Data-Driven Acceleration of Hall Thruster Simulations with a Sliding-Window Method

IEPC-2025-041

Presented at the 39th International Electric Propulsion Conference, Imperial College London, London, United Kingdom
14-19 September 2025

Joshua D. Eckels* and Alex A. Gorodetsky†
University of Michigan, Ann Arbor, MI 48105, USA

Alejandro Lopez Ortega‡
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA

This work develops an online method for robustly training data-driven reduced-order models (ROMs) for Hall thruster plasma simulations. Startup transients from simulation initialization are known to degrade long-term prediction accuracy of ROMs, so it is desirable for both training ROMs and terminating simulations to accurately detect the end-of-transience. To this end, the dynamic mode decomposition (DMD) is applied within a sliding-window algorithm to detect end-of-transience in the Hall2De fluid simulation code. The sliding-window method is shown to accurately detect the relaxation of simulation startup transients using a non-intrusive, data-driven approach. Furthermore, the method produces a ROM with more accurate long-term predictions compared to a ROM trained naively on initial simulation data, in some cases by an order of magnitude. Acceleration of the Hall2De simulation is achieved by early termination of the expensive physics solver. Nonlinear effects are observed to limit the accuracy of the method, which may be addressed in future work by using extensions to DMD or nonlinear alternatives.

Nomenclature

$u(\boldsymbol{x}, t)$	The ground truth physics solution field at spatial location $\boldsymbol{x}$ and time t, assumed to be a solution of a PDE of the form $\frac{\partial u}{\partial t} = \mathcal{F}(u)$, with spatial operator $\mathcal{F}$
$\boldsymbol{u}(t)$	The ground truth numerical solution at time t obtained by spatial discretization of $\mathcal{F}$ so that $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = F(\boldsymbol{u})$, with $\boldsymbol{u} \in \mathbb{R}^N$, $N = MQ$ for M mesh cells and Q field quantities at each cell. More generally, $\boldsymbol{u}(t)$ can be treated as the state vector at time t for a generic ODE $\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = F(\boldsymbol{u})$ with initial conditions $\boldsymbol{u}(0)$
$\hat{\boldsymbol{u}}(t)$	The data-driven prediction of the solution at time t using the reduced-order model $\frac{\mathrm{d}\hat{\boldsymbol{u}}}{\mathrm{d}t} = \hat{F}(\hat{\boldsymbol{u}}; b)$, with $\hat{\boldsymbol{u}} \in \mathbb{R}^N$ and tuneable model parameters b
t_l	The initial time of the training window
t_p	The initial time of the prediction window
T_l	The duration of the training window
T_p	The duration of the prediction window

*PhD candidate, Department of Aerospace Engineering, eckelsjd@umich.edu
†Associate professor, Department of Aerospace Engineering, goroda@umich.edu
‡Member of technical staff, alejandro.lopez.ortega@jpl.nasa.gov

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

L	The time between the end of the training window and the start of the prediction window
S	The sliding distance for the training and prediction windows
ε_{targ}	The target relative error below which to terminate the sliding-window algorithm
R	The number of successive iterations to meet the error target ε_{targ} before terminating
$\boldsymbol{u}_l, \hat{\boldsymbol{u}}_l$	The ground truth physics and reduced-order model solutions over the training window, respectively
$b^{(w)}$	The reduced-order model parameters learned using training data from the training window at window index w
$\hat{\boldsymbol{u}}_p^{(w)}$	The data-driven prediction over the prediction window, obtained with the reduced-order model trained over the training window at index w, i.e. $\hat{F}(\hat{\boldsymbol{u}}; b^{(w)})$
$\varepsilon^{(w)}$	The relative error between successive data-driven predictions

I. Introduction

Hall thrusters are leading candidates for in-space electric propulsion due to their high efficiency, cost-effectiveness, and high level of maturity. They produce thrust by ionizing and electrostatically accelerating a neutral gas to high exit velocities, obtaining a high specific impulse with low fuel requirements. Along with the proliferation of in-flight Hall thrusters has come the increasing need for physics-based modeling capabilities to aid testing and flight qualification, and furthermore to assist in design and optimization. Physical models of Hall thrusters typically involve capturing the flow of neutral gas from the anode down a cylindrical discharge channel, where the gas is ionized by a strong azimuthal electron current, accelerated by a strong electric field, and finally joined by a neutralizing stream of cathode electrons.[1] The dynamics of the discharge channel plasma are ultimately governed by the Boltzmann equation,[2] which provides a micro-scale kinetic description of the partially ionized gas. However, device-scale phenomena occur on much larger time and length scales. The primary challenge in modeling Hall thrusters at the device scale, then, is to accurately capture the physics at all relevant scales in a computationally efficient manner.

To this end, Hall thruster models have grown in both scale and complexity, ranging from 1D fluid descriptions[3,4] to 3D particle-in-cell (PIC).[5,6] Indeed, there is growing consensus that a highly resolved 3D kinetic simulation may be required to capture all of the complexity and physics of interest in a Hall thruster plasma.[7] In this case, increasing resolution and model fidelity quickly outpace modern computing capabilities. Furthermore, large-scale simulations of this nature are not practical for engineering tasks such as design and uncertainty quantification. There is therefore a need to develop methods to accelerate or otherwise reduce the cost of high-fidelity models while maintaining high accuracy.

One promising method for accelerating expensive physics solvers is through data-driven reduced-order models (ROMs). The primary advantage of most data-driven ROMs is that they are cheap to evaluate and can be learned in a black-box, non-intrusive manner; the only requirement is access to solution data from the physics solver. Once trained, a ROM provides a cheap approximation of the expensive solver that can be used for downstream tasks like design or optimization. Data-driven ROMs have been successfully applied to plasma systems[8,9] and to PIC Hall thruster simulations.[10,11] The primary challenge with this approach is building a ROM that extrapolates outside of the data it was trained on. This problem arises in most data-driven frameworks and typically stems either from poor training data or from model complexity. The goal is to use a model that is expressive enough to capture the behavior of the underlying physics solver without overfitting, and to train on data that is representative of all desired prediction regions. In the context of learning dynamics, such as the time-history of a PDE solver, this means handling nonlinearity in the physics model and training over non-transient regions of the response.

This work develops an online method to address these limitations in training accurate data-driven ROMs for Hall thruster simulations. The sliding-window method from Ref. 12 is extended to a general ROM setting and applied to the Hall2De fluid simulation code[13] for data-driven acceleration. The sliding-window method is shown to accurately detect end-of-transience and produce a ROM with better extrapolation performance

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

compared to simple forecasting. Acceleration is obtained by reducing the total time over which the expensive physics solver is evaluated. The contributions of this work include the extension of the sliding-window method to a general ROM setting and the demonstration of data-driven acceleration on a Hall thruster simulation. Section II outlines the sliding-window method and the Hall2De simulation setting. In Section III, the performance of the ROM is demonstrated on a test problem and on the Hall thruster simulation. Finally, the results are discussed in the broader context of data-driven reduced-order modeling and several paths for future work are suggested.

II. Methods

This section outlines the Hall2De simulation setting and the generalized sliding-window ROM method originally proposed in Ref. 12. Similar to prior work,[11,14] the dynamic mode decomposition is chosen to test the ROM methods and is also described in this section.

A. Hall2De simulation setting

Hall2De[13] solves the 2D fluid conservation equations for each plasma species in an axisymmetric axial-radial (z, r) coordinate system, as shown in Figure 1. Propellant neutrals are injected at the left anode boundary

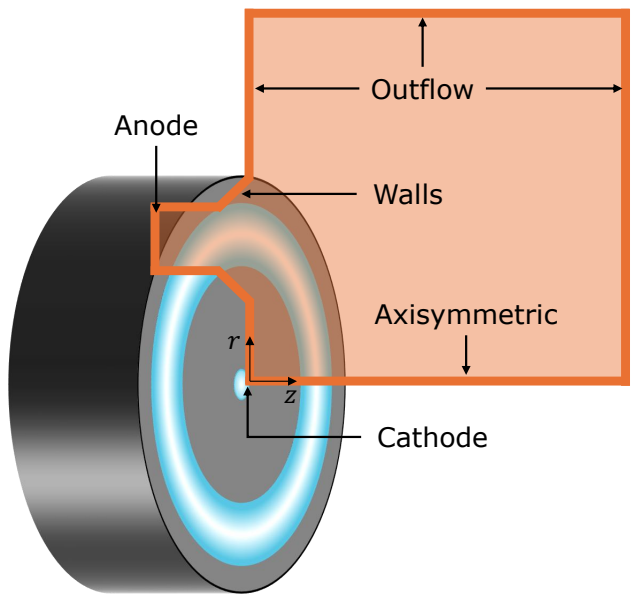

Figure 1. Illustration of the 2D axial-radial (z, r) simulation domain for the Hall2De code. The simulation domain is depicted in orange and boundary surfaces are labeled.

condition at a specified flow rate, and the anode is held fixed at a constant discharge voltage. Neutrals are ionized in the discharge channel (multiple ionization is allowed, but only single ionization is considered) and accelerated into the near-field plume. Thruster walls use insulating boundary conditions, pole covers are cathode-tied conductors, and all other domain exits use outflow conditions. This study considers the magnetically-shielded H9 Hall thruster[15] operating on Krypton propellant at a discharge current and voltage of 15 A and 300 V, respectively. Table 2 summarizes the primary simulation settings, and the original works[13,16] describe the equations of motion and numeric solvers in detail. Since the purpose of this study is to test the performance of ROM methods, the chosen simulation uses a standard setup and operating mode for clarity and ease of demonstration.

Hall2De solves for the time-evolution of plasma properties on a 2D quadrilateral magnetic-field-aligned mesh. The goal of a ROM in this context is to approximate the dynamics of these plasma properties in a computationally efficient manner, so that the Hall2De field solution may be cheaply estimated at a future time. Table 3 lists the primitive plasma properties of interest in the Hall2De solution. The solver obtains all primitives as solutions of a system of PDEs (note that neutral velocity is solved with a view factor algorithm[17] and electron current is obtained from the algebraic generalized Ohm's law, so these properties

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

Table 2. Hall2De numerical settings and operating conditions for the H9 Hall thruster simulation.

Simulation setting	Value
Propellant	Krypton
Mass flow rate	11.8 mg/s
Cathode flow fraction	7%
Discharge voltage	300 V
Discharge current	15 A
Background pressure	5 μTorr
Simulation duration	1 ms
Time step	10 ns
Number of cells	3925
Number of ion charge states	1

are excluded from the analysis).

Table 3. Primitive plasma properties of interest in the Hall2De simulation.

Field quantity	Symbol	Units
Ion density	n_i	m^{-3}
Ion axial velocity	$u_{i,z}$	m/s
Ion radial velocity	$u_{i,r}$	m/s
Electron temperature	T_e	eV
Electrostatic potential	ϕ	V

With the provided simulation settings, and saving the cell-center discretized solution every 20 time steps (0.2 μs), this amounts to a solution array of shape $(3925, 5, 5000)$ for 3925 cells, 5 primitives, and 5000 time snapshots (a total of roughly 785 MB in double-precision). Note that the simulation data size is comparatively small to 3D PIC[6] which may be $\mathcal{O}(10\text{ GB})$ per snapshot, yet it serves in this study as a useful platform for testing the ROM methods, which may be applied to larger-scale simulations in future work. All quantities were normalized separately to the range $(0, 1)$ for the ROM analyses.

B. The sliding-window method

This section outlines an adaptation of the sliding-window method[12] to the general online setting of training reduced-order models from simulation data. Typically, an expensive solver runs for a set length of time and a ROM is fit to time snapshots of the solution (via least squares regression, loss minimization, or similar). Then, the trained ROM forecasts the solution at future, unseen times. This approach involves truncating an initial "transient" portion of the simulation before training the ROM, since transient data obfuscates the long-term, "steady-state" behavior of the dynamics and reduces the prediction accuracy of the ROM. The sliding-window method automatically detects simulation end-of-transience in an online fashion, providing a robust way to truncate simulation data and to train a more accurate ROM for long-time forecasting. As in Ref. 12, the end-of-transience can be used as an early stopping criteria for the expensive solver, thereby providing computational speedup for any downstream analyses using the ROM in place of the solver. In this sense, the present work also seeks data-driven "acceleration" by providing an early termination for the Hall2De simulation. Algorithm 1 outlines the sliding-window method, using the nomenclature defined previously, and Figure 2 provides an illustration of the method. The method is modified in three primary ways from Ref. 12: **1)** the support of general reduced-order models (via $\hat{F}(\hat{\boldsymbol{u}}; b)$) beyond DMD (although only DMD has been tested here, the extension allows consideration of nonlinear models), **2)** a more robust error termination criteria, and **3)** a consolidation of the important hyperparameters.

To detect end-of-transience, the method incrementally trains a ROM over a sliding portion of the sim-

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

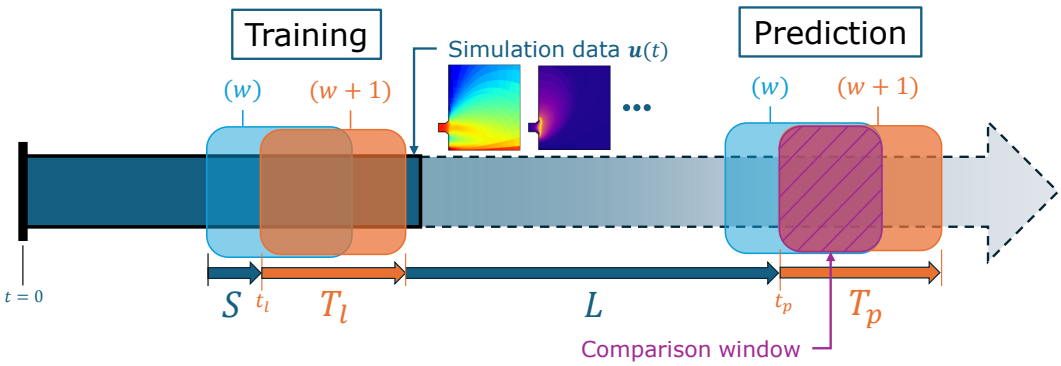

Figure 2. Illustration of the sliding-window method for learning a data-driven reduced-order model. A ROM is learned using data in the training window and forecasts are made in the prediction window. The windows are incremented in time by distance S from index w (cyan) to index $w+1$ (orange). ROMs are compared in the overlapping portion of the prediction windows (purple). Simulation data $\boldsymbol{u}(t)$ from the ground truth physics solver accumulates over time (solid blue bar). The same nomenclature is used as in Algorithm 1.

Algorithm 1 Sliding-window method for training a data-driven reduced-order model, adapted from Ref. 12.

Input: Durations: (T_l, T_p, L, S), Targets: (ε_{targ}, R), Initial conditions: $\boldsymbol{u}(0)$
Initialize: $w = 1,\ t_l = 0,\ \hat{\boldsymbol{u}}(0) = \boldsymbol{u}(0)$
  $t_p \leftarrow t_l + T_l + L$
  $t_{f,\text{prev}} \leftarrow t_p + T_p$
  $\boldsymbol{u} \leftarrow \text{SolveODE } \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = F(\boldsymbol{u})$ for $t \in [t_l, t_l + T_l]$
loop
  $\boldsymbol{u}_l \leftarrow [\boldsymbol{u}(t) : t \in [t_l, t_l + T_l]]$ ▷ Train reduced-order model
  $\hat{\boldsymbol{u}}_l \leftarrow \text{SolveODE } \frac{\mathrm{d}\hat{\boldsymbol{u}}}{\mathrm{d}t} = \hat{F}(\hat{\boldsymbol{u}}; b)$ for $t \in [t_l, t_l + T_l]$
  $b^{(w)} \leftarrow \text{argmin}_b \|\boldsymbol{u}_l - \hat{\boldsymbol{u}}_l\|$
  $\hat{\boldsymbol{u}}_p^{(w)} \leftarrow \text{SolveODE } \frac{\mathrm{d}\hat{\boldsymbol{u}}}{\mathrm{d}t} = \hat{F}(\hat{\boldsymbol{u}}; b^{(w)})$ for $t \in [t_p, t_p + T_p]$

  if $w > 1$ **then** ▷ Compute error over comparison window
    $\hat{\boldsymbol{u}}_\varepsilon^{(w)} \leftarrow \left[\hat{\boldsymbol{u}}_p^{(w)}(t) : t \in [t_p, t_{f,\text{prev}}]\right]$
    $\hat{\boldsymbol{u}}_\varepsilon^{(w-1)} \leftarrow \left[\hat{\boldsymbol{u}}_p^{(w-1)}(t) : t \in [t_p, t_{f,\text{prev}}]\right]$
    $\varepsilon^{(w)} \leftarrow \frac{\|\hat{\boldsymbol{u}}_\varepsilon^{(w)} - \hat{\boldsymbol{u}}_\varepsilon^{(w-1)}\|_2}{\|\hat{\boldsymbol{u}}_\varepsilon^{(w-1)}\|_2}$
    $t_{f,\text{prev}} \leftarrow t_p + T_p$
    if $w \geq R$ **then**
      if $\text{all}(\varepsilon^{(w)}, \varepsilon^{(w-1)}, \ldots, \varepsilon^{(w-R)}) < \varepsilon_{targ}$ **then**
        exit loop ▷ Terminate simulation
      end if
    end if
  end if

  $\boldsymbol{u} \leftarrow \text{concatenate}(\boldsymbol{u}, \text{SolveODE } \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = F(\boldsymbol{u}) \text{ for } t \in [t_l + T_l, t_l + T_l + S])$ ▷ Slide windows
  $t_l \leftarrow t_l + S$
  $t_p \leftarrow t_l + T_l + L$
  $w \leftarrow w + 1$
end loop

ulation data (i.e. the training window) and makes predictions over a future prediction window. As the simulation progresses, the training and prediction windows advance forward in time. The ROM predictions from successive iterations are compared in the overlapping section of the prediction windows. When the

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

difference between successive ROM predictions converges below a set tolerance, this indicates a stable ROM solution and end-of-transience in the simulation, and the solver terminates. Table 4 summarizes the main hyperparameters of the algorithm and rough guidelines for choosing their values.

Table 4. Description and guidelines for the sliding-window algorithm hyperparameters.

Parameter	Symbol	Guidelines
Training window length	T_l	Should be large enough to cover several oscillation cycles, which may require some prior knowledge.
Prediction window length	T_p	Larger values encourage a greater degree of agreement between successive ROM predictions. Typically set equal to training window length.
Prediction window offset	L	Determines the extent of time past training where non-transient behavior should be expected. Typically set to a multiple $\sim 5-10$ of the training window length.
Sliding distance	S	Balances precision in identifying end-of-transience with algorithm cost. Small values identify end-of-transience more accurately but increase computational cost due to more ROM train/predict iterations.
Target relative error	ε_{targ}	When to terminate the expensive solver is based on whether relative error in ROM predictions is below this threshold for R consecutive iterations. ε_{targ} reflects desired accuracy in ROM extrapolations (i.e. 1%, 10%, etc.)
Repetitions for termination	R	Larger values increase robustness against noise but may delay detection of end-of-transience. Typically set to $\sim 1-5$.

Note: While the sliding-window method is presented in continuous time, practically one must work with discrete-time data, so it is advisable to choose sliding distance and window lengths that are integer multiples of the simulation time step for ease of implementation.

C. Dynamic mode decomposition

This section summarizes the operating principles of the dynamic mode decomposition (DMD).[18,19] In the notation of Algorithm 1, DMD is used in this study as the specific choice of ROM $\hat{F}(\hat{\boldsymbol{u}}; b)$ in the sliding-window method. DMD has been developed and applied to many simulation settings, including several plasma systems,[8,10,11] due to its simplicity, interpretability, and ease of implementation. DMD seeks a linear approximation of the dynamics of a system via:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = F(\boldsymbol{u}) \approx A\boldsymbol{u}, \quad \boldsymbol{u}_0 \triangleq \boldsymbol{u}(t=0) \tag{1}$$

so that the approximate state $\hat{\boldsymbol{u}}$ has a closed-form analytical solution:

$$\hat{\boldsymbol{u}}(t) = e^{At}\boldsymbol{u}_0. \tag{2}$$

The system dynamics matrix A is learned from m discrete-time snapshots of simulation data via a least-squares fit:

$$U = \begin{bmatrix} | & & | \\ \boldsymbol{u}(t_1) & \dots & \boldsymbol{u}(t_{m-1}) \\ | & & | \end{bmatrix}, \; U' = \begin{bmatrix} | & & | \\ \boldsymbol{u}(t_2) & \dots & \boldsymbol{u}(t_m) \\ | & & | \end{bmatrix}, \tag{3}$$

$$U' \approx AU, \quad \text{with} \tag{4}$$

$$A = \underset{B}{\operatorname{argmin}} \|U' - BU\|_2 = U'U^{\dagger}. \tag{5}$$

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

Typically, the state vector $\hat{\boldsymbol{u}}$ has high dimension and so a rank-r truncated singular value decomposition (SVD) is used to obtain the modal-decomposition form of Eq.(2):

$$\hat{\boldsymbol{u}}(t) \approx \sum_{i=1}^{r} v_i e^{\omega_i t} a_i, \tag{6}$$

where $v_i \in \mathbb{R}^N$ and ω_i are the DMD modes and eigenvalues, respectively, and a_i is an amplitude from the i-th component of the product $[v_i \dots v_N]\boldsymbol{u}_0$. This study additionally considers a constant offset in the linear model via $\mathrm{d}\hat{\boldsymbol{u}}/\mathrm{d}t = A\hat{\boldsymbol{u}} + c$ to capture non-zero steady-state values by solving for an augmented state vector $[\hat{\boldsymbol{u}} \mid 1]^T$. As mentioned previously, the two primary limitations of data-driven ROM prediction accuracy are model complexity and training data quality. This study adopts a simple linear model and primarily focuses on improving the latter aspect by detecting and training on non-transient data via the sliding-window algorithm. Training with steady-state data is especially important for linear DMD, where nonlinear transients can significantly degrade long-time prediction accuracy. Future work will additionally consider nonlinearity in the model.[20–22] Presently, only the exact-DMD method is considered.[19]

III. Results

This section presents the performance of the sliding-window method on two problems: first on the nonlinear Duffing oscillator to highlight end-of-transience detection, and second on the H9 Hall thruster simulation described in Section II. The results demonstrate the accuracy of the method in detecting end-of-transience and emphasize its potential for providing data-driven acceleration to expensive physics solvers. Several challenges remain with respect to capturing model nonlinearity.

A. Duffing oscillator

The Duffing oscillator is used as a toy problem with controllable transient behavior to test the sliding-window algorithm. For position x and velocity v of the oscillator, the dynamics of the state vector $\boldsymbol{u} = [x, v]$ are expressed as:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} - s(t)\beta \begin{bmatrix} 0 \\ x^3 \end{bmatrix} + f(t) \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{7}$$

with damping ζ, natural frequency ω_0, forcing $f(t)$, and a nonlinear spring force scaled by β. The envelope $s(t)$ enforces a smooth transition from $s(t = 0) = 1$ to $s(t = t_c) = 0$, effectively removing the nonlinear term at the specified cutoff time t_c (a quintic smoothstep polynomial is used for the envelope). When $t > t_c$, the system is exactly linear and the sliding-window method with DMD should capture this nonlinear→linear transition. The system is integrated with a fourth-order Runge-Kutta scheme to a final time of $t_f = 100$ and a constant forcing $f(t) = c$ is used. Table 5 lists the values of all simulation parameters.

Table 5. Simulation parameters for the Duffing oscillator problem Eq.(7).

Parameter	Symbol	Value
Initial condition	$\boldsymbol{u}_0$	$[0, 0]$
Time step	Δt	0.02
Final time	t_f	100
Transient cutoff time	t_c	40
Damping ratio	ζ	0.02
Natural frequency	ω_0	1
Nonlinear scale	β	5
Constant forcing	c	0.5

The goal of the sliding-window method is to detect the cutoff t_c by incrementally training over simulation data until the linear DMD model converges. Table 6 summarizes all sliding-window hyperparameters used for the Duffing oscillator and the Hall2De simulation.

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

Table 6. Sliding-window hyperparameters (Algorithm 1) for the Duffing oscillator problem and the Hall2De simulation.

Parameter	Symbol	Duffing	Hall2De
Training window length	T_l	13	0.15 ms
Prediction window length	T_p	13	0.15 ms
Prediction window offset	L	20	0.2 ms
Sliding distance	S	1	0.01 ms
Target relative error	ε_{targ}	0.001	0.01
Repetitions for termination	R	7	3

Figure 3 shows the true oscillator position (x) over time (t) compared to three ROM predictions: **1)** the initial ROM trained over the initial training window ($w = 0$), **2)** a ROM trained over an intermediate window ($w = 30$), and **3)** the final ROM upon termination of the sliding-window algorithm at $w = 45$. The nonlinear→linear transition at $t_c = 40$ is evident by the transition to a damped harmonic oscillator decaying to the steady-state value of $c = 0.5$. The high prediction error for ROMs trained in the nonlinear region $t < t_c$

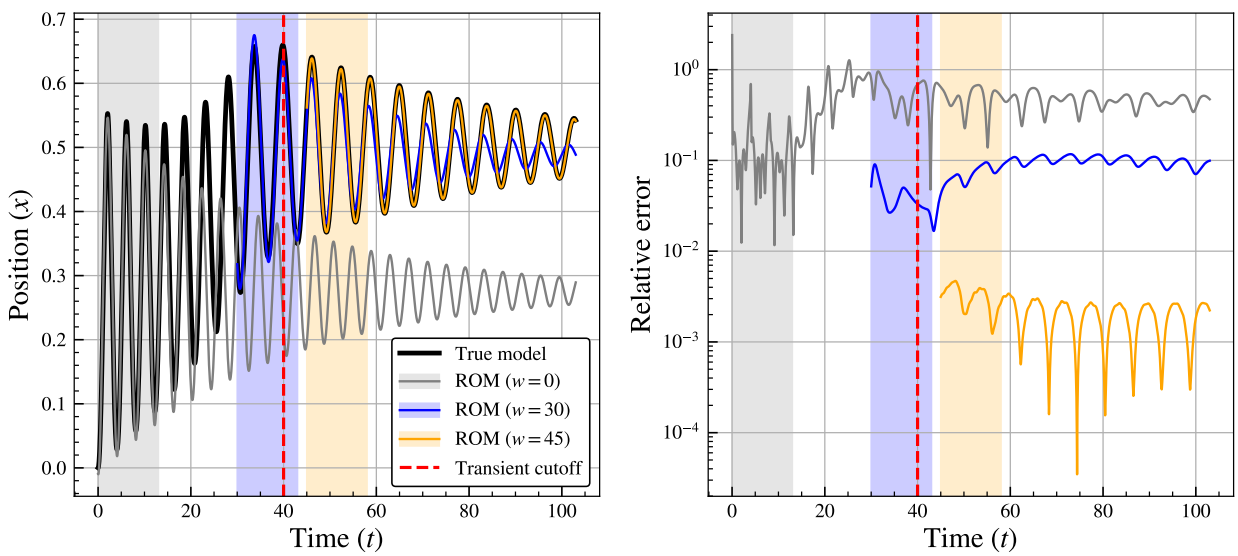

Figure 3. The true oscillator position (x) over time (t) compared to the ROM for three training windows (left) and the relative error between the ROMs and the true model (right). The training windows are highlighted for each case: the initial window at $w = 0$ (gray), an intermediate window at $w = 30$ (blue), and the final window ($w = 45$) at termination of the sliding-window algorithm (orange). The transient cutoff time $t_c = 40$ is indicated by a red vertical line. Overall, the final window ROM performs best for long-term forecasting, indicating successful detection of simulation end-of-transience.

emphasizes the intuitive result that the linear ROM (DMD) performs best when trained on linear simulation data. Furthermore, the sliding-window method accurately detects this transition by terminating at window $w = 45$, just slightly past the true cutoff at $w = 40$. Figure 4 highlights this result by showing the decay in ROM comparison error ($\varepsilon^{(w)}$) below the target (ε_{targ}) right at the transient cutoff. For robust detection of the cutoff in the presence of noise, the algorithm terminates only when $\varepsilon^{(w)} < \varepsilon_{targ}$ for R consecutive iterations; this is why the detected cutoff is slightly past the true cutoff. The goal of the sliding-window method is primarily to detect end-of-transience; to improve the accuracy of the DMD prediction itself, it is likely necessary to extend the training window, reduce the time step, or consider other DMD variants.[19]

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

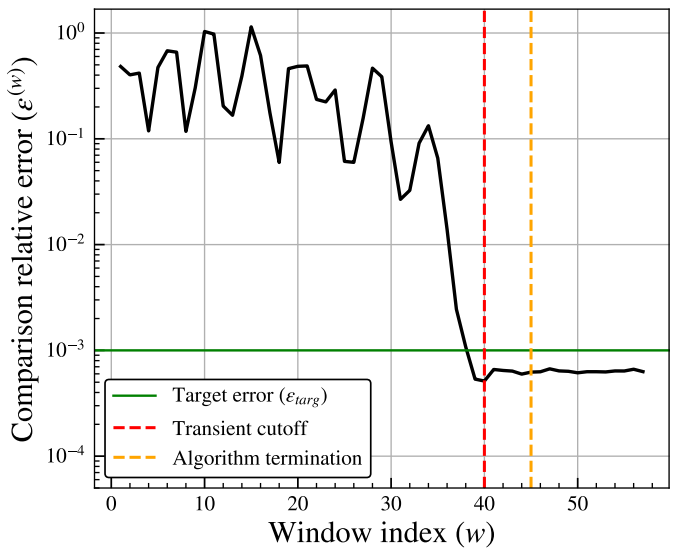

Figure 4. Decay of ROM comparison error ($\varepsilon^{(w)}$) over sliding window index (w) for the Duffing oscillator problem. The transient cutoff occurs at $w = 40$ (red) and the algorithm terminates at $w = 45$ (orange). The comparison error drops below the threshold ε_{targ} (green) when ROM predictions converge, which occurs in the linear region past the transient cutoff, indicating successful detection of simulation end-of-transience.

B. Hall2De simulation

Hall2De imposes several challenges beyond the simple Duffing oscillator, most prominently high-dimensionality and nonlinearity. The high dimension stems from a state vector $\boldsymbol{u} \in \mathbb{R}^{MQ}$ for $M = 3925$ finite-volume cells and $Q = 5$ primitive plasma properties at each cell (see Table 3). A rank-r truncated SVD largely mitigates the problem of dimension, as the numerical solution of PDEs can often be accurately represented in a much lower-dimensional "latent" space. For SVD, the latent space corresponds to a basis of r eigenvectors (or modes) that reconstruct the data within some error tolerance; greater truncation trades off with greater reconstruction error.

The nonlinearity stems from the underlying system of PDEs $\frac{\partial u}{\partial t} = \mathcal{F}(u)$, where the differential operator $\mathcal{F}(u)$ contains nonlinear terms such as advection ($u\frac{\partial u}{\partial x}$), fluxes ($\nabla \cdot (n\vec{u})$), etc. In this context, the goal of the sliding-window method is to detect the relaxation of startup transients from initial conditions. In the context of Hall thruster plasma simulations, this transition corresponds to advection of the initial plasma downstream and a relaxation to equilibrium operation. Ideally, equilibrium entails quasi-periodic oscillations (i.e. the breathing mode) or constant, steady-state values, both which can be accurately captured by linear DMD. By detecting the onset of equilibrium, the sliding-window method permits early termination of the expensive solver and cheap, long-term ROM forecasts. Table 6 includes the sliding-window hyperparameters for the Hall2De analysis, and a rank $r = 330$ SVD truncation was used for all DMD fits.

Figure 5 shows the thrust computed from the true Hall2De simulation over time compared to three ROM predictions: **1)** the initial ROM ($w = 0$), **2)** an intermediate ROM ($w = 18$), and **3)** the final ROM upon termination of the sliding-window algorithm ($w = 44$). The thrust is computed in all cases as the surface integral over the outflow boundaries of ion momentum flux, and provides a single, scalar metric to compare the ROMs to the true simulation. While thrust provides an indication of goodness of fit at the domain boundaries, the relative error in Figure 5 is computed over the entire simulation domain, providing an average performance metric over all cell quantities.

In this specific Hall2De configuration, the target discharge current is achieved by iterating a global factor on the anomalous electron transport. The relaxation of discharge current (and thrust by proxy) to a constant value characterizes the transition from transient dynamics to steady-state. Figure 5 illustrates this transition in the true model around $t = 0.4$ ms where thrust reaches a constant value (note that discharge current is not shown because it requires prediction of electron current density, which is not considered in the present analysis). Accordingly, ROM prediction accuracy improves as the training window slides past the transition, resulting in a final training window around $t = 0.42$ ms where the sliding-window accurately detects the

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

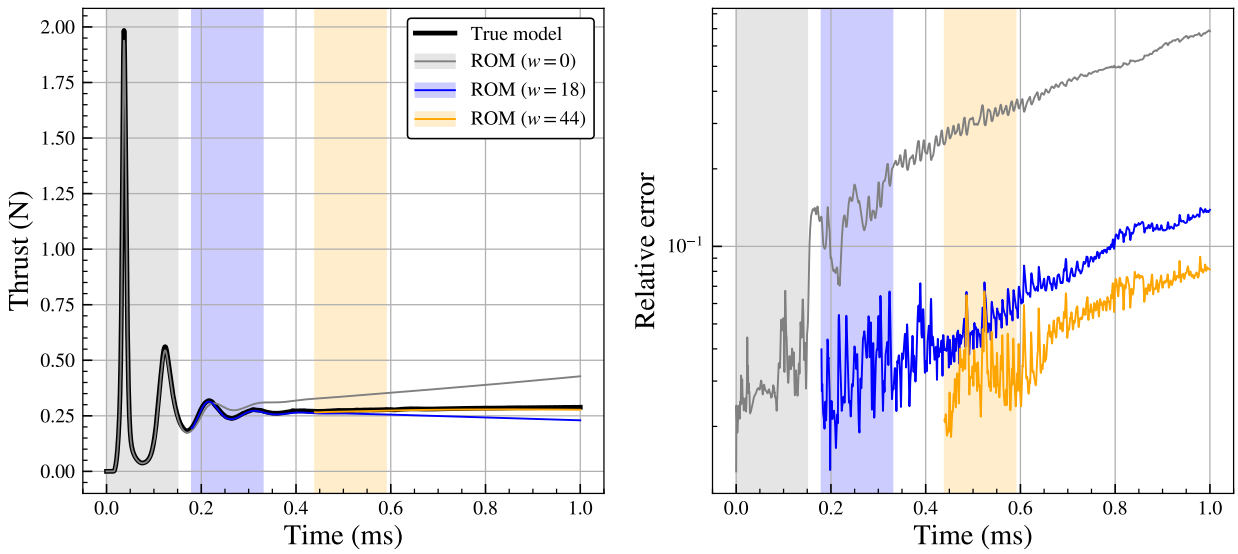

Figure 5. The thrust computed from the true Hall2De simulation over time compared to the ROM for three training windows (left) and the relative error between the ROMs and the true model (right). The training windows are highlighted for each case: the initial window at $w = 0$ (gray), an intermediate window at $w = 18$ (blue), and the final window ($w = 44$) at termination of the sliding-window algorithm (orange). The thrust is computed over the outflow boundaries and the relative error is computed over the full simulation domain. Overall, the final window ROM performs best for long-term forecasting, indicating successful detection of simulation end-of-transience.

transition and terminates. Figure 6 highlights this result by showing the decay in ROM comparison error $\varepsilon^{(w)}$ below the target ε_{targ}, sending the flag for termination of the expensive solver at window index $w = 44$. Note that immediately after termination, the comparison error increases back *above* the threshold. This may

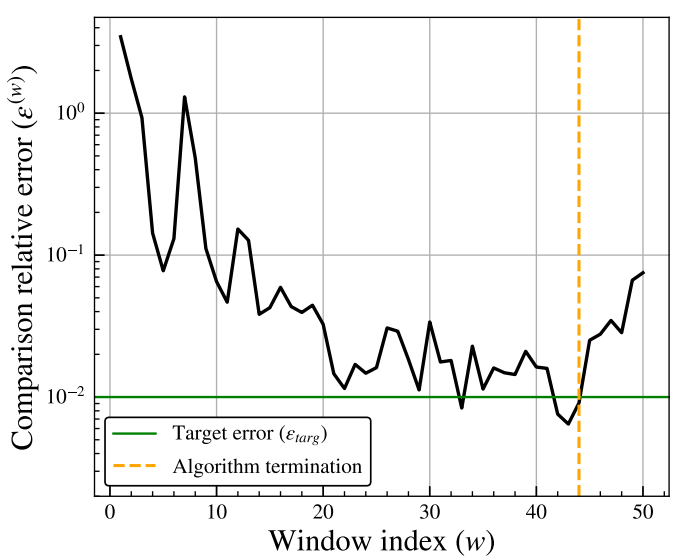

Figure 6. Decay of ROM comparison error ($\varepsilon^{(w)}$) over sliding window index (w) for the Hall2De simulation. The algorithm terminates at $w = 44$ (orange) when the comparison error drops below the threshold ε_{targ} (green) for $R = 3$ consecutive iterations, indicating the end of simulation transience. However, nonlinear effects or numerical instabilities may cause the comparison error to increase back above the threshold.

indicate the presence of a small nonlinear effect or noise in the training data that causes ROM prediction error to increase. Another possibility is numerical instability related to performing DMD on a nearly constant signal; if there are no dynamics in the data, then the snapshot matrix is ill-conditioned, leading to spurious

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

results in the DMD fit. Longer training windows, data augmentation, or increasing the repetition R may resolve these numerical issues. Accounting for nonlinearity in the model may be required for more complicated effects.

Figures 7 and 8 provide full-field comparisons of the ROM to the true model for ion density and velocity, respectively. The first row shows the final time prediction ($t = 1$ ms) for the ROM trained on the initial window ($w = 0$), and the second row shows the final time prediction for the ROM at algorithm termination ($w = 44$).

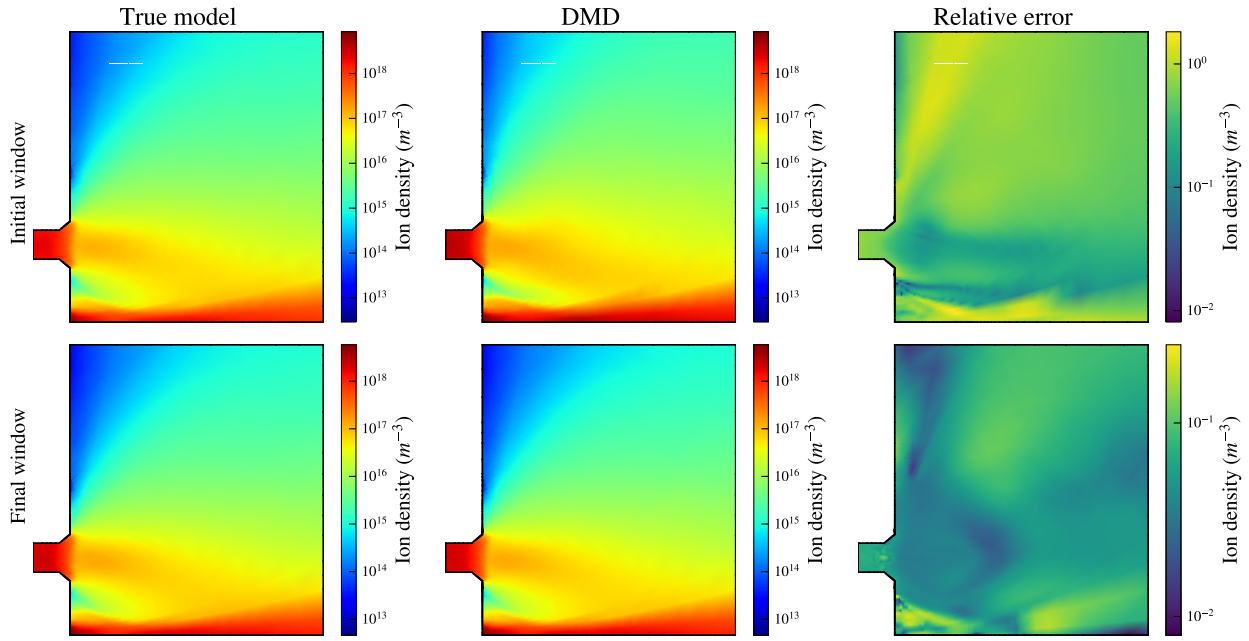

Figure 7. Full-field comparison of the ROM final time prediction ($t = 1$ ms) of ion density to the true model. The ROM predictions are shown when trained on the initial window (top row) and when trained on the final window at algorithm termination (second row). The last column provides point-wise relative error between the ROM and true model within the simulation domain, showing an order of magnitude reduction in error from the initial window to the final window.

The comparisons provide further validation that ROMs trained only on steady-state data (as identified by the sliding-window) perform better for long-time forecasting than when trained on transient initial data, as evidenced by the order of magnitude reduction in error for the final window versus the initial window. Furthermore, the algorithm terminates correctly after startup transients have relaxed to equilibrium. For both ion density and velocity, the initial window ROM exhibits a slowly-growing, unstable mode, which artificially increases the field values over time. Figure 5 also displayed this result as a growing thrust prediction for the initial window ($w = 0$). Conversely, the final window ROM accurately predicts the primary stable mode and the constant value of thrust, albeit with a slightly decaying mode that appears to damp the field values over time. The off-channel-center region, characterized by radial beam expansion, exhibits the largest errors in the simulation domain for both ion density and velocity. The presence of nonlinear advection terms in the fluid PDEs likely explains this discrepancy and highlights the limitations of the linear approximation in DMD.

Note that the present Hall2De configuration attempts to equilibrate the plasma to a constant (non-oscillating) beam profile to achieve the target, constant discharge current. It would be considerably easier in this case to detect end-of-transience by simply monitoring the discharge current and terminating when its time-derivative falls below some threshold. A more interesting case might be a nontrivial equilibrium like the breathing mode, or even long-term nonlinear effects. In this context, the present analysis serves as a proof-of-concept for training robust ROMs for Hall thruster plasma simulations; future work includes extending these methods to more complicated scenarios.

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

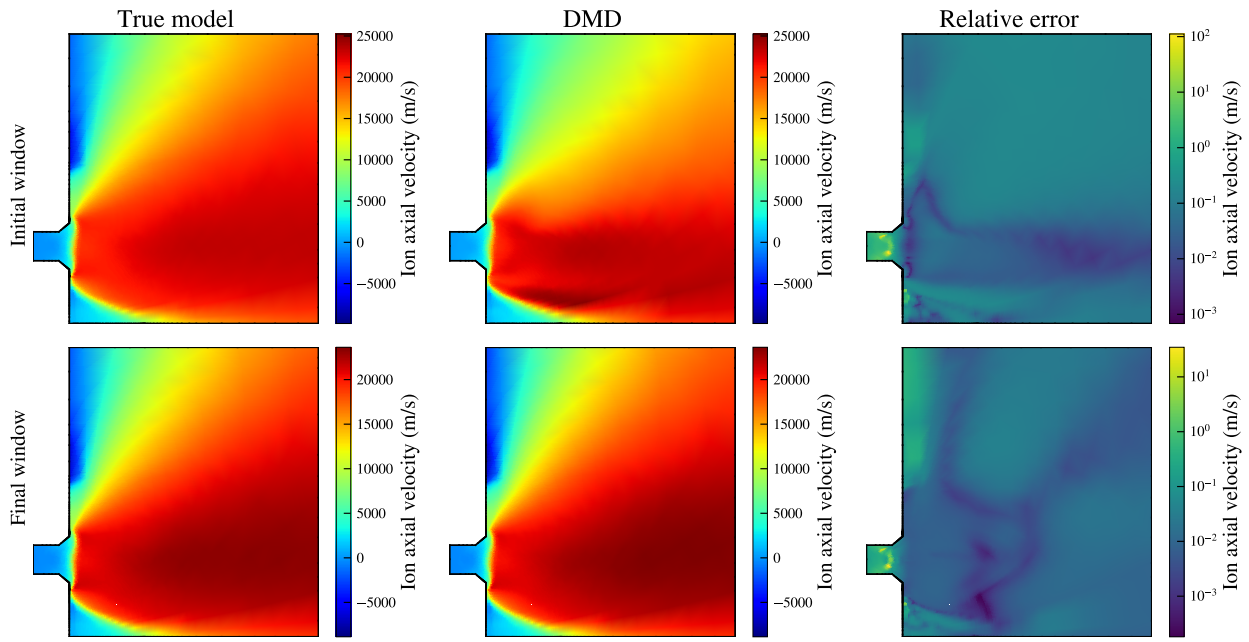

Figure 8. Full-field comparison of the ROM final time prediction ($t = 1$ ms) of axial ion velocity to the true model. The ROM predictions are shown when trained on the initial window (top row) and when trained on the final window at algorithm termination (second row). The last column provides point-wise relative error between the ROM and true model within the simulation domain, showing an order of magnitude reduction in error from the initial window to the final window.

IV. Conclusion

This work developed an online method for robustly training data-driven reduced-order models (ROMs) for Hall thruster plasma simulations. The linear dynamic mode decomposition (DMD) was applied within a sliding-window algorithm to detect the end-of-transience in a fluid Hall thruster simulation. The sliding-window method demonstrated the ability to accurately detect the equilibration of simulation startup transients using a non-intrusive, data-driven framework. Furthermore, the method produced a ROM with more accurate long-term predictions compared to a ROM trained naively on initial simulation data. Acceleration of the Hall thruster simulation was achieved by early termination of the expensive physics solver.

Primary challenges for the method include high-dimensionality and nonlinearity. As physical simulations grow in size and complexity to capture the challenging, multiscale plasma dynamics in Hall thrusters, so does the need for data compression and reduced-order modeling methods. A rank-truncated singular value decomposition was applied within DMD to handle the high dimension in this work, but larger-scale simulations may require more sophisticated compression methods. Nonlinearity stems from the PDEs governing most physical phenomena and limits the performance of linear methods like DMD. Future work includes accounting for nonlinear effects in the ROM and potentially integrating physics or structural constraints. A related avenue to explore is feeding physics-constrained ROMs back into the underlying solver for greater computational speedup.

Acknowledgments

Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). Funding for this work was provided by NASA in part through the Joint Advanced Propulsion Institute (JANUS) under grant number 80NSSC21K1118, as well as in part through a NASA Space Technology Graduate Research Opportunity grant 80NSSC23K1181. The authors would also like to acknowledge Dr. Thomas Marks for assistance in setting up the Hall2De simulation code.

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.

References

[1]Boeuf, J.-P., "Tutorial: Physics and Modeling of Hall Thrusters," *Journal of Applied Physics*, Vol. 121, No. 1, Jan. 2017, pp. 011101.

[2]Kramer, R. M. J., et al., "A Plasma Modeling Hierarchy and Verification Approach," Tech. Rep. SAND-2020-3576, Sandia National Lab. (SNL-NM), Albuquerque, NM (United States), March 2020.

[3]Sahu, R., Mansour, A. R., and Hara, K., "Full Fluid Moment Model for Low Temperature Magnetized Plasmas," *Physics of Plasmas*, Vol. 27, No. 11, Nov. 2020, pp. 113505.

[4]Schedler, P., *HallThruster.Jl 1D Fluid Model for a Hall Thruster Discharge*, Ph.D. thesis, Institute of Fluid Dynamics, Zurich, Switzerland, 2022.

[5]Taccogna, F. and Minelli, P., "Three-Dimensional Particle-in-Cell Model of Hall Thruster: The Discharge Channel," *Physics of Plasmas*, Vol. 25, No. 6, June 2018, pp. 061208.

[6]Villafana, W., Cuenot, B., and Vermorel, O., "3D Particle-in-Cell Study of the Electron Drift Instability in a Hall Thruster Using Unstructured Grids," *Physics of Plasmas*, Vol. 30, No. 3, March 2023, pp. 033503.

[7]Kaganovich, I. D., et al., "Physics of E × B Discharges Relevant to Plasma Propulsion and Similar Technologies," *Physics of Plasmas*, Vol. 27, No. 12, Dec. 2020, pp. 120601.

[8]Nicolini, J. L., Na, D.-Y., and Teixeira, F. L., "Model Order Reduction of Electromagnetic Particle-in-Cell Kinetic Plasma Simulations via Proper Orthogonal Decomposition," *IEEE Transactions on Plasma Science*, Vol. 47, No. 12, Dec. 2019, pp. 5239–5250.

[9]Nayak, I., Teixeira, F. L., and Burkholder, R. J., "On-the-Fly Dynamic Mode Decomposition for Rapid Time-Extrapolation and Analysis of Cavity Resonances," *IEEE Transactions on Antennas and Propagation*, Vol. 72, No. 1, Jan. 2024, pp. 131–146.

[10]Faraji, F., Reza, M., Knoll, A., and Kutz, J. N., "Dynamic Mode Decomposition for Data-Driven Analysis and Reduced-Order Modeling of E × B Plasmas: I. Extraction of Spatiotemporally Coherent Patterns," *Journal of Physics D: Applied Physics*, Vol. 57, No. 6, Nov. 2023, pp. 065201.

[11]Eckels, J. D., Marks, T. A., Aksoy, D., Vutukury, S., and Gorodetsky, A., "Dynamic Mode Decomposition for Particle-in-Cell Simulations of a Hall Thruster and Plume," *38th International Electric Propulsion Conference*, Toulouse, France, June 2024.

[12]Nayak, I., Teixeira, F. L., Na, D.-Y., Kumar, M., and Omelchenko, Y. A., "Accelerating Particle-in-Cell Kinetic Plasma Simulations via Reduced-Order Modeling of Space-Charge Dynamics Using Dynamic Mode Decomposition," May 2024.

[13]Mikellides, I. G. and Katz, I., "Numerical Simulations of Hall-effect Plasma Accelerators on a Magnetic-Field-Aligned Mesh," *Physical Review E*, Vol. 86, No. 4, Oct. 2012, pp. 046703.

[14]Nayak, I., Kumar, M., and Teixeira, F. L., "Detection and Prediction of Equilibrium States in Kinetic Plasma Simulations via Mode Tracking Using Reduced-Order Dynamic Mode Decomposition," *Journal of Computational Physics*, Vol. 447, Dec. 2021, pp. 110671.

[15]Hofer, R. R., Cusson, S. E., Lobbia, R. B., and Gallimore, A. D., "The H9 Magnetically Shielded Hall Thruster," *35th International Electric Propulsion Conference*, Ann Arbor, MI, USA, Oct. 2017.

[16]Lopez Ortega, A. and Mikellides, I. G., "A New Cell-Centered Implicit Numerical Scheme for Ions in the 2-D Axisymmetric Code Hall2De," *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, AIAA Propulsion and Energy Forum, American Institute of Aeronautics and Astronautics, July 2014.

[17]Katz, I. and Mikellides, I. G., "Neutral Gas Free Molecular Flow Algorithm Including Ionization and Walls for Use in Plasma Simulations," *Journal of Computational Physics*, Vol. 230, No. 4, Feb. 2011, pp. 1454–1464.

[18]Schmid, P. J., "Dynamic Mode Decomposition of Numerical and Experimental Data," *Journal of Fluid Mechanics*, Vol. 656, Aug. 2010, pp. 5–28.

[19]Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., and Kutz, J. N., "On Dynamic Mode Decomposition: Theory and Applications," *Journal of Computational Dynamics*, Vol. 1, No. 2, Mon Dec 01 00:00:00 UTC 2014, pp. 391–421.

[20]Li, Z., et al., "Fourier Neural Operator for Parametric Partial Differential Equations," May 2021.

[21]Menier, E., Kaltenbach, S., Yagoubi, M., Schoenauer, M., and Koumoutsakos, P., "Interpretable Learning of Effective Dynamics for Multiscale Systems," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 481, No. 2306, Jan. 2025, pp. 20240167.

[22]Alves, E. P. and Fiuza, F., "Data-Driven Discovery of Reduced Plasma Physics Models from Fully Kinetic Simulations," *Physical Review Research*, Vol. 4, No. 3, Sept. 2022, pp. 033192.

Copyright 2025 by the Electric Rocket Propulsion Society. All rights reserved.