Design Process of a Gas Nozzle for Laser Wakefield Acceleration

NSERC USRA – Summer 2026

Luke Concini

May 1^{st} – August 28^{th}, 2026

Supervisor: Dr. Amina Hussein
Mentors: Dr. Nicholas Beier, Abdulhakeem Ojonoka Yusuf

Abstract

With the advent of ultrashort, high-intensity laser pulses in the optical range, the previously-theorized blowout regime of laser wakefield acceleration (LWFA) emerged as a contender on the particle accelerator scene. Capable of sustaining accelerating gradients over 1000 times greater than that of a conventional RF accelerator, LWFA has the potential to drastically shrink the spatial and financial requirements to perform high-energy particle physics research. LWFA requires a gas target of well-defined density and spatial extent in order to produce a plasma meeting the sensitive, laser system-specific conditions to achieve LWFA. The project involves the development of a design process for custom gas jets capable of facilitating future LWFA experiments. This process involves a review of current LWFA scaling laws to tailor parameters to a given laser system, Ansys Fluent computational fluid dynamics simulations to evaluate de Laval nozzle parameter causality relationships, and WarpX particle-in-cell simulations to understand gas profile influence on LWFA interactions. This report also investigates resin printing as a resource-friendly method of manufacturing a gas nozzle, and experimentally validates a prototype with interferometric techniques. In benchmarking our experimental interferometry results against our Ansys Fluent simulations, we found some agreement between the predicted and produced plumes. However, the lack of precise agreement in exhaust plume contour and density gradient shape exposes the manufacturing challenges in creating a high-quality nozzle. This opens the door for future work testing and optimizing different manufacturing methods to obtain higher-fidelity nozzles, alongside existing pathways to improve particle-in-cell modelling for the design of more complex nozzle geometries.

CONTENTS

I. Introduction

This project involves the design, simulation, and experimental validation of a gas nozzle for use in Laser Wakefield Acceleration (LWFA) experiments.

In LWFA, an ultra-short, high-intensity pulse interacts with a plasma, driving a wake of plasma waves. Electrons 'surf' on these electromagnetic waves, attaining high energies over short distances to form an electron beam, which can be used for high-energy electron and x-ray research. This process has been successfully demonstrated on laser systems ranging from gigawatt, high repetition rate (kHz) to single-shot, petawatt lasers.

A precise gas cloud, of well-defined density profile and spatial extent, is required to form the plasma used in LWFA. While the experiment is executed in vacuum, there are various ways to inject this gas cloud into the path of the laser, from fixed gas cells to custom-designed gas nozzles. Each laser system requires a specifically-tailored gas cloud, which can be manipulated to change the injection and acceleration of electrons in the plasma. One of the largest challenges facing LWFA is repeatability, as produced electron beams are subject to variability in spatial coherence and energies. The quality of the gas cloud dictates LWFA electron beam output, and as such is a significant research focus.

We explore the process of designing a gas nozzle for a terawatt laser system, hoping to facilitate future design and optimization work. This project offered exposure to synthesizing complex academic literature, performing advanced computational fluid dynamics and particle-in-cell electromagnetic simulations, developing comprehensive Python data analysis frameworks, leading optical experiment design, and more. This report offers a view of the theoretical and technical processes used to design and validate a prototype, from theoretical scaling laws to nozzle manufacturing to interferometric analysis.

II. LWFA Scaling Laws

The parameters of the laser system used to perform LWFA dictates the required plasma parameters. We can use heuristic scaling laws to estimate the optimal parameters for the lab's terawatt laser.

A. Theory

A plasma is a grouping of electromagnetically-coupled electrons and much heavier positive ions. If an electron cloud is displaced within the net neutral plasma, a restoring electric force is generated. The more massive ions will pull the electron cloud back towards equilibrium, which will overshoot and cause oscillations. The frequency of electron oscillation ω_p in this phenomenon, called *Langmuir waves*, is determined strictly by the electron density n_e, with appropriate physical constants. The plasma frequency is defined as:

$$\omega_p = \sqrt{\frac{n_e e^2}{\epsilon_0 m_e}} \implies n_e = \frac{\epsilon_0 m_e \omega_p^2}{e^2} \tag{1}$$

When a laser is incident upon a plasma, its EM fields interact with the charged particles. The resulting interaction depends on whether the frequency of the incoming light is greater than the plasma frequency. Since ω_p is characterized by the electron density, we define a critical electron density n_c to sit on the demarcating value of the incoming light frequency ω_0:

$$n_c = \frac{\epsilon_0 m_e \omega_0^2}{e^2} \tag{2}$$

Underdense plasmas obey $n_e < n_c$, typically satisfying $\omega_0 \gg \omega_p$, and have the property that they remain transparent to an incoming laser pulse. In contrast, light incident on an *overdense plasma* will in general be reflected or absorbed.

Recall that the *normalized laser vector potential* a_0 is a dimensionless measure of the relativistic strength of a laser's field:

$$a_0 = \frac{eE_0}{m_e c\omega_0} = \frac{e\lambda_0}{m_e \pi}\sqrt{\frac{I}{2\epsilon_0 c^5}} \tag{3}$$

$$\approx 0.855 \cdot \lambda_0\,[\mu m] \cdot 10^{-9}\sqrt{I\,[W \cdot cm^{-2}]}. \tag{4}$$

Where ω_0 is the angular frequency of the laser and E_0 is the electric field amplitude of the pulse. In the second form, $\lambda_0\,(\mu m)$ is the laser wavelength and $I\,(W \cdot cm^{-2})$ is the peak intensity of the pulse.

Before chirped pulse amplification (CPA) was demonstrated in the optical range, scientists lacked the ability to create ultra-short high-intensity laser pulses. The proposed method of driving a plasma wave used the interference of two longer laser pulses, whose beat frequency resonated with the plasma frequency ω_p. Called Plasma Beatwave Acceleration (PBWA), this laser-plasma interaction would gradually build up large-amplitude density ripples in the plasma, forming the plasma wake required to support hyper-relativistic acceleration gradients [1].

CPA was eventually adapted to the optical range, in which a laser pulse is spread out temporally and spectrally, amplified, and then recombined into a high-intensity short pulse. The fast and intense pulse allows for a different mechanism of building a plasma wake, LWFA. In general, LWFA permits higher and monochromatic electron energies, and improved beam quality.

In LWFA, a laser pulse propagates through an underdense plasma and interacts with electrons in the ion cloud. This microscopic interaction of the pulse's fields acting on surrounding electrons is called the *ponderomotive force*. As the pulse passes through the plasma, the bi-transverse electromagnetic wave interacts with the charged particles via the Lorentz force:

$$F_{\text{Lorentz}} = q\,(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \tag{5}$$

The sinusoidal electric field pushes electrons transversely away from the optical axis due to the decreasing intensity gradient of the field further from the pulse. As the electrons gain transverse speed through these oscillations, there is a longitudinal $\boldsymbol{v} \times \boldsymbol{B}$ contribution which also acts to move electrons out of the way of the pulse. The positive ions in the plasma are effectively stationary on the ultra-short timescale of this process, as their mass is more than three orders of magnitude greater than that of the electrons.

Electromagnetic forces mean that once disturbed, the electrons want to return to their original positions. The restoring electric field due to this separation of charge is on the order of hundreds of $GV \cdot m^{-1}$, much greater than those found in conventional particle accelerators limited by dielectric breakdown. This results in a positive ion cavity forming behind the pulse, whose boundaries move at approximately the speed of the pulse. The inevitable collapse of this bubble-like electron void results in a wake of high electron density. A 'bunch' of injected electrons can ride the wake, i.e. the plasma wave, constituting the electron acceleration portion of laser wakefield acceleration. The holistic transfer of the pulse's intense electromagnetic energy to the forward motion of an electron beam is tied to the eventual depletion of the laser pulse.

A laser's a_0 parameter can be used to roughly characterize the nature of laser-matter interaction:

- $a_0 < 1$ corresponds with non-relativistic electron oscillation due to interaction with the laser's EM fields. Their behaviour aligns with linear optical effects.
- $a_0 \approx 1$ corresponds with electrons approaching c, leading to the onset of relativistic effects like an increased relativistic mass.

- $a_0 > 1$ corresponds with strongly relativistic behaviour required for LWFA experiments.

When a laser enters density non-uniformities in a plasma, the fluctuating refractive index causes the plasma to act as a phase object, changing the optical path length and inducing diffraction. However, the ponderomotive-based acceleration of electrons in the plasma also induces changes in the refractive index, which can act counter to the diffraction through density gradients. For self-focusing of the beam to occur, the normalized laser vector potential must obey an approximate lower bound:

$$a_0 > a_{0c} \approx \left(\frac{n_c}{n_e}\right)^{\frac{1}{5}} \tag{6}$$

The onset of self-focusing of the laser is correlated with strong electron generation. This bound ensures that the threshold is met for self-injection in the blowout regime.

In particular, we are considering the bubble, or blowout regime of LWFA, in which the high-intensity pulse is able to completely expel plasma electrons in a roughly spherical cavity after the pulse. This regime has two limitations:

- Dephasing, in which the accelerated electron bunches outrun the accelerating portion of the spherical wake, reducing electron energy and beam quality;
- Depletion, in which the driving pulse transfers sufficient energy to the plasma wave to halt the acceleration process.

The length over which these phenomena occur places an upper bound on the acceleration length, and therefore attainable energies of the electron beam.

B. Scaling Laws

Consider a laser system with defining characteristics wavelength λ_0 and pulse energy E_{pulse}. Note that the critical plasma density is specified solely by the laser wavelength:

$$n_c = \frac{\epsilon_0 m_e \omega_0^2}{e^2} \tag{7}$$

For the pulse's angular frequency $\omega_0 = 2\pi f_0$. The wave number for the plasma's natural oscillations k_p is then found as:

$$k_p = \frac{2\pi}{\lambda_p} = \frac{\omega_p}{c} \tag{8}$$

Relating this quantity to our plasma wavelength λ_p. It is interesting that λ_p denotes the physical length of the plasma waves, meaning the length of bubbles produced by the laser.

Using pulse compression techniques alongside the focusing optics in the experiment, the experimentalists have control over the pulse length τ and normalized laser vector potential a_0. Using LWFA scaling laws developed by Lu et al., we can use an iterative technique to find the optimal experimental conditions to achieve LWFA [2], [3]. We will first fix a 'seeded' value of a_0, which will allow us to optimize the pulse duration and waist radius. From this, we can calculate a_{0c}, which we will use to refine our next iteration's seeded a_0 until $a_0 > Ca_{0c}$, for some arbitrary buffer $C > 1$.

In more detail, we first fix the value of a_0 and τ. From a_0, we can find the peak on-axis intensity as:

$$I = \left(\frac{a_0}{0.855 \cdot 10^{-9}\lambda_0}\right)^2 \tag{9}$$

Note that $\lambda_0\,(\mu m)$ and $I\left(\frac{W}{cm^2}\right)$ are not in standard units, but rather in a more conventional form. Assuming the pulse is Gaussian, we can use our fixed τ to find the peak power of the pulse as:

$$P = 0.94\frac{E_{\text{pulse}}}{\tau} \tag{10}$$

The peak intensity and power of the pulse constrain the optical parameters of the laser at focus, allowing us to find the minimum waist radius:

$$w_0 = \sqrt{\frac{2P}{\pi I}} \tag{11}$$

Where $P\,(W)$ and $I\left(\frac{W}{m^2}\right)$ must be in standard units, and w_0 is measured as the $\frac{1}{e^2}$ radius. Finally, we can use a resonance approximation to match an optimal laser pulse length given our spot size:

$$\tau \approx \frac{2}{3}\frac{w_0}{c} \tag{12}$$

Conceptually, this formula claims that for resonance, the spatial extent of the pulse $c\tau$ must be on the order of the spot size w_0. We can repeat the calculations from Equation 10 to Equation 12, using our optimal τ as the next iteration's seeded τ until convergence is achieved.

Next, we must confirm that our a_0 meets the self-focusing threshold a_{0c} for self-injection to occur. Given our optimized parameters at a seeded a_0, we can apply another resonance matching scaling law to find the optimal plasma frequency:

$$\omega_p = \frac{2\sqrt{a_0}c}{w_0} \tag{13}$$

This condition focuses on transverse resonance between the plasma wavelength and the laser pulse, in that the transverse ponderomotive force on the electrons matches the transverse restoring force of the ion cavity. The plasma frequency defines the required electron density in the plasma:

$$n_e = \frac{\epsilon_0 m_e \omega_p^2}{e^2} \tag{14}$$

Which we can use in one last self-guiding estimate to find the critical normalized laser vector potential:

$$a_0 > a_{0c} \approx \left(\frac{n_c}{n_e}\right)^{\frac{1}{5}} \tag{15}$$

If $a_0 < Ca_{0c}$, where $C > 1$ is our arbitrary buffer, then we can set a_{0c} to be the seeded value of a_0 in our next iteration, and repeat our calculations from Equation 9.

Using these optimized results, we can calculate values for the dephasing and depletion lengths of our setup. Considering depletion to be the $\frac{1}{e}$ decay of a Gaussian pulse's energy, an estimate for this characteristic length is:

$$L_{\text{dep}} \sim \lambda_p \frac{n_c}{n_e} \tag{16}$$

To a dimensionless constant for $a_0 \gg 1$ [4].

The dephasing length is estimated by assuming the electron bunch travels at approximately c in the lab frame, catching up to the pulse travelling with phase velocity v_p over a distance of $\lambda_p/2$:

$$L_d \approx \lambda_p \left(\frac{n_c}{n_e}\right)^{\frac{3}{2}} \tag{17}$$

However in the matched bubble regime, the etching speed due to depletion slows the phase velocity of the pulse, combined with an altered bubble radius estimate to offer a second, shorter dephasing length of [1]:

$$L_d \approx \lambda_p \left(\frac{n_c}{n_e}\right) \frac{2\sqrt{a_0}}{3\pi} \tag{18}$$

Note that these length estimates are not always accurate, as past the self-focusing threshold, the pulse spot size and intensity will deviate from the expected input parameters. This must be kept in mind when examining simulation results to get a more accurate picture of how dephasing and depletion impact the system.

For optimal acceleration, the target (and therefore the plasma) is as long as the dephasing length. Any shorter and the electrons are under-accelerated, while any longer, the electrons decelerate and beam quality diminishes. As such, the dephasing length is an upper bound on the length of acceleration, and attainable energies for the system.

Often, $L_{deph} > z_R$, which means the laser diffracts before the electrons dephase, catching up to within $\lambda_p/2$ of the laser pulse. An alternative case occurs when the laser is able to self-focus beyond z_R, i.e. when $a_0 > a_{0c}$, which can greatly expand acceleration capabilities.

C. Optimization Script

The parameter optimization program lends itself well to a script that takes in the laser wavelength λ_0 and pulse energy E_{pulse} and returns a list of the solved parameters. Both an individual Python file `lwfa_params_optimizer.by` and batch processing script `batch_lwfa_params.sh` were created. The bash script runs the Python file, sweeping across a number of pulse energies at a customizable laser wavelength. A table of results is presented in Table I.

TABLE I
SCALING LAW OUTPUTS FOR DIFFERENT LASER PULSE ENERGIES OBTAINED USING OUR OPTIMIZATION SCRIPT AND THE PROCESS DESCRIBED ABOVE.

E_{pulse}(mJ)	a_0	a_{0c}	τ(fs)	w_0(μm)	z_R(μm)	L_{deph}(μm)	I_{peak}(W cm^{-2})	P_{peak}(W)	n_e(cm^{-3})	ΔE(eV)
1	2.0	1.8	3.3	1.5	8.7	1.8×10^1	8.1×10^{18}	2.8×10^{11}	9.9×10^{19}	1.2×10^7
3	2.2	2.0	4.5	2.0	16	3.9×10^1	9.9×10^{18}	6.3×10^{11}	6.0×10^{19}	2.1×10^7
5	2.3	2.1	5.1	2.3	21	5.6×10^1	1.1×10^{19}	9.1×10^{11}	4.8×10^{19}	2.8×10^7
7	2.3	2.1	5.6	2.5	25	7.2×10^1	1.2×10^{19}	1.2×10^{12}	4.1×10^{19}	3.4×10^7
9	2.4	2.2	6.0	2.7	29	8.6×10^1	1.2×10^{19}	1.4×10^{12}	3.7×10^{19}	3.9×10^7
11	2.4	2.2	6.4	2.9	32	1.0×10^2	1.3×10^{19}	1.6×10^{12}	3.3×10^{19}	4.3×10^7
13	2.5	2.2	6.7	3.0	35	1.1×10^2	1.3×10^{19}	1.8×10^{12}	3.1×10^{19}	4.7×10^7
100	3.0	2.7	11.6	5.2	110	5.0×10^2	1.9×10^{19}	8.1×10^{12}	1.2×10^{19}	1.4×10^8
300	3.3	3.0	15.7	7.1	200	1.1×10^3	2.3×10^{19}	1.8×10^{13}	7.4×10^{18}	2.6×10^8
500	3.4	3.1	18.0	8.1	260	1.6×10^3	2.5×10^{19}	2.6×10^{13}	5.9×10^{18}	3.5×10^8
1000	3.7	3.3	21.8	9.8	380	2.7×10^3	2.9×10^{19}	4.3×10^{13}	4.3×10^{18}	5.1×10^8
30000	5.0	4.5	55.1	24.8	2400	3.1×10^4	5.3×10^{19}	5.1×10^{14}	9.2×10^{17}	3.2×10^9

III. NOZZLE DESIGN REVIEW

A. Considerations

The laser-matter interaction of a short and intense laser pulse with a gas target generates a plasma providing the medium for LWFA. The gas target used in these vacuum-based experiments must have a well-defined density and spatial extent, with limited-enough mass flow rate to avoid overwhelming the vacuum pump.

When translating our scaling laws into gas target parameters, the density of the body of the gas target must be controlled in order to transform into a plasma of appropriate electron density. It must meet the electron density requirement n_e of the LWFA parameters, noting that each gas molecule may ionize multiple times (e.g. $He \rightarrow He^{2+}$).

A second important consideration is the dephasing and depletion lengths in our specific blowout regime. The shorter of the two lengths places an upper bound on the length of viable electron acceleration. Given that past the dephasing length, the spatial and energy spectrum coherence of a generated electron beam degrades, we want to ensure the plasma

bounds the possible wakefield interaction to no longer than L_{deph}. Similarly, beyond the depletion length, the laser pulse has transferred too much energy into the plasma wakefield to continue acceleration. Assuming a 1:1 correlation between gas target and plasma length, these constraints place an upper bound on the length of the high-density region of our gas target.

Note that this condition is only valid provided the self-focusing regime can be attained. Earlier, we found that this condition is met if $a_0 > a_{0c}$, and will allow the beam to stay focused longer than the Rayleigh length. Otherwise the Rayleigh length $z_R < L_{\text{deph}}$ will prevent the wakefield interaction from reaching extended lengths due to foundational Gaussian beam principles.

A further dimensional constraint on our plasma, and by extension our gas target, is the length over which the electron density increases from near-vacuum to target levels, or the so-called *density gradient*. The length of this transition into the region meeting our target electron density found via the scaling laws must be on the order of, or smaller than, z_R, or there will be diffraction through the gas jet [2]. This is one of the main limiting factors in scaling down LWFA, as the decreasing z_R places a difficult-to-meet constraint on the sharpness of our density gradient. The gas target naturally has some fuzziness at the edges, which increases as the gas diffuses farther from the nozzle exit. We must balance the nozzle geometry, inlet gas pressure, and distance from the exit to confine density gradients.

Additionally, the interaction point of the laser and the target must sit clear of the nozzle exit, or the laser pulse will damage the nozzle. A damaged nozzle will compromise the parameters of the gas target, which would be both unsustainable for repeated experiments, and render pulsed usage within a single experiment impossible. For a high power (TW or greater) laser system, this buffer region lies in the hundreds of μm to single-digit mm range, while a novel low-power, high-repetition rate LWFA experiment might be able to approach the $100\mu m$ mark.

Within these constraints, there is room to play with advanced ideas. For example, it is possible to abandon a uniform density profile in favour of one with a sharp entrance density spike in a technique called density transition injection. This can manipulate electron injection at the base of the plasma wake at the onset of plasma generation. This will be explored in our nozzle design review.

B. Design Review

The simplest nozzle to consider is a **subsonic jet design**, consisting of a simple cylindrical nozzle and opening. This has the disadvantages of high mass flow rate and broader and weaker gas density clouds, making it less suitable [5].

A **de Laval nozzle**, or converging-diverging nozzle, is an hourglass-shaped tube that can accelerate a gas to supersonic speeds. These types of nozzles are commonly used in rocketry, as the high exhaust velocity imparts a large impulse to propel the rocket forwards. However, our use case is slightly different; we target a supersonic nozzle for its ability to produce a dense and collimated exhaust plume to serve as a consistent gas target for LWFA.

An **axisymmetric de Laval nozzle** is its most common form. This is an axisymmetric revolve of the 2D profile idiomatic to a de Laval nozzle. Within LWFA, this is useful when only a small gas target is needed. However, when scaling to longer L_{deph} and longer acceleration lengths, increasing the diameter of an axisymmetric de Laval nozzle to increase the gas target length also increases its width. This constitutes a lot of wasted mass flow rate, as the laser only interacts with a thin central slice of the profile along the optical axis.

For longer plasma lengths with less waste gas, a **slot de Laval nozzle** design can be used, which consists of an extruded 2D de Laval nozzle profile with planar end caps. This results

in the desired converging-diverging exhaust profile along the optical axis, but avoids excess transverse width when viewed from the optical axis.

Longitudinally, removing one of the capping walls can produce a density gradient at the entrance or exit. This plays into an idea called *density transition injection*, using variable longitudinal density profile to alter the plasma wavelength as the pulse travels through the gas target. Inducing a sharp downward step in density immediately after the start of the plasma 'rephases' the injected electrons into the acceleration part of the plasma wave all at once [6]. This can be done using a shock or blade in the exhaust path of the jet, as discussed below. Alternatively, a more gradual down ramp over many λ_p results in gradual trapping of plasma electrons. This would be the case for the aforementioned example of removing the exit planar cap on a slot de Laval nozzle [5].

Let's examine the design of a shocked nozzle in more detail [7]. A shock in supersonic flow is a sudden reduction in the Mach number and increase in density. If a supersonic flow changes direction abruptly, it can produce a shock wave. This can reshape the density profile of the exhaust, resulting in sharp changes in density or focusing of the exhaust plume. One application of a shocked nozzle is density transition injection, as mentioned above. Furthermore, the ultra-small scales achievable, leveraging micrometer dimensions on the nozzle alongside the tight gradients induced by the shock, are particularly useful for LWFA with kHz systems with tighter scaling law constraints.

First let us look at the **axisymmetric shock using a cylindrical collar** at the exit of a de Laval nozzle, at an angle θ to initial flow. This generates an oblique shock wave at an angle β to initial flow, or $\beta - \theta$ with respect to the axis, where θ is the divergence angle of the nozzle. The formula:

$$\tan\theta = 2\cot\beta\left[\frac{M_1^2\sin^2\beta - 1}{M_1^2\left(\gamma + \cos 2\beta\right) + 2}\right].$$

Relates θ, β, and the initial Mach number before shock M_1 [5]. Note that for low $\theta < 20°$ at a given M_1, the angle with respect to the axis $\beta - \theta$ is relatively unchanged. Another important trend is that increasing M_1 decreases $\beta - \theta$, i.e. faster input flow results in a shallower convergence angle of the shock wave.

Note. This relation may have two solutions for β given a set of inputs: one corresponding to a strong shock at high deflection resulting in subsonic flow, and another corresponding to a weak shock at shallower angle and continued supersonic flow. The weak shock is almost always seen. △

If we measure z_m to be the on-axis distance from the end of the *diverging part* of the de Laval nozzle where the oblique fronts converge into a target maxima, we find:

$$z_m = \frac{d_E/2}{\tan\left(\beta - \theta\right)} - L.$$

Where L is the length of the straight collar and d_E is the exit diameter of the nozzle. Note that the effects of boundary layers means this will likely overestimate what is seen in simulation and experiment [7].

Increasing L, to a point, will bring the convergence point closer to the nozzle. This is likely due to the fact that shorter than a given L, not all the gas particle trajectories 'see' the shock, i.e. an inner cone of particles propagates as normal and therefore does not contribute to forming the oblique shock. This also increases the density at the convergence point, again up to a given limiting L.

An **asymmetrically shocked jet** can be used to produce a sharp density drop, which is useful for density transition injection. In the case of a sharp gradient, where the gradient transition length $L_{\text{grad}} < \lambda_p$, then the electron density will drop and the plasma wavelength

will increase sharply [6]. This will result in the plasma wave increasing in size, 'rephasing' and trapping the slower moving injected electrons, resulting in better beam quality. It is more repeatable than other methods of creating a sharp downwards density spike, such as placing a blade on top of the nozzle, as it is built into the nozzle's form.

For this exploratory project, we opt to investigate the axisymmetric case of a de Laval nozzle design. We will explore the design causality relationships in further detail and work towards developing and characterizing a prototype nozzle for LWFA with the terawatt (TW) laser. This work will explore the relevant design tools and scientific questions involved in the development of a gas target nozzle. It will facilitate further optimization and exploration of other nozzle designs in future continuation of the wider project.

C. Axisymmetric de Laval Design

We will explore the design causality relationships of a de Laval nozzle, drawing on the work of Schmid et al. in their experimental study of a de Laval nozzle with throat diameter $d^* = 1mm$, exit diameter $d_E = 3mm$, and throat-to-exit length $L = 6mm$ [8]. The study uses helium and argon at $T_0 = 300K$ with a backing pressure of $50bar$ in their exploration of de Laval design principles.

As a gas flows through the throat of a de Laval nozzle, the pressure, temperature, and density all plummet, while the velocity and Mach number increase. This process is considered isentropic, so the pressure and temperature drop as flow accelerates. Note that this can 'artificially' inflate the Mach number due to the dropping speed of sound as the gas flows through the exit:

$$c = \sqrt{\kappa R T}.$$

Where κ is the specific heat ratio of the gas, R is the specific gas constant and T is the temperature. The low pressure and low temperature at the exit lead to low transversal expansion (i.e. low divergence), creating well-defined density gradients compared to a subsonic jet. If we *ignore boundary layers and other 2D/3D effects*, we can calculate the Mach number, temperature, pressure, and density at some point L along the central axis as a function of the cross-section ratios between the throat f^* and the downstream point of interest f.

$$\frac{f^*}{f} = M\left[1 + \frac{\kappa - 1}{\kappa + 1}\left(M^2 - 1\right)\right]^{-\frac{\kappa+1}{2(\kappa-1)}} \tag{19}$$

$$\frac{T}{T_0} = \left(1 + \frac{\kappa - 1}{2}M^2\right)^{-1} \tag{20}$$

$$\frac{p}{p_0} = \left(1 + \frac{\kappa - 1}{2}M^2\right)^{-\frac{\kappa}{\kappa-1}} \tag{21}$$

$$\frac{\rho}{\rho_0} = \left(1 + \frac{\kappa - 1}{2}M^2\right)^{-\frac{1}{\kappa-1}}. \tag{22}$$

Where M is the Mach number at that axial point, and the naught subscript indicates the parameter value at the inlet. At the narrowest point of constriction, or the throat, the gas reaches $M = 1$, and is said to be in *choked flow*. At this point, further drops in downstream pressure do not increase the speed of the fluid in choked flow. Since the pressure wave travelling at the speed of sound cannot travel upstream to the point of choked flow, the upstream fluid cannot 'see' the downstream conditions.

However, these 1D isentropic laws gloss over non-idealities present in any real nozzle. Notably, *boundary layers* are slow moving fluid between the walls and the fast flow in the

main volume, often defined as the outer layer of fluid carrying the transition from 0% to 90% axial gas velocity. They govern the steepness of the density gradient of the gas jet near the nozzle exit (~ 1 nozzle diameter), and take up some of the cross-section of the nozzle, reducing the 'effective' cross-sectional geometry.

Causality Relationships

The **boundary layer thickness** increases as the pressure drops further down the nozzle as the molecular mean free path increases. Therefore, thin boundary layers occur for high backing pressure, low cross-section ratio $f_E/f^* \gtrsim 1$, and short nozzles.

A **larger exit diameter to throat diameter ratio d_E/d^* produces more collimated gas jets**, as the increased expansion results in lower exit pressures and transverse expansion forces. For nozzle half-angles up to $10°$, divergence is determined solely by exit diameter and nozzle angle can be neglected, while above $10°$, increasing the nozzle angle also contributes to increasing the divergence. A larger expansion ratio also decreases the output density[1]. Over-expansion of the gas jet, wherein the exhaust pressure drops below ambient pressure and causes various issues, is not a concern in an LWFA setup as we are projecting the plume into vacuum.

The **length of density gradients increases for longer nozzles** since there is more time for a boundary layer to build up. There is linear dependence between the length of the density gradient and the product Ld_E for a given throat size. Increasing the length or the exit diameter results in more divergence and more diffuse gradients.

The **flat-top profile of the density is affected by the propagation of Mach waves, which results in more distortion for larger nozzle angles**. This is because there are stronger changes in flow direction, contributing to Mach wave severity. Longer nozzles also have less distortion as the Mach waves have more time to attenuate.

In general, **larger nozzles and higher backing pressure** converge to the $1D$ isentropic results as there are relatively fewer particle-wall collisions. Scaling down a nozzle means increasing the Knudsen number K_n, which is the ratio of smallest geometry to mean free path of molecules, which acts to increase the boundary layer thickness. This results in a more Gaussian profile with larger density gradients, compared to a more flat-top density profile for lower K_n.

In summary, lower divergence occurs for higher d_E/d^* ratios due to more expansion. However, sharper gradients occur for lower nozzle half-angles. Prioritizing both design points means a longer nozzle, which accumulates larger boundary layers and also results in less-defined density gradients. Higher backing pressure gives sharper gradient edges as well as higher densities, attributable to higher mass flow rate. Changing the throat diameter also affects mass flow rate.

IV. ANSYS FLUENT CFD NOZZLE SIMULATION

A. Design Choice

We will design our prototype to target the TW laser. A kHz repetition rate laser has more stringent scaling law margins and will involve a more complex nozzle design to optimize the

[1] See 1D isentropic equations above.

gas target, which is a suitable future iteration of this nozzle design project. Consider rough parameters taken from scaling law estimates using the characteristics of the TW laser:

$$\begin{aligned} E &= 500mJ \\ a_0 &= 3.4 > a_{0c} = 3.1 \\ \tau &= 18.0fs \\ z_R &= 260\mu m \\ L_{\text{deph}} &= 1.60mm \\ n_e &\approx 5.9 \cdot 10^{18} cm^{-3}. \end{aligned}$$

Our nozzle design must conform to these parameters; we will iterate CFD simulations while tuning elements investigated in Section III-C until we have a suitable candidate to move forward with validation.

B. Simulation Overview

ANSYS Fluent is the computational fluid dynamics tool we will use to simulate gas flow through our nozzles. This involves numerically solving the Navier-Stokes equations, a set of PDEs used to model the flow of viscous fluids, considering their mass, inertia, pressure, and viscosity. The numerical turbulence model used to solve these equations is the $k-\omega$ shear stress transport (SST) variant, which combines $k-\omega$ and $k-\epsilon$ models depending on the region. The $k-\omega$ model is used for near solid boundaries, since it can handle flow separation and other interactions. The $k-\epsilon$ model is best for free shear flow, and is less computationally expensive. 2D simulations are sufficient for axisymmetric geometries, whereas 3D simulations can be used in the excepting cases. The 2D simulation grid is typically $\sim 10^5$ cells, where adaptive size ensures higher resolution at critical points, like along boundaries or in constricted regions.

We are simulating the flow of gas through a nozzle, consisting of an inlet shaft, the nozzle's internal geometry, and an exhaust region to analyse the plume characteristics, shown in Figure 43. The inlet shaft, sized to fit the gas inlet, serves to stabilize incoming flow. An input parameter is set to hold the entrance of the inlet shaft at a target backing pressure using the 'pressure inlet' boundary condition, which is manipulated in our parameter sweeps. The input temperature is set to $300\,\mathrm{K}$, and the turbulence specification method is set to 'Intensity and Hydraulic Diameter', with 'Hydraulic Diameter' on the order of our smallest dimension[2]. Within the nozzle geometry, we increase the resolution of our quadrilateral mesh near the throat and at the boundaries to capture boundary layer effects. We used helium as our medium, configured as a compressible ideal gas.

At the outlet, we face one of the limitations of this CFD approach. The continuum approximation within the numerical solver requires that the smallest geometry is much larger than the molecular mean free path (MFP) λ_{MF}. The Knudsen number K_n is a measure of the ratio between a characteristic geometric dimension L and the MFP:

$$K_n = \frac{\lambda_{MF}}{L} = \frac{1}{\sqrt{2}\sigma\rho_P L} = \frac{1}{L} \cdot \frac{k_B T}{\sqrt{2}\pi d^2 p} \tag{23}$$

Where ρ_P is the number density, σ is the scattering cross section of fluid molecules or atoms $\sigma = 4r_W^2\pi$, r_W is the Van der Waals radius, $p\,(Pa)$ is the pressure, $T(K)$ is the temperature, and d is the kinetic collision diameter of the molecules. We require $K_n < 10^{-3}$, or $K_n < 10^{-1}$ if slip-flow modifications are made in Ansys Fluent, which means we can only simulate a small exhaust region within which the gas pressure remains relatively high, $\gtrsim 0.1\,\mathrm{Pa}$. The

[2]The throat radius.

outlet is then defined using a 'pressure outlet' boundary condition at the desired pressure. We verified that the simulation did not change significantly for outlet pressures of $0.1\,\mathrm{Pa}$, $1\,\mathrm{Pa}$, and $10\,\mathrm{Pa}$. All presented simulation results were obtained with a $1\,\mathrm{Pa}$ outlet boundary condition.

C. CFD Analysis Details

The geometry to be simulated was prepared in SolidWorks and processed using the Ansys suite of design and meshing tools, DesignModeler and Fluent Meshing. See Section XI-A for a guide on creating, importing, and editing the sketch, as well as creating a high quality mesh for simulation. The simulations were performed on the Digital Research Alliance of Canada's (DRAC) High-Performance Computing system (HPC) Fir. See Section XI-B for a guide on configuring and submitting Ansys Fluent case files to the Fir HPC.

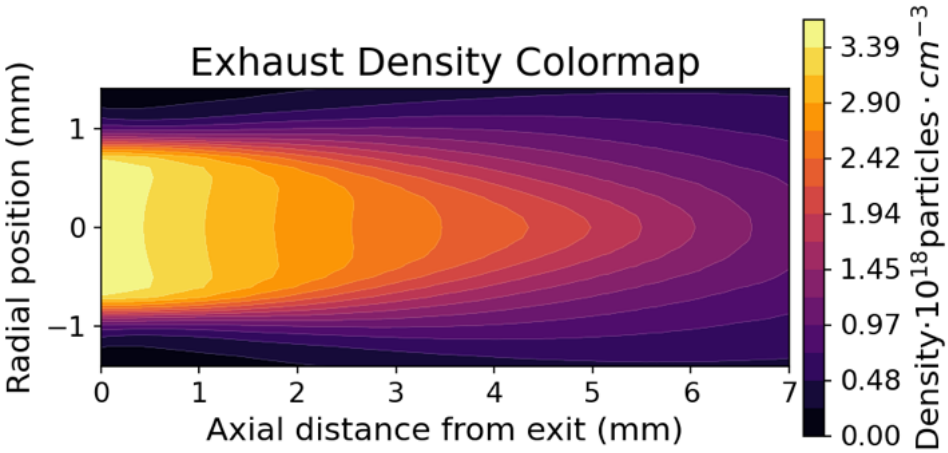

Fig. 1. An example exhaust density colourmap from our Ansys Fluent simulations. The nozzle exit is aligned with the left wall of the image; the gas jet points to the right, with tapering decreasing density contours as the plume travels and diffuses.

An example density colormap of an exhaust plume obtained from our simulations is shown in Figure 1. Note that our simulations are axisymmetric, so the profile is mirrored across the axis of symmetry for visualization purposes. In this view, the laser beam would pass through a vertical line at some axial clearance from the exit. As such, we will continue our analysis of the exhaust profiles using vertical line-outs at varying distances from the nozzle exit, each representing a potential beam path. Sample density line-outs from the contours in Figure 1 are displayed in Figure 2.

We must devise ways of quantitatively characterizing our density line-outs, in order to compare across conditions. See Figure 3 for an annotated view of the metrics we use to characterize and quantify each exhaust profile output.

One important metric is the width of the gas plume. We can consider both the full width at half maximum (FWHM) as well as the length of just the *density plateau*, or tabletop feature of our density profiles. To define the density plateau region, we use a sliding window method, starting from the centre of the profiles (where the plateau is guaranteed to be defined) and moving outwards. We continue outwards until the profile falls below the starting window's mean, minus several standard deviations $\rho \geq \overline{x} - n\sigma$. We calculate the plateau density using the mean density of the isolated plateau region.

Another metric we must consider is the length of our density gradient. Borrowing from signal processing, we define this to be the 'rise time' of the density profile, meaning the distance over which the density rises from 10% to 90% of the plateau density minus the tail density.

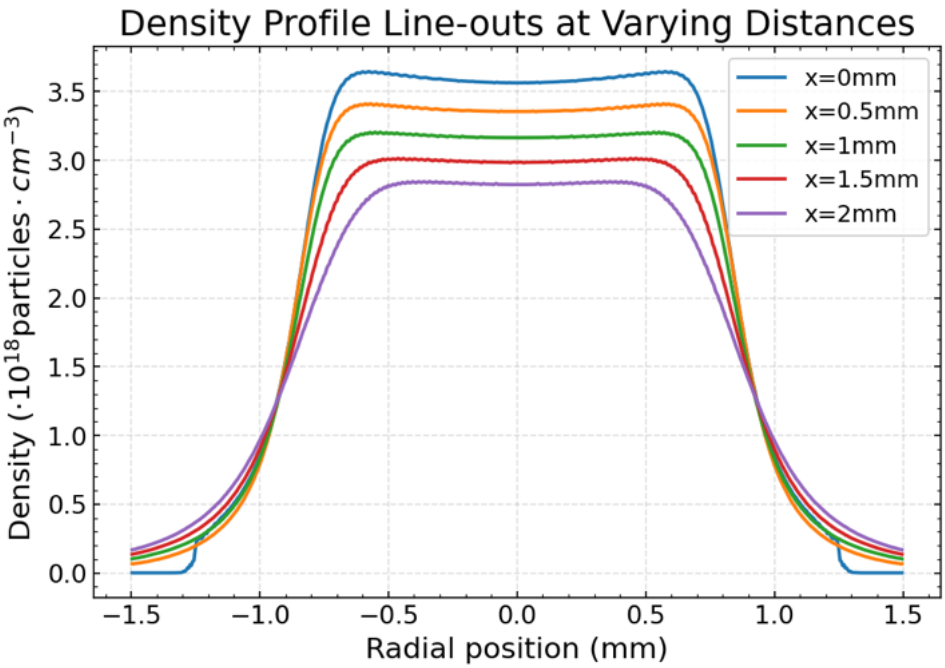

Fig. 2. A view of 5 density line-outs at varying distances from the nozzle exit. Each profile has a tabletop-esque profile, with density gradient ramps on either side of the central plateau.

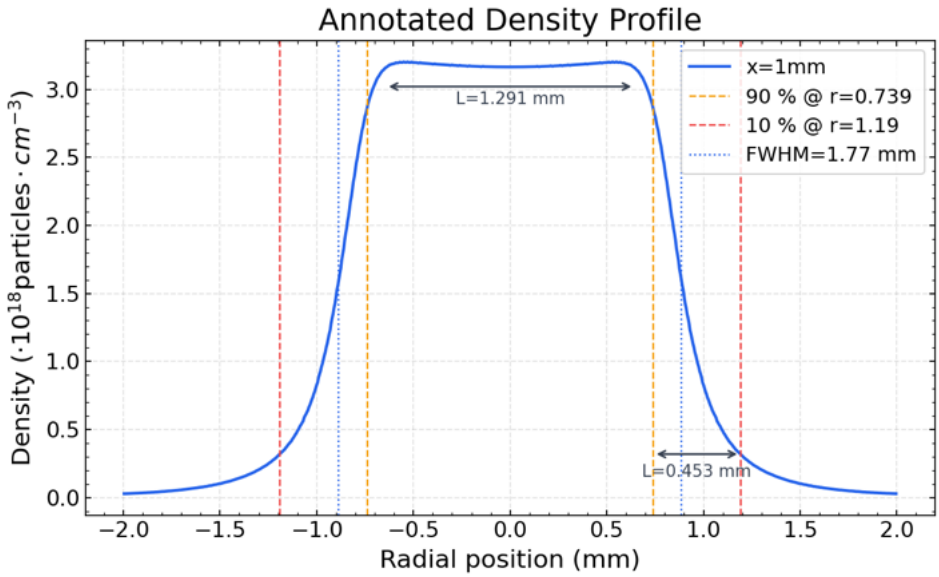

Fig. 3. An annotated density cross-sectional profile of an exhaust plume from our Ansys Fluent simulations, displaying our FWHM, plateau length, and density gradient length metrics. These metrics will be used to compare density profiles across simulation conditions.

D. Results

Using the parametric scan capabilities of Ansys Fluent, we simulated each individual geometry at a range of inlet pressures, allowing us to see the impact of backing pressure on output plume characteristics, and how this effect varied with changing geometry.

We can first examine the *linear correlation of backing pressure on exhaust plume density*. As shown in Figure 4, there is a clear linear relationship between the plateau density, or density of the tabletop region, and the inlet pressure. While the throat of the nozzle is in

choked flow, meaning the gas passes through at Mach 1, the pressure in the throat is not fixed. Increasing the inlet pressure manifests as a proportional increase in throat pressure and density, which results in a higher mass flow rate and therefore exhaust density:

$$\dot{m} = \rho A_t a_t.$$

Where the throat area A_t and gas velocity a_t are fixed. We also notice a decrease in plateau

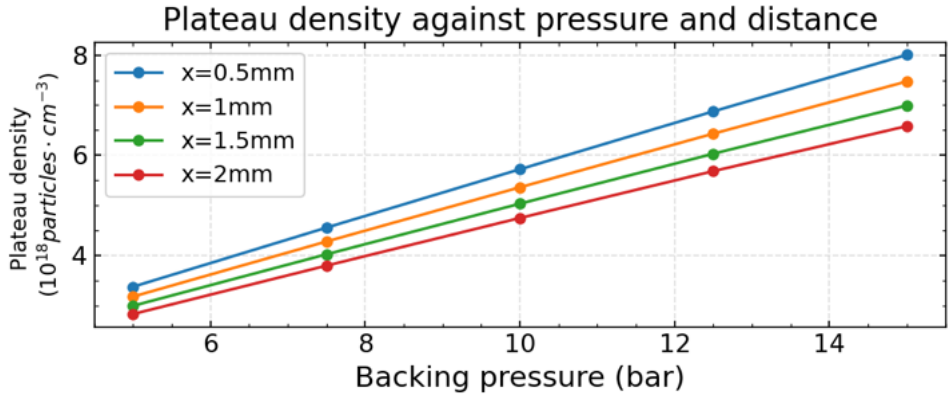

Fig. 4. Plot of the extracted plateau density against backing pressure for various line-out distances.

density with increasing distance from the nozzle exit, which intuitively follows from the expansion and diffusion of the gas as it leaves the nozzle.

Next, we can examine the *inverse correlation of the nozzle's radial expansion ratio on exhaust density*. Consider a nozzle with exit radius r_E and throat radius r^*. From the 1D isentropic laws above, we can calculate the expected Mach number and density ratio $\frac{\rho}{\rho_0}$ for the density at the nozzle exit ρ compared to the throat density ρ_0. A radial expansion ratio of 4, using $\kappa = \frac{5}{3}$ for an ideal monatomic gas, predicts a density ratio of $\frac{\rho}{\rho_0} \approx 0.021$. A radial expansion ratio of 5 predicts a density ratio of $\frac{\rho}{\rho_0} \approx 0.0135$. The plateau density simulation

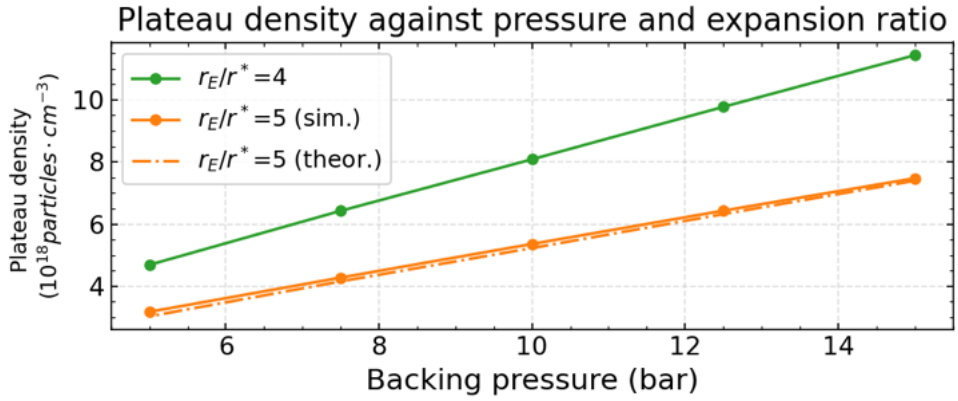

Fig. 5. The plateau density against pressure for two simulations with differing radial expansion ratios. A dashed theoretical line for the series $r_E/r^* = 5$ is displayed, calculated using the other series' data and the 1D isentropic laws. The two series are labelled 'theor.' for theoretical, referring to the calculated predictions, and 'sim.' for the results of our simulation. Each plateau density is calculated using a line-out at a distance of $1\,\mathrm{mm}$ from the nozzle exit.

results of two nozzles with expansion ratios of 4 and 5, respectively, are presented in Figure 5. The calculations above are used to predict the plateau density series for the expansion ratio of 5 trial based on the expansion ratio of 4's plateau density values. Note that our prediction is effectively superimposed on simulation results. There is slightly better agreement at higher

backing pressure than lower backing pressure, likely due to the increasingly present effect of boundary layers at low pressure, discussed below.

Next, let us explore the *negative correlation between backing pressure and gradient length*. We can quantify the boundary layers using the gradient length, as larger boundary layers result in more poorly-defined edges of our density profile. From Figure 6, we observe that increasing backing pressure reduces gradient length due to erosion of boundary layers, as discussed in Section III-C. Additionally, taking line-outs closer to the nozzle exit also reduces

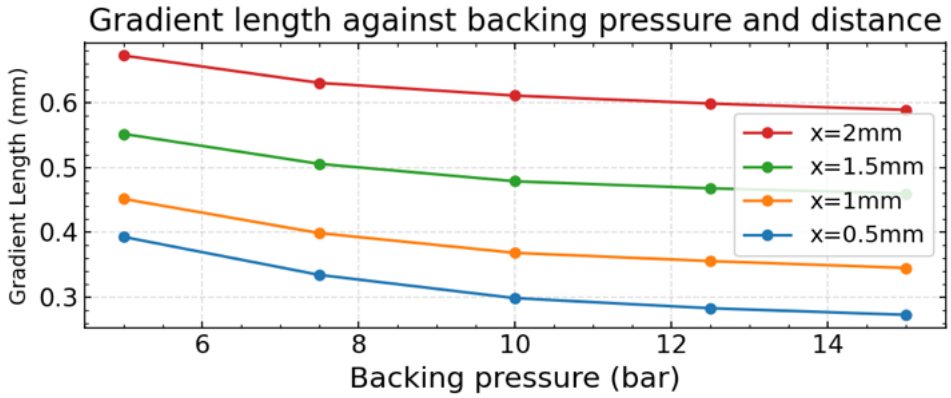

Fig. 6. The gradient length against backing pressure at a given distance from nozzle exit, plotted for varying distances. Both increasing the backing pressure and decreasing the distance from the nozzle exit serve to reduce gradient length and sharpen exhaust profile.

gradient length since the gas has less space to diffuse.

At close range, the *distance from the nozzle exit has negligible effects on the FWHM of the exhaust plume*. This is because as the edges of the gas target become more diffuse further from the nozzle, our tabletop profile gradually shifts towards a more Gaussian shape. There is a rounding out of the corners of the plateau, and the 'excess density' is reallocated into the tails of the profile, while the half-maximum point remains roughly constant. From Figure 7,

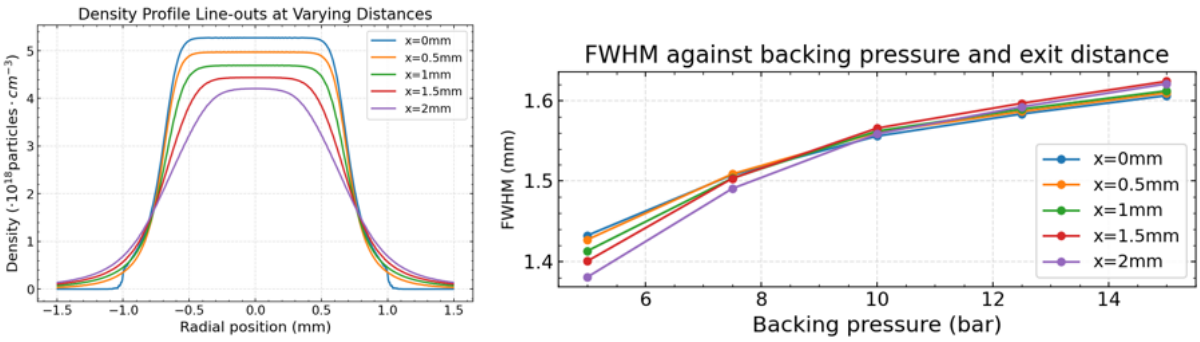

Fig. 7. (left) A comparison of exhaust density line-outs at increasing distances from the nozzle exit, demonstrating the rounding of the tabletop into a Gaussian profile. (right) A comparison of density profile FWHM against backing pressure, shown for different distances from the nozzle exit. Each series represents a set distance from the nozzle exit; the series are nearly superimposed, indicating little correlation.

we can also glean that increasing the *backing pressure increases the FWHM of the exhaust plume* due to the erosion of boundary layers and widening of the 'effective nozzle geometry' seen by the exiting gas.

A more revealing metric is the plateau length. Like the FWHM, increasing the backing pressure increases the plateau length due to the erosion of boundary layers producing a

larger effective nozzle geometry. However, unlike FWHM, *the plateau length decreases with increasing distance from the nozzle exit*. This metric is able to capture the diffusion of the gas

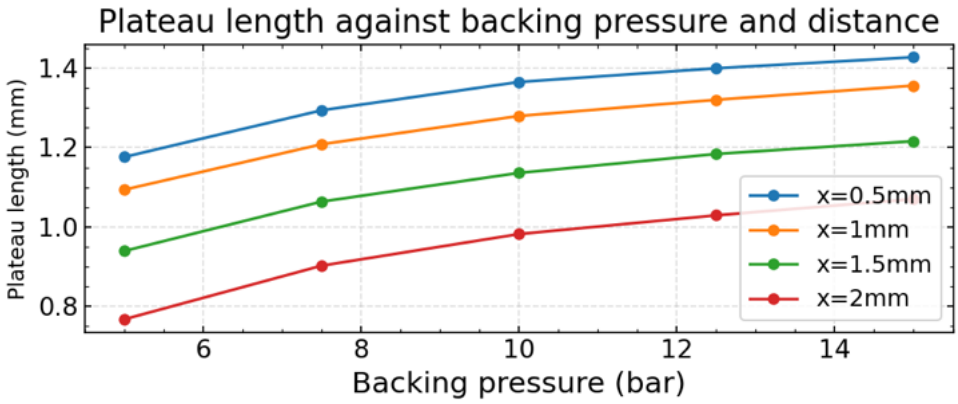

Fig. 8. Relationship between plateau length and backing pressure, shown across varying line-out distances from the nozzle exit. There is clear separation between line-outs at different distances.

target, which is important when analyzing the plasma length of appropriate electron density that the laser will experience.

Using these comparisons, we tuned the parameters of our nozzle geometry until landing upon the following values:

- $d^* = 500\,\mu\text{m}$, $d_E = 2500\,\mu\text{m}$;
- 7° divergence angle;
- $5\,\text{bar}$ backing pressure.

The throat uses a sketch fillet with radius of $1\,\text{mm}$ to smooth the contour. The inlet shaft is sized to a gas inlet with diameter of $4.2\,\text{mm}$, is $4\,\text{mm}$ in length, and has a converging section angled at a convergence half-angle of 20°. At a distance of $x = 1\,\text{mm}$ from the nozzle exit, this results in a design with:

- Plateau density of $\approx 3.2 \cdot 10^{18}$ he $\cdot\, cm^{-3}$;
- Gradient lengths of $470\,\mu\text{m}$;
- Plateau length of $1\,\text{mm}$, FWHM of $1.8\,\text{mm}$.

With annotated lineout shown in Figure 9. While the parameters, particularly the gradient length, stray from our idealized scaling laws, exact conformity in Ansys Fluent is not the goal. The experimental values of the parameters will differ from simulation estimates by an unknown margin, and is something to be re-visited after characterizing the gas jet. For this exploratory project, these specifications are sufficient to move towards the other aspects of the process.

We ran an additional simulation using the same geometry, but this time using argon gas as the medium. For neutral gas density interferometric characterization, it is easier to use argon than helium due to argon's higher polarizability, resulting in larger phase shifts for a given density. We ran these simulations at backing pressures of 250PSI, 300PSI, and 400PSI, into vacuum pressures of $100\,\text{Pa}$. This aligns with the experimental conditions in Section VII. Discussion of these results can be found alongside discussion of the experimental interferometric results.

V. Resin Printing Optimization

This section will present the problems, analysis, and enacted solutions during the process of successfully resin printing the supersonic nozzle.

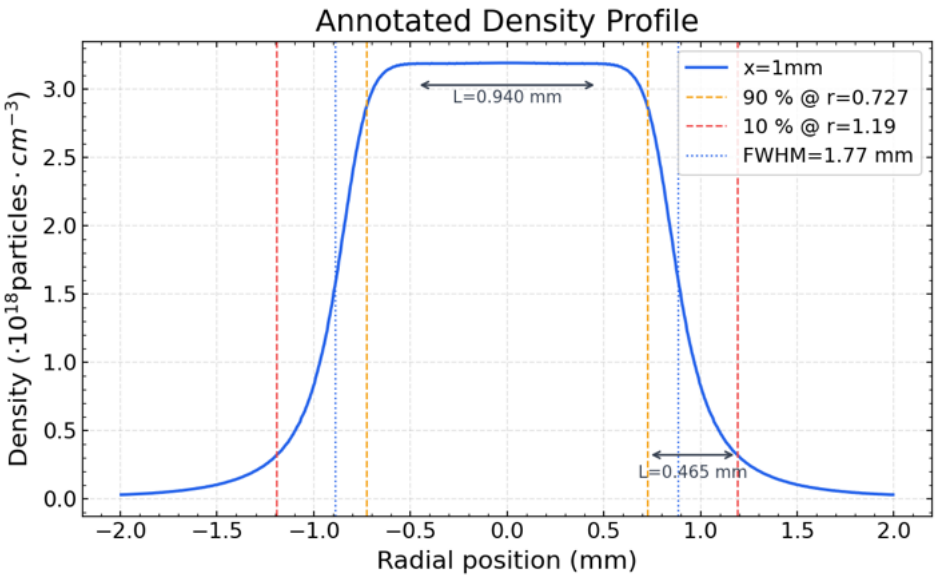

Fig. 9. An annotated lineout of the finalized prototype obtained via Ansys Fluent CFD simulations.

Throughout the optimization process, we emphasized a methodical approach, changing one parameter at a time to isolate causation. We kept a meticulous log of notes, detailing the print parameters, changes, and input files, as well as an analysis of each print to gauge parameter causality relationships. Reviewing the print history allowed us to make informed decisions, as we could analyse an observed problem, propose a theoretical explanation, and take steps to address the underlying cause of failure.

A. Manufacturing Problem Overview

The results of our simulations allow us to arrive at a candidate for an experimentally-viable design. However, experiment never exactly matches simulation, so it is important to validate our physical design by comparing it to expected results. In order to experimentally characterize our nozzle using interferometric techniques, we require a physical prototype of the nozzle. The prototype must be manufactured precisely enough to faithfully represent the most limiting geometry in the design. Additionally, the prototype should be cost-effective and quick to manufacture in the case of damage or design iteration.

A method frequently encountered in literature is microspark erosion, a form of subtractive manufacturing which uses small electrical discharges to vaporize unwanted material on a conductive body [2], [9]. This method has the advantage of extreme (few-micron) precision with hard materials, but can be expensive and inaccessible. We will study the optimization of an alternative approach: using a resin printer to additively manufacture our nozzle prototypes. The lab's Anycubic Photon M3 printer is capable of $40\,\mu\mathrm{m}$ XY resolution and $50\,\mu\mathrm{m}$ layer heights. This is within 10% of our most limiting dimension. The main advantage of resin printing the nozzles is that it is quick and cheap, taking mere hours and several cents worth of resin to produce a prototype.

To get to this point, we must optimize the parameters of a test resin print to find print conditions that will allow for consistent and high-fidelity prints.

B. Optimization Process

We used a tentative 'final' design of a de Laval gas nozzle, obtained via our simulation work, as a test print subject to optimization.

Fig. 10. (left) A view of the lab's Anycubic Photon M3 resin printer. (right) A view of the printer with resin tray and build plate removed, revealing the UV screen under the tray.

In addressing any issues with the resin printing, we have three main controls:

1) The external[3] geometry of the nozzle;
2) The printing layout and support geometry;
3) The resin printer settings.

Shown in Figure 11, an initial print of the nozzle in the most intuitive arrangement (i.e. with the nozzle base flat against the build plate) revealed two issues:

- The throat of the nozzle was plugged by solidified resin, rendering the nozzle unusable;
- The part had an 'elephant's foot,' or bulging at the base of the part due to longer exposure times for the burn-in layers, which degraded the quality of the inlet geometry.

The most critical of these issues is clearly the blocked throat. As the resin printer cycles up and down in the resin, it pauses at the bottom for the UV exposure, and lifts to break the seal between the new layer and the resin vat. During this process, viscous resin gets trapped in the interior of the nozzle, unable to easily pass through the $500\mu m$ throat and exacerbated by the lack of external airflow passage to break the seal between the bottom of the nozzle and the print bed. With each new formed layer, the UV curing can 'bleed' outside the desired geometry, over time solidifying the trapped resin in the throat. The elephant's foot issue, where layers contacting the build plate are deliberately overexposed to ensure good adherence, results in unwanted geometry around the base due to a similar UV bleeding effect. It is possible to tweak printer settings to reduce the severity of the elephant's foot bleeding, but the only way to remove it entirely is to stop the part from contacting the build plate.

We can address these problems in tandem. The solution is two-fold:

- Improve drainage of the resin by lifting the base of the print off of the print bed for airflow;
- Change the curing settings to mitigate the UV bleed.

Given that the throat, as internal geometry, is fixed, we must change the print settings and layout. To encourage resin drainage, we critically decided to elevate the print off of the bed. This allows air to travel into the inner passage from the print bed, breaking the seal which may have trapped resin and simultaneously eliminating elephant's foot on the part. For print settings changes, we increased the 'light-off time' to $1.5s$, which is the duration between the

[3]Not the inner profile itself, which is dictated by the results of our Ansys Fluent simulations.

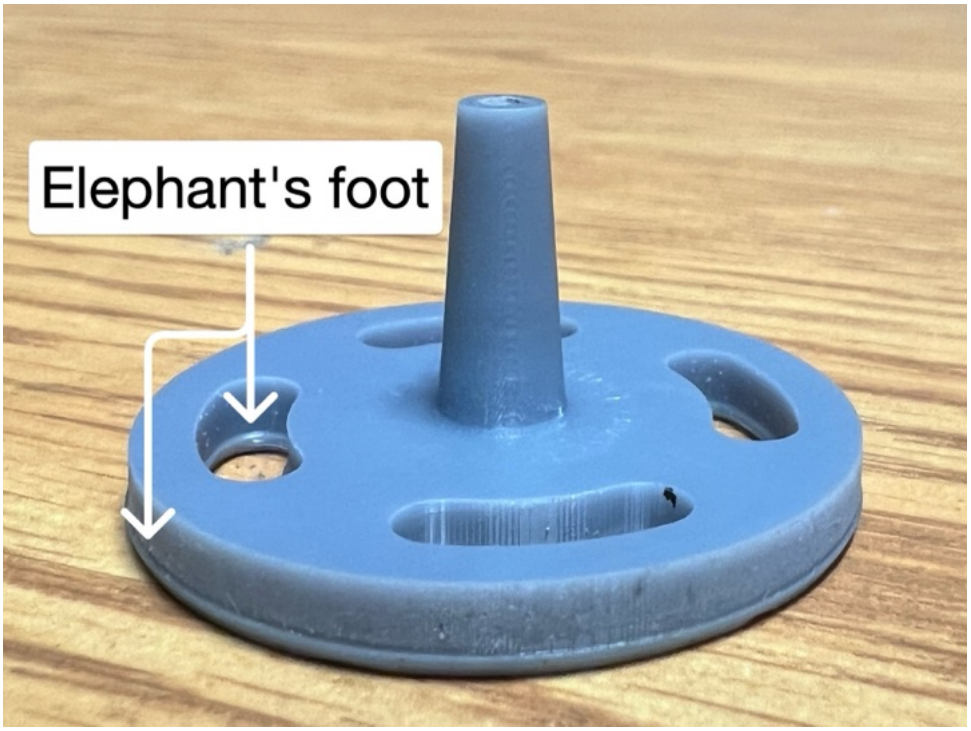

Fig. 11. The initial print of the full nozzle, with elephant's foot highlighted. The blocked throat cannot be seen in the photo, nor the inlet shaft at the base which was also affected by elephant's foot.

print bed halting movement and the UV light turning on. We hypothesize that this gives the resin more of a chance to drain, aided by the addition of the airflow channels. To reduce the UV bleed, we decreased the UV exposure time until just avoiding compromising the print quality, settling around $1.9s$ when using a layer thickness of $50\mu m$. These changes were iteratively made using a section of our test print, which only comprised of the relevant nozzle section with limiting internal geometry to reduce waste time and material. A successful test is shown in Figure 12.

Next, we moved to re-printing the full nozzle with the settings changes, with print geometry similar to that of Figure 13.

This revealed two new problems:

- Printing the nozzle off the base resulted in adhesion issues and delamination of the print where the supports meet the part;
- Remnants of the supports resulted in a rough finished surface where the supports meet the part.

Analyzing the cause of the issue in Figure 14, we notice that the separation of the layers occurs at the base, where the layers are significantly deformed. When the print reaches the transition from supports to the base, the first few layers of the base have a very large surface area, and induce a lot of drag when the build plate moves vertically through the resin. This drag causes warping of the thin, flexible layers, which in turn results in poor adhesion of future layers to their counterpart's wavy surface.

To solve this issue, we need to improve the strength and uniformity of adherence of the supports to the base layer. We want to prevent separation as well as warping at any location along the base, all while minimizing the print damage incurred when removing the supports. We achieved this by developing the following print arrangement conventions:

1) No supports should touch internal geometry of the nozzle, where print quality is paramount;

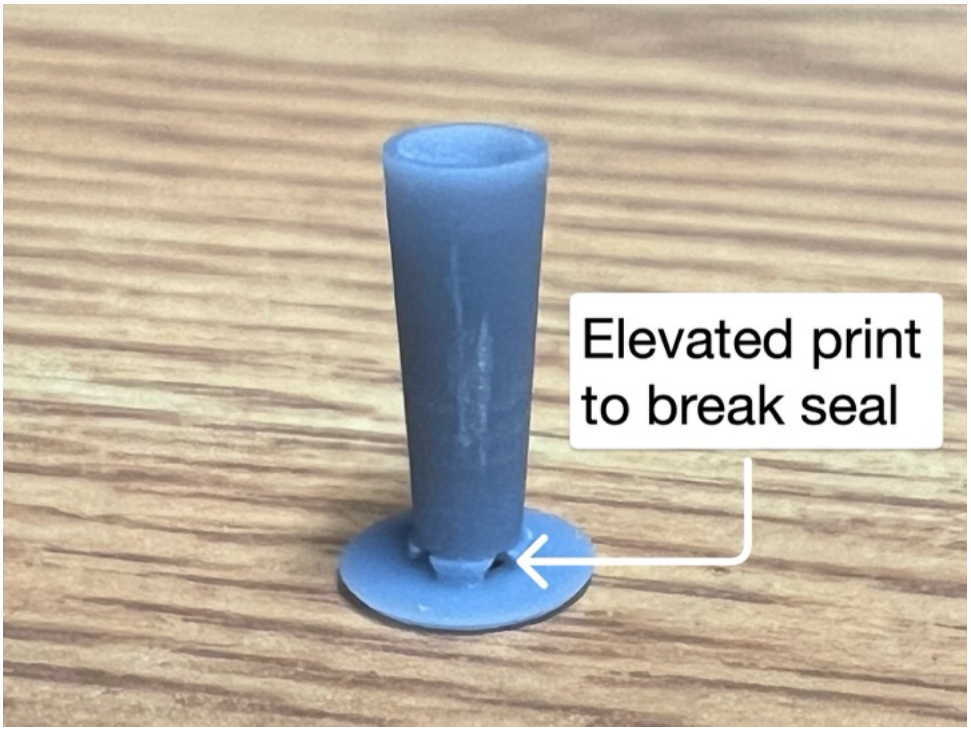

Fig. 12. The successful partial print of the nozzle, featuring the elevated print with fluid flow channels and improved printer settings. The nozzle featured a clear throat (not visible) and no elephant's foot on the part itself.

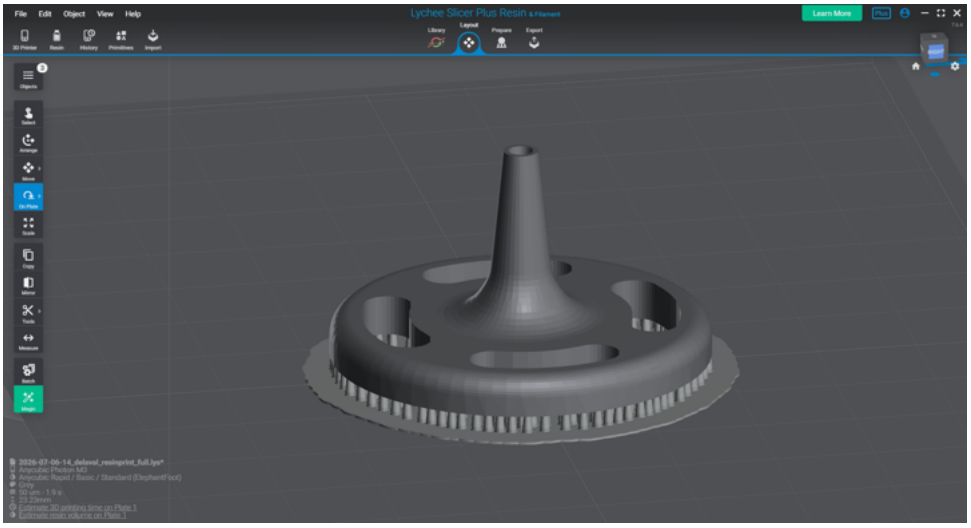

Fig. 13. A screenshot from Lychee slicer of the lifted, right-side up print layout. The $2mm$ layer of supports enables flow from the build plate out the nozzle at the cost of a rougher bottom print surface.

2) A closely-spaced border of supports should encircle any edges of the print, such as around the outside edge or the screw slots;
   - Every second support was made to be thicker ($0.75\,\mathrm{mm}$ tip diameter, $1\,\mathrm{mm}$ diameter, and $3\,\mathrm{mm}$ tip length) to improve adherence at the edges, where separation is prone to start;
   - Every second support was left thin ($0.55\,\mathrm{mm}$ tip diameter, $1\,\mathrm{mm}$ diameter, and $3\,\mathrm{mm}$ tip length) to allow fluid flow through the ring of supports.
3) Infill supports are placed at regular intervals along the interior of the base to mitigate

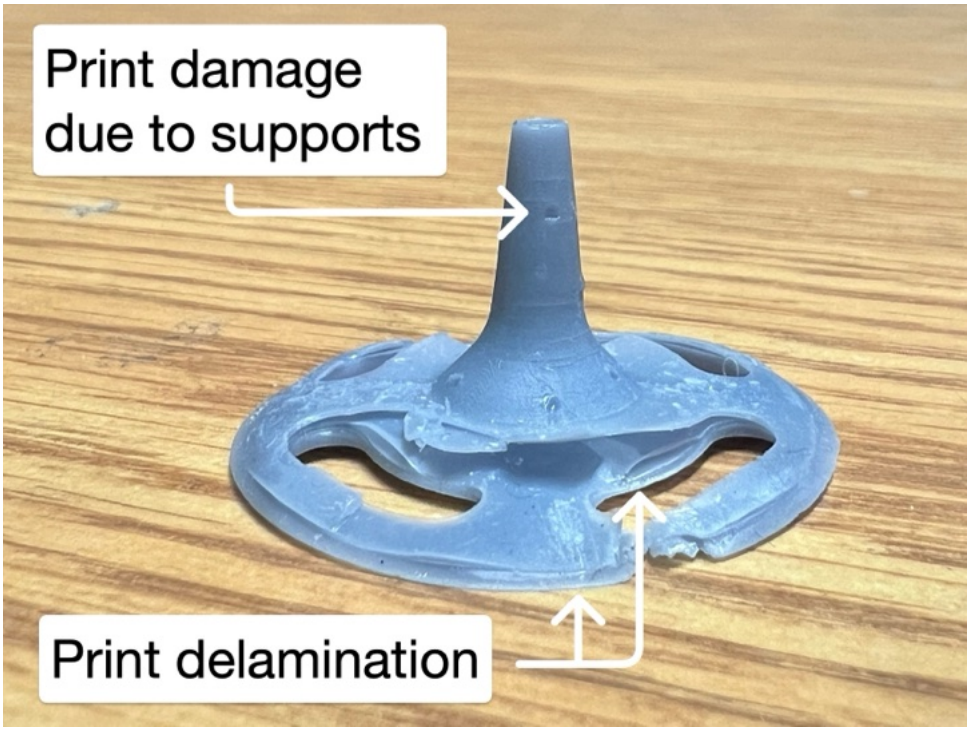

Fig. 14. An iteration of the print showing severe delamination of the base, as well as physical damage to the print from removal of supports. Note that we attempted printing the nozzle upside-down (with the nozzle exit towards the base) to avoid this problem, but encountered incorrigible asymmetries in the final print.

any warping or bubbling;

4) Around critical geometry that must be supported (i.e. the O-ring inset), supports with an extra-thin tip were used to minimize damage;
   - Supports had the parameters: $0.35\,\mathrm{mm}$ tip diameter, $1\,\mathrm{mm}$ diameter, and $3\,\mathrm{mm}$ tip length.

Taken together, we were able to successfully print iterations of our nozzle, with sufficient print quality to seal and test the exhaust plume. See Figure 15.

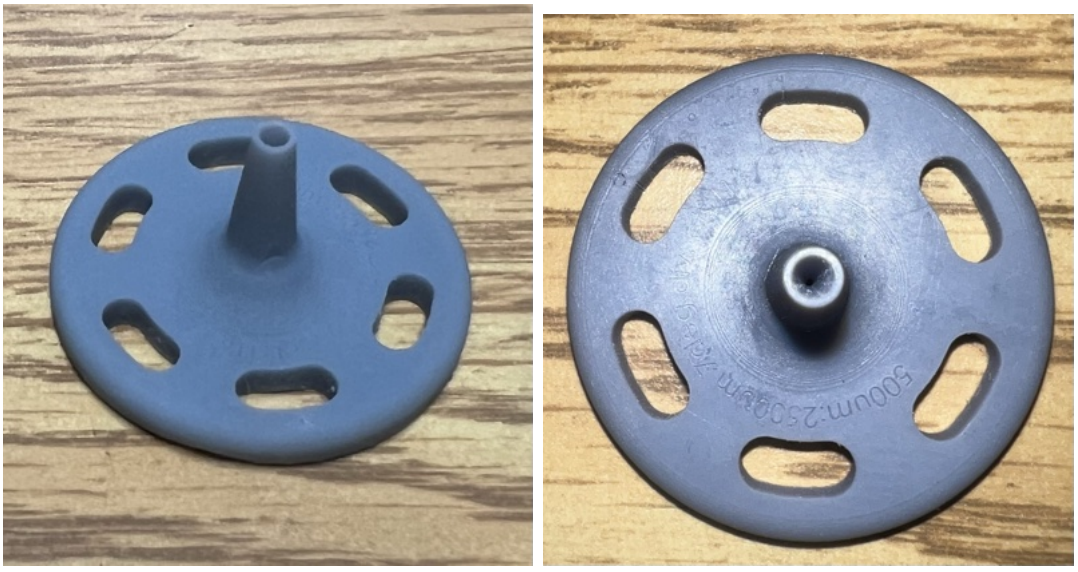

Fig. 15. A top and side view of the final nozzle, with clear throat.

C. Findings

Below is a summary of the conventions we developed to produce successful de Laval nozzle prints:

- The base of the print must be lifted off the build plate to allow for air and resin flow through the throat, without a seal;
  - This convention addresses both the plugged throat by improving drainage, and the elephant's foot by eliminating contact between the print and the bed.
- Overhanging edges of the part should be supported with a tight ring of supports, alternating between thick and thin support tips to balance rigidity at the print edges, where flexing is exacerbated, with ease of removal and continued air and resin flow;
- Large overhangs must be supported with a less-dense infill of thin supports for even coverage without significant surface damage upon removal;
- No supports should touch internal geometry of the nozzle to preserve print quality.

VI. WarpX PIC Codes

Particle-in-cell (PIC) codes simulate the complex interactions of charged particles in the presence of electromagnetic fields. WarpX is a set of PIC codes that can be used to simulate the LWFA interaction of a laser propagating through a plasma [10]. WarpX ensures self-consistent evolution, in which electromagnetic fields impact particle movement, which in turn impacts electromagnetic fields. As simulating each particle individually is computationally impossible, WarpX solves Maxwell's equations over a grid of 'macroparticles', where each macroparticle represents a grouping of many individual particles.

We will present the installation and basic usage of WarpX 26.06 on Fir for LWFA simulations, including a description of fundamental analysis concepts.

A. Installation

WarpX is a set of simulation codes that must be downloaded and built from source for use. We will detail the process of installing WarpX 26.06 as of July 2026 on the Digital Research Alliance of Canada's (formerly Compute Canada) Fir computing cluster.

When logging into Fir, the cluster will place the user on a 'login node,' a 1 CPU node with 1GB of RAM meant for lightweight tasks only. The heavy installation and compilation we will conduct demands the use of compute nodes, which can be requested and checked out by the user. This is done via Fir's resource manager and job scheduler, Simple Linux Utility for Resource Management (SLURM). We can check out interactive resources, meaning for directly user-interfacing jobs, using the `salloc` (SLURM allocate) command:

```
salloc --time=2:0:0 --nodes=1 --ntasks=1 --gpus-per-node=h100:1
↪  --cpus-per-task=32 --mem-per-cpu=2G --account=def-group
```

The options shown are configured to provide us with 32 CPUs per task on 1 task, each CPU with 2GB[4] of memory, alongside a single H100 GPU per node across 1 node, for 2 hours. The request will pend until SLURM successfully allocates the resources, at which point the terminal will spawn a bash session on the allocated node.

First, we will download the source code files from the WarpX project repository into a folder named `WarpX-26.06`. This is located within a dedicated source code folder in our home directory, and should be renamed to the WarpX version downloaded.

```
cd
mkdir -p $HOME/code/src
```

[4]SLURM uses binary prefixes, so 2GB is equivalent to 2048MB.

```
git clone https://github.com/ECP-WarpX/WarpX.git
↪  $HOME/code/src/WarpX-26.06
```

The Digital Research Alliance of Canada uses a module system to load different software packages and versions. Using `module avail python`, we can view a list of all related modules and version options. We can run `module load python/3.11` to load Python 3.11, satisfying WarpX's Python ≥ 3.6 requirement and allowing us to construct a virtual environment. A virtual environment is an isolated sandbox containing a copy[5] of a Python interpreter and installed packages. Once activated, any installed dependencies will only be installed in the context of the venv, and will not interact with packages installed globally or in other venvs.

It is best practice to store a venv in the project root, or in the context of Fir, in a dedicated directory on the user's home. The following lines of code ensure an empty hidden folder `.venvs` is created in the user's home directory, and uses the loaded Python 3.11 module to create a new venv called `warpx_gpu`.

```
cd
rm -rf .venvs
mkdir .venvs && cd .venvs
python -m venv warpx_gpu
source warpx_gpu/bin/activate
```

We can activate, or enter the venv using `source $HOME/.venvs/warpx_gpu/bin/activate`, and deactivate using `deactivate`. Now inside our venv, we can install relevant packages:

```
python3 -m pip install --upgrade pip
python3 -m pip install --upgrade build
python3 -m pip install --upgrade packaging
python3 -m pip install --upgrade wheel
python3 -m pip install --upgrade setuptools[core]
python3 -m pip install --upgrade pipx
python3 -m pip install --upgrade cython
```

Building on the more foundational packages above, we move towards more specific packages for WarpX. It is worth noting that there is a `numpy` versioning conflict that will pop up and must be resolved. After installing `numpy`, subsequent packages will reinstall it to their preferred version. The penultimate time this occurs, `yt` installs `numpy` 1.26.4 with requirements $1.14.0 \leq$ `numpy` $\leq 2.0.0$, which satisfies `matplotlib`'s requirement of `numpy` ≥ 1.25, `scipy`'s of $1.26.4 \leq$ `numpy` < 2.7, and `pandas`' of `numpy` $\geq 1.26.0$.

```
python3 -m pip install --upgrade numpy
python3 -m pip install --upgrade pandas
python3 -m pip install --upgrade scipy
python3 -m pip install --upgrade mpi4py --no-cache-dir
↪  --no-build-isolation --no-binary mpi4py
python3 -m pip install --upgrade openpmd-viewer
python3 -m pip install --upgrade laserbeamsize
python3 -m pip install --upgrade matplotlib
python3 -m pip install --upgrade yt
```

However, the following `requirements.txt` install will target `numpy` 2.4.2, which is incompatible with the previous requirements. In `requirements.txt`, it notes a minimum version of `numpy` ≥ 1.15, meaning the newest version is not necessary and can be safely reverted back to `numpy` 1.26.4.

[5] It's often a symlink, so changing the root Python installation can impact a venv.

```
python3 -m pip install --upgrade -r
↪  $HOME/code/src/WarpX-26.06/requirements.txt
python3 -m pip uninstall numpy
python3 -m pip install numpy==1.26.4
```

Next, we must download and source `warpx.profile`, a bash script which loads relevant modules and exports useful environment variable tweaks before using WarpX each time.

```
vim ${HOME}/warpx.profile
source ${HOME}/warpx.profile
```

Note. One example of a module loaded by `warpx.profile` is `paraview/6.0.0`. This module contains an HDF5-enabled version of `openpmd-api`, which allows for reading and writing of scientific files, specifically `.h5` files. This is helpful when using `.h5` files for the diagnostic outputs of WarpX, with AMReX configuration options:

```
diagnostic.format = "openpmd"
diagnostic.openpmd_backend = "h5"
```

Previously, `openpmd-api` was installed as a python package, but I had difficulty compiling with HDF5 support due to challenges recognizing dependencies within Compute Canada's file system. △

The next step is installing two key requirements for WarpX, `blaspp` and `lapackpp`, each of which being linear algebra library wrappers. These must be compiled from source code as followed, using the same dedicated source code directory as before. First we will clone and build `blaspp`.

```
export SW_DIR=${HOME}/code/src
git clone https://github.com/icl-utk-edu/blaspp.git $HOME/code/src/blaspp
cd $HOME/code/src/blaspp
rm -rf $HOME/code/src/blaspp/build
mkdir $HOME/code/src/blaspp/build
cd $HOME/code/src/blaspp/build
cmake .. -B . -Duse_openmp=OFF -Dgpu_backend=cuda -DCMAKE_CXX_STANDARD=20
↪  -DCMAKE_CUDA_ARCHITECTURES=90 -DCMAKE_INSTALL_PREFIX=${SW_DIR}/blaspp
cmake --build . --target install --parallel 16
```

Note that we used cmake to build the package, but there are other options. Consult the documentation for more information. Now we build `lapackpp`.

```
git clone https://github.com/icl-utk-edu/lapackpp.git
↪  $HOME/code/src/lapackpp
cd $HOME/code/src/lapackpp
rm -rf $HOME/code/src/lapackpp/build
mkdir $HOME/code/src/lapackpp/build
cd $HOME/code/src/lapackpp/build
CXXFLAGS="-DLAPACK_FORTRAN_ADD_" cmake -S .. -DCMAKE_CXX_STANDARD=20
↪  -Dbuild_tests=OFF -DCMAKE_INSTALL_RPATH_USE_LINK_PATH=ON
↪  -DCMAKE_CUDA_ARCHITECTURES=90
↪  -DCMAKE_INSTALL_PREFIX="${SW_DIR}/lapackpp"
cmake --build . --target install --parallel 16
```

We are finally at the stage where we can build WarpX itself. First, create a clean dedicated build directory, and configure the `lapackpp_DIR` environment variable, which WarpX will look for when building.

```
cd $HOME/code/src/WarpX-26.06
rm -rf $HOME/code/src/WarpX-26.06/build
mkdir $HOME/code/src/WarpX-26.06/build
cd $HOME/code/src/WarpX-26.06/build
export lapackpp_DIR=${HOME}/code/src/lapackpp/build
```

Now we build the package.

```
cmake .. -DWarpX_DIMS="1;2;RZ;3" -DWarpX_COMPUTE=CUDA -DWarpX_MPI=ON
 ↪  -DWarpX_OPENPMD=ON -DWarpX_PYTHON=OFF -DWarpX_FFT=ON
 ↪  -DWarpX_QED_TABLE_GEN=ON -DWarpX_QED_TOOLS=ON
cmake --build . --parallel 24 # run from the build directory
```

We repeat with the Python interface.

```
cd $HOME/code/src/WarpX-26.06
rm -rf $HOME/code/src/WarpX-26.06/build_py
mkdir $HOME/code/src/WarpX-26.06/build_py
cd $HOME/code/src/WarpX-26.06/build_py
cmake .. -DWarpX_DIMS="1;2;RZ;3" -DWarpX_COMPUTE=CUDA -DWarpX_MPI=ON
 ↪  -DWarpX_OPENPMD=ON -DWarpX_PYTHON=ON -DWarpX_FFT=ON
 ↪  -DWarpX_QED_TABLE_GEN=ON -DWarpX_QED_TOOLS=ON
cmake --build . -target pip_install --parallel 24
```

Note the use of the `-target pip_install` argument to ensure an executable is made available.

WarpX should now be fully built and ready for use!

B. Using WarpX

WarpX is a complicated simulation software. Beyond tuning the many configuration options, it is important to understand how to analyse and interpret results of a simulation to glean meaningful results. We will present the execution and analysis of an LWFA simulation to better understand the environment.

Consider the working directory `/project/def-group/user/WarpX/learning_param_sweeps`, using my user folder in the group's broader projects folder. This *home folder* for our parameter sweeps will contain a master SLURM job submission script, as well as an AMReX input parameter file for WarpX to be customized before each run. Running the master SLURM script will execute the job and copy input and output files elsewhere. The relevant input parameter and logging files are copied into a metadata sub-directory in the `projects` folder, labelled by the test's SLURM job ID. The data files

```
[lconcini@login1 learning_param_sweeps]$ pwd
/project/def-      /lconcini/WarpX/learning_param_sweeps
[lconcini@login1 learning_param_sweeps]$ ls
inputs_2d_params  runWarpX2D.sh  TestRun_50050809
[lconcini@login1 learning_param_sweeps]$
```

Fig. 16. A readout of the files in the parameter scan projects directory, showing the input parameter file, the SLURM job script, and one project directory.

are written to `/scratch/user/WarpX/TestRuns`. The `scratch` folder is a large, temporary, per-user storage space designed for large files. The files are also copied into a

sub-directory labelled by the SLURM job ID, consistent with the metadata folder previously mentioned. For a given run, we edit the `input_2d_params` configuration file as desired,

```
[lconcini@login1 TestRuns]$ pwd
/scratch/lconcini/WarpX/TestRuns
[lconcini@login1 TestRuns]$ ls
TestRun_46287002  TestRun_47458277  TestRun_47681412
TestRun_46383385  TestRun_47460892  TestRun_47854249
TestRun_47254305  TestRun_47518165  TestRun_47855999
TestRun_47256065  TestRun_47637447  TestRun_47857281
TestRun_47257738  TestRun_47657997  TestRun_49726232
TestRun_47270024  TestRun_47676715  TestRun_49734989
TestRun_47272155  TestRun_47677267  TestRun_49906910
TestRun_47454430  TestRun_47679687  TestRun_50050809
[lconcini@login1 TestRuns]$
```

Fig. 17. A readout of the files in the parameter scan scratch directory, showing the data files for several runs.

and then submit the `runWarpX2D.sh` job submission script via `sbatch runWarpX2D.sh`. The directory parameters inside of the job script may need to be altered depending on the project, as well as the compute parameters depending on the intensity of the simulation.

Note. Increasing the resolution of the simulation grid via the `amr.n_cell` n_1 n_2 parameter will greatly increase memory requirements. Occasionally this manifested in an out-of-memory (OOM) error, terminating the job before completion. The `seff` (SLURM efficiency) command can be used to inspect the computational statistics of a completed simulation. A low CPU Efficiency percentage may indicate too many requested cores, while a low memory efficiency may indicate too much memory per core. △

The diagnostics outputs of a WarpX simulation can result in a huge amount of generated data, in the hundreds of gigabytes. As such, it is impractical to transfer data written to `scratch` elsewhere for analysis. We use an interactively-hosted JupyterLab Notebook on the Fir HPC to write and run analysis code with seamless access to our file system, and therefore our data.

We can host this on Fir by installing the relevant Python packages into a virtual environment, creating a simple launch script, and making it executable. We have already done this during installation via:

```
pip install --no-index --upgrade pip
pip install --no-index jupyterlab
echo -e '#!/bin/bash\nunset XDG_RUNTIME_DIR\njupyter lab --ip $(hostname
↪  -f) --no-browser' > $VIRTUAL_ENV/bin/jupyterlab.sh
chmod u+x $VIRTUAL_ENV/bin/jupyterlab.sh
```

Once logged onto the cluster, we source the virtual environment through the `warpx.profile` file set up during installation to prepare the relevant modules and virtual environment for WarpX. We also run a wrapper script designed to start JupyterLab on remote servers.

```
source warpx.profile
$VIRTUAL_ENV/bin/jupyterlab.sh
```

The script is simply running these commands underneath the hood to unset conflicting variable and bind the server to the specific compute node, or host:

```
#!/bin/bash
unset XDG_RUNTIME_DIR
jupyter lab --ip $(hostname -f) --no-browser
```

Note. Alternatively, see Section XI-E for another way to start JupyterLab on Fir to avoid connection issues △

To access the GUI, we must forward the port that the server is using to host our JupyterLab service to a port on our local machine via SSH tunnelling, e.g.:

```
ssh -L 8888:fc12345:1234 user@fir.alliancecan.ca
```

This command forwards the HPC port (e.g. port 1234 on node fc12345) to the port specified on our local machine (e.g. port 8888), which can then be accessed at localhost:8888. Two additional flags that can be added are:

- `-N`: the 'no command' flag tells the `ssh` command to avoid opening a terminal shell to execute remote commands;
- `-f`: the 'fork to background' flag will push the task into the background instead of letting it 'hang' in the terminal, otherwise preventing you from executing further commands from that shell.

C. Analysis Methods

Once on our JupyterLab server, we use a system of `.ipynb` python notebooks and `.py` code files to analyse the simulation results. The notebooks allow for iterative testing without having to needlessly repeat expensive data processing steps, while the source code files are a convenient way to store classes or helper functions.

We first created a 'discovery' notebook, which will scan the data and metadata folders created by our job submission script, parsing the 'metadata' (input parameter files) to build a catalogue of the stored simulations. This code can be found in `lwfa_analysis_p1.ipynb`. Usage is simple: change the `PROJECTS_DIR` and `SCRATCH_DIR` paths to match your system, and run the cells. Then, change the `SLURM_IDS` list to select the list of runs to be further explored. The output will show a list of characterizing parameters from each simulation file found, as well as an overview of the available iterations and diagnostics for plotting.

Next, we created a laser visualizations notebook, which is used for helpful visualizations of a specific simulation run. Similarly, usage of the notebook requires setting the `PROJECTS_DIR` and `SCRATCH_DIR` paths as well as `JOB_ID` variable for the specific name `TestRun_12345678` of the data and metadata folder to be analyzed. The notebook dynamically parses parameter data from the metadata folder to find the optical axis, polarization direction, or other simulation parameters relevant for configuring analysis.

Using the extracted data fields for a given iteration, we can plot a colormap of the electron density within the simulation frame, with laser intensity contour lines superimposed. This allows us to visually track the simulation's progression. In Figure 18, we compare the electron density at two points: shortly after the laser has entered our plasma electron region (step function density profile), and later into the interaction. We can see the formation of clear bubbles with injection of electrons, indicating we are well into the relativistic regime. We also notice degradation of the pulse's initial Gaussian quality as it depletes.

Another parameter we can extract is the electric field x-component (perpendicular to the optical axis, which is along z). Since most perturbations in the plasma electrons will occur symmetrically above and below the optical axis, they will have little contribution to an E_x

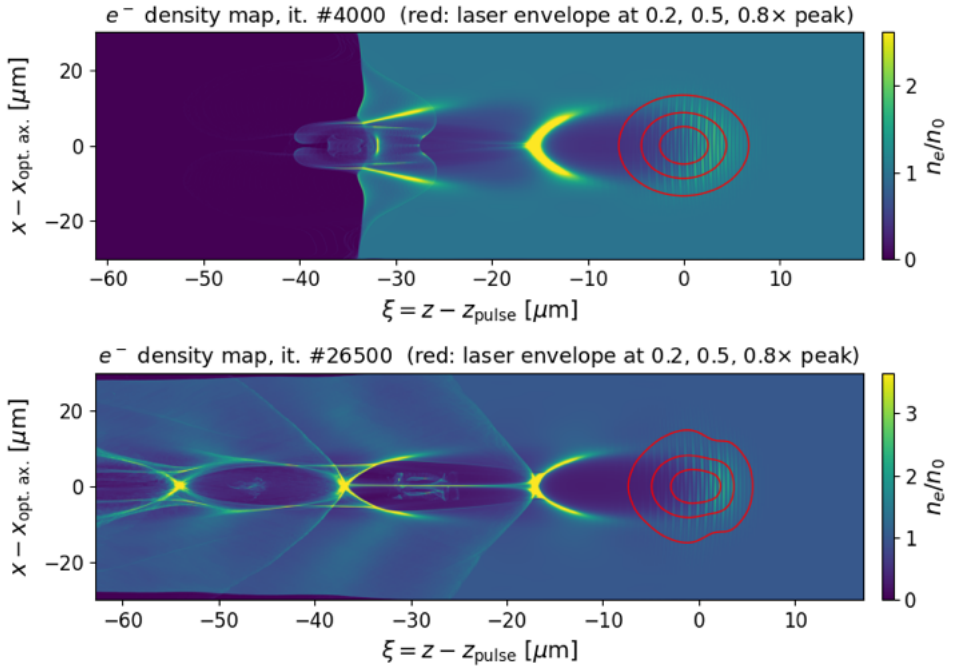

Fig. 18. A comparison of electron density colormap with superimposed laser intensity profile between iterations 4000 and 26500, displaying the evolution of wakefield bubbles as the pulse propagates. The x-axis is normalized around the optical axis, and the z-axis is normalized to the laser pulse centre $\xi = z - z_{\text{pulse}}$ for more accurate and aligned comparison.

lineout along the optical axis. However, the laser is polarized in the x direction, meaning we can extract the profile of our laser pulse as it propagates. The raw field will have a form similar to:

$$E_x(z) = A(z)\cos(k_0 z + \phi).$$

Where k_0 is the wave number of the laser and $A(z)$ is the desired amplitude envelope. Using a Hilbert transform, we can extract a 90° phase-shifted version of this signal to sum with our original following Euler's identity:

$$A(z)\cos(k_0 z + \phi) + iA(z)\sin(k_0 z + \phi) = A(z)\,e^{i(k_0 z + \phi)}.$$

Whose magnitude is just our desired $A(z)$. In the frequency domain, this is equivalent to taking the Fourier transform of our $E_x(z)$, dropping all negative frequency components, doubling the positive frequency components (since a real signal has a symmetric magnitude spectrum), and then performing an inverse Fourier transform.

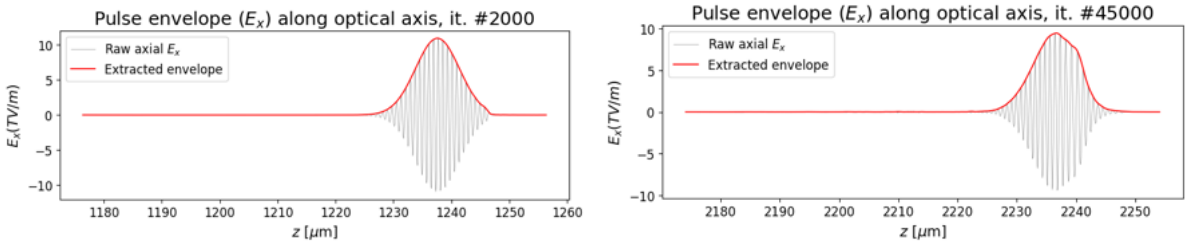

Fig. 19. An example of our amplitude envelope extraction for two iterations of our simulation using the E_x component of the net electric field. This allows us to visualize the depletion of the laser along the optical axis. Note how the leading edge of the near-Gaussian envelope has deformed from an early iteration (top) to after the pulse has begun to propagate through the plasma electrons (bottom) due to depletion of the pulse.

Examining the E_z component of the electric field along the optical axis yields information about the accelerating gradients seen by the injected electrons. In Figure 20, we notice steep

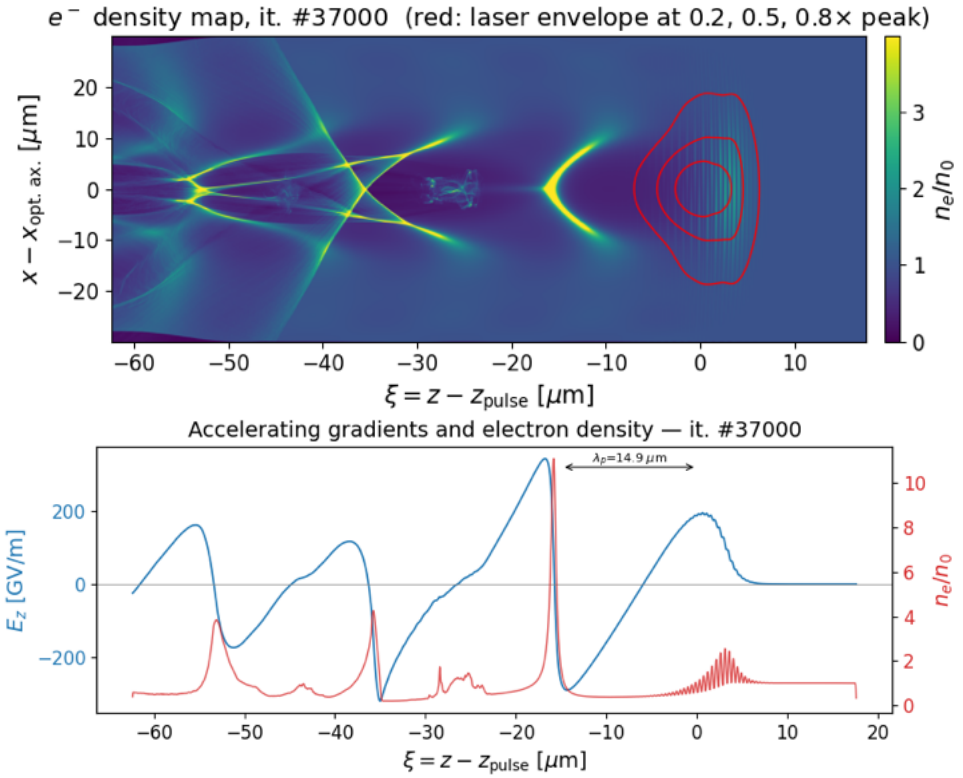

Fig. 20. A stacked view of (top) a snapshot of the electron density profile and (bottom) a lineout showing longitudinal accelerating gradient and electron density along the optical axis. The theoretical plasma wavelength is juxtaposed against the length of one plasma bubble for reference.

accelerating gradients at points of high electron density, indicating the plasma wake. There are also noticeable perturbations in the electron density where the injected electron bunches sit.

Another metric we can examine is the evolution of the spot size as the laser propagates, to understand the effects of self-focusing on laser profile. Using the electric field envelope extracted as described above, we can square it (since $I \propto A^2$) before processing to find a best-fit mask. This was done using an algorithm similar to that in `laserbeamsize` package's `.beam_size` function, adhering to the ISO-11146 standard of the beam intensity second moment method [11], [12]. This algorithm calculates the second moment of intensity along each axis:

$$\sigma_x^2 = \frac{\iint (x - x_c)^2 \, I(x,z) \, dxdz}{\iint I(x,z) \, dxdz}.$$

Where the 'centre of intensity' along said axis is:

$$x_c = \frac{\iint x I(x,z) \, dxdz}{\iint I(x,z) \, dxdz}.$$

And then iterates, considering only the pixels within $n\sigma_x$, where $n \sim 2-4$, until convergence. Along each axis, the beam extent is taken to be $4\sigma_x$ by convention.

Third, we created a notebook specifically for plotting and evaluating the resulting electron beam. Similarly, usage of the notebook requires setting the `PROJECTS_DIR` and `SCRATCH_DIR` paths as well as `JOB_ID` variable for the specific name

`TestRun_12345678` of the data and metadata folder to be analyzed. The notebook uses an HDF5-enabled version of `openpmd-api` to read the `.h5` files and extract their time series, which gives access to particle data like normalized momenta, energy, and position.

A first useful visualization is to bin the electrons by kinetic energy:

$$KE = (\gamma - 1)\, m_e c^2.$$

As shown in Figure 21. Note that particle-in-cell codes use the concept of 'macroparticles', which group together multiple 'real' particles (e.g. several electrons constitute one macroparticle) for computational efficiency. To get a representative frequency count for our histogram bins, we must account for the *weight* of each macroparticle, or how much charge it represents. We can see a clear separation between the lower energy electrons ejected from the plasma

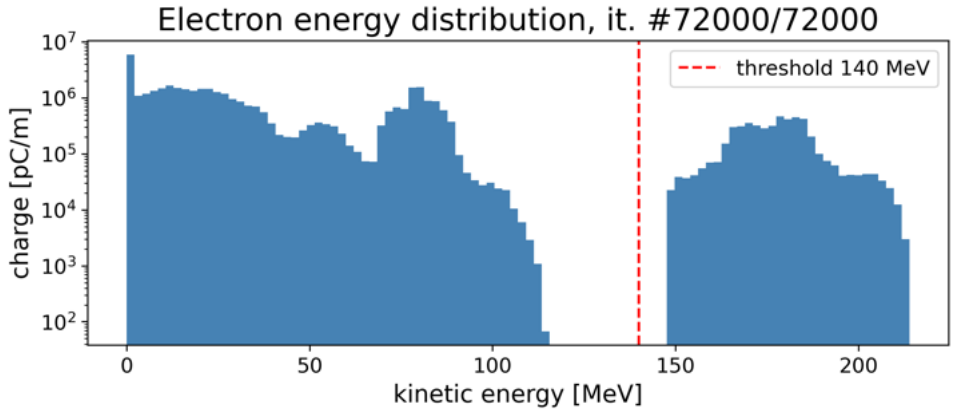

Fig. 21. A histogram of the total charge in each kinetic energy bin at the end of a WarpX simulation once the beam has exited. Note that since this is a 2D xz WarpX simulation, the charge is reported as $\frac{C}{m}$ extruded into the y-direction. In general, this is not a very accurate way to get total charge expected in a real LWFA experiment, but is still appropriate for relative distributions.

and the high-energy injected bunch in the first bucket.

Similarly, we can plot the energy spectrum as a spectral charge distribution line plot as in Figure 22, superimposing multiple iterations of the simulation to view beam evolution. This uses the histogram bins to calculate a $\frac{dQ}{dE}$ for each energy interval, measuring the amount of charge contained in each energy interval. Numerically integrating this curve across the entire energy range recovers the net beam charge.

A second characteristic diagnostic plot for the electron beam is the *phase space*, where the kinetic energy is plotted against position. This plot gives view of any high energy particle bunches, their location, and their spectral spread. A compact blob at high energy indicates a high-quality electron beam, with defined spatial extent and monochromaticity. A diagonal slash indicates a semi-localized electron bunch with a range of energies, due to the varying accelerating gradients present at differing positions in the plasma wake. This is referred to as an 'energy chirp', where a range of particle energies in an electron beam degrades spectral quality due to non-uniformity in the accelerating gradient. See Figure 23 for a comparison of two phase spaces from two different conditions, producing varying electron beams. If there are multiple bunches of electrons, we will see a periodic pattern with spatial period of $\sim \lambda_p$, as clusters of high-energy electrons will exist in sequential plasma bubbles.

While the longitudinal phase space can help us understand the physical distribution and density of electrons with varying energies, it cannot tell us about the coherence of the beam, an important aspect of beam quality. We can instead plot the transverse momentum against transverse position with respect to the optical axis, or the transverse phase space. In our case, we plot the dimensionless normalized x-momentum $u_x = p_x/m_e c$. The transverse phase space

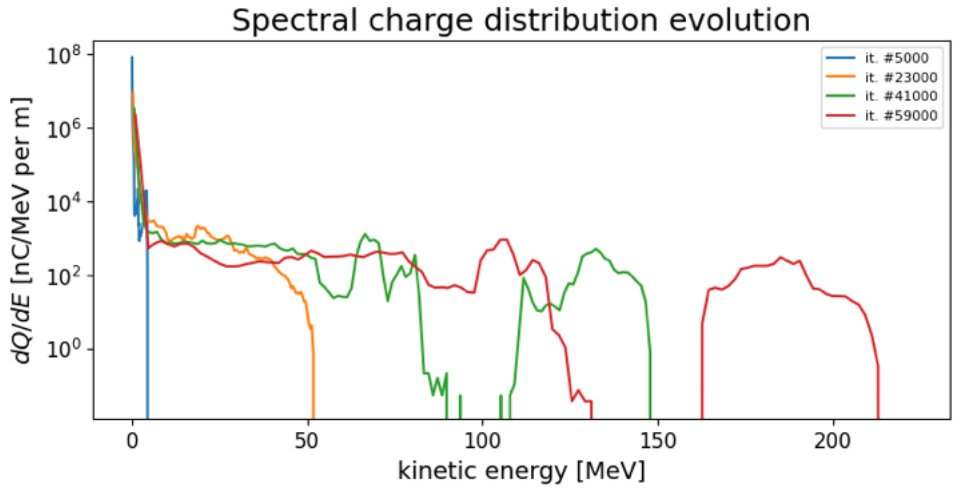

Fig. 22. A view of the spectral charge distribution of the run evolved through multiple iterations. The acceleration of electrons is seen through the flattening of the distribution, as well as the development of an 'island', or our electron beam.

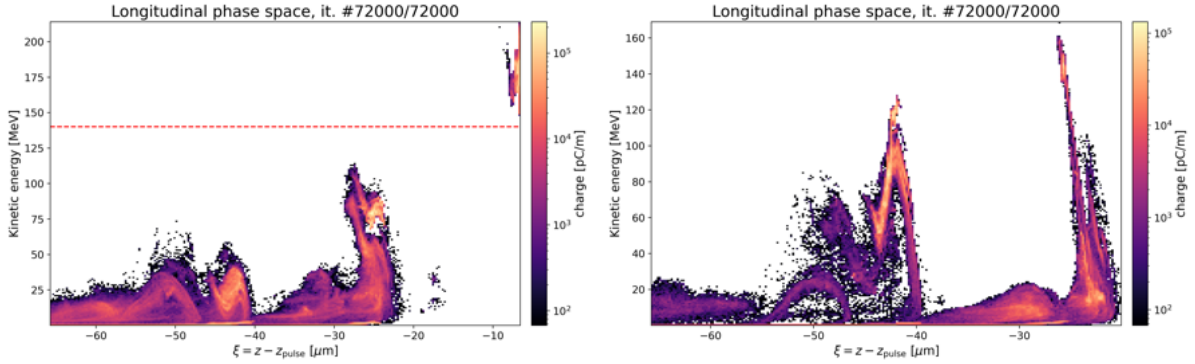

Fig. 23. A comparison of two phase spaces of an electron beam after exiting the accelerating plasma. The left figure shows a 'good' phase space, where there is a clearly localized electron beam from the first bucket with good spatial and energy coherence. The right figure shows a 'bad' phase space, with significant energy chirp and multiple smeared bunches of electrons.

of a particle beam tells us about its transverse profile – whether it is converging, at focus, or diverging, and to what extent it has spread in transverse position and momentum.

The particles will form an ellipse when plotting their transverse phase space. The equation for this ellipse is defined by the Courant-Snyder parameters [13]:

$$\gamma x^2 + 2\alpha x u_x + \beta u_x^2 = \epsilon.$$

Where:

- β describes the beam size, or how much it spreads out. As β increases with ϵ held constant, the beam occupies a larger area with a lower transverse velocity spread;
- α describes whether the beam is converging or diverging, i.e. the slope of the ellipse's major axis in the phase space. The beam is converging for $\alpha > 0$, focused at $\alpha = 0$, and diverging for $\alpha < 0$;
- γ relates to the beam's angular divergence, and is found using $\beta\gamma - \alpha^2 = 1$. As γ increases with ϵ held constant, the beam occupies a smaller area but has a larger transverse velocity spread;
- ϵ is the (transverse) emittance, and describes the size of the ellipse in the (transverse) phase space.

The RMS normalized emittance is defined as:

$$\epsilon_{rms} = \sqrt{\langle x^2 \rangle \langle u_x^2 \rangle - \langle x u_x \rangle^2}.$$

For normalized momenta $u_x = \beta\gamma$, where β and γ are the traditional relativistic quantities for that electron.

There is a relationship between the statistical definition for RMS emittance and the Courant-Snyder parameters:

$$\beta = \frac{\langle x^2 \rangle}{\epsilon_{rms}}, \quad \alpha = -\frac{\langle x u_x \rangle}{\epsilon_{rms}}, \quad \gamma = \frac{\langle u_x^2 \rangle}{\epsilon_{rms}}.$$

In general, a lower emittance indicates a beam with better spatial resolution and limited divergence. Under Hamiltonian forces, i.e. when the beam propagates in free space after the plasma acceleration stage, the emittance of the beam is constant. This follows from Liouville's theorem [14].

WarpX has a reduced diagnostic called 'BeamRelevant', which computes these parameters (and more) for a specific beam species. Unfortunately, as our only species is electrons, this diagnostic has no way of differentiating between the background plasma electrons and the injected electron beam, as we do this filtering with our threshold energy. As such, our processing must be done manually.

D. Parameter Scan

We will present a simple parameter scan, evaluating the effect of focal position within the plasma on beam production. We are using parameters obtained via our scaling laws:

- $A_0 = 13.6 \cdot 10^{12} \frac{N}{C}$ for $a_0 \approx 3.4 > a_{0c} \approx 3.1$;
- $\tau = 18\,\text{fs}$;
- $w_0 = 8.1\,\mu\text{m}$;
- $n_e = 5.9 \cdot 10^{18} cm^{-3}$;
- Plasma length = $1.5\,\text{mm}$.

We will scan focal positions at $z = 0\mu m, 300\mu m, 600\mu m, 900\mu m, 1200\mu m$.

We note that there is significant self-focusing of the laser as it propagates through the plasma. This leads to an earlier and tighter focal spot that the input parameters would lead to believe. Tightening w_0 results in an increased a_0 at peak focus. However, this deviation

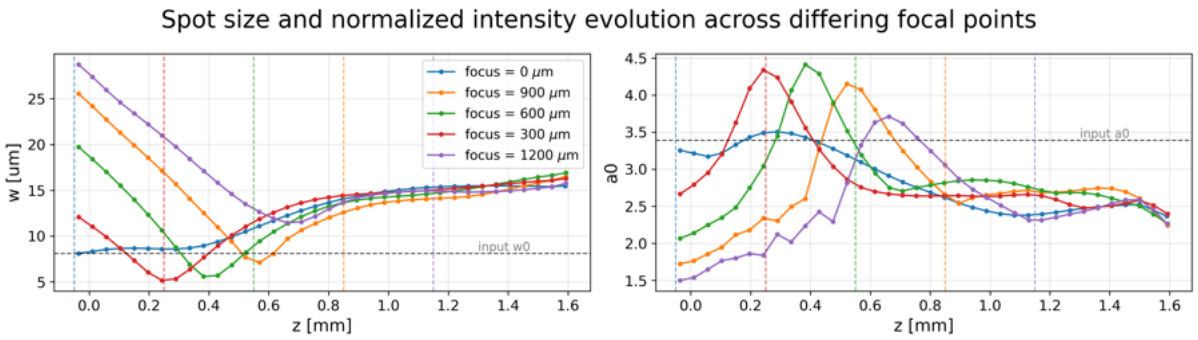

Fig. 24. The evolution of laser spot size and normalized laser vector potential as the laser pulse propagates. The input focal points are annotated with color-coded vertical lines. The input w_0 and a_0 are annotated with horizontal lines. Observe the self-focusing effect of the laser pulse travelling through the plasma.

from expected w_0 means our scaling law estimates for L_{deph} are not accurate. The actual dephasing length contracts due to the self-focusing, meaning the target must shrink as well.

As seen in Figure 25, the energy of the first bunch of electrons immediately trailing the laser pulse varies depending on simulation progression and focal spot. The varying focal spot changes electron injection behaviour. Along each series, we notice a rise and subsequent

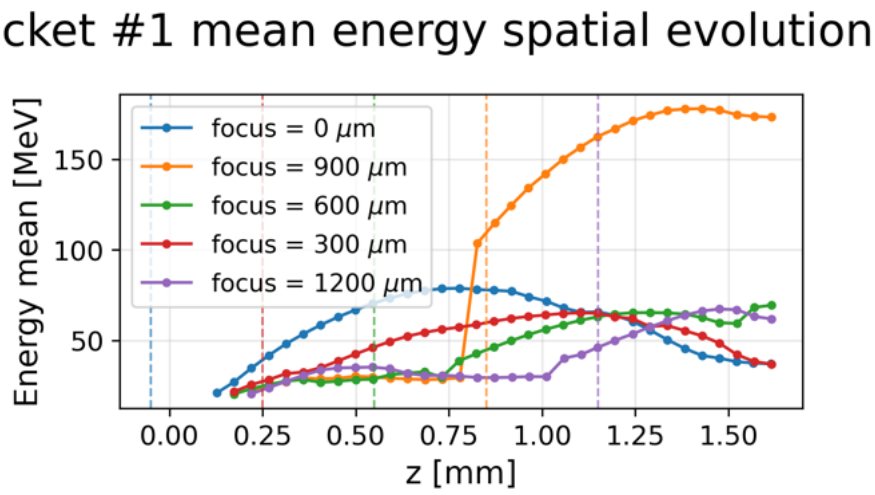

Fig. 25. Evolution of the mean electron energy in the immediately trailing bucket as the pulse progresses, shown for various focal spots within the plasma.

fall of bunch energies due to dephasing at the end of the plasma. Despite our scaling law estimates of $L_{\text{deph}} \approx 1.6mm$, this occurs over a much shorter distance due to self-focusing altering the laser parameters. However, more work must be done to better understand electron self-injection and acceleration for varying parameters.

VII. Interferometric Characterization

We seek to conduct experimental validation of our gas nozzles by measuring the characteristics of their produced exhaust plumes. We will first do this at atmospheric ambient pressure to walk through the analysis of an interferogram. We will then present the results found at a vacuum pressure of $100\,\text{Pa}$ to compare with our Ansys Fluent simulation results.

A. *Theory*

During the creation of a plasma from our gas target, each gas particle can be assumed to ionize a certain number of times, depending on the specific gas used. For example, each helium atom will produce two free electrons in the plasma, providing a simple stoichiometric relation between neutral gas density and free-electron plasma density.

The density of a neutral gas can be determined by the phase shift picked up by a probe beam with respect to a known reference. This occurs due to the density-dependent refractive index of a gas. We can quantify this via the Gladstone-Dale relation, declaring simple proportionality (via gas-specific Gladstone-Dale constant k) between refractivity and density:

$$n - 1 = k\rho \tag{24}$$

We could alternatively use the Lorentz-Lorenz equation, which can be seen as a complication of the Gladstone-Dale relationship [15]:

$$\frac{n^2 - 1}{n^2 + 2} = \frac{4\pi}{3} N\alpha \tag{25}$$

Where n is the refractive index of the medium, N is the number of molecules per unit volume, and α is the mean polarizability of a single molecule. Note that the interferometry

data can only give us information on the average density along the geometrical path taken by the light, with accumulated phase:

$$\phi(x,y) = \frac{2\pi}{\lambda}\int_L n(x,y,z,\lambda)\,dz = \frac{2\pi}{\lambda}L\overline{n}(x,y,\lambda) \tag{26}$$

Where $\overline{n}(\lambda)$ is the average refractive index for wavelength λ along the taken path [16]. This intuitively follows from thinking of the measurement arm's phase shift as an integration of the density-induced phase shift while the beam passes through the plume.

The laser we will use for our interferometric characterization is a green HeNe laser, with $\lambda = 543.5nm$. Relevant gas characteristics were calculated from parameters retrieved at `refractiveindex.info` and the NIST Computational Chemistry Comparison and Benchmark DataBase [17], [18]. This gives the Gladstone-Dale constants for argon and helium as:

$$k_{Ar} = 0.0001585\frac{m^3}{kg}, \quad k_{He} = 0.0001957\frac{m^3}{kg}.$$

And the molecular polarizabilities as:

$$\alpha_{Ar} = 1.664\,\text{Å}^3, \quad \alpha_{He} = 0.208\,\text{Å}^3.$$

Using the density profiles found from our Ansys Fluent CFD simulations, we can calculate the expected phase shift for a ray along the optical axis in our interferometer. We take the density lineout, and using the constants found above, numerically integrate along the laser path, calculating the refractive index at each point, and sum the relative phase shift acquired due to the density modulations. We find that for higher pressures, we should expect phase

TABLE II
CALCULATED PHASE SHIFT FOR A BEAM PASSING THROUGH THE SIMULATED EXHAUST PLUME WITH BACKING PRESSURE P, ALONG A PLANE OF CYLINDRICAL SYMMETRY, AT VARIOUS DISTANCES x_t FROM THE NOZZLE EXIT.

	x_t (mm)						
P (bar)	0.0	0.4	0.8	1.2	1.6	2.0	2.4
5.0	0.64	0.62	0.60	0.58	0.55	0.53	0.51
13.8	1.8	1.7	1.7	1.6	1.5	1.5	1.4
17.2	2.3	2.2	2.1	2.0	2.0	1.9	1.8
20.7	2.7	2.6	2.5	2.4	2.3	2.3	2.2
27.6	3.6	3.4	3.3	3.2	3.1	3.0	2.9

shifts approaching π, meaning a light fringe is shifted to overlap with a dark fringe.

A two-arm interferometer compares the phase shift acquired by a beam passing through the gas target to a reference beam travelling in vacuum. For example, a standard form of a Mach-Zehnder interferometer could be used, splitting the beam in vacuum and passing the measurement arm through the exhaust plume, before recombining to recover the interference pattern. An alternative form of a two-arm interferometer is a folded Mach-Zehnder. In this design, the incident beam first passes through the target, where only a specific, spatially distinct portion of the beam interacts with the gas. Then, the idiomatic Mach-Zehnder beam splitter optics are aligned to overlap an undisturbed portion of the incident beam with the part that has interacted with the target [1]. This has the advantage of not requiring opening the vacuum chamber for realignment, as unlike in the first Mach-Zehnder form, the optics are not required to be inside the chamber. For simplicity, we will construct a standard Mach-Zehnder interferometer.

Let's examine the operating principle of a standard Mach-Zehnder interferometer. We will approximate our collimated laser beam as a plane wave, which can interact with our optics

without aberration. Consider two plane waves of equivalent angular frequency travelling at a relative angle of θ in the x-z plane:

$$E_1(\boldsymbol{r}, t) = E_0 \exp\left[i\left(\boldsymbol{k_1} \cdot \boldsymbol{r} - \omega t\right)\right]$$
$$E_2(\boldsymbol{r}, t) = E_0 \exp\left[i\left(\boldsymbol{k_2} \cdot \boldsymbol{r} - \omega t\right)\right].$$

For position vector $\boldsymbol{r} = x\hat{x} + y\hat{y} + z\hat{z}$. Take the two wave vectors to be spaced symmetrically above and below the z-axis:

$$\boldsymbol{k_1} \cdot \boldsymbol{r} = kx \sin\left(\frac{\theta}{2}\right) + kz \cos\left(\frac{\theta}{2}\right)$$
$$\boldsymbol{k_2} \cdot \boldsymbol{r} = -kx \sin\left(\frac{\theta}{2}\right) + kz \cos\left(\frac{\theta}{2}\right).$$

The superposition of these two plane waves is the sum of their electric fields:

$$E_{\text{net}} = E_1 + E_2 = E_0 e^{-i\omega t} e^{ikz\cos\frac{\theta}{2}} \left(e^{ikx\sin\frac{\theta}{2}} + e^{-ikx\sin\frac{\theta}{2}}\right).$$

Applying Euler's:

$$E_{\text{net}} = 2E_0 \cos\left(kx \sin\frac{\theta}{2}\right) e^{i\left(kz\cos\frac{\theta}{2} - \omega t\right)}.$$

Note that the intensity is proportional to $|E|^2$:

$$I \propto 4E_0^2 \cos^2\left(kx \sin\frac{\theta}{2}\right) = 2E_0^2\left[1 + \cos\left(2kx \sin\frac{\theta}{2}\right)\right].$$

When $\theta = 0$, the oscillatory term of our transverse intensity pattern disappears, resulting in a constant intensity. This reflects the idealized case where the two beams undergo the exact same optical path length in the reference and measurement arms, and recombine perfectly aligned and superimposed. In this scenario, it is as if we had not split the beam in the first place. However, adding $\theta \neq 0$ introduces sinusoidally oscillating light and dark fringes along the x-axis, which become more tightly spaced with increasing relative beam angle, as shown in Figure 26.

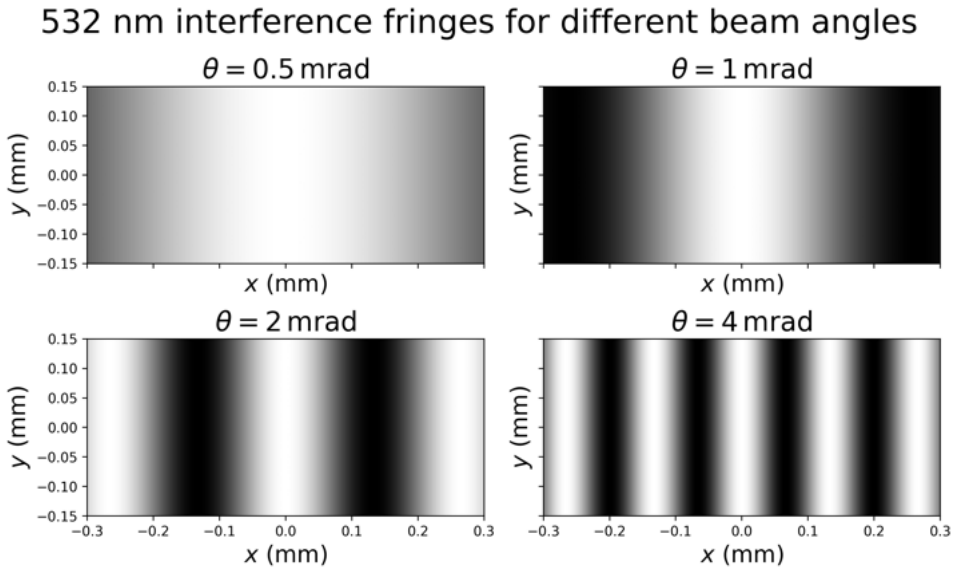

Fig. 26. A visual showing the differing fringe spacing for relative angle θ.

Increasing the relative beam angle increases the spatial resolution attainable by our interferogram, in that the fluctuations in fringe pattern can be made out to more detail. Of course, this model is an idealization. A first practical limitation to our interferogram is the 'bleeding' of light fringes into dark fringes at higher resolution, which lowers the contrast of our image. This blurring of our data limits the realistic spatial resolution practical for collecting our interferogram; we must balance the tightness of our fringes with maintaining high contrast. A second practical consideration is the presence of aberrations in our setup. A perfect plane wave is not a reasonable expectation in the lab, which will affect the characteristics of our interferogram. See figure Figure 27 for an example of astigmatic aberration present in our interferogram.

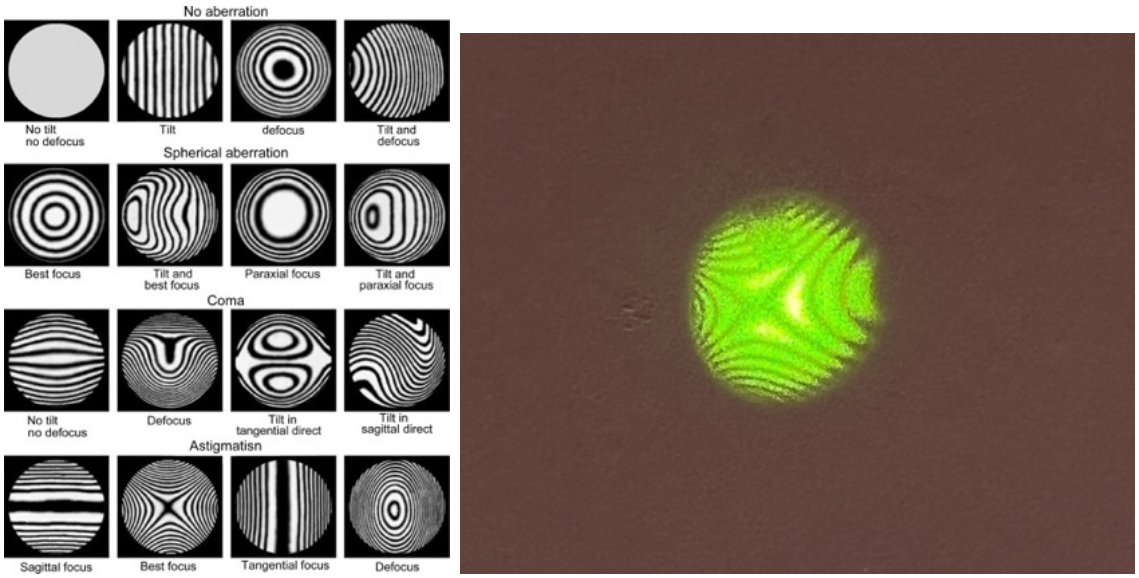

Fig. 27. (left) A visual of possible interferogram manifestations of aberrations [19]. (right) A view of an in-focus astigmatism aberration, imaged using the setup described below.

B. Experimental Setup

Our optical experimental setup can be seen in Figure 28. This involved a HeNe laser, expanded to roughly 1 inch in diameter, and re-aligned to the beam height and through hole ports of the vacuum chamber. All optics up to the first beam splitter serve this purpose, such as the beam expander and the 'periscope' mirrors to vertically shift the beam height to adapt to the vacuum chamber. The beam then enters the canonical Mach-Zehnder interferometer, where mirror 7 is in the reference arm and mirror 8 is in the measurement arm. The re-combined beam then passes through our 4f imaging optics, a set of two converging lenses with overlapping focal points. This captures the target on the object plane, one focal length behind the first imaging optic, and the CCD on the image plane, one focal length after the second imaging optic. Together, the system spans 4 focal lengths, hence the name '4f imaging system'.

Because of diffraction at density gradients, placing a CCD camera at the output of the interferometer will suffer from blurring and loss of phase localization. The imaging setup can be used to retain the phase detail at the plane of the target for imaging. Note that the fringes and their spacing are an artefact of the relative angles between the two beams, and can be adjusted by the mirror and beam splitter setup in the interferometer. As such, any imaging optics will not affect the fringes [1].

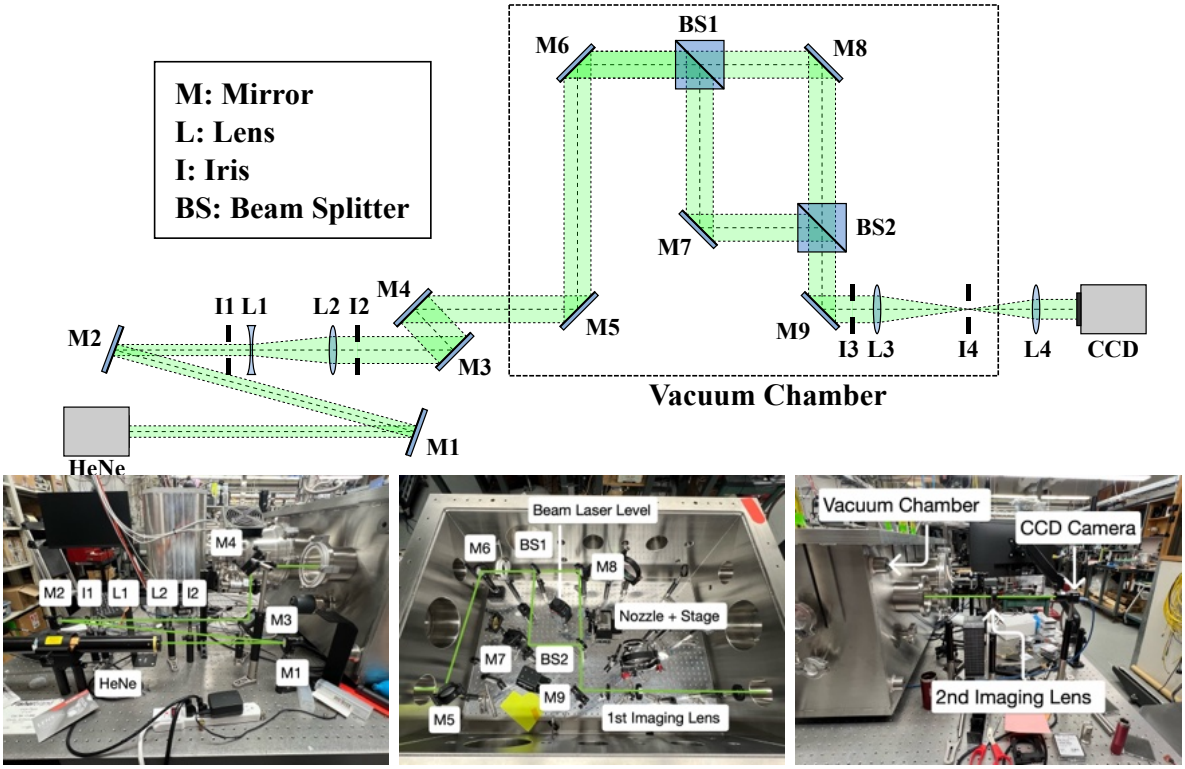

Fig. 28. A schematic of our optical setup for the neutral gas density interferometry, featuring the input HeNe alignment, Mach-Zehnder interferometer, and 4f imaging system. The diagram is not to scale. The laser level is used to set the beam height in the vacuum chamber.

Once aligned, we can pass the laser through the system without activating the gas jet and view our interferogram. See Figure 29 for our reference interferogram at different beam relative angles θ.

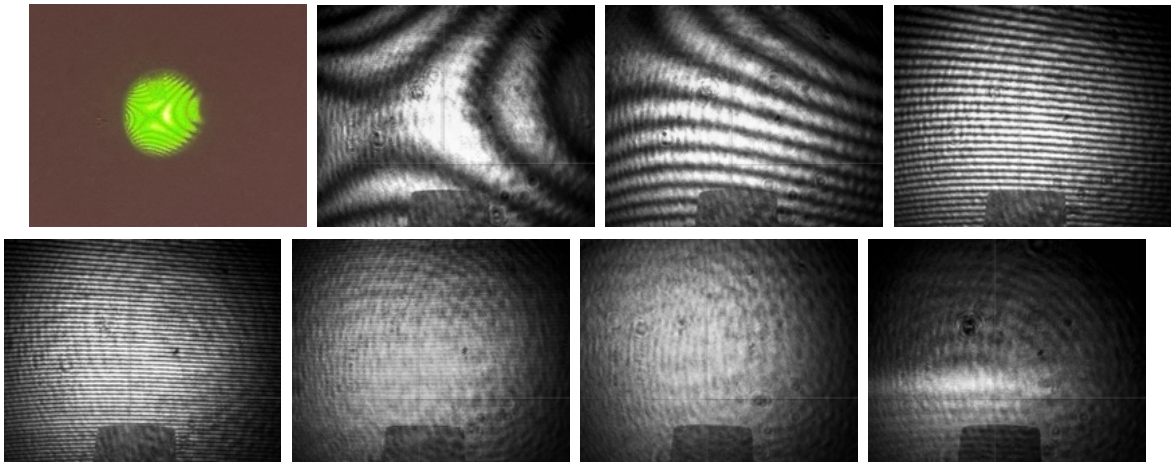

Fig. 29. (top left) An image of our baseline interferogram, without the gas jet activated, on an index card. (left to right, top to bottom) A CCD camera image of the interferogram at increasing angles. The notch at the bottom of each image is the nozzle tip. In the last image, the reference beam is vertically shifted out of superposition with the measurement beam, losing the interference fringes.

We used this setup to collect interferograms in two ways:

- First, we collected interferograms using argon expelled into atmospheric pressure, which we use to walk through our interferogram processing methods;

- Second, we collected interferogram using argon expelled into $100\,\mathrm{Pa}$ vacuum conditions, which we use to benchmark our nozzle against our Ansys Fluent simulations.

See Figure 30 for a view of the argon gas line setup. Copper tubing was used to withstand

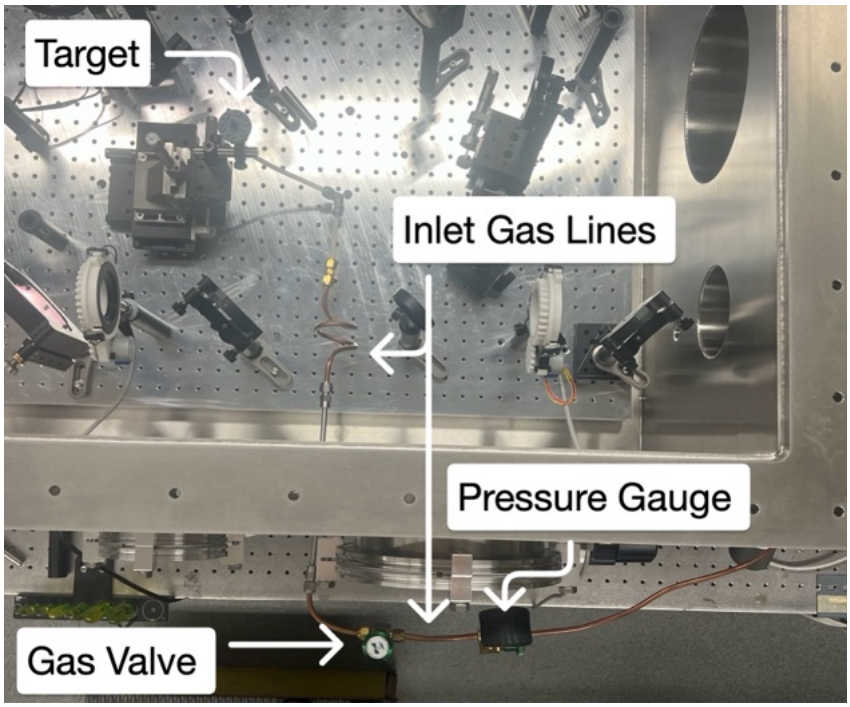

Fig. 30. The gas lines for supplying argon into the de Laval nozzle.

the high pressures, exceeding 10 atmospheres.

C. *Interferogram Processing*

Consider the case of argon at 200PSI and 250PSI expelled into atmospheric air pressure through the nozzle.

To process our interferograms, we followed the procedure outlined in Vojtěch Janota's thesis [20]. For visualization purposes, consider a 1D Fourier transform collected perpendicular

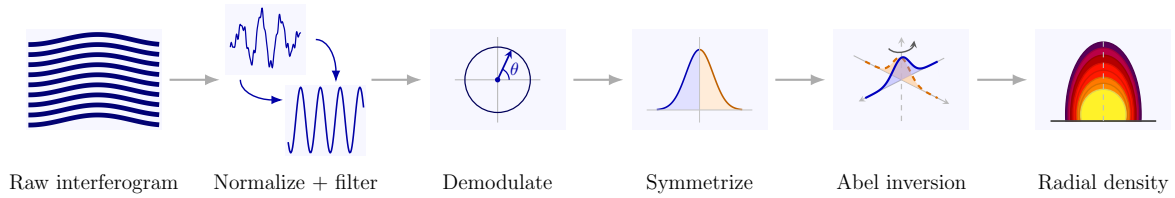

Fig. 31. A simplified line diagram illustrating the interferogram processing workflow.

to the fringes, shown in Figure 32. From the frequency spectra, we notice that there are three primary spectral regions:

- Two spikes at $\pm f_0$, where f_0 is the spatial frequency of the interference fringes;
- A central spike around $f = 0$ due to low-frequency noise.

The other, smaller local maxima around $+f_0$ can be attributed to the lower-frequency airy ring pattern of the laser beam faintly visible in the interferograms, and higher harmonics

of f_0. Figure 32 displays a before filtering and after filtering comparison of the frequency spectra, highlighting these three lobes and the filter's bandpass effect.

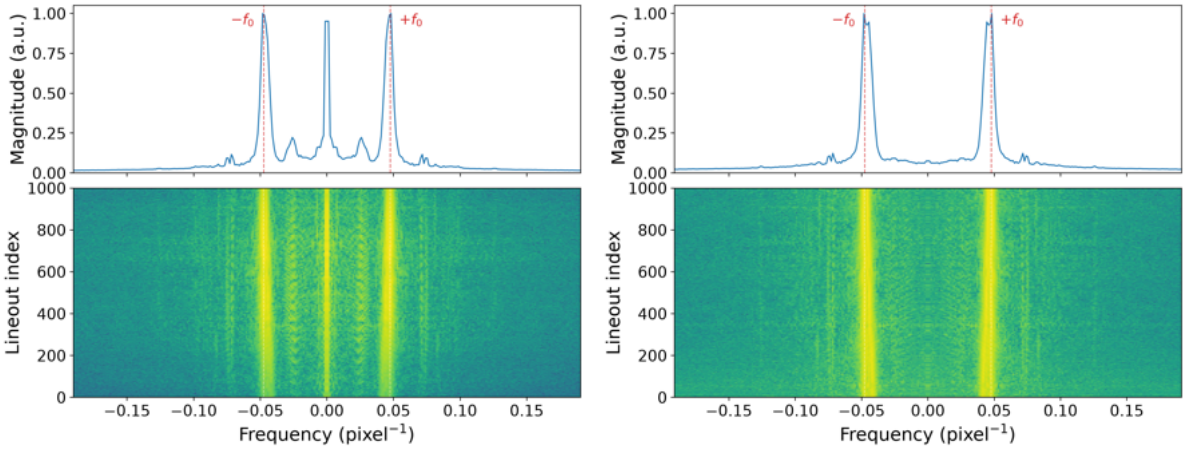

Fig. 32. A comparison of the raw (left) and normalized (right) frequency spectra, showing the noise suppression.

In reality, we use a 2D Fourier transform to capture spatial frequencies in either direction, which will allow us to better reduce noise in the interferogram. However, the underlying principle is the same as in the 1D case. We use the spikes corresponding to the interference fringes to find the spatial period (in pixels) of the fringes. Knowing which frequency to look for helps us tailor our filter to reduce noise, and also helps us normalize our fringe intensities by telling us what spacing to look for between consecutive minima and maxima.

First, we normalize the interferograms. We interpolate between consecutive fringe minima, using the known period to minimize noise impact, subtract the piecewise noise floor, then normalize the interval between each minimum by the intensity of the contained local maximum. See Figure 33 for a view of the processed interferogram.

Next, we filter the frequency spectrum. After plotting and filtering the spectrum, with example shown in Figure 34, we can inverse transform to recover the cleaned interferogram. However, by discarding all but the positive side-band, including filtering out the negative frequency side band, our inverse Fourier Transform will recover a complex signal, with magnitude and phase information. Completing this process twice, once with a reference measurement and once with the target, allows us to multiply by the complex conjugate of our processed reference fringes to demodulate the desired phase signal. What results is a map containing the phase change associated with differing refractive indices. The complex magnitude tells us about the intensity of the fringe information at that location, i.e. it is a measure of data quality. We use a mask to cover any nozzle geometry present in the interferogram, and erode the low-quality pixels around the edges to avoid phase unwrapping errors. The phase information is what we can use to recover our density, but first we must 'unwrap' the phase, as our complex phase is wrapped between $\pm\pi$[6]. An example is shown in Figure 35, with the interferogram, amplitude map, and unwrapped phase map with the applied mask.

The final step is to perform Abel inversion on our cylindrically symmetric jet to recover a 2D slice of the 3D density cloud. We must take care to align the exhaust plume vertically along a known axis of symmetry at horizontal position $x = x_{\text{axis}}$. To do so, we proceeded

[6]While phase unwrapping is in general necessary, our experimental phase shifts were found to be slightly lower than those predicted in Table II, which already seldom reached π. As such, it is not useful to visualize the effect of phase unwrapping with our data.

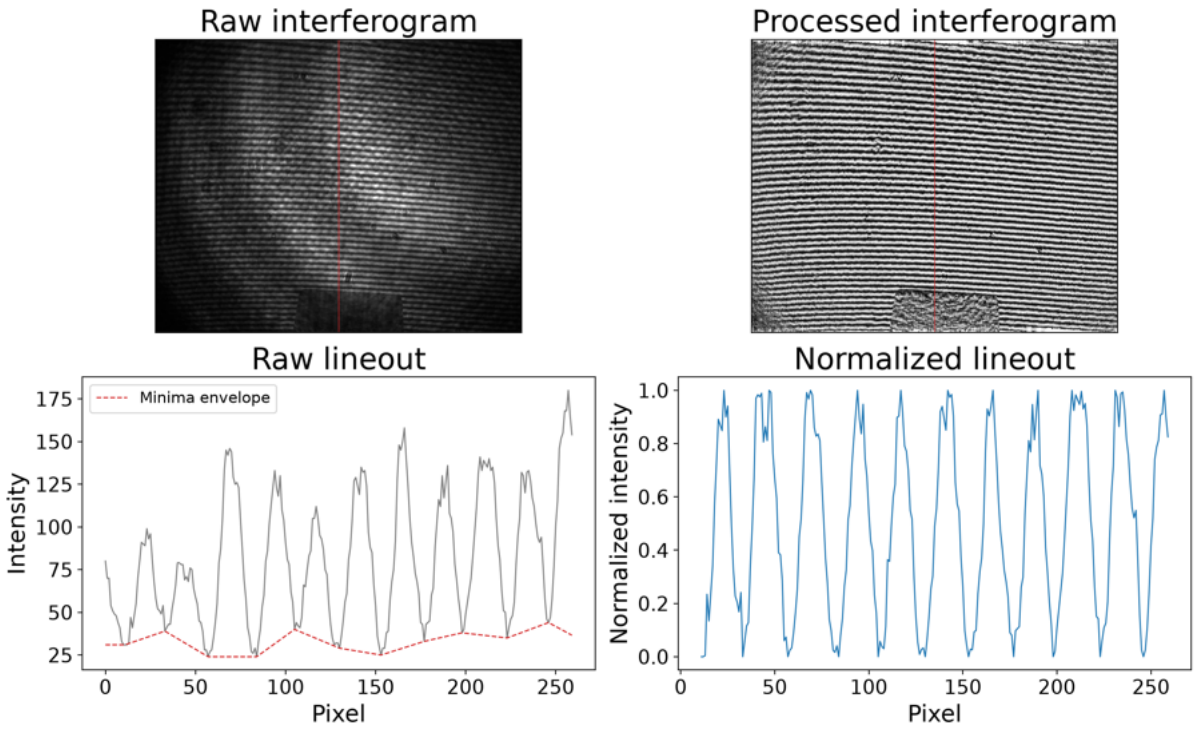

Fig. 33. A view of the raw interferogram and associated intensity lineout (left) against the processed and normalized interferogram, with associated lineout (right).

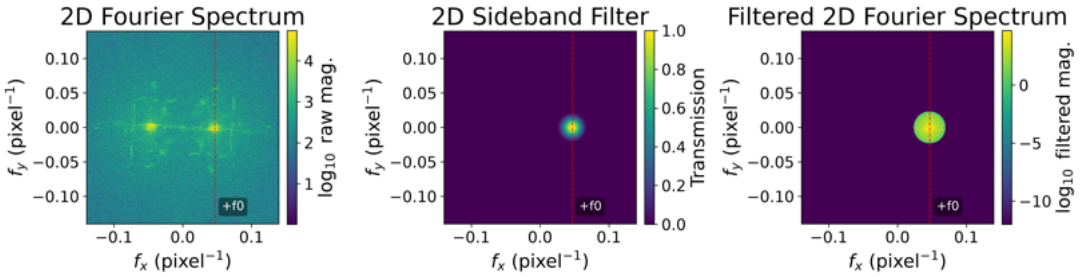

Fig. 34. An example of 2D Fourier spectra and filtering obtained via our Fourier interferogram analysis. The two bright spots in the unfiltered spectrum (left) are at $\pm f_0$. The filter (centre) targets just the spike at f_0, resulting in the filtered Fourier spectrum (right). Because our fringes are relatively horizontal, there is little offset into non-zero f_y, although there is a faintly noticeable tilt.

row-wise (perpendicular to the gas jet), fitting each intensity lineout with a linear + Gaussian fit:

$$y(x) = mx + c + A\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

This captured both a level of background noise in the lineout, while isolating the peak caused by the gas jet. Recording the position of each of these Gaussian peaks and filtering to improve data quality, we could apply a linear fit to the peak locations to recover the axis of the jet. We rotated the image to align this axis with the vertical and repeated until convergence. See Figure 36 for a visual of the pre-processing, post-processing, and fitting process.

We can now perform Abel inversion on the processed plume. Using the known diameter of the nozzle exit, being $3.5\,\mathrm{mm}$ from corner to corner, we can calculate a pixel conversion ratio to scale our image and calculate the density values. See Figure 37 for an example of

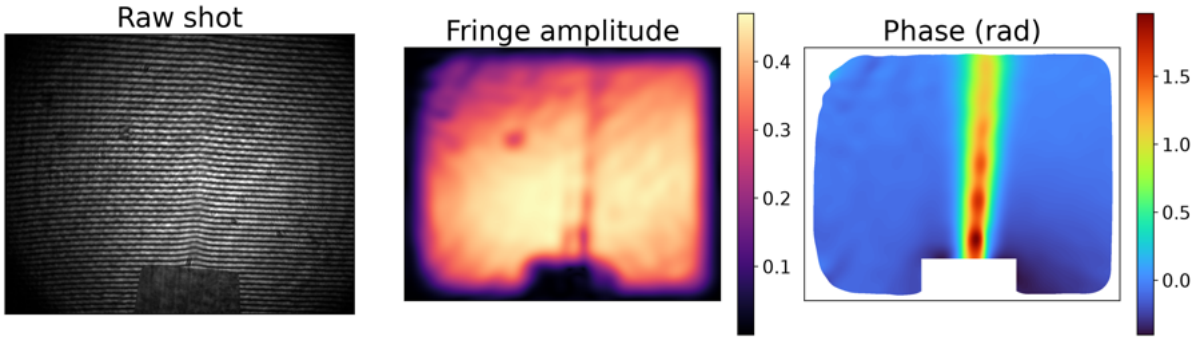

Fig. 35. A view of the 250PSI argon interferogram (left), associated amplitude map (centre), and lastly unwrapped phase map (right). Notice the rectangular cutout at the phase map bottom due to the nozzle shadow.

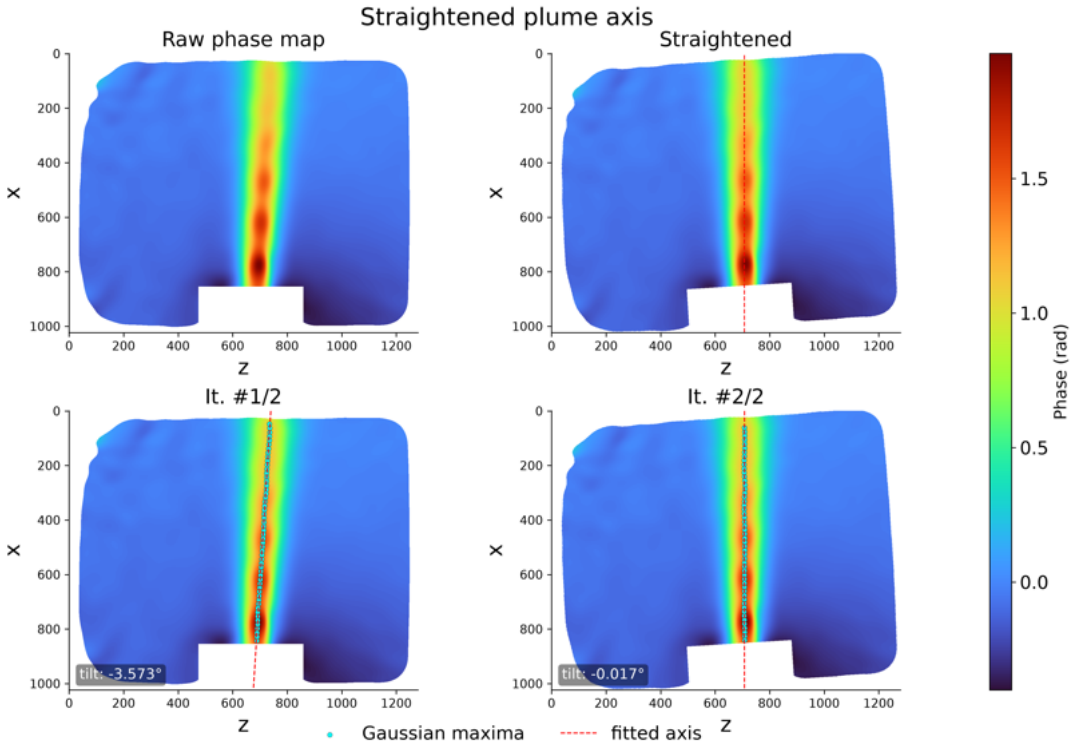

Fig. 36. A comparison of the phase map before and after straightening and aligning with a central axis. Along the top row, from left to right, shows the phase map before and after straightening using our fitted Gaussian method. Along the bottom row, the fitted maxima are overlaid on the phase map for the two iterations used.

the Abel inverted density cloud. From both the phase map and Abel-inverted density contour collected with argon at 250PSI under atmospheric conditions, we can clearly observe a series of ‘bubbles’ emanating from the exit. This occurs due to over expansion of the gas jet, which is not a concern for our target vacuum conditions, but does manifest when testing at kPa ambient pressures. See a reference example of over-expanded exhaust flow in Figure 38. While it is not directly comparable to our Ansys results, it is nonetheless useful to visually confirm our interferogram processing algorithm.

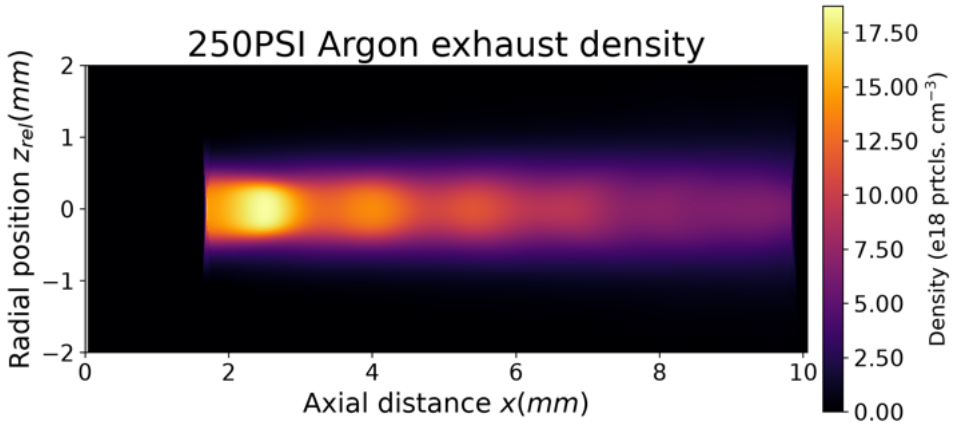

Fig. 37. The Abel inversion for our argon gas jet at 250PSI into atmospheric conditions. While the number density is not meaningful due to the displacement of air, it is a good example for the end-to-end interferogram processing.

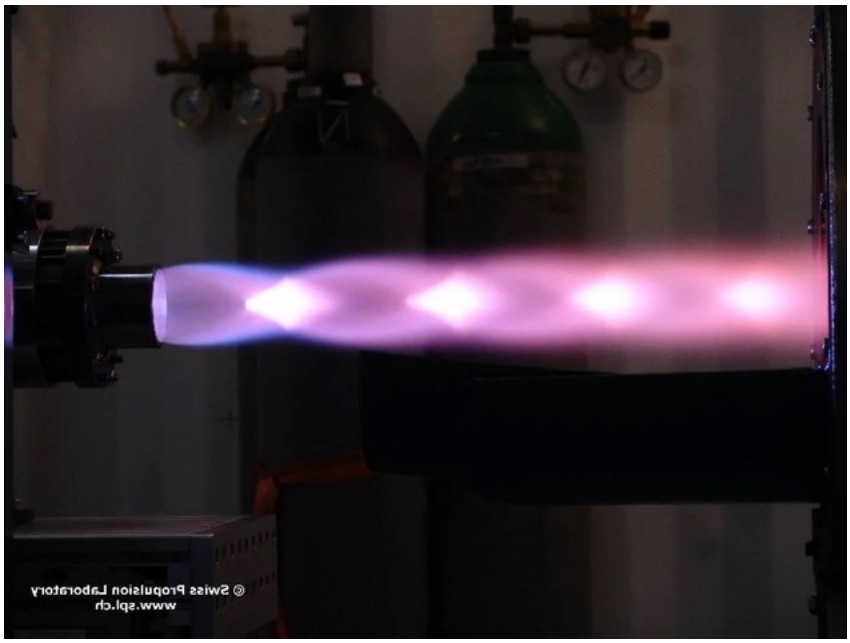

Fig. 38. An example of shock diamonds in the over-expanded exhaust flow of a rocket engine. Photo credit: Swiss Propulsion Laboratory

D. Results and Analysis

We can now compare the experimental data collected under vacuum and the results obtained from our Ansys Fluent simulations. We collected data for argon into $100Pa$ vacuum at three backing pressures: 250PSI, 300PSI, and 400PSI. The results are visualized in Figure 39.

We can first compare the density contour outputs between the Ansys Fluent CFD simulations, and the experimental data collected at the same conditions. The comparisons are shown in Figure 40. Qualitatively, the experimental contours are significantly less homogeneous than the simulated contours. The simulated contours have very broad 'tabletop' regions of high density that extend to either radial edge of the contour, at which point they meet a sharp drop-off to vacuum density. In the experimental density contours, the central regions of high density only occupy a portion of the full plume width, with a slower down ramp towards the radial edges of the contours. Additionally, there are fringes that appear axially downstream in the exhaust plume at the edges of the contour. This does not match

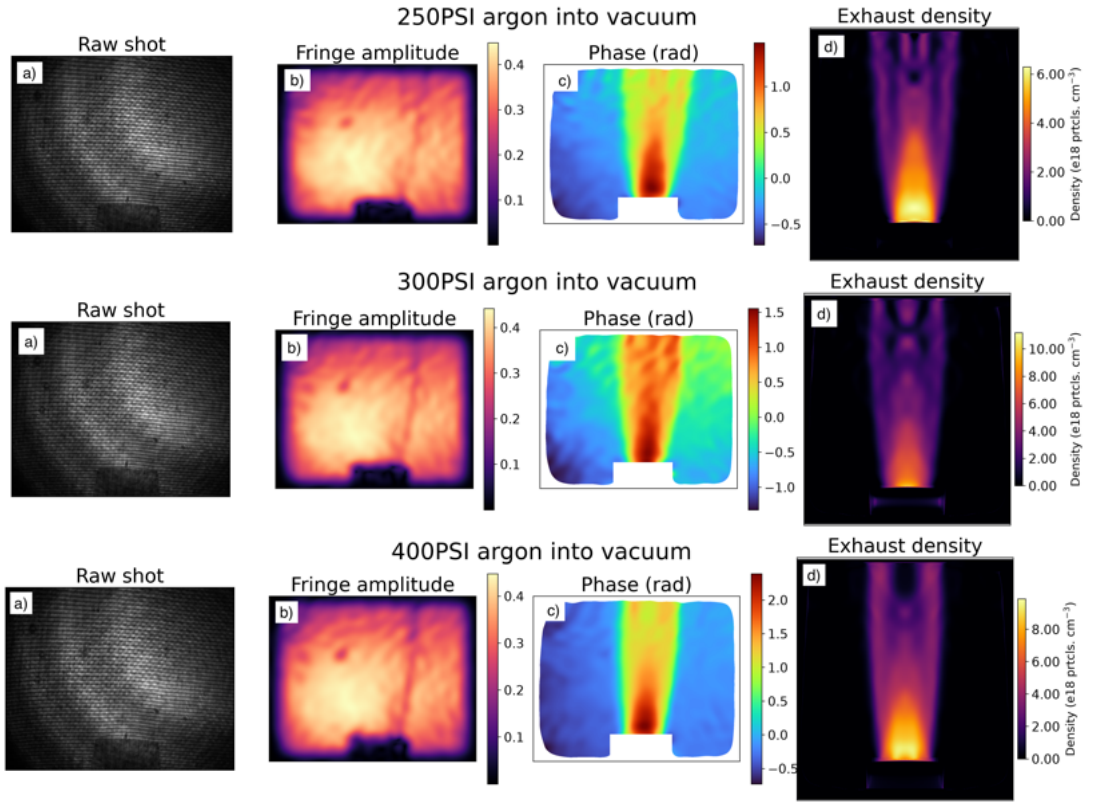

Fig. 39. A comparison of the interferograms and Abel inversions for each argon into vacuum trial. a) is the raw interferogram, b) is the complex amplitude map, c) is the phase map, and d) is the Abel-inverted radial plume density.

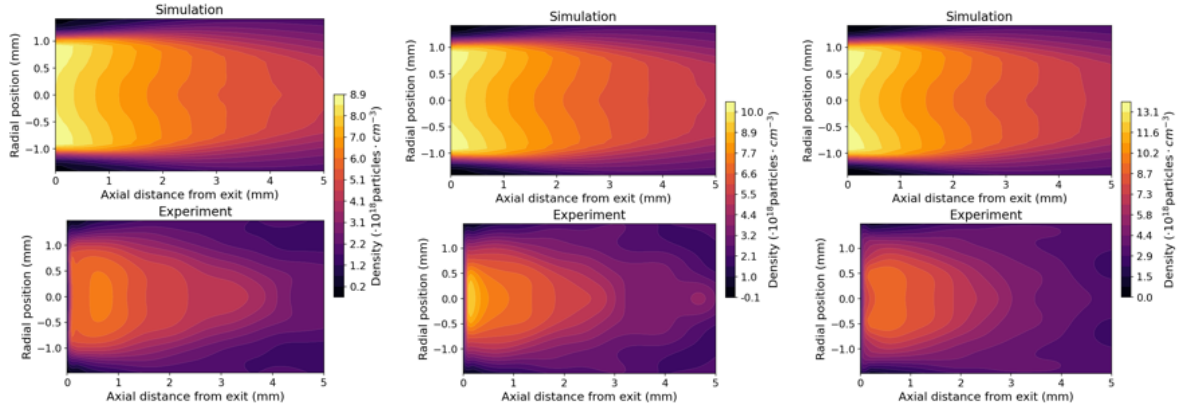

Fig. 40. Comparison between the simulated and experimental contours at the recorded backing pressures. From left to right, the pressures are 250PSI, 300PSI, and 400PSI.

the smooth taper of the simulated contours, likely indicating either imperfections in the nozzle geometry, or artefacts of the Abel inversion due to phase map noise further from the nozzle exit.

For a quantitative comparison, we can take radial lineouts at differing distances from the nozzle exit, similar to how we characterized our CFD profiles alone. The comparison for the 400PSI trial is shown in Figure 41; the other trials have similar results. We can observe that the quantitative analysis matches our qualitative observations. The experimental profiles in general have a lower density than the simulated outputs, likely due to variability in the dimensions of our manufactured nozzles. While efficient, the resin printing is limited in

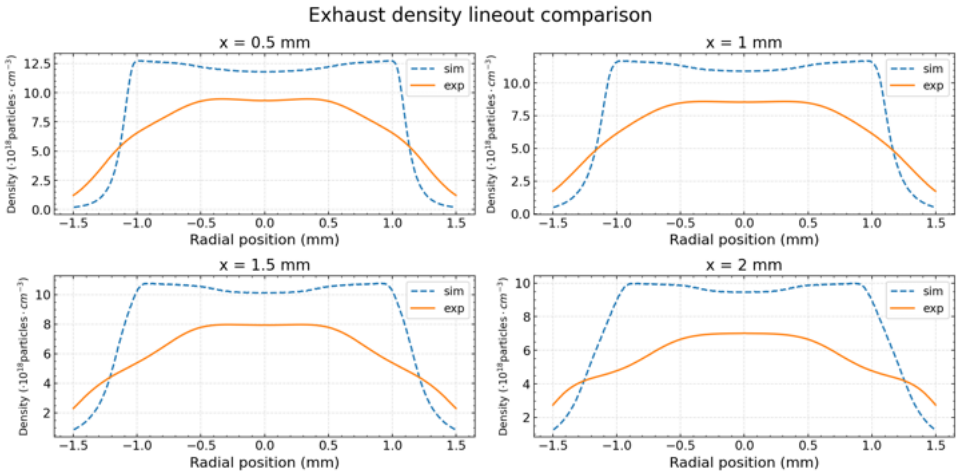

Fig. 41. A comparison of the experimental and theoretical lineouts at differing distances for the 400PSI trial.

accuracy, resulting in variations in our exact diameters and expansion ratios. Disregarding this error, the experimental profiles lack the sharp tabletop feature of the simulated outputs. They are more rounded, featuring a flat-esque section with broad density gradient shoulders. We must also attribute this error to manufacturing characteristics. We hypothesize that the layered construction of the nozzles exacerbates the development of boundary layers throughout the de Laval nozzle, which manifests as diffuse density gradient transitions. A more precise manufacturing method, such as micro spark erosion, has the potential to eliminate a significant portion of these gradients.

Lastly, we can extract the FWHM and the plateau density at varying distances from the exit, and compare these metrics to those of our simulated exhaust plumes. The comparison is shown in Figure 42 for the 400PSI trial; the other trials have similar results. We can observe that the plateau densities follow a similar linear trend between simulation and experiment, however with the experimental densities being lower. Similarly, the FWHM of the exhaust plume expands with distance similar to the simulated plumes, however, the experimental FWHM are in general larger. The larger FWHM exacerbates the density discrepancy, as the nozzle is spreading the expelled gas over a larger area.

Taken together, the nozzles successfully create a supersonic gas plume with densities close to our target. However, limitations in resin printing as an efficient manufacturing method take away from the precise emulation of our simulation results. To improve the density profiles, care must be taken to obtain higher-fidelity nozzles, which will push the rounded plume edges closer to the well-defined density gradients of our simulation. Additionally, our experimental characterization of our nozzles can help inform future Ansys Fluent simulations to better target our scaling law parameters.

VIII. Conclusion

We have presented the theoretical and technical processes involved in the development of a gas nozzle prototype for LWFA. We conducted reviews of literature to extract LWFA scaling laws and nozzle design practice, completed Ansys Fluent computational fluid dynamics simulations to fine-tune our nozzle geometry, explored WarpX particle-in-cell simulations to model the wakefield interaction, and experimentally validated a nozzle prototype with interferometric techniques. We were successfully able to produce a nozzle prototype generating a supersonic exhaust plume similar to that of our simulations. However, precise agreement between simulation and experiment demands more precise manufacturing methods and further design refinement.

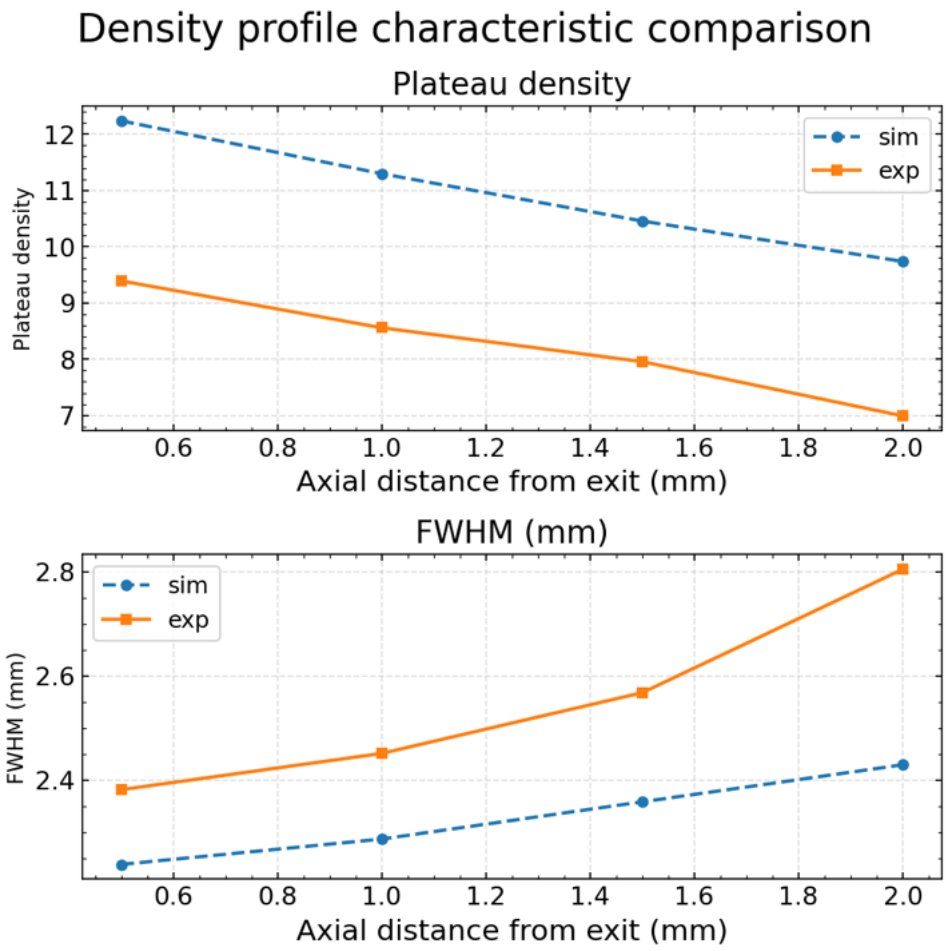

Fig. 42. A comparison of the density profile metrics between simulation and experiment for 400PSI

There is still room for significant future research on this topic. Some major areas of focus include:

1) Using WarpX to conduct deeper research into the causality relationships in LWFA, including but not limited to: electron injection regimes, gas profile variation effects, and laser focusing geometry;
2) Developing of more complex nozzle geometries to facilitate differing electron injection methods, such as shocked nozzles for density transition injection, or staged nozzles for longer acceleration distances;
3) Performing LWFA and measuring electron beam characteristics benchmarked against simulation results.

IX. References

[1] J. M. Cole, "Diagnosis and application of laser wakefield accelerators," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:114468308

[2] J. Faure et al., "A review of recent progress on laser-plasma acceleration at khz repetition rate," *Plasma Physics and Controlled Fusion*, vol. 61, no. 1, p. 014 012, 2019.

[3] W. Lu et al., "Generating multi-gev electron bunches using single stage laser wakefield acceleration in a 3d nonlinear regime," *Physical Review Special Topics—Accelerators and Beams*, vol. 10, no. 6, p. 061 301, 2007.

[4] B. A. Shadwick, C. B. Schroeder, and E. Esarey, "Nonlinear laser energy depletion in laser-plasma accelerators," *Physics of Plasmas*, vol. 16, no. 5, 2009.

[5] S. Lorenz et al., "Characterization of supersonic and subsonic gas targets for laser wakefield electron acceleration experiments," *Matter and Radiation at Extremes*, vol. 4, no. 1, 2019.

[6] K. Schmid et al., "Density-transition based electron injector for laser driven wakefield accelerators," *Phys. Rev. ST Accel. Beams*, vol. 13, p. 091 301, 9 Sep. 2010. DOI: 10.1103/PhysRevSTAB.13.091301 [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevSTAB.13.091301

[7] L. Rovige et al., "Symmetric and asymmetric shocked gas jets for laser-plasma experiments," *Review of Scientific Instruments*, vol. 92, no. 8, 2021.

[8] K. Schmid and L. Veisz, "Supersonic gas jets for laser-plasma experiments," *Review of Scientific Instruments*, vol. 83, no. 5, 2012.

[9] D. Gustas et al., "High-charge relativistic electron bunches from a khz laser-plasma accelerator," *Physical Review Accelerators and Beams*, vol. 21, no. 1, p. 013 401, 2018.

[10] J.-L. Vay et al., *WarpX: An advanced Particle-In-Cell code*, https://blast-warpx.github.io, https://doi.org/10.5281/zenodo.4571577, 2018. DOI: 10.5281/zenodo.4571577 [Online]. Available: https://doi.org/10.5281/zenodo.4571577

[11] S. Prahl, *Laserbeamsize: Iso 11146 calculation of laser beam center, diameter, and m^2*, version 2.5.0, 2026. DOI: 10.5281/zenodo.8346799 [Online]. Available: https://github.com/scottprahl/laserbeamsize

[12] International Organization for Standardization, "Lasers and laser-related equipment — test methods for laser beam widths, divergence angles and beam propagation ratios — part 1: Stigmatic and simple astigmatic beams," ISO, Geneva, Switzerland, Standard ISO 11146-1:2021, 2021. [Online]. Available: https://www.iso.org/standard/77769.html

[13] C. Boucher, *Phase space distributions and emittance in 2d charged particle beams*, https://www.comsol.com/blogs/phase-space-distributions-and-emittance-in-2d-charged-particle-beams, Accessed: 2026-08-19, COMSOL AB. Accessed: Aug. 19, 2026.

[14] E. J. N. Wilson, "Circulating beams," in *An Introduction to Particle Accelerators*, Oxford University Press, May 2001, ISBN: 9780198508298. DOI: 10.1093/acprof:oso/9780198508298.003.0004 eprint: https://academic.oup.com/book/0/chapter/160646543/chapter-ag-pdf/58291207/book_11694_section_160646543.ag.pdf. [Online]. Available: https://doi.org/10.1093/acprof:oso/9780198508298.003.0004

[15] H. Kragh, "The lorenz-lorentz formula: Origin and early history," *Substantia*, vol. 2, no. 2, pp. 7–18, Sep. 2018. DOI: 10.13128/Substantia-56 [Online]. Available: https://riviste.fupress.net/index.php/subs/article/view/56

[16] F. Brandi and L. A. Gizzi, "Optical diagnostics for density measurement in high-quality laser-plasma electron accelerators," *High Power Laser Science and Engineering*, vol. 7, e26, 2019. DOI: 10.1017/hpl.2019.11

[17] M. N. Polyanskiy, "Refractiveindex.info database of optical constants," in *Nature Scientific Data*, ser. Nature Scientific Data, vol. 11, Springer, Jan. 2024, p. 94. DOI: 10.1038/s41597-023-02898-2

[18] NIST, *Experimental Polarizabilities (release 22)*, [Online]. Available: https://cccbdb.nist.gov/pollistx.asp, Gaithersburg, MD, 2022.

[19] D. Malacara-Hernández and D. Malacara-Doblado, "Chapter two - optical testing and interferometry," in ser. Progress in Optics, T. D. Visser, Ed., vol. 62, Elsevier, 2017, pp. 73–156. DOI: https://doi.org/10.1016/bs.po.2016.12.001 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0079663816300178

[20] V. Janota, “Optical probing of laser-plasma accelerators,” Master’s thesis, M.S. thesis, Czech Technical University in Prague, Jun. 2026. [Online]. Available: https://hdl.handle.net/10467/179788

X. Acknowledgements

I would like to acknowledge the support of my supervisor, Dr. Amina Hussein, for the opportunity to work in her lab and her encouragement throughout the project. I would also like to thank Dr. Nicholas Beier and Abdulhakeem Ojonoka Yusuf for their mentorship. They offered significant advice and constructive criticism when I encountered challenges or presented my work, enabling me to grow as a student and researcher. I would lastly like to acknowledge the support of the UAlberta Engineering Co-op Office and the Government of Canada NSERC/CRSNG USRA grant for making this work term possible.

XI. Appendix

A. A Guide to 2D Meshing in Ansys Fluent

In ideality, it is the design geometry and collection of setup parameters that dictate the output of our simulation. A representative and converged simulation is mesh-agnostic, meaning tweaking the resolution or arrangement of cells does not change the results. However, a good mesh can be the difference between a high-fidelity simulation and a simulation that fails to converge at all.

In our work, we used 2D Axisymmetric simulations to model the flow through our cylindrically symmetric nozzles. In this workflow, the nozzle contour is designed and dimensioned in SolidWorks, where it is then imported into Ansys Workbench and partitioned in DesignModeler for later meshing work. The finalized sketch is then meshed using Ansys Fluent Meshing.

The sketch is defined on the front plane as a radial profile slice in SolidWorks according to the specifications of the simulation, seen in Figure 43. In our case, we include an inlet shaft to emulate the input gas line and stabilize incoming flow, as well as an exhaust region large enough to capture the majority of the exhaust plume for analysis. Before importing into

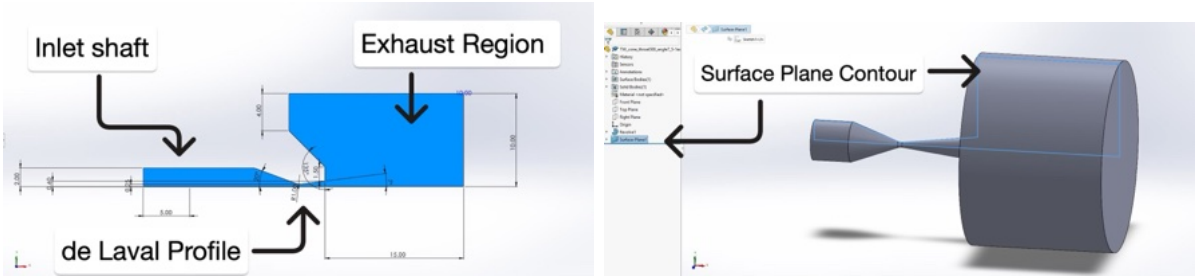

Fig. 43. (left) An example view of the 2D radial profile sketch of a nozzle in SolidWorks. From left to right, the sketch features an inlet shaft, the converging-diverging geometry of the de Laval nozzle, and a large exhaust buffer region. (right) A view of the surface plane selected in SolidWorks, highlighted against the 3D revolve of the 2D profile. This part is then saved as an `.x_t` for importing into Ansys.

Ansys, the sketch is revolved around its axis of symmetry into a solid body. Additionally, a 'Surface Plane' is added by selecting Insert → Surface → Planar and bounding it using the 2D profile sketch, which allows Ansys to recognize it when imported into DesignModeler. This is seen in Figure 43. The file is exported as an `.x_t` parasolid model part file, which is one of the possible file types that Ansys recognizes on import.

We then open Ansys Workbench 2026 R1 and create a 'Fluid Flow (Fluent)' project. This will create a pop-up menu showing the workflow required to complete a simulation, seen in Figure 44. Right-clicking on a menu item and selecting 'Properties' will open up configuration settings for the item. We use this to change 'Analysis Type' to '2D' before importing the `.x_t` file and selecting 'Edit Geometry in DesignModeler'.

We will use DesignModeler to partition our sketch into several roughly-quadrilateral sections, which will allow us to create an adaptive mesh. First, click 'Generate' and suppress the solid body generated, leaving only the facial plane we are interested in. Select the surface and click 'Sketching'. Use the line tool to partition the surface, and the dimension and/or constraint tools to fully define the sketch. Aim to divide the surface into quadrilaterals such that each subdivision will require a similar mesh resolution to effectively capture the fluid dynamics within. See Figure 45 for an example. For example, the converging-diverging section is divided into three subdivisions:

- The converging section, which will transition from the coarse mesh in the inlet shaft to the throat;

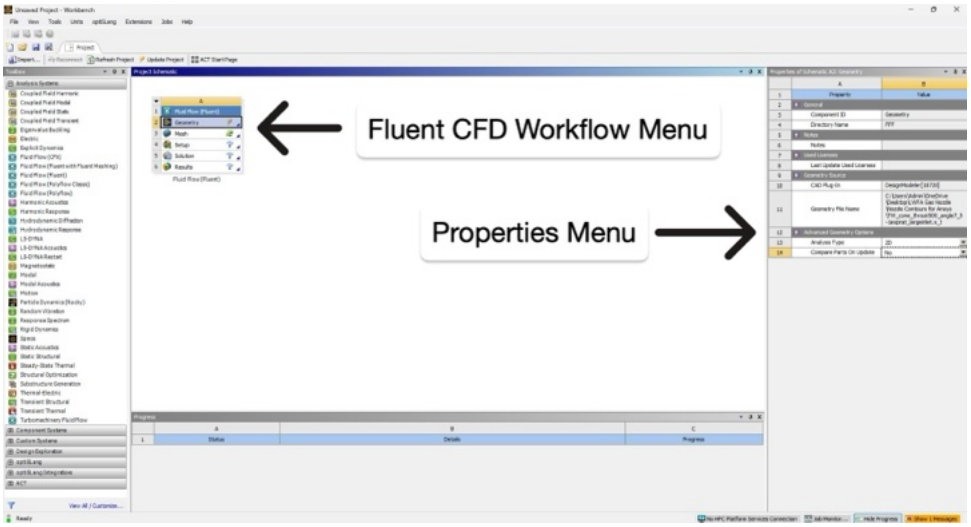

Fig. 44. Screenshot of the Workbench view of a Fluent workflow, used for configuring the mesh and case files.

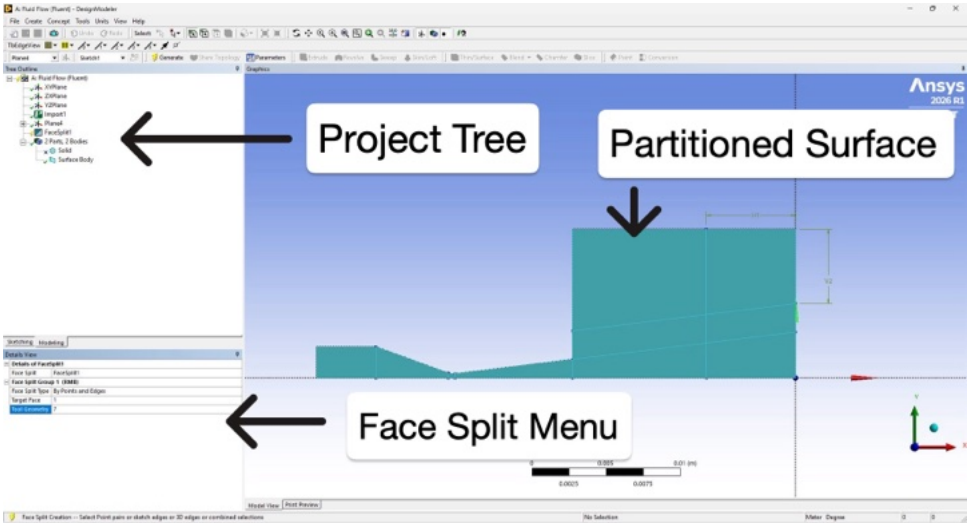

Fig. 45. Example of a partitioned surface in DesignModeler, right before generating the Face Split feature. Constrained and dimensioned lines are used to split the face, and are each added to the target geometry of the Face Split tool.

- The throat itself, which will have a very high mesh resolution for the choked flow;
- The diverging section, which will maintain this high meshing resolution to capture the expansion and boundary layer phenomena critical to shaping the exhaust plume parameters.

Once completed, use the ‘Face Split’ tool to actually split the surface into the newly-defined pieces, and exit.

From the Workbench workflow menu, we can then launch the meshing tool. First, use the ‘Create Named Selection’ tool (with shortcut ‘n’ when selecting desired geometry) to select and label the ‘Inlet’, ‘Outlet’, ‘Axis’, ‘Fluid Flow’, and ‘Wall’; each of which are important sections whose properties will be configured in Fluent. A screenshot from fluent is shown in Figure 46. This helps Fluent auto-detect the types of boundary conditions available to the relevant parts of the geometry. Note that Ansys’ point, edge, or face selection filters can be quite helpful to facilitate clicking sections! They are found along the top menu bar as cube icons with a green element representing what they filter for.

Now, we use the Insert → Sizing tool to impose division constraints on each edge in our

Fig. 46. An image of our surface for meshing with some of the named selections highlighted. Notice how multiple edges are all selected for the ‘Axis’ named selection, as they each make up the axis of symmetry.

surface. This will let us enforce our custom meshing rules to produce a regular and tunable grid mesh. First, we will define all the vertical edges, working outwards from the axis. In a

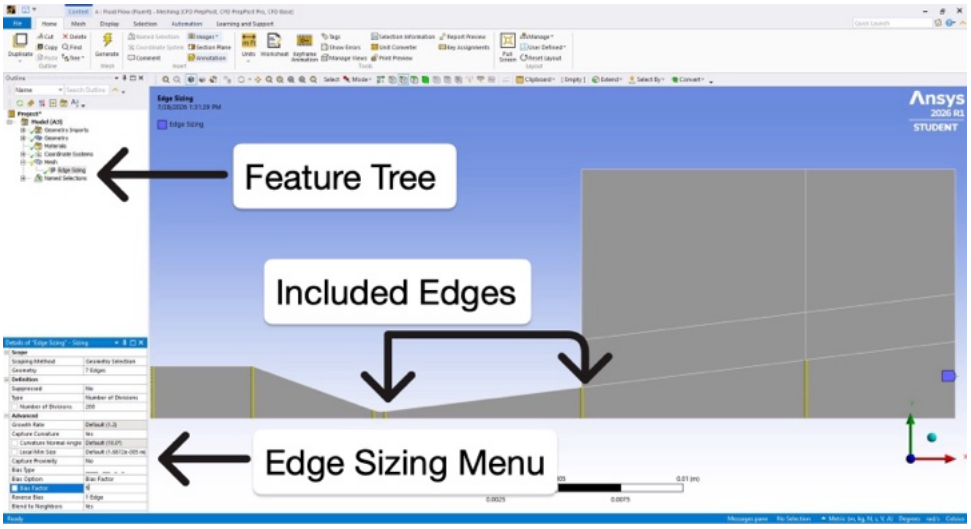

Fig. 47. A screenshot from Mesher while applying an edge size constraint to all vertical edges in the ‘first horizontal strip’. The selected edges are highlighted in yellow; two of such edges are indicated by the label. Each edge must share the same number of divisions, and in this case will share the same bias direction and factor used to control spacing and scaling.

given ‘horizontal strip’ of the surface sharing the same upper and lower boundaries, we want to ensure there is a set number of vertical divisions to prevent edge dislocations in our mesh. Therefore, each vertical edge in this strip must be configured to have the same ‘Number of divisions’, but the gradient and sizing of these N cells can change from edge to edge. For example, all vertical lines directly touching the axis can be grouped into one horizontal strip with N vertical subdivisions, shown in Figure 47. However, each edge can be individually specified to bias the size of the cells along the line depending on the resolution requirements. Repeat this process horizontally, dividing the mesh into vertical strips which will each share the same N subdivisions, but distribute them according to need.

Finally, use the ‘Face Meshing’ tool to enforce a quadrilateral mesh before clicking ‘Generate’. A well-defined grid mesh will take just a few seconds to generate. However,

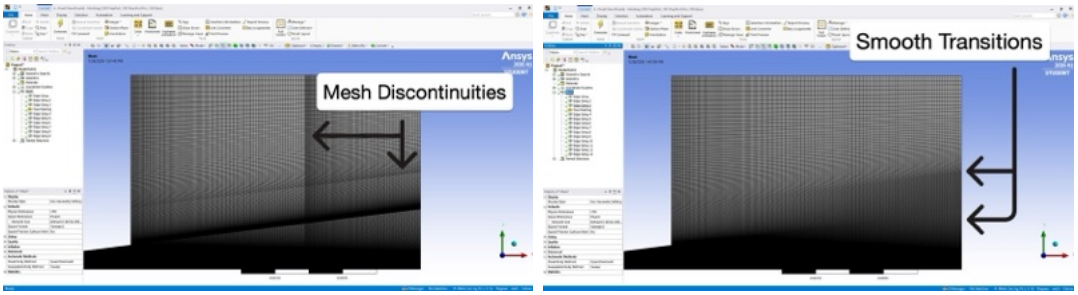

Fig. 48. A comparison between a poorly scaled mesh (left) and a tuned, smooth mesh (right) after iteratively adjusting the bias factor for the different edge sizing features. Notice how transitions between quadrilateral partitions are scarcely noticeable in the smooth mesh. If transition discontinuities are visible, the numerical solver will 'see them' too.

creating a smooth mesh is an iterative process, with before and after results shown in Figure 48. The user then must inspect the boundaries between each quadrilateral subdivision to ensure the mesh sizing and distribution flows continuously and smoothly from one to the next. Ideally, it should not be possible to tell where one subdivision ends and the next begins, which is important to avoid discontinuities that affect the numerical solver.

When completed, use the mesh diagnostics to evaluate the number of cells and their orthogonal quality. The meshes used in simulation typically had around $3 \cdot 10^5$ cells, with orthogonal quality $\sim 99.5\%$. Save and close the mesh, before ensuring that all completed elements of the Workbench workflow menu are updated. The project can now be saved, and the case can be set up in Fluent and exported to an HPC for numerical solving.

B. Configuring an Ansys Fluent Project for HPC Batch Submission

First, log onto the HPC and set up a project folder for adding the pre-initialized mesh file.

To set up the case file, we need GUI access to Ansys. We can navigate to JupyterHub, log into Fir, and request an interactive job. Ensure at least $16000MB$ of memory is requested, as Ansys consumes a lot of RAM. Navigate to `Modules` and load `ansys/2026R1`. Launch `Fluent (VNC)` from the home page, and open with 'Double Precision' enabled and using $n-1$ cores, where n is the number of cores reserved from the interactive job request menu. Change the working directory to the project folder and start Fluent.

We can open the `.msh` file using File $\rightarrow$ Read $\rightarrow$ Mesh..., and proceed with our verification and setup. If Fluent crashes while loading case or mesh files, ensure enough memory is requested for the interactive job. To verify the file is loaded properly, select General $\rightarrow$ Check, and follow any instructions in the terminal. Ensure that all listed volumes and areas are positive. Often, floating point errors can result in the mesh stretching below the axis of symmetry, which may require:

```
/mesh/repair-improve/repair
```

And re-checking of the case. Change the solver to 'Axisymmetric' and 'Density-Based'. Next, we can select General $\rightarrow$ Scale to change our default length units to mm. Similarly, in General $\rightarrow$ Units $\rightarrow$ Density, we can configure a custom unit 'e20particles/cm-3' with conversion factor indicating the conversion *from the custom unit to* $kg \cdot m^{-3}$. For example, for Argon gas, we find a conversion factor from particles $\cdot cm^{-3}$ to $kg \cdot m^{-3}$ as:

$$N\frac{\text{particles}}{cm^3} \cdot \frac{1\,\text{mol}}{6.0221408 \cdot 10^{23}\text{particles}} \cdot \frac{39.948g}{1\,\text{mol}} \cdot \frac{1\,\text{kg}}{1000\,\text{g}} \frac{10^6 cm^3}{1m^3} = N \cdot 6.634 \cdot 10^{-20} \frac{kg}{m^3}.$$

For helium, we use $0.6646 \cdot 10^{-20}$. Note that since our units are in 'e20particles', we can drop the exponential factor.

Next, enable Models → Energy → Energy Equation. In Materials → Fluid, add a new fluid from the Fluent Database, for example, argon, by copying parameters with the 'Copy' button, changing the density to 'Ideal Gas' and saving with 'Change/Create'. Add this material to Cell Zone Conditions → Fluid → fluid flow. Under Boundary Conditions, configure:

- Inlet as a 'Pressure Inlet', with 'Gauge Total Pressure' set to a 'New Input Parameter' created in the drop-down. Set 'Supersonic/Initial Gauge Pressure' to slightly below this value using the expression

  $$\text{backing\_pressure} - 1000[Pa].$$

  For whatever the input parameter is called in place of `backing_pressure`. Change the Turbulence Specification Method to 'Intensity and Hydraulic Diameter', with 'Hydraulic Diameter' set as the smallest geometry found in the case file, in this case the throat radius.
- Outlet to a 'Pressure Outlet', with the same Turbulence configuration as for the Inlet.

Adjust Boundary Conditions → Operating Conditions…to be $0\,\mathrm{Pa}$ to make all gauge pressures absolute.

To adjust our end condition, change Solution → Monitors → Residual to have absolute criteria of $1e-05$. To monitor convergence, Solution → Report Definitions → New → Flux Report → Mass Flow Rate…, create MFR monitors for the Inlet, the Outlet, and both the Inlet/Outlet. Ensure each is named, with 'Report Plot' and 'Create Output Parameter' checked and 'Report File' unchecked. Then, under Solution → Monitors → Report Plots, configure each plot's axes to add major gridlines and adjust the precision on the y-axis to 9 decimal places. We will use these monitors to gauge convergence via the decreasing difference between Inlet and Outlet mass flow rate.

Once the case file is prepared, it can be saved and exported as a case file with extension `.cas.h5`. If the case was set up externally, this file can be placed in the user's project directory on Fir, for example:

```
rsync -avP ~/Downloads/ansys_case.cas.h5
↪  user@fir.alliancecan.ca:/project/def-group/user/Ansys/project_name
```

Where relevant paths should be changed to copy files correctly. Submitting the case file to SLURM requires two additional files:

1) The `.sh` job submission file, as usual – examples can be found at https://docs.alliancecan.ca/wiki/Ansys;
2) A `.jou` file, which contains the commands to load the case file, run the solver, and save the results.

The journal file deserves a bit more attention, as the commands it uses are specific to Ansys Fluent's TUI. Search for 'Ansys Fluent TUI Reference' to find a complete and up-to-date list. While the CCDB's Ansys page also has reference `.jou` files, not included is how to change the value of an input parameter. It requires two simple commands:

```
/define/parameters/enable-in-TUI yes
/define/parameters/input-parameters/edit backing_pressure, 500000
```

The first of which enables TUI control of input parameters, and the second sets the parameter to a comma-separated value (e.g. sets the parameter named `backing_pressure` to a value of $5e5$).

Once the various options are configured in the `.sh` and `.jou` files, e.g. input file and directory names, the job can be submitted with `sbatch` as usual.

Once complete, open the Case & Data `.dat.h5` file in Fluent and examine the output for consistency and convergence. View the input and output parameters in Parameters &

Customization → Parameters to print them to the terminal. If necessary, save the file in an HDF5-compliant format via File → Export… → Solution Data, selecting fluid flow and the required information, e.g. Density.

C. Config File: Ansys .sh and .jou Example

Here is a sample `.jou` file for submitting an Ansys Fluent job to the Fir HPC. Note that the line `/file/read-case FFF` tells Fluent to look for a file `FFF.cas.h5` in the job directory as the input case file.

```
; SAMPLE FLUENT JOURNAL FILE - STEADY SIMULATION
; ----------------------------------------------
; lines beginning with a semicolon are comments

; Overwrite files by default
/file/confirm-overwrite no

; Preferentially read/write files in legacy format
/file/cff-files yes

; Read input case and data files
/file/read-case FFF

; Change the input parameter
/define/parameters/enable-in-TUI yes
/define/parameters/input-parameters/edit backing_pressure, 500000

; Run the solver for this many iterations
/solve/iterate 7500

; Overwrite output files by default
/file/confirm-overwrite n

; Write final output data file
/file/write-case-data sample_output_data_file

; Cleanly shutdown fluent
/exit
```

Here is a sample `.sh` file for submitting an Ansys Fluent job to the Fir HPC.

```
#!/bin/bash

#SBATCH --account=def-group   # Specify account name
#SBATCH --time=00-00:10       # Specify time limit dd-hh:mm
#SBATCH --nodes=1             # Specify number of compute nodes (narval 1
↪  node max)
#SBATCH --ntasks-per-node=48  # Specify upto maximum number of cores per
↪  compute node
#SBATCH --mem=0               # Specify memory per compute node (0
↪  allocates all memory)
#SBATCH --cpus-per-task=1     # Do not change

module load StdEnv/2023       # Do not change
module load ansys/2026R1      # or newer versions

MYJOURNALFILE=sample_ansys.jou      # Specify your journal file name
MYVERSION=2d                  # Specify 2d, 2ddp, 3d or 3ddp
```

```
# ------- do not change any lines below --------

if [[ "$CC_CLUSTER" == narval ]]; then
 module load intel/2023 intelmpi
 export INTELMPI_ROOT=$I_MPI_ROOT
 unset I_MPI_ROOT
fi

if [[ ("${EBVERSIONANSYS//R*}" -ge 2025 && "${CC_CLUSTER}" == nibi) ||
↪  "${CC_CLUSTER}" == narval ]]; then
 export I_MPI_HYDRA_BOOTSTRAP=ssh
 unset I_MPI_HYDRA_BOOTSTRAP_EXEC_EXTRA_ARGS
fi

slurm_hl2hl.py --format ANSYS-FLUENT > /tmp/machinefile-$SLURM_JOB_ID
NCORES=$SLURM_NTASKS

if [ "$SLURM_NNODES" == 1 ]; then
 fluent -g $MYVERSION -t $NCORES -mpi=intel -pshmem -i $MYJOURNALFILE
else
 if [[ "${CC_CLUSTER}" == nibi ]]; then
   fluent -g $MYVERSION -t $NCORES -mpi=intel -peth
   ↪  -cnf=/tmp/machinefile-$SLURM_JOB_ID -i $MYJOURNALFILE
 else
   fluent -g $MYVERSION -t $NCORES -mpi=intel -pib
   ↪  -cnf=/tmp/machinefile-$SLURM_JOB_ID -i $MYJOURNALFILE
 fi
fi
```

D. Config File: warpx.profile

```
#required dependencies for warpx
module load gcc/12.3 openmpi/4.1.5 cuda/12.6 hdf5-mpi/1.14.2 mpi4py/4.0.3
↪  paraview/6.0.0

#optional: imkl contains BLAS and LAPACK
module load flexiblas

# optional: for QED support with detailed tables
module load boost

# compiler environment hints
export CXX=$(which g++)
export CC=$(which gcc)
export FC=$(which gfortran)
export CUDACXX=$(which nvcc)
export CUDAHOSTCXX=${CXX}

module load python/3.11.5

if [ -d "$HOME/.venvs/warpx_gpu" ]
then
 source $HOME/.venvs/warpx_gpu/bin/activate
fi

#Pre-installed libraries
```

```
export CMAKE_PREFIX_PATH=${HOME}/code/src/blaspp:$CMAKE_PREFIX_PATH
export LD_LIBRARY_PATH=${HOME}/code/src/blaspp/lib64:$LD_LIBRARY_PATH
export CMAKE_PREFIX_PATH=${HOME}/code/src/lapackpp:$CMAKE_PREFIX_PATH
export LD_LIBRARY_PATH=${HOME}/code/src/lapackpp/lib64:$LD_LIBRARY_PATH

# optimize CUDA compilation for H100
export AMREX_CUDA_ARCH=9.0
```

E. A Better Way to Host JupyterHub on Fir

Simulation outputs can result in massive datasets, making file transfer for external analysis impractical. As such, it is useful to run JupyterLab on Fir to have direct access to data files. There are two established methods to run JupyterLab on Fir for interactive code development and data analysis:

1) Use the JupyterHub browser portal to request resources and access the JupyterLab servers for live hosting;
2) Use the `jupyterlab.sh` script to emulate this process from a terminal as described in Section VI-B.

However, each approach has issues. The browser-based approach limits the number of cores to 8, which can limit computational capacity. This is less of an issue for JupyterLab, as any interactive data processing should be lightweight compared to a batch job. More significantly, however, is that it is very challenging to start such a session with a virtual environment activated, or profile configuration file sourced. For WarpX in particular, which requires first sourcing `warpx.profile`, this is problematic due to the extensive Python dependencies involved in the analysis. The script approach resolves both of these issues, but introduces a new one: in the case of a dropped SSH connection between the computer used to access the HPC and the HPC itself, SLURM may kill the interactive job and reclaim the requested resources. This stops the JupyterLab server running on Fir, requiring re-allocation of resources and launching of the script, both of which take time.

A workaround method is to use the script-based approach, but to run it via a batch job rather than interactive resources. This will keep the job running in the background regardless of connection issues, and still allows for sourcing of environments or configuration files. The sample job submission file demonstrates the commands required to do this.

```
#!/bin/bash
#SBATCH --job-name=jupyter_notebook
#SBATCH --account=def-group    # RAC account
#SBATCH --time=02:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=24G
#SBATCH --output=slurm-jupyter-%j.out
#SBATCH --error=slurm-jupyter-%j.err
#SBATCH --mail-type=begin,end,fail
#SBATCH --mail-user=user@email.ca

set -e

source warpx.profile # source from home directory, where job is run

unset XDG_RUNTIME_DIR
PORT=8890 # change if necessary, just keep note
```

```
# Print useful information for connecting
echo
echo "JupyterLab Slurm job: $SLURM_JOB_ID"
echo "Compute node: $(hostname -f)"
echo "Port: $PORT"
echo
echo "SSH tunnel from your local machine with:"
echo "ssh -L ${PORT}:$(hostname -f):${PORT} user@fir.alliancecan.ca"
echo
echo "Then open:"
echo "http://localhost:${PORT}"
echo

# Start JupyterLab
jupyter lab --ip="$(hostname -f)" --port="$PORT" --no-browser
```