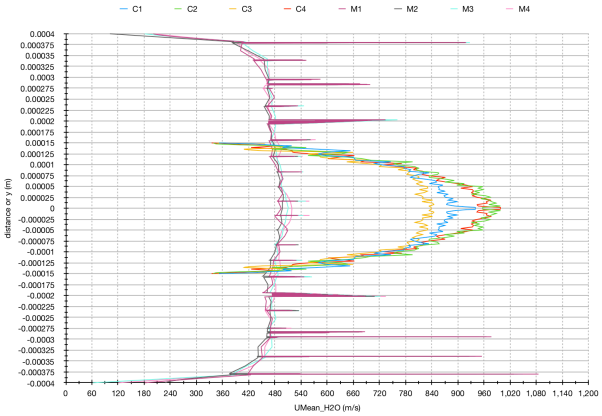



Figure 4.68: Nozzle outlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. velocity for all simulations

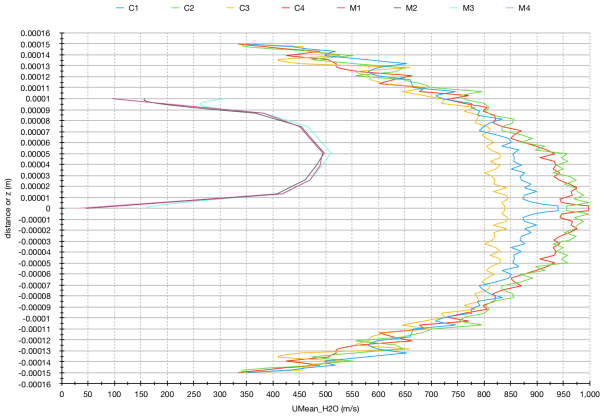



Figure 4.69: Nozzle outlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. velocity for all simulations

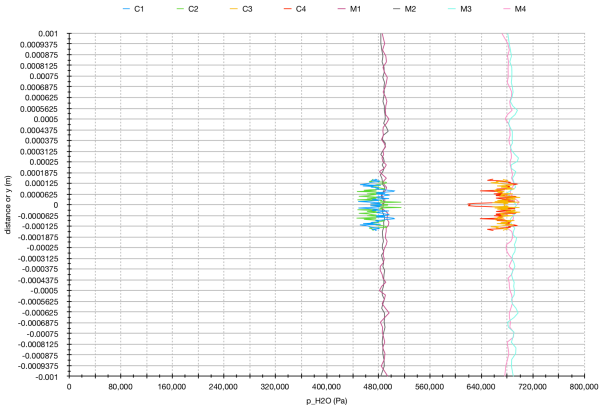



Figure 4.70: Nozzle inlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. pressure for all simulations

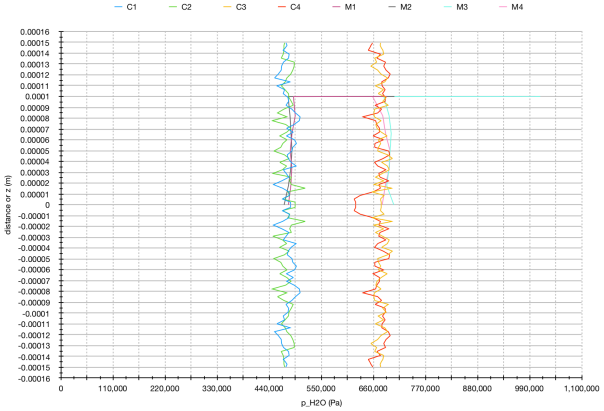



Figure 4.71: Nozzle inlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. pressure for all simulations

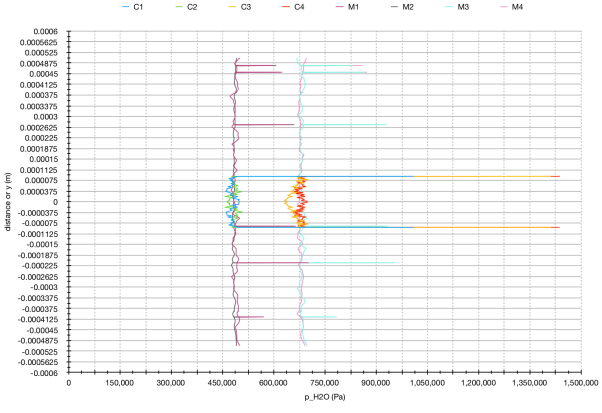



Figure 4.72: Nozzle (middle of) converging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. pressure for all simulations

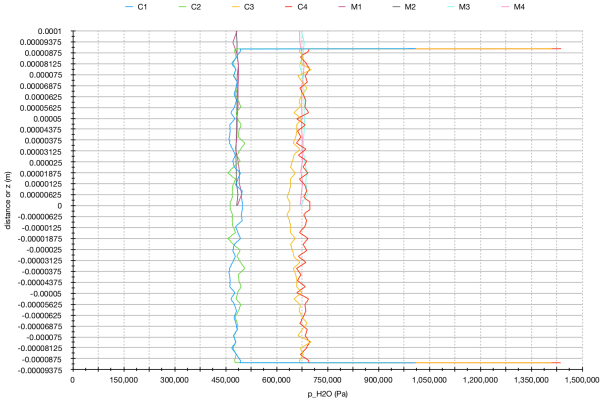



Figure 4.73: Nozzle (middle of) converging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. pressure for all simulations

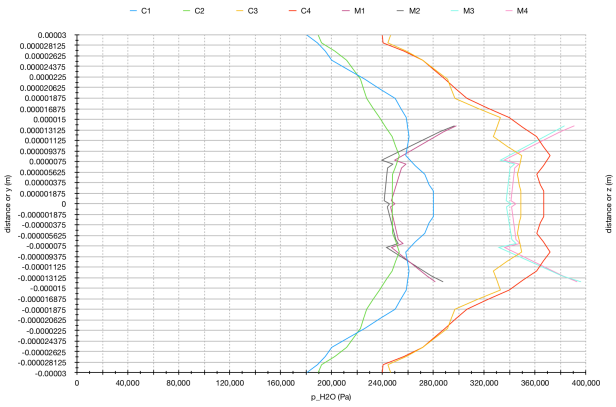



Figure 4.74: Nozzle throat distance (with full conventional nozzle) or y (of MEMS nozzle) vs. pressure for all simulations

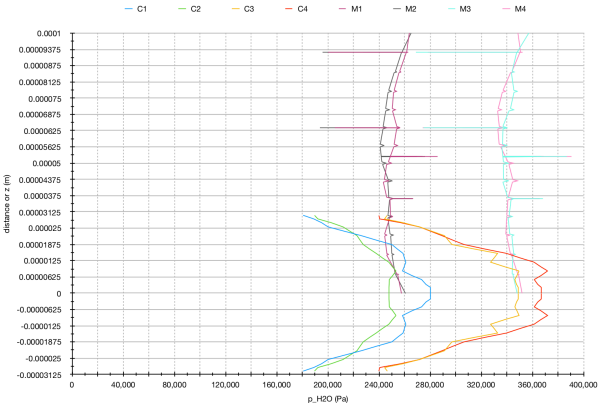



Figure 4.75: Nozzle throat distance (with full conventional nozzle) or z (of MEMS nozzle) vs. pressure for all simulations

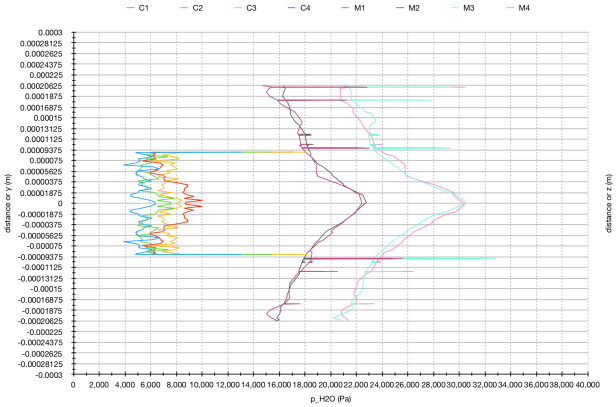



Figure 4.76: Nozzle (middle of) diverging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. pressure for all simulations

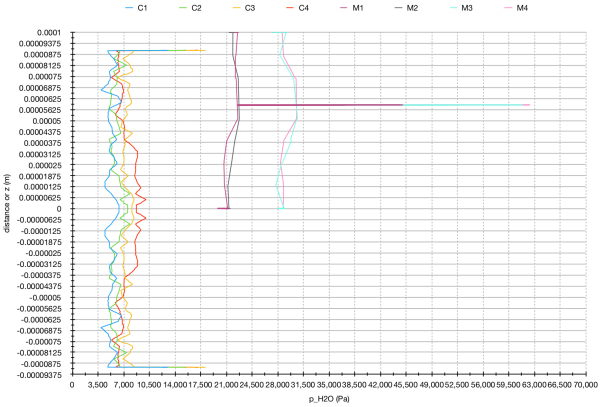



Figure 4.77: Nozzle (middle of) diverging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. pressure for all simulations

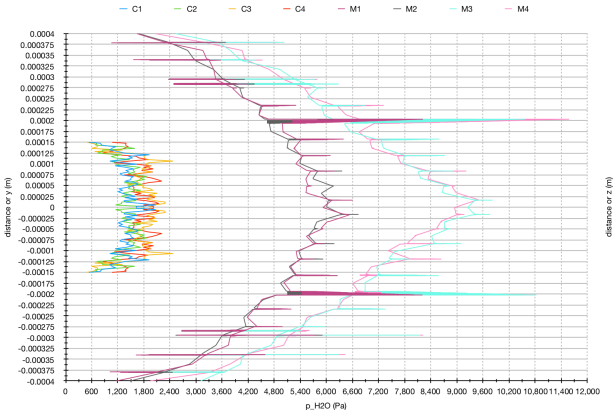



Figure 4.78: Nozzle outlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. pressure for all simulations

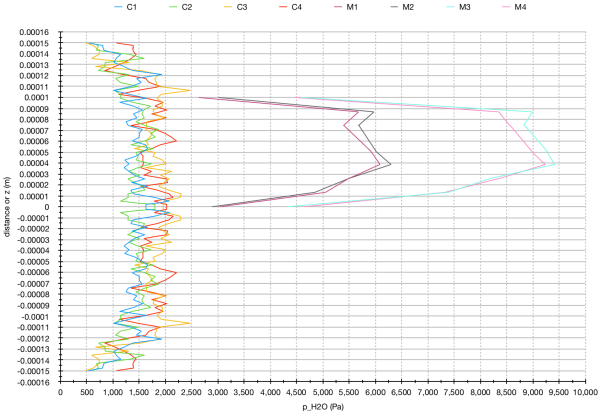



Figure 4.79: Nozzle outlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. pressure for all simulations

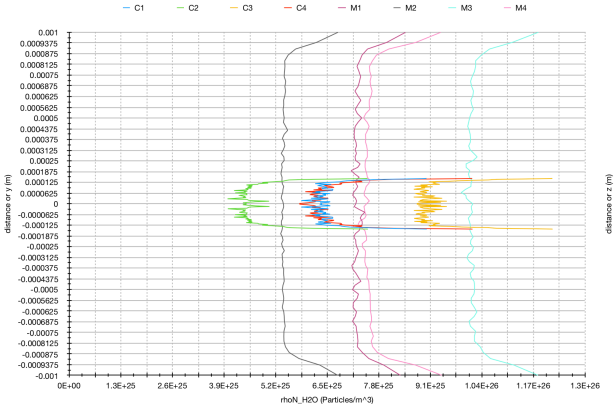



Figure 4.80: Nozzle inlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. particle number density for all simulations

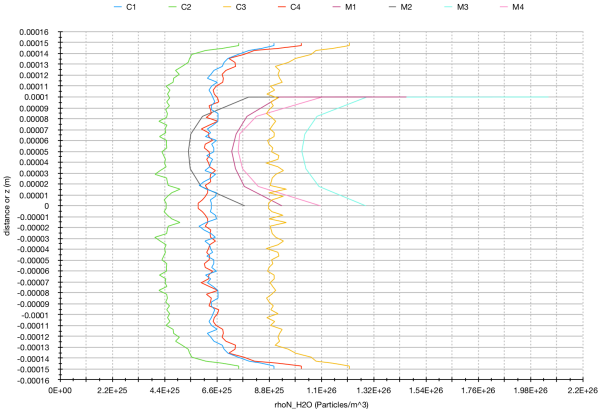



Figure 4.81: Nozzle inlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. particle number density for all simulations

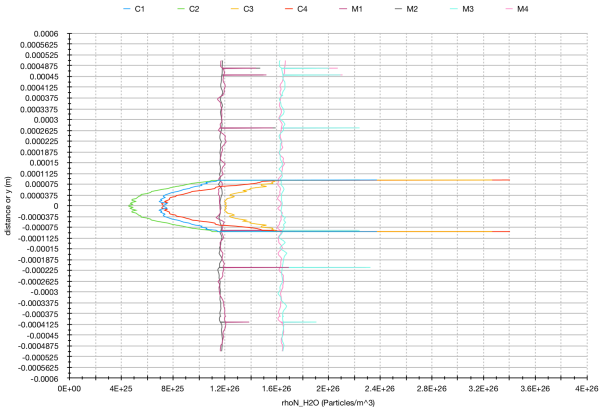



Figure 4.82: Nozzle (middle of) converging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. particle number density for all simulations

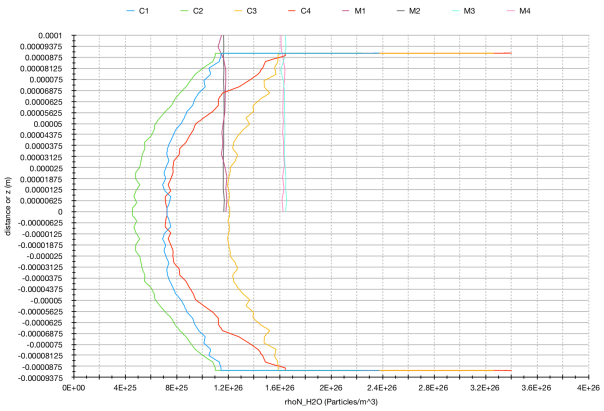



Figure 4.83: Nozzle (middle of) converging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. particle number density for all simulations

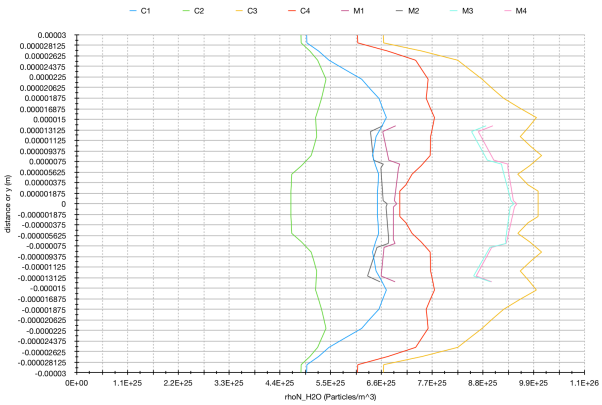



Figure 4.84: Nozzle throat distance (with full conventional nozzle) or y (of MEMS nozzle) vs. particle number density for all simulations

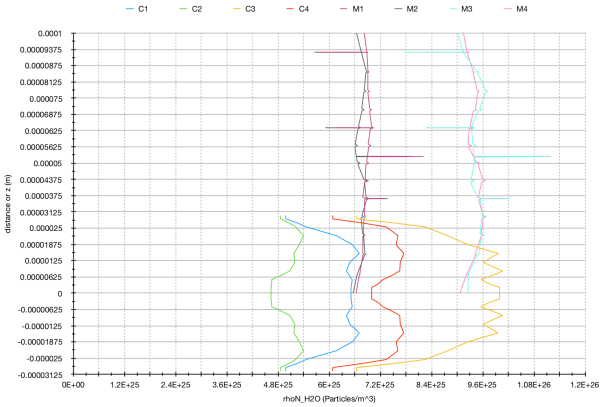



Figure 4.85: Nozzle throat distance (with full conventional nozzle) or z (of MEMS nozzle) vs. particle number density for all simulations

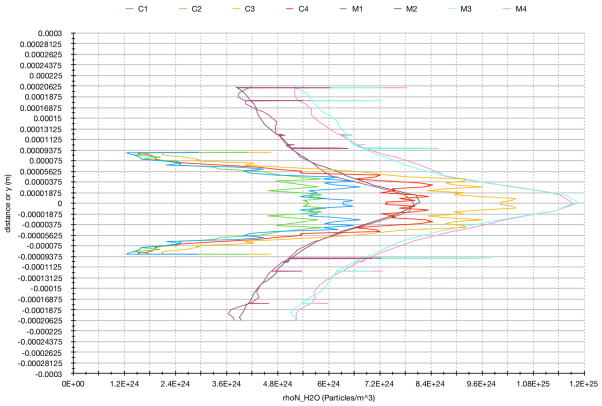



Figure 4.86: Nozzle (middle of) diverging section distance (with full conventional nozzle) or y (of MEMS nozzle) vs. particle number density for all simulations

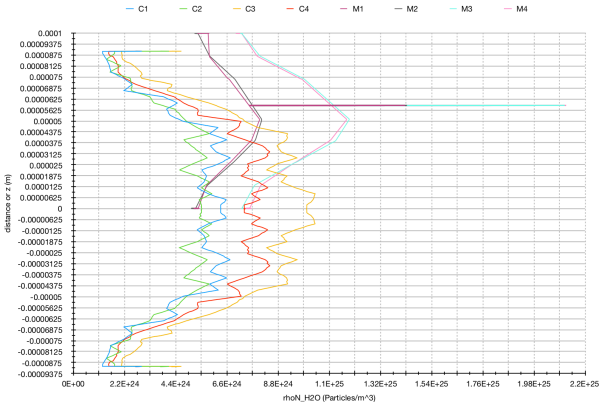



Figure 4.87: Nozzle (middle of) diverging section distance (with full conventional nozzle) or z (of MEMS nozzle) vs. particle number density for all simulations

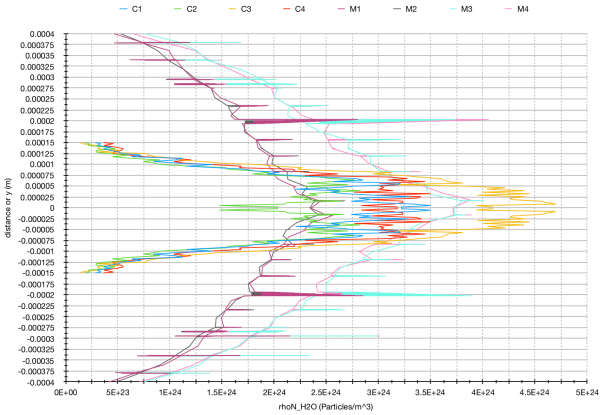



Figure 4.88: Nozzle outlet distance (with full conventional nozzle) or y (of MEMS nozzle) vs. particle number density for all simulations

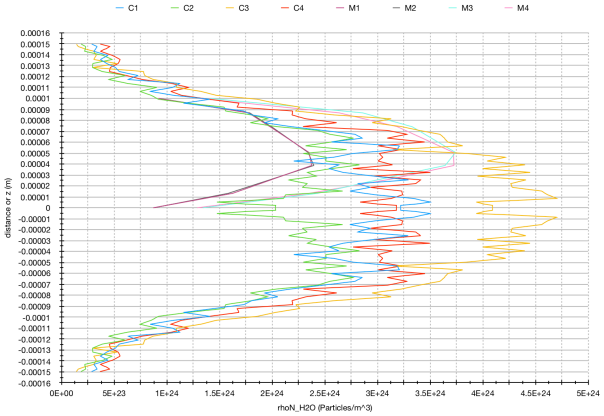



Figure 4.89: Nozzle outlet distance (with full conventional nozzle) or z (of MEMS nozzle) vs. particle number density for all simulations

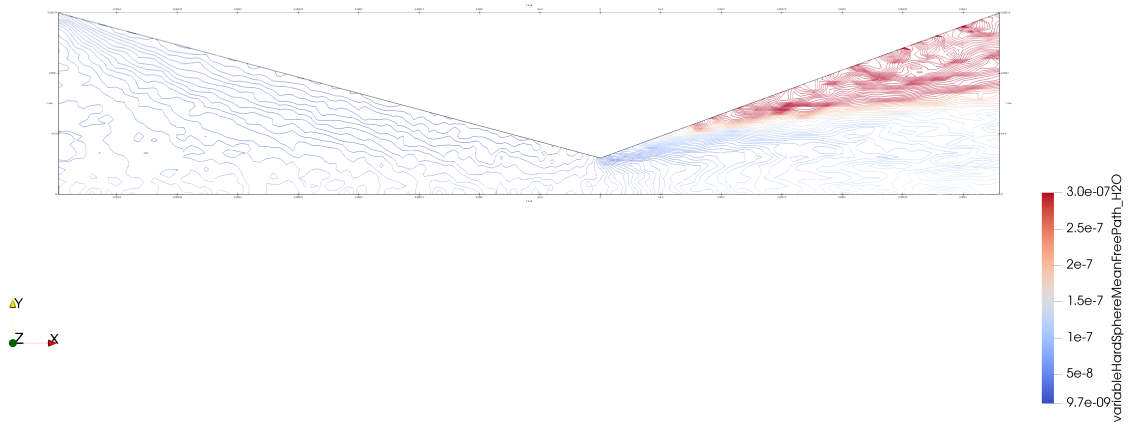



Figure 4.90: C1 nozzle variable hard sphere mean free path contour plot

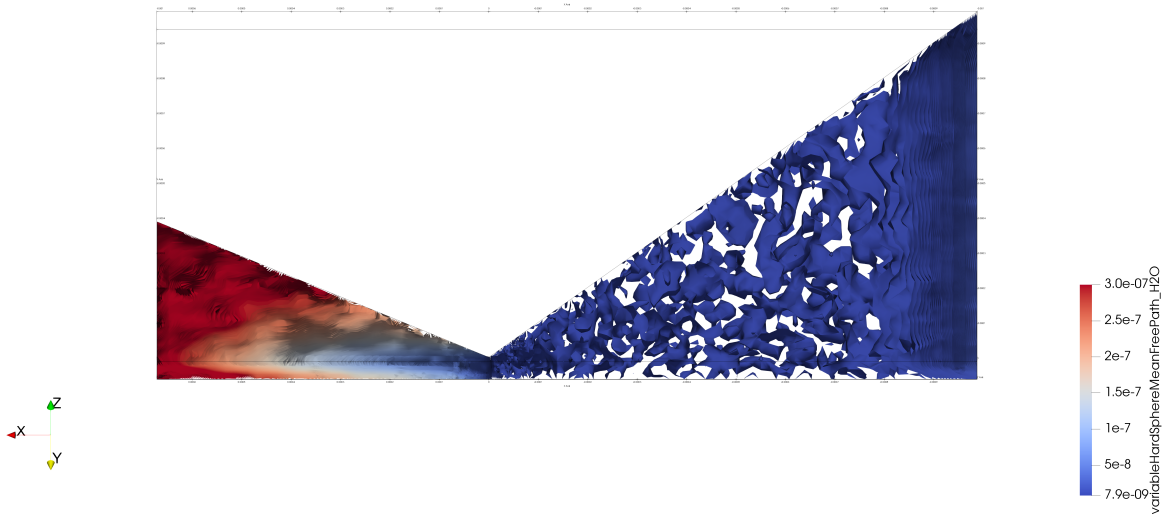



Figure 4.91: M1 nozzle variable hard sphere mean free path contour plot

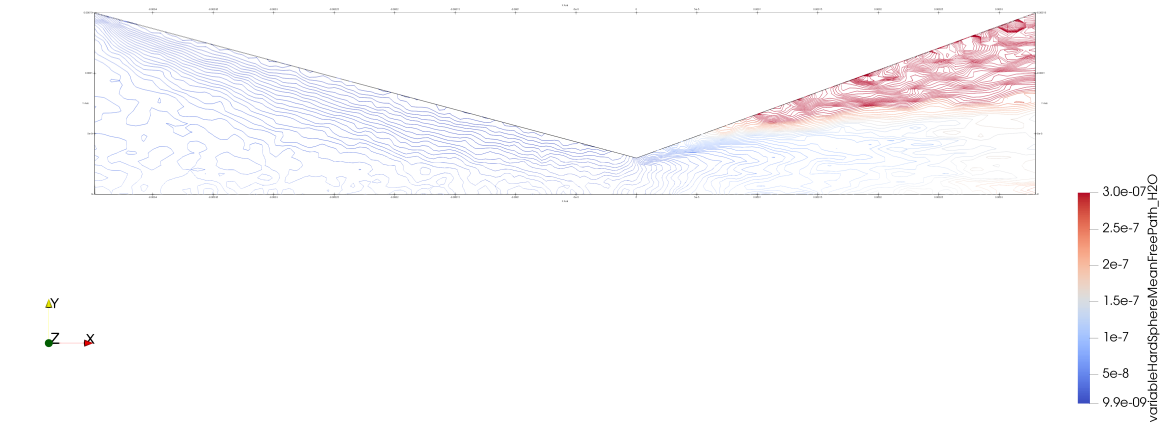



Figure 4.92: C2 nozzle variable hard sphere mean free path contour plot

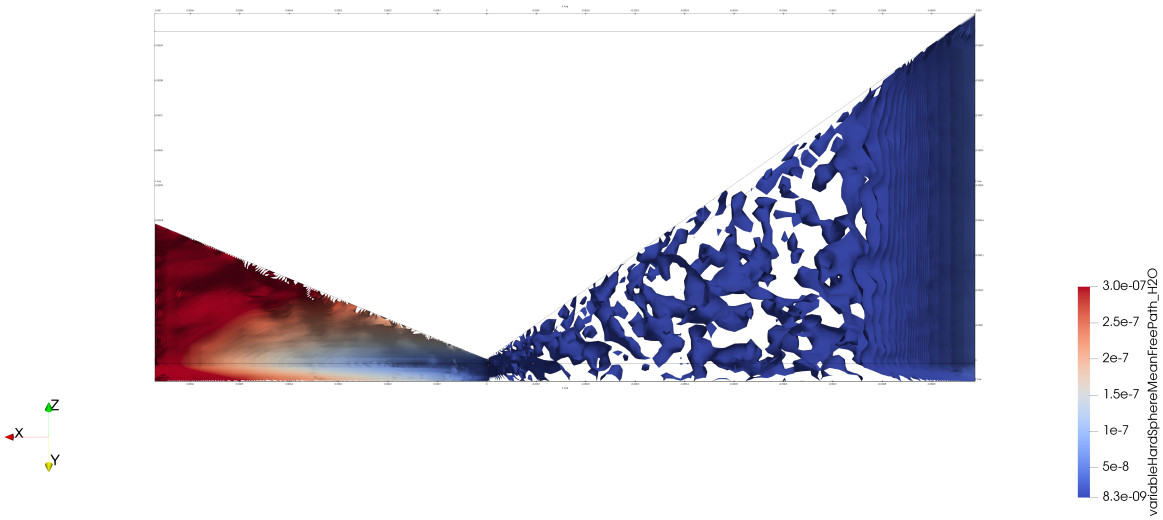



Figure 4.93: M2 nozzle variable hard sphere mean free path contour plot

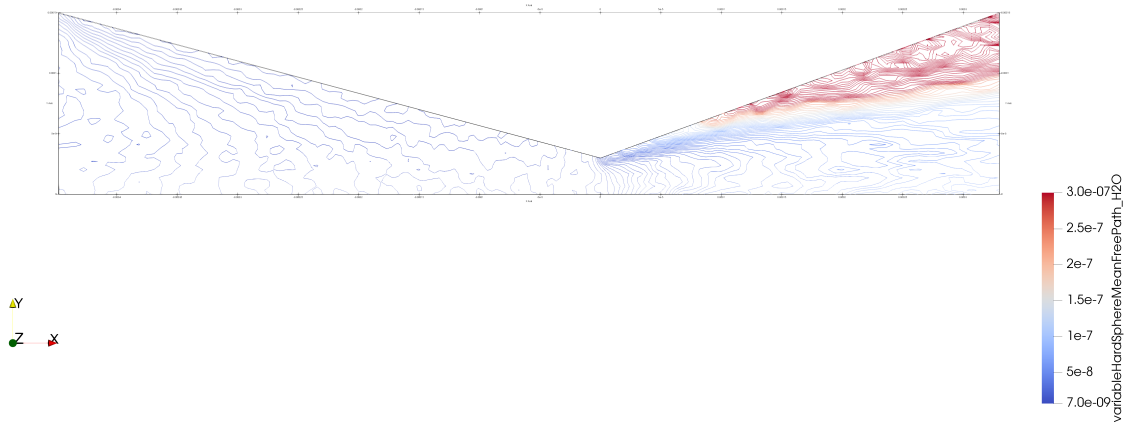



Figure 4.94: C3 nozzle variable hard sphere mean free path contour plot

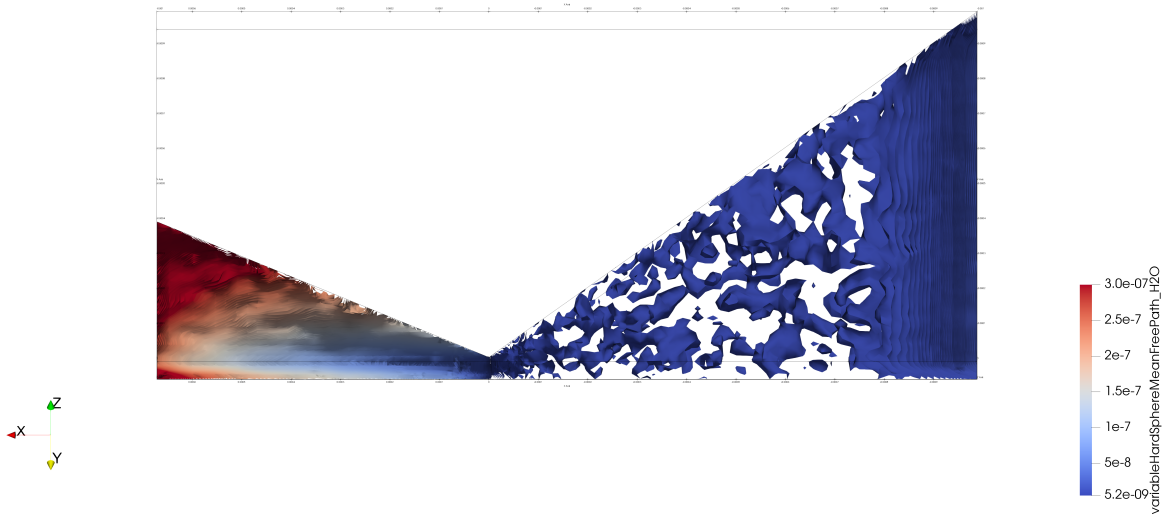



Figure 4.95: M3 nozzle variable hard sphere mean free path contour plot

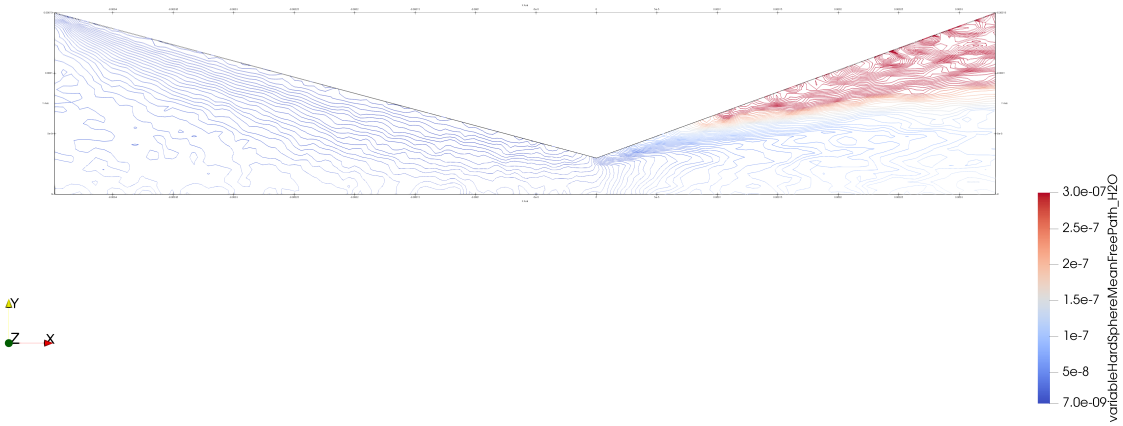



Figure 4.96: C4 nozzle variable hard sphere mean free path contour plot

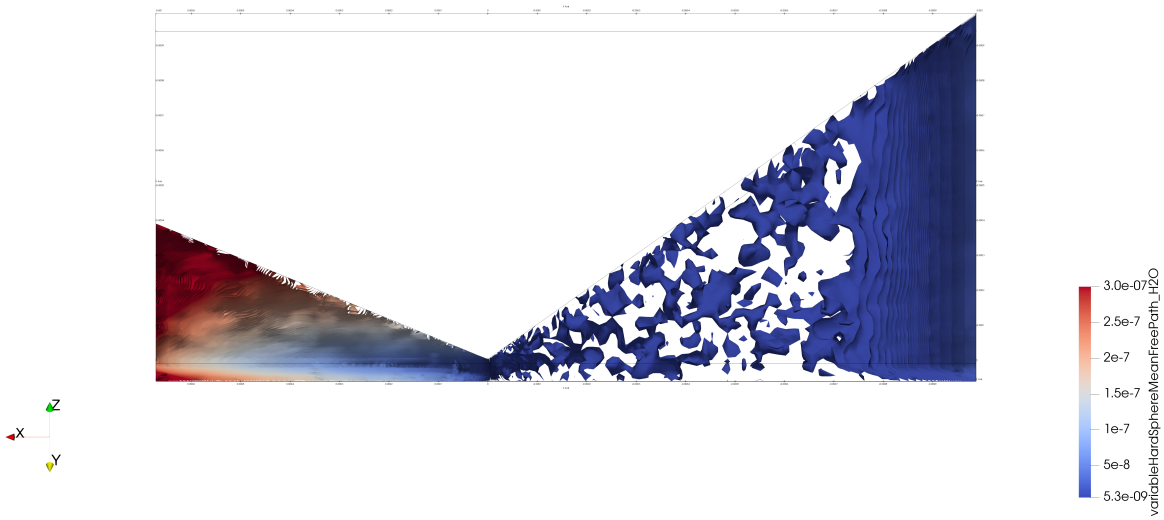



Figure 4.97: M4 nozzle variable hard sphere mean free path contour plot

### 4.1.14. Centerline Data

Figures 4.98, 4.99, 4.100, and 4.101 show the nozzle centerline pressure, temperature, axial velocity, and $Ma$ respectively vs. $x$ for all simulations at the final time step. The data is obtained using ParaView's Plot Over Line Filter with the conventional (probed along X Axis with Point2 at 0.00032969) and MEMS (probed along X Axis with Point2 at 0.00067116) nozzles until right before the outlet using data from Table 4.1 (ParaView Section 3.4). Note that since the conventional nozzle is simulated as a wedge due to axial symmetry, the Y and Z coordinates for both Point1 and Point2 are set to 0 (at the centerline). Afterwards, Save Data is used with the Files of Type Comma or Tab Delimited Files(*.csv *.tsv *.txt).

The purpose of the centerline data is to obtain differently interpretable results, which could also be more realistically expectable depending on the approach considered. While referring to the Pressure and Temperature (4.1.4), Velocity, Root Mean Square Speed, and Most Probable Speed (4.1.5), and Mach Number (4.1.9) Subsections with the axial averaged data, all profiles still follow comparable trends. However, it is clear that the temperature and velocity arching effects on the MEMS nozzle are less effective at the centerline along with that the inlet temperature is closer to the set temperature, as the wall temperatures are lower than the set inlet temperatures, where this effect is actually amplified for the MEMS nozzle due to its enhanced heat transfer geometrical properties. Also, the MEMS nozzle's temperature and (axial) velocity arches' peaks are delayed more towards the outlet mainly due to the delayed reach of the boundary layer (and shockwaves) at the centerline. The (axial) velocity and Mach numbers at the centerline are considerably higher than when using axial averaged data, where the Mach number of the MEMS nozzle actually becomes sonic at the throat and briefly continues to accelerate to supersonic speeds afterwards. It actually only slightly drops below Mach 1 for the MEMS nozzle simulations at lower pressures due to the velocity arch, considering its different geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)), where it becomes relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5 and Boundary Layers and Rarefaction Phenomena Subsection 4.1.13). This reaffirms that there is incomplete expansion in the MEMS nozzle.

### 4.1.15. Transient Solution (Animation Videos)

As explained in ParaView Section 3.4, the animation videos (at 3 fps using foamCreateVideo) are created with their final time step images shown in Figures 4.108 to 4.115, 4.116 to 4.123, 4.124 to 4.131, and 4.132 to 4.139 for the nozzle and plume region temperature, pressure, velocity, and density respectively with each group containing figures of C1, M1, C2, M2, C3, M3, C4, and M4 respectively. Also, lagrangian/dsmc is unselected from Mesh Regions for both conventional and MEMS nozzles and Z and Y Normal Clip Filters are applied for the MEMS nozzle to visualize inside the domain. The range is set for the final time step and the temperature, pressure, and velocity are colored by cells, while the mass density is colored by points. In the Color Map Editor, the use log scale when mapping data to colors is chosen for the pressure and mass density to show the largely varying data, while the jet present is chosen for the velocity. To note, very small negative value minima are negligibly shown on the color legend in the computational domains for temperature most likely due to numerical inaccuracies. Note that the time steps for the video are not consecutive as set (at 5e-10 with writeInterval of 5e-8), as the data is saved from the dynamic load balancing and reconstructPar (after the third 24 hour run for MEMS nozzle simulations as a backup and at the end of all simulations for the new time steps as applicable), which means that the dynamic load balancing can automatically save the writeInterval time steps with more significant changes considering that it does that when there is sufficient (as set) parallel load imbalance detected (especially for the MEMS nozzle with higher maximumAllowableImbalance, as the conventional nozzle's saved time steps are much more consecutive with the lower set maximumAllowableImbalance) (refer to decomposeParDict, balanceParDict, and loadBalanceDict Subsection 3.2.10).

Although the transient solution is deemed less significant as steady state is achieved within microseconds, considering that the typical response time of these thrusters is in milliseconds (Steady

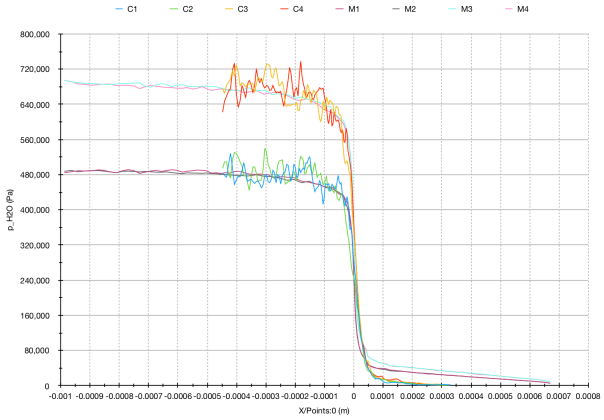



Figure 4.98: Nozzle centerline pressure vs. $x$ for all simulations

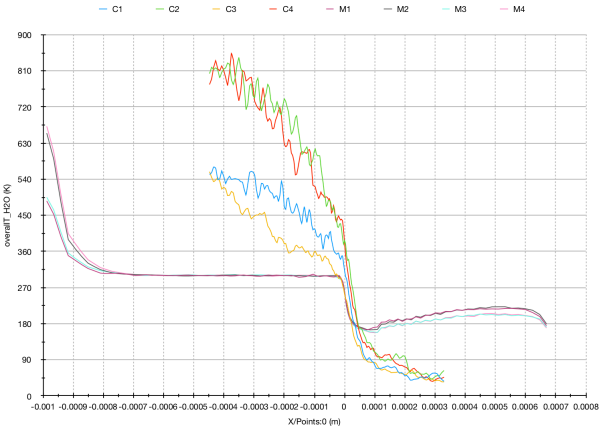



Figure 4.99: Nozzle centerline temperature vs. $x$ for all simulations

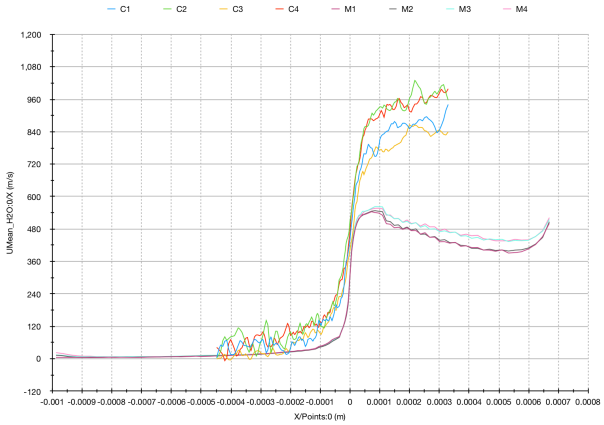



Figure 4.100: Nozzle centerline axial velocity vs. $x$ for all simulations

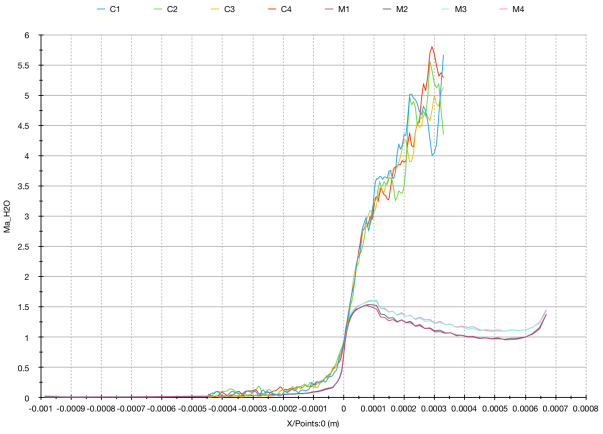



Figure 4.101: Nozzle centerline $Ma$ vs. $x$ for all simulations

State Convergence and General Final Simulation Data Subsection 4.1.1), it is still investigated in this subsection. The temperature profiles prove the MEMS nozzle wall's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4), as it is relatively greater at the lateral sides due to the closeness with the walls. The pressure and mass density (except for the temperature stagnation region of the converging section caused by the wall temperature with the MEMS nozzle's enhanced heat transfer (Pressure and Temperature Subsection 4.1.4)) profiles are respectively comparable. The velocity profiles for the conventional nozzle generally extend to the outlet from the throat maintaining the shape of the nozzle with lower values at the diverging section's sides, while the MEMS nozzle's velocity profiles have more of a straight shape from the throat to the outlet (before further expansion), which leads to a relatively greater velocity at the sides along with the front, considering that the MEMS nozzle flow's different geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) make it become relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5) (see Boundary Layers and Rarefaction Phenomena Subsection 4.1.13). Therefore, the flow appears to mainly separate earlier and more so for the MEMS nozzle compared to the conventional nozzle with a better performance. Furthermore, the quasi-2D MEMS nozzle with a larger diverging half angle (Table 3.1) appears to spread the flow at a wider angle from the outlet compared to the conical 3D conventional nozzle, which could have a greater impact on the

outer walls (relatively lower performance, possible heat transfer, greater mass, and propellant contamination, though water is not quite corrosive, flammable, or toxic, but freezing might pose a problem). For more information on the steady state solution, see Pressure and Temperature Subsection 4.1.4, Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5, Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2, and Boundary Layers and Rarefaction Phenomena Subsection 4.1.13.

To briefly discuss the transient solution, considering some of the more interesting flows' behaviors, Figures 4.102 and 4.103, 4.104 and 4.105, and 4.106 and 4.107 respectively show snapshots of the transient solution represented differently before, during, and after the backward forming shock diamonds for different conventional and MEMS nozzle cases. To explain, very quick (within a fraction of a microsecond) backward forming shock diamonds starting from the sides and taking the overall shape of the respective nozzle is detected in both conventional (Figure 4.104) and MEMS (Figure 4.105) nozzles' converging sections after firing, as the accumulation at the sides causes an increased (back) pressure at the throat (where some of the kinetic energy is converted back into thermal energy momentarily within less than microseconds, which could still raise the need for material strength considerations) after the typical converging section de Laval nozzle behavior (temperature and pressure drop as well as velocity rise) (Figures 4.102 and 4.103) starting from the initially vacuum (relatively larger pressure gradient) domain with the MEMS nozzle relatively less affected by its enhanced wall heat transfer (Pressure and Temperature Subsection 4.1.4) and velocity arch (Boundary Layers and Rarefaction Phenomena Subsection 4.1.13). Also considering the inlet pressure boundary condition used (boundariesDict Subsection 3.2.5), a second wave of higher pressure and temperature flow coming from the inlet is noticeable afterwards in the conventional nozzle (at the inlet in Figure 4.106), which is not exactly seen in the MEMS nozzle (Figure 4.107) (as found in Steady State Convergence and General Final Simulation Data 4.1.1), before the higher pressure at the throat is fully balanced out (especially typically starting from the sides as can be seen in the figures) throughout the nozzle and steady state is reached. Note that the thrusters are assumed to be firing when the gas is fully at the desired temperature and pressure, which might not be exact as a transient solution representation, considering that the flow starts to develop from way back while passing the resistive microheaters section in application. Notice that the heat transfer is still enhanced at the lateral sides in the converging section, though its effect is diminished by the accumulation at the beginning (Boundary Layers and Rarefaction Phenomena Subsection 4.1.13). For the simulations' equivalent time for assumed estimated steady state convergence time steps, refer to Table 4.2 in the Steady State Convergence and General Final Simulation Data Subsection 4.1.1.

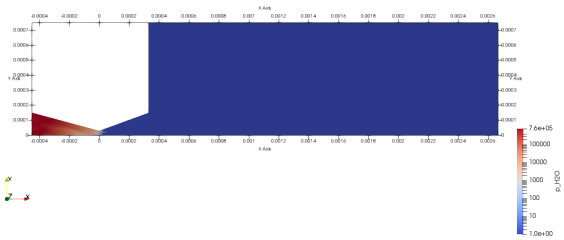


Figure 4.102: C3 nozzle and plume region pressure at 1.5e-7 s

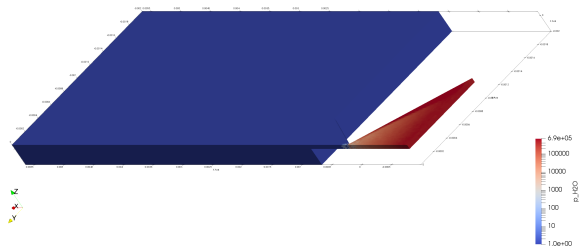


Figure 4.103: M1 nozzle and plume region pressure at 4e-7 s

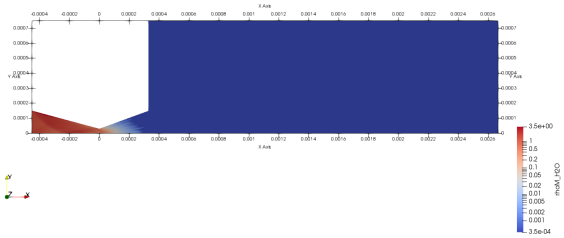


Figure 4.104: C2 nozzle and plume region mass density at 2e-7 s

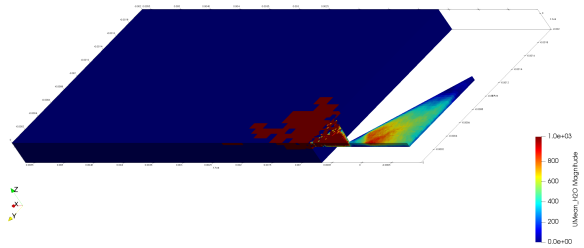


Figure 4.105: M1 nozzle and plume region velocity at 9e-7 s

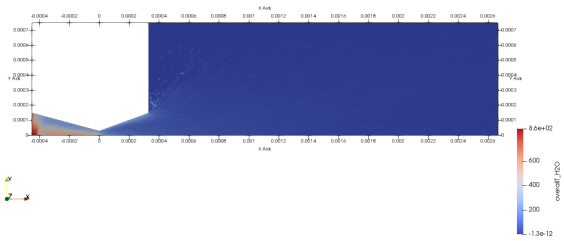


Figure 4.106: C4 nozzle and plume region temperature at 1.45e-6 s

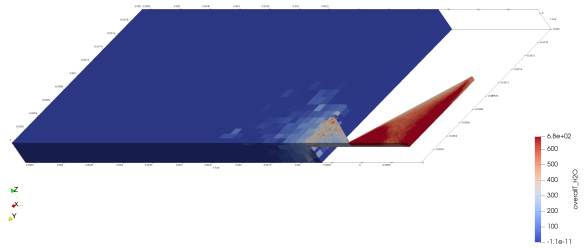


Figure 4.107: M4 nozzle and plume region temperature at 1.3e-6 s

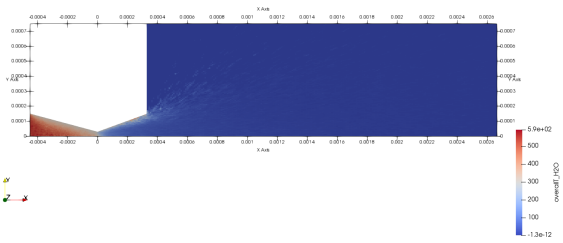


Figure 4.108: C1 nozzle and plume region temperature at final time step

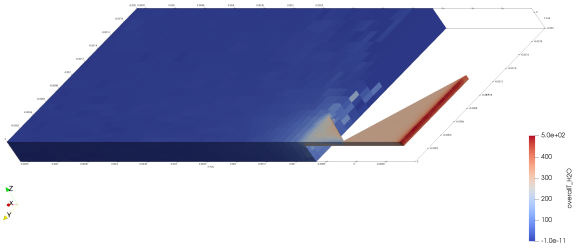


Figure 4.109: M1 nozzle and plume region temperature at final time step

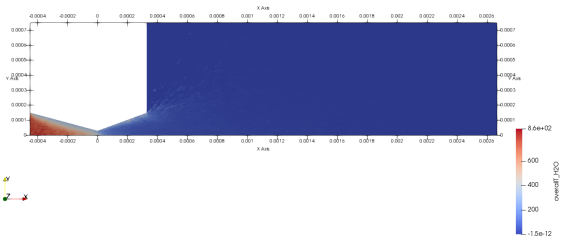


Figure 4.110: C2 nozzle and plume region temperature at final time step

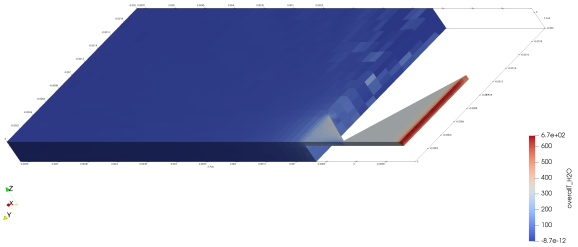


Figure 4.111: M2 nozzle and plume region temperature at final time step

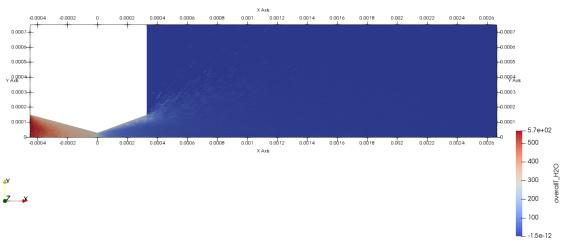


Figure 4.112: C3 nozzle and plume region temperature at final time step

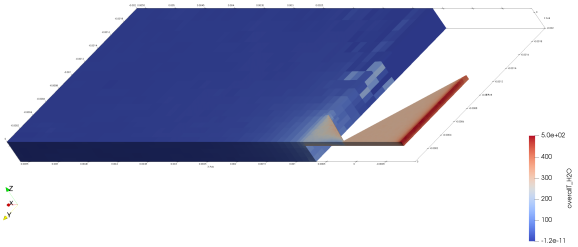


Figure 4.113: M3 nozzle and plume region temperature at final time step

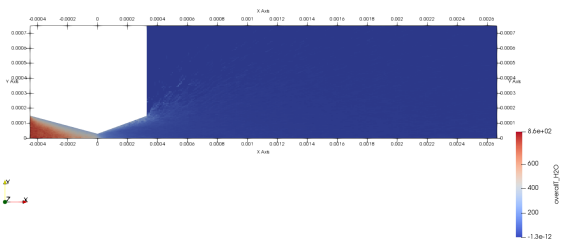


Figure 4.114: C4 nozzle and plume region temperature at final time step

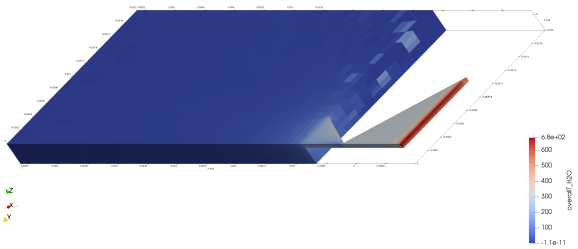


Figure 4.115: M4 nozzle and plume region temperature at final time step

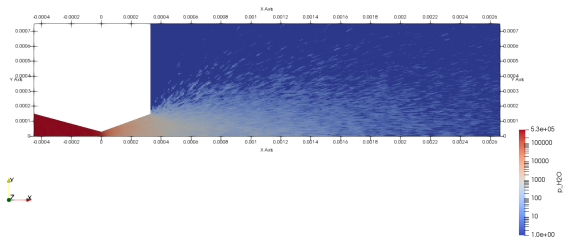


Figure 4.116: C1 nozzle and plume region pressure at final time step

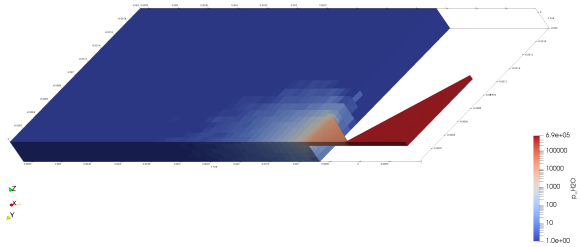


Figure 4.117: M1 nozzle and plume region pressure at final time step

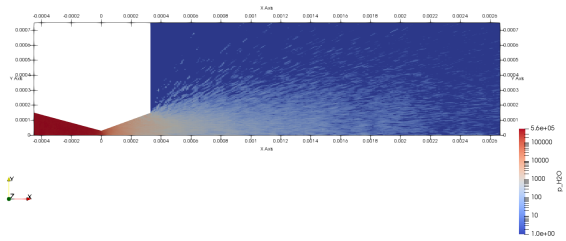


Figure 4.118: C2 nozzle and plume region pressure at final time step

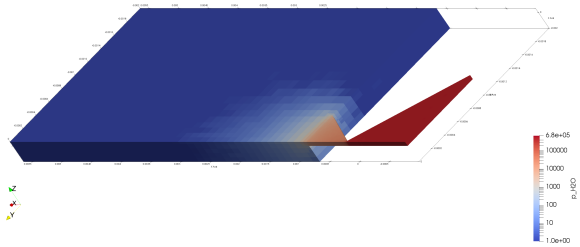


Figure 4.119: M2 nozzle and plume region pressure at final time step

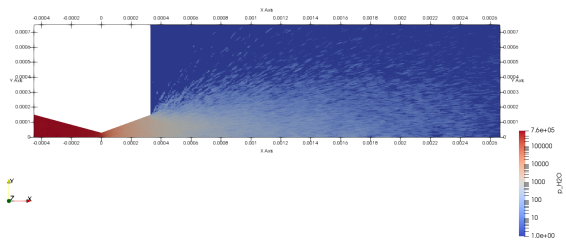


Figure 4.120: C3 nozzle and plume region pressure at final time step

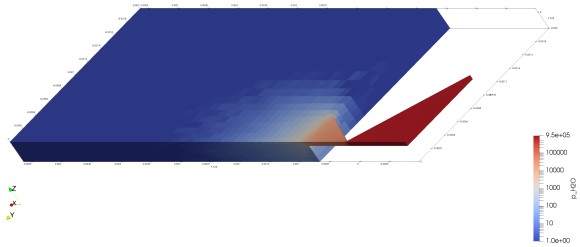


Figure 4.121: M3 nozzle and plume region pressure at final time step

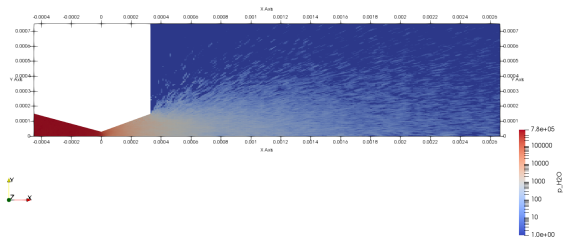


Figure 4.122: C4 nozzle and plume region pressure at final time step

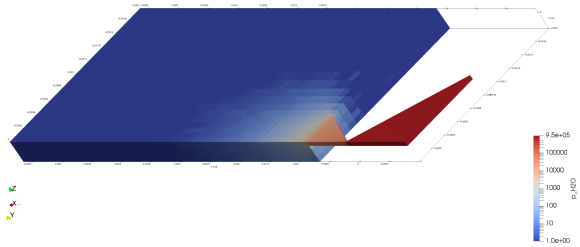


Figure 4.123: M4 nozzle and plume region pressure at final time step

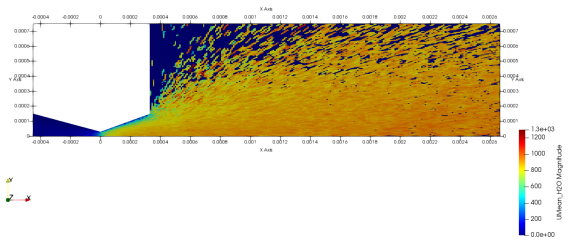


Figure 4.124: C1 nozzle and plume region velocity at final time step

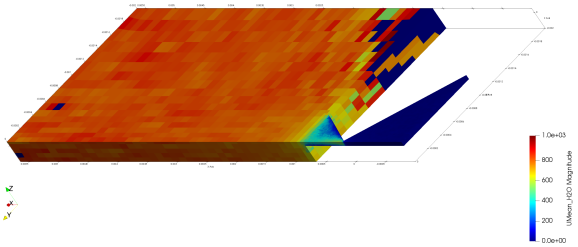


Figure 4.125: M1 nozzle and plume region velocity at final time step

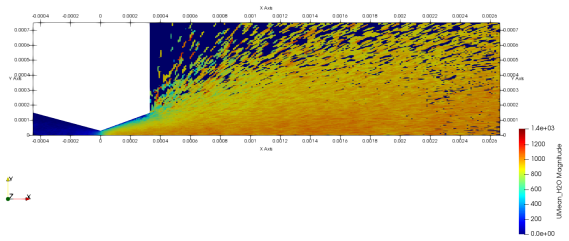


Figure 4.126: C2 nozzle and plume region velocity at final time step

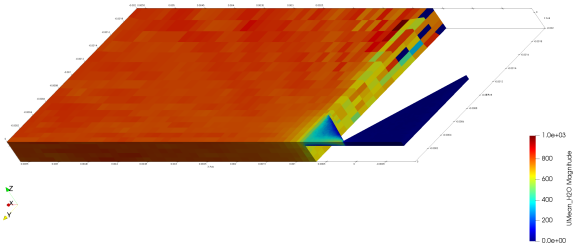


Figure 4.127: M2 nozzle and plume region velocity at final time step

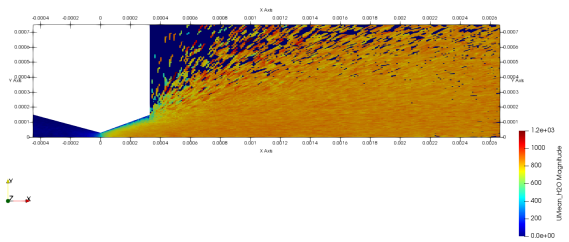


Figure 4.128: C3 nozzle and plume region velocity at final time step

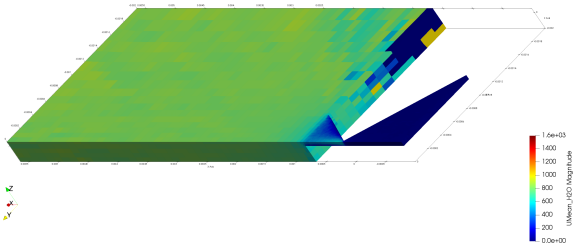


Figure 4.129: M3 nozzle and plume region velocity at final time step

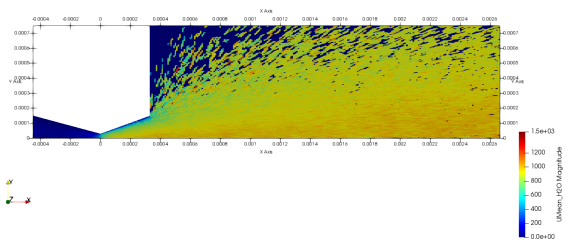


Figure 4.130: C4 nozzle and plume region velocity at final time step

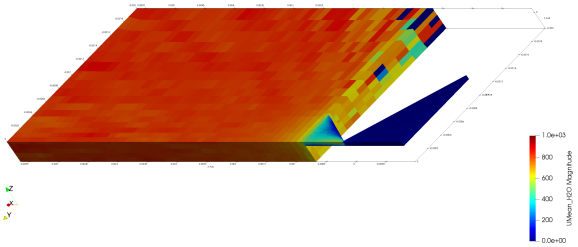


Figure 4.131: M4 nozzle and plume region velocity at final time step

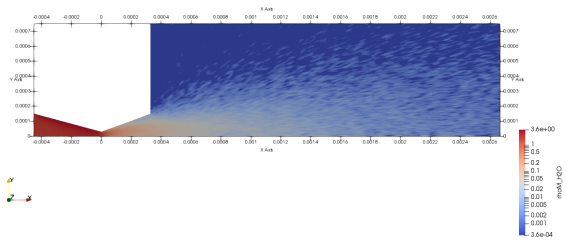


Figure 4.132: C1 nozzle and plume region mass density at final time step

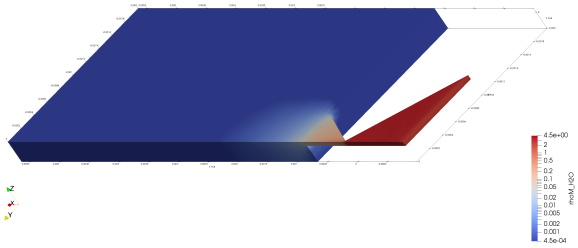


Figure 4.133: M1 nozzle and plume region mass density at final time step

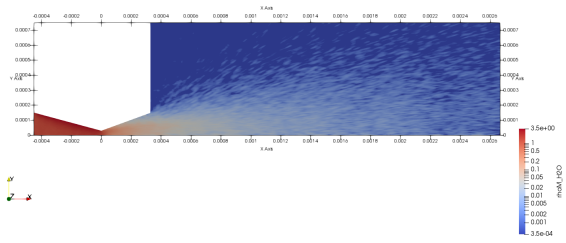


Figure 4.134: C2 nozzle and plume region mass density at final time step

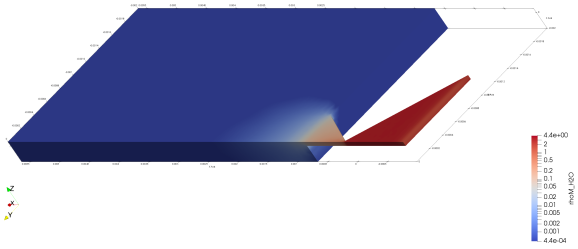


Figure 4.135: M2 nozzle and plume region mass density at final time step

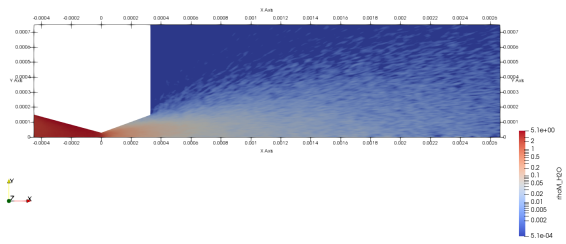


Figure 4.136: C3 nozzle and plume region mass density at final time step

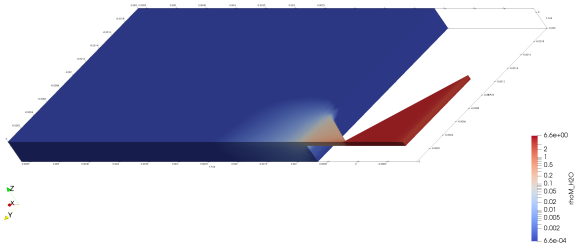


Figure 4.137: M3 nozzle and plume region mass density at final time step

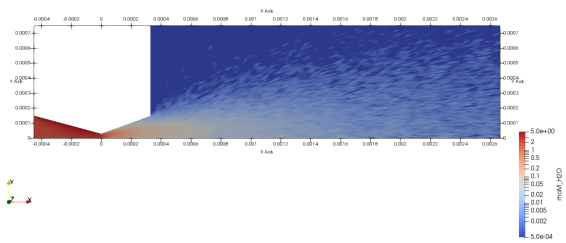


Figure 4.138: C4 nozzle and plume region mass density at final time step

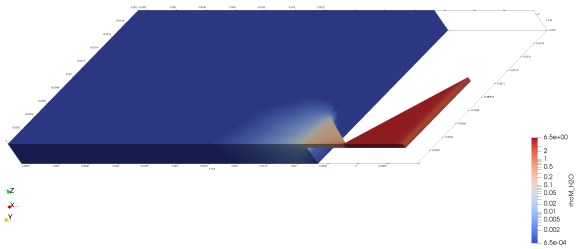


Figure 4.139: M4 nozzle and plume region mass density at final time step

## 4.2. DSMC vs. Continuum Modeling

The final DSMC results are compared to the results obtained when taking a more conventional approach to solving the problem using continuum flow modeling.

### 4.2.1. Analytical Model

An analytical analysis has been conducted for the both MEMS and conventional nozzle geometries from Table 3.1 at all considered cases of temperatures and pressures to mainly estimate the throat Reynolds number and ideal vacuum thrust with no losses using basic simplified equations. A MATLAB code provided in Subsection B.1.1 is created to obtain the results, which are based on equations from Section 2.6.

As shown in the MATLAB code (Subsection B.1.1) for all models, the required quantities needed for the calculations are input, as the areas of the inlet, throat, and exit are calculated. The density, dynamic viscosity, and specific heat capacities at constant pressure and volume are obtained for water at the selected inlet pressure and temperature from the US National Institute of Standards and Technology's (NIST) publicly accessible database of thermodynamic data. The NIST database is used to compare the variation of the thermodynamic properties for the decrease in temperature and pressure along the nozzle as expected with the increase of Mach number in Equations 2.109 and 2.110, as the equations apply along the nozzle and not just at the exit, and it has been determined that using inlet values throughout the converging part of the nozzle for the properties obtained using the NIST database to roughly evaluate the analytical model is an invalid assumption, as compressibility effects must be considered near the throat considering that the Mach number becomes close to one, where the density change becomes close to the velocity change. Then, the ratio of specific heats (Equation 2.107), Vandenkerckhove function (Equation 2.103), and the mass flow rate along the nozzle (Equation 2.102) are sequentially calculated from inlet values.

The inlet velocity can then be calculated inversely using Equation 2.102 with the density and area at the inlet already specified. Next, to consider compressibility effects on density in the converging section of the nozzle, Equations 2.109 to 2.111 are used to calculate the temperature, pressure, and density at the throat using $M = 1$, as done for the choked mass flow rate equation used. Then, a MATLAB wrapper for CoolProp, which is a C++ library for thermodynamic data, is used for readily extracting thermodynamic properties, so the density (reevaluated to double-check), dynamic viscosity, and specific heat capacities at constant pressure and volume at the throat are found leading to the ratio of specific heats (Equation 2.107), speed of sound (Equation 2.100), velocity (Equation 2.102), and ultimately the Reynolds number (Equation 2.1) at the throat, with the throat height as the characteristic linear dimension for the conventional nozzle and the hydraulic diameter equaling four times the cross sectional area divided by the cross sectional wetted perimeter, considering that it reduces to an equivalent diameter for a circular cross section, for the MEMS nozzle.

To calculate the thrust, Equation 2.108 is inversely applied using the expansion ratio to calculate the Mach number at the exit. To note, a symbolical solution is attempted in solving the equation inversely, though in some cases it is unable to solve it symbolically and switches to find a numerical solution where it might return an incorrect value, so this is manually spotted and fixed to expect the solution within a correct range as a supersonic Mach number. Then, the exit temperature (Equation 2.109), pressure (Equation 2.110), and speed of sound (Equation 2.100) are calculated. Since it is difficult to find thermodynamic data for the resulting exit temperature and the specific heat ratio is not expected to vary significantly for the considered analysis, as proven by the difference between inlet and throat specific heat ratios even though their difference with the exit specific heat ratio is expected to be greater, the specific heat ratio at the inlet is still used. Afterwards, the exit velocity can be computed using Equation 2.99 using the Mach number and speed of sound along with Equation 2.105 mainly using the exit and inlet pressure ratio and inlet temperature, which result in equivalent values as a confirmation of the used formulas. Lastly, the thrust is computed using the basic rocket thrust Equation 2.98 to also lead to the specific impulse (Equation 2.92) with the calculated mass flow rate and Earth's gravity as 9.81 $\frac{m}{s^2}$.

Furthermore, the mean free path (Equation 2.6 with a water molecular diameter of $2.75 \cdot 10^{-10}$ m and Avogadro constant of $6.022140857 \cdot 10^{23}$ mol$^{-1}$) is calculated at the inlet, throat, and exit followed by the respective the Knudsen number (Equation 2.5 with the same respective characteristic linear dimensions used for the throat Reynolds number). The $DN$ and $VDN$ analysis assumes basic prior

understanding of the introduced Extended Continuum/Kinetic Dimensionless Numbers for Diffusivity and Rarefaction Intensity in Appendix A. The root mean square speed (Equation 4.5) at the inlet, throat, and outlet is obtained leading to the respective $VDN$ (Equation A.12) and Mach number using $VDN$ (Equation A.15 with the inlet specific heat ratio as explained above). It is compared to the conventionally determined respective Mach numbers (Equation 2.99), where the speed of sound (Equation 2.100) is also calculated at the inlet. The inlet, throat, and exit $DN$ (Equation A.10 with the same respective characteristic linear dimensions used for the throat Reynolds number) values are also determined as the ratio of advective transport rate to intrinsic diffusive transport rate before using them to find the respective Reynolds numbers (Equation A.9) and comparing them with the conventionally determined respective Reynolds numbers (Equation 2.1 with the same respective characteristic linear dimensions used for the throat Reynolds number and the inlet dynamic viscosity for similar reasons for the usage of the inlet specific heat ratio as explained above, even though the difference is expected to be greater, along with its usage in the DSMC model results). Note that the the exit Reynolds number uses the exit density determined from Equation 2.111 with the inlet specific heat ratio as explained above. The main results are shown below in Table 4.5 and the complete MATLAB Workspace results along with the code are available in Subsection B.1.2.

Table 4.5: Analytical model general results

| | *Chamber Pressure = 5 bar Chamber Temperature = 550 K* | | *Chamber Pressure = 5 bar Chamber Temperature = 773 K* | | *Chamber Pressure = 7 bar Chamber Temperature = 550 K* | | *Chamber Pressure = 7 bar Chamber Temperature = 773 K* | |
|---|---|---|---|---|---|---|---|---|
| | **MEMS Nozzle** | **Conventional Nozzle** | **MEMS Nozzle** | **Conventional Nozzle** | **MEMS Nozzle** | **Conventional Nozzle** | **MEMS Nozzle** | **Conventional Nozzle** |
| **Mass Flow Rate (mg/s)** | 1.6645 | 1.8825 | 1.3894 | 1.5713 | 2.3349 | 2.6408 | 1.9462 | 2.2011 |
| **Inlet Velocity (m/s)** | 4.1646 | 13.3267 | 4.9377 | 15.8005 | 4.1481 | 13.2738 | 4.9330 | 15.7856 |
| **Throat Velocity (m/s)** | 529.7669 | 529.7669 | 630.6352 | 630.6352 | 527.2577 | 527.2577 | 629.9085 | 629.9085 |
| **Throat Speed of Sound (m/s)** | 539.1125 | 539.1125 | 635.6329 | 635.6329 | 539.7785 | 539.7785 | 635.8076 | 635.8076 |
| **Throat Reynolds Number** | 1645.8747 | 2468.8120 | 902.3439 | 1353.5159 | 2323.4565 | 3485.1848 | 1265.3402 | 1898.0103 |
| **Exhaust Velocity (m/s)** | 1281.7170 | 1266.4486 | 1553.9678 | 1533.4452 | 1276.3645 | 1261.4611 | 1552.1420 | 1531.7506 |
| **Exhaust Mach Number** | 4.7740 | 4.5256 | 4.5358 | 4.3150 | 4.8217 | 4.5677 | 4.5476 | 4.3255 |
| **Thrust (mN)** | 2.2043 | 2.4723 | 2.2409 | 2.5105 | 3.0768 | 3.4515 | 3.1345 | 3.5119 |
| **Specific Impulse (s)** | 134.9964 | 133.8721 | 164.4151 | 162.8660 | 134.3234 | 133.2313 | 164.1816 | 162.6444 |
| **Inlet Knudsen Number** | 2.3730e-04 | 1.5067e-04 | 3.3352e-04 | 2.1176e-04 | 1.6950e-04 | 1.0762e-04 | 2.3823e-04 | 1.5126e-04 |
| **Throat Knudsen Number** | 0.001797 | 0.001199 | 0.002536 | 0.001690 | 0.001282 | 8.5498e-04 | 0.001811 | 0.001207 |
| **Exit Knudsen Number** | 0.03085 | 0.01411 | 0.04481 | 0.02047 | 0.02190 | 0.01002 | 0.03195 | 0.01460 |
| **Inlet $VDN$** | 0.004772 | 0.01527 | 0.004773 | 0.01527 | 0.004753 | 0.01521 | 0.004768 | 0.01526 |
| **Inlet Mach Number Using $VDN$** | 0.007194 | 0.02302 | 0.007303 | 0.02337 | 0.007145 | 0.02286 | 0.007290 | 0.02333 |
| **Inlet Mach Number** | 0.007194 | 0.02302 | 0.007303 | 0.02337 | 0.007145 | 0.02286 | 0.007290 | 0.02333 |
| **Inlet $DN$** | 20.1109 | 1.0136e+02 | 14.3104 | 72.1246 | 28.0434 | 1.4134e+02 | 20.0157 | 1.0088e+02 |
| **Inlet Reynolds Number Using $DN$** | 43.6568 | 2.2003e+02 | 31.0651 | 1.5657e+02 | 60.8766 | 3.0682e+02 | 43.4501 | 2.1899e+02 |
| **Inlet Reynolds Number** | 82.2694 | 4.1464e+02 | 46.3094 | 2.3340e+02 | 1.1558e+02 | 5.8252e+02 | 64.8617 | 3.2690e+02 |
| **Throat $VDN$** | 0.6539 | 0.6539 | 0.6511 | 0.6511 | 0.6518 | 0.6518 | 0.6506 | 0.6509 |
| **Throat Mach Number Using $VDN$** | 0.9857 | 0.9857 | 0.9962 | 0.9962 | 0.9798 | 0.9798 | 0.9947 | 0.9947 |
| **Throat Mach Number** | 0.9827 | 0.9827 | 0.9921 | 0.9921 | 0.9768 | 0.9768 | 0.9907 | 0.9907 |
| **Throat $DN$** | 3.6391e+02 | 5.4586e+02 | 2.5677e+02 | 3.8516e+02 | 5.0827e+02 | 7.6240e+02 | 3.5930e+02 | 5.3895e+02 |
| **Throat Reynolds Number Using $DN$** | 7.8997e+02 | 1.1850e+03 | 5.5740e+02 | 8.3610e+02 | 1.1033e+03 | 1.6550e+03 | 7.7996e+02 | 1.1699e+03 |
| **Exit $VDN$** | 3.1670 | 3.0022 | 2.9644 | 2.8201 | 3.2077 | 3.0387 | 2.9744 | 2.8291 |
| **Exit Mach Number Using $VDN$** | 4.7740 | 4.5256 | 4.5358 | 4.3150 | 4.8217 | 4.5677 | 4.5476 | 4.3255 |
| **Exit $DN$** | 1.0265e+02 | 2.1273e+02 | 66.1515 | 1.3775e+02 | 1.4646e+02 | 3.0323e+02 | 93.0844 | 1.9379e+02 |
| **Exit Reynolds Number Using $DN$** | 2.2284e+02 | 4.6179e+02 | 1.4360e+02 | 2.9903e+02 | 3.1794e+02 | 6.5826e+02 | 2.0207e+02 | 4.2067e+02 |
| **Exit Reynolds Number** | 43.8193 | 84.1330 | 24.4058 | 46.8592 | 61.9298 | 1.18905e+02 | 34.2342 | 65.7298 |

Based on the throat Reynolds number data in Table 4.5, the throat flow is transient (transitional flow between laminar and turbulent with $2300 < Re < 4000$) for the conventional nozzle at 550 K and both pressures along with the MEMS nozzle at 550 K and 7 bar, while laminar ($Re < 2300$) for all other cases considered, as the viscosity of the gas generally increases with increasing temperature due to the increase in molecular collisions in the gas, contrary to the decrease of a liquid viscosity with an increase in temperature, as it decreases the dominant cohesive force between the liquid molecules. Another combined reason for the transient flow in the mentioned cases is the density increase with lower temperature. Viscous losses are more significant in MEMS nozzles compared to conventional nozzles generally and higher temperature and lower pressure nozzles specifically, as the Reynolds numbers are lower in these cases indicating the dominance of viscous forces (which is a negative sign in CD nozzles). At lower Reynolds numbers especially in quasi-2D nozzle flows in MEMS devices, the 3D endwall (flat walls that cap the side flow channel walls) effects could cause considerable inaccuracies using 2D CFD simulations, as shown in [11].

The Knudsen number remains slightly rarefied within the slip flow regime at the outlet, while at continuum flow with normal density levels at the inlet. This presents a significant challenge to simulate using DSMC, as the number of particles behind and beyond the throat needs to be optimized. $DN$ and $VDN$ also show consistent results compared to the Reynolds and Mach numbers respectively, though the deviation becomes relatively greater at the outlet for $DN$, where thermodynamic data may become

more difficult to obtain for the Reynolds number, so assumptions have to be made. This makes using $DN$ an attractive kinetic theory solution for the Reynolds number with possibly enhanced accuracy in certain situations within its assumptions.

### 4.2.2. TU Delft VLM CFD Continuum Model

As the ideal rocket theory does not provide accurate insight into the performance of the thruster due to phase change, computational fluid dynamics (CFD) simulations were done for different nozzles using ANSYS Fluent [16]. An SST $k-\omega$ model with low Reynolds numbers and compressibility effects corrections was applied. Figure 4.140 shows the four considered nozzle options for the CFD simulation [16].

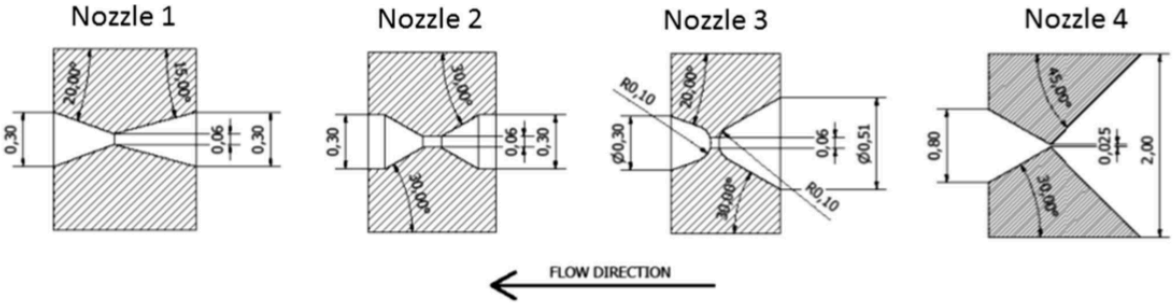



Figure 4.140: Nozzle geometries considered using ANSYS Fluent in mm. Only Nozzle 4 is a slit 2D nozzle with rectangular section length of 0.1 mm and relatively higher expansion ratio of 32 compared to 25 in the other three axisymmetric nozzles [16].

Table 4.6 shows the ANSYS Fluent CFD results of the different VLM nozzles at a chamber temperature of 550 K, where the expansion occurs to vacuum [16]. The one-dimensional ideal rocket theory provides the discharge coefficient, which is the ratio of actual to ideal mass flow rate, and specific impulse quality, which is the ratio of actual specific impulse to ideal specific impulse [16]. The results indicate that Nozzle 1 (also used as DSMC conventional nozzle) provides best performance at ideal conditions considering the specific impulse quality, where the other nozzles show lower performance, even though Nozzle 4 (also used as DSMC MEMS nozzle) has a higher expansion ratio.

Table 4.6: ANSYS Fluent results for VLM nozzles with water propellant at chamber temperature of 550 K [16]

| *At Chamber Pressure of 7 bar* | **Nozzle 1** | **Nozzle 2** | **Nozzle 3** | **Nozzle 4** | **Ideal** |
|---|---|---|---|---|---|
| **Mass Flow Rate** $\left(\frac{mg}{s}\right)$ | 2.65 | 2.43 | 2.18 | 2.28 | 2.62 |
| **Discharge Coefficient** | 1.010 | 0.926 | 0.831 | 0.869 | 1 |
| **Exhaust Velocity** $\left(\frac{m}{s}\right)$ | 1147.7 | 1107.1 | 1069.6 | 970.9 | 1328.0 |
| **Thrust (mN)** | 3.04 | 2.69 | 2.33 | 2.22 | 3.48 |
| **Specific Impulse (s)** | 117.0 | 112.9 | 109.0 | 99.0 | 135.4 |
| **Specific Impulse Quality (%)** | 86.4 | 83.3 | 80.5 | 73.1 | 100 |
| *At Chamber Pressure of 5 bar* | | | | | |
| **Mass Flow Rate** $\left(\frac{mg}{s}\right)$ | 1.88 | 1.72 | 1.55 | 1.63 | 1.87 |
| **Discharge Coefficient** | 1.003 | 0.918 | 0.827 | 0.870 | 1 |
| **Exhaust Velocity** $\left(\frac{m}{s}\right)$ | 1135.8 | 1092.8 | 1065.3 | 930.4 | 1328.0 |
| **Thrust (mN)** | 2.13 | 1.88 | 1.65 | 1.52 | 2.49 |
| **Specific Impulse (s)** | 115.8 | 111.4 | 108.6 | 94.9 | 135.4 |
| **Specific Impulse Quality (%)** | 85.5 | 82.3 | 80.2 | 70.0 | 100 |

The results showed that Nozzle 4 did not achieve the highest performance, yet it will still be considered as the main thruster considering its MEMS modular manufacturability. Table 4.7 shows the approximated thruster performance, where the set nozzle inlet pressure and temperature are equivalent to the heating chamber outlet values [16]. Overall, a chamber temperature of 773 K provides a higher specific impulse and lower required power, as the mass flow rate is lower at a higher temperature

and constant pressure [16]. The power transferred to water increases with increasing chamber pressure, which increases the mass flow rate leading to more required power for an equivalent increase in fluid temperature [16]. A maximum chamber pressure of 6 bar is required to completely vaporize the water using the set thruster properties, as vaporization uses over 60 % of the power transferred to water, which has a high latent heat of evaporation [16].

Table 4.7: Nozzle 4 microresistojet estimated performance at different propellant temperatures in heating chamber at pressure of 5 bar and propellant consumption totaling 50 g [16]

| | **Case 1** | **Case 2** |
|---|---|---|
| **Chamber Temperature (K)** | 550 | 773 |
| **Thrust (mN)** | 1.52 | 1.48 |
| **Mass Flow Rate** $\left(\frac{\text{mg}}{\text{s}}\right)$ | 1.63 | 1.36 |
| **Power Transferred to Water (W)** | 5.25 | 5.06 |
| **Specific Impulse (s)** | 94.9 | 111 |
| **Total Impulse (Ns)** | 46.4 | 54.7 |

To note, a current TU Delft VLM design also considers a nozzle expansion ratio $\left(\frac{A_e}{A^*}\right)$ of 11, chamber pressure ($P_c$) from 200 kPa to 500 kPa, and accordingly varied chamber temperature depending on the fluid, as for water at 200 kPa, it is 400-550 K with a mass flow rate of 1.56-1.33 $\left(\frac{\text{mg}}{\text{s}}\right)$ and thrust of 1.69 mN, and at 500 kPa, it is 430-550 K with a mass flow rate of 3.76-3.33 $\left(\frac{\text{mg}}{\text{s}}\right)$ and thrust of 4.22 mN [27]. The inlet microresistojet temperature is 283.16 K [27].

### 4.2.3. Discussion

A comparison between MEMS vs. Conventional Thrusters (DSMC) Section 4.1 and DSMC vs. Continuum Modeling Section 4.2's Analytical Model Subsection 4.2.1 and TU Delft VLM CFD Continuum Model Subsection 4.2.2 is conducted to provide an overall image of the different methodologies along with a final conventional vs. MEMS nozzle comparison. Mainly, the data discussed is retrieved from the DSMC and Analytical model general results Tables 4.4 and 4.5 respectively along with TU Delft VLM CFD Continuum Model Subsection 4.2.2's Tables 4.6 and 4.7.

For all models, the mass flow rate drops with increasing temperature at constant pressure and decreasing pressure at constant temperature. Although a general increase in thrust is seen with increasing temperatures (at constant pressures) and pressures (at constant temperatures) in the analytical model, the inverse applies for increasing temperatures at constant pressures in the DSMC and NS models. Note that the thrust variables are related to Equation 2.98. Furthermore, the specific impulse (Equation 2.92) increases with increasing temperatures for the analytical and DSMC models, while it decreases for the NS model. On the other hand, it decreases with higher pressures at constant temperatures for the analytical and DSMC models, while it increases for the NS model. This places the DSMC model's variation somewhat in between the analytical and NS models. Note that the data at the bottom of the TU Delft VLM CFD Continuum Model Subsection 4.2.2 follows general data trends. The $DN$, $VDN$, $Re$, and $Kn$ trends for the analytical and DSMC models are generally comparable, though $Kn$ and $VDN$ along with $Ma$ (at the exit as well as the velocity even compared to the NS model) are generally lower for the DSMC models, where especially the MEMS nozzle's wall temperature with enhanced heat transfer (Pressure and Temperature Subsection 4.1.4) causes a greater deviation from realistic expectations, along with $DN$ being higher at the throat. Furthermore, the MEMS nozzle's geometrical features (quasi-2D with smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow (Number of Simulation Particles per Grid Cell, Mass Density, Particle Number Density, Aspect Ratio, and Perimeter to Cross Sectional Area Ratio Subsection 4.1.2)) lead to its flow beyond the throat becoming relatively more rarefied at the lateral sides acting in a way like a channel with comparatively more faster particles traveling in straight line trajectories beaming out of the aperture center as rays considering the notably thick (cross sectional) boundary layer at the sides from the throat (viscous dissipation of flow kinetic energy from shear on the walls), ultimately leading to a slowdown at the front prompting its expansion (especially towards the sides) and following speedup as a velocity arch (Velocity, Root Mean Square Speed, and Most Probable Speed Subsection 4.1.5) (see Boundary Layers

and Rarefaction Phenomena Subsection 4.1.13).

Concerning the conventional and MEMS nozzle performances, the conventional nozzle has higher (better) thrust for all models, but its specific impulse is only higher (better) for the DSMC and NS models, which realistically portray a more accurate representation of the MEMS nozzle's geometrical limitations. This enables the conventional nozzle to surpass the MEMS nozzle, when only perceiving the end result. However, as proven by the MEMS nozzle's enhanced wall heat transfer in the DSMC model (even with a lower mass flow rate), the resistive microheaters embedded in the MEMS nozzle's geometry supplying the power needed to vaporize/heat the water are expected to be considerably more efficient (assuming proper insulation and depending on the resistive heaters' placement). Thus, there is a tradeoff, which needs to be further investigated in the heater section. It is also important to note that the performance of the MEMS nozzle is more negatively affected by the wall temperature set, though the resulting trends for both conventional and MEMS nozzles are applicable.

Table 4.8 contains the power transferred to water for all model results evaluated using Equations 2.116 and 2.115 with the enthalpy data obtained for water at the mentioned pressures and temperatures from the US National Institute of Standards and Technology's (NIST) publicly accessible database of thermodynamic data along with the thrust to power ratio, which is useful for small spacecraft with limited power generation capabilities. Based on information in the TU Delft VLM CFD Continuum Model Subsection 4.2.2, the inlet microresistojet temperature is 283.16 K [27]. Since the enthalpy does not change much within the applicable pressure range, the applicable nozzle inlet pressure is used. Note that the power transferred to water depends on the enthalpies along with the mass flow rate, which drops by increasing the temperature at constant pressure, explaining the variation difference for the DSMC model. The power transferred to water is higher for the conventional nozzle due to the greater mass flow rate, which could also enhance the heat transfer, though the MEMS nozzle's quasi-2D geometry is expected to surpass that heat transfer enhancement assuming proper insulation and depending on the resistive heaters' placement. If the resistive heaters are unobtrusively placed in the center, the conventional nozzle could have an advantage. Furthermore, the thrust to power ratio results are lowest for the DSMC model followed by the NS model, mainly due to their respectively lower thrusts compared to the analytical model. The conventional nozzle has a lower thrust to power ratio compared to the MEMS nozzle in the analytical model, though it is higher for both DSMC and NS (with a relatively larger difference between the conventional and MEMS nozzle thrust to power ratio values) models.

Table 4.8: Power transferred to water and thrust to power ratio for the analytical, DSMC, and NS model results

| | *Chamber Pressure = 5 bar Chamber Temperature = 550 K* | | *Chamber Pressure = 5 bar Chamber Temperature = 773 K* | | *Chamber Pressure = 7 bar Chamber Temperature = 550 K* | | *Chamber Pressure = 7 bar Chamber Temperature = 773 K* | |
|---|---|---|---|---|---|---|---|---|
| | **Conventional Nozzle** | **MEMS Nozzle** | **Conventional Nozzle** | **MEMS Nozzle** | **Conventional Nozzle** | **MEMS Nozzle** | **Conventional Nozzle** | **MEMS Nozzle** |
| **Mass Flow Rate (mg/s) (Analytical Model)** | 1.8825 | 1.6645 | 1.5713 | 1.3894 | 2.6408 | 2.3349 | 2.2011 | 1.9462 |
| **Mass Flow Rate (mg/s) (DSMC Model)** | 1.9872 | 1.3294 | 1.7223 | 1.3255 | 2.4287 | 1.9282 | 2.1671 | 1.9212 |
| **Mass Flow Rate (mg/s) (NS Model)** | 1.88 | 1.63 | N/A | 1.36 | 2.65 | 2.28 | N/A | N/A |
| **Enthalpy at 283.16 K and 5 or 7 bar (kJ/kg)** | 42.550 | 42.550 | 42.550 | 42.550 | 42.745 | 42.745 | 42.745 | 42.745 |
| **Enthalpy at 550 or 773 K (kJ/kg)** | 3016.7 | 3016.7 | 3484.1 | 3484.1 | 3010.8 | 3010.8 | 3482.0 | 3482.0 |
| **Power Transferred to Water (W) (Analytical Model)** | 5.5988 | 4.9505 | 5.4077 | 4.7817 | 7.8380 | 6.9301 | 7.5701 | 6.6935 |
| **Power Transferred to Water (W) (DSMC Model)** | 5.9102 | 3.9538 | 5.9274 | 4.5618 | 7.2085 | 5.7230 | 7.4532 | 6.6075 |
| **Power Transferred to Water (W) (NS Model)** | 5.5914 | 4.8479 | N/A | 4.6805 | 7.8653 | 6.7672 | N/A | N/A |
| **Thrust (mN) (Analytical Model)** | 2.4723 | 2.2043 | 2.5105 | 2.2409 | 3.4515 | 3.0768 | 3.5119 | 3.1345 |
| **Thrust (mN) (DSMC Model)** | 1.4418 | 0.8747 | 1.2513 | 0.8745 | 1.6013 | 1.2604 | 1.5340 | 1.2598 |
| **Thrust (mN) (NS Model)** | 2.13 | 1.52 | N/A | 1.48 | 3.04 | 2.22 | N/A | N/A |
| **Thrust to Power Ratio (mN/W) (Analytical Model)** | 0.44157 | 0.44527 | 0.46424 | 0.46864 | 0.44035 | 0.44398 | 0.46391 | 0.46829 |
| **Thrust to Power Ratio (mN/W) (DSMC Model)** | 0.24395 | 0.22123 | 0.21110 | 0.19170 | 0.22214 | 0.22023 | 0.20582 | 0.19066 |
| **Thrust to Power Ratio (mN/W) (NS Model)** | 0.38094 | 0.31354 | N/A | 0.31620 | 0.38651 | 0.32805 | N/A | N/A |

# 5

# Conclusions and Recommendations

## 5.1. Conclusions

After briefly introducing micropropulsion and discussing the propellant selection and nozzle fabrication along with the background theory related to micropropulsion as well as the analytical and OpenFOAM numerical (DSMC, continuum, and a hybrid approach containing both to accommodate to the variation in Knudsen number throughout the computational domain) modeling methods, the used methodology is based on using OpenFOAM's DSMC solver (dsmcFoam+) following the mesh creation using blockMesh and snappyHexMesh and developed analytical model (using MATLAB and CoolProp) along with an additional VLM ANSYS Fluent CFD model prepared in advance at TU Delft, where their (steady state as well as transient with very quick backward forming shock diamonds detected from the throat for DSMC) results (including the same conventional and MEMS nozzles) are processed and discussed. The nozzles are simulated for inlet pressures of 5 and 7 bar at inlet temperatures of 550 and 773 K for a total of four cases for each nozzle. To note, many of dsmcFoam+'s functionalities (mass flow rate measurements, inlet pressure boundary condition, axisymmetric capabilities, statistical error measurements, and dynamic load balancing) are implemented and described along with the full methodology, as Blender (with add-ons) and ParaView with a Python script to extract averaged data (along the nozzle and plume region) along with sampleDict are also used in pre and post-processing respectively and the simulations are carried out on a computer cluster. Furthermore, a quite interesting theoretical project on the side has been independently worked on in parallel. It started as a noticed idea that was decided to be explored using equations, which led to extended continuum/kinetic dimensionless numbers for diffusivity ($DN$) and rarefaction intensity relative to the studied object's timescale ($VDN$). $DN$ represents the continuum advective transport rate to intrinsic kinetic diffusive transport rate ratio of an object/particle in its fluid medium (ideal gas) defining how efficiently an object or particle with a constant interface can blend or diffuse into the fluid medium and between the fluid's own molecules at the instant of evaluation, as a larger and faster or smaller and slower object/particle will experience relatively greater resistance as determined by the fluid medium, which seeks optimal balance with its own properties. $VDN$ is analogous to the Mach number with an average molecular speed term instead of speed of sound (as found within $DN$ along with the Knudsen number), where a faster object/particle will observe a relatively slower flow medium average molecular speed leading to a greater rarefaction intensity and vice versa. See Appendix A for the theory derivation and its general implementation along with the explanation and evaluation. It is also tested in the present study and shows promising results, including that $DN$ is capable of approximately detecting the flow regime within its assumptions and could be more helpful when thermodynamic data (such as dynamic viscosity) for calculating the Reynolds number is difficult to obtain and $VDN$'s approach becomes rather different compared to the Mach number in flows such as the initially vacuum plume region, as the Mach number could vary significantly (increase), which allows for potential applications for $VDN$ and its understanding in highly rarefied flows.

Concerning the conventional and MEMS nozzle comparisons, it is quite clear that the conical 3D conventional nozzle (simulated as a wedge with single cell thickness using axial symmetry) is superior in performance realistically, due to the quasi-2D MEMS nozzle's (simulated as 3D) significant boundary

layer considering the viscous dissipation of flow kinetic energy from shear on the walls, especially after the throat in the diverging section. However, it reaffirms that the MEMS nozzle's geometry provides easier heat transfer with proper exterior insulation mitigating undesired heat rejection, which could be an advantage with the propellant heating involved, as the heat for these thrusters is not coming from chemical reactions, but from resistive microheaters instead, along with the possible uses for (regenerative along with potential film, curtain, transpiration, and radiation) cooling to avoid melting (in different conditions considering that the (stored) inlet microresistojet temperature considered is 283.16 K if it were to be used [27]), or decreasing viscosity, as gas viscosity generally increases as temperature increases due to the gas molecular collisions increase, contrary to the liquid viscosity decrease with a temperature increase, considering that it decreases the dominant cohesive force between the liquid molecules. To note, the DSMC and NS simulation results better resemble the significant impact of the viscous dissipation of flow kinetic energy from shear on the walls on the MEMS nozzle compared to the analytical model. If the spacecraft happens to become warmer, especially at the diverging section of the nozzle, for any reason during its mission, the MEMS nozzle will be significantly affected by this temperature change compared to the conventional nozzle. The power transferred to water and thrust to power ratio are generally higher for the conventional nozzle, as it has a greater mass flow rate that could also enhance the heat transfer, though the MEMS nozzle's quasi-2D geometry is expected to surpass that heat transfer enhancement assuming proper insulation and depending on the resistive heaters' placement. If the resistive heaters are unobtrusively placed in the center, the conventional nozzle could have an advantage. The methodology used highlights the importance of insulation for the MEMS nozzle's thrust, as a material with a lower thermal conductivity (higher thermal resistivity) would be desired for the MEMS nozzle, so that the heat before the throat and converging section (from the heater) does not easily conduct heat towards the outlet. Furthermore, the pressure at the throat lateral sides (and beyond) is higher for the MEMS nozzle, which raises the need for material strength considerations. From the rarefaction phenomena investigation considering the Knudsen number, the flows become slightly rarefied within the slip flow regime after the throat towards the outlet, which means that non-equilibrium regions emerge near surfaces as their interactions with molecules decrease leading to nonequivalent macroscopic gas and surface velocities and temperatures respectively translating to velocity-slip and temperature-jump. Some of the causes are the MEMS nozzle's smaller throat and (less than 1) throat aspect ratio inverting from rectangular horizontal flow to rectangular vertical flow at the throat and then back to a rectangular horizontal flow along with its greater total lateral surface area to total volume and perimeter to cross sectional area ratios.

MEMS components could allow for economical and technical advantages [12]. The lower volume (relatively flatter with potentially easier stacking, though not near a sensitive element of the spacecraft without sufficient thermal insulation due to the MEMS nozzle's enhanced heat transfer) and mass of the systems allow for cheaper access to space with better qualifications considering the set constraints. MEMS are highly multifunctional, where modular and integrated components can allow for multiple uses within the same chip, comparable to the MEMS "lab-on-a-chip", which is not feasible using conventional methods. Integrated fabrication allows for a more feasible, though less flexible, way to qualify components for space, as the whole module is then treated as a "black box" [12]. MEMS can also provide smaller components for better performance, such as decreasing the $I_{bit}$ and relatively increasing the thrust-to-power ratio.

The novel MEMS micropropulsion drawbacks might include compatibility complications between the MEMS material or thin film and the usually conventionally manufactured storage tank, propellants used, or possibly module sandwiching issues. The MEMS materials, such as silicon, are also likely to be undesirably conductive or brittle. At TU Delft however, a silicon dioxide layer is deposited on the silicon wafer below and above the heating layer for electrical insulation between the heating layer and conductive silicon [16]. MEMS manufacturing is also generally highly expensive compared to conventional manufacturing [49].

There are tradeoffs to choosing either thruster. So in some ways, it is like comparing apples and oranges, where it is impractical to fault one nozzle for not performing better than the other, as it ultimately comes back to the desired features and nature of the mission each is undertaking, where compromises have to be made.

## 5.2. Recommendations and Ideas for Future Work

- After referring to related literature, a geometrical optimization study on the viscous dissipation of flow kinetic energy from shear on the walls and enhanced heat transfer effects on the MEMS nozzle compared to the conventional nozzle could be performed. Note that a longer diverging section for the MEMS nozzle might not result in a continuously increasing velocity (expansion) along with that it leads to a heavier system and more friction/viscous losses, while larger diverging half angles could have potential performance benefits in general. Furthermore, a comparative study between MEMS and conventional resistive micro heat exchangers with two-phase flow to evaluate the heat transfer mechanism and optimize the geometry would be useful, where the nozzles' mass flow rates and performances are taken into consideration along with the power needed to vaporize/heat the stored liquid propellant. It would also be informative to consider the wall heat transfer (conduction/convection/radiation), as thermal insulation could be crucial for the MEMS nozzle. Also, the wall material strength could be studied, as it could play an important factor in consideration of the operating conditions.

- Implement a more elaborate DSMC model (considering the internal energy modes and intermolecular forces, as the water vapor propellant used has strong intermolecular forces, for example), referring to the DSMC Models Overview Subsection 2.4.1, along with different gas-surface interaction wall models.

- For dsmcFoam+, in a similar fashion to the solver's existing axisymmetric simulation considerations (used for the conventional nozzle simulated as a wedge with single cell thickness), which allow DSMC particles (representing a number of real atoms/molecules) to have radial weighting factors based on the radial position, where particles moving radially away from or towards the radial center could probabilistically (depending on the old and new weighting factor ratio) be discarded (due to the new larger weighting factor) or cloned (due to the new smaller weighting factor) respectively, a method for discarding and duplicating DSMC particles along the nozzle could be developed to combat the large difference in flow regime rarefaction with evolving density gradients behind (more molecules) (deletion) and beyond (less molecules) (duplication) the throat, which would result in a much less computationally expensive simulation due to the significantly higher number of particles needed to maintain a sufficient number of particles per grid cell when done without the particle weighting factors (refer to dsmcProperties Subsection 3.2.6). Another method is to automatically convert the output of a solution using a continuum solver as the input of the DSMC solver (and vice versa) using a method similar to the one described in General Modeling Properties and Procedure Section 3.1.

- Devise $DN$ and $VDN$ experiments by varying their relative terms and studying the results on different objects/particles, where they could also be tested to respectively find related properties to the Reynolds and Mach numbers, especially when thermodynamic data is difficult to obtain or there is a large difference in rarefaction (refer to Appendix A).

# A

# Extended Continuum/Kinetic Dimensionless Numbers for Diffusivity and Rarefaction Intensity

## A.1. Summary

A new dimensionless number ($DN$) based on continuum and kinetic theory is introduced. It initiates as a derivable extension of the Reynolds numbers, which might additionally allow the dimensionless number to predict the flow transition from laminar to turbulent and account for the relative intensity of rarefaction using another introduced velocity-based dimensionless number ($VDN$), which is analogous to the Mach number ($M$) with an average molecular speed term instead of speed of sound, as found within $DN$ along with the Knudsen number ($Kn$). In a sentence, $DN$ represents the ratio of an object/particle's continuum advective transport rate to the intrinsic kinetic diffusive transport rate in its fluid medium (ideal gas), which simply defines how efficiently an object or particle with a constant interface can blend or diffuse into the fluid medium and between the fluid's own molecules at the instant of evaluation. A larger and faster or smaller and slower object/particle will experience relatively greater resistance as determined by the fluid medium, which seeks optimal balance with its own properties. Alongside the transport qualities of $DN$, $VDN$ represents the proposed concept of rarefaction intensity relative to the studied object's timescale, where a faster object/particle will observe a relatively slower flow medium average molecular speed leading to a greater rarefaction intensity and vice versa. The dimensionless numbers were theoretically tested and provided logical results as shown in this work.

Although this theory could find applications in a wide variety of fields including aerodynamics, it was developed during a project on spacecraft electrothermal micropropulsion systems, but the water vapor propellant used has strong intermolecular forces that may result in a considerable deviation from an ideal gas, which is a main assumption used in the derivation of $DN$. However, at higher rarefaction/rarefaction intensity (relatively lower pressure and higher temperature), it is expected that the fluid's intermolecular forces become less relevant and the ideal gas relations become more applicable. As a topic of interest to space engineering, $DN$ might also apply in different atmospheres under ideal gas assumptions.

## A.2. Theory

The scaling analysis of the Reynolds number is used as an initial starting point, as this work has been mainly inspired to explore the noticed abstract idea of "fluid likeness", which is how much an object/particle is like the medium it flows in or how much the fluid affects it differently from the fluid's own molecules, where an example could be that turbulence transition features are considered out of the norm of the uniform laminar flow properties, so an object/particle exhibiting such features stands out. It is related to diffusivity as will be discussed, taking into consideration that the object/particle and its fluid medium will naturally attempt to reach a balanced situation based on their own properties, where their interaction will stand out more as their differences increase. It was interestingly noticed in the popular example comparing the fluid medium flow behavior of a bacterium and human.

To note, the calculations in this work are based on a rough scaling used to mainly compare the orders of magnitude. Certainly, the solution includes various assumptions, but lays a steppingstone

directed towards development for more general applications. The used assumptions' derivation might not be fully explained in this text, as that can be found from the references, but their applicability is clear. The word molecule is used to refer to a fluid's particle, while the word particle is used to refer to a significantly small object.

A bacterium's characteristic linear dimension ($L$) is around $1 \cdot 10^{-6}$ m and can reach a relative speed ($v$), which is the local flow velocity with respect to the internal or external boundaries, of around $30 \cdot 10^{-6} \frac{\mathrm{m}}{\mathrm{s}}$. The density ($\rho$) and dynamic viscosity ($\mu$) of water are considered to be $1000 \frac{\mathrm{kg}}{\mathrm{m}^3}$ and $10 \cdot 10^{-4}$ Pa·s respectively. The flow comparison is based on the Reynolds number ($Re$):

$$Re = \frac{\rho v L}{\mu} \tag{A.1}$$

For a bacterium in liquid water with the mentioned properties, $Re$ is around $3 \cdot 10^{-5}$, which falls in the laminar flow regime. The scale becomes different for a human, which for simplicity will have $L \sim 2$ m and $v \sim 2 \frac{\mathrm{m}}{\mathrm{s}}$, resulting in a turbulent flow regime with $Re \sim 4 \cdot 10^{6}$. For honey, with $\rho \sim 1420 \frac{\mathrm{kg}}{\mathrm{m}^3}$ and $\mu \sim 10$ Pa·s, a human theoretically swimming at the same speed as in water would result in $Re \sim 568$, which would be more similar to the bacterium swimming in water as it is also laminar. This represents honey's naturally larger characteristic length scale acceptance for "fluid likeness", which is related to its higher viscosity, as the bacterium is also expected to experience a more laminar flow in honey than in water. Interestingly, bacteria can travel many times their characteristic linear dimension compared to humans and that is attributed to their higher "fluid likeness" (diffusivity). In a similar manner for gas dynamics (ideal gas), the actual fluid particles travel even faster than larger objects within their medium because they intrinsically advect or diffuse themselves to appear as a bulk fluid while allocating a relative balanced speed for larger objects.

### A.2.1. Derivation

The derivation is based on certain assumptions to result in a final applicable equation. Note that $v$ and $L$ are still the only properties related to the object/particle, while the rest are properties of the internal/external fluid medium. Starting from the Reynolds number (Equation A.1), since it hints towards a more indicative solution, the mean free path ($\lambda$) equation related to dynamic viscosity (obtained from http://hyperphysics.phy-astr.gsu.edu/hbase/Kinetic/menfrevis.html) is substituted:

$$\lambda = \frac{\mu}{p} \sqrt{\frac{\pi k T}{2 M_w}}, \tag{A.2}$$

$$\frac{\rho v L}{\mu} = \frac{\rho v L}{\lambda p} \sqrt{\frac{\pi k T}{2 M_w}}, \tag{A.3}$$

where $p$ is the pressure, $T$ is the temperature, $M_w$ is the molecular mass, and $k$ is the Boltzmann constant. Next, the instantaneous density is simply considered as the number of molecules ($N$) multiplied by their individual (average) molecular mass ($M_w$) and divided by the total volume ($V$) occupied by the fluid:

$$\rho = \frac{N M_w}{V} \tag{A.4}$$

$$\frac{\rho v L}{\lambda p} \sqrt{\frac{\pi k T}{2 M_w}} = \frac{N M_w v L}{V \lambda p} \sqrt{\frac{\pi k T}{2 M_w}} = \frac{v L}{\lambda} \sqrt{\frac{N^2 M_w^2 \pi k T}{p^2 V^2 2 M_w}}, \tag{A.5}$$

The equation is then simplified along with substituting the ideal gas law, where $p$ is the absolute pressure, $n$ is the number of moles, and $R_A$ is the universal gas constant:

$$pV = n R_A T = N k T, \tag{A.6}$$

$$\frac{v L}{\lambda} \sqrt{\frac{N^2 M_w^2 \pi k T}{p^2 V^2 2 M_w}} = \frac{v L}{\lambda} \sqrt{\frac{\pi N M_w}{2 p V}}, \tag{A.7}$$

At this point, the square root of the density can be seen in the equation. The kinetic theory (ambient) gas pressure as determined in the fluid medium of the object/particle will be applied (modified from http://hyperphysics.phy-astr.gsu.edu/hbase/Kinetic/kinthe.html), where the fluid medium's (weighted) average molecular speed ($\bar{v}$) expected to (elastically) interact with the object/particle could be averaged over the object/particle's entire surface area, which creates an analogy between internal and external flows:

$$p = \frac{F_{avg.}}{A} = \frac{NM_w}{lA}\frac{(\bar{v}_x^2 + \bar{v}_y^2 + \bar{v}_z^2)}{3} = \frac{NM_w\bar{v}^2}{3lA} = \frac{NM_w\bar{v}^2}{3V}, \tag{A.8}$$

$$\frac{vL}{\lambda}\sqrt{\frac{\pi NM_w}{2pV}} = \frac{vL}{\lambda}\sqrt{\frac{3V\pi NM_w}{2NM_w\bar{v}^2V}} = \sqrt{\frac{3\pi}{2}}\frac{vL}{\bar{v}\lambda}, \tag{A.9}$$

$$DN = \frac{vL}{\bar{v}\lambda}, \tag{A.10}$$

where $F_{avg.}$ is the average force on an object/particle surface, $A$ is the fluid's area, and $l$ is the bulk fluid's characteristic length. Note that setting the velocity might require accounting for its relative direction as well, as a wind tunnel approach may need to be considered for determining the velocity of the object/particle. For fluids other than ideal gases, $DN$ is expected to relate to $Re$ by using other additional terms or a modified $DN$, yet the equation's form with a constant is quite convenient in this case. Different equations based on kinetic theory have also been used resulting in $DN$, though some assumption relations might slightly change the constant term in its relation to $Re$, but the version in this work proved to be more accurate and can also be built upon to match proven relations as shown below. In addition to its relation to the Reynold's number, the Knudsen number ($Kn$) is also relatable as it can be found in the $DN$ Equation A.10 along with another dimensionless number ($VDN$):

$$Kn = \frac{\lambda}{L} \tag{A.11}$$

$$VDN = \frac{v}{\bar{v}}, \tag{A.12}$$

$$DN = \frac{VDN}{Kn}, \tag{A.13}$$

$VDN$ is the ratio of the object/particle's speed to the fluid medium's average molecular speed, which would support the idea that as the object/particle's velocity naturally becomes closer to the fluid medium's average molecular speed, then its "fluid likeness" inherently increases. To fully be a fluid particle, the object/particle's physical length scale needs to match the fluid's molecular physical length scale (mean free path) as well. $VDN$ is analogous to the Mach number ($M$), as it replaces the Mach number's speed of sound (wave propagation) with a term for the average molecular speed instead. A $DN$ relation with the Mach number can also be established based on the relation between $Kn$, $Re$, and $M$, which ultimately becomes related to $VDN$:

$$Re = \frac{M}{Kn}\sqrt{\frac{\gamma\pi}{2}} = \sqrt{\frac{3\pi}{2}}\frac{vL}{\bar{v}\lambda} = \sqrt{\frac{3\pi}{2}}DN, \tag{A.14}$$

$$M = \sqrt{\frac{3}{\gamma}}VDN = \sqrt{\frac{3}{\gamma}}\frac{v}{\bar{v}} = \frac{v}{c}, \tag{A.15}$$

where $\gamma$ is the ratio of specific heats and $c$ is the speed of sound in the fluid medium. By rearranging, this actually results in the already established relation between speed of sound and average molecular speed for an ideal gas:

$$\frac{\bar{v}}{c} = \sqrt{\frac{3}{\gamma}} \tag{A.16}$$

Moreover, $DN$ can be seen to directly compare with the continuum Péclet number ($Pe$), though $Pe$ uses a mass diffusion coefficient ($D$) or thermal diffusivity ($\alpha$) with a different conceptual approach and there are other physical features clarified using the kinetic theory based $DN$:

$$Pe = \frac{vL}{D} = \frac{vL}{\alpha} \tag{A.17}$$

### A.2.2. Analysis

The fluid medium needs to be an ideal gas, while the object/particle is restricted to maintain its relative shape against diffusion unless it is divided into smaller elements. The resulting Equation A.9 provides a handy tool for another physical approach in determining chaotic behavior in ideal gases using kinetic theory. It is clear that the unit of the product of velocity and distance in both the numerator and denominator of $DN$ (Equation A.10) is identical to diffusivity's unit $\left(\frac{m^2}{s}\right)$, which is expectable. To elaborate on the reasoning behind the $DN$ Equation A.10, the product of the object/particle's instantaneous velocity and physical length scale determines its rate of advection transport and their magnitudes determine how the object/particle is advected, where a higher product of its two mentioned properties leads to a higher advective transport rate. As physical length scale decreases or the velocity increases, the object/particle is shown to be capable of traveling a greater distance relative to its size, but this observation is instantaneous and its relative stability is determined by $DN$. The product of the fluid medium's average molecular speed and mean free path determines the fluid medium's intrinsic diffusive transport rate, which could be described as its natural accepted balance of speed and dimension, and their magnitudes affect the object/particle's advection's variance, where a higher product of its two mentioned properties leads to a larger object/particle experiencing less difference with the fluid medium's flow and vice versa in smaller conditions.

Therefore, $DN$ is high when object/particle advection in the medium is relatively higher and vice versa, but an interesting feature is noticed at the value of one, where the advective transport rate matches the intrinsic diffusive transport rate, below which the fluid medium's intrinsic diffusion transport rate attempts to restabilize the object/particle's advective transport rate. In a theoretical sense, either a larger object at zero velocity is slightly accelerated or an infinitesimally small particle at zero velocity is extremely accelerated in analogy to momentum with the physical length scale replacing the mass term. If the object/particle has a relatively higher velocity, it will certainly be decelerated (drag). To rephrase, particles need to have a physical length scale that is smaller than the medium's mean free path, which would mean that they have to be smaller than the fluid medium's molecules to have a better probabilistic chance of traveling at speeds faster than the medium's average molecular speed, while larger objects probabilistically become slower. An example could be in a ball pit, where a large human can actually easily move the balls and make way although being significantly decelerated compared to how an object significantly smaller than the physical characteristic length of the medium (balls) could probabilistically go a much farther distance relative to its own physical length scale, but be required to divert. The system naturally seeks equilibrium. It is important to note that the kinetic theory analysis is mainly based on probability, which has derivable meaning. This theory can actually create a directly proportional relationship between the fluid medium's intrinsic diffusive transport rate and viscosity divided by density. It even provides a relation for advection/diffusion and rarefaction (intensity), as Equations A.13 and A.10 provide insight into the effect of rarefaction and rarefaction intensity on advection/diffusion, where as $VDN$ and $Kn$ become closer in value, whether high or low, the object/particle's advective transport rate becomes more stable within the fluid medium considering its intrinsic diffusive transport rate. Therefore, this provides a different perspective for understanding $DN$, where rarefaction and rarefaction intensity are used to interpret its significance. This can be further understood below, as a more in-depth meaning for $VDN$ is established.

## A.3. Testing

To put this theory to the test, a diverse collection of $Re$ values was collected for several objects flying in air (obtained from https://physics.info/turbulence/), which will be assumed to be an ideal gas. Again, the calculations in this work are based on a rough scaling used to mainly compare the orders of magnitude. From air's $\mu = 1.81 \cdot 10^{-5}$ Pa·s, $\rho = 1.225 \frac{\text{kg}}{\text{m}^3}$, and the objects' roughly estimated physical length scales, the velocity can be calculated using Equation A.1. Then, $VDN$ (Equation A.12), $Kn$ (Equation A.11),

$M$ (Equation A.15), advective transport rate/intrinsic diffusive transport rate/$DN$ (Equation A.10) are calculated followed by recalculating $Re$ and $M$ using $DN$ (Equation A.9) and $VDN$ (Equations A.15) respectively. For air, the average molecular speed is $500\frac{\mathrm{m}}{\mathrm{s}}$, speed of sound ($c$) of air is $343\frac{\mathrm{m}}{\mathrm{s}}$, specific heat ratio ($\gamma$) is 1.4, and mean free path ($\lambda$) is $70 \cdot 10^{-9}$ m. Clearly, the average diameter of air's molecules is $4 \cdot 10^{-10}$ m, which is smaller than its fluid medium's mean free path, so it should be considered that as the object/particle gets smaller than the mean free path, its physical characteristic length decreases towards the mean free path, which is reached when the particle is the same size as the fluid medium's molecules, though its physical characteristic length keeps decreasing below that as well. It can be imagined as a molecular channel flow with the mean free path as the transport gap length (diameter). The results are presented in Table A.1 and discussed below.

### A.3.1. Discussion

What is clear from the very beginning (besides how crammed the table looks) is that there is a discernible pattern in the results. $DN$ is directly proportional to $Re$ due to the constant term in its relation to $Re$ considering it being an ideal gas, but has an interesting quality when it comes to advection, diffusivity, and the "fluid likeness" concept explained earlier. As the object's physical length scale and speed decreases, $DN$ also decreases, which indicates that the fluid medium's intrinsic diffusive transport rate is resisting the advective transport rate less compared to cases with a higher $DN$ for a larger object. To add an example, when a relatively large high-speed object, such as an airplane, transfers considerable energy to its opposing flow, it inherently modifies the fluid medium's properties so that it becomes more natural within the fluid medium. An average air particle in a fluid medium of air has $DN = 1$, which is expected, as it is the most common and dominant type of particle with probabilistically averaged properties in the medium, so its advective transport rate is its own (and its fluid medium's) diffusive transport rate. Another thing to note, is that the intensity of rarefaction might not only be measured by length, but velocity could play a considerable role depending on the situation. Consider a thought experiment with a box containing a rarefied gas of extremely fast particles at the same density. Their rarefaction intensity will be observed lower in the relative continuum timescale, even though their theoretical mean free path is the same, as their high velocities will result in more collisions during the same time period. $Kn$ determines rarefaction in the flow and its intensity is relatable to $VDN$, where a lower $VDN$ means that the rarefaction intensity is lower, considering that a relatively slower object/particle would be experiencing a surrounding higher average molecular speed fluid medium, and vice versa, though the flow would be considered neutrally rarefied for $VDN = 1$. It is very important to note that in Table A.1, all $VDN$ results are based on the same fluid medium average molecular speed and different object/particle speeds, which results in a seemingly unintuitive relation at first, though explainable by considering that the rarefaction is relative to the object/particle, similar to the continuum timescale. Rarefaction is then relative to the considered object/particle timescale, as shown by the direct simulation Monte Carlo (DSMC) numerical simulations done at TU Delft on the rarefied flow in Low Pressure Microresistojet (LPM) micropropulsion systems [16], which include a channel wall heater to speed up the water vapor molecules before exiting through an expansion slot and it was determined that a higher temperature in the channel wall results in a blockage (thermal barrier) at the inlet due to higher rarefaction, which would be caused by the inlet's higher object/particle velocity due to heating leading to a relatively higher rarefaction intensity (higher $VDN$) considering that the volume of the plenum chamber and expansion slot is the same.

In the last rows of Table A.1, a set of theoretical air particles under different conditions has been included for analysis. The reason for their inclusion is to mainly visualize $DN$ around the value of one and compare $VDN$ to $Kn$. For the slower, extremely slow, and extremely slower theoretical air particles, $DN$ is below one (and even less so for a very small particle, where its physical length scale is just smaller than the mean free path for relative data visualization), reasserting the initial expectation that a slower or smaller particle in a fluid medium of fast molecules has an unstable advective transport rate, where at the instant of evaluation, it is naturally expected to be accelerated through collisions to reach the desired "equilibrium" with the other molecules. The results can be easily inferred from the equation, yet for the faster, extremely fast, and extremely faster theoretical air particles, $DN$ is above one (and even more so for a larger particle or inside a box with a larger physical length scale, which would probably typically be many orders of magnitude larger, but it is again just an example for relative data visualization), which indicates that a faster or larger particle in a fluid medium of slower molecules has an unstable advective transport rate and for the instant of evaluation is expected to be

decelerated through collisions to reach the desired “equilibrium” with the other molecules. Note that it is not expected to reach absolute “equilibrium”, as the probabilistic kinetic theory would indicate that the object/particle would also destabilize the fluid medium and the energy exchange is expected to continue. For the probably more common cases in nature due to this reason, a theoretical air particle with a combination of slow and large or fast and small presents variable results depending on the relative values between its speed and physical length scale. A slow and large or fast and small theoretical air particle presents a $DN$ of 1 (balanced rarefaction and rarefaction intensity), which indicates that it theoretically is at “equilibrium” and the momentum it gains or loses is expected to be returned through collisional probability. Therefore, for maximum fluid likeness, not only does the object/particle speed need to be similar to the fluid medium’s average molecular speed, but the object/particle’s physical length scale needs to be similar to the fluid medium’s mean free path as well. $DN$ also accounts for faster/larger particles in a fluid medium with an identically faster average molecular velocity/larger mean free path resulting in the same value as for slower/smaller particles in a fluid medium with an identically slower average molecular velocity/smaller mean free path, which means that the studied particle would ultimately have the same fluid likeness relative to its own fluid medium.

$Kn$ is found to be equal to one for all (theoretical) air particle speeds with the same physical characteristic length, decreases for larger physical characteristic lengths, and increases for smaller physical characteristic lengths, which is prone to error in determining the fluid’s rarefaction in an enclosed region with a different timescale for example. Within the assumptions used and most importantly relative to the object/particle timescale, $VDN$ approaches rarefaction in a complementary way, as it indicates the intensity of rarefaction, with higher values for higher rarefaction intensity when the collisions happen less often for faster theoretical air particles and lower values for lower rarefaction intensity when the collisions happen more often for slower theoretical air particles. Perhaps parameters other than velocity can also be added for different fluids to characterize rarefaction beyond the used assumptions’ limits. Also, the recalculated values of $Re$ and $M$ using $DN$ and $VDN$ respectively are relatively very accurate compared to their conventionally calculated values considering the multiple assumptions and rough estimate and constant values. A higher intrinsic diffusive transport rate ($DN$ denominator) with a greater fluid medium average molecular speed and mean free path would adapt better with objects/particles that have larger physical characteristic lengths and faster speeds than a lower intrinsic diffusive transport rate would, but then it becomes less adaptive with objects/particles that have smaller scales and slower speeds and the inverse applies, so it is all relative to the fluid medium’s own average molecular speed and mean free path.

$DN$ could be directly related to $Re$ as it redefines $Re$’s ratio of inertial to viscous forces using kinetic theory for an ideal gas, which brings new insight from a different perspective. $Re$ does not directly and physically characterize the specific mentioned features without at least the kinetic understanding of $DN$ and its related constant term to $Re$ for an ideal gas. $DN$ also contains the (inverse of the) Knudsen number and rarefaction intensity dimensionless number ($VDN$), which resembles an average molecular speed Mach number. A physical kinetic meaning of turbulence could also be derived, as it can be seen that a higher $DN$ disrupts the flow due to its low “fluid likeness”, so the product of the object’s speed and physical characteristic length needs to be accounted for relative to the fluid medium’s product of average molecular speed and mean free path.

This has also been tested for objects in water and the circulatory system (assuming that the molecular diameter is of the same order of magnitude as the mean free path for the fluid medium’s physical characteristic length) with generally comparably patterened results for $DN$ and $VDN$, though this work’s assumptions are based for an ideal gas and to directly apply them for a liquid would be unwise, as its viscous properties may significantly vary. This hopefully presents an idea for imagining the physical characteristic length and timescale not only in the continuum scale, but including intuition from kinetic theory as well. Perhaps the laminar/transitional/turbulent flow regimes can be further explored. Therefore, this study reaches two powerful suggested dimensionless numbers to characterize rarefaction intensity and relative advection and diffusion properties within their domains. It would now be pleasantly easy to imagine how a dust particle naturally flies in the air and almost becomes part of it with higher “fluid likeness” compared to a more flow obtrusive tree leaf. In a way, higher “fluid likeness” might probabilistically resemble the increased flow efficiency while traveling in the direction of the medium’s flow.

Table A.1: Testing $DN$ and $VDN$ Results

| (Air) | $Re$ | $L$ (m) | $v$ (m/s) | $VDN$ | $Kn$ | $M$ | Advective Transport Rate ($vL$) (m$^2$/s) | Intrinsic Diffusive Transport Rate ($\bar{v}\lambda$) (m$^2$/s) | $DN$ | $\frac{Re}{DN}$ Using | $\frac{M}{VDN}$ Using |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Boeing 747** | 2000000000 | 60 | 492.5170068 | 0.985034014 | 1.16667E-09 | 1.435909641 | 29551.02041 | 0.000035 | 844314868.8 | 1832841895 | 1.441942149 |
| **Cumulus Cloud Formation** | 250000000 | 2000 | 1.846938776 | 0.003693878 | 3.5E-11 | 0.005384661 | 3693.877551 | 0.000035 | 105539358.6 | 229105236.9 | 0.005407283 |
| **Typical Commercial Jet** | 110000000 | 35 | 46.43731778 | 0.092874636 | 0.000000002 | 0.135385766 | 1625.306122 | 0.000035 | 46437317.78 | 100806304.2 | 0.135954545 |
| **Cessna** | 6300000 | 10 | 9.308571429 | 0.018617143 | 0.000000007 | 0.027138692 | 93.08571429 | 0.000035 | 2659591.837 | 5773451.969 | 0.027252707 |
| **Light Plane** | 4700000 | 10 | 6.944489796 | 0.01388898 | 0.000000007 | 0.020246326 | 69.44489796 | 0.000035 | 1984139.942 | 4307178.453 | 0.020331384 |
| **Glider** | 1600000 | 8 | 2.955102041 | 0.005910204 | 8.75E-09 | 0.008615458 | 23.64081633 | 0.000035 | 675451.895 | 1466273.516 | 0.008651653 |
| **Model Airplane** | 250000 | 0.5 | 7.387755102 | 0.01477551 | 0.00000014 | 0.021538645 | 3.693877551 | 0.000035 | 105539.3586 | 229105.2369 | 0.021629132 |
| **Seagull** | 62000 | 0.5 | 1.832163265 | 0.003664327 | 0.00000014 | 0.005341584 | 0.916081633 | 0.000035 | 26173.76093 | 56818.09874 | 0.005364025 |
| **Paper Airplane** | 47000 | 0.2 | 3.472244898 | 0.00694449 | 0.00000035 | 0.010123163 | 0.69444898 | 0.000035 | 19841.39942 | 43071.78453 | 0.010165692 |
| **Butterfly** | 3900 | 0.005 | 11.52489796 | 0.023049796 | 0.000014 | 0.033600286 | 0.05762449 | 0.000035 | 1646.413994 | 3574.041695 | 0.033741446 |
| **Honeybee** | 1000 | 0.002 | 7.387755102 | 0.01477551 | 0.000035 | 0.021538645 | 0.01477551 | 0.000035 | 422.1574344 | 916.4209475 | 0.021629132 |
| **Housefly** | 120 | 0.006 | 0.295510204 | 0.00059102 | 1.16667E-05 | 0.000861546 | 0.001773061 | 0.000035 | 50.65889213 | 109.9705137 | 0.000865165 |
| **Chalcid Wasp** | 15 | 0.003 | 0.073877551 | 0.000147755 | 2.33333E-05 | 0.000215386 | 0.000221633 | 0.000035 | 6.332361516 | 13.74631421 | 0.000216291 |
| **Air Particle** | 2.36878453 | 0.00000007 | 500 | 1 | 1 | 1.457725948 | 0.000035 | 0.000035 | 1 | 2.170803764 | 1.463850109 |
| **(Theoretical Air Particle) Faster** | 23.6878453 | 0.00000007 | 5000 | 10 | 1 | 14.57725948 | 0.00035 | 0.000035 | 10 | 21.70803764 | 14.63850109 |
| **(Theoretical Air Particle) Extremely Fast** | 23687845.3 | 0.00000007 | 5000000000 | 10000000 | 1 | 14577259.48 | 350 | 0.000035 | 10000000 | 21708037.64 | 14638501.09 |
| **(Theoretical Air Particle) Extremely Faster** | 23687845304 | 0.00000007 | 5E+12 | 10000000000 | 1 | 14577259475 | 350000 | 0.000035 | 10000000000 | 21708037637 | 14638501094 |
| **(Theoretical Air Particle) Slower** | 0.236878453 | 0.00000007 | 50 | 0.1 | 1 | 0.145772595 | 0.0000035 | 0.000035 | 0.1 | 0.217080376 | 0.146385011 |
| **(Theoretical Air Particle) Extremely Slow** | 0.023687845 | 0.00000007 | 5 | 0.01 | 1 | 0.014577259 | 0.00000035 | 0.000035 | 0.01 | 0.021708038 | 0.014638501 |
| **(Theoretical Air Particle) Extremely Slower** | 0.002368785 | 0.00000007 | 0.5 | 0.001 | 1 | 0.001457726 | 0.000000035 | 0.000035 | 0.001 | 0.002170804 | 0.00146385 |
| **(Theoretical Air Particle) Larger or Inside Box** | 23.6878453 | 0.0000007 | 500 | 1 | 0.1 | 1.457725948 | 0.00035 | 0.000035 | 10 | 21.70803764 | 1.463850109 |
| **(Theoretical Air Particle) Faster (Larger or Inside Box)** | 236.878453 | 0.0000007 | 5000 | 10 | 0.1 | 14.57725948 | 0.0035 | 0.000035 | 100 | 217.0803764 | 14.63850109 |
| **(Theoretical Air Particle) Extremely Fast (Larger or Inside Box)** | 236878453 | 0.0000007 | 5000000000 | 10000000 | 0.1 | 14577259.48 | 3500 | 0.000035 | 100000000 | 217080376.4 | 14638501.09 |
| **(Theoretical Air Particle) Extremely Faster (Larger or Inside Box)** | 2.36878E+11 | 0.0000007 | 5E+12 | 10000000000 | 0.1 | 14577259475 | 3500000 | 0.000035 | 1E+11 | 2.1708E+11 | 14638501094 |
| **(Theoretical Air Particle) Slower (Larger or Inside Box)** | 2.36878453 | 0.0000007 | 50 | 0.1 | 0.1 | 0.145772595 | 0.000035 | 0.000035 | 1 | 2.170803764 | 0.146385011 |
| **(Theoretical Air Particle) Extremely Slow (Larger or Inside Box)** | 0.236878453 | 0.0000007 | 5 | 0.01 | 0.1 | 0.014577259 | 0.0000035 | 0.000035 | 0.1 | 0.217080376 | 0.014638501 |
| **(Theoretical Air Particle) Extremely Slower (Larger or Inside Box)** | 0.023687845 | 0.0000007 | 0.5 | 0.001 | 0.1 | 0.001457726 | 0.00000035 | 0.000035 | 0.01 | 0.021708038 | 0.00146385 |
| **(Theoretical Air Particle) Very Small** | 0.236878453 | 0.000000007 | 500 | 1 | 10 | 1.457725948 | 0.0000035 | 0.000035 | 0.1 | 0.217080376 | 1.463850109 |
| **(Theoretical Air Particle) Faster Very Small** | 2.36878453 | 0.000000007 | 5000 | 10 | 10 | 14.57725948 | 0.000035 | 0.000035 | 1 | 2.170803764 | 14.63850109 |
| **(Theoretical Air Particle) Extremely Fast Very Small** | 2368784.53 | 0.000000007 | 5000000000 | 10000000 | 10 | 14577259.48 | 35 | 0.000035 | 1000000 | 2170803.764 | 14638501.09 |
| **(Theoretical Air Particle) Extremely Faster Very Small** | 2368784530 | 0.000000007 | 5E+12 | 10000000000 | 10 | 14577259475 | 35000 | 0.000035 | 1000000000 | 2170803764 | 14638501094 |
| **(Theoretical Air Particle) Slow Very Small** | 0.023687845 | 0.000000007 | 50 | 0.1 | 10 | 0.145772595 | 0.00000035 | 0.000035 | 0.01 | 0.021708038 | 0.146385011 |
| **(Theoretical Air Particle) Extremely Slow Very Small** | 0.002368785 | 0.000000007 | 5 | 0.01 | 10 | 0.014577259 | 0.000000035 | 0.000035 | 0.001 | 0.002170804 | 0.014638501 |
| **(Theoretical Air Particle) Extremely Slower Very Small** | 0.000236878 | 0.000000007 | 0.5 | 0.001 | 10 | 0.001457726 | 3.5E-09 | 0.000035 | 0.0001 | 0.00021708 | 0.00146385 |

# B

# MATLAB Code

## B.1. Analytical Model for MEMS and Conventional Nozzles

### B.1.1. MATLAB Code

```
close all
clear all
clc
addpath('/Users/Khamis/Documents/MATLAB/main'); % CoolProp
% set(0,'DefaultFigureWindowStyle','Docked');
set(0,'DefaultAxesFontSize',16);
set(0,'DefaultTextFontSize',16);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultAxesXGrid','on');
set(0,'DefaultAxesYGrid','on');
set(0,'DefaultAxesZGrid','on');
set(0,'DefaultAxesGridLineStyle','--');
set(0,'DefaultLineMarkerSize',8);

%% Estimated Reynolds Number at Throat and Ideal Vacuum Thrust (With No Losses)
% load('Analytical_Model')

% MEMS Nozzle Case 1: Pressure of 5 bar and Temperature of 550 K

% Data
p_c_M1_bar = 5; %bar - Chamber Pressure
p_c_M1 = p_c_M1_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
T_c_M1 = 550; %K - Chamber Temperature
D = 0.1 * 10^(-3); %m - Nozzle Depth
H_inlet = 2 * 10^(-3); %m - Nozzle Inlet Height
H_t = 0.025 * 10^(-3); %m - Nozzle Throat Height
H_exit = 0.8 * 10^(-3); %m - Nozzle Exit Height
A_inlet_M1 = H_inlet * D; %m^2 - Nozzle Inlet Area
A_t_M1 = H_t * D; %m^2 - Nozzle Throat Area
A_exit_M1 = H_exit * D; %m^2 - Nozzle Exit Area
R_A_M1 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
M_w_M1_g = 18.01528; %g/mol - Molecular Mass
M_w_M1 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
WP_inlet = (H_inlet * 2) + (D * 2); %m - Nozzle Inlet Wetted Perimeter
HD_inlet = 4 * A_inlet_M1 / WP_inlet; %m - Nozzle Inlet Hydraulic Diameter
WP_t = (H_t * 2) + (D * 2); %m - Nozzle Throat Wetted Perimeter
HD_t = 4 * A_t_M1 / WP_t; %m - Nozzle Throat Hydraulic Diameter
WP_exit = (H_exit * 2) + (D * 2); %m - Nozzle Exit Wetted Perimeter
HD_exit = 4 * A_exit_M1 / WP_exit; %m - Nozzle Exit Hydraulic Diameter
Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant

% NIST
rho_inlet_M1 = 1.9984; %kg/m^3 - Inlet Density (NIST)
mu_inlet_M1 = 1.9269 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
C_p_inlet_M1 = 37.269; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
```

```
47 C_v_inlet_M1 = 28.229; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)
48
49 % Mass Flow Rate
50 gamma_inlet_M1 = C_p_inlet_M1 / C_v_inlet_M1; % - Inlet Specific Heat Ratio
51 Gamma_M1 = sqrt(gamma_inlet_M1*((1+gamma_inlet_M1)/2)^((1+gamma_inlet_M1)/(1-gamma_inlet_M1))
       ); %Vandenkerckhove Function of gamma
52 mdot_M1 = Gamma_M1 * p_c_M1 * A_t_M1 / (sqrt(T_c_M1 * R_A_M1 / M_w_M1)); %kg/s - Mass Flow
       Rate
53
54 % Throat Reynolds Number
55 v_inlet_M1 = mdot_M1 / (rho_inlet_M1 * A_inlet_M1); %m/s - Inlet Velocity
56 T_t_M1 = T_c_M1 * (1 + (1^2 * (gamma_inlet_M1 - 1) / 2))^(-1); %K - Throat Temperature
57 p_t_M1 = p_c_M1 * (1 + (1^2 * (gamma_inlet_M1 - 1) / 2))^(-gamma_inlet_M1 / (gamma_inlet_M1 -
       1)); %Pa - Throat Pressure
58 rho_t_M1 = rho_inlet_M1 * (1 + (1^2 * (gamma_inlet_M1 - 1) / 2))^(-1 / (gamma_inlet_M1 - 1));
       %kg/m^3 - Throat Density
59 rho_t_M1_check = CoolProp.PropsSI('D','T', T_t_M1, 'P', p_t_M1, 'Water'); %kg/m^3 - Throat
       Density
60 mu_t_M1 = CoolProp.PropsSI('V','T', T_t_M1, 'P', p_t_M1, 'Water'); %Pa.s - Throat Dynamic
       Viscosity
61 C_p_t_M1 = CoolProp.PropsSI('CPMOLAR','T', T_t_M1, 'P', p_t_M1, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Pressure
62 C_v_t_M1 = CoolProp.PropsSI('CVMOLAR','T', T_t_M1, 'P', p_t_M1, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Volume
63 gamma_t_M1 = C_p_t_M1 / C_v_t_M1; % - Throat Specific Heat Ratio
64 c_t_M1 = sqrt(gamma_t_M1 * R_A_M1 * T_t_M1 / M_w_M1); %m/s - Throat Speed of Sound
65 v_t_M1 = mdot_M1 / (A_t_M1 * rho_t_M1); %m/s - Throat Velocity
66 Re_t_M1 = rho_t_M1 * v_t_M1 * HD_t / mu_t_M1; % Throat Reynolds Number
67
68 % Thrust
69 syms M_exit_M1
70 M_exit_M1 = solve((A_exit_M1 / A_t_M1) == ((gamma_inlet_M1 + 1)/2)^(-(gamma_inlet_M1 + 1)/(2
       * (gamma_inlet_M1-1))) * M_exit_M1^(-1) * (1 + (M_exit_M1^2 * (gamma_inlet_M1 - 1) / 2))
       ^((gamma_inlet_M1 + 1) / (2 * (gamma_inlet_M1 - 1))), M_exit_M1);
71 M_exit_M1 = double(M_exit_M1); % Exit Mach Number
72 T_exit_M1 = T_c_M1 * (1 + (M_exit_M1^2 * (gamma_inlet_M1 - 1) / 2))^(-1); %K - Exit
       Temperature
73 p_exit_M1 = p_c_M1 * (1 + (M_exit_M1^2 * (gamma_inlet_M1 - 1) / 2))^(-gamma_inlet_M1 / (
       gamma_inlet_M1 - 1)); %Pa - Exit Pressure
74 c_exit_M1 = sqrt(gamma_inlet_M1 * R_A_M1 * T_exit_M1 / M_w_M1); %m/s - Exit Speed of Sound
75 v_exit_Mach_M1 = M_exit_M1 * c_exit_M1; %m/s - Exit Velocity Using Mach Number
76 p_0_M1 = 0; %Pa - External Pressure
77 v_exit_M1 = sqrt((1 - (p_exit_M1 / p_c_M1)^((gamma_inlet_M1 - 1) / gamma_inlet_M1)) * (2 *
       gamma_inlet_M1 * R_A_M1 * T_c_M1 / (M_w_M1 * (gamma_inlet_M1 - 1)))); %m/s - Exit
       Velocity
78 F_M1 = mdot_M1 * v_exit_M1 + (p_exit_M1 - p_0_M1) * A_exit_M1; %N - Rocket Thrust Equation
79 I_sp_M1 = F_M1 / (mdot_M1 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2
80
81 % Knudsen Number
82 MFP_inlet_M1 = R_A_M1 * T_c_M1 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_M1); %m - Inlet Mean
       Free Path
83 Kn_inlet_M1 = MFP_inlet_M1 / HD_inlet; % Inlet Knudsen Number
84 MFP_t_M1 = R_A_M1 * T_t_M1 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_M1); %m - Throat Mean
       Free Path
85 Kn_t_M1 = MFP_t_M1 / HD_t; % Throat Knudsen Number
86 MFP_exit_M1 = R_A_M1 * T_exit_M1 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_M1); %m - Exit
       Mean Free Path
87 Kn_exit_M1 = MFP_exit_M1 / HD_exit; % Exit Knudsen Number
88
89 % DN and VDN
90 V_rms_inlet_M1 = (3 * R_A_M1 * T_c_M1 / M_w_M1)^(0.5); %m/s - Inlet Root Mean Square Speed
91 VDN_inlet_M1 = v_inlet_M1 / V_rms_inlet_M1; % Inlet VDN
92 ATR_inlet_M1 = v_inlet_M1 * HD_inlet; %m^2/s - Inlet Advective Transport Rate
93 IDTR_inlet_M1 = V_rms_inlet_M1 * MFP_inlet_M1; %m^2/s - Inlet Intrinsic Diffusive Transport
       Rate
94 DN_inlet_M1 = ATR_inlet_M1 / IDTR_inlet_M1; % Inlet DN
95 M_VDN_inlet_M1 = (3 / gamma_inlet_M1)^(0.5) * VDN_inlet_M1; % Inlet Mach Number Using VDN
96 Re_DN_inlet_M1 = (3 * pi / 2)^(0.5) * DN_inlet_M1; % Inlet Reynolds Number Using DN
97 c_inlet_M1 = sqrt(gamma_inlet_M1 * R_A_M1 * T_c_M1 / M_w_M1); %m/s - Inlet Speed of Sound
98 M_inlet_M1 = v_inlet_M1 / c_inlet_M1; % Inlet Mach Number
99 Re_inlet_M1 = rho_inlet_M1 * v_inlet_M1 * HD_inlet / mu_inlet_M1; % Inlet Reynolds Number
```

```
V_rms_t_M1 = (3 * R_A_M1 * T_t_M1 / M_w_M1)^(0.5); %m/s - Throat Root Mean Square Speed
VDN_t_M1 = v_t_M1 / V_rms_t_M1; % Throat VDN
ATR_t_M1 = v_t_M1 * HD_t; %m^2/s - Throat Advective Transport Rate
IDTR_t_M1 = V_rms_t_M1 * MFP_t_M1; %m^2/s - Throat Intrinsic Diffusive Transport Rate
DN_t_M1 = ATR_t_M1 / IDTR_t_M1; % Throat DN
M_VDN_t_M1 = (3 / gamma_inlet_M1)^(0.5) * VDN_t_M1; % Throat Mach Number Using VDN
Re_DN_t_M1 = (3 * pi / 2)^(0.5) * DN_t_M1; % Throat Reynolds Number Using DN
M_t_M1 = v_t_M1 / c_t_M1; % Throat Mach Number
V_rms_exit_M1 = (3 * R_A_M1 * T_exit_M1 / M_w_M1)^(0.5); %m/s - Exit Root Mean Square Speed
VDN_exit_M1 = v_exit_M1 / V_rms_exit_M1; % Exit VDN
ATR_exit_M1 = v_exit_M1 * HD_exit; %m^2/s - Exit Advective Transport Rate
IDTR_exit_M1 = V_rms_exit_M1 * MFP_exit_M1; %m^2/s - Exit Intrinsic Diffusive Transport Rate
DN_exit_M1 = ATR_exit_M1 / IDTR_exit_M1; % Exit DN
M_VDN_exit_M1 = (3 / gamma_inlet_M1)^(0.5) * VDN_exit_M1; % Exit Mach Number Using VDN
Re_DN_exit_M1 = (3 * pi / 2)^(0.5) * DN_exit_M1; % Exit Reynolds Number Using DN
rho_exit_M1 = rho_inlet_M1 * (1 + (M_exit_M1^2 * (gamma_inlet_M1 - 1) / 2))^(-1 / (
    gamma_inlet_M1 - 1)); %kg/m^3 - Exit Density
Re_exit_M1 = rho_exit_M1 * v_exit_M1 * HD_t / mu_inlet_M1; % Exit Reynolds Number

% MEMS Nozzle Case 2: Pressure of 5 bar and Temperature of 773 K

% Data
p_c_M2_bar = 5; %bar - Chamber Pressure
p_c_M2 = p_c_M2_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
T_c_M2 = 773; %K - Chamber Temperature
D = 0.1 * 10^(-3); %m - Nozzle Depth
H_inlet = 2 * 10^(-3); %m - Nozzle Inlet Height
H_t = 0.025 * 10^(-3); %m - Nozzle Throat Height
H_exit = 0.8 * 10^(-3); %m - Nozzle Exit Height
A_inlet_M2 = H_inlet * D; %m^2 - Nozzle Inlet Area
A_t_M2 = H_t * D; %m^2 - Nozzle Throat Area
A_exit_M2 = H_exit * D; %m^2 - Nozzle Exit Area
R_A_M2 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
M_w_M2_g = 18.01528; %g/mol - Molecular Mass
M_w_M2 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
WP_inlet = (H_inlet * 2) + (D * 2); %m - Nozzle Inlet Wetted Perimeter
HD_inlet = 4 * A_inlet_M2 / WP_inlet; %m - Nozzle Inlet Hydraulic Diameter
WP_t = (H_t * 2) + (D * 2); %m - Nozzle Throat Wetted Perimeter
HD_t = 4 * A_t_M2 / WP_t; %m - Nozzle Throat Hydraulic Diameter
WP_exit = (H_exit * 2) + (D * 2); %m - Nozzle Exit Wetted Perimeter
HD_exit = 4 * A_exit_M2 / WP_exit; %m - Nozzle Exit Hydraulic Diameter
Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant

% NIST
rho_inlet_M2 = 1.4069; %kg/m^3 - Inlet Density (NIST)
mu_inlet_M2 = 2.8573 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
C_p_inlet_M2 = 38.714; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
C_v_inlet_M2 = 30.212; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)

% Mass Flow Rate
gamma_inlet_M2 = C_p_inlet_M2 / C_v_inlet_M2; % - Inlet Specific Heat Ratio
Gamma_M2 = sqrt(gamma_inlet_M2*((1+gamma_inlet_M2)/2)^((1+gamma_inlet_M2)/(1-gamma_inlet_M2))
    ); %Vandenkerckhove Function of gamma
mdot_M2 = Gamma_M2 * p_c_M2 * A_t_M2 / (sqrt(T_c_M2 * R_A_M2 / M_w_M2)); %kg/s - Mass Flow
    Rate

% Throat Reynolds Number
v_inlet_M2 = mdot_M2 / (rho_inlet_M2 * A_inlet_M2); %m/s - Inlet Velocity
T_t_M2 = T_c_M2 * (1 + (1^2 * (gamma_inlet_M2 - 1) / 2))^(-1); %K - Throat Temperature
p_t_M2 = p_c_M2 * (1 + (1^2 * (gamma_inlet_M2 - 1) / 2))^(-gamma_inlet_M2 / (gamma_inlet_M2 -
    1)); %Pa - Throat Pressure
rho_t_M2 = rho_inlet_M2 * (1 + (1^2 * (gamma_inlet_M2 - 1) / 2))^(-1 / (gamma_inlet_M2 - 1));
    %kg/m^3 - Throat Density
rho_t_M2_check = CoolProp.PropsSI('D','T', T_t_M2, 'P', p_t_M2, 'Water'); %kg/m^3 - Throat
    Density
mu_t_M2 = CoolProp.PropsSI('V','T', T_t_M2, 'P', p_t_M2, 'Water'); %Pa.s - Throat Dynamic
    Viscosity
C_p_t_M2 = CoolProp.PropsSI('CPMOLAR','T', T_t_M2, 'P', p_t_M2, 'Water'); %J/(mol.K) - Throat
    Specific Heat Capacity at Constant Pressure
```

```
C_v_t_M2 = CoolProp.PropsSI('CVMOLAR','T', T_t_M2, 'P', p_t_M2, 'Water'); %J/(mol.K) - Throat Specific Heat Capacity at Constant Volume
gamma_t_M2 = C_p_t_M2 / C_v_t_M2; % - Throat Specific Heat Ratio
c_t_M2 = sqrt(gamma_t_M2 * R_A_M2 * T_t_M2 / M_w_M2); %m/s - Throat Speed of Sound
v_t_M2 = mdot_M2 / (A_t_M2 * rho_t_M2); %m/s - Throat Velocity
Re_t_M2 = rho_t_M2 * v_t_M2 * HD_t / mu_t_M2; % Throat Reynolds Number

% Thrust
syms M_exit_M2
% M_exit_M2 = solve((A_exit_M2 / A_t_M2) == ((gamma_inlet_M2 + 1)/2)^(-(gamma_inlet_M2 + 1)/(2 * (gamma_inlet_M2-1))) * M_exit_M2^(-1) * (1 + (M_exit_M2^2 * (gamma_inlet_M2 - 1) / 2))^((gamma_inlet_M2 + 1) / (2 * (gamma_inlet_M2 - 1))), M_exit_M2);
% M_exit_M2 = double(max(M_exit_M2)); % Exit Mach Number
eqn = (A_exit_M2 / A_t_M2) == ((gamma_inlet_M2 + 1)/2)^(-(gamma_inlet_M2 + 1)/(2 * (gamma_inlet_M2-1))) * M_exit_M2^(-1) * (1 + (M_exit_M2^2 * (gamma_inlet_M2 - 1) / 2))^((gamma_inlet_M2 + 1) / (2 * (gamma_inlet_M2 - 1)));
M_exit_M2 = double(vpasolve(eqn, M_exit_M2, [1 10])); % Exit Mach Number
T_exit_M2 = T_c_M2 * (1 + (M_exit_M2^2 * (gamma_inlet_M2 - 1) / 2))^(-1); %K - Exit Temperature
p_exit_M2 = p_c_M2 * (1 + (M_exit_M2^2 * (gamma_inlet_M2 - 1) / 2))^(-gamma_inlet_M2 / (gamma_inlet_M2 - 1)); %Pa - Exit Pressure
c_exit_M2 = sqrt(gamma_inlet_M2 * R_A_M2 * T_exit_M2 / M_w_M2); %m/s - Exit Speed of Sound
v_exit_Mach_M2 = M_exit_M2 * c_exit_M2; %m/s - Exit Velocity Using Mach Number
p_0_M2 = 0; %Pa - External Pressure
v_exit_M2 = sqrt((1 - (p_exit_M2 / p_c_M2)^((gamma_inlet_M2 - 1) / gamma_inlet_M2)) * (2 * gamma_inlet_M2 * R_A_M2 * T_c_M2 / (M_w_M2 * (gamma_inlet_M2 - 1)))); %m/s - Exit Velocity
F_M2 = mdot_M2 * v_exit_M2 + (p_exit_M2 - p_0_M2) * A_exit_M2; %N - Rocket Thrust Equation
I_sp_M2 = F_M2 / (mdot_M2 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2

% Knudsen Number
MFP_inlet_M2 = R_A_M2 * T_c_M2 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_M2); %m - Inlet Mean Free Path
Kn_inlet_M2 = MFP_inlet_M2 / HD_inlet; % Inlet Knudsen Number
MFP_t_M2 = R_A_M2 * T_t_M2 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_M2); %m - Throat Mean Free Path
Kn_t_M2 = MFP_t_M2 / HD_t; % Throat Knudsen Number
MFP_exit_M2 = R_A_M2 * T_exit_M2 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_M2); %m - Exit Mean Free Path
Kn_exit_M2 = MFP_exit_M2 / HD_exit; % Exit Knudsen Number

% DN and VDN
V_rms_inlet_M2 = (3 * R_A_M2 * T_c_M2 / M_w_M2)^(0.5); %m/s - Inlet Root Mean Square Speed
VDN_inlet_M2 = v_inlet_M2 / V_rms_inlet_M2; % Inlet VDN
ATR_inlet_M2 = v_inlet_M2 * HD_inlet; %m^2/s - Inlet Advective Transport Rate
IDTR_inlet_M2 = V_rms_inlet_M2 * MFP_inlet_M2; %m^2/s - Inlet Intrinsic Diffusive Transport Rate
DN_inlet_M2 = ATR_inlet_M2 / IDTR_inlet_M2; % Inlet DN
M_VDN_inlet_M2 = (3 / gamma_inlet_M2)^(0.5) * VDN_inlet_M2; % Inlet Mach Number Using VDN
Re_DN_inlet_M2 = (3 * pi / 2)^(0.5) * DN_inlet_M2; % Inlet Reynolds Number Using DN
c_inlet_M2 = sqrt(gamma_inlet_M2 * R_A_M2 * T_c_M2 / M_w_M2); %m/s - Inlet Speed of Sound
M_inlet_M2 = v_inlet_M2 / c_inlet_M2; % Inlet Mach Number
Re_inlet_M2 = rho_inlet_M2 * v_inlet_M2 * HD_inlet / mu_inlet_M2; % Inlet Reynolds Number
V_rms_t_M2 = (3 * R_A_M2 * T_t_M2 / M_w_M2)^(0.5); %m/s - Throat Root Mean Square Speed
VDN_t_M2 = v_t_M2 / V_rms_t_M2; % Throat VDN
ATR_t_M2 = v_t_M2 * HD_t; %m^2/s - Throat Advective Transport Rate
IDTR_t_M2 = V_rms_t_M2 * MFP_t_M2; %m^2/s - Throat Intrinsic Diffusive Transport Rate
DN_t_M2 = ATR_t_M2 / IDTR_t_M2; % Throat DN
M_VDN_t_M2 = (3 / gamma_inlet_M2)^(0.5) * VDN_t_M2; % Throat Mach Number Using VDN
Re_DN_t_M2 = (3 * pi / 2)^(0.5) * DN_t_M2; % Throat Reynolds Number Using DN
M_t_M2 = v_t_M2 / c_t_M2; % Throat Mach Number
V_rms_exit_M2 = (3 * R_A_M2 * T_exit_M2 / M_w_M2)^(0.5); %m/s - Exit Root Mean Square Speed
VDN_exit_M2 = v_exit_M2 / V_rms_exit_M2; % Exit VDN
ATR_exit_M2 = v_exit_M2 * HD_exit; %m^2/s - Exit Advective Transport Rate
IDTR_exit_M2 = V_rms_exit_M2 * MFP_exit_M2; %m^2/s - Exit Intrinsic Diffusive Transport Rate
DN_exit_M2 = ATR_exit_M2 / IDTR_exit_M2; % Exit DN
M_VDN_exit_M2 = (3 / gamma_inlet_M2)^(0.5) * VDN_exit_M2; % Exit Mach Number Using VDN
Re_DN_exit_M2 = (3 * pi / 2)^(0.5) * DN_exit_M2; % Exit Reynolds Number Using DN
rho_exit_M2 = rho_inlet_M2 * (1 + (M_exit_M2^2 * (gamma_inlet_M2 - 1) / 2))^(-1 / (gamma_inlet_M2 - 1)); %kg/m^3 - Exit Density
Re_exit_M2 = rho_exit_M2 * v_exit_M2 * HD_t / mu_inlet_M2; % Exit Reynolds Number
```

```
219
220 % MEMS Nozzle Case 3: Pressure of 7 bar and Temperature of 550 K
221
222 % Data
223 p_c_M3_bar = 7; %bar - Chamber Pressure
224 p_c_M3 = p_c_M3_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
225 T_c_M3 = 550; %K - Chamber Temperature
226 D = 0.1 * 10^(-3); %m - Nozzle Depth
227 H_inlet = 2 * 10^(-3); %m - Nozzle Inlet Height
228 H_t = 0.025 * 10^(-3); %m - Nozzle Throat Height
229 H_exit = 0.8 * 10^(-3); %m - Nozzle Exit Height
230 A_inlet_M3 = H_inlet * D; %m^2 - Nozzle Inlet Area
231 A_t_M3 = H_t * D; %m^2 - Nozzle Throat Area
232 A_exit_M3 = H_exit * D; %m^2 - Nozzle Exit Area
233 R_A_M3 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
234 M_w_M3_g = 18.01528; %g/mol - Molecular Mass
235 M_w_M3 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
236 WP_inlet = (H_inlet * 2) + (D * 2); %m - Nozzle Inlet Wetted Perimeter
237 HD_inlet = 4 * A_inlet_M3 / WP_inlet; %m - Nozzle Inlet Hydraulic Diameter
238 WP_t = (H_t * 2) + (D * 2); %m - Nozzle Throat Wetted Perimeter
239 HD_t = 4 * A_t_M3 / WP_t; %m - Nozzle Throat Hydraulic Diameter
240 WP_exit = (H_exit * 2) + (D * 2); %m - Nozzle Exit Wetted Perimeter
241 HD_exit = 4 * A_exit_M3 / WP_exit; %m - Nozzle Exit Hydraulic Diameter
242 Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
243 N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant
244
245 % NIST
246 rho_inlet_M3 = 2.8145; %kg/m^3 - Inlet Density (NIST)
247 mu_inlet_M3 = 1.9240 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
248 C_p_inlet_M3 = 37.932; %%J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
249 C_v_inlet_M3 = 28.569; %%J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)
250
251 % Mass Flow Rate
252 gamma_inlet_M3 = C_p_inlet_M3 / C_v_inlet_M3; % - Inlet Specific Heat Ratio
253 Gamma_M3 = sqrt(gamma_inlet_M3*((1+gamma_inlet_M3)/2)^((1+gamma_inlet_M3)/(1-gamma_inlet_M3))
      ); %Vandenkerckhove Function of gamma
254 mdot_M3 = Gamma_M3 * p_c_M3 * A_t_M3 / (sqrt(T_c_M3 * R_A_M3 / M_w_M3)); %kg/s - Mass Flow
      Rate
255
256 % Throat Reynolds Number
257 v_inlet_M3 = mdot_M3 / (rho_inlet_M3 * A_inlet_M3); %m/s - Inlet Velocity
258 T_t_M3 = T_c_M3 * (1 + (1^2 * (gamma_inlet_M3 - 1) / 2))^(-1); %K - Throat Temperature
259 p_t_M3 = p_c_M3 * (1 + (1^2 * (gamma_inlet_M3 - 1) / 2))^(-gamma_inlet_M3 / (gamma_inlet_M3 -
       1)); %Pa - Throat Pressure
260 rho_t_M3 = rho_inlet_M3 * (1 + (1^2 * (gamma_inlet_M3 - 1) / 2))^(-1 / (gamma_inlet_M3 - 1));
      %kg/m^3 - Throat Density
261 rho_t_M3_check = CoolProp.PropsSI('D','T', T_t_M3, 'P', p_t_M3, 'Water'); %kg/m^3 - Throat
      Density
262 mu_t_M3 = CoolProp.PropsSI('V','T', T_t_M3, 'P', p_t_M3, 'Water'); %Pa.s - Throat Dynamic
      Viscosity
263 C_p_t_M3 = CoolProp.PropsSI('CPMOLAR','T', T_t_M3, 'P', p_t_M3, 'Water'); %J/(mol.K) - Throat
       Specific Heat Capacity at Constant Pressure
264 C_v_t_M3 = CoolProp.PropsSI('CVMOLAR','T', T_t_M3, 'P', p_t_M3, 'Water'); %J/(mol.K) - Throat
       Specific Heat Capacity at Constant Volume
265 gamma_t_M3 = C_p_t_M3 / C_v_t_M3; % - Throat Specific Heat Ratio
266 c_t_M3 = sqrt(gamma_t_M3 * R_A_M3 * T_t_M3 / M_w_M3); %m/s - Throat Speed of Sound
267 v_t_M3 = mdot_M3 / (A_t_M3 * rho_t_M3); %m/s - Throat Velocity
268 Re_t_M3 = rho_t_M3 * v_t_M3 * HD_t / mu_t_M3; % Throat Reynolds Number
269
270 % Thrust
271 syms M_exit_M3
272 M_exit_M3 = solve((A_exit_M3 / A_t_M3) == ((gamma_inlet_M3 + 1)/2)^(-(gamma_inlet_M3 + 1)/(2
      * (gamma_inlet_M3-1))) * M_exit_M3^(-1) * (1 + (M_exit_M3^2 * (gamma_inlet_M3 - 1) / 2))
      ^((gamma_inlet_M3 + 1) / (2 * (gamma_inlet_M3 - 1))), M_exit_M3);
273 M_exit_M3 = double(M_exit_M3); % Exit Mach Number
274 T_exit_M3 = T_c_M3 * (1 + (M_exit_M3^2 * (gamma_inlet_M3 - 1) / 2))^(-1); %K - Exit
      Temperature
275 p_exit_M3 = p_c_M3 * (1 + (M_exit_M3^2 * (gamma_inlet_M3 - 1) / 2))^(-gamma_inlet_M3 / (
      gamma_inlet_M3 - 1)); %Pa - Exit Pressure
276 c_exit_M3 = sqrt(gamma_inlet_M3 * R_A_M3 * T_exit_M3 / M_w_M3); %m/s - Exit Speed of Sound
277 v_exit_Mach_M3 = M_exit_M3 * c_exit_M3; %m/s - Exit Velocity Using Mach Number
```

```
278 p_0_M3 = 0; %Pa - External Pressure
279 v_exit_M3 = sqrt((1 - (p_exit_M3 / p_c_M3)^((gamma_inlet_M3 - 1) / gamma_inlet_M3)) * (2 *
        gamma_inlet_M3 * R_A_M3 * T_c_M3 / (M_w_M3 * (gamma_inlet_M3 - 1)))); %m/s - Exit
        Velocity
280 F_M3 = mdot_M3 * v_exit_M3 + (p_exit_M3 - p_0_M3) * A_exit_M3; %N - Rocket Thrust Equation
281 I_sp_M3 = F_M3 / (mdot_M3 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2
282
283 % Knudsen Number
284 MFP_inlet_M3 = R_A_M3 * T_c_M3 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_M3); %m - Inlet Mean
        Free Path
285 Kn_inlet_M3 = MFP_inlet_M3 / HD_inlet; % Inlet Knudsen Number
286 MFP_t_M3 = R_A_M3 * T_t_M3 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_M3); %m - Throat Mean
        Free Path
287 Kn_t_M3 = MFP_t_M3 / HD_t; % Throat Knudsen Number
288 MFP_exit_M3 = R_A_M3 * T_exit_M3 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_M3); %m - Exit
        Mean Free Path
289 Kn_exit_M3 = MFP_exit_M3 / HD_exit; % Exit Knudsen Number
290
291 % DN and VDN
292 V_rms_inlet_M3 = (3 * R_A_M3 * T_c_M3 / M_w_M3)^(0.5); %m/s - Inlet Root Mean Square Speed
293 VDN_inlet_M3 = v_inlet_M3 / V_rms_inlet_M3; % Inlet VDN
294 ATR_inlet_M3 = v_inlet_M3 * HD_inlet; %m^2/s - Inlet Advective Transport Rate
295 IDTR_inlet_M3 = V_rms_inlet_M3 * MFP_inlet_M3; %m^2/s - Inlet Intrinsic Diffusive Transport
        Rate
296 DN_inlet_M3 = ATR_inlet_M3 / IDTR_inlet_M3; % Inlet DN
297 M_VDN_inlet_M3 = (3 / gamma_inlet_M3)^(0.5) * VDN_inlet_M3; % Inlet Mach Number Using VDN
298 Re_DN_inlet_M3 = (3 * pi / 2)^(0.5) * DN_inlet_M3; % Inlet Reynolds Number Using DN
299 c_inlet_M3 = sqrt(gamma_inlet_M3 * R_A_M3 * T_c_M3 / M_w_M3); %m/s - Inlet Speed of Sound
300 M_inlet_M3 = v_inlet_M3 / c_inlet_M3; % Inlet Mach Number
301 Re_inlet_M3 = rho_inlet_M3 * v_inlet_M3 * HD_inlet / mu_inlet_M3; % Inlet Reynolds Number
302 V_rms_t_M3 = (3 * R_A_M3 * T_t_M3 / M_w_M3)^(0.5); %m/s - Throat Root Mean Square Speed
303 VDN_t_M3 = v_t_M3 / V_rms_t_M3; % Throat VDN
304 ATR_t_M3 = v_t_M3 * HD_t; %m^2/s - Throat Advective Transport Rate
305 IDTR_t_M3 = V_rms_t_M3 * MFP_t_M3; %m^2/s - Throat Intrinsic Diffusive Transport Rate
306 DN_t_M3 = ATR_t_M3 / IDTR_t_M3; % Throat DN
307 M_VDN_t_M3 = (3 / gamma_inlet_M3)^(0.5) * VDN_t_M3; % Throat Mach Number Using VDN
308 Re_DN_t_M3 = (3 * pi / 2)^(0.5) * DN_t_M3; % Throat Reynolds Number Using DN
309 M_t_M3 = v_t_M3 / c_t_M3; % Throat Mach Number
310 V_rms_exit_M3 = (3 * R_A_M3 * T_exit_M3 / M_w_M3)^(0.5); %m/s - Exit Root Mean Square Speed
311 VDN_exit_M3 = v_exit_M3 / V_rms_exit_M3; % Exit VDN
312 ATR_exit_M3 = v_exit_M3 * HD_exit; %m^2/s - Exit Advective Transport Rate
313 IDTR_exit_M3 = V_rms_exit_M3 * MFP_exit_M3; %m^2/s - Exit Intrinsic Diffusive Transport Rate
314 DN_exit_M3 = ATR_exit_M3 / IDTR_exit_M3; % Exit DN
315 M_VDN_exit_M3 = (3 / gamma_inlet_M3)^(0.5) * VDN_exit_M3; % Exit Mach Number Using VDN
316 Re_DN_exit_M3 = (3 * pi / 2)^(0.5) * DN_exit_M3; % Exit Reynolds Number Using DN
317 rho_exit_M3 = rho_inlet_M3 * (1 + (M_exit_M3^2 * (gamma_inlet_M3 - 1) / 2))^(-1 / (
        gamma_inlet_M3 - 1)); %kg/m^3 - Exit Density
318 Re_exit_M3 = rho_exit_M3 * v_exit_M3 * HD_t / mu_inlet_M3; % Exit Reynolds Number
319
320 % MEMS Nozzle Case 4: Pressure of 7 bar and Temperature of 773 K
321
322 % Data
323 p_c_M4_bar = 7; %bar - Chamber Pressure
324 p_c_M4 = p_c_M4_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
325 T_c_M4 = 773; %K - Chamber Temperature
326 D = 0.1 * 10^(-3); %m - Nozzle Depth
327 H_inlet = 2 * 10^(-3); %m - Nozzle Inlet Height
328 H_t = 0.025 * 10^(-3); %m - Nozzle Throat Height
329 H_exit = 0.8 * 10^(-3); %m - Nozzle Exit Height
330 A_inlet_M4 = H_inlet * D; %m^2 - Nozzle Inlet Area
331 A_t_M4 = H_t * D; %m^2 - Nozzle Throat Area
332 A_exit_M4 = H_exit * D; %m^2 - Nozzle Exit Area
333 R_A_M4 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
334 M_w_M4_g = 18.01528; %g/mol - Molecular Mass
335 M_w_M4 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
336 WP_inlet = (H_inlet * 2) + (D * 2); %m - Nozzle Inlet Wetted Perimeter
337 HD_inlet = 4 * A_inlet_M4 / WP_inlet; %m - Nozzle Inlet Hydraulic Diameter
338 WP_t = (H_t * 2) + (D * 2); %m - Nozzle Throat Wetted Perimeter
339 HD_t = 4 * A_t_M4 / WP_t; %m - Nozzle Throat Hydraulic Diameter
340 WP_exit = (H_exit * 2) + (D * 2); %m - Nozzle Exit Wetted Perimeter
341 HD_exit = 4 * A_exit_M4 / WP_exit; %m - Nozzle Exit Hydraulic Diameter
```

```
342 Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
343 N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant
344
345 % NIST
346 rho_inlet_M4 = 1.9726; %kg/m^3 - Inlet Density (NIST)
347 mu_inlet_M4 = 2.8576 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
348 C_p_inlet_M4 = 38.847; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
349 C_v_inlet_M4 = 30.269; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)
350
351 % Mass Flow Rate
352 gamma_inlet_M4 = C_p_inlet_M4 / C_v_inlet_M4; % - Inlet Specific Heat Ratio
353 Gamma_M4 = sqrt(gamma_inlet_M4*((1+gamma_inlet_M4)/2)^((1+gamma_inlet_M4)/(1-gamma_inlet_M4))
        ); %Vandenkerckhove Function of gamma
354 mdot_M4 = Gamma_M4 * p_c_M4 * A_t_M4 / (sqrt(T_c_M4 * R_A_M4 / M_w_M4)); %kg/s - Mass Flow
        Rate
355
356 % Throat Reynolds Number
357 v_inlet_M4 = mdot_M4 / (rho_inlet_M4 * A_inlet_M4); %m/s - Inlet Velocity
358 T_t_M4 = T_c_M4 * (1 + (1^2 * (gamma_inlet_M4 - 1) / 2))^(-1); %K - Throat Temperature
359 p_t_M4 = p_c_M4 * (1 + (1^2 * (gamma_inlet_M4 - 1) / 2))^(-gamma_inlet_M4 / (gamma_inlet_M4 -
        1)); %Pa - Throat Pressure
360 rho_t_M4 = rho_inlet_M4 * (1 + (1^2 * (gamma_inlet_M4 - 1) / 2))^(-1 / (gamma_inlet_M4 - 1));
        %kg/m^3 - Throat Density
361 rho_t_M4_check = CoolProp.PropsSI('D','T', T_t_M4, 'P', p_t_M4, 'Water'); %kg/m^3 - Throat
        Density
362 mu_t_M4 = CoolProp.PropsSI('V','T', T_t_M4, 'P', p_t_M4, 'Water'); %Pa.s - Throat Dynamic
        Viscosity
363 C_p_t_M4 = CoolProp.PropsSI('CPMOLAR','T', T_t_M4, 'P', p_t_M4, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Pressure
364 C_v_t_M4 = CoolProp.PropsSI('CVMOLAR','T', T_t_M4, 'P', p_t_M4, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Volume
365 gamma_t_M4 = C_p_t_M4 / C_v_t_M4; % - Throat Specific Heat Ratio
366 c_t_M4 = sqrt(gamma_t_M4 * R_A_M4 * T_t_M4 / M_w_M4); %m/s - Throat Speed of Sound
367 v_t_M4 = mdot_M4 / (A_t_M4 * rho_t_M4); %m/s - Throat Velocity
368 Re_t_M4 = rho_t_M4 * v_t_M4 * HD_t / mu_t_M4; % Throat Reynolds Number
369
370 % Thrust
371 syms M_exit_M4
372 M_exit_M4 = solve((A_exit_M4 / A_t_M4) == ((gamma_inlet_M4 + 1)/2)^(-(gamma_inlet_M4 + 1)/(2
        * (gamma_inlet_M4-1))) * M_exit_M4^(-1) * (1 + (M_exit_M4^2 * (gamma_inlet_M4 - 1) / 2))
        ^((gamma_inlet_M4 + 1) / (2 * (gamma_inlet_M4 - 1))), M_exit_M4);
373 M_exit_M4 = double(M_exit_M4); % Exit Mach Number
374 T_exit_M4 = T_c_M4 * (1 + (M_exit_M4^2 * (gamma_inlet_M4 - 1) / 2))^(-1); %K - Exit
        Temperature
375 p_exit_M4 = p_c_M4 * (1 + (M_exit_M4^2 * (gamma_inlet_M4 - 1) / 2))^(-gamma_inlet_M4 / (
        gamma_inlet_M4 - 1)); %Pa - Exit Pressure
376 c_exit_M4 = sqrt(gamma_inlet_M4 * R_A_M4 * T_exit_M4 / M_w_M4); %m/s - Exit Speed of Sound
377 v_exit_Mach_M4 = M_exit_M4 * c_exit_M4; %m/s - Exit Velocity Using Mach Number
378 p_0_M4 = 0; %Pa - External Pressure
379 v_exit_M4 = sqrt((1 - (p_exit_M4 / p_c_M4)^((gamma_inlet_M4 - 1) / gamma_inlet_M4)) * (2 *
        gamma_inlet_M4 * R_A_M4 * T_c_M4 / (M_w_M4 * (gamma_inlet_M4 - 1)))); %m/s - Exit
        Velocity
380 F_M4 = mdot_M4 * v_exit_M4 + (p_exit_M4 - p_0_M4) * A_exit_M4; %N - Rocket Thrust Equation
381 I_sp_M4 = F_M4 / (mdot_M4 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2
382
383 % Knudsen Number
384 MFP_inlet_M4 = R_A_M4 * T_c_M4 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_M4); %m - Inlet Mean
        Free Path
385 Kn_inlet_M4 = MFP_inlet_M4 / HD_inlet; % Inlet Knudsen Number
386 MFP_t_M4 = R_A_M4 * T_t_M4 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_M4); %m - Throat Mean
        Free Path
387 Kn_t_M4 = MFP_t_M4 / HD_t; % Throat Knudsen Number
388 MFP_exit_M4 = R_A_M4 * T_exit_M4 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_M4); %m - Exit
        Mean Free Path
389 Kn_exit_M4 = MFP_exit_M4 / HD_exit; % Exit Knudsen Number
390
391 % DN and VDN
392 V_rms_inlet_M4 = (3 * R_A_M4 * T_c_M4 / M_w_M4)^(0.5); %m/s - Inlet Root Mean Square Speed
393 VDN_inlet_M4 = v_inlet_M4 / V_rms_inlet_M4; % Inlet VDN
394 ATR_inlet_M4 = v_inlet_M4 * HD_inlet; %m^2/s - Inlet Advective Transport Rate
```

```
395 IDTR_inlet_M4 = V_rms_inlet_M4 * MFP_inlet_M4; %m^2/s - Inlet Intrinsic Diffusive Transport
        Rate
396 DN_inlet_M4 = ATR_inlet_M4 / IDTR_inlet_M4; % Inlet DN
397 M_VDN_inlet_M4 = (3 / gamma_inlet_M4)^(0.5) * VDN_inlet_M4; % Inlet Mach Number Using VDN
398 Re_DN_inlet_M4 = (3 * pi / 2)^(0.5) * DN_inlet_M4; % Inlet Reynolds Number Using DN
399 c_inlet_M4 = sqrt(gamma_inlet_M4 * R_A_M4 * T_c_M4 / M_w_M4); %m/s - Inlet Speed of Sound
400 M_inlet_M4 = v_inlet_M4 / c_inlet_M4; % Inlet Mach Number
401 Re_inlet_M4 = rho_inlet_M4 * v_inlet_M4 * HD_inlet / mu_inlet_M4; % Inlet Reynolds Number
402 V_rms_t_M4 = (3 * R_A_M4 * T_t_M4 / M_w_M4)^(0.5); %m/s - Throat Root Mean Square Speed
403 VDN_t_M4 = v_t_M4 / V_rms_t_M4; % Throat VDN
404 ATR_t_M4 = v_t_M4 * HD_t; %m^2/s - Throat Advective Transport Rate
405 IDTR_t_M4 = V_rms_t_M4 * MFP_t_M4; %m^2/s - Throat Intrinsic Diffusive Transport Rate
406 DN_t_M4 = ATR_t_M4 / IDTR_t_M4; % Throat DN
407 M_VDN_t_M4 = (3 / gamma_inlet_M4)^(0.5) * VDN_t_M4; % Throat Mach Number Using VDN
408 Re_DN_t_M4 = (3 * pi / 2)^(0.5) * DN_t_M4; % Throat Reynolds Number Using DN
409 M_t_M4 = v_t_M4 / c_t_M4; % Throat Mach Number
410 V_rms_exit_M4 = (3 * R_A_M4 * T_exit_M4 / M_w_M4)^(0.5); %m/s - Exit Root Mean Square Speed
411 VDN_exit_M4 = v_exit_M4 / V_rms_exit_M4; % Exit VDN
412 ATR_exit_M4 = v_exit_M4 * HD_exit; %m^2/s - Exit Advective Transport Rate
413 IDTR_exit_M4 = V_rms_exit_M4 * MFP_exit_M4; %m^2/s - Exit Intrinsic Diffusive Transport Rate
414 DN_exit_M4 = ATR_exit_M4 / IDTR_exit_M4; % Exit DN
415 M_VDN_exit_M4 = (3 / gamma_inlet_M4)^(0.5) * VDN_exit_M4; % Exit Mach Number Using VDN
416 Re_DN_exit_M4 = (3 * pi / 2)^(0.5) * DN_exit_M4; % Exit Reynolds Number Using DN
417 rho_exit_M4 = rho_inlet_M4 * (1 + (M_exit_M4^2 * (gamma_inlet_M4 - 1) / 2))^(-1 / (
        gamma_inlet_M4 - 1)); %kg/m^3 - Exit Density
418 Re_exit_M4 = rho_exit_M4 * v_exit_M4 * HD_t / mu_inlet_M4; % Exit Reynolds Number
419
420 % Conventional Nozzle Case 1: Pressure of 5 bar and Temperature of 550 K
421
422 % Data
423 p_c_C1_bar = 5; %bar - Chamber Pressure
424 p_c_C1 = p_c_C1_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
425 T_c_C1 = 550; %K - Chamber Temperature
426 H_inlet = 0.3 * 10^(-3); %m - Nozzle Inlet Height
427 H_t = 0.06 * 10^(-3); %m - Nozzle Throat Height
428 H_exit = 0.3 * 10^(-3); %m - Nozzle Exit Height
429 A_inlet_C1 = (H_inlet/2)^2 * pi; %m^2 - Nozzle Inlet Area
430 A_t_C1 = (H_t/2)^2 * pi; %m^2 - Nozzle Throat Area
431 A_exit_C1 = (H_exit/2)^2 * pi; %m^2 - Nozzle Exit Area
432 R_A_C1 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
433 M_w_C1_g = 18.01528; %g/mol - Molecular Mass
434 M_w_C1 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
435 Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
436 N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant
437
438 % NIST
439 rho_inlet_C1 = 1.9984; %kg/m^3 - Inlet Density (NIST)
440 mu_inlet_C1 = 1.9269 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
441 C_p_inlet_C1 = 37.269; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
442 C_v_inlet_C1 = 28.229; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)
443
444 % Mass Flow Rate
445 gamma_inlet_C1 = C_p_inlet_C1 / C_v_inlet_C1; % - Inlet Specific Heat Ratio
446 Gamma_C1 = sqrt(gamma_inlet_C1*((1+gamma_inlet_C1)/2)^((1+gamma_inlet_C1)/(1-gamma_inlet_C1))
        ); %Vandenkerckhove Function of gamma
447 mdot_C1 = Gamma_C1 * p_c_C1 * A_t_C1 / (sqrt(T_c_C1 * R_A_C1 / M_w_C1)); %kg/s - Mass Flow
        Rate
448
449 % Throat Reynolds Number
450 v_inlet_C1 = mdot_C1 / (rho_inlet_C1 * A_inlet_C1); %m/s - Inlet Velocity
451 T_t_C1 = T_c_C1 * (1 + (1^2 * (gamma_inlet_C1 - 1) / 2))^(-1); %K - Throat Temperature
452 p_t_C1 = p_c_C1 * (1 + (1^2 * (gamma_inlet_C1 - 1) / 2))^(-gamma_inlet_C1 / (gamma_inlet_C1 -
        1)); %Pa - Throat Pressure
453 rho_t_C1 = rho_inlet_C1 * (1 + (1^2 * (gamma_inlet_C1 - 1) / 2))^(-1 / (gamma_inlet_C1 - 1));
        %kg/m^3 - Throat Density
454 rho_t_C1_check = CoolProp.PropsSI('D','T', T_t_C1, 'P', p_t_C1, 'Water'); %kg/m^3 - Throat
        Density
455 mu_t_C1 = CoolProp.PropsSI('V','T', T_t_C1, 'P', p_t_C1, 'Water'); %Pa.s - Throat Dynamic
        Viscosity
456 C_p_t_C1 = CoolProp.PropsSI('CPMOLAR','T', T_t_C1, 'P', p_t_C1, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Pressure
```

```
457 C_v_t_C1 = CoolProp.PropsSI('CVMOLAR','T', T_t_C1, 'P', p_t_C1, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Volume
458 gamma_t_C1 = C_p_t_C1 / C_v_t_C1; % - Throat Specific Heat Ratio
459 c_t_C1 = sqrt(gamma_t_C1 * R_A_C1 * T_t_C1 / M_w_C1); %m/s - Throat Speed of Sound
460 v_t_C1 = mdot_C1 / (A_t_C1 * rho_t_C1); %m/s - Throat Velocity
461 Re_t_C1 = rho_t_C1 * v_t_C1 * H_t / mu_t_C1; % Throat Reynolds Number
462
463 % Thrust
464 syms M_exit_C1
465 M_exit_C1 = solve((A_exit_C1 / A_t_C1) == ((gamma_inlet_C1 + 1)/2)^(-(gamma_inlet_C1 + 1)/(2
        * (gamma_inlet_C1-1))) * M_exit_C1^(-1) * (1 + (M_exit_C1^2 * (gamma_inlet_C1 - 1) / 2))
        ^((gamma_inlet_C1 + 1) / (2 * (gamma_inlet_C1 - 1))), M_exit_C1);
466 M_exit_C1 = double(M_exit_C1); % Exit Mach Number
467 T_exit_C1 = T_c_C1 * (1 + (M_exit_C1^2 * (gamma_inlet_C1 - 1) / 2))^(-1); %K - Exit
        Temperature
468 p_exit_C1 = p_c_C1 * (1 + (M_exit_C1^2 * (gamma_inlet_C1 - 1) / 2))^(-gamma_inlet_C1 / (
        gamma_inlet_C1 - 1)); %Pa - Exit Pressure
469 c_exit_C1 = sqrt(gamma_inlet_C1 * R_A_C1 * T_exit_C1 / M_w_C1); %m/s - Exit Speed of Sound
470 v_exit_Mach_C1 = M_exit_C1 * c_exit_C1; %m/s - Exit Velocity Using Mach Number
471 p_0_C1 = 0; %Pa - External Pressure
472 v_exit_C1 = sqrt((1 - (p_exit_C1 / p_c_C1)^((gamma_inlet_C1 - 1) / gamma_inlet_C1)) * (2 *
        gamma_inlet_C1 * R_A_C1 * T_c_C1 / (M_w_C1 * (gamma_inlet_C1 - 1)))); %m/s - Exit
        Velocity
473 F_C1 = mdot_C1 * v_exit_C1 + (p_exit_C1 - p_0_C1) * A_exit_C1; %N - Rocket Thrust Equation
474 I_sp_C1 = F_C1 / (mdot_C1 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2
475
476 % Knudsen Number
477 MFP_inlet_C1 = R_A_C1 * T_c_C1 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_C1); %m - Inlet Mean
        Free Path
478 Kn_inlet_C1 = MFP_inlet_C1 / H_inlet; % Inlet Knudsen Number
479 MFP_t_C1 = R_A_C1 * T_t_C1 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_C1); %m - Throat Mean
        Free Path
480 Kn_t_C1 = MFP_t_C1 / H_t; % Throat Knudsen Number
481 MFP_exit_C1 = R_A_C1 * T_exit_C1 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_C1); %m - Exit
        Mean Free Path
482 Kn_exit_C1 = MFP_exit_C1 / H_exit; % Exit Knudsen Number
483
484 % DN and VDN
485 V_rms_inlet_C1 = (3 * R_A_C1 * T_c_C1 / M_w_C1)^(0.5); %m/s - Inlet Root Mean Square Speed
486 VDN_inlet_C1 = v_inlet_C1 / V_rms_inlet_C1; % Inlet VDN
487 ATR_inlet_C1 = v_inlet_C1 * H_inlet; %m^2/s - Inlet Advective Transport Rate
488 IDTR_inlet_C1 = V_rms_inlet_C1 * MFP_inlet_C1; %m^2/s - Inlet Intrinsic Diffusive Transport
        Rate
489 DN_inlet_C1 = ATR_inlet_C1 / IDTR_inlet_C1; % Inlet DN
490 M_VDN_inlet_C1 = (3 / gamma_inlet_C1)^(0.5) * VDN_inlet_C1; % Inlet Mach Number Using VDN
491 Re_DN_inlet_C1 = (3 * pi / 2)^(0.5) * DN_inlet_C1; % Inlet Reynolds Number Using DN
492 c_inlet_C1 = sqrt(gamma_inlet_C1 * R_A_C1 * T_c_C1 / M_w_C1); %m/s - Inlet Speed of Sound
493 M_inlet_C1 = v_inlet_C1 / c_inlet_C1; % Inlet Mach Number
494 Re_inlet_C1 = rho_inlet_C1 * v_inlet_C1 * H_inlet / mu_inlet_C1; % Inlet Reynolds Number
495 V_rms_t_C1 = (3 * R_A_C1 * T_t_C1 / M_w_C1)^(0.5); %m/s - Throat Root Mean Square Speed
496 VDN_t_C1 = v_t_C1 / V_rms_t_C1; % Throat VDN
497 ATR_t_C1 = v_t_C1 * H_t; %m^2/s - Throat Advective Transport Rate
498 IDTR_t_C1 = V_rms_t_C1 * MFP_t_C1; %m^2/s - Throat Intrinsic Diffusive Transport Rate
499 DN_t_C1 = ATR_t_C1 / IDTR_t_C1; % Throat DN
500 M_VDN_t_C1 = (3 / gamma_inlet_C1)^(0.5) * VDN_t_C1; % Throat Mach Number Using VDN
501 Re_DN_t_C1 = (3 * pi / 2)^(0.5) * DN_t_C1; % Throat Reynolds Number Using DN
502 M_t_C1 = v_t_C1 / c_t_C1; % Throat Mach Number
503 V_rms_exit_C1 = (3 * R_A_C1 * T_exit_C1 / M_w_C1)^(0.5); %m/s - Exit Root Mean Square Speed
504 VDN_exit_C1 = v_exit_C1 / V_rms_exit_C1; % Exit VDN
505 ATR_exit_C1 = v_exit_C1 * H_exit; %m^2/s - Exit Advective Transport Rate
506 IDTR_exit_C1 = V_rms_exit_C1 * MFP_exit_C1; %m^2/s - Exit Intrinsic Diffusive Transport Rate
507 DN_exit_C1 = ATR_exit_C1 / IDTR_exit_C1; % Exit DN
508 M_VDN_exit_C1 = (3 / gamma_inlet_C1)^(0.5) * VDN_exit_C1; % Exit Mach Number Using VDN
509 Re_DN_exit_C1 = (3 * pi / 2)^(0.5) * DN_exit_C1; % Exit Reynolds Number Using DN
510 rho_exit_C1 = rho_inlet_C1 * (1 + (M_exit_C1^2 * (gamma_inlet_C1 - 1) / 2))^(-1 / (
        gamma_inlet_C1 - 1)); %kg/m^3 - Exit Density
511 Re_exit_C1 = rho_exit_C1 * v_exit_C1 * H_t / mu_inlet_C1; % Exit Reynolds Number
512
513 % Conventional Nozzle Case 2: Pressure of 5 bar and Temperature of 773 K
514
515 % Data
```

```
516 p_c_C2_bar = 5; %bar - Chamber Pressure
517 p_c_C2 = p_c_C2_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
518 T_c_C2 = 773; %K - Chamber Temperature
519 H_inlet = 0.3 * 10^(-3); %m - Nozzle Inlet Height
520 H_t = 0.06 * 10^(-3); %m - Nozzle Throat Height
521 H_exit = 0.3 * 10^(-3); %m - Nozzle Exit Height
522 A_inlet_C2 = (H_inlet/2)^2 * pi; %m^2 - Nozzle Inlet Area
523 A_t_C2 = (H_t/2)^2 * pi; %m^2 - Nozzle Throat Area
524 A_exit_C2 = (H_exit/2)^2 * pi; %m^2 - Nozzle Exit Area
525 R_A_C2 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
526 M_w_C2_g = 18.01528; %g/mol - Molecular Mass
527 M_w_C2 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
528 Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
529 N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant
530
531 % NIST
532 rho_inlet_C2 = 1.4069; %kg/m^3 - Inlet Density (NIST)
533 mu_inlet_C2 = 2.8573 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
534 C_p_inlet_C2 = 38.714; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
535 C_v_inlet_C2 = 30.212; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)
536
537 % Mass Flow Rate
538 gamma_inlet_C2 = C_p_inlet_C2 / C_v_inlet_C2; % - Inlet Specific Heat Ratio
539 Gamma_C2 = sqrt(gamma_inlet_C2*((1+gamma_inlet_C2)/2)^((1+gamma_inlet_C2)/(1-gamma_inlet_C2))
        ); %Vandenkerckhove Function of gamma
540 mdot_C2 = Gamma_C2 * p_c_C2 * A_t_C2 / (sqrt(T_c_C2 * R_A_C2 / M_w_C2)); %kg/s - Mass Flow
        Rate
541
542 % Throat Reynolds Number
543 v_inlet_C2 = mdot_C2 / (rho_inlet_C2 * A_inlet_C2); %m/s - Inlet Velocity
544 T_t_C2 = T_c_C2 * (1 + (1^2 * (gamma_inlet_C2 - 1) / 2))^(-1); %K - Throat Temperature
545 p_t_C2 = p_c_C2 * (1 + (1^2 * (gamma_inlet_C2 - 1) / 2))^(-gamma_inlet_C2 / (gamma_inlet_C2 -
        1)); %Pa - Throat Pressure
546 rho_t_C2 = rho_inlet_C2 * (1 + (1^2 * (gamma_inlet_C2 - 1) / 2))^(-1 / (gamma_inlet_C2 - 1));
        %kg/m^3 - Throat Density
547 rho_t_C2_check = CoolProp.PropsSI('D','T', T_t_C2, 'P', p_t_C2, 'Water'); %kg/m^3 - Throat
        Density
548 mu_t_C2 = CoolProp.PropsSI('V','T', T_t_C2, 'P', p_t_C2, 'Water'); %Pa.s - Throat Dynamic
        Viscosity
549 C_p_t_C2 = CoolProp.PropsSI('CPMOLAR','T', T_t_C2, 'P', p_t_C2, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Pressure
550 C_v_t_C2 = CoolProp.PropsSI('CVMOLAR','T', T_t_C2, 'P', p_t_C2, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Volume
551 gamma_t_C2 = C_p_t_C2 / C_v_t_C2; % - Throat Specific Heat Ratio
552 c_t_C2 = sqrt(gamma_t_C2 * R_A_C2 * T_t_C2 / M_w_C2); %m/s - Throat Speed of Sound
553 v_t_C2 = mdot_C2 / (A_t_C2 * rho_t_C2); %m/s - Throat Velocity
554 Re_t_C2 = rho_t_C2 * v_t_C2 * H_t / mu_t_C2; % Throat Reynolds Number
555
556 % Thrust
557 syms M_exit_C2
558 M_exit_C2 = solve((A_exit_C2 / A_t_C2) == ((gamma_inlet_C2 + 1)/2)^(-(gamma_inlet_C2 + 1)/(2
        * (gamma_inlet_C2-1))) * M_exit_C2^(-1) * (1 + (M_exit_C2^2 * (gamma_inlet_C2 - 1) / 2))
        ^((gamma_inlet_C2 + 1) / (2 * (gamma_inlet_C2 - 1))), M_exit_C2);
559 M_exit_C2 = double(M_exit_C2); % Exit Mach Number
560 T_exit_C2 = T_c_C2 * (1 + (M_exit_C2^2 * (gamma_inlet_C2 - 1) / 2))^(-1); %K - Exit
        Temperature
561 p_exit_C2 = p_c_C2 * (1 + (M_exit_C2^2 * (gamma_inlet_C2 - 1) / 2))^(-gamma_inlet_C2 / (
        gamma_inlet_C2 - 1)); %Pa - Exit Pressure
562 c_exit_C2 = sqrt(gamma_inlet_C2 * R_A_C2 * T_exit_C2 / M_w_C2); %m/s - Exit Speed of Sound
563 v_exit_Mach_C2 = M_exit_C2 * c_exit_C2; %m/s - Exit Velocity Using Mach Number
564 p_0_C2 = 0; %Pa - External Pressure
565 v_exit_C2 = sqrt((1 - (p_exit_C2 / p_c_C2)^((gamma_inlet_C2 - 1) / gamma_inlet_C2)) * (2 *
        gamma_inlet_C2 * R_A_C2 * T_c_C2 / (M_w_C2 * (gamma_inlet_C2 - 1)))); %m/s - Exit
        Velocity
566 F_C2 = mdot_C2 * v_exit_C2 + (p_exit_C2 - p_0_C2) * A_exit_C2; %N - Rocket Thrust Equation
567 I_sp_C2 = F_C2 / (mdot_C2 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2
568
569 % Knudsen Number
570 MFP_inlet_C2 = R_A_C2 * T_c_C2 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_C2); %m - Inlet Mean
        Free Path
571 Kn_inlet_C2 = MFP_inlet_C2 / H_inlet; % Inlet Knudsen Number
```

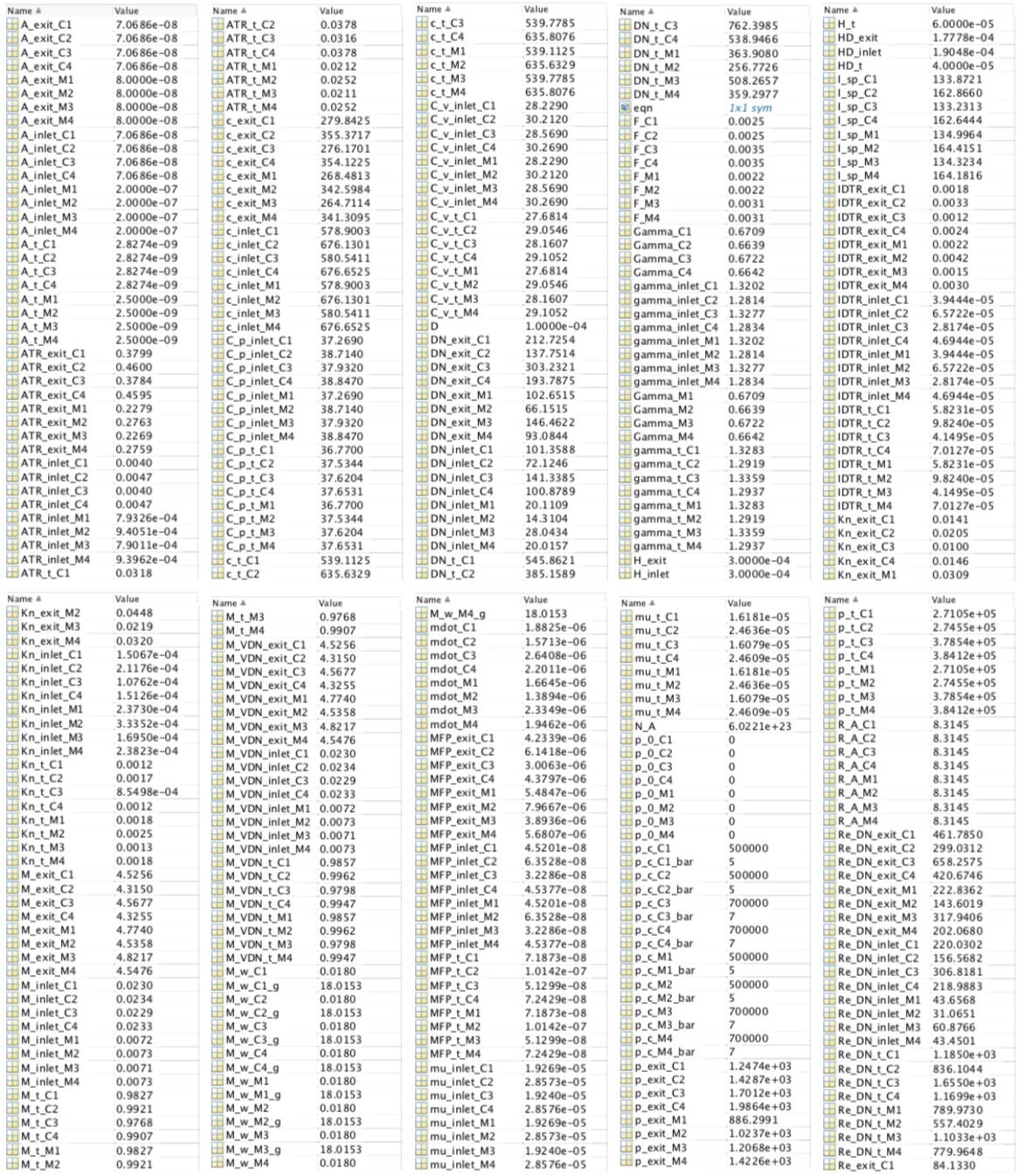
```
572 MFP_t_C2 = R_A_C2 * T_t_C2 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_C2); %m - Throat Mean
        Free Path
573 Kn_t_C2 = MFP_t_C2 / H_t; % Throat Knudsen Number
574 MFP_exit_C2 = R_A_C2 * T_exit_C2 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_C2); %m - Exit
        Mean Free Path
575 Kn_exit_C2 = MFP_exit_C2 / H_exit; % Exit Knudsen Number
576
577 % DN and VDN
578 V_rms_inlet_C2 = (3 * R_A_C2 * T_c_C2 / M_w_C2)^(0.5); %m/s - Inlet Root Mean Square Speed
579 VDN_inlet_C2 = v_inlet_C2 / V_rms_inlet_C2; % Inlet VDN
580 ATR_inlet_C2 = v_inlet_C2 * H_inlet; %m^2/s - Inlet Advective Transport Rate
581 IDTR_inlet_C2 = V_rms_inlet_C2 * MFP_inlet_C2; %m^2/s - Inlet Intrinsic Diffusive Transport
        Rate
582 DN_inlet_C2 = ATR_inlet_C2 / IDTR_inlet_C2; % Inlet DN
583 M_VDN_inlet_C2 = (3 / gamma_inlet_C2)^(0.5) * VDN_inlet_C2; % Inlet Mach Number Using VDN
584 Re_DN_inlet_C2 = (3 * pi / 2)^(0.5) * DN_inlet_C2; % Inlet Reynolds Number Using DN
585 c_inlet_C2 = sqrt(gamma_inlet_C2 * R_A_C2 * T_c_C2 / M_w_C2); %m/s - Inlet Speed of Sound
586 M_inlet_C2 = v_inlet_C2 / c_inlet_C2; % Inlet Mach Number
587 Re_inlet_C2 = rho_inlet_C2 * v_inlet_C2 * H_inlet / mu_inlet_C2; % Inlet Reynolds Number
588 V_rms_t_C2 = (3 * R_A_C2 * T_t_C2 / M_w_C2)^(0.5); %m/s - Throat Root Mean Square Speed
589 VDN_t_C2 = v_t_C2 / V_rms_t_C2; % Throat VDN
590 ATR_t_C2 = v_t_C2 * H_t; %m^2/s - Throat Advective Transport Rate
591 IDTR_t_C2 = V_rms_t_C2 * MFP_t_C2; %m^2/s - Throat Intrinsic Diffusive Transport Rate
592 DN_t_C2 = ATR_t_C2 / IDTR_t_C2; % Throat DN
593 M_VDN_t_C2 = (3 / gamma_inlet_C2)^(0.5) * VDN_t_C2; % Throat Mach Number Using VDN
594 Re_DN_t_C2 = (3 * pi / 2)^(0.5) * DN_t_C2; % Throat Reynolds Number Using DN
595 M_t_C2 = v_t_C2 / c_t_C2; % Throat Mach Number
596 V_rms_exit_C2 = (3 * R_A_C2 * T_exit_C2 / M_w_C2)^(0.5); %m/s - Exit Root Mean Square Speed
597 VDN_exit_C2 = v_exit_C2 / V_rms_exit_C2; % Exit VDN
598 ATR_exit_C2 = v_exit_C2 * H_exit; %m^2/s - Exit Advective Transport Rate
599 IDTR_exit_C2 = V_rms_exit_C2 * MFP_exit_C2; %m^2/s - Exit Intrinsic Diffusive Transport Rate
600 DN_exit_C2 = ATR_exit_C2 / IDTR_exit_C2; % Exit DN
601 M_VDN_exit_C2 = (3 / gamma_inlet_C2)^(0.5) * VDN_exit_C2; % Exit Mach Number Using VDN
602 Re_DN_exit_C2 = (3 * pi / 2)^(0.5) * DN_exit_C2; % Exit Reynolds Number Using DN
603 rho_exit_C2 = rho_inlet_C2 * (1 + (M_exit_C2^2 * (gamma_inlet_C2 - 1) / 2))^(-1 / (
        gamma_inlet_C2 - 1)); %kg/m^3 - Exit Density
604 Re_exit_C2 = rho_exit_C2 * v_exit_C2 * H_t / mu_inlet_C2; % Exit Reynolds Number
605
606 % Conventional Nozzle Case 3: Pressure of 7 bar and Temperature of 550 K
607
608 % Data
609 p_c_C3_bar = 7; %bar - Chamber Pressure
610 p_c_C3 = p_c_C3_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
611 T_c_C3 = 550; %K - Chamber Temperature
612 H_inlet = 0.3 * 10^(-3); %m - Nozzle Inlet Height
613 H_t = 0.06 * 10^(-3); %m - Nozzle Throat Height
614 H_exit = 0.3 * 10^(-3); %m - Nozzle Exit Height
615 A_inlet_C3 = (H_inlet/2)^2 * pi; %m^2 - Nozzle Inlet Area
616 A_t_C3 = (H_t/2)^2 * pi; %m^2 - Nozzle Throat Area
617 A_exit_C3 = (H_exit/2)^2 * pi; %m^2 - Nozzle Exit Area
618 R_A_C3 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
619 M_w_C3_g = 18.01528; %g/mol - Molecular Mass
620 M_w_C3 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
621 Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
622 N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant
623
624 % NIST
625 rho_inlet_C3 = 2.8145; %kg/m^3 - Inlet Density (NIST)
626 mu_inlet_C3 = 1.9240 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
627 C_p_inlet_C3 = 37.932; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
628 C_v_inlet_C3 = 28.569; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)
629
630 % Mass Flow Rate
631 gamma_inlet_C3 = C_p_inlet_C3 / C_v_inlet_C3; % - Inlet Specific Heat Ratio
632 Gamma_C3 = sqrt(gamma_inlet_C3*((1+gamma_inlet_C3)/2)^((1+gamma_inlet_C3)/(1-gamma_inlet_C3))
        ); %Vandenkerckhove Function of gamma
633 mdot_C3 = Gamma_C3 * p_c_C3 * A_t_C3 / (sqrt(T_c_C3 * R_A_C3 / M_w_C3)); %kg/s - Mass Flow
        Rate
634
635 % Throat Reynolds Number
636 v_inlet_C3 = mdot_C3 / (rho_inlet_C3 * A_inlet_C3); %m/s - Inlet Velocity
```

```
637 T_t_C3 = T_c_C3 * (1 + (1^2 * (gamma_inlet_C3 - 1) / 2))^(-1); %K - Throat Temperature
638 p_t_C3 = p_c_C3 * (1 + (1^2 * (gamma_inlet_C3 - 1) / 2))^(-gamma_inlet_C3 / (gamma_inlet_C3 -
        1)); %Pa - Throat Pressure
639 rho_t_C3 = rho_inlet_C3 * (1 + (1^2 * (gamma_inlet_C3 - 1) / 2))^(-1 / (gamma_inlet_C3 - 1));
        %kg/m^3 - Throat Density
640 rho_t_C3_check = CoolProp.PropsSI('D','T', T_t_C3, 'P', p_t_C3, 'Water'); %kg/m^3 - Throat
        Density
641 mu_t_C3 = CoolProp.PropsSI('V','T', T_t_C3, 'P', p_t_C3, 'Water'); %Pa.s - Throat Dynamic
        Viscosity
642 C_p_t_C3 = CoolProp.PropsSI('CPMOLAR','T', T_t_C3, 'P', p_t_C3, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Pressure
643 C_v_t_C3 = CoolProp.PropsSI('CVMOLAR','T', T_t_C3, 'P', p_t_C3, 'Water'); %J/(mol.K) - Throat
        Specific Heat Capacity at Constant Volume
644 gamma_t_C3 = C_p_t_C3 / C_v_t_C3; % - Throat Specific Heat Ratio
645 c_t_C3 = sqrt(gamma_t_C3 * R_A_C3 * T_t_C3 / M_w_C3); %m/s - Throat Speed of Sound
646 v_t_C3 = mdot_C3 / (A_t_C3 * rho_t_C3); %m/s - Throat Velocity
647 Re_t_C3 = rho_t_C3 * v_t_C3 * H_t / mu_t_C3; % Throat Reynolds Number
648
649 % Thrust
650 syms M_exit_C3
651 M_exit_C3 = solve((A_exit_C3 / A_t_C3) == ((gamma_inlet_C3 + 1)/2)^(-(gamma_inlet_C3 + 1)/(2
        * (gamma_inlet_C3-1))) * M_exit_C3^(-1) * (1 + (M_exit_C3^2 * (gamma_inlet_C3 - 1) / 2))
        ^((gamma_inlet_C3 + 1) / (2 * (gamma_inlet_C3 - 1))), M_exit_C3);
652 M_exit_C3 = double(M_exit_C3); % Exit Mach Number
653 T_exit_C3 = T_c_C3 * (1 + (M_exit_C3^2 * (gamma_inlet_C3 - 1) / 2))^(-1); %K - Exit
        Temperature
654 p_exit_C3 = p_c_C3 * (1 + (M_exit_C3^2 * (gamma_inlet_C3 - 1) / 2))^(-gamma_inlet_C3 / (
        gamma_inlet_C3 - 1)); %Pa - Exit Pressure
655 c_exit_C3 = sqrt(gamma_inlet_C3 * R_A_C3 * T_exit_C3 / M_w_C3); %m/s - Exit Speed of Sound
656 v_exit_Mach_C3 = M_exit_C3 * c_exit_C3; %m/s - Exit Velocity Using Mach Number
657 p_0_C3 = 0; %Pa - External Pressure
658 v_exit_C3 = sqrt((1 - (p_exit_C3 / p_c_C3)^((gamma_inlet_C3 - 1) / gamma_inlet_C3)) * (2 *
        gamma_inlet_C3 * R_A_C3 * T_c_C3 / (M_w_C3 * (gamma_inlet_C3 - 1)))); %m/s - Exit
        Velocity
659 F_C3 = mdot_C3 * v_exit_C3 + (p_exit_C3 - p_0_C3) * A_exit_C3; %N - Rocket Thrust Equation
660 I_sp_C3 = F_C3 / (mdot_C3 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2
661
662 % Knudsen Number
663 MFP_inlet_C3 = R_A_C3 * T_c_C3 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_C3); %m - Inlet Mean
        Free Path
664 Kn_inlet_C3 = MFP_inlet_C3 / H_inlet; % Inlet Knudsen Number
665 MFP_t_C3 = R_A_C3 * T_t_C3 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_C3); %m - Throat Mean
        Free Path
666 Kn_t_C3 = MFP_t_C3 / H_t; % Throat Knudsen Number
667 MFP_exit_C3 = R_A_C3 * T_exit_C3 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_C3); %m - Exit
        Mean Free Path
668 Kn_exit_C3 = MFP_exit_C3 / H_exit; % Exit Knudsen Number
669
670 % DN and VDN
671 V_rms_inlet_C3 = (3 * R_A_C3 * T_c_C3 / M_w_C3)^(0.5); %m/s - Inlet Root Mean Square Speed
672 VDN_inlet_C3 = v_inlet_C3 / V_rms_inlet_C3; % Inlet VDN
673 ATR_inlet_C3 = v_inlet_C3 * H_inlet; %m^2/s - Inlet Advective Transport Rate
674 IDTR_inlet_C3 = V_rms_inlet_C3 * MFP_inlet_C3; %m^2/s - Inlet Intrinsic Diffusive Transport
        Rate
675 DN_inlet_C3 = ATR_inlet_C3 / IDTR_inlet_C3; % Inlet DN
676 M_VDN_inlet_C3 = (3 / gamma_inlet_C3)^(0.5) * VDN_inlet_C3; % Inlet Mach Number Using VDN
677 Re_DN_inlet_C3 = (3 * pi / 2)^(0.5) * DN_inlet_C3; % Inlet Reynolds Number Using DN
678 c_inlet_C3 = sqrt(gamma_inlet_C3 * R_A_C3 * T_c_C3 / M_w_C3); %m/s - Inlet Speed of Sound
679 M_inlet_C3 = v_inlet_C3 / c_inlet_C3; % Inlet Mach Number
680 Re_inlet_C3 = rho_inlet_C3 * v_inlet_C3 * H_inlet / mu_inlet_C3; % Inlet Reynolds Number
681 V_rms_t_C3 = (3 * R_A_C3 * T_t_C3 / M_w_C3)^(0.5); %m/s - Throat Root Mean Square Speed
682 VDN_t_C3 = v_t_C3 / V_rms_t_C3; % Throat VDN
683 ATR_t_C3 = v_t_C3 * H_t; %m^2/s - Throat Advective Transport Rate
684 IDTR_t_C3 = V_rms_t_C3 * MFP_t_C3; %m^2/s - Throat Intrinsic Diffusive Transport Rate
685 DN_t_C3 = ATR_t_C3 / IDTR_t_C3; % Throat DN
686 M_VDN_t_C3 = (3 / gamma_inlet_C3)^(0.5) * VDN_t_C3; % Throat Mach Number Using VDN
687 Re_DN_t_C3 = (3 * pi / 2)^(0.5) * DN_t_C3; % Throat Reynolds Number Using DN
688 M_t_C3 = v_t_C3 / c_t_C3; % Throat Mach Number
689 V_rms_exit_C3 = (3 * R_A_C3 * T_exit_C3 / M_w_C3)^(0.5); %m/s - Exit Root Mean Square Speed
690 VDN_exit_C3 = v_exit_C3 / V_rms_exit_C3; % Exit VDN
691 ATR_exit_C3 = v_exit_C3 * H_exit; %m^2/s - Exit Advective Transport Rate
```

```
692 IDTR_exit_C3 = V_rms_exit_C3 * MFP_exit_C3; %m^2/s - Exit Intrinsic Diffusive Transport Rate
693 DN_exit_C3 = ATR_exit_C3 / IDTR_exit_C3; % Exit DN
694 M_VDN_exit_C3 = (3 / gamma_inlet_C3)^(0.5) * VDN_exit_C3; % Exit Mach Number Using VDN
695 Re_DN_exit_C3 = (3 * pi / 2)^(0.5) * DN_exit_C3; % Exit Reynolds Number Using DN
696 rho_exit_C3 = rho_inlet_C3 * (1 + (M_exit_C3^2 * (gamma_inlet_C3 - 1) / 2))^(-1 / (
        gamma_inlet_C3 - 1)); %kg/m^3 - Exit Density
697 Re_exit_C3 = rho_exit_C3 * v_exit_C3 * H_t / mu_inlet_C3; % Exit Reynolds Number
698
699 % Conventional Nozzle Case 4: Pressure of 7 bar and Temperature of 773 K
700
701 % Data
702 p_c_C4_bar = 7; %bar - Chamber Pressure
703 p_c_C4 = p_c_C4_bar * 100000; %Pa = kg/(m.s^(-2)) - Chamber Pressure
704 T_c_C4 = 773; %K - Chamber Temperature
705 H_inlet = 0.3 * 10^(-3); %m - Nozzle Inlet Height
706 H_t = 0.06 * 10^(-3); %m - Nozzle Throat Height
707 H_exit = 0.3 * 10^(-3); %m - Nozzle Exit Height
708 A_inlet_C4 = (H_inlet/2)^2 * pi; %m^2 - Nozzle Inlet Area
709 A_t_C4 = (H_t/2)^2 * pi; %m^2 - Nozzle Throat Area
710 A_exit_C4 = (H_exit/2)^2 * pi; %m^2 - Nozzle Exit Area
711 R_A_C4 = 8.3144598; %J/(mol.K) = kg.m^(2)/(s^(-2).mol.K) - Universal Gas Constant
712 M_w_C4_g = 18.01528; %g/mol - Molecular Mass
713 M_w_C4 = 18.0153 * 10^(-3); %kg/mol - Molecular Mass
714 Water_D = 2.75 * 10^(-10); %m - Water Molecular Diameter
715 N_A = 6.022140857 * 10^(23); %/mol - Avogadro Constant
716
717 % NIST
718 rho_inlet_C4 = 1.9726; %kg/m^3 - Inlet Density (NIST)
719 mu_inlet_C4 = 2.8576 * 10^(-5); %Pa.s - Inlet Dynamic Viscosity (NIST)
720 C_p_inlet_C4 = 38.847; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Pressure (NIST)
721 C_v_inlet_C4 = 30.269; %J/(mol.K) - Inlet Specific Heat Capacity at Constant Volume (NIST)
722
723 % Mass Flow Rate
724 gamma_inlet_C4 = C_p_inlet_C4 / C_v_inlet_C4; % - Inlet Specific Heat Ratio
725 Gamma_C4 = sqrt(gamma_inlet_C4*((1+gamma_inlet_C4)/2)^((1+gamma_inlet_C4)/(1-gamma_inlet_C4))
        ); %Vandenkerckhove Function of gamma
726 mdot_C4 = Gamma_C4 * p_c_C4 * A_t_C4 / (sqrt(T_c_C4 * R_A_C4 / M_w_C4)); %kg/s - Mass Flow
        Rate
727
728 % Throat Reynolds Number
729 v_inlet_C4 = mdot_C4 / (rho_inlet_C4 * A_inlet_C4); %m/s - Inlet Velocity
730 T_t_C4 = T_c_C4 * (1 + (1^2 * (gamma_inlet_C4 - 1) / 2))^(-1); %K - Throat Temperature
731 p_t_C4 = p_c_C4 * (1 + (1^2 * (gamma_inlet_C4 - 1) / 2))^(-gamma_inlet_C4 / (gamma_inlet_C4 -
        1)); %Pa - Throat Pressure
732 rho_t_C4 = rho_inlet_C4 * (1 + (1^2 * (gamma_inlet_C4 - 1) / 2))^(-1 / (gamma_inlet_C4 - 1));
        %kg/m^3 - Throat Density
733 rho_t_C4_check = CoolProp.PropsSI('D','T', T_t_C4, 'P', p_t_C4, 'Water'); %kg/m^3 - Throat
        Density
734 mu_t_C4 = CoolProp.PropsSI('V','T', T_t_C4, 'P', p_t_C4, 'Water'); %Pa.s - Throat Dynamic
        Viscosity
735 C_p_t_C4 = CoolProp.PropsSI('CPMOLAR','T', T_t_C4, 'P', p_t_C4, 'Water'); %J/(mol.K) - Throat
         Specific Heat Capacity at Constant Pressure
736 C_v_t_C4 = CoolProp.PropsSI('CVMOLAR','T', T_t_C4, 'P', p_t_C4, 'Water'); %J/(mol.K) - Throat
         Specific Heat Capacity at Constant Volume
737 gamma_t_C4 = C_p_t_C4 / C_v_t_C4; % - Throat Specific Heat Ratio
738 c_t_C4 = sqrt(gamma_t_C4 * R_A_C4 * T_t_C4 / M_w_C4); %m/s - Throat Speed of Sound
739 v_t_C4 = mdot_C4 / (A_t_C4 * rho_t_C4); %m/s - Throat Velocity
740 Re_t_C4 = rho_t_C4 * v_t_C4 * H_t / mu_t_C4; % Throat Reynolds Number
741
742 % Thrust
743 syms M_exit_C4
744 M_exit_C4 = solve((A_exit_C4 / A_t_C4) == ((gamma_inlet_C4 + 1)/2)^(-(gamma_inlet_C4 + 1)/(2
        * (gamma_inlet_C4-1))) * M_exit_C4^(-1) * (1 + (M_exit_C4^2 * (gamma_inlet_C4 - 1) / 2))
        ^((gamma_inlet_C4 + 1) / (2 * (gamma_inlet_C4 - 1))), M_exit_C4);
745 M_exit_C4 = double(M_exit_C4); % Exit Mach Number
746 T_exit_C4 = T_c_C4 * (1 + (M_exit_C4^2 * (gamma_inlet_C4 - 1) / 2))^(-1); %K - Exit
        Temperature
747 p_exit_C4 = p_c_C4 * (1 + (M_exit_C4^2 * (gamma_inlet_C4 - 1) / 2))^(-gamma_inlet_C4 / (
        gamma_inlet_C4 - 1)); %Pa - Exit Pressure
748 c_exit_C4 = sqrt(gamma_inlet_C4 * R_A_C4 * T_exit_C4 / M_w_C4); %m/s - Exit Speed of Sound
749 v_exit_Mach_C4 = M_exit_C4 * c_exit_C4; %m/s - Exit Velocity Using Mach Number
```

```
750 p_0_C4 = 0; %Pa - External Pressure
751 v_exit_C4 = sqrt((1 - (p_exit_C4 / p_c_C4)^((gamma_inlet_C4 - 1) / gamma_inlet_C4)) * (2 *
        gamma_inlet_C4 * R_A_C4 * T_c_C4 / (M_w_C4 * (gamma_inlet_C4 - 1)))); %m/s - Exit
        Velocity
752 F_C4 = mdot_C4 * v_exit_C4 + (p_exit_C4 - p_0_C4) * A_exit_C4; %N - Rocket Thrust Equation
753 I_sp_C4 = F_C4 / (mdot_C4 * 9.81); %s - Specific Impulse Using Gravity of 9.81 m/s^2
754
755 % Knudsen Number
756 MFP_inlet_C4 = R_A_C4 * T_c_C4 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_c_C4); %m - Inlet Mean
        Free Path
757 Kn_inlet_C4 = MFP_inlet_C4 / H_inlet; % Inlet Knudsen Number
758 MFP_t_C4 = R_A_C4 * T_t_C4 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_t_C4); %m - Throat Mean
        Free Path
759 Kn_t_C4 = MFP_t_C4 / H_t; % Throat Knudsen Number
760 MFP_exit_C4 = R_A_C4 * T_exit_C4 / ((2)^(0.5) * pi * Water_D^2 * N_A * p_exit_C4); %m - Exit
        Mean Free Path
761 Kn_exit_C4 = MFP_exit_C4 / H_exit; % Exit Knudsen Number
762
763 % DN and VDN
764 V_rms_inlet_C4 = (3 * R_A_C4 * T_c_C4 / M_w_C4)^(0.5); %m/s - Inlet Root Mean Square Speed
765 VDN_inlet_C4 = v_inlet_C4 / V_rms_inlet_C4; % Inlet VDN
766 ATR_inlet_C4 = v_inlet_C4 * H_inlet; %m^2/s - Inlet Advective Transport Rate
767 IDTR_inlet_C4 = V_rms_inlet_C4 * MFP_inlet_C4; %m^2/s - Inlet Intrinsic Diffusive Transport
        Rate
768 DN_inlet_C4 = ATR_inlet_C4 / IDTR_inlet_C4; % Inlet DN
769 M_VDN_inlet_C4 = (3 / gamma_inlet_C4)^(0.5) * VDN_inlet_C4; % Inlet Mach Number Using VDN
770 Re_DN_inlet_C4 = (3 * pi / 2)^(0.5) * DN_inlet_C4; % Inlet Reynolds Number Using DN
771 c_inlet_C4 = sqrt(gamma_inlet_C4 * R_A_C4 * T_c_C4 / M_w_C4); %m/s - Inlet Speed of Sound
772 M_inlet_C4 = v_inlet_C4 / c_inlet_C4; % Inlet Mach Number
773 Re_inlet_C4 = rho_inlet_C4 * v_inlet_C4 * H_inlet / mu_inlet_C4; % Inlet Reynolds Number
774 V_rms_t_C4 = (3 * R_A_C4 * T_t_C4 / M_w_C4)^(0.5); %m/s - Throat Root Mean Square Speed
775 VDN_t_C4 = v_t_C4 / V_rms_t_C4; % Throat VDN
776 ATR_t_C4 = v_t_C4 * H_t; %m^2/s - Throat Advective Transport Rate
777 IDTR_t_C4 = V_rms_t_C4 * MFP_t_C4; %m^2/s - Throat Intrinsic Diffusive Transport Rate
778 DN_t_C4 = ATR_t_C4 / IDTR_t_C4; % Throat DN
779 M_VDN_t_C4 = (3 / gamma_inlet_C4)^(0.5) * VDN_t_C4; % Throat Mach Number Using VDN
780 Re_DN_t_C4 = (3 * pi / 2)^(0.5) * DN_t_C4; % Throat Reynolds Number Using DN
781 M_t_C4 = v_t_C4 / c_t_C4; % Throat Mach Number
782 V_rms_exit_C4 = (3 * R_A_C4 * T_exit_C4 / M_w_C4)^(0.5); %m/s - Exit Root Mean Square Speed
783 VDN_exit_C4 = v_exit_C4 / V_rms_exit_C4; % Exit VDN
784 ATR_exit_C4 = v_exit_C4 * H_exit; %m^2/s - Exit Advective Transport Rate
785 IDTR_exit_C4 = V_rms_exit_C4 * MFP_exit_C4; %m^2/s - Exit Intrinsic Diffusive Transport Rate
786 DN_exit_C4 = ATR_exit_C4 / IDTR_exit_C4; % Exit DN
787 M_VDN_exit_C4 = (3 / gamma_inlet_C4)^(0.5) * VDN_exit_C4; % Exit Mach Number Using VDN
788 Re_DN_exit_C4 = (3 * pi / 2)^(0.5) * DN_exit_C4; % Exit Reynolds Number Using DN
789 rho_exit_C4 = rho_inlet_C4 * (1 + (M_exit_C4^2 * (gamma_inlet_C4 - 1) / 2))^(-1 / (
        gamma_inlet_C4 - 1)); %kg/m^3 - Exit Density
790 Re_exit_C4 = rho_exit_C4 * v_exit_C4 * H_t / mu_inlet_C4; % Exit Reynolds Number
791
792 %%
793 % save('Analytical_Model');
```

## B.1.2. MATLAB Workspace Results



| Name | Value |
|---|---|
| A_exit_C1 | 7.0686e-08 |
| A_exit_C2 | 7.0686e-08 |
| A_exit_C3 | 7.0686e-08 |
| A_exit_C4 | 7.0686e-08 |
| A_exit_M1 | 8.0000e-08 |
| A_exit_M2 | 8.0000e-08 |
| A_exit_M3 | 8.0000e-08 |
| A_exit_M4 | 8.0000e-08 |
| A_inlet_C1 | 7.0686e-08 |
| A_inlet_C2 | 7.0686e-08 |
| A_inlet_C3 | 7.0686e-08 |
| A_inlet_C4 | 7.0686e-08 |
| A_inlet_M1 | 2.0000e-07 |
| A_inlet_M2 | 2.0000e-07 |
| A_inlet_M3 | 2.0000e-07 |
| A_inlet_M4 | 2.0000e-07 |
| A_t_C1 | 2.8274e-09 |
| A_t_C2 | 2.8274e-09 |
| A_t_C3 | 2.8274e-09 |
| A_t_C4 | 2.8274e-09 |
| A_t_M1 | 2.5000e-09 |
| A_t_M2 | 2.5000e-09 |
| A_t_M3 | 2.5000e-09 |
| A_t_M4 | 2.5000e-09 |
| ATR_exit_C1 | 0.3799 |
| ATR_exit_C2 | 0.4600 |
| ATR_exit_C3 | 0.3784 |
| ATR_exit_C4 | 0.4595 |
| ATR_exit_M1 | 0.2279 |
| ATR_exit_M2 | 0.2763 |
| ATR_exit_M3 | 0.2269 |
| ATR_exit_M4 | 0.2759 |
| ATR_inlet_C1 | 0.0040 |
| ATR_inlet_C2 | 0.0047 |
| ATR_inlet_C3 | 0.0040 |
| ATR_inlet_C4 | 0.0047 |
| ATR_inlet_M1 | 7.9326e-04 |
| ATR_inlet_M2 | 9.4051e-04 |
| ATR_inlet_M3 | 7.9011e-04 |
| ATR_inlet_M4 | 9.3962e-04 |
| ATR_t_C1 | 0.0318 |

| Name | Value |
|---|---|
| ATR_t_C2 | 0.0378 |
| ATR_t_C3 | 0.0316 |
| ATR_t_C4 | 0.0378 |
| ATR_t_M1 | 0.0212 |
| ATR_t_M2 | 0.0252 |
| ATR_t_M3 | 0.0211 |
| ATR_t_M4 | 0.0252 |
| c_exit_C1 | 279.8425 |
| c_exit_C2 | 355.3717 |
| c_exit_C3 | 276.1701 |
| c_exit_C4 | 354.1225 |
| c_exit_M1 | 268.4813 |
| c_exit_M2 | 342.5984 |
| c_exit_M3 | 264.7114 |
| c_exit_M4 | 341.3095 |
| c_inlet_C1 | 578.9003 |
| c_inlet_C2 | 676.1301 |
| c_inlet_C3 | 580.5411 |
| c_inlet_C4 | 676.6525 |
| c_inlet_M1 | 578.9003 |
| c_inlet_M2 | 676.1301 |
| c_inlet_M3 | 580.5411 |
| c_inlet_M4 | 676.6525 |
| C_p_inlet_C1 | 37.2690 |
| C_p_inlet_C2 | 38.7140 |
| C_p_inlet_C3 | 37.9320 |
| C_p_inlet_C4 | 38.8470 |
| C_p_inlet_M1 | 37.2690 |
| C_p_inlet_M2 | 38.7140 |
| C_p_inlet_M3 | 37.9320 |
| C_p_inlet_M4 | 38.8470 |
| C_p_t_C1 | 36.7700 |
| C_p_t_C2 | 37.5344 |
| C_p_t_C3 | 37.6204 |
| C_p_t_C4 | 37.6531 |
| C_p_t_M1 | 36.7700 |
| C_p_t_M2 | 37.5344 |
| C_p_t_M3 | 37.6204 |
| C_p_t_M4 | 37.6531 |
| c_t_C1 | 539.1125 |
| c_t_C2 | 635.6329 |

| Name | Value |
|---|---|
| c_t_C3 | 539.7785 |
| c_t_C4 | 635.8076 |
| c_t_M1 | 539.1125 |
| c_t_M2 | 635.6329 |
| c_t_M3 | 539.7785 |
| c_t_M4 | 635.8076 |
| C_v_inlet_C1 | 28.2290 |
| C_v_inlet_C2 | 30.2120 |
| C_v_inlet_C3 | 28.5690 |
| C_v_inlet_C4 | 30.2690 |
| C_v_inlet_M1 | 28.2290 |
| C_v_inlet_M2 | 30.2120 |
| C_v_inlet_M3 | 28.5690 |
| C_v_inlet_M4 | 30.2690 |
| C_v_t_C1 | 27.6814 |
| C_v_t_C2 | 29.0546 |
| C_v_t_C3 | 28.1607 |
| C_v_t_C4 | 29.1052 |
| C_v_t_M1 | 27.6814 |
| C_v_t_M2 | 29.0546 |
| C_v_t_M3 | 28.1607 |
| C_v_t_M4 | 29.1052 |
| D | 1.0000e-04 |
| DN_exit_C1 | 212.7254 |
| DN_exit_C2 | 137.7514 |
| DN_exit_C3 | 303.2321 |
| DN_exit_C4 | 193.7875 |
| DN_exit_M1 | 102.6515 |
| DN_exit_M2 | 66.1515 |
| DN_exit_M3 | 146.4622 |
| DN_exit_M4 | 93.0844 |
| DN_inlet_C1 | 101.3588 |
| DN_inlet_C2 | 72.1246 |
| DN_inlet_C3 | 141.3385 |
| DN_inlet_C4 | 100.8789 |
| DN_inlet_M1 | 20.1109 |
| DN_inlet_M2 | 14.3104 |
| DN_inlet_M3 | 28.0434 |
| DN_inlet_M4 | 20.0157 |
| DN_t_C1 | 545.8621 |
| DN_t_C2 | 385.1589 |

| Name | Value |
|---|---|
| DN_t_C3 | 762.3985 |
| DN_t_C4 | 538.9466 |
| DN_t_M1 | 363.9080 |
| DN_t_M2 | 256.7726 |
| DN_t_M3 | 508.2657 |
| DN_t_M4 | 359.2977 |
| eqn | *1x1 sym* |
| F_C1 | 0.0025 |
| F_C2 | 0.0025 |
| F_C3 | 0.0035 |
| F_C4 | 0.0035 |
| F_M1 | 0.0022 |
| F_M2 | 0.0022 |
| F_M3 | 0.0031 |
| F_M4 | 0.0031 |
| Gamma_C1 | 0.6709 |
| Gamma_C2 | 0.6639 |
| Gamma_C3 | 0.6722 |
| Gamma_C4 | 0.6642 |
| gamma_inlet_C1 | 1.3202 |
| gamma_inlet_C2 | 1.2814 |
| gamma_inlet_C3 | 1.3277 |
| gamma_inlet_C4 | 1.2834 |
| gamma_inlet_M1 | 1.3202 |
| gamma_inlet_M2 | 1.2814 |
| gamma_inlet_M3 | 1.3277 |
| gamma_inlet_M4 | 1.2834 |
| Gamma_M1 | 0.6709 |
| Gamma_M2 | 0.6639 |
| Gamma_M3 | 0.6722 |
| Gamma_M4 | 0.6642 |
| gamma_t_C1 | 1.3283 |
| gamma_t_C2 | 1.2919 |
| gamma_t_C3 | 1.3359 |
| gamma_t_C4 | 1.2937 |
| gamma_t_M1 | 1.3283 |
| gamma_t_M2 | 1.2919 |
| gamma_t_M3 | 1.3359 |
| gamma_t_M4 | 1.2937 |
| H_exit | 3.0000e-04 |
| H_inlet | 3.0000e-04 |

| Name | Value |
|---|---|
| H_t | 6.0000e-05 |
| HD_exit | 1.7778e-04 |
| HD_inlet | 1.9048e-04 |
| HD_t | 4.0000e-05 |
| I_sp_C1 | 133.8721 |
| I_sp_C2 | 162.8660 |
| I_sp_C3 | 133.2313 |
| I_sp_C4 | 162.6444 |
| I_sp_M1 | 134.9964 |
| I_sp_M2 | 164.4151 |
| I_sp_M3 | 134.3234 |
| I_sp_M4 | 164.1816 |
| IDTR_exit_C1 | 0.0018 |
| IDTR_exit_C2 | 0.0033 |
| IDTR_exit_C3 | 0.0012 |
| IDTR_exit_C4 | 0.0024 |
| IDTR_exit_M1 | 0.0022 |
| IDTR_exit_M2 | 0.0042 |
| IDTR_exit_M3 | 0.0015 |
| IDTR_exit_M4 | 0.0030 |
| IDTR_inlet_C1 | 3.9444e-05 |
| IDTR_inlet_C2 | 6.5722e-05 |
| IDTR_inlet_C3 | 2.8174e-05 |
| IDTR_inlet_C4 | 4.6944e-05 |
| IDTR_inlet_M1 | 3.9444e-05 |
| IDTR_inlet_M2 | 6.5722e-05 |
| IDTR_inlet_M3 | 2.8174e-05 |
| IDTR_inlet_M4 | 4.6944e-05 |
| IDTR_t_C1 | 5.8231e-05 |
| IDTR_t_C2 | 9.8240e-05 |
| IDTR_t_C3 | 4.1495e-05 |
| IDTR_t_C4 | 7.0127e-05 |
| IDTR_t_M1 | 5.8231e-05 |
| IDTR_t_M2 | 9.8240e-05 |
| IDTR_t_M3 | 4.1495e-05 |
| IDTR_t_M4 | 7.0127e-05 |
| Kn_exit_C1 | 0.0141 |
| Kn_exit_C2 | 0.0205 |
| Kn_exit_C3 | 0.0100 |
| Kn_exit_C4 | 0.0146 |
| Kn_exit_M1 | 0.0309 |

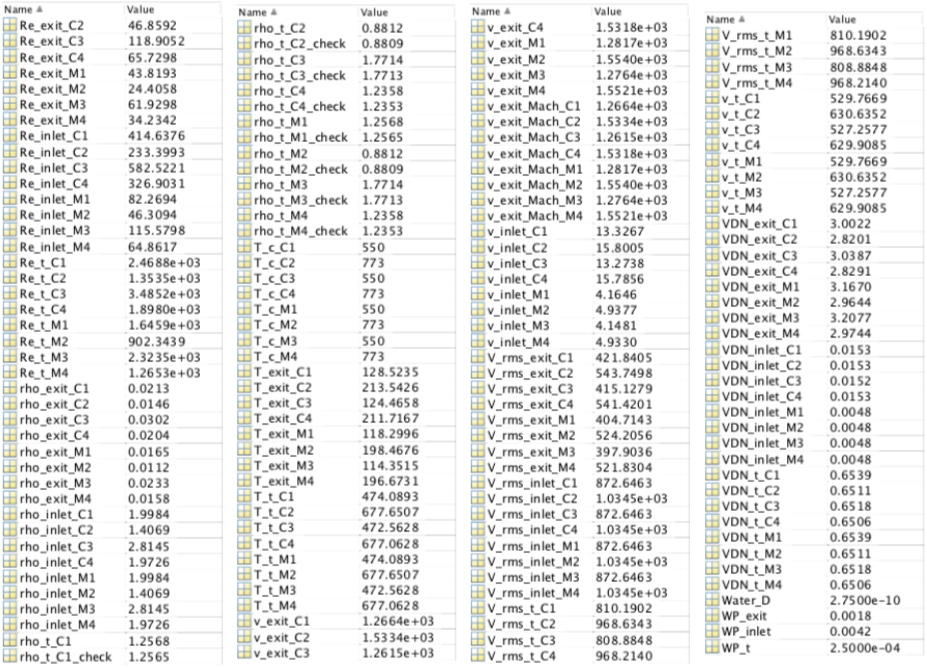
| Name | Value |
|---|---|
| Kn_exit_M2 | 0.0448 |
| Kn_exit_M3 | 0.0219 |
| Kn_exit_M4 | 0.0320 |
| Kn_inlet_C1 | 1.5067e-04 |
| Kn_inlet_C2 | 2.1176e-04 |
| Kn_inlet_C3 | 1.0762e-04 |
| Kn_inlet_C4 | 1.5126e-04 |
| Kn_inlet_M1 | 2.3730e-04 |
| Kn_inlet_M2 | 3.3352e-04 |
| Kn_inlet_M3 | 1.6950e-04 |
| Kn_inlet_M4 | 2.3823e-04 |
| Kn_t_C1 | 0.0012 |
| Kn_t_C2 | 0.0017 |
| Kn_t_C3 | 8.5498e-04 |
| Kn_t_C4 | 0.0012 |
| Kn_t_M1 | 0.0018 |
| Kn_t_M2 | 0.0025 |
| Kn_t_M3 | 0.0013 |
| Kn_t_M4 | 0.0018 |
| M_exit_C1 | 4.5256 |
| M_exit_C2 | 4.3150 |
| M_exit_C3 | 4.5677 |
| M_exit_C4 | 4.3255 |
| M_exit_M1 | 4.7740 |
| M_exit_M2 | 4.5358 |
| M_exit_M3 | 4.8217 |
| M_exit_M4 | 4.5476 |
| M_inlet_C1 | 0.0230 |
| M_inlet_C2 | 0.0234 |
| M_inlet_C3 | 0.0229 |
| M_inlet_C4 | 0.0233 |
| M_inlet_M1 | 0.0072 |
| M_inlet_M2 | 0.0073 |
| M_inlet_M3 | 0.0071 |
| M_inlet_M4 | 0.0073 |
| M_t_C1 | 0.9827 |
| M_t_C2 | 0.9921 |
| M_t_C3 | 0.9768 |
| M_t_C4 | 0.9907 |
| M_t_M1 | 0.9827 |
| M_t_M2 | 0.9921 |

| Name | Value |
|---|---|
| M_t_M3 | 0.9768 |
| M_t_M4 | 0.9907 |
| M_VDN_exit_C1 | 4.5256 |
| M_VDN_exit_C2 | 4.3150 |
| M_VDN_exit_C3 | 4.5677 |
| M_VDN_exit_C4 | 4.3255 |
| M_VDN_exit_M1 | 4.7740 |
| M_VDN_exit_M2 | 4.5358 |
| M_VDN_exit_M3 | 4.8217 |
| M_VDN_exit_M4 | 4.5476 |
| M_VDN_inlet_C1 | 0.0230 |
| M_VDN_inlet_C2 | 0.0234 |
| M_VDN_inlet_C3 | 0.0229 |
| M_VDN_inlet_C4 | 0.0233 |
| M_VDN_inlet_M1 | 0.0072 |
| M_VDN_inlet_M2 | 0.0073 |
| M_VDN_inlet_M3 | 0.0071 |
| M_VDN_inlet_M4 | 0.0073 |
| M_VDN_t_C1 | 0.9857 |
| M_VDN_t_C2 | 0.9962 |
| M_VDN_t_C3 | 0.9798 |
| M_VDN_t_C4 | 0.9947 |
| M_VDN_t_M1 | 0.9857 |
| M_VDN_t_M2 | 0.9962 |
| M_VDN_t_M3 | 0.9798 |
| M_VDN_t_M4 | 0.9947 |
| M_w_C1 | 0.0180 |
| M_w_C1_g | 18.0153 |
| M_w_C2 | 0.0180 |
| M_w_C2_g | 18.0153 |
| M_w_C3 | 0.0180 |
| M_w_C3_g | 18.0153 |
| M_w_C4 | 0.0180 |
| M_w_C4_g | 18.0153 |
| M_w_M1 | 0.0180 |
| M_w_M1_g | 18.0153 |
| M_w_M2 | 0.0180 |
| M_w_M2_g | 18.0153 |
| M_w_M3 | 0.0180 |
| M_w_M3_g | 18.0153 |
| M_w_M4 | 0.0180 |

| Name | Value |
|---|---|
| M_w_M4_g | 18.0153 |
| mdot_C1 | 1.8825e-06 |
| mdot_C2 | 1.5713e-06 |
| mdot_C3 | 2.6408e-06 |
| mdot_C4 | 2.2011e-06 |
| mdot_M1 | 1.6645e-06 |
| mdot_M2 | 1.3894e-06 |
| mdot_M3 | 2.3349e-06 |
| mdot_M4 | 1.9462e-06 |
| MFP_exit_C1 | 4.2339e-06 |
| MFP_exit_C2 | 6.1418e-06 |
| MFP_exit_C3 | 3.0063e-06 |
| MFP_exit_C4 | 4.3797e-06 |
| MFP_exit_M1 | 5.4847e-06 |
| MFP_exit_M2 | 7.9667e-06 |
| MFP_exit_M3 | 3.8936e-06 |
| MFP_exit_M4 | 5.6807e-06 |
| MFP_inlet_C1 | 4.5201e-08 |
| MFP_inlet_C2 | 6.3528e-08 |
| MFP_inlet_C3 | 3.2286e-08 |
| MFP_inlet_C4 | 4.5377e-08 |
| MFP_inlet_M1 | 4.5201e-08 |
| MFP_inlet_M2 | 6.3528e-08 |
| MFP_inlet_M3 | 3.2286e-08 |
| MFP_inlet_M4 | 4.5377e-08 |
| MFP_t_C1 | 7.1873e-08 |
| MFP_t_C2 | 1.0142e-07 |
| MFP_t_C3 | 5.1299e-08 |
| MFP_t_C4 | 7.2429e-08 |
| MFP_t_M1 | 7.1873e-08 |
| MFP_t_M2 | 1.0142e-07 |
| MFP_t_M3 | 5.1299e-08 |
| MFP_t_M4 | 7.2429e-08 |
| mu_inlet_C1 | 1.9269e-05 |
| mu_inlet_C2 | 2.8573e-05 |
| mu_inlet_C3 | 1.9240e-05 |
| mu_inlet_C4 | 2.8576e-05 |
| mu_inlet_M1 | 1.9269e-05 |
| mu_inlet_M2 | 2.8573e-05 |
| mu_inlet_M3 | 1.9240e-05 |
| mu_inlet_M4 | 2.8576e-05 |

| Name | Value |
|---|---|
| mu_t_C1 | 1.6181e-05 |
| mu_t_C2 | 2.4636e-05 |
| mu_t_C3 | 1.6079e-05 |
| mu_t_C4 | 2.4609e-05 |
| mu_t_M1 | 1.6181e-05 |
| mu_t_M2 | 2.4636e-05 |
| mu_t_M3 | 1.6079e-05 |
| mu_t_M4 | 2.4609e-05 |
| N_A | 6.0221e+23 |
| p_0_C1 | 0 |
| p_0_C2 | 0 |
| p_0_C3 | 0 |
| p_0_C4 | 0 |
| p_0_M1 | 0 |
| p_0_M2 | 0 |
| p_0_M3 | 0 |
| p_0_M4 | 0 |
| p_c_C1 | 500000 |
| p_c_C1_bar | 5 |
| p_c_C2 | 500000 |
| p_c_C2_bar | 5 |
| p_c_C3 | 700000 |
| p_c_C3_bar | 7 |
| p_c_C4 | 700000 |
| p_c_C4_bar | 7 |
| p_c_M1 | 500000 |
| p_c_M1_bar | 5 |
| p_c_M2 | 500000 |
| p_c_M2_bar | 5 |
| p_c_M3 | 700000 |
| p_c_M3_bar | 7 |
| p_c_M4 | 700000 |
| p_c_M4_bar | 7 |
| p_exit_C1 | 1.2474e+03 |
| p_exit_C2 | 1.4287e+03 |
| p_exit_C3 | 1.7012e+03 |
| p_exit_C4 | 1.9864e+03 |
| p_exit_M1 | 886.2991 |
| p_exit_M2 | 1.0237e+03 |
| p_exit_M3 | 1.2068e+03 |
| p_exit_M4 | 1.4226e+03 |

| Name | Value |
|---|---|
| p_t_C1 | 2.7105e+05 |
| p_t_C2 | 2.7455e+05 |
| p_t_C3 | 3.7854e+05 |
| p_t_C4 | 3.8412e+05 |
| p_t_M1 | 2.7105e+05 |
| p_t_M2 | 2.7455e+05 |
| p_t_M3 | 3.7854e+05 |
| p_t_M4 | 3.8412e+05 |
| R_A_C1 | 8.3145 |
| R_A_C2 | 8.3145 |
| R_A_C3 | 8.3145 |
| R_A_C4 | 8.3145 |
| R_A_M1 | 8.3145 |
| R_A_M2 | 8.3145 |
| R_A_M3 | 8.3145 |
| R_A_M4 | 8.3145 |
| Re_DN_exit_C1 | 461.7850 |
| Re_DN_exit_C2 | 299.0312 |
| Re_DN_exit_C3 | 658.2575 |
| Re_DN_exit_C4 | 420.6746 |
| Re_DN_exit_M1 | 222.8362 |
| Re_DN_exit_M2 | 143.6019 |
| Re_DN_exit_M3 | 317.9406 |
| Re_DN_exit_M4 | 202.0680 |
| Re_DN_inlet_C1 | 220.0302 |
| Re_DN_inlet_C2 | 156.5682 |
| Re_DN_inlet_C3 | 306.8181 |
| Re_DN_inlet_C4 | 218.9883 |
| Re_DN_inlet_M1 | 43.6568 |
| Re_DN_inlet_M2 | 31.0651 |
| Re_DN_inlet_M3 | 60.8766 |
| Re_DN_inlet_M4 | 43.4501 |
| Re_DN_t_C1 | 1.1850e+03 |
| Re_DN_t_C2 | 836.1044 |
| Re_DN_t_C3 | 1.6550e+03 |
| Re_DN_t_C4 | 1.1699e+03 |
| Re_DN_t_M1 | 789.9730 |
| Re_DN_t_M2 | 557.4029 |
| Re_DN_t_M3 | 1.1033e+03 |
| Re_DN_t_M4 | 779.9648 |
| Re_exit_C1 | 84.1330 |



| Name | Value |
|---|---|
| Re_exit_C2 | 46.8592 |
| Re_exit_C3 | 118.9052 |
| Re_exit_C4 | 65.7298 |
| Re_exit_M1 | 43.8193 |
| Re_exit_M2 | 24.4058 |
| Re_exit_M3 | 61.9298 |
| Re_exit_M4 | 34.2342 |
| Re_inlet_C1 | 414.6376 |
| Re_inlet_C2 | 233.3993 |
| Re_inlet_C3 | 582.5221 |
| Re_inlet_C4 | 326.9031 |
| Re_inlet_M1 | 82.2694 |
| Re_inlet_M2 | 46.3094 |
| Re_inlet_M3 | 115.5798 |
| Re_inlet_M4 | 64.8617 |
| Re_t_C1 | 2.4688e+03 |
| Re_t_C2 | 1.3535e+03 |
| Re_t_C3 | 3.4852e+03 |
| Re_t_C4 | 1.8980e+03 |
| Re_t_M1 | 1.6459e+03 |
| Re_t_M2 | 902.3439 |
| Re_t_M3 | 2.3235e+03 |
| Re_t_M4 | 1.2653e+03 |
| rho_exit_C1 | 0.0213 |
| rho_exit_C2 | 0.0146 |
| rho_exit_C3 | 0.0302 |
| rho_exit_C4 | 0.0204 |
| rho_exit_M1 | 0.0165 |
| rho_exit_M2 | 0.0112 |
| rho_exit_M3 | 0.0233 |
| rho_exit_M4 | 0.0158 |
| rho_inlet_C1 | 1.9984 |
| rho_inlet_C2 | 1.4069 |
| rho_inlet_C3 | 2.8145 |
| rho_inlet_C4 | 1.9726 |
| rho_inlet_M1 | 1.9984 |
| rho_inlet_M2 | 1.4069 |
| rho_inlet_M3 | 2.8145 |
| rho_inlet_M4 | 1.9726 |
| rho_t_C1 | 1.2568 |
| rho_t_C1_check | 1.2565 |

| Name | Value |
|---|---|
| rho_t_C2 | 0.8812 |
| rho_t_C2_check | 0.8809 |
| rho_t_C3 | 1.7714 |
| rho_t_C3_check | 1.7713 |
| rho_t_C4 | 1.2358 |
| rho_t_C4_check | 1.2353 |
| rho_t_M1 | 1.2568 |
| rho_t_M1_check | 1.2565 |
| rho_t_M2 | 0.8812 |
| rho_t_M2_check | 0.8809 |
| rho_t_M3 | 1.7714 |
| rho_t_M3_check | 1.7713 |
| rho_t_M4 | 1.2358 |
| rho_t_M4_check | 1.2353 |
| T_c_C1 | 550 |
| T_c_C2 | 773 |
| T_c_C3 | 550 |
| T_c_C4 | 773 |
| T_c_M1 | 550 |
| T_c_M2 | 773 |
| T_c_M3 | 550 |
| T_c_M4 | 773 |
| T_exit_C1 | 128.5235 |
| T_exit_C2 | 213.5426 |
| T_exit_C3 | 124.4658 |
| T_exit_C4 | 211.7167 |
| T_exit_M1 | 118.2996 |
| T_exit_M2 | 198.4676 |
| T_exit_M3 | 114.3515 |
| T_exit_M4 | 196.6731 |
| T_t_C1 | 474.0893 |
| T_t_C2 | 677.6507 |
| T_t_C3 | 472.5628 |
| T_t_C4 | 677.0628 |
| T_t_M1 | 474.0893 |
| T_t_M2 | 677.6507 |
| T_t_M3 | 472.5628 |
| T_t_M4 | 677.0628 |
| v_exit_C1 | 1.2664e+03 |
| v_exit_C2 | 1.5334e+03 |
| v_exit_C3 | 1.2615e+03 |

| Name | Value |
|---|---|
| v_exit_C4 | 1.5318e+03 |
| v_exit_M1 | 1.2817e+03 |
| v_exit_M2 | 1.5540e+03 |
| v_exit_M3 | 1.2764e+03 |
| v_exit_M4 | 1.5521e+03 |
| v_exit_Mach_C1 | 1.2664e+03 |
| v_exit_Mach_C2 | 1.5334e+03 |
| v_exit_Mach_C3 | 1.2615e+03 |
| v_exit_Mach_C4 | 1.5318e+03 |
| v_exit_Mach_M1 | 1.2817e+03 |
| v_exit_Mach_M2 | 1.5540e+03 |
| v_exit_Mach_M3 | 1.2764e+03 |
| v_exit_Mach_M4 | 1.5521e+03 |
| v_inlet_C1 | 13.3267 |
| v_inlet_C2 | 15.8005 |
| v_inlet_C3 | 13.2738 |
| v_inlet_C4 | 15.7856 |
| v_inlet_M1 | 4.1646 |
| v_inlet_M2 | 4.9377 |
| v_inlet_M3 | 4.1481 |
| v_inlet_M4 | 4.9330 |
| V_rms_exit_C1 | 421.8405 |
| V_rms_exit_C2 | 543.7498 |
| V_rms_exit_C3 | 415.1279 |
| V_rms_exit_C4 | 541.4201 |
| V_rms_exit_M1 | 404.7143 |
| V_rms_exit_M2 | 524.2056 |
| V_rms_exit_M3 | 397.9036 |
| V_rms_exit_M4 | 521.8304 |
| V_rms_inlet_C1 | 872.6463 |
| V_rms_inlet_C2 | 1.0345e+03 |
| V_rms_inlet_C3 | 872.6463 |
| V_rms_inlet_C4 | 1.0345e+03 |
| V_rms_inlet_M1 | 872.6463 |
| V_rms_inlet_M2 | 1.0345e+03 |
| V_rms_inlet_M3 | 872.6463 |
| V_rms_inlet_M4 | 1.0345e+03 |
| V_rms_t_C1 | 810.1902 |
| V_rms_t_C2 | 968.6343 |
| V_rms_t_C3 | 808.8848 |
| V_rms_t_C4 | 968.2140 |

| Name | Value |
|---|---|
| V_rms_t_M1 | 810.1902 |
| V_rms_t_M2 | 968.6343 |
| V_rms_t_M3 | 808.8848 |
| V_rms_t_M4 | 968.2140 |
| v_t_C1 | 529.7669 |
| v_t_C2 | 630.6352 |
| v_t_C3 | 527.2577 |
| v_t_C4 | 629.9085 |
| v_t_M1 | 529.7669 |
| v_t_M2 | 630.6352 |
| v_t_M3 | 527.2577 |
| v_t_M4 | 629.9085 |
| VDN_exit_C1 | 3.0022 |
| VDN_exit_C2 | 2.8201 |
| VDN_exit_C3 | 3.0387 |
| VDN_exit_C4 | 2.8291 |
| VDN_exit_M1 | 3.1670 |
| VDN_exit_M2 | 2.9644 |
| VDN_exit_M3 | 3.2077 |
| VDN_exit_M4 | 2.9744 |
| VDN_inlet_C1 | 0.0153 |
| VDN_inlet_C2 | 0.0153 |
| VDN_inlet_C3 | 0.0152 |
| VDN_inlet_C4 | 0.0153 |
| VDN_inlet_M1 | 0.0048 |
| VDN_inlet_M2 | 0.0048 |
| VDN_inlet_M3 | 0.0048 |
| VDN_inlet_M4 | 0.0048 |
| VDN_t_C1 | 0.6539 |
| VDN_t_C2 | 0.6511 |
| VDN_t_C3 | 0.6518 |
| VDN_t_C4 | 0.6506 |
| VDN_t_M1 | 0.6539 |
| VDN_t_M2 | 0.6511 |
| VDN_t_M3 | 0.6518 |
| VDN_t_M4 | 0.6506 |
| Water_D | 2.7500e-10 |
| WP_exit | 0.0018 |
| WP_inlet | 0.0042 |
| WP_t | 2.5000e-04 |

# Bibliography


[1] URL http://sci.esa.int/sci-ft/50124-technology-readiness-level/.

[2] URL https://www.nanosats.eu/.

[3] URL https://cfd.direct/openfoam/user-guide/.

[4] URL https://www.openfoam.com/documentation/user-guide/.

[5] URL https://www.paraview.org/Wiki/ParaView.

[6] URL https://www.tudelft.nl/en/ae/organisation/departments/space-engineering/space-systems-engineering/research/miniaturization/micro-propulsion/.

[7] *Basics of Space Flight, Chapter 11. Typical Onboard Systems, Electrical Power Supply and Distribution Subsystems*. NASA JPL Publication, 2008.

[8] G Abbate. *Multi-Scale Modeling of Gas Flows with Continuum-Rarefied Transitions Application to Expanding Gas Jets in Thin Film Deposition Processes*. PhD thesis, TU Delft, Delft University of Technology, 2009.

[9] Alina A Alexeenko. Modeling of microscale gas flows using the direct simulation monte carlo method. 2003.

[10] Timothy J Bartel, Steve Plimpton, and Michael A Gallis. Icarus: A 2-d direct simulation monte carlo (dsmc) code for multi-processor computers. *Sandia National Laboratories Report SAND2001-2901 (October, 2001)*, 2001.

[11] Robert L Bayt. Analysis, fabrication and testing of a mems-based micropropulsion system. Technical report, Aerospace Computational Design Laboratory, Dept. of Aeronautics & Astronautics, Massachusetts Institute of Technology, 1999.

[12] Johan Bejhed. *Fluidic Microsystems for Micropropulsion Applications in Space*. PhD thesis, Acta Universitatis Upsaliensis, 2006.

[13] G. A. Bird. *The DSMC Method*. CreateSpace, version 1.2 edition, 2013.

[14] GA Bird. Molecular gas dynamics and the direct simulation of gas flows (oxford engineering science series). *Clarendon,*, 1994.

[15] JW Cen and JL Xu. Performance evaluation and flow visualization of a mems based vaporizing liquid micro-thruster. *Acta Astronautica*, 67(3-4):468–482, 2010.

[16] A Cervone, B Zandbergen, DC Guerrieri, M De Athayde Costa e Silva, I Krusharev, and H van Zeijl. Green micro-resistojet research at delft university of technology: new options for cubesat propulsion. *CEAS Space Journal*, 9(1):111–125, 2017.

[17] KH Cheah and JK Chin. Performance improvement on mems micropropulsion system through a novel two-depth micronozzle design. *Acta Astronautica*, 69(1-2):59–70, 2011.

[18] Masoud Darbandi and Ehsan Roohi. Applying a hybrid dsmc/navier–stokes frame to explore the effect of splitter catalyst plates in micro/nanopropulsion systems. *Sensors and actuators A: Physical*, 189:409–419, 2013.

[19] Amin Ebrahimi and Ehsan Roohi. Dsmc investigation of rarefied gas flow through diverging micro- and nanochannels. *Microfluidics and Nanofluidics*, 21(2):18, 2017.

[20] B. G. Evans, P. T. Thompson, G. E. Corazza, A. Vanelli-Coralli, and E. A. Candreva. 1945–2010: 65 years of satellite history from early visions to latest missions. *Proceedings of the IEEE*, 99(11): 1840–1857, Nov 2011. ISSN 0018-9219. doi: 10.1109/JPROC.2011.2159467.

[21] MA Gallis, NP Bitter, TP Koehler, JR Torczynski, SJ Plimpton, and G Papadakis. Molecular-level simulations of turbulence and its decay. *Physical review letters*, 118(6):064501, 2017.

[22] J Gomez and R Groll. Pressure drop and thrust predictions for transonic micronozzle flows. *Physics of Fluids*, 28(2):022008, 2016.

[23] Martin Grabe. Numerical simulation of nitrogen nozzle expansion using kinetic and continuum approaches. In *59th International Astronautical Conference*, September 2008. URL https://elib.dlr.de/55152/.

[24] Christopher J Greenshields, Henry G Weller, Luca Gasparini, and Jason M Reese. Implementation of semi-discrete, non-staggered central schemes in a colocated, polyhedral, finite volume framework, for high-speed viscous flows. *International journal for numerical methods in fluids*, 63 (1):1–21, 2010.

[25] Joel Guerrero. Running in parallel. URL http://www.wolfdynamics.com/wiki/parallel.pdf.

[26] Daduí C Guerrieri, Angelo Cervone, and Eberhard Gill. Analysis of nonisothermal rarefied gas flow in diverging microchannels for low-pressure microresistojets. *Journal of Heat Transfer*, 138 (11):112403, 2016.

[27] Daduí C Guerrieri, Marsil AC Silva, Angelo Cervone, and Eberhard Gill. Selection and characterization of green propellants for micro-resistojets. *Journal of Heat Transfer*, 139(10):102001, 2017.

[28] Nicolas G. Hadjiconstantinou, Alejandro L. Garcia, Martin Z. Bazant, and Gang He. Statistical error in particle simulations of hydrodynamic phenomena. *Journal of Computational Physics*, 187(1): 274 – 297, 2003. ISSN 0021-9991. doi: https://doi.org/10.1016/S0021-9991(03)00099-8. URL http://www.sciencedirect.com/science/article/pii/S0021999103000998.

[29] M Ivanov, G Markelov, A Ketsdever, and D Wadsworth. Numerical study of cold gas micronozzle flows. In *37th Aerospace Sciences Meeting and Exhibit*, page 166, 1999.

[30] Andre Jackson. A comprehensive tour of snappyhexmesh. In *7th OpenFOAM workshop*, 2012.

[31] Patrick Jenny, Manuel Torrilhon, and Stefan Heinz. A solution algorithm for the fluid dynamic equations based on a stochastic model for molecular motion. *Journal of computational physics*, 229(4):1077–1098, 2010.

[32] Gottfried Konecny. Small satellites–a tool for earth observation? In *XXth ISPRS Congress, Commission*, volume 4, pages 12–23, 2004.

[33] Pijus Kundu, Tarun Kanti Bhattacharyya, and Soumen Das. Design, fabrication and performance evaluation of a vaporizing liquid microthruster. *Journal of Micromechanics and Microengineering*, 22(2):025016, 2012.

[34] Igor Levchenko, Kateryna Bazaka, Yongjie Ding, Yevgeny Raitses, Stéphane Mazouffre, Torsten Henning, Peter J Klar, Shunjiro Shinohara, Jochen Schein, Laurent Garrigues, et al. Space micropropulsion systems for cubesats and small satellites: From proximate targets to furthermost frontiers. *Applied Physics Reviews*, 5(1):011104, 2018.

[35] William W Liou and Yichuan Fang. Heat transfer in microchannel devices using dsmc. *Journal of Microelectromechanical Systems*, 10(2):274–279, 2001.

[36] MC Louwerse, Henricus V Jansen, MNW Groenendijk, and Michael Curt Elwenspoek. Nozzle fabrication for micropropulsion of a microsatellite. *Journal of micromechanics and microengineering*, 19(4):045008, 2009.

[37] LF Gutiérrez Marcantoni, José P Tamagno, and SA Elaskar. High speed flow simulation using openfoam. *Mecánica Computacional*, 31:2939–2959, 2012.

[38] DK Maurya, S Das, and SK Lahiri. An analytical model of a silicon mems vaporizing liquid microthruster and some experimental studies. *Sensors and Actuators A: Physical*, 122(1):159–166, 2005.

[39] Robert Osiander, M Ann Garrison Darrin, and John L Champion. *MEMS and microstructures in aerospace applications*. CRC press, 2005.

[40] Jose Padilla and Iain Boyd. Assessment of gas-surface interaction models in dsmc analysis of rarefied hypersonic flow. In *39th AIAA Thermophysics Conference*, page 3891, 2007.

[41] Jose F Padilla and Iain D Boyd. Assessment of gas-surface interaction models for computation of rarefied hypersonic flow. *Journal of Thermophysics and Heat Transfer*, 23(1):96–105, 2009.

[42] Vidhya Pallichadath, Marsil AC Silva, Daduí Cordeiro Guerrieri, Angelo Cervone, and Silvana Radu. Integration and miniaturization challenges in the design of micro-propulsion systems for picosatellite platforms. *Spacepropulsion 2018 by 3AF*, 2018.

[43] Kristoffer Palmer, Ernesto Vargas Catalan, Ville Lekholm, and Greger Thornell. Investigation of exhausts from fabricated silicon micronozzles with rectangular and close to rotationally symmetric cross-sections. *Journal of Micromechanics and Microengineering*, 23(10):105001, 2013.

[44] François Pellegrini and Jean Roman. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In *International Conference on High-Performance Computing and Networking*, pages 493–498. Springer, 1996.

[45] PS Prasanth and Jose K Kakkassery. Molecular models for simulation of rarefied gas flows using direct simulation monte carlo method. *Fluid dynamics research*, 40(4):233, 2008.

[46] Steve Price, Tony Phillips, and Gil Knier. Staying cool on the iss. URL https://science.nasa.gov/science-news/science-at-nasa/2001/ast21mar_1.

[47] T.J. Scanlon, E. Roohi, C. White, M. Darbandi, and J.M. Reese. An open source, parallel dsmc code for rarefied gas flows in arbitrary geometries. *Computers & Fluids*, 39(10):2078 – 2089, 2010. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2010.07.014. URL http://www.sciencedirect.com/science/article/pii/S0045793010001891.

[48] C Shu, XH Mao, and YT Chew. Particle number per cell and scaling factor effect on accuracy of dsmc simulation of micro flows. *International Journal of Numerical Methods for Heat & Fluid Flow*, 15(8):827–841, 2005.

[49] Marsil AC Silva, Daduí C Guerrieri, Angelo Cervone, and Eberhard Gill. A review of mems micropropulsion technologies for cubesats and pocketqubes. *Acta Astronautica*, 2017.

[50] Marsil AC Silva, Daduí C Guerrieri, Henk van Zeijl, Angelo Cervone, and Eberhard Gill. Vaporizing liquid microthrusters with integrated heaters and temperature measurement. *Sensors and Actuators A: Physical*, 265:261–274, 2017.

[51] MR Wang and ZX Li. Numerical simulations on performance of mems-based nozzles at moderate or low temperatures. *Microfluidics and Nanofluidics*, 1(1):62–70, 2004.

[52] C. White, M.K. Borg, T.J. Scanlon, S.M. Longshaw, B. John, D.R. Emerson, and J.M. Reese. dsmcfoam+: An openfoam based direct simulation monte carlo solver. *Computer Physics Communications*, 224:22 – 43, 2018. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2017.09.030. URL http://www.sciencedirect.com/science/article/pii/S0010465517303375.

[53] Craig White, Matthew K Borg, Thomas J Scanlon, and Jason M Reese. Accounting for rotational non-equilibrium effects in subsonic dsmc boundary conditions. In *Journal of Physics: Conference Series*, volume 362, page 012016. IOP Publishing, 2012.

[54] Craig White, Matthew K Borg, Thomas J Scanlon, and Jason M Reese. A dsmc investigation of gas flows in micro-channels with bends. *Computers & Fluids*, 71:261–271, 2013.