Compressed Analysis of Electric Propulsion Simulations using Low Rank Tensor Networks

IEPC-2024-795

Presented at the 38th International Electric Propulsion Conference, Toulouse, France
June 23-28, 2024

Doruk Aksoy[*], Sruti Vutukury[†], Thomas A. Marks[‡], Joshua D. Eckels[§], and Alex A. Gorodetsky[¶]

This work presents an approach to mitigate high storage and computational costs associated with analysis of high-fidelity simulation data. Specifically, we demonstrate that low-rank tensor compression of field quantities from particle-based plasma simulation data of electric propulsion devices can yield significant benefits. Moreover, compression in low-rank tensor format enables doing typical analysis without decompression. Our test-case is a 3D simulation of an SPT-100 thruster operating in vacuum using AFRL's Thermophysics Universal Research Framework (TURF) code. We measure the TT-ICE method's performance on the simulation data based on compression ratio, compression time, and integrity of the reconstructed dataset to the original. For particle number densities stored in structured meshes, we achieve a compression ratio (CR) of $8.08\times$. For ion current density and electric field components, we achieve a CR of $6.23\times$ and $354,796\times$, respectively. For unstructured surface mesh data, we see near exact floating potential approximations and derived total current to surface quantities.

1. Introduction

Particle-in-cell methods are widely used to simulate electric propulsion (EP) devices such as Hall thrusters. Fully-kinetic particle-in-cell simulations have been used to gain insight into the micro-scale physics governing the macroscopic performance of these thrusters.[?,?] Hybrid methods, where the heavier species are treated as particles while the electrons are treated as a fluid, have been extensively used for many years for device-scale discharge plasma simulation,[?] performance prediction,[?] and plume simulation.[?]

Despite their utility, performing such simulations can be a computationally intensive procedure that generates large amounts of data, particularly if particle trajectory information is stored. The size of the generated simulation data makes it challenging to store and use the data in further analysis and downstream learning tasks. As the spatial and temporal resolution of these simulations grow, and the number of particles needed increase, even storing the data may become a serious bottleneck.

Tensor decompositions have been shown to be an effective approach to reduce the size of various types of data including scientific simulations.[?,?,?,?] In this approach, data is interpreted as a high-dimensional tensor, which can then be represented using one of the well known a low-rank tensor formats.[?,?,?] *Tensor train format*[?] (TT-format) is one of the well-studied formats shown to be suitable to compress scientific data.[?] Furthermore, introduction of efficient incremental algorithms to compute the TT-decomposition has made it possible to apply the TT-format to large-scale data such as PIC simulations.[?]

In addition to approximating the full tensor with much smaller linear algebra objects such as vectors, matrices, or tensors, the TT-format allows us to perform analyses in the reduced format without requiring reconstruction of the full tensor. Performing compressed analysis using low-rank tensor decompositions has demonstrated success for a large variety of scientific simulations and gathering increased attention in the literature.[?,?,?,?,?] With this work, we aim to extend the use of tensor network compressed analysis to

[*]PhD candidate, Department of Aerospace Engineering, University of Michigan
[†]Graduate Student, Department of Aerospace Engineering, University of Michigan
[‡]Postdoctoral Research Fellow, Department of Aerospace Engineering, University of Michigan
[§]PhD candidate, Department of Aerospace Engineering, University of Michigan
[¶]Associate Professor, Department of Aerospace Engineering, University of Michigan

particle-in-cell simulations of electric thrusters. Specifically, we propose to use tensor decompositions to reduce the size of the simulation data generated by EP simulations.

In this study, we simulate an electric propulsion device under on-orbit conditions using *Thermophysics Universal Research Framework* (TURF) software package, developed by the United States Air Force Research Laboratory (AFRL). The simulation outputs a structured volume mesh capturing both field quantities and particle number densities, as well as an unstructured triangular surface mesh capturing electric potential and current on the thruster surfaces. We compress the different mesh files individually and group the states within each mesh component according to their properties. Note that our objective is not to match the numerical results of the simulation[?] exactly, but to generate a typical and scalable dataset for a Hall thruster plume for compression analysis. This 3D plume simulation in TURF would require significantly more particles to replicate the resolution of Boyd's 2D axis-symmetric plume simulation, particularly in the backflow region of the computational domain.

Finally, we compare the analysis outputs generated from the original and compressed datasets to quantify the loss in simulation output fidelity from tensor network compression both qualitatively and quantitatively.

2. Approach

This section presents our approach to compressing and analyzing the simulation data generated by electric propulsion simulations. We first describe the simulation that generates the data of interest. We then describe our compression methodology. Finally, we describe the outputs that are compared between the original and compressed datasets. **??** shows a high-level overview of the proposed compressed analysis pipeline.

Figure 1: Proposed compressed analysis pipeline. High-fidelity simulation environments generate large-scale simulation data. This data can be compressed incrementally into a tensor-network format and then analyzed without decompression.

a. SPT-100 plume simulation test case

Our test case is a 3D hybrid-particle-in-cell (PIC) simulation of the plume of an SPT-100 Hall thruster performed in TURF.[?] The case is an extension that is based on the 2D axisymmetric model of Boyd et al.[?] The domain consists of a quarter of the plume, and extends from (-2, 0, 0) meters to (10, 10, 10) meters. The computational domain is partitioned into 8 subdomains, uniformly in each direction. The subdomains are evenly distributed to Message Passing Interface (MPI) processes to parallelize the simulation in TURF.[?] We discretize the larger computational domain into a structured grid of $120 \times 100 \times 100$ cells, giving us grid cells of 10 cm.

The simulation contains three species — Xenon atoms, singly-charged Xenon ions, and electrons. We treat both the atoms and ions as computational macroparticles, and model the electrons as an unmagnetized, isothermal, collisionless fluid. The virtual thruster exit plane is an annulus of inner diameter 60 mm and outer diameter of 100 mm centered at (0, 0, 0) meters, with the plume centerline aligned with the x-axis. The neutral atoms are injected with a number density of $10^{18}\,\mathrm{m}^{-3}$, face-normal velocity of $300\,\mathrm{m/s}$, and a temperature of $1000\,\mathrm{K}$. The ions are injected with a density of $10^{17}\,\mathrm{m}^{-3}$, a normal velocity of $17\,\mathrm{km/s}$, and a temperature of $4\,\mathrm{eV}$. We assume both species have an effective divergence angle of 15° and are injected with radially-uniform properties. The Boltzmann relation is used with an electron temperature of 3 eV, fixed over the computational domain.

We model the thruster surface as a solid plate of aluminum with dimensions $120\,\mathrm{mm}\times 120\,\mathrm{mm}\times 10\,\mathrm{mm}$. The potential at the surface is set to zero, and ions that collide with this surface neutralize and reflect diffusely. We remove particles that cross the domain boundaries from the simulation, and set the electric field normal to these boundaries to zero. **??** provides a sketch for the simulation domain as well as the positioning of the thruster exit plane.

TURF models charge-exchange (CEX) collisions between neutrals and ions using the direct simulation Monte Carlo (DSMC)[?,?] method, with a total cross section, σ_{cex}, of

$$\sigma_{cex} = \left(0.8423 \times 10^{-20}\right)\left[-23.30 \log_{10}(v_{\mathrm{rel}}) + 142.21\right] \quad \mathrm{m}^2, \tag{1}$$

where v_{rel} is the relative velocity of the colliding particles in m/s. The code handles elastic momentum-exchange (MEX) collisions with the Monte Carlo collision (MCC) method assuming a background neutral temperature of 1000 K and a total collision cross-section, σ_{mex}, of

$$\sigma_{mex}(\mathrm{Xe}, \mathrm{Xe}^{+}) = \frac{6.42 \times 10^{-16}}{v_{\mathrm{rel}}} \quad \mathrm{m}^2. \tag{2}$$

To compute the post-collision velocities, TURF employs a modified variable hard sphere (VHS) model and assumes isotropic scattering.

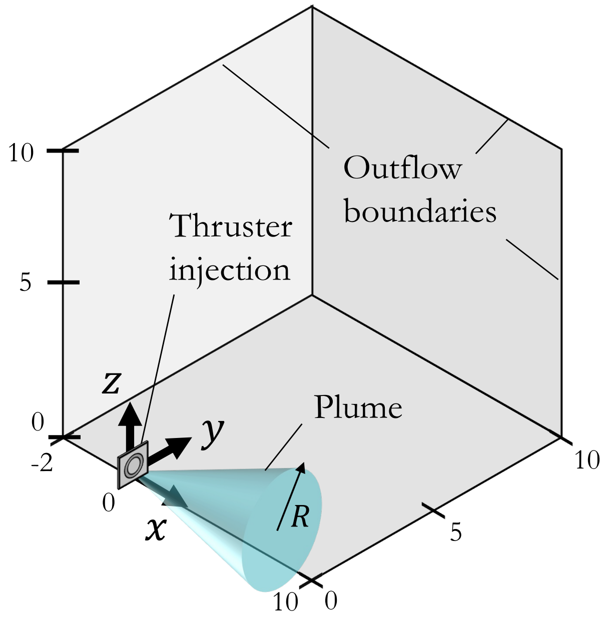

Figure 2: Visualisation of the simulation domain. The domain is [-2,10] meters for the x axis, and [0,10] meters for y and z axes. The thruster centerline is aligned with the x axis and the exit plane of the thruster is located at $x = 0$ m. Note that the thruster is not shown to scale for visualisation purposes.

Following Boyd,[?] the simulation proceeds in two steps: an initial warm-up phase where neutral atoms are injected until a steady state is reached, followed by a second phase where ions are introduced. The warm-up phase uses a timestep of $\Delta t = 5 \times 10^{-6}$ s and proceeds for 1500 iterations, corresponding to a simulation time of $t = 7.5$ ms. Once the neutral atoms reach steady state, ions are injected with a $\Delta t = 5 \times 10^{-6}$ s, continuing for an additional 10800 iterations until a simulation end time of $t_f = 61.5$ ms. A simulation with the grid spacing described above and 3.6 million physical particles takes about 17 hours using 16 CPU cores over 4 compute nodes, with Intel Xeon Gold 6154 processors and 25 GB RAM per core (100 GB memory in total).

The simulation has three types of outputs: 1) uniform structured cut-cell volume data, 2) uniform unstructured surface field data, and 3) trajectories of neutral atom and ion particles. Volume data is captured on a structured mesh and consists of particle number densities, ion current density, and electric field quantities.

Using the settings described above, the simulation generates a structured mesh file (.vts) of size 337 MB for each saved iteration for volume data. Surface data is captured on an unstructured triangular mesh and consists of floating potential and total current to the thruster surface from the plasma. Using the settings described above, the simulation generates an unstructured mesh file (.vtu) of size 228 KB for each saved iteration for surface data. At each saved iteration the simulation generates a trajectory file in the form of a geometric structure data object (.vtp) with size 3.64 GB. The .vtp files consist of points, lines, and/or polygons and holds the positions and velocities of each computational neutral and ion particle in the simulation at a particular time step. An area of future effort is compression of such polydata files.

For a typical plasma simulation such as this one, storing all trajectory files and further utilizing them for analysis becomes quickly intractable. Luckily, one would not need to save the entire simulation dataset to do fully-kinetic analyses such as anomalous electron transport and closures of kinetic plume models. Adopting incremental tensor decomposition algorithms would allow efficient compression to be done on simulation iteration files as they are generated.

In this study, we focus on capturing the steady state behavior of the particles in the simulated system and therefore discard the snapshot files corresponding to the first 5000 iterations. These 5000 iterations encompass neutral atoms reaching steady state and transient behavior after initial ion injection. Between iterations $5000-12300$, we save the structured volume mesh at every 50th iteration. A subset of 1000 iterations, corresponding to iterations $11300-12300$ where plasma quantities are in steady state, are reconstructed and used for structured mesh analysis presented in **??**.

For our compression analysis, we consider a surface dataset corresponding to a simulation with a slightly modified setting to the one presented above for the volume mesh compression. A larger surface mesh with an area of $0.14\,\mathrm{m}^2$ is used to provide a substantial dataset for compression demonstrative purposes. Further, the neutrals atoms and ions are injected at the same time and the simulation time step is reduced to $\Delta t = 1\times10^{-7}$ to see sufficient transient behavior. We compress and reconstruct snapshots for iterations 0-15000, where every 100 iterations are saved. These reconstructions are used for unstructured surface mesh analysis presented in **??**.

b. Data compression pipeline

This section presents the data compression pipeline for simulation data generated by TURF. We first provide a general overview to tensor decompositions and describe the tensor decomposition format we prefer to compress the simulation outputs. We then describe the normalization methods we employ to preprocess the data for increased compression performance.

i. Tensor decompositions

The simulation data can be viewed as large arrays of numbers, or tensors, and our compression approach seeks to find low-rank structure of these high-dimensional arrays. Specifically, we seek to construct low-rank *approximations* of the results. Unlike the matrix case, there exists no unique optimal tensor-decomposition approach. Instead, there exist a variety of formats, such as CANDECOMP[?]/PARAFAC[?] (CP), Tucker,[?] and tensor train[?] (TT), which have different advantages and exploit different types of structure. In this paper, we employ the TT-format since its storage grows linear to the number of dimensions d in contrast to Tucker format's exponential growth and it does not rely on an accurate a-priori guess of the tensor rank for accurate approximations like the CP format, which is an NP-complete problem.[?,?,?] We refer the curious reader to review papers on tensor decompositions for further details.[?,?]

The TT-format represents a d-dimensional tensor $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ as the tensor product of d 3-dimensional tensors $\mathcal{G}_i \in \mathbb{R}^{r_{i-1}\times n_i\times r_i}$. Specifically, $\mathcal{X}$ is said to be in TT-format if it is represented as

$$\mathcal{X} \approx \mathcal{G}_1 \times \cdots \times \mathcal{G}_d, \tag{3}$$

where the $\mathcal{G}_i$ are called TT-cores and their internal dimensions r_i are called TT-ranks and enforce $r_0 = r_d = 1$. Note that the TT-ranks depend on both the complexity of the data represented and the relationship between the neighboring dimensions. Therefore, the TT-ranks are heavily influenced by the order and the magnitude of the dimensions. To this end, we can leverage reshaping the original arrays into new shapes to exploit more structure within the data.

The TT-format allows us to reconstruct an entry/slice of the approximated tensor $\mathcal{X}$ without reconstructing the full tensor. Additionally, TT-format enables other analyses and operations such as integration, addition, differentiation etc. in the compressed state. As an example of entry retrieval, the entry $\mathcal{X}(i_1,\ldots,i_d)$ can be computed as

$$\mathcal{X}(i_1,\ldots,i_{d-1}) = \sum_{\alpha_1,\ldots,\alpha_d}^{r_1,\ldots,r_{d-1}} \mathcal{G}_1[:,\alpha_1]\mathcal{G}_2[\alpha_1,i_1,\alpha_2]\cdots\mathcal{G}_{d-2}[\alpha_{d-2},i_{d-1},\alpha_{d-1}]\mathcal{G}_d[\alpha_{d-1},i_d]. \tag{4}$$

Note that when the snapshots are accumulated along the last dimension, the first $d-1$ TT-cores will be same for all snapshots and only the last TT-core will be unique to a snapshot. Therefore, the slices of the last TT-core $\mathcal{G}_d$ can also be used as a latent representation for individual snapshots.

The TT-SVD algorithm[?] provides a way to compute the TT-representation of a d-dimensional array using a series of error-truncated singular value decompositions (SVDs). However, since the simulation snapshots arrive over time in a streaming fashion, one-shot algorithms like the TT-SVD algorithm may not suitable. If one waits for all simulation snapshots to be accumulated, then this limits the maximum size of data that can be represented. Instead, we employ an incremental alternatives to TT-SVD.[?, ?, ?, ?] From the existing alternatives, we use the TT-ICE* algorithm[?] due to its low memory footprint and high computational speed while compressing scientific simulations.[a] For the compression experiments in this work, we reshape the mesh data arrays into shapes with size $\mathcal{O}(10)$ to obtain a balanced growth in TT-ranks.[?, ?] In addition to the reshaping, we create an additional dimension and increment along that last dimension using the TT-ICE* algorithm. Finally, this algorithm takes a relative approximation tolerance ε and guarantees relative compression accuracy in a Frobenius norm sense for the entire duration of the stream, even for infinite streams.

ii. Data normalization

Due to the nature of the simulations, there are high magnitude differences across states and spatial domain. These values can range 18 orders of magnitude, from 10^0 to 10^{18}. Since TT-ICE* performs a sequence of error-truncated SVDs to expand the basis vectors stored in each TT-core, the algorithm favors the accuracy of representation at locations with higher number densities and allows errors in the sparsely populated regions. This becomes an issue especially when modelling the backflow region.

To address this issue, we employ different normalization methods such as z-score, logarithmic, square root, and unit vector normalization. Z-score normalization scales the data to have a mean of zero and a standard deviation of one. With μ_x,σ_x being the mean and standard deviation of the data $x \in \mathbb{R}^k$, z-score normalization computes the normalized data $x_{norm} \in \mathbb{R}^k$ as

$$x_{norm} = \frac{x - \mu_x}{\sigma_x}. \tag{5}$$

Another normalization method we adopted is the unit vector normalization. Given the same x as above, unit vector normalization scales x to have a magnitude of one such that

$$x_{norm} = \frac{x}{\|x\|_F}, \tag{6}$$

where $\|x\|_F$ is the Frobenius norm of x.

In addition to (**??**) and (**??**), we also experimented with nonlinear functions to normalize the states. Logarithmic scaling with base e ($x_{norm} = \log(x)$), logarithmic scaling with base 10 ($x_{norm} = \log_{10}(x)$), and square root scaling ($x_{norm} = \sqrt{x}$) are some example nonlinear functions we included in this work.

Note that even when states for different quantities of interest (QoI) are compressed together, the normalization methods are applied to each state separately. In some cases, we also experimented using a combination of these normalization methods. This allowed us to first reduce the spread of values and then perform a statistically significant normalization. When cascading 2 normalization methods, we first apply the normalization methods in the order of mention. That is, in case of unit vector-logarithmic normalization, the data is first processed through unit vector normalization and then the outputs of that normalization are further processed using logarithmic normalization.

To summarize, we preprocess the simulation data prior to compression using the TT-ICE* algorithm. We provide up to two normalization methods and a target relative error as an input to the compression pipeline. As output, the pipeline returns normalization constants (for each state of each time snapshot) and the TT-cores.

3. Results

This section presents the results we obtained from compressing TURF simulation data using the TT-ICE* algorithm. Specifically, we compare reconstruction of different analysis using the compressed and uncompressed formats to determine if their is significant loss of information. We first discuss the metrics we use

[a]Of the heuristics available to accelerate TT-ICE, we only use the "occupancy" and "skip" heuristics of TT-ICE*, which avoid computing an SVD if a given TT-core is already full rank and skip the update of a TT-core if the presented data can already be represented within the desired target error using the existing set of TT-cores.

to compare the performance of the TT-decomposition briefly and present the results of the compression experiments. We conducted all compression experiments on a single compute node, with 24 Intel Xeon Gold 8468 processors and 24 GB RAM.

a. Metrics

To evaluate the performance of the TT-decomposition, we use the following metrics: 1) size on disk, 2) compression ratio and 3) compression time.

Size on disk refers to the size of the compressed dataset stored on disk. In addition to the TT-cores, this also includes the normalization constants and other metadata required to reconstruct the original dataset. Compression ratio (CR) is simply the ratio of the number of elements in the original dataset to the number of elements in the compressed dataset (i.e., TT-cores). Compression ratio is simply computed as

$$CR = \frac{\prod_{i=1}^{d} n_i}{\sum_{i=1}^{d} r_{i-1} n_i r_i}, \tag{7}$$

where r_i is the i-th TT-rank and n_i is the i-th dimension of the tensor. Note that CR only considers the states included in compression as an input to the compression pipeline. That means, if a subset of all mesh states are compressed, the numerator of CR will be adjusted accordingly.

Finally, compression time is the time taken to compute the TT-decomposition of the dataset using the `TT-ICE`* algorithm. Note that this time neither includes the time taken to normalize the data nor takes into account the time taken to convert the VTK files to NumPy arrays.

b. Compression experiments

We only consider snapshots from iterations that correspond to the steady-state portion of the simulation for compression. In particular, we consider the snapshots corresponding to iterations $5000 - 12300$ for compression. We group the states with similar units together. For example, number densities are compressed together, electric field potentials in x,y,z directions are compressed together, and the ion current is compressed on its own.

For qualitative analyses, we select a subset of the steady state simulation iterations for the compression analyses. In particular, we consider 20 snapshots, corresponding to the last 1000 iterations of the baseline simulation, where every 50 iterations are saved. We reconstruct snapshots corresponding to these iterations of interest using the relevant slices of the TT-cores and compare against the original snapshots from the dataset. In the main text we report the results with highest qualitative agreement and provide other combinations that might be of interest to the community in the appendices.

We report the following QoI using the considered snapshots for the volume data: 1) angular profiles of ion current density 2) contours of neutral atom and ion number density, and 3) electric field contours in the plume. We average those plasma quantities over the selected subset of 1000 iterations for volume and compare the the averaged original and averaged compressed datasets. We perform the following analyses for the surface data: 1) floating potential on the thruster surface and 2) total current to the thruster surface from the plasma. The trends of these quantities are compared between the original and compressed datasets for the full simulation. Since we want to capture the transient behavior on the thruster surface, we do not average the surface snapshots. Note that each field in the mesh files can be compressed individually or together with other fields. A separate compression pipeline for each state allows for greater flexibility in choosing input parameters to yield more accurate approximations. We report the normalization and target relative error ε settings that yield QoI with the greatest integrity to the original dataset; supplementary results are discussed in Appendix.

i. Volume data in structured mesh files

The states stored in the structured mesh files are the neutral atom number density, ion number density, ion current, and electric field values along the x, y and z directions in the computational domain across n_s state variables. We reshape this 5-dimensional $60 \times 50 \times 50 \times 8 \times n_s$ tensor into $6 \times 10 \times 10 \times 5 \times 10 \times 5 \times 8 \times n_s \times 1$ after normalizing each state of the snapshot individually. The results in this section compress the neutral and 3 ion species density together for $n_s = 4$. The ion current density is compressed by itself for $n_s = 1$.

The three components of the electric field are compressed together for $n_s = 3$. We investigate alternative compression groupings in the Appendix.

For **????** below, we compress the number densities, i.e., neutral and ion number density fields ($n_s = 4$). These figures depict the neutral atom and ion number density contours, respectively. The contours correspond to the quantities on the $X - R$ plane, where R describes the distance from the thruster centerline as shown in Figure **??**. The reconstructed dataset is an average of the reconstructions of iterations $11300 - 12300$ using square root normalization and a target relative error of $\varepsilon = 10^{-2}$. This compression setting results in a compression ratio of $8.075\times$. **??** present further details on compression performance including compression time, size on disk and TT-ranks.

The reconstructions in **????** accurately capture the near-field and far-field particle number densities and the backflow phenomena behind the thruster caused by CEX collisions with sufficient resolution, maintaining integrity of the data trends even after compression. The square root normalization allows for an accurate approximation of both the densely populated region near the thruster injection plane and the sparsely populated region behind the thruster. Further, a compression time of 135.6 seconds for the dataset is marginal considering potential substantial reduction in analytical computation time and storage savings afforded by the compressed format.

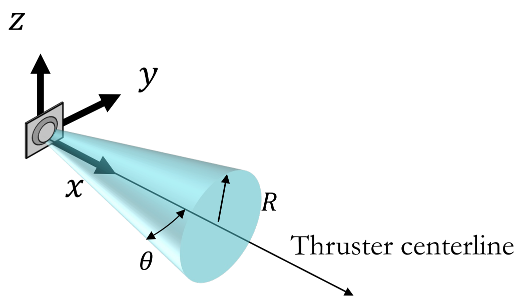

Figure 3: Visualisation of the plume in the simulation domain where $R = \sqrt{y^2 + z^2}$. θ is a coordinate to define distance from the thruster injection surface and is defined as $\theta = tan^{-1}(z/x) \approx tan^{-1}(y/x)$. The plume is approximately symmetric about the thruster centerline.

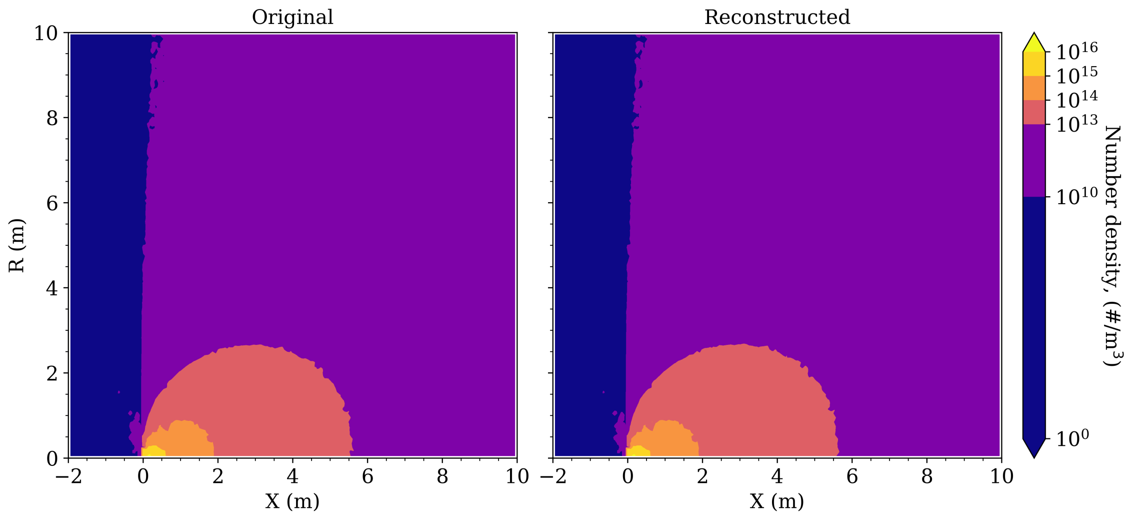

Figure 4: (Left) Neutral atom number density contours from the original dataset. (Right) Neutral atom number density from the reconstructed dataset using a square root normalization, and a target relative error $\varepsilon = 10^{-2}$. Neutral and ion number density fields are compressed together. The reconstructed dataset has a compression ratio of 8.08×. These comparisons show that the reconstructed dataset accurately captures near-field and far-field neutral number densities as well as the backflow phenomena behind the thruster caused by CEX collisions.

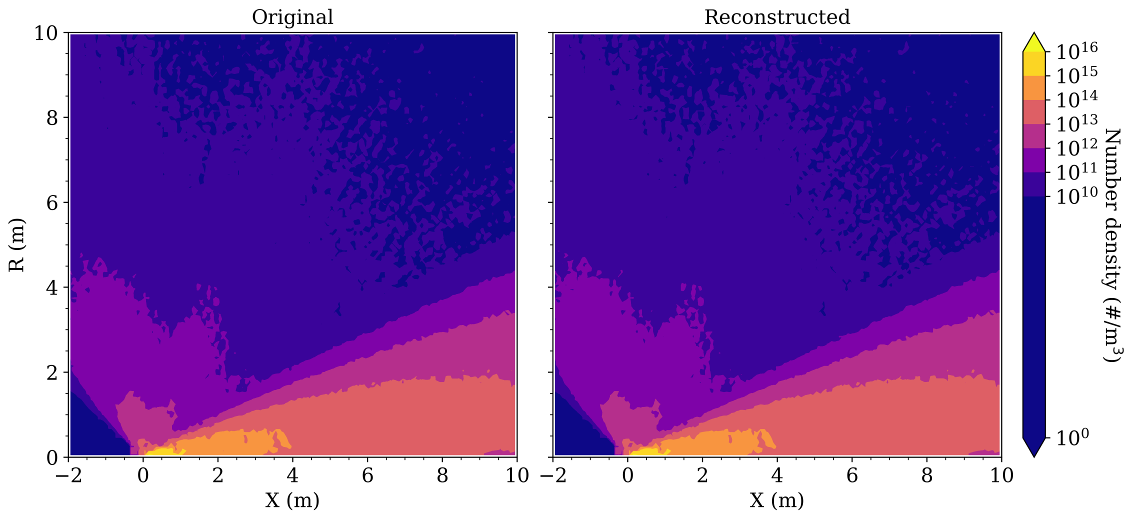

Figure 5: (Left) Ion number density contours from the original dataset. (Right) Ion number density contours from the reconstructed dataset using a square root normalization, and a target relative error $\varepsilon = 10^{-2}$. Neutral and ion number density fields are compressed together. As in the neutral number density contours, the reconstructed dataset captures the ion number density contours in the near-field, far-field, and CEX plasma with sufficient resolution.

Ion current densities in the plume from thruster injection are obtained directly from the computational ion particles in the TURF simulation. **??** shows a current density profile at 100 cm ± 12.5 cm from the thruster injection surface. The reconstructions presented are from different normalization combinations at a fixed target relative error of $\varepsilon = 10^{-4}$. **??** and **??** present further details on compression performance including compression time, size on disk and TT-ranks. The reconstructed current density profiles have little to no loss of accuracy close to the thruster centerline. Because at larger plume angles, i.e. further away from the thruster injection surface, there are orders of magnitudes fewer ion particles than in the near-field plume, quantities are prone to more statistical noise and are harder to capture accurately. At

Table 1: Compressed data metrics for neutral and ion number density states for a time averaged snapshot with 3.6 million particles. With a square root normalization combination and a target relative error of $\varepsilon = 10^{-2}$, the TT-decomposition results in states with a reduced size on disk by 8.076×.

	Size	Time (s)	ε	CR	TT-ranks	Latent size
Original	5.5 GB	-	-	-	-	-
Compressed	681 MB	135.6	10^{-2}	8.075	1, 6, 60, 586, 2829, 2590, 535, 71, 34, 1	34

large plume angles, the reconstructed dataset deviates from the original with a maximum relative error of 0.615, corresponding to the TT-compression done with a unit vector normalization.

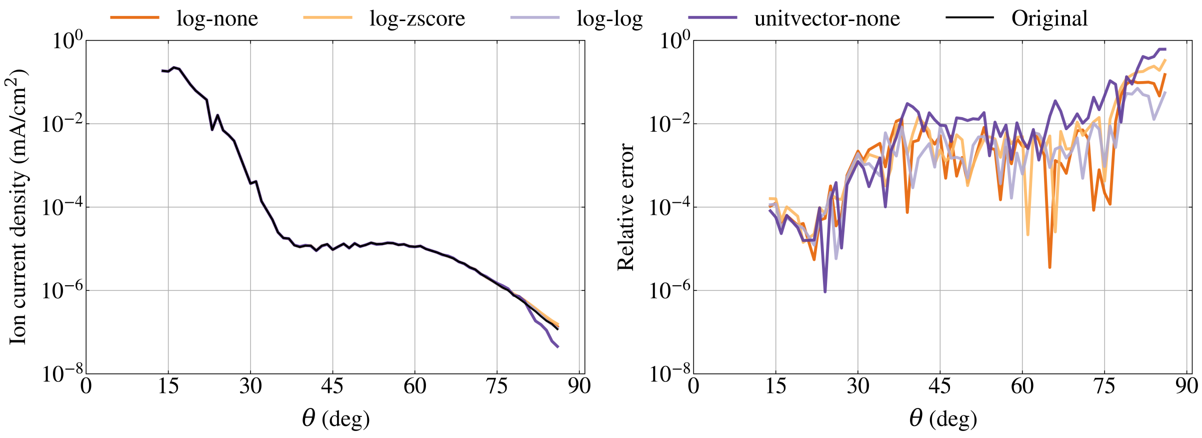

Figure 6: (Left) Angular profile of ion current density of the plume at 100 cm ±12.5 cm from the thruster injection plate. (Right) Relative error between the ion current density profiles between the original and reconstructed datasets with various normalization combinations and a fixed target relative error of $\varepsilon = 10^{-4}$. At higher angles, i.e. further away from the thruster injection surface, the reconstructed dataset deviates from the original with a max relative error of 0.615. A logarithmic-logarithmic normalization with target relative error $\varepsilon = 10^{-4}$ results in a compression ratio of 6.23× with relatively low relative error even at the higher plume angles. log: logarithmic normalization, zscore: z-score normalization, unitvector: unit vector normalization, none: no normalization.

Table 2: Compressed data metrics for the ion current density state for multiple normalization combinations. Compression of the ion current density is very sensitive to the normalization method selected. log: logarithmic normalization, zscore: z-score normalization, unitvector: unit vector normalization, none: no normalization.

	Size	Time (s)	ε	CR	TT-ranks	Latent size
Original	1.38 GB	-	-	-	-	-
log-none	157 MB	45.89	10^{-4}	8.76	1, 6, 51, 392, 1525, 1027, 265, 131, 131, 1	131
log-zscore	158 MB	46.82	10^{-4}	8.71	1, 6, 51, 392, 1527, 1033, 267, 132, 132, 1	132
log-log	221 MB	62.51	10^{-4}	6.23	1, 6, 53, 443, 1793, 1247, 304, 150, 150, 1	150
unitvector-none	17 MB	13.61	10^{-4}	81.9	1, 6, 41, 219, 636, 206, 70, 35, 35, 1	35

?? considers the the electric field components in the plasma close to the thruster injection plane, i.e. E_x, E_y, E_z ($n_s = 3$). CEX collisions, modeled by DSMC in TURF, result in slow ions that are sensitive to the electric fields in this region. These slow ions can be pulled behind the thruster creating the backflow plasma. We use z-score normalization and $\varepsilon = 10^{-1}$ target relative error for compression. **??**, present further details on compression performance including compression time, size on disk and TT-ranks. These inputs yield a very high $CR = 354,796\times$ with little to no loss in resolution and accuracy of the QoI. Further, a compression time of only 15.31 seconds for a dataset of size 4.2 GB can afford a significant reduction in analysis time. In the near-field plume during steady state, the ions are accelerated electrostatically; the electric fields are not time-varying. This is supported by the fact that electric field latent space size is 1.

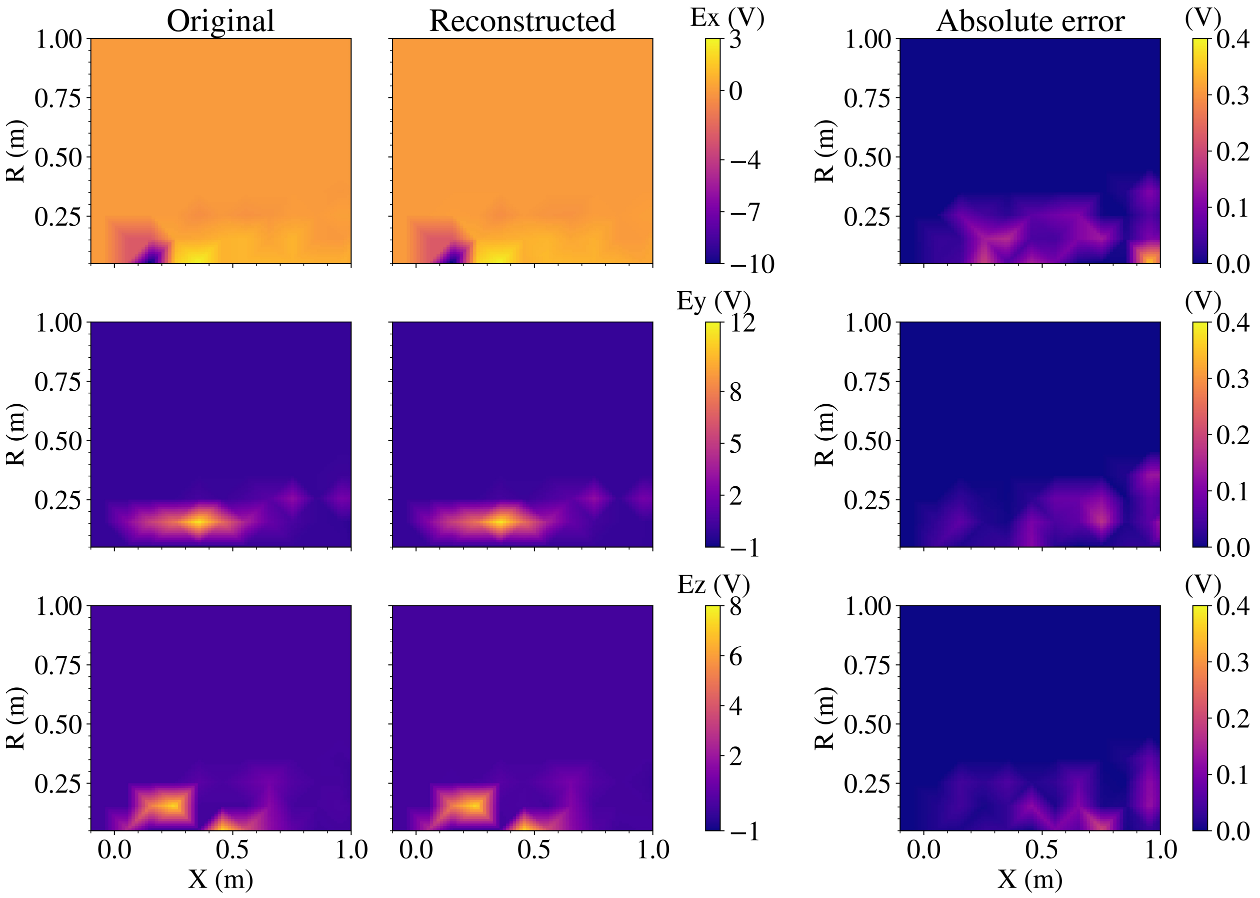

Figure 7: (Left) Electric field in the near-field plume region from the original dataset. (Middle) Electric field from the reconstructed dataset using a z-score normalization, and $\varepsilon = 10^{-1}$ target relative error. The components of the electric field are compressed together. This yields a very high $CR = 354,796\times$ as well as high accuracy to the original electric field contours. (Right) shows the absolute error in electric field components. The largest error is $0.356\,\mathrm{V}$ in the E_x component.

Table 3: Compressed data metrics for electric fields. The components of the electric field are compressed together using a z-score normalization. The electric field data is highly compressible and achieves extreme compression even at relatively high target error levels. The electric fields are not time-varying, and the ions are only accelerated electrostatically, leading to a latent space size of 1 and high compression.

	Size	Time (s)	ε	CR	TT-ranks	Latent size
Original	4.2 GB	-	-	-	-	-
Compressed	35 KB	15.31	10^{-1}	354,796	1, 1, 3, 3, 11, 8, 3, 3, 1, 1	1

ii. Surface data in unstructured mesh files

The fields stored in the unstructured mesh files are floating potential, electron current to the surface, ion current to the surface, total current to the surface, and charge accumulation on the surface of the thruster. The surface data has 1100 data-points for each of the 5 QoI. We reshape the 1100×5 surface snapshot array into a $10 \times 10 \times 11 \times 5$ array after normalizing each state of the snapshot individually, similar to the case with volume snapshots. For this class of experiments, we compress all 5 surface states together and report the following derived surface QoI: 1) floating potential integrated over the surface and 2) total current from the plasma to the surface.

Electrons and ions injected by the EP device contribute to the total currents in the plasma. When the surface potential is less than the plasma potential, the electron current to the thruster surface is calculated by integrating the Maxwellian distribution and assuming quasi-neutrality.[?] When the surface potential is larger than the plasma potential and the thin-sheath assumption can be made, the electron current is equal to the saturation current.[?] The ion current to the surface is computed using the electric flux through the surface.[?] The total current to the device surface from the plasma is shown in **??**. In the steady state portion of the simulation after around iteration 10000, under the assumption of quasi-neutrality, the electron current is equal to the ion current and the total current value asymptotes to 0 mA.

The floating potential on the thruster surface is calculated in TURF using the ion and electron temperatures, plasma potential, and the total current in the plasma. The floating potential on the surface is shown in **??**.

?? and **??** show the comparison between the derived surface QoI from the original and compressed datasets and the compression metrics. The reconstructed states, compressed without applying any normalization and using a target error of $\varepsilon = 10^{-6}$, results in a relatively high CR of 5.08×. The electric potential approximation is nearly exact to the original with a relative error of 3.52×10^{-10}. The reconstructed total current quantity closely follows the trend of the original until iteration 10000, after that point, the reconstructions deviate towards a total current of zero. The maximum relative error of 1.56 seen around iteration 13500 can be attributed to the fact that the entire simulation dataset was included in the approximation, including the iterations with transient behavior. Compressing the transient and steady state parts of the simulation separately may yield a better overall approximation performance.

Figure 8: Derived surface QoI from the original and reconstructed unstructured mesh data. Compression with no normalization and a target relative error of $\varepsilon = 10^{-6}$ results in a reconstructed dataset with CR = 5.077×. (Left) Electric potential quantities from compressed and reconstructed datasets are nearly one-to-one. (Right) The total current quantity from reconstruction follows the trend of the original quantity with a maximum relative error of 1.56 seen around iteration 13500.

Table 4: Compressed data metrics for surface quantity states. The components of the surface mesh data are compressed together without normalization and using a target error of $\varepsilon = 10^{-6}$. Even though the transient part of the simulation is included for this case study, the surface fields turn out to be compressible at a very high speed.

	Size (GB)	Time (s)	ε	CR	TT-ranks	Latent size
Original	6.6 MB	-	-	-	-	
Compressed	1.3 MB	0.39	10^{-6}	5.077	1, 5, 16, 176, 125, 1	125

4. Conclusion

The results of this study demonstrate the feasibility of using tensor network compression to reduce the size of simulation data generated by electric propulsion simulations. By learning an approximate TT-representation incrementally from the simulation snapshots, we can reduce data size $8 - 350,000\times$ without significant sacrifice in the accuracy. This reduction may also allow for increasing the frequency at which simulations are saved.

In addition to providing storage savings, tensor network compression allows for efficient analysis of the simulation data. The analysis outputs generated from the original and compressed datasets are compared to quantify the loss in simulation output fidelity from tensor network compression. The results show that the compressed datasets can capture the key features of the original datasets with high fidelity. Furthermore, the latent representation that is learned during the compression process can be used in reduced order modeling.[?] Future work will also focus on extending the use of tensor network compression to other types of EP simulations? and to develop a tool that can work in-tandem with simulation codes to progressively compress their outputs.

5. Acknowledgements

This research was supported in part through computational resources provided by Advanced Research Computing at the University of Michigan. Funding for this work was provided by NASA in part through the Joint Advanced Propulsion Institute (JANUS) under grant number 80NSSC21K1118. Additionally, we would like to acknowledge support by Los Alamos National Laboratory under the project "Algorithm/Software/Hardware Co-design for High Energy Density applications."

Appendix

A. Effect of normalization method and target relative error

The normalization methods and target relative error are an important design parameter in the compression pipeline for high-quality data reconstructions. Some of the states, such as particle number density, have large magnitude ranges over the computational domain, and the normalization method helps to reduce the range of these states when computing incremental updates to the TT-approximation. In this section we explore their effects on neutral number density contours. For many of the normalization methods and relative target errors tested, the reconstructed ion number contours resulted in satisfactory qualitative accuracy. The neutral number contours are more difficult to approximate due to wide range of density magnitudes in the domain and less diffuse plume. To see more discernable variations in reconstructions and for a more nuanced discussion on the effects of the compression pipeline parameters, we highlight the neutral number density contours.

As seen in **??**, the square root normalization with target error $\varepsilon = 10^{-1}$ leads to an extreme CR of $53,384\times$, but the higher target relative error results in a less accurate overall TT-ICE approximation, as expected, and numerical artifacts appear. This is in contrast to **??** which has lower target relative error of $\varepsilon = 10^{-2}$. The TT-approximation overestimates the number density in the backflow region. In **??**, the logarithmic normalization with base 10 $\varepsilon = 10^{-1}$ introduces statistical noise in the plume.

Table 5: Compression metrics for number density states using various normalization methods and target relative errors ε.

	Size	Time (s)	ε	CR	TT-ranks	Latent size
Original	5.5 GB	-	-	-	-	-
square root	133 KB	29.697	10^{-1}	53,384	1, 6, 25, 23, 26, 9, 7, 4, 2, 1	2
logarithmic base-10	333 MB	102.156	10^{-1}	16.53	1, 6, 59, 564, 2623, 1299, 254, 49, 25, 1	25
logarithmic-logarithmic	3.9 GB	454.1	10^{-2}	1.38	1, 6, 60, 600, 3000, 11767, 2500, 450, 150, 1	150
unitvector	1.1 MB	36.83	10^{-2}	5287.30	1, 6, 40, 133, 90, 21, 10, 5, 2, 1	2

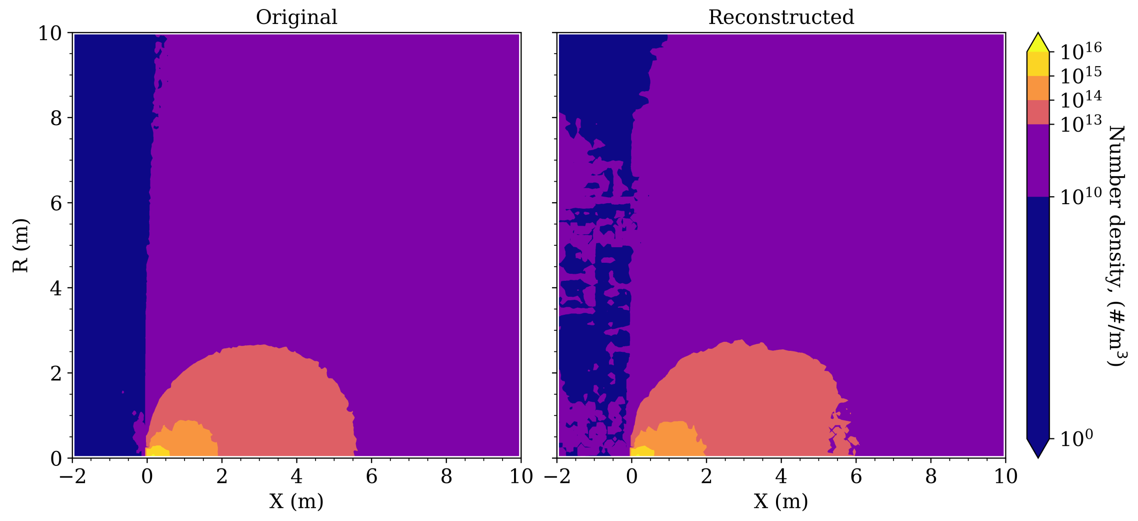

Figure 9: (Right) Neutral number density contours are shown for reconstructions using a square root normalization, $\varepsilon = 10^{-1}$ target relative error, and where particle number density fields are compressed together. Increasing the input target relative error in results in extreme compression with a CR $53,384\times$, however, decreases contour resolution and fails to capture the correct backflow phenomena when compared to the reconstruction seen in **??**. Here, the TT approximation overestimates the number density in the backflow region.

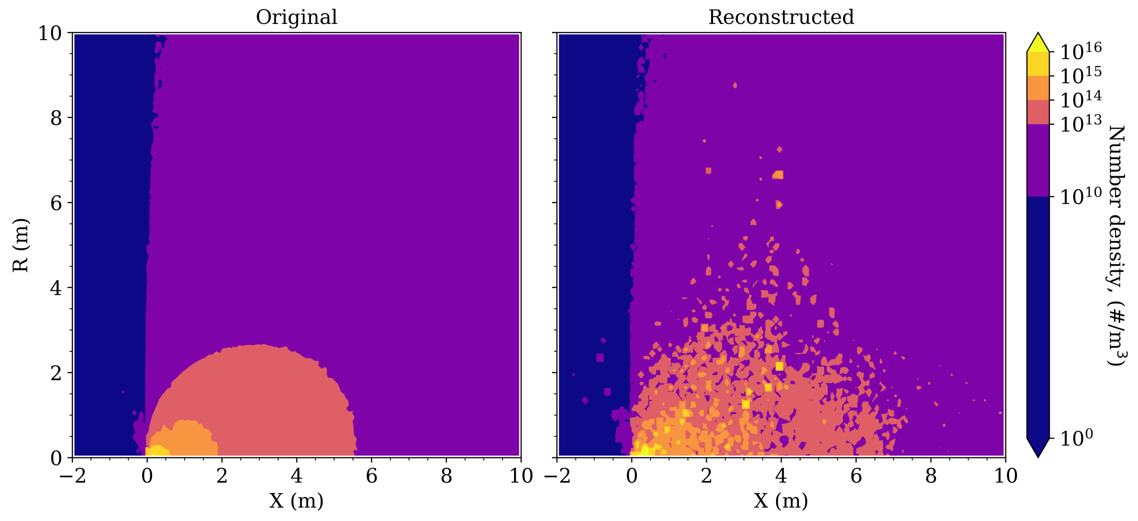

Figure 10: (Right) Neutral number density contours are shown for reconstructions using a logarithmic normalization with base 10, $\varepsilon = 10^{-1}$ target relative error, and where particle number density fields are compressed together. Although the CR is $16\times$ here, the reconstructed dataset has decreased resolution and more statistical noise in the plume contours for X $<$ 4m when compared to **??**. This is likely due to the normalization method used; the base-10 logarithm shrinks the range of values aggressively and makes the reconstruction sensitive to numerical errors.

As seen in **??**, applying the same normalization method twice does not necessarily provide accurate contour reconstructions. The second application of the normalization maps the span of the number density to a very small range and makes the reconstruction extremely sensitive to the changes in the reconstruction prior to normalization. Another possible explanation may be how the .vts files are read into NumPy arrays. Since the reconstruction quality of the plume degrades after X$>$ 4m, a highly likely explanation to this is that the subdomains are directly written into a separate dimension of a reshaped NumPy array, rather than being written into their correct spatial locations.

On the other hand, some normalization methods result in the compression favoring regions with higher number densities over sparsely populated regions while computing a a low-rank approximation. As seen in **??**, unit vector normalization with a target error of $\varepsilon = 10^{-2}$ fails to capture the backflow region behavior. Nevertheless, since this combination achieves extreme compression of 5287.3×, the accuracy around the plume might be useful if only the plume is of interest.

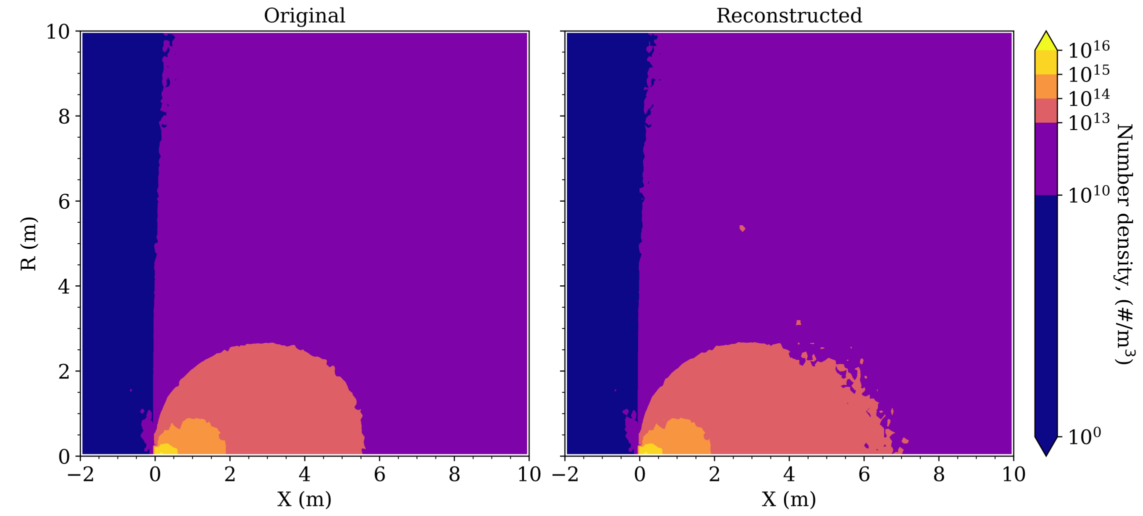

Figure 11: (Right) Neutral number density contours are shown for reconstructions using a logarithmic-logarithmic normalization, $\varepsilon = 10^{-2}$ relative error, and where neutral and ion number density fields are compressed together. cascading the same normalization method two times resulted in accurate contour modeling for X < 4 m but decreased resolution in the 10^{14} contour. CR = 1.38×.

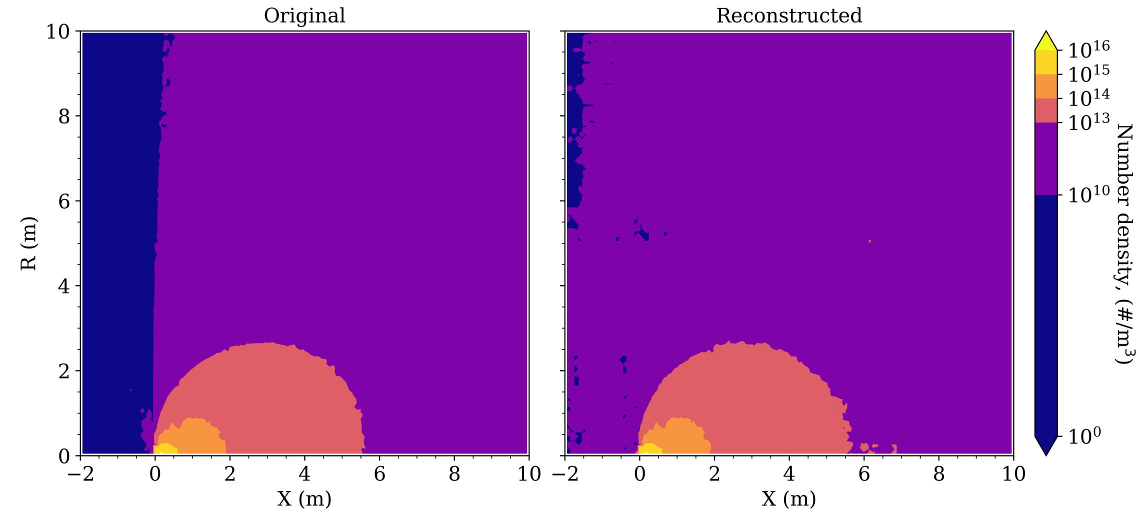

Figure 12: (Right) Neutral number density contours are shown for reconstructions using a unit vector normalization, $\varepsilon = 10^{-2}$ target relative error, and where particle number density fields are compressed together. Although we have a CR of 5287×, we see an inaccurate approximation of the backflow region.

B. Effect of individual state compression

Each of the volume mesh states from the TURF simulation used for analysis (ie. neutral number density, three types of ion number densities, ion current density, electric field components per cell) can be compressed separately or together. A separate compression pipeline for each state allows for flexibility in choosing a normalization method that exploits the state's data structures effectively and potentially yield a higher CR and/or a less statistically noisy approximation. As in ??, in this section, we present mainly neutral number density contours to see more discernible variations in reconstructions to observe the relationship between TT-ranks and quality of reconstructions.

The neutral number density contours in ??, are reconstructed using a z-score normalization with a relative error of $\varepsilon = 10^{-2}$, with the neutral number and ion number density fields compressed together resulting in a CR of 5213×. However the reconstruction suffers from reduced accuracy, especially in the far-plume region.

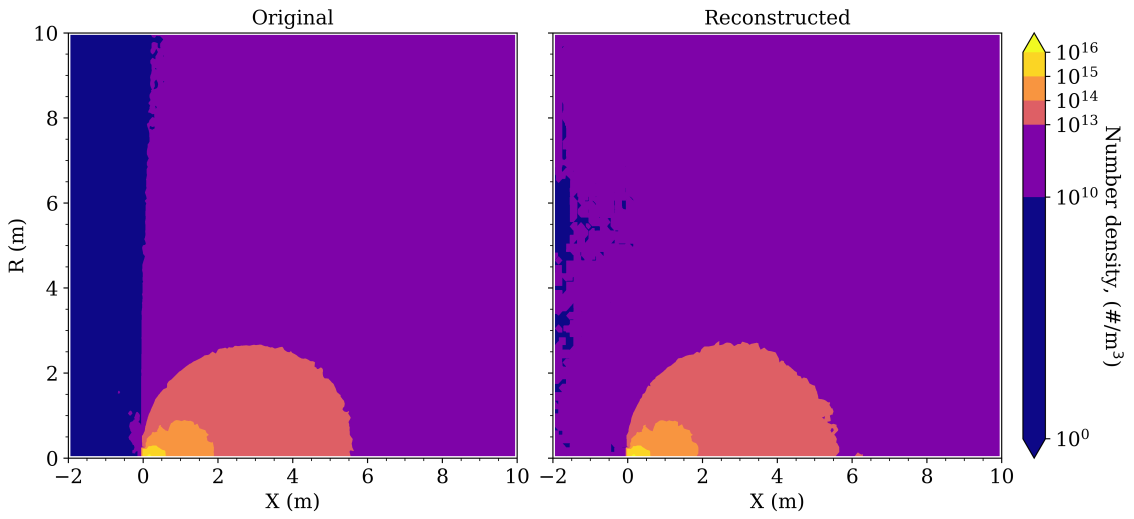

Figure 13: (Right) Neutral number density contours are shown for reconstructions using a z-score normalization, $\varepsilon = 10^{-2}$ target relative error, and where neutral and ion number density fields are compressed together. This results in an approximation with a CR of 5213×.

When neutral atoms are compressed separately from the ions using the same compression parameters, there is a significant increase in the CR to 24, 563×. As seen in ??, the resolution in the plume is not degraded significantly. For scenarios where the priority is respecting data storage limitations, this compression schema becomes highly attractive.

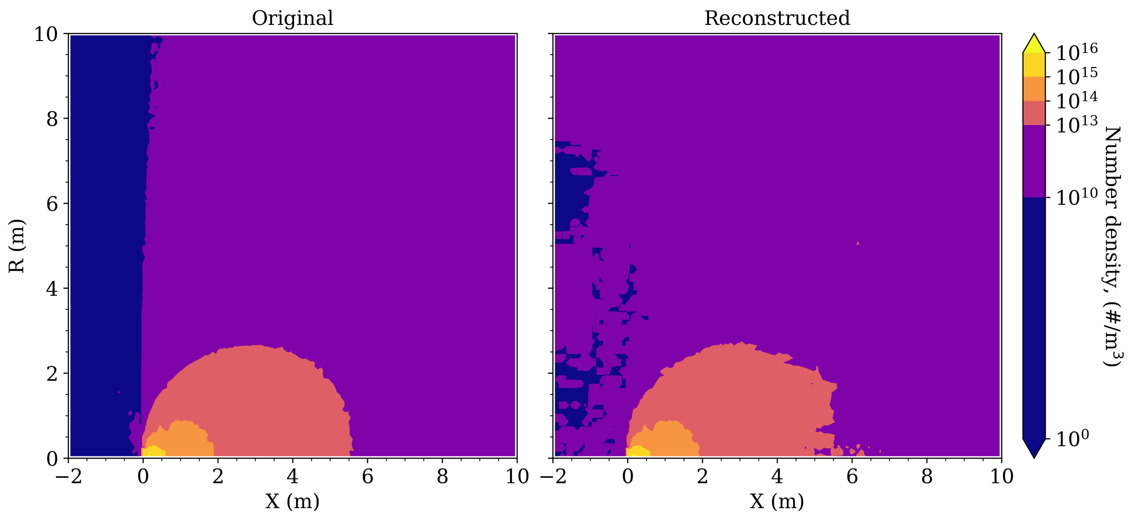

Figure 14: The reconstructed dataset used the same compression parameters as in ??, however, here, the neutral atoms are compressed separately from the other fields. Compressing separately increased the CR to 24,563×.

Similarly, in the case of using a logarithmic normalization and a $\varepsilon = 10^{-2}$ relative error, compressing the neutral state separately from the other states results in an increased CR from 1.37×, seen in **??**, to a CR of 1.91× seen in **??**. This reconstruction dataset exhibited a similar resolution and a marginal increase in CR. Separating the compression of the neutral number state from the other states seems to offer a balance of maintaining reconstruction resolution, with a slightly improved CR.

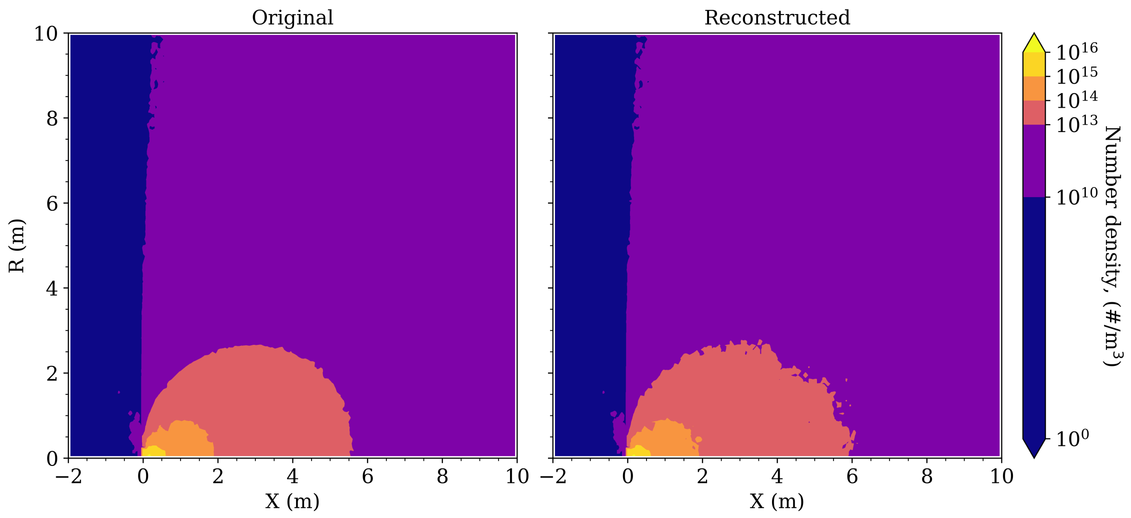

Figure 15: Neutral number density contours are shown for reconstructions using a logarithmic normalization, $\varepsilon = 10^{-2}$ target relative error, and where neutral and ion number density fields are compressed together. The CR is 1.37×.

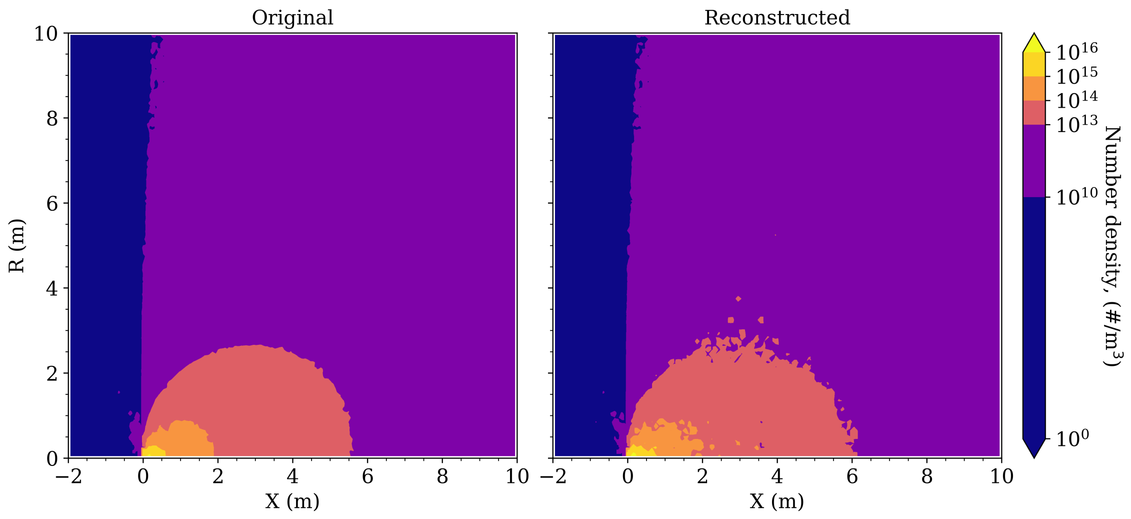

Figure 16: (Right) The reconstructed dataset has a CR of 1.91× and the same compression input parameters as in **??**, however, the neutral atom field is compressed separately from the other species. We see similar resolution of the reconstruction contours as in **??** and a marginally increased CR.

Table 6: Quantitative compression performance comparison between compressing number density states together vs compressing neutral number density separately. Compressing number density separately increases the CR. However, the amount of increase depends heavily on the normalization method preferred. Note that we approximate the size corresponding to the neutral atom density by dividing the total size of the compression by 4. All other metrics reported for number densities compressed together are for the entire compression.

	Norm	Figure #	Size	Time (s)	ε	CR	TT-ranks	Latent size
				z-score				
Original	-	-	1.3 GB		-	-	-	-
Together	z-score	??	275 KB[b]	36.12	10^{-2}	5213	1, 6, 40, 133, 91, 22, 10, 6, 2, 1	2
Separately	z-score	??	71 KB	4.78	10^{-2}	24,563	1, 5, 18, 15, 22, 7, 4, 2, 2, 1	2
				logarithmic				
Original	-	-	1.3 GB		-	-	-	-
Together	logarithmic	??	1 GB[b]	606.2	10^{-2}	1.371	1, 6, 60, 600, 2999, 11891, 2523, 450, 150, 1	150
Separately	logarithmic	??	719 MB	112.08	10^{-2}	1.910	1, 6, 60, 599, 2984, 2569, 585, 148, 148, 1	148

The ion particles are modeled as three different types in TURF: 1) slow-moving ions resulting from MEX collisions, fast-moving ions resulting from MEX collisions, and 3) slow-moving ions from CEX collisions. The total ion number density is a sum of these three types. **??** shows a reconstruction where the three ion states are compressed separately, each with its own normalization combination and target relative error input (please refer to **??** for details on compression settings). This flexible approach to different states results in ion number density contours that are very close to the originals, indicating that fine-tuning the compression parameters for each species or state can be a strategy to improve the accuracy of the reconstruction and have a nontrivial compression ratio.

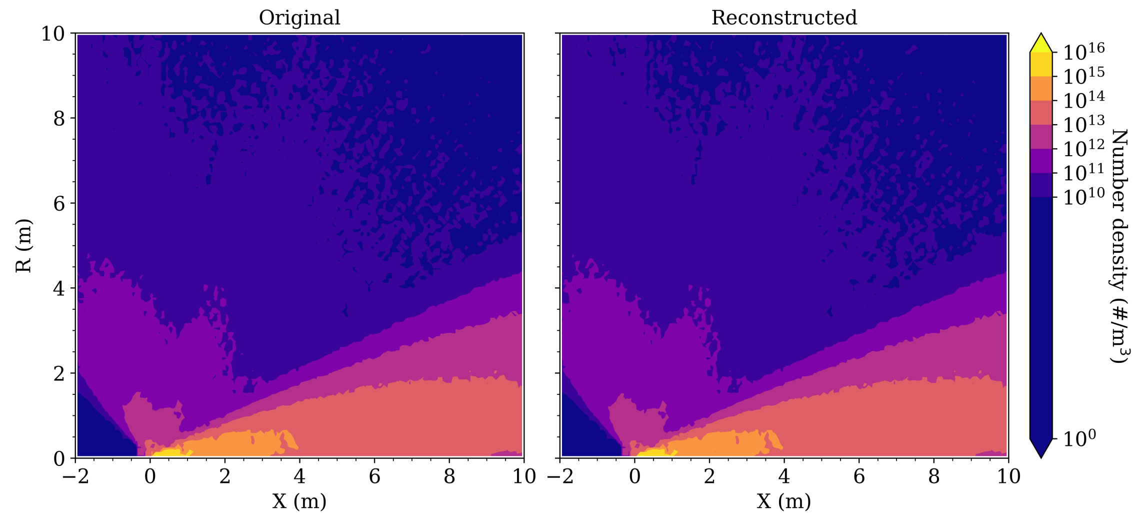

Figure 17: (Right) The reconstructed ion number density contours here correspond to when the neutral atoms and three types of ions are all compressed separately. Please refer to **??** for the details of compression parameters. The reconstructed ion number density contours are very close to the original contours while having an average CR of 2.88×.

[b]We approximated this value by dividing the corresponding compression's size on disk by 4.

Table 7: Compressed data metrics for ion number density states compressed separately. By compressing states separately, we can achieve extreme compression up to $553,846\times$. Note that the total CR is severely impacted by the compression of slow-moving CEX ions and can be greatly improved by finding a more performant compression setting.

	Normalization 1	Normalization 2	ε	CR
Slow-moving MEX ions	logarithmic	-	10^{-1}	553,846
Fast-moving MEX ions	z-score	-	10^{-4}	14.17
Slow-moving CEX ions	logarithmic	z-score	10^{-2}	1.03
Total CR				**2.881**

C. Effect of spatial contextualization

Another parameter to consider is how different subdomains within the same snapshot are inputted into the compression pipeline. They can be inputted in a way to preserve locality and spatial relations between each other.

To parallelize the simulation in TURF, we partition the larger computational (-2, 0, 0) m × (10, 10, 10) m domain into 2 along each direction, resulting in 8 subdomains. In **??**, **??**, and **??**, snapshots of size $60 \times 50 \times 50 \times n_s$ are accumulated along a new dimension. To determine the order in which each subdomain is written, this method only considered the binary codes (e.g., `0_0_0`, see **??** for more details) in the names of the files. In this section, we investigate the effect of "stitching" the subdomains together in a spatially-aware way before preprocessing and sending to the compression pipeline. The subdomains are stitched to their respective positions in physical space and passed into the algorithm. Coordinates for the two domain input structures are defined in **??**. As a result of this, we write the 8 subdomain snapshots having n_s states corresponding to the same iteration into a NumPy array of shape $120 \times 100 \times 100 \times n_s$. We then reshape this 4-dimensional array into a shape $10 \times 12 \times 10 \times 10 \times 10 \times 10 \times n_s \times 1$ similar to all previous cases and increment over the last dimension.

Table 8: Domain input structures for compression. The eight subdomains can be fed into the algorithm in a binary order which does not preserve the spatial relation between the subdomains in physical space, i.e (x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}). We investigate inputting a single spatially-aware domain where the subdomain respective positions in physical space are considered.

Binary subdomain ordering	**Spatially-aware subdomain ordering**
1. `0_0_0`: $(0, 60, 0, 50, 0, 50)$	1. `0_0_0`: $(0, 60, 0, 50, 0, 50)$
2. `0_0_1`: $(0, 60, 0, 50, 50, 100)$	2. `1_0_0`: $(60, 120, 0, 50, 0, 50)$
3. `0_1_0`: $(0, 60, 50, 100, 0, 50)$	3. `0_1_0`: $(0, 60, 50, 100, 0, 50)$
4. `0_1_1`: $(0, 60, 50, 100, 50, 100)$	4. `0_0_1`: $(0, 60, 0, 50, 50, 100)$
5. `1_0_0`: $(60, 120, 0, 50, 0, 50)$	5. `0_1_1`: $(0, 60, 50, 100, 50, 100)$
6. `1_0_1`: $(60, 120, 0, 50, 50, 100)$	6. `1_0_1`: $(60, 120, 0, 50, 50, 100)$
7. `1_1_0`: $(60, 120, 50, 100, 0, 50)$	7. `1_1_0`: $(60, 120, 50, 100, 0, 50)$
8. `1_1_1`: $(60, 120, 50, 100, 50, 100)$	8. `1_1_1`: $(60, 120, 50, 100, 50, 100)$

???? show the neutral and ion number contours when the subdomains are input into the compression algorithm as a single stitched domain, taking into account the spatial relation between the subdomains. Z-score normalization and a target relative error of $\varepsilon = 10^{-2}$ are used. For the neutral number densities, the contours of the near-field plume are representative of the original dataset, but the backflow region is not well approximated. There is decreased resolution in the ion number density contours, as if the simulation was run with fewer particles. However, the CR slightly increases from 5213.6× to 5287.6× by taking into account spatial relations. Providing spatially-aware input to the compression pipeline is another setting to consider based on analysis interest, resolution desired, and computational resource requirements. Compression metrics for this compression input schema are summarized in **??**.

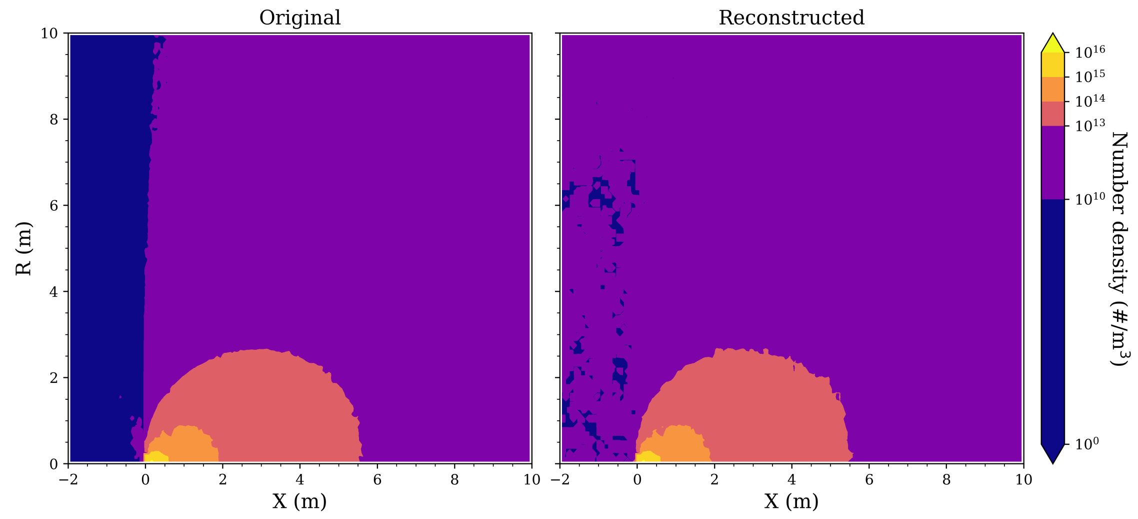

Figure 18: (Right) Neutral number density contours obtained through compression with binary ordering of subdomains, z-score normalization, and a target relative error of $\varepsilon = 10^{-2}$. This combination achieves a high CR of 5213.6×.

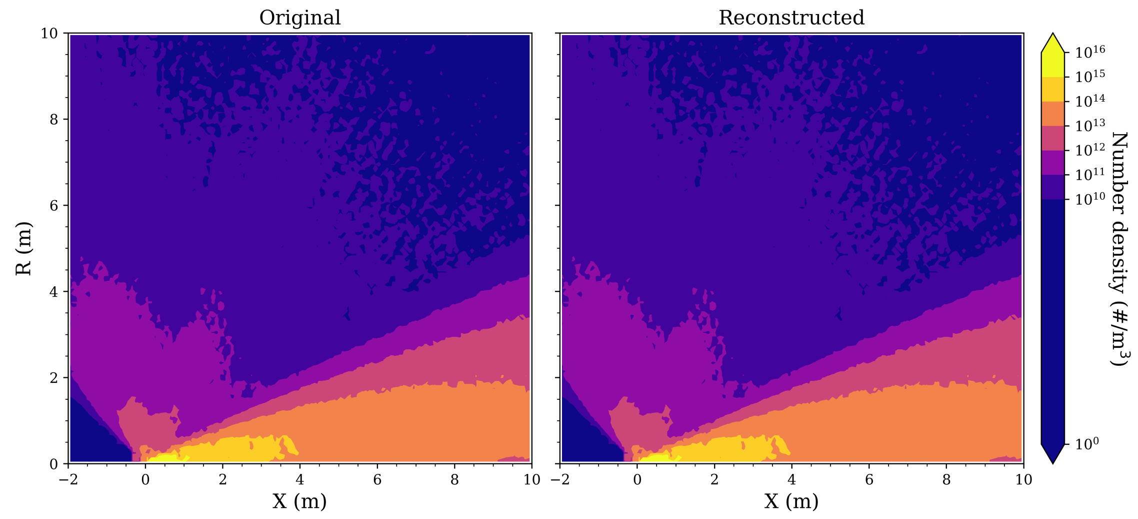

Figure 19: (Right) Ion number density contours obtained through compression with a spatially-aware domain input, z-score normalization, and a target relative error of $\varepsilon = 10^{-2}$. The CR is increases to 5287.6×.

?? and **??** show the effect of compression without and with considering spatial relations of the subdomains on the neutral number density contours, respectively. In both cases, square root normalization with $\varepsilon = 10^{-2}$ is used and one can see that the contours are near exact to those from the original dataset in both cases. Spatially-aware inputs do not significantly increase resolution over binary ordered inputs, and compression ratios between the two input schemas are comparable.

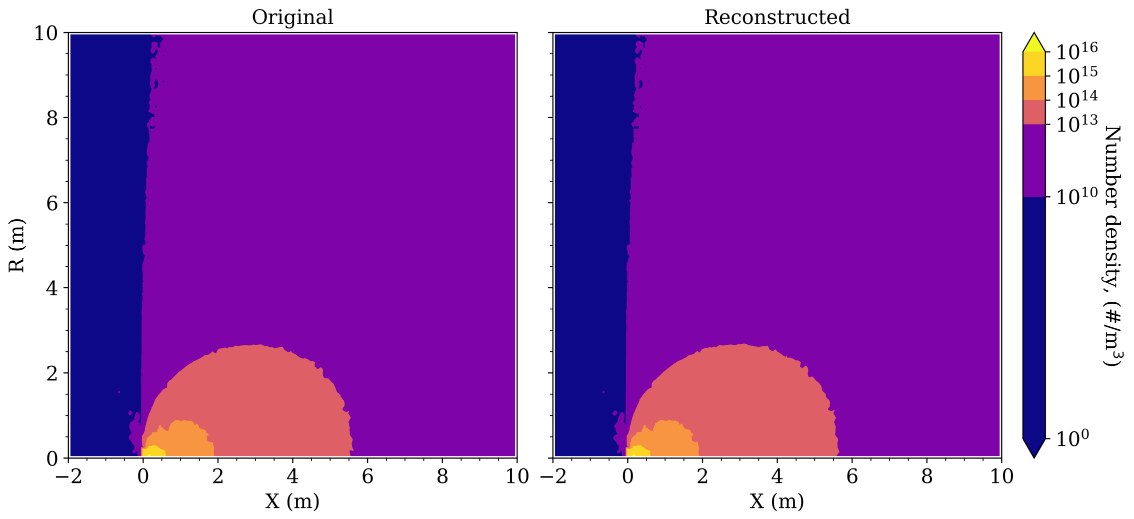

Figure 20: Neutral number density contours square root normalization and $\varepsilon = 10^{-2}$ using a binary ordering of subdomains in compression, yielding a CR of 8.07×

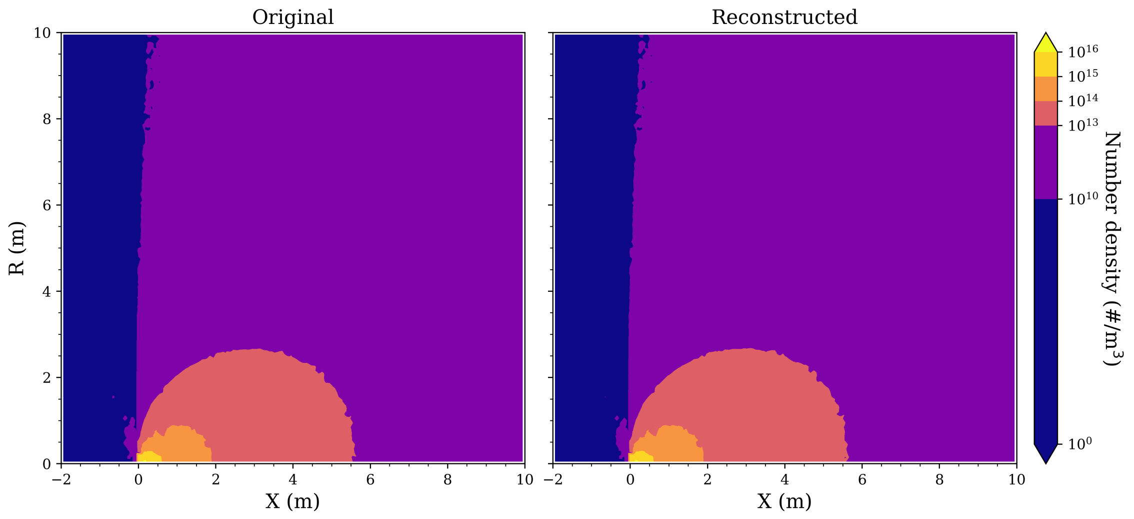

Figure 21: Neutral number density contours square root normalization and $\varepsilon = 10^{-2}$ using the spatially-aware domain input to compression. This input schema still yields accurate reconstructions to the original with reasonable compression ratio of 7.23×.

As in the neutral number density contours, the ion contours are near exact to those from the original dataset in both compression with and without considering the relations between subdomains, seen in **??** and **??** respectively.

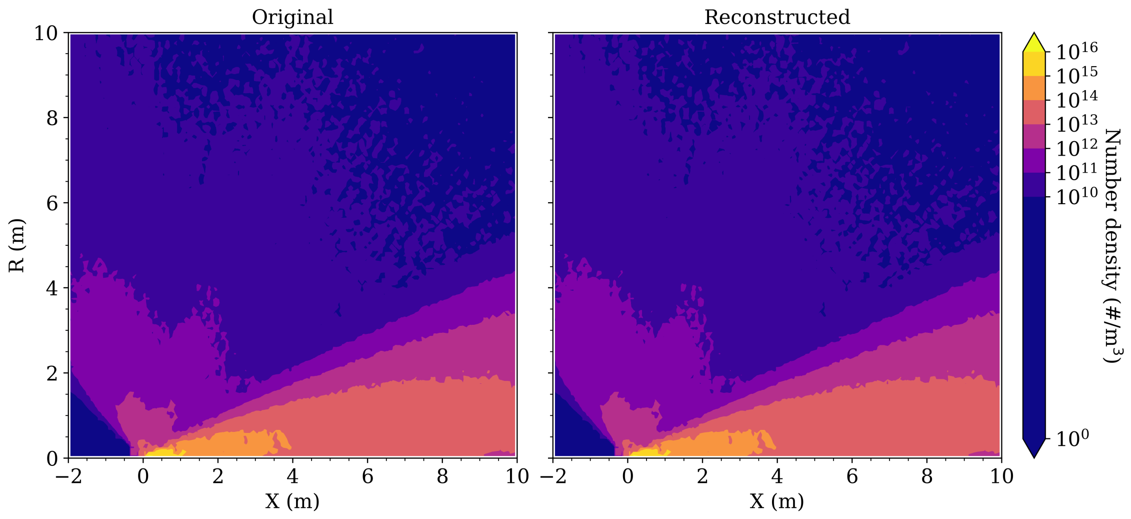

Figure 22: Ion number density contours square root normalization and $\varepsilon = 10^{-2}$ using the binary ordering of subdomains into compression, yielding a CR of 8.07×.

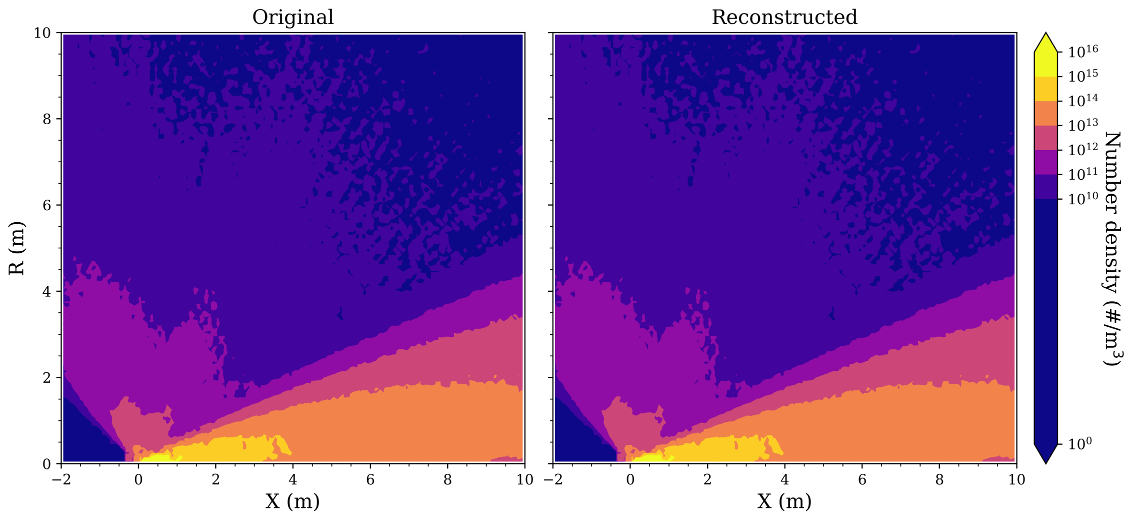

Figure 23: Ion number density contours square root normalization and $\varepsilon = 10^{-2}$ using the spatially-aware domain input to compression. This yields accurate reconstructions while having a nontrivial CR of 7.23×.

Table 9: Compression metrics for number density states using binary ordering and spatially-aware inputs. Using spatially-aware subdomain inputs results in accurate reconstructions, but not significantly increased resolution over binary ordered inputs. Further, the compression ratios between the two input schemas are comparable.

	Norm 1	Figure #	Size	Time (s)	ε	CR	TT-ranks	Latent size
				z-score, 10^{-2}				
Original	-	-	5.5 GB	-	-	-	-	-
Binary	z-score	??	1.1 MB	36.31	10^{-2}	5213.6	1, 6, 40, 133, 91, 22, 10, 6, 2, 1	2
Spatially-aware	z-score	??,??	1.1 MB	28.25	10^{-2}	5287.6	1, 10, 57, 97, 64, 17, 5, 2, 1	2
				square root, 10^{-2}				
Original	-	-	5.5 GB	-	-	-	-	-
Binary	square root	??,??	681 MB	135.63	10^{-2}	8.07	1, 6, 60, 586, 2829, 2590, 535, 71, 34, 1	34
Spatially-aware	square root	??,??	755 MB	127.17	10^{-2}	7.23	1, 12, 119, 1143, 5416, 650, 69, 34, 1	34

References

[1]Rafael Ballester-Ripoll, Peter Lindstrom, and Renato Pajarola. Tthresh: Tensor compression for multidimensional visual data. *IEEE transactions on visualization and computer graphics*, 26(9):2891–2903, 2019.

[2]Femke van Belzen and Siep Weiland. A tensor decomposition approach to data compression and approximation of nd systems. *Multidimensional Systems and Signal Processing*, 23(1):209–236, 2012.

[3]Woody Austin, Grey Ballard, and Tamara G Kolda. Parallel tensor compression for large-scale scientific data. In *2016 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 912–922. IEEE, 2016.

[4]Saibal De, Eduardo Corona, Paramsothy Jayakumar, and Shravan Veerapaneni. Tensor-train compression of discrete element method simulation data. *Journal of Terramechanics*, 113:100967, 2024.

[5]Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA working papers in phonetics*, 16(1):84, 1970.

[6]Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[7]Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[8]Doruk Aksoy, David J Gorsich, Shravan Veerapaneni, and Alex A Gorodetsky. An incremental tensor train decomposition algorithm. *SIAM Journal on Scientific Computing*, 46(2):A1047–A1075, 2024.

[9]Grey Ballard, Alicia Klinvex, and Tamara G Kolda. Tuckermpi: A parallel c++/mpi software package for large-scale data compression via the tucker tensor decomposition. *ACM Transactions on Mathematical Software (TOMS)*, 46(2):1–31, 2020.

[10]Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE signal processing magazine*, 32(2):145–163, 2015.

[11]Yipeng Liu, Jiani Liu, Zhen Long, and Ce Zhu. *Tensor computation for data analysis*. Springer, 2022.

[12]Longzhuang Li and Douglas Boulware. High-order tensor decomposition for large-scale data analysis. In *2015 IEEE International Congress on Big Data*, pages 665–668. IEEE, 2015.

[13]Iain D Boyd and Rainer A Dressler. Far field modeling of the plasma plume of a hall thruster. *Journal of Applied Physics*, 92(4):1764–1774, 2002.

[14]Shih Yu Chang and Hsiao-Chun Wu. Tensor quantization: High-dimensional data compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(8):5566–5580, 2022.

[15]R. Martin and J. Koo. Thermophysics universal research framework - infrastructure release. Air Force Research Laboratory (AFRL/RQRS), April 2015.

[16]Samuel J. Araki and Robert S. Martin. Dynamic load balancing with over decomposition in plasma plume simulations. *Journal of Parallel and Distributed Computing*, 163:136–146, 2022.

[17]Samuel J. Araki. Fast computation of high energy elastic collision scattering angle for electric propulsion plume simulation. *AIP Conference Proceedings*, 1786(1):170004, 2016.

[18]Samuel J. Araki and Richard E. Wirz. Ion–neutral collision modeling using classical scattering with spin-orbit free interaction potential. *IEEE Transactions on Plasma Science*, 41(3):470–480, 2013.

[19]J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[20]Johan Håstad. Tensor rank is np-complete. In *Automata, Languages and Programming: 16th International Colloquium Stresa, Italy, July 11–15, 1989 Proceedings 16*, pages 451–460. Springer, 1989.

[21]Hastad Johan. Tensor rank is np-complete. *Journal of Algorithms*, 4(11):644–654, 1990.

[22]Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

[23]Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[24]Andrzej Cichocki. Tensor networks for big data analytics and large-scale optimization problems. *arXiv preprint arXiv:1407.3124*, 2014.

[25]Huazhong Liu, Laurence T Yang, Yimu Guo, Xia Xie, and Jianhua Ma. An incremental tensor-train decomposition for cyber-physical-social big data. *IEEE Transactions on Big Data*, 7(2):341–354, 2018.

[26]Karim Abed-Meraim, Nguyen Linh Trung, Remy Boyer, et al. Adaptive algorithms for tracking tensor-train decomposition of streaming tensors. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 995–999. IEEE, 2021.

[27]Xiaokang Wang, Laurence T Yang, Yihao Wang, Lei Ren, and M Jamal Deen. Adtt: A highly efficient distributed tensor-train decomposition method for iiot big data. *IEEE Transactions on Industrial Informatics*, 17(3):1573–1582, 2020.

[28]Samuel J. Araki. Multiscale coupling of spacecraft charging model with electric propulsion plume simulation. *IEEE Transactions on Plasma Science*, 47(11):4898–4908, 2019.

[29]Samuel J Araki and Alexander Barrie. Electric propulsion plume simulation coupled with spacecraft charging. In *2018 Joint Propulsion Conference*, page 4906, 2018.

[30]Joshua D. Eckels, Thomas A. Marks, Doruk Aksoy, Sruti Vutukury, and Alex A. Gorodetsky. Dynamic mode decomposition for particle-in-cell simulations of a hall thruster and plume. In *IEPC 2024*, 2024.