A Novel In Situ Machine Learning Framework for Intelligent Data Capture and Event Detection

T. M. Shead, I. K. Tezaur, W. L. Davis IV, M. L. Carlson, D. M. Dunlavy, E. J. Parish, P. J. Blonigan, J. Tencer, F. Rizzi, and H. Kolla

Abstract We present a novel framework for automatically detecting spatial and temporal events of interest in situ while running high performance computing (HPC)

T. M. Shead
Sandia National Laboratories, Albuquerque, NM, USA
e-mail: tshead@sandia.gov

I. K. Tezaur
Sandia National Laboratories, Livermore, CA, USA
e-mail: ikalash@sandia.gov

W. L. Davis IV
Sandia National Laboratories, Albuquerque, NM, USA
e-mail: wldavis@sandia.gov

M. L. Carlson
Sandia National Laboratories, Livermore, CA, USA
e-mail: maxcarl@sandia.gov

D. M. Dunlavy
Sandia National Laboratories, Albuquerque, NM, USA
e-mail: dmdunla@sandia.gov

E. J. Parish
Sandia National Laboratories, Livermore, CA, USA
e-mail: ejparis@sandia.gov

P. J. Blonigan
Sandia National Laboratories, Livermore, CA, USA
e-mail: pblonig@sandia.gov

J. Tencer
Sandia National Laboratories, Albuquerque, NM, USA
e-mail: jtencer@sandia.gov

F. Rizzi
NexGen Analytics, Sheridan, WY, USA
e-mail: francesco.rizzi@ng-analytics.com

H. Kolla (✉)
Sandia National Laboratories, Livermore, CA, USA
e-mail: hnkolla@sandia.gov

N. Swaminathan and A. Parente (eds.), *Machine Learning and Its Application to Reacting Flows*, Lecture Notes in Energy 44, https://doi.org/10.1007/978-3-031-16248-0_3

simulations. The new framework – composed from *signature*, *measure*, and *decision* building blocks with well-defined semantics – is tailored for parallel and distributed computing, has bounded communication and storage requirements, is generalizable to a variety of applications, and operates in an unsupervised fashion. We demonstrate the efficacy of our framework on several cases spanning scientific domains and applications of event detection: optimized input/output (I/O) in computational fluid dynamics simulations, detecting events that can lead to irreversible climate changes in simulations of polar ice sheets, and identifying optimal space-time subregions for projection-based model reduction. Additionally, we demonstrate the scalability of our framework using a HPC combustion application on the Cori supercomputer at the National Energy Research Scientific Computing Center (NERSC).

1 Introduction

Scientific investigations – whether computational, experimental or observational – are ever expanding to include larger sets of coupled physics spanning broader ranges of scales, and the volumes of data generated from these investigations consistently outpace the growth of computational and data storage resources. As a consequence, specifically in the area of HPC modeling and simulation, the process of mining scientific data to glean insight is shifting from one of a posteriori to one of in situ analysis, i.e., analysis performed simultaneously with a simulation while sharing resources with it. Capturing events of interest to scientists in complex, high-fidelity HPC simulations is difficult because it is rarely feasible to export the entire simulation state at every timestep. Crucial stages in the development of events can be lost between checkpoints, and ephemeral events can be missed altogether, making a posteriori event detection problematic. Identifying events in situ is equally challenging, as traditional analysis algorithms that assume global access to data require excessive communication bandwidth.

Machine learning (ML) is being applied to scientific data for various purposes, including establishing constitutive laws, developing mathematically and statistically compact models of governing physics, identifying embedded patterns, dimensionality reduction, parameter importance and sensitivity analysis, and uncertainty quantification (UQ). In this work we focus on one specific application of ML: in situ *event detection*. Specifically, we seek to develop event detection algorithms that are:

- **Generalizable**: deployable in a variety of different scientific computing domains without the need for application-specific tuning;
- **Unsupervised**: able to operate without labeled examples defining events of interest;
- **Low Overhead**: requiring minimal communication between processors;
- **Online**: able to make predictions with minimal retention of data from prior timesteps.

To motivate the main contributions of this chapter, we first provide a brief overview of related past work.

1.1 Overview of Related Work

Event detection is related to anomaly detection, since the purpose of each is to detect behavior that is locally different. There has been substantial previous research on developing streaming anomaly detection algorithms for HPC simulation data. However, many of these algorithms require significant communication between processors. For example, Wu et al. (2014) proposed the Random Subspace Forest (RS-Forest) algorithm in which decision trees with random splits and random thresholds are used to construct a density estimate over the data observations in a continuous feature space. While this algorithm is very fast for local or shared memory applications, it is not communication efficient in this context because it requires sharing the entire RS-Forest data model across all processors. Similarly, Kernel Density Estimation (KDE) has been proposed for online anomaly detection (Ahmed 2009), but also requires significant communication between processors.

Some anomaly detection methods have been designed for parallel implementation with low communication overhead. Zhao et al. (2009) proposed a parallel framework for k-means clustering that could be adapted for anomaly detection. However, k-means clustering requires a user-defined number of clusters k, and performance is often strongly dependent on the selected value of this variable. Such sensitivity to algorithm parameters is undesirable for unsupervised in situ event detection.

Application-specific event detectors have also been developed. These include detectors to flag when ignition has occurred in combustion simulations (Bennett et al. 2016) and tropical cyclone trackers for climate simulations (Ullrich and Zarzycki 2016; Zhao et al. 2009). These algorithms make use of significant domain knowledge and are only applicable in the specific field for which they were developed, which is contrary to our goal of developing generalizable algorithms.

Ensemble anomaly detection techniques, such as iForest (Liu et al. 2012) and iNNE (Bandaragoda et al. 2014), are often considered to be robust and highly generalizable. Furthermore, these techniques have been shown to be compatible with data sub-sampling. The disadvantage of these methods is that they require communication to share the ensemble model between processors. For large ensembles this overhead can be prohibitively high.

Finally, it is not clear that conventional anomaly detection algorithms are well-suited for event detection in simulations. Because simulations often make use of highly refined meshes to resolve complex physical phenomena, an event of interest could occur over tens of thousands of mesh points, making it well-represented in the data, and therefore not anomalous. Moreover, comparisons to previous timesteps also are not straightforward, since many simulations exhibit significant drift over time: what is unusual at one timestep might become the norm later in time.

1.2 Contributions and Organization

We present herein a novel framework for applying ML to detect events of interest in situ in HPC simulation data. In this context, "events of interest" can be defined as any local dynamics in a region that differ significantly from the dynamics of other regions or timesteps. Our framework is tailored for parallel and distributed computing with the data typically representing a space-time domain of interest, with the spatial domain distributed across computing resources (processors/nodes) and data along the time dimension arriving in a streaming manner.

Consider a region handled by a single processor exhibiting behavior that differs significantly from the regions on other processors. Such a region could be considered interesting even if the behavior persists over multiple timesteps. An example of this type of event could be a tropical cyclone that persists over many timesteps in a weather simulation but is geographically localized. We refer to events of this type as *spatial* events of interest. Conversely, a sudden change across all processors from one timestep to the next could also be considered interesting. An example of this type of event could be simultaneous ignition across an entire domain in a combustion simulation. We refer to these as *temporal* events of interest.

This research presents a framework for developing in situ spatial and temporal event detection algorithms with tightly bounded communication and storage requirements, composed from *signature*, *measure*, and *decision* building blocks with well-defined semantics. The goal of this framework is to facilitate event detection in a computationally scalable and efficient manner, while allowing the flexibility to compose a learning workflow best suited for the scientific domain and problem at hand. The proposed framework can be used not only to optimize I/O within an HPC simulation (by flagging the locations where events of interest occur so that only a subset of the simulation state is stored to disk), but also to detect scientifically meaningful phenomena within HPC simulations and even to improve a simulation's accuracy/efficiency. A detected event can be used as a trigger for mesh and/or timestep refinement, e.g., Adaptive Mesh Refinement (AMR) (Berger and Oliger 1984).

The remainder of this chapter is organized as follows. The specific components of the proposed event detection framework are detailed in Sect. 2. In Sect. 3 we present results from three use cases that demonstrate the versatility and composability of the framework. The use cases span different scientific domains and different applications of event detection: optimized I/O in fluid flow simulations (Sect. 3.1), detecting events that are scientifically interesting in ice sheet simulations (Sect. 3.2), and identifying optimal space-time sub-regions for projection-based model reduction (Sect. 3.3). Section 3.4 presents results, using an exemplar turbulent combustion simulation, that demonstrate the scalability and computational efficiency of the framework when deployed in parallel computing simulations. Finally, conclusions are provided in Sect. 4.

2 Approach

Our framework for event detection is as follows. First, we assume a simulation domain with any number of dimensions. We further assume that the domain is divided into a set of P analysis partitions, where each analysis partition $p_i = 0, \ldots, P - 1$ is a spatially-contiguous subset of mesh points of the simulation domain. Each partition is always associated with a single processor so that analysis partitions never straddle processor boundaries or migrate from one processor to another throughout the simulation. Thus, a single processor will be responsible for one-to-many analysis partitions, with the size and number of partitions chosen based on the problem domain (Fig. 1).

Next, we execute the following workflow at each timestep of the running simulation. For each analysis partition p_i we compute a *signature* $\mathbf{s}_i$, a fixed-length vector representing the simulation state within that partition where $|\mathbf{s}_i| \ll |p_i|$ (Fig. 2). Conceptually, signatures are compressed, low-dimensional representations of an analysis partition's content, and our intent is that the signature should contain crucial aspects

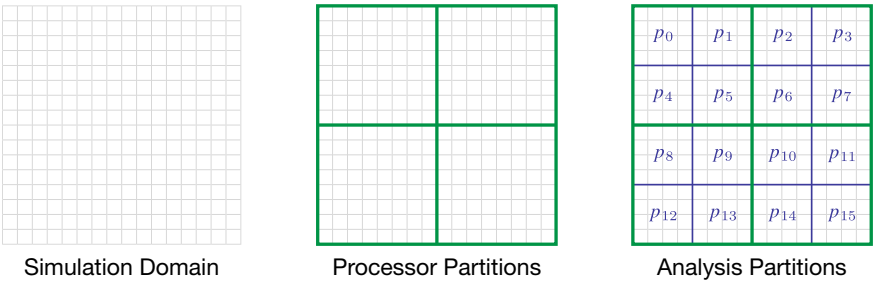


Fig. 1 Example simulation domain (gray), split across processors (green), and divided into analysis partitions (blue)

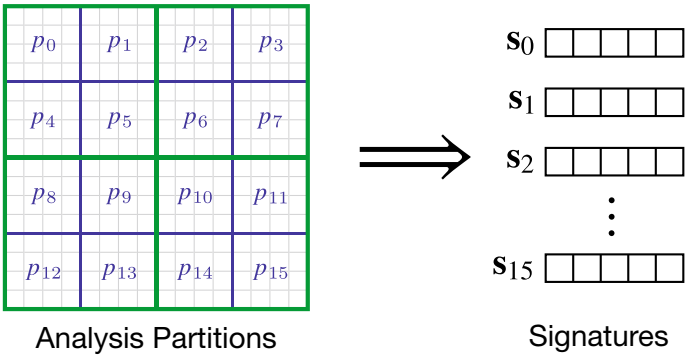


Fig. 2 Each analysis partition is represented by a low-dimensional signature

Table 1 Signature functions

Signature	Description
`fieda`	Vector of feature importance values described in Ling et al. (2017)
`fmm`	Vector based on the Feature Moment Metric algorithm in Konduri et al. (2018)
`mean`	Vector of mean values for each simulation feature
`minimax`	Vector of minimum and maximum values for each simulation feature
`quartile`	Vector of quartile boundaries for each simulation feature (a generalization of minimax)
`svd`	Vector of singular values computed using an SVD on the flattened partition state matrix

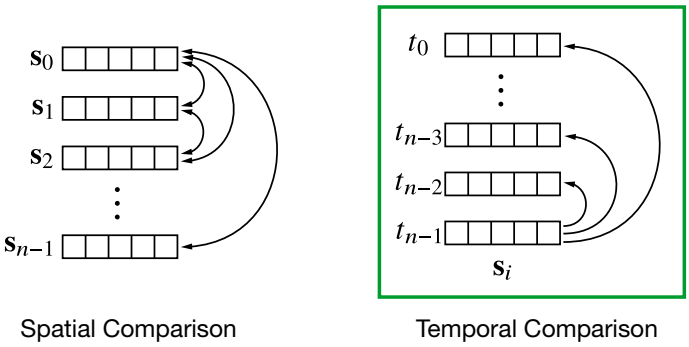


Fig. 3 Signatures can be compared all-to-all across analysis partitions to identify spatial events (left), and current signatures can be compared to previous signatures within partitions to identify temporal events (right)

of the state of the simulation within that partition, stored in such a way that changes across space or time can be detected by subsequent analysis of that representation.

As an example, for a simulation with state variables $F \in \mathbb{R}^n$, a signature could be vector of size $2|F|$ containing the minimum and maximum value for each variable $f \in F$ within the partition. Of course, this is only one possible signature type among many (we call this type *minimax*); we provide the subset of signature functions used in our experiments in Table 1. Note that, because analysis partitions are always associated with a single processor, computing signatures can be a purely local operation. Further, because signatures are a small, fixed size relative to the partitions they represent, they can be broadcast to other processors for spatial (partition-to-partition) comparisons and stored between timesteps for temporal (timestep-to-timestep) comparisons (Fig. 3). The user can choose, based on domain knowledge and the problem specifics, the set of features used to compute signatures. This set could consist of all of the state variables, a subset, derived variables, or any combination thereof; the only requirement is that the same set of features be used across all analysis partitions.

Given a set of signatures, we can compute spatial or temporal *measures* to identify events. Measures are functions applied to signatures that detect changes across space

Table 2 Spatial measures

Measure	Description
`dbscan`	Uses DBSCAN (Ester et al. 1996) to flag outlier signatures as events
`m1`	M1 metric described in Ling et al. (2017)
`m1-hellinger`	Modified version of the Hellinger distance used as a spatial metric, described in Konduri et al. (2018)
`msd`	Compares the distance between one signature and the mean signature for all partitions
`sigscal`	Normalizes the signature matrix using the product of the inverse of each signature and measures the ability of each signature to drive values in the product to zero

Table 3 Temporal measures

Measure	Description
`changefreq`	Counts the number of changes (dramatic increases or decreases across any two timesteps within a temporal window)
`maxchange`	Uses the maximum change across any timesteps within the current temporal window
`mse`	Based on mean squared error between the current temporal block ($S \times T_{\text{window}}$ matrix) and previous temporal block
`psd`	Estimates power spectral density (power spectrum) for each feature within the temporal block ($S \times T_{\text{window}}$ matrix) using Welch's method (Welch 1967)
`svd`	Ratio of the largest non-zero singular value to the smallest non-zero singular value

or time. Spatial measures compare signatures across analysis partitions to identify spatial events; typically, they compare the signature for a given partition to every other partition's signature, which requires communication. Temporal measures compare an analysis partition's current signature to its past signatures and are thus completely local, requiring only storage of a finite number of signatures from previous timesteps. In both cases, the output of the measure is a per-analysis-partition continuous scalar value indicating how interesting the partition's state is at the current timestep. We list representative spatial and temporal measures implemented for our experiments in Tables 2 and 3, respectively.

Finally, we use *decision* functions to convert continuous per-analysis-partition measures into boolean values to indicate whether the partitions should be flagged as containing events of interest for the current timestep. Decision functions are purely local, requiring no communication. Table 4 describes the decision functions that we used in our experiments.

We refer to a combination of signature, measure, and decision functions as an *algorithm* for in situ event detection; because we have many instances of each type, and they can be combined almost without exception, there are many possible algo-

Table 4 Decision functions

Decision	Description
`threshold`	Flag a partition when its measure exceeds a fixed threshold
`percentile`	Flag a partition when its measure exceeds the *n*th percentile of the measure for all partitions
`memory`	Decision modifier which continues to flag a partition for a fixed number of timesteps after another decision function has flagged it

rithms that can be created with just a few components (and the set of components continues to grow as we explore new ideas). The few incompatibilities tend to be driven by the expected inputs for a component. For example, it makes little sense to combine the *dbscan* measure with the *percentile* decision, since the former only produces binary values as output, and the latter is only useful with a continuous distribution as input.

3 Results

In this section we demonstrate our methodology on three important use cases for in situ machine learning: data capture for optimizing I/O (Sect. 3.1), detection of interesting physical events (Sect. 3.2), and facilitating reduced order model construction (Sect. 3.3). The use cases represent different scientific domains, but have similarities with reacting flows: Sect. 3.1 pertains to low speed non-reacting turbulent flows with passive tracers; Sect. 3.2 pertains to an incompressible fluid flow (glacier ice) solved using Stokes flow equations; Sect. 3.3 pertains to supersonic flow with shock. The purpose behind choosing such different use cases is to illustrate the generality of our detection algorithms.

3.1 Data Capture for Optimal I/O: Mantaflow Experiments

In our initial round of experiments, our focus is on testing the utility of our framework, and quantifying whether it could be used for meaningful reductions in I/O. We begin by creating a reference implementation using Python (2022), Numpy (Walt et al. 2011), Scipy (Jones et al. 2001) and Scikit-Learn (Pedregosa et al. 2011). To simplify development and support rapid iteration, these experiments use Mantaflow (Thuerey and Pfaff 2018) – an open source library targeting fluid simulation research in computer graphics and machine learning – for the simulation. Despite being a serial code, Mantaflow's Python scene definition interface makes it ideal for integration and rapid testing with our algorithms. All of our Mantaflow experiments are conducted using two-dimensional (2D) simulations for speed and ease of visualization.

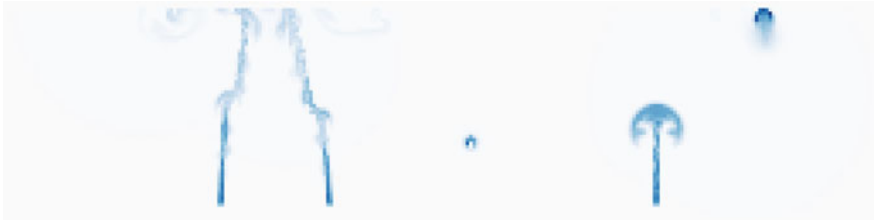


Fig. 4 Density field visualization from the *small plumes* Mantaflow simulation at one timestep. Darker colors signify higher density

To run the simulations, we created a driver script that loads an experiment definition file specifying the simulation setup, analysis partitions, simulation features to use for signature generation, as well as the signature, measure and decision functions to use for the analysis. Because the driver script also provides the simulation outer loop, it is trivial to run our analysis code alongside the simulation in situ.

We designed several Mantaflow simulations to test our event detection approach at different scales; for this chapter, we focus on our *small plumes* simulation, which has four state variables (density, pressure, x-velocity and y-velocity) and features three steady turbulent plumes of buoyant fluid using a 64×256 grid and running for 300 timesteps (Fig. 4).

Since the goal for our I/O use case is to minimize the amount of data saved to disk while simultaneously maximizing the number of events captured, a fundamental challenge is defining a sensible ground truth: for any given simulation, there is no well-defined way to specify which parts of the simulation should be considered events of interest (and thus flagged by our framework for subsequent storage to disk). To address this, we opted to create our own explicit ground truth by injecting random "depth charge" anomalies into the simulation. To do so, we generate a random value for each simulation cell at each timestep. At any cell where the random value exceeds a threshold, the simulation density is increased by a substantial amount, and the cell is marked as anomalous using an additional simulation state variable. Thus, the depth charge anomalies occur at random timesteps and locations within the simulation domain, and the anomalies state variable keeps track of where they occur (Fig. 5). The

Fig. 5 Per-cell ground truth for the *small plumes* simulation, at the same timestep as Fig. 4. The dark cells are anomalies, intentionally introduced by our "depth charges"

overall impact is to introduce physically-implausible aberrations into the simulation which surely qualify as events worthy of detection. Having created the anomalies ourselves, we can then evaluate the algorithm's ability to flag them as events of interest. Note that, even with our explicitly injected anomalies, there is still ambiguity surrounding the question of which cells/partitions should be flagged as events: while the sudden onset of an anomaly is obviously an event worth noting, the threshold at which it should cease to be anomalous as it disperses is still arbitrary. Despite these shortcomings, our "depth charges" provide a quantitative way to compare performance among different algorithms tested using the framework.

The behavior of our driver script is as follows. First, at each timestep, we use the Mantaflow API to run the solver for that step. Next, we extract the simulation state (density, pressure, velocities and anomaly ground truth) and save the raw data to disk. We then divide the simulation grid into 8×8 analysis partitions, since our framework requires multiple analysis partitions even when there is a single processor, as is the case for the serial Mantaflow simulations. Next, we compute the per-partition signatures. To support computing temporal measures and because the Mantaflow simulations are so small, we store every signature computed at every timestep, though we assume in practice that an HPC simulation would retain a smaller number of the most recent signatures. The set of per-analysis-partition signatures are then passed to the measure function to generate per-partition measures. Since the measure function has access to the signatures for every partition and every timestep, it can calculate a measure based on a comparison of signatures across every analysis partition (a spatial measure), a comparison of signatures across time for a single partition (a temporal measure), or a hybrid of the two. Because our Mantaflow experiments run on a single process, no communication is necessary, unlike the HPC experiments described in Sect. 3.4. We save the measures computed for each partition to disk for subsequent visualization. Finally, the measure values are passed to the decision function to be flagged as *events* or not, and those decisions are written to disk.

Once the simulation is complete, we convert the simulation features, anomalies, measures and decisions stored on disk to color-mapped images, generating movies using the open source Imagecat (2022) library for compositing and Ffmpeg (2019) for encoding. The simulation movies provide a qualitative way to evaluate algorithm behaviors (Fig. 6).

For quantitative comparisons, we used the decision data to generate several metrics, including: (1) the percentage of simulation domain cells that are flagged as events by our framework, both per-timestep and for the simulation as a whole, and (2) the percentage of ground truth anomalous cells that are contained within partitions flagged as events, per-timestep and for the simulation as a whole. We refer to this latter metric as "recall".

Our early experiments were focused on identifying useful combinations of signature-measure-decision building blocks and developing intuition around their strengths and weaknesses. In this preliminary exploration, the percentage of simulation cells flagged as events ranges from 4.3% (excellent, a twenty-fold decrease in storage requirements) to 75% (likely not worth the effort), while our recall metric ranges from 35.4% (good) to 99.8% (excellent). One combination that produces con-

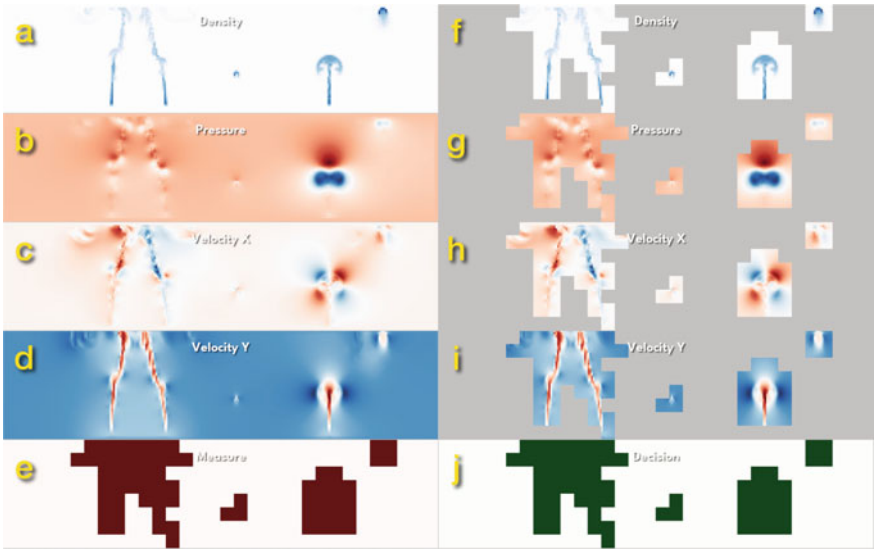


Fig. 6 Sample frame from a Mantaflow experiment movie: simulation state (**a**)–(**d**), per-analysis-partition measure (**e**) and decisions (**j**), simulation state masked by decisions (**f**)–(**i**)

sistently good results for a wide range of parameters used is the *quartile* signature, *dbscan* spatial measure with Euclidean distance, and *threshold* decision function. Figure 7 plots the total percentage of flagged analysis partitions (lower is better) versus the anomaly recall (higher is better) for a set of experiments using this combination. The result is intentionally evocative of a receiver operating characteristic curve, emphasizing the trade-offs inherent in our desire to maximize the number of detected events while minimizing the total number of analysis partitions flagged for storage to disk.

The *dbscan* measure used in these experiments has two main parameters: ε, the threshold distance below which two signatures are considered "neighbors"; and N_p, the minimum number of neighboring signatures required to form a "neighborhood." Once all of the neighborhoods in a collection of signatures are identified, any signatures not in a neighborhood are, by definition, flagged as interesting events.

We tested combinations of ε and N_p using grid search, varying ε values between 0.1 and 1.0 and N_p values between 1% and 50% of the total number of analysis partitions. At very low values of ε, we rapidly achieved high recall, approaching 100%. Values over 0.3 led to a rapid reduction in recall, dropping to around 8% for an ε of 1.0. Varying N_p had much less effect, with most values below 40% having little effect on recall. We are encouraged that many parameter combinations produce results near the knee of the curve in Fig. 7, indicating that the algorithm is robust for a wide range of reasonable DBSCAN parameters. We chose $\varepsilon = 0.2$ and $N_p = 2\%$ as the best parameters for this data, with results shown in Fig. 8.

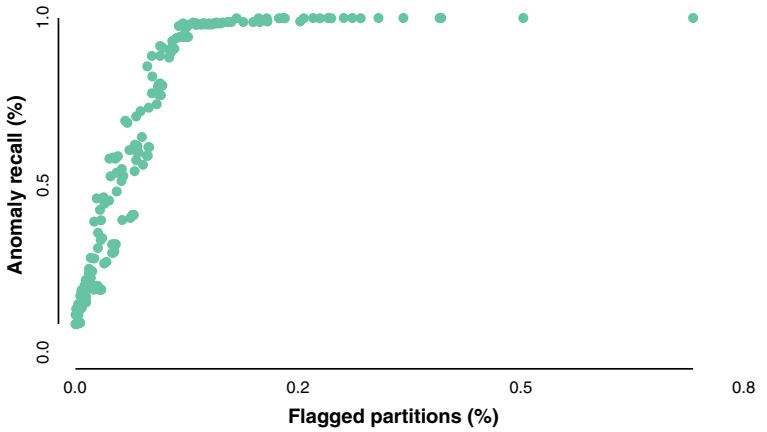


Fig. 7 Flagged analysis partitions versus anomaly recall for the *quartile-dbscan* Mantaflow experiments

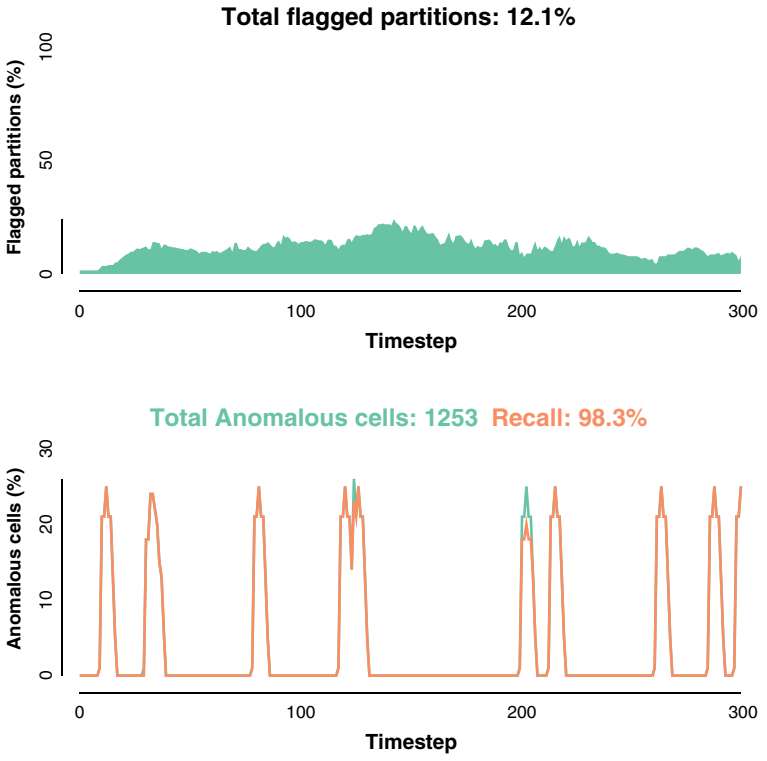


Fig. 8 Flagged analysis partitions (top) versus recall (bottom) for *quartile-dbscan* Mantaflow experiment with $\varepsilon = 0.2$ and $N_p = 2\%$

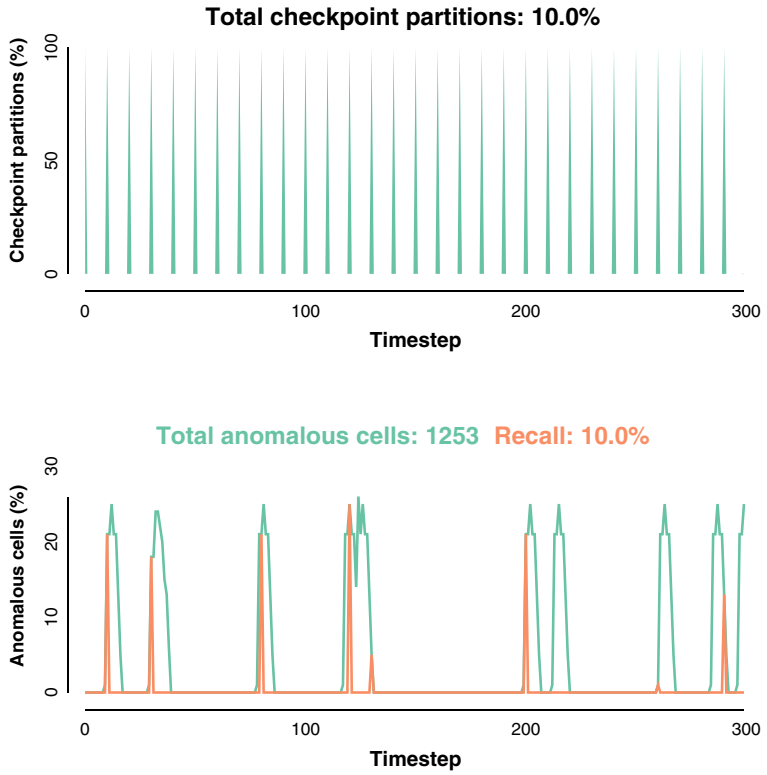


Fig. 9 Saved analysis partitions (top) versus recall (bottom) for a simulation saving a checkpoint at every tenth timestep for the Mantaflow experiment

In this case, an experimenter using the *quartile-dbscan* algorithm to decide which analysis partitions should be saved to disk would end up capturing 98.3% of the anomalies, while storing just 12.1% of the data. This is especially striking when we compare it to typical uniform temporal check-pointing of HPC simulation data: the experimenter who simply saves the entire simulation state at every tenth timestep as in Fig. 9 would use roughly the same amount of disk space (10% vs. 12.1%), while only capturing 10% of the interesting events!

We performed temporal anomaly detection experiments using similar techniques. One comparable result used the *minimax* signature, the *maxchange* measure, and the *threshold* decision function, producing a recall of 96.3% while flagging only 24% of the data.

3.2 *Detecting Physical Phenomena: Marine Ice Sheet Instability (MISI)*

While the in situ event detection framework described herein was originally developed for the purpose of optimizing HPC simulation output, the proposed approach can also be used to detect physical phenomena present in HPC simulation data to further our understanding of the underlying physical processes. Here, we describe a specific instance of this use case, in which our framework facilitates the study of the hypothesized Marine Ice Sheet Instability (MISI) using simulation data from the MPAS-Albany Land Ice (MALI) model (Hoffman et al. 2018), the land ice component of the U.S. Department of Energy's Energy Exascale Earth System Model (E3SM) (Leung et al. 2020).

The Marine Ice Sheet Instability, first introduced in the 1970s (Weertman et al. 1974; Thomas and Bentley 1978), hypothesizes that ice sheets grounded below sea-level may destabilize in a runaway fashion once the grounding line, the boundary between where the ice sheet is grounded and floating, reaches a point where the bedrock has a reverse slope gradient (Fig. 10) (Bamber et al. 2009). Once the bedrock beneath the grounding line is reverse sloping (i.e., it becomes deeper moving inland), ice thickness at the grounding line increases, leading to faster ice flow and greater ice flux divergence. As the flux at the grounding line increases, thinning at and upstream of the grounding line increases, causing the boundary between floating and grounded ice to move further inland. The result is a self-reinforcing mechanism that can cause rapid and irreversible ice sheet retreat and rapid sea level rise (Robel et al. 2019; Joughin and Alley 2019). Since the grounding line is often stabilized by the presence of an ice shelf (an extended region of floating ice that is dynamically connected to the grounded ice upstream of it), which has the effect of buttressing the ice and limiting ice flux at the grounding line, MISI is often triggered by the thinning or loss of ice shelves (Pattyn and Morlighem 2020). Satellite and modeling evidence suggests that MISI is underway in parts of the West (e.g., the Thwaites and Pine Island glacier) and East (e.g., the Totten glacier) Antarctic Ice Sheet (Robel et al. 2019; Joughin

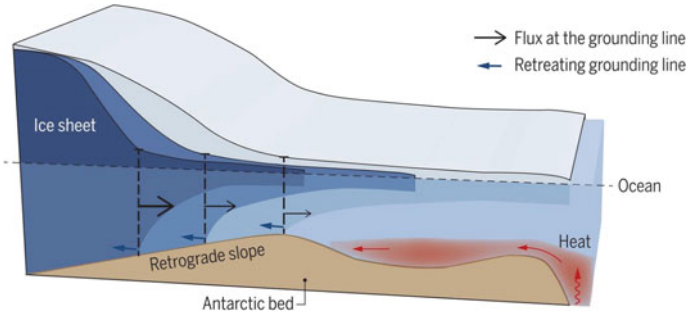


Fig. 10 Marine Ice Sheet Instability triggered by an unstable grounding line retreat on retrograde bedrock slope. Figure adapted from Pattyn and Morlighem (2020)

and Alley 2019; Gardner et al. 2018; Young et al. 2011). While it is theoretically possible to identify locations prone to MISI by combining bedrock elevation data with information on retrograde bedrock slopes, this approach is not feasible since bedrock elevation data are limited. Moreover, the retrograde bed slope alone is likely not a sufficient proxy for MISI, as it does not take into account important features relevant to MISI, e.g., ice flow speed and ice flux.

Our approach herein is to investigate the utility of our event detection framework in identifying the onset of MISI. Accordingly, we applied our event detection algorithms to two simulations datasets: (1) an idealized Antarctic BUttressing Model Intercomparison Project simulation (ABUMIP) (Sun et al. 2020), and (2) a predictive simulation of the Antarctic Ice Sheet with realistic climate forcing (Seroussi et al. 2020). Following the naming convention introduced in Sun et al. (2020) and Seroussi et al. (2020), respectively, we refer to these datasets as `abuk` and `exp05`, respectively. Both simulations start with a realistic present-day initial condition obtained by performing an adjoint-based optimization using the MALI model (Perego et al. 2014). They then simulate ice flow over Antarctica on a variable-resolution three-dimensional (3D) tetrahedral grid. The output from these simulations is subsequently mapped onto a 2D structured quadrilateral grid having a uniform resolution of 8 km (Fig. 11), for the purposes of analysis and comparison to other land ice models (Seroussi et al. 2020). In the `abuk` experiment, Antarctica's ice shelves are removed instantaneously, and we perform a simulation in which the formation of new floating ice is prevented and no change in external atmospheric or oceanic forcing is applied. Although unrealistic, this scenario provides an extreme upper bound on sea-level contributions from Antarctica, and exhibits the full potential of MISI (Sun et al. 2020). As such, the `abuk` dataset is ideal for "calibrating" (i.e., determining a reasonable set of features and analysis partition sizes) and "validating" (i.e., ensuring that reasonable analysis partitions are flagged as interesting) our event detection framework before applying it to the more realistic `exp05` scenario. The second experiment, `exp05`, is a standard test case in the ISMIP6 (Ice Sheet Model Intercomparison Project 6) experiments (Seroussi et al. 2020), and is meant to be a realistic predictive simulation of the Antarctic Ice Sheet state with atmospheric and oceanic forcing[1] under the RCP8.5 (Representative Concentration Pathway 8.5) (IPCC 2021) radiative forcing emissions scenario, which corresponds to the likely outcome if society does not make concerted efforts to cut greenhouse emissions during the remainder of the twenty-first century (Edwards et al. 2021). For initial prototyping, our event detection algorithms are applied to the datasets a posteriori; integration of these algorithms into the MALI code for true in situ analyses will be the subject of future work. For the `abuk` dataset, there are 51 solution snapshots, corresponding to a 500 year simulation, with data saved every 10 years; for `exp05`, there are 86 solution snapshots, corresponding to an 85 year simulation, with data saved every year.

Prior to presenting our main results, we discuss some nuances pertaining to the generation of analysis partitions for the land ice datasets considered herein. For both the `abuk` and `exp05` datasets, the underlying computational domain onto which the

[1] For details regarding these forcings, the reader is referred to Table 2 of Seroussi et al. (2020).

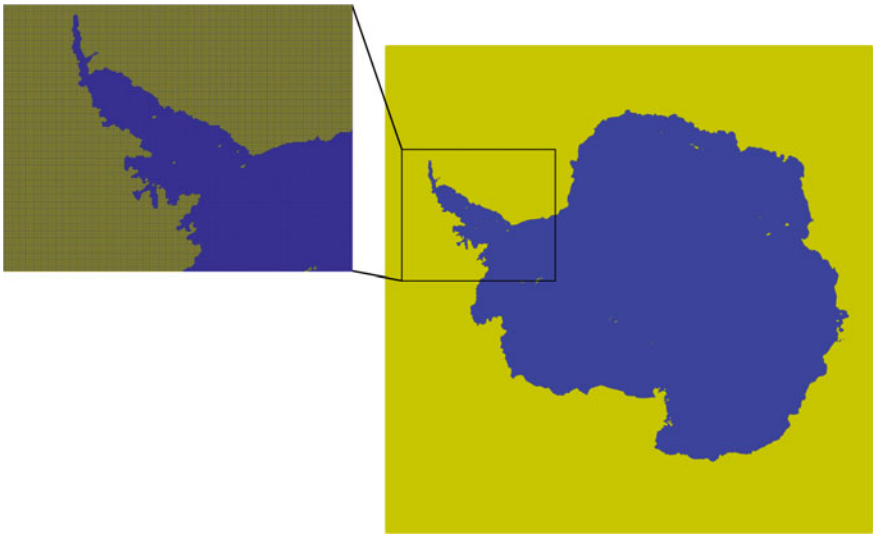
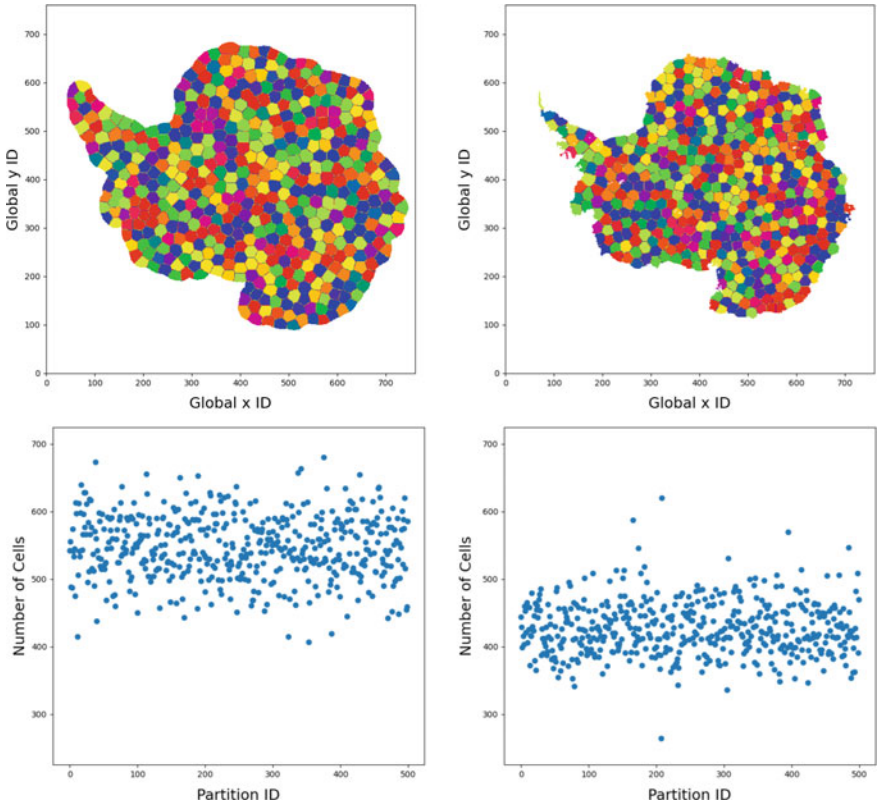


Fig. 11 "Full" 6088 km × 6088 km domain for the `exp05` dataset, with active cells shown in blue. Left panel shows a close-up of the Antarctic peninsula and the structured 8 km quadrilateral mesh with which the problem is discretized

MALI output is mapped is a 6088 km × 6088 km square grid, discretized using 761 quadrilateral elements in each coordinate direction (Fig. 11). To determine which cells within this computational domain are "active" (ice-covered), a time-dependent mask derived from the ice thickness was computed at each timestep based on an ice thickness criterion: only cells in which the ice thickness is greater than 10 m are deemed "active" in each timestep. An important feature of masks derived in this way is that the masks, and hence the geometries on which the simulation proceeds, change in time: before solving for the ice sheet state at each time-step, inactive cells are removed from the mesh on which the simulation proceeds. While it would be possible to uniformly partition the "full" 761×761 element grid into P analysis partitions to use for our event detection workflow, such an approach would lead to an imbalanced set of partitions, in which many partitions would have few (or even zero) elements. Using an analysis partition set of this type could bias the event detection, especially when statistics-based signatures are employed. One approach to avoid this problem is to partition only the active grid, but this second approach also has several downsides: (1) its computational cost would likely preclude in situ analyses, and (2) with analysis partitions that change in time, it is not clear how to track temporal events using this methodology. To avoid these issues, we adopted a third approach, in which we created a mask (termed the "analysis partitioning mask") that was only slightly larger than the maximum ice extent across all simulation times for a given dataset, and created a single partition of the geometry defined by this mask prior to performing event detection. In the present study, we consider two types of analysis partitioning masks:



(a) abuk experiment (b) exp05 experiment

Fig. 12 Illustration of 500 analysis partitions (top panel) obtained using k-means clustering, and cell-counts for each analysis partition (bottom panel) for an active mesh with buffer (**a**) and the union of active meshes (**b**) analysis partitioning mask. The latter analysis partitioning mask was used for the `abuk` experiment, and the former was used for the `exp05` experiment. Different colors in the top panel represent distinct partitions

- *Active mesh with buffer:* a mask in which a buffer region is included around the maximum footprint of the underlying Antarctic geometry (Fig. 12a) is created;
- *Union of active meshes:* a mask is created by performing a union of the active cells across all simulation times (Fig. 12b).

Each approach to analysis partitioning mask creation has its pros and cons. The former approach is amenable to in situ analyses, but is likely to give rise to some analysis partitions with little to no elements. The latter approach minimizes the likelihood of empty/imbalanced analysis partitions, but would not be possible to generate in situ. Our preliminary numerical results, described below, suggest that both approaches to creating the analysis partitioning mask produce reasonable results for the datasets considered.

Having settled on an approach for dealing with the temporal variability of the active mesh in our land ice datasets, we now discuss the choice of partitioning scheme for generating the analysis partitions required by our event detection algorithm. We explored the use of several partitioning algorithms, including space-filling curve partitioning (e.g., Hilbert, Morton) (Sasidharan et al. 2015), quad-tree partitioning (Ansar et al. 2019), and k-means clustering (Hartigan and Wong 1979). Of these three approaches, k-means clustering produced the most balanced analysis partitions, shown in Fig. 12. These partitions are balanced in the sense that each partition has roughly the same number of cells, with the partition size appearing to be normally distributed around the target number of cells per partition. Our results below utilize the k-means partitioning algorithm implemented within Scikit-Learn (Pedregosa et al. 2011), seeded with a random initialization. The reader can observe by examining the bottom panel of Fig. 12 that this partitioning scheme produces a partitioning with fairly balanced cell counts per analysis partition. Applying the space-filling curve and quad-tree partitioning approaches to our datasets in contrast gives rise to partition sizes ranging from a single cell to the maximum number of cells/partition requested (partitions not shown). As mentioned earlier, having analysis partitions of widely disparate sizes is particularly problematic for statistics-based signatures within our framework, since these signatures are highly dependent on the number of cells per partition.

As discussed in Sun et al. (2020) and Seroussi et al. (2020), the `abuk` and `exp05` datasets contain a number of fields that can be used as features in our event detection workflow. In the preliminary study presented here, we considered the following four solution fields as features, denoted by $\mathcal{F}_i$ for $i = 1, \ldots, 4$:

- $\mathcal{F}_1$: the ice sheet thickness,
- $\mathcal{F}_2$: the norm of the ice velocity at the ice surface,
- $\mathcal{F}_3$: the norm of the ice velocity at the ice base,
- $\mathcal{F}_4$: the norm of the ice velocity averaged over the vertical extent of the ice.

The ice sheet thickness is selected as a feature because it is a function of the bedrock geometry/topography; the ice velocity fields are used as features as fast-moving ice may correlate with the presence of MISI. In addition to employing the raw solution fields $\mathcal{F}_i$ in our analysis, we also considered logarithms of these fields, denoted by $\log(\mathcal{F}_i)$. We employed the *quartile* signature (Table 1), the *dbscan* measure with parameters $\varepsilon = 0.3$ and $N_p = 5\%$ (Ester et al. 1996) (see Table 2 and Sect. 3.1 for a discussion of this measure and parameters) and the *threshold* decision (Table 4). In this initial proof-of-concept study, only spatial events of interest were considered. The threshold decision flagged partitions with a measure less than zero. The k-means clustering algorithm was used to generate 14,000 partitions, each having approximately 16 cells, for both the `abuk` and `exp05` experiments. For the `abuk` dataset, we partitioned the active mesh with a buffer region around it (Fig. 12a), whereas for the `exp05` dataset, we partitioned an active mesh consisting of the union of all active meshes during the simulation (Fig. 12b).

Our main results are shown below, in Figs. 13, 14, 15 and 16, which plot the interesting analysis partitions in green, overlaying the ice thickness field feature used

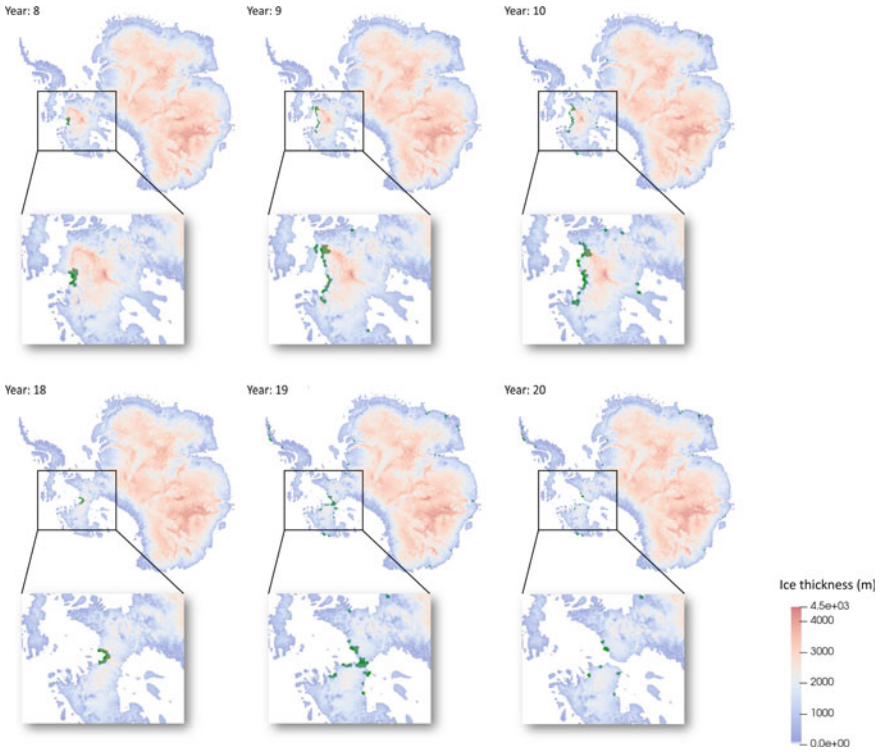


Fig. 13 Event detection results for `abuk` experiment with the four raw fields $\mathcal{F}_i$ for $i = 1, \ldots, 4$ as features. Analysis partitions identified as interesting are plotted in green, overlaying the ice thickness field for several years. Our results show that ice contained within analysis partitions identified as interesting in one timestep will in general melt (become inactive) in the following timestep

in the analysis. We emphasize that these results are preliminary and intended only to demonstrate the potential usefulness of the proposed framework in data-driven studies of land ice; scientific studies using our event detection framework will be the subject of future research.

3.2.1 Results for the `abuk` Experiment

We first apply our event detection framework to the `abuk` dataset, as this dataset is most likely to contain evidence of MISI. Figure 13 shows snapshots of the solution for the `abuk` dataset at several times, with a close-up in the vicinity of the Pine Island and Thwaites glaciers. Analysis partitions identified as interesting using our algorithm when employing the full set of fields $\{\mathcal{F}_i\}$ for $i = 1, \ldots, 4$ as features are plotted in green, overlaying the ice thickness field for several years. The reader can observe by inspecting this figure that cells comprising the analysis partitions

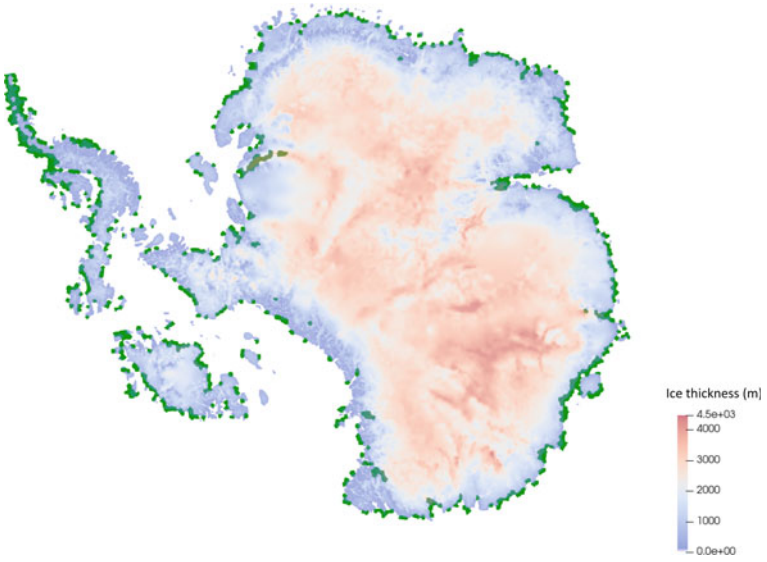


Fig. 14 Event detection results for the `abuk` experiment with $\mathcal{F}_1$ and $\log(\mathcal{F}_4)$ as features. Analysis partitions identified as interesting are plotted in green, overlaying the ice thickness field for year 33

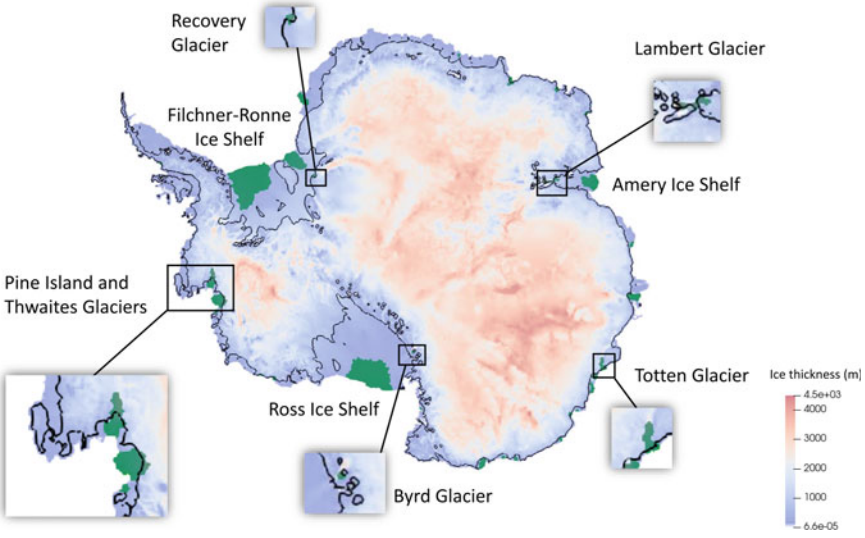


Fig. 15 Event detection results for `exp05` experiment with the four raw fields $\mathcal{F}_i$ for $i = 1, \ldots, 4$ as features. Analysis partitions identified as interesting are plotted in green, overlaying the ice thickness field for year 33. The grounding line is shown with a black contour. Our event detection framework identifies the fastest moving areas along Antarctica's coast (ice shelves, outlet glaciers), where MISI is more likely to initiate

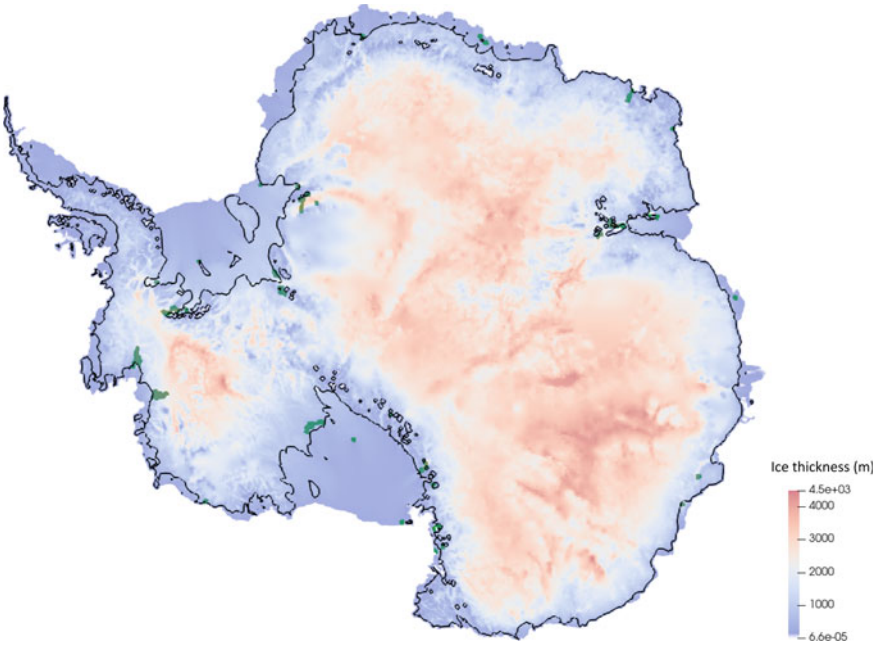


Fig. 16 Event detection results for the `exp05` experiment with $\mathcal{F}_1$ and $\log(\mathcal{F}_4)$ as features. Analysis partitions identified as interesting are plotted in green, overlaying the ice thickness field for year 33. The grounding line is shown with a black contour

identified as interesting in one timestep subsequently become inactive (based on previously-described active mask criterion) in the following timestep. In other words, the ice that is flagged by our algorithm melts shortly after it is flagged, a behavior consistent with MISI.

Next, we perform event detection using a reduced set of features, namely $\mathcal{F}_1$ and $\log(\mathcal{F}_4)$. Figure 14 plots the anomalies identified by our framework in year 33 of the simulation, again in green and overlaying the norm of the ice thickness field. It is interesting to observe that significantly more interesting partitions are identified with the new set of features. This is not surprising, as applying a logarithm transform of an analysis feature when using the *dbscan* measure has the effect of emphasizing small differences in small-magnitude values. An additional noteworthy observation is that, with the new set of features, not all of the interesting analysis partitions identified by our algorithm are at or near the grounding line. In particular, several of the flagged locations are located a large distance inland. These locations appear to be regions where the ice retreats the fastest, and should be inspected further in search of MISI.

3.2.2 Results for the `exp05` Experiment

Having obtained plausible results for the `abuk` experiment, we now turn our attention to the more realistic `exp05` case. Figure 15 plots results for the `exp05` dataset corresponding to year 33 in the simulation, with analysis partitions identified as interesting plotted in green and the grounding line (the boundary between where the ice sheet is grounded and floating) plotted with a black contour. From this figure, one can see that our event detection framework identifies the fastest moving areas along Antarctica's coast (the ice shelves and outlet glaciers) as events. These are locations where MISI is more likely to originate. In particular, the following glaciers are identified as containing events of interest: Pine Island, Thwaites, Totten, Byrd, Recovery and Lambert (see Fig. 15). Observational evidence suggests MISI is underway at Thwaites, Pine Island and Totten glaciers (Robel et al. 2019; Joughin and Alley 2019; Gardner et al. 2018; Young et al. 2011). The other regions identified as interesting by our framework are worth taking a closer look at – in both model simulations and observational datasets – in search of MISI (Hoffman et al. 2022).

The most intriguing result is shown in Fig. 16, which plots the interesting analysis partitions for the `exp05` dataset with the new set of features, again in green. The reader can observe that our algorithm flags several regions located inland relative to the grounding line (shown by a black contour). Additionally, the analysis partitions identified as interesting on and near Antarctica's ice shelves closely match the locations that have a significant impact on grounding line flux identified by Reese et al. (2018). While a more rigorous study is required for validating this result, the fact that there is corroboration with previously published results appears promising. A more rigorous investigation, towards understanding the physical mechanisms driving the events identified by our framework, will be the subject of future work. Future work will also explore the use of alternate features in the event detection workflow (including lateral buttressing in shear zones, basal friction, and flux fields, such as the ice velocity flux divergence), as well as alternate signatures and measures, including temporal measures (Table 3). We additionally plan to apply our methodology to higher-resolution datasets (e.g., 3D unstructured datasets produced by running the MALI model/code (Hoffman et al. 2018)) and to land ice datasets expected to exhibit stochastic behavior, e.g., simulations that include parameterizations of physical processes for ice calving and subglacial hydrology.

Finally, it is worth remarking that interesting events or anomalous behaviors identified in land ice simulations using the proposed framework could be relevant for scientists even if they are not an indication of MISI. In this context, an analysis partition flagged by our framework could be indicative of something incorrect in the data or underlying land ice model (e.g., a software flaw or missing physics), or of interesting physical phenomena other than MISI.

3.3 *Reduced Order Modeling: Sample Mesh Generation for Hyper-Reduction*

To highlight the breadth of application spaces that can benefit from the proposed event detection algorithms, we discuss a fundamentally new use case for our framework within the field of projection-based model reduction.

Projection-based reduced order modeling is a promising strategy for reducing the computational cost of high-fidelity HPC simulations, which are often too expensive for use in a design or analysis setting (e.g., optimization, UQ). Reduced order models (ROMs) have two key features: they are constructed to retain the essential physics and dynamics of their corresponding full order models (FOMs) and they incur a substantially lower (in some cases by orders of magnitude) computational cost. In projection-based model reduction, the state variables are approximated within a low-dimensional subspace, which is typically obtained offline by first applying data compression on a set of snapshots collected from a high-fidelity simulation or physical experiment. A typical projection-based ROM workflow consists of three steps, depicted in Fig. 17 and described succinctly below. In this figure, and the discussion that follows, it is assumed that the FOM is given by the following nonlinear ordinary differential equation (ODE):

$$\frac{d\boldsymbol{w}}{dt} = \boldsymbol{f}(\boldsymbol{w}; t, \boldsymbol{\mu}), \tag{1}$$

where $\boldsymbol{w}$ denotes the solution vector t denotes time, $\boldsymbol{\mu}$ is a vector of parameters Note that (1) is very generic: an ODE of the form (1) is obtained, for example, by semi-discretizing the partial differential equations (PDEs) defining the FOM in space using a numerical method, such as the finite element or the finite volume method.

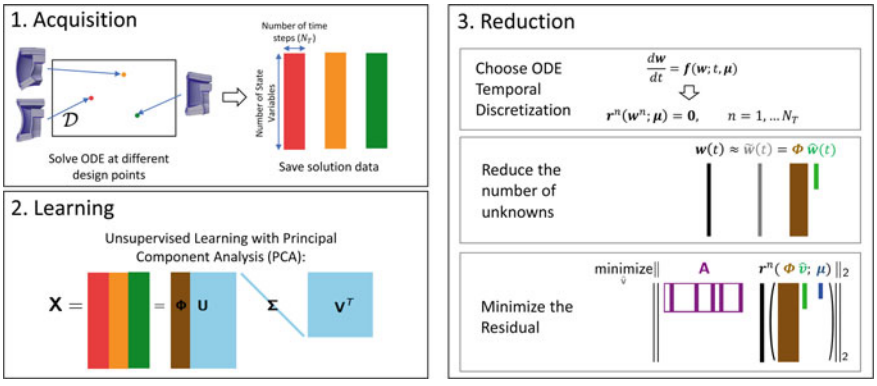


Fig. 17 Illustration of a projection-based model reduction workflow using the POD/LSPG method with hyper-reduction of a full-order model given by the ODE $\frac{d\boldsymbol{w}}{dt} = \boldsymbol{f}(\boldsymbol{w}; t, \boldsymbol{\mu})$. In this figure, $(\cdot)$ denotes "function of" rather than multiplication. The matrices and vectors appearing in this figure have the following dimensions: $\mathbf{X} \in \mathbb{R}^{N \times K}$; $\boldsymbol{\Phi} \in \mathbb{R}^{N \times M}$; $\boldsymbol{w}, \tilde{\boldsymbol{w}}, \boldsymbol{f}, \boldsymbol{r}^n \in \mathbb{R}^N$; $\hat{\boldsymbol{w}}\hat{\boldsymbol{v}} \in \mathbb{R}^M$; $\mathbf{A} \in \mathbb{R}^{q \times N}$; $\boldsymbol{\mu} \in \mathbb{R}^L$, where $L \in \mathbb{N}$ is the number of parameters

Step 1. Acquisition of high-fidelity snapshot data. The first step in a typical projection-based model reduction workflow is the acquisition of a set of K instantaneous snapshots of a numerical solution field. Typically snapshots are collected for K values of a parameter of interest (see Fig. 17), at K different times, or both.

Step 2. Learning a reduced basis. Given an ensemble of high-fidelity snapshots denoted by $\{\boldsymbol{w}^n\}_{n=1}^K$, the next step is the calculation of a basis of reduced dimension $M \ll N$, where N denotes the number of degrees of freedom (dofs) in the FOM. There are numerous approaches in the literature for computing a low-dimensional subspace, but we restrict the discussion here to the Proper Orthogonal Decomposition (POD) method (Sirovich 1987; Holmes et al. 1996) for calculating reduced bases, due to its simplicity and prevalence in practice. Mathematically, POD is closely related to Principal Component Analysis (PCA), and seeks an M-dimensional subspace (with $M \ll K$) spanned by a set of modes $\{\boldsymbol{\phi}_i\}_{i=1}^M$ such that the difference between the snapshot ensemble $\{\boldsymbol{w}^n\}_{n=1}^K$ and the projection of this ensemble onto the reduced subspace is minimized on average. It is a well-known result that the solution to the POD optimization problem reduces to a singular value decomposition problem involving the snapshot matrix $\mathbf{X}$, as shown in Fig. 17; specifically, the modes $\{\boldsymbol{\phi}_i\}_{i=1}^M$ are the M left singular vectors corresponding to the M largest singular values of $\mathbf{X}$. The interested reader is referred to Holmes et al. (1996), Kunisch and Volkwein (2002), Rathinam and Petzold (2003) for details.

Step 3. Projection-based reduction. The final step is the actual reduction, obtained by projecting the equations defining the FOM onto the reduced basis, denoted by $\boldsymbol{\Phi} := [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_M] \in \mathbb{R}^{N \times M}$. Common projection methods are Galerkin projection and Least-Squares Petrov-Galerkin (LSPG) projection; herein, we focus on the latter approach, as it has been shown to exhibit better stability properties, especially for fluid systems (Carlberg et al. 2017). This approach operates on a FOM that has been fully discretized in both space and time, which can be written as:

$$\boldsymbol{r}^n(\boldsymbol{w}^n; \boldsymbol{\mu}) = \mathbf{0}, \tag{2}$$

where $\boldsymbol{r}$ denotes the residual, and the super-script n denotes the time index, with $n = 1, \ldots, N_T$, so that $\boldsymbol{w}^n := \boldsymbol{w}(t^n)$, where t^n is the nth timestep within a simulation based on (2). The high-fidelity solution $\boldsymbol{w}(t)$ is approximated as a linear combination of the reduced basis modes:

$$\boldsymbol{w}(t) \approx \boldsymbol{w}_M(t) = \boldsymbol{\Phi}\hat{\boldsymbol{w}}(t), \tag{3}$$

where $\hat{\boldsymbol{w}}(t) \in \mathbb{R}^M$, with $M \ll N$. Given this definition, in the LSPG approach, solving for the ROM solution amounts to solving the following least-squares optimization problem:

$$\hat{\boldsymbol{w}}^n = \arg\min_{\boldsymbol{y} \in \mathbb{R}^M} ||\boldsymbol{r}^n(\boldsymbol{\Phi}\boldsymbol{y}; \boldsymbol{\mu})||_2^2, \tag{4}$$

for $n = 1, \ldots, N_T$ and $\hat{\boldsymbol{w}}^n := \hat{\boldsymbol{w}}(t^n)$. Equation (4) can be solved using the Gauss-Newton approach following the method of Carlberg et al. (2013). Unfortunately, the approach described thus far is inefficient for nonlinear problems, as the solution of the ROM problem (4) requires algebraic operations that scale with N, the dimension of the original FOM. This problem can be circumvented through the use of hyper-reduction, the basic idea of which is to compute the residual at some small number of points q with $q \ll N$, encapsulated in a "sampling matrix" $\mathbf{A}$ computed as a pre-processing step of the model reduction procedure using available snapshot data. The set of q points is typically referred to as the "sample mesh", and a variety of quasi-optimal approaches aimed to minimize the representation error of a given nonlinear function appearing in the FOM residual exist—examples include the (discrete) empirical interpolation method (D)EIM (Barrault et al. 2004; Chaturantabut and Sorensen 2010), "best points" interpolation (Nguyen et al. 2008; Nguyen and Peraire 2008), collocation (LeGresley 2006), gappy POD (Everson and Sirovich 1995), and p–sampling (Drmac and Gugercin 2016). These approaches approximate the solution to the NP-hard optimization problem of minimizing the representation of a nonlinear residual using different greedy approaches. Typically, as one may expect based on intuition, the sample mesh points returned by these algorithms are clustered in regions where the simulated solution exhibits "interesting" behavior/features, e.g., shocks, vortices, etc. (see e.g., Fig. 18). With the introduction of hyper-reduction, the LSPG optimization problem takes the form

$$\hat{\boldsymbol{w}}^n = \arg\min_{\boldsymbol{v}\in\mathbb{R}^M} ||\mathbf{A}\boldsymbol{r}^n(\boldsymbol{\Phi}\boldsymbol{y};\boldsymbol{\mu})||_2^2. \tag{5}$$

As illustrated in Fig. 17, the matrix $\mathbf{A} \in \mathbb{R}^{q\times N}$ is sparse, and has the effect of "sub-selecting" the residual $\boldsymbol{r}$ at some small number of points q, corresponding to the non-zero columns of $\mathbf{A}$.

Current state-of-the-art methods employ a *single static* sample mesh computed offline, and use the *same* sample mesh for hyper-reduction for all the timesteps at which the ROM solution is computed. It has been observed that, for certain applications, sample meshes computed using standard hyper-reduction methods (gappy POD (Everson and Sirovich 1995), p–sampling (Drmac and Gugercin 2016)) are inadequate; in particular, they yield ROMs that are less accurate than ROMs constructed with a *random* sample mesh that knows nothing about the problem dynamics (Blonigan et al. 2021).

We hypothesize herein that it may be possible to improve the accuracy of hyper-reduced ROMs through the creation of a set of *evolving* sample meshes, calculated using the unique features present in the solution at each time, or within time windows. The parallel to AMR (Berger and Oliger 1984) should be clear. To explore this idea, we perform a preliminary study in which we use our event detection framework to calculate dynamically-changing sample meshes, with readily-available snapshots of the FOM solution and the solution residual as features. In this approach, we use the analysis partitions flagged as anomalous to define the sample mesh points.

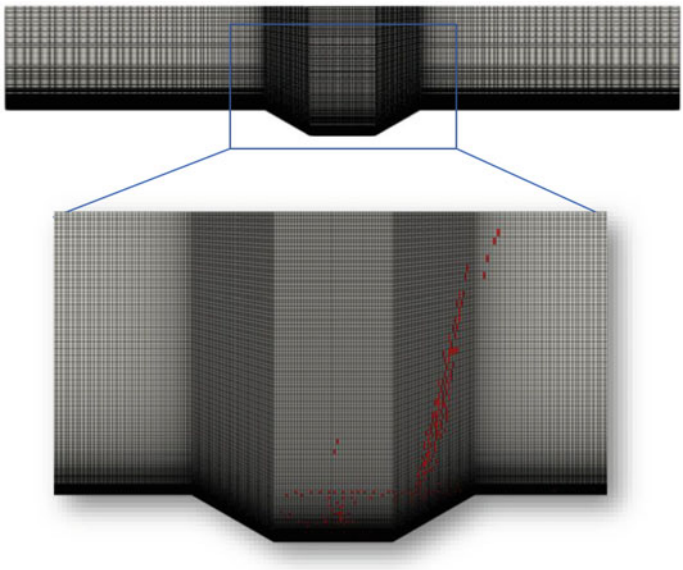
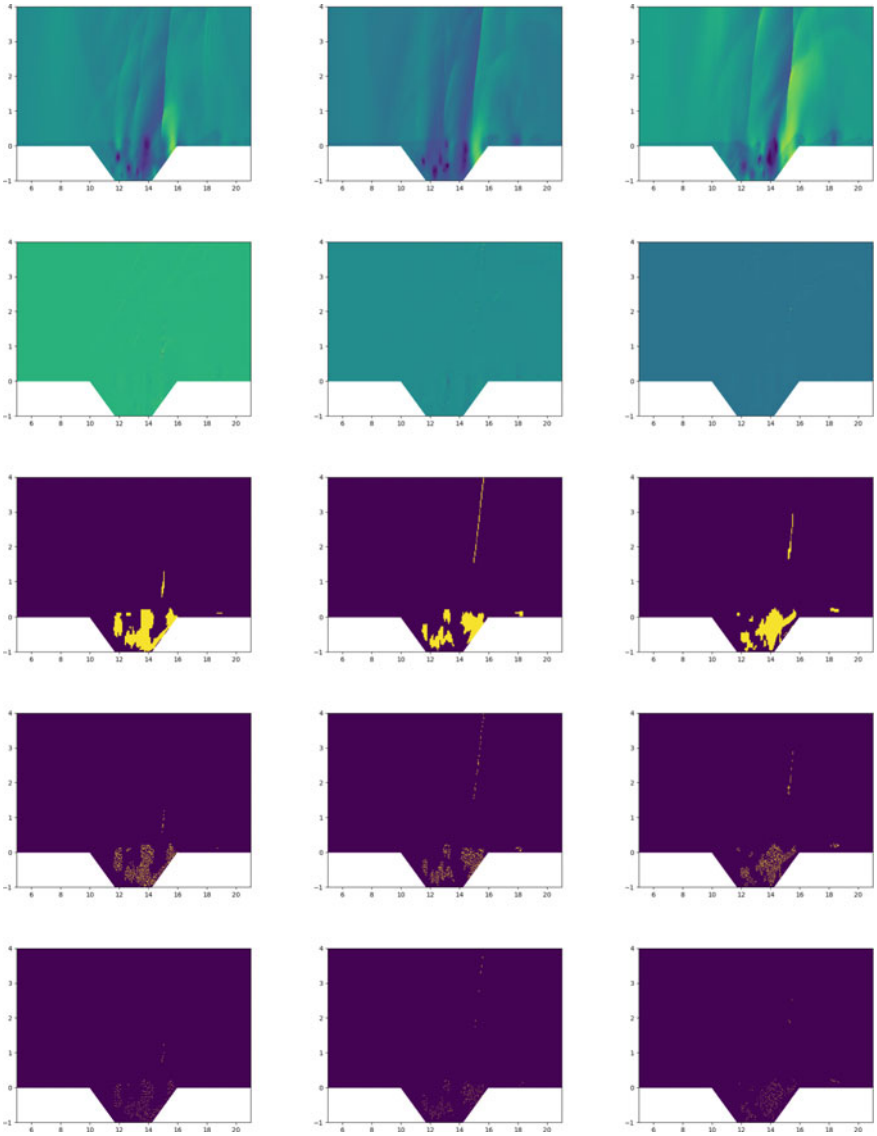


Fig. 18 Computational domain (top) and representative sample mesh points shown in red (bottom) for the 2D open cavity geometry. The sample mesh was obtained using the p–sampling approach (Drmac and Gugercin 2016)

Below, we present and describe some preliminary results exploring the viability of our proposed approach to dynamic sample mesh generation in the context of a problem involving a 2D viscous compressible flow with a Reynolds number of 10,000 over an open cavity geometry, pictured in Fig. 18. To generate a FOM of the form (2), the governing compressible Navier-Stokes equations are discretized in space using a third order Discontinuous Galerkin (DG) method with 600×240 elements in the streamwise and wall-normal direction, respectively, and in time with a Crank-Nicolson time-stepper having a timestep of 5×10^{-3}. The mesh for this geometry is obtained by discretizing a rectangular region with a uniform 600×240 mesh, and transforming it to fit the cavity geometry of interest. More details pertaining to the high-fidelity discretization can be found in Parish and Carlberg (2021) and are not repeated here for the sake of brevity. The free-stream Mach number is unity, which causes a shock to form in the problem solution (see Fig. 19, top row). A POD basis is constructed from 1000 snapshots of the high-fidelity solution. These same snapshots are employed to calculate a sample mesh having 1000 points using the p–sampling approach. This sample mesh is shown in Fig. 18.

The objective of the present section is to explore the viability of constructing *dynamic* sample meshes using our event detection framework. The natural choice of features to use for this task are the solution (Fig. 19, top row) and the solution residual (Fig. 19, second row). The former is a vector of the four primary conserved variables, ρ, ρu, ρv and ρe, where ρ is the fluid density, u and v are the fluid velocities, and e



(a) Snapshot 100 (b) Snapshot 500 Snapshot 928

Fig. 19 Plots of the density solution (top row), the density residual (second row) and dynamic sample meshes calculated using our event detection framework (rows 3–5) for the 2D compressible cavity flow problem at the times of snapshots 100 (**a**), 500 (**b**) and 928 (**c**). In rows 3–5, sample mesh points are shown in yellow. The sample meshes in rows 4 and 5 are obtained by randomly selecting one-fourth and one-sixteenth of the points, respectively, within each interesting analysis partition shown in the third row

is the fluid energy; the latter is the residual of the governing PDEs for each of these variables, which contains the nonlinear terms in the governing partial differential equations, the compressible Navier-Stokes equations. For the purpose of the event detection, we partition our geometry into 150×60 analysis partitions, each having 4×4 cells. In this preliminary study, we consider the *quartile* signature (Table 1), the *dbscan* measure (Table 2) with $\varepsilon = 0.3$ and $N_p = 1\%$ (Table 2) and the *threshold* decision with a threshold of 0.5 (Table 4). The sample meshes returned by this approach are plotted in Fig. 19. Row 3 of this figure shows in yellow the interesting partitions, which define a dynamic sample mesh, identified by our event detection framework at the time of snapshots 100, 500, and 928, respectively. The reader can observe that the dynamic sample meshes are changing in time. Additionally, the sample mesh points are in general concentrated within the cavity and in the vicinity of the shock that is seen in the density solutions (Fig. 19, top row).

The reader can observe by comparing the third row of Fig. 19 with Fig. 18 that the sample meshes identified by our event detection framework are qualitatively similar to the static sample mesh obtained using the p–sampling algorithm. In an effort to measure the quality of the dynamic sample meshes calculated using our framework, we calculate the following quantity given a sample mesh represented by the matrix $\mathbf{A}$:

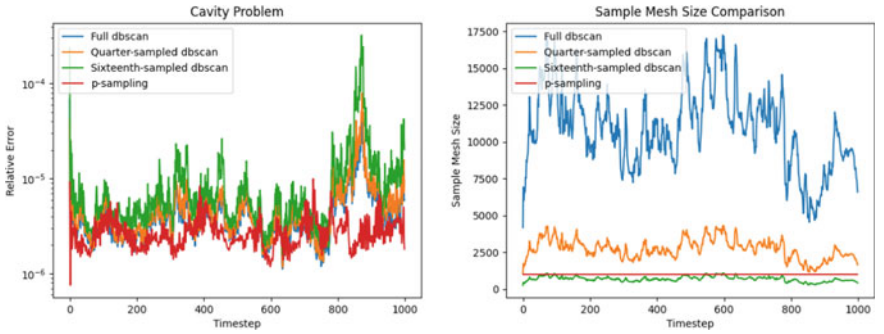
$$\epsilon := \frac{||\boldsymbol{w} - \boldsymbol{w}_s||_2}{||\boldsymbol{w}||_2}, \tag{6}$$

where $\boldsymbol{w}_s := \boldsymbol{\Phi}\hat{\boldsymbol{w}}_s$ and

$$\hat{\boldsymbol{w}}_s = \arg\min_{\hat{\boldsymbol{w}} \in \mathbb{R}^M} ||\mathbf{A}\mathbf{X} - \mathbf{A}\boldsymbol{\Phi}\hat{\boldsymbol{x}}||_2^2. \tag{7}$$

In this context, $\boldsymbol{x}_s$ is the optimal state one can reconstruct given knowledge of only the FOM state and the sample mesh. The quantity (6) has the advantage that it is computable offline (without running the full model reduction workflow).

Figure 20a plots the quantity ϵ from (6) for the fluid density solution as a function of time for the dynamic sample meshes obtained using our approach and for the static sample mesh obtained using p–sampling. As noted earlier, this comparison is not entirely consistent, since our dynamic sample meshes contain far more points than the static sample mesh we are comparing to (see Fig. 20b). A very simple strategy for reducing the sizes of our dynamic samples is to randomly drop a fixed fraction of the sample mesh points within each analysis partition flagged by our approach. Figure 19 shows the resulting sample meshes when one-quarter (fourth row) and one-sixteenth (fifth row) of the sample mesh points are kept within each interesting analysis partition. By randomly selecting just one sample mesh point within each interesting analysis partition (which corresponds to the one-sixteenth sub-sampling shown in Fig. 19, the fifth row), it is possible to reduce the sizes of our dynamic sample meshes to be on the order of the static sample mesh obtained through p–sampling (Fig. 20b). Remarkably, as the reader can see from examining Fig. 20a, reducing the number of sample mesh points in this way does not increase the error (6). While the



(a) Density error ϵ (b) Sample mesh size

Fig. 20 Comparison of errors ϵ in the density solution and the sample mesh size as a function of time for the cavity flow problem for sample meshes calculated using our event detection framework versus p–sampling

fact that the error (6) for the dynamic sample meshes obtained using our approach are roughly comparable to the errors of the p–sampling sample mesh may seem negative, it is actually encouraging, given that our approach is unsupervised and not based on an underlying optimization problem. Future work will focus on improving the sample meshes calculated using our approach, e.g., by bringing in ideas from traditional sample mesh approaches, which are based on minimizing the approximation error on a given sample mesh. Additionally, we plan to deploy our approach on test cases with more sophisticated dynamics, for which a dynamic sample mesh procedure will likely yield a greater benefit (e.g., problems with moving shocks). Future work will also include the design of signature-measure pairs that can guarantee that a given number of analysis partitions are selected at any given timestep; in order to achieve this, it is necessary to use a non-boolean measure.

3.4 HPC Experiments

As discussed in Sect. 1, an important requirement for an in situ event detection framework is that it be scalable and communication-minimizing. In this section, we verify the scalability of our framework in an HPC application utilizing MPI (Message Passing Interface Forum 1994) for coordinating the parallel communication and computation. In order to perform this study, we embedded a Python interpreter in the S3D combustion simulation code (Chen et al. 2009) which is written in Fortran 90. References to the raw data from the Fortran side were passed to the Python framework at each timestep, without duplication. The mpi4py package (Dalcín et al. 2005) was used to access the MPI environment from Python and perform collective communication between processors.

We ran our experiment using the Cori Cray XC40 machine at NERSC. The simulation represented conditions of a homogeneous charge compression ignition (HCCI) combustion of ethanol-air mixture at conditions typical of internal combustion engines. The mixture undergoes compression heating and auto-ignition kernels appear locally in small pockets, as shown in Fig. 21, that lead to the eventual combustion of the entire mixture. The goal for an event detection algorithm in this case is to identify the partitions where the auto-ignition kernels appear.

We decomposed the 2D simulation domain into 1024 partitions, with one partition per MPI rank, and processed 626 snapshots with 3136 grid points per partition and 33 features at each grid point. The event detection involved the following steps:

- global *min-max* pre-processing utilizing two MPI all-reduces, one each for the per-feature global *min* and *max*, over a vector of a size equal to the number of features.
- *mean* signature on the data locally on each partition (no MPI communication involved).
- *msd* measure, which involves computing a global mean of signatures and requires an MPI all-reduce of a vector of size equal to the number of features.

In a previous work (Konduri et al. 2018), we used this simulation as a motivation for designing a new signature – *feature moment metric* (fmm) – which represents the distribution of a given joint statistical moment (e.g., Kurtosis) across all the features. Here our focus is only on demonstrating the parallel performance of the framework and hence we use the simpler *mean* signature.

The execution times for the solver and the event detection components were recorded for the simulation. The solver execution time was 0.126 s for every simulation timestep. The event detection execution time ranged from a minimum of 0.012 s per timestep to a maximum of 2.28 s, with an average of 0.2 s. Because the

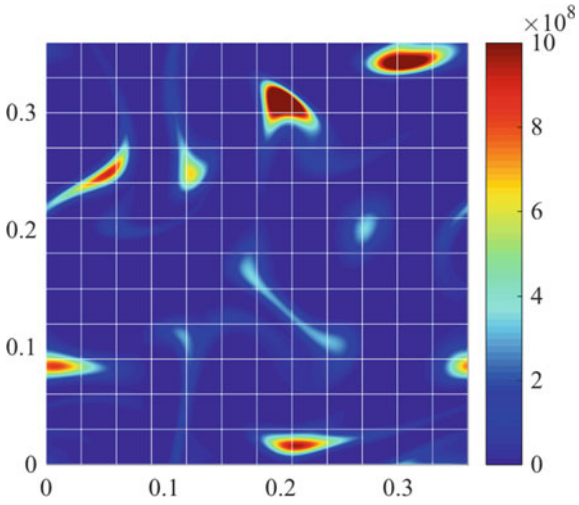


Fig. 21 Contour plot of heat release rate ($J/m^3/s$) at an early instant of the HCCI combustion simulation. A 12×12 partitioning of the domain is shown with white lines, and auto-ignition kernels are denoted by regions of high (red) heat release rate

workflow was identical from one timestep to the next, the large variation in the times can be attributed to system noise. While not negligible, the average event detection time was on the same order of magnitude as the solver, and thus within the realm of practicality, depending on the application. Encouragingly, the minimum time was an order of magnitude smaller than simulation time, suggesting that – under conditions free of system noise – the event detection could be performed in a fraction of the simulation time.

Note too that we used Python in situ to run this experiment for expediency, and that the analysis time could be drastically reduced by porting our framework to compiled code. Finally, analysis overhead could be further reduced for large-scale applications by reducing the number of event detection checks. Performing the event detection at, for instance, every Nth timestep would be an effective compromise between traditional check-pointing and fine-grained event detection, reducing the event detection load to a negligible portion of the runtime.

4 Conclusion

This work represents a first step in the development of event detection algorithms that can automatically identify events of interest in situ. Specifically, we presented a signatures-measures-decisions framework for the development of in situ HPC event detection algorithms. This framework is a useful decomposition that supports generalizability, unsupervised detection, low communication requirements and online processing. We have developed components under this framework which enable the use of standard event detection algorithms under the aforementioned constraints, in addition to entirely new combinations. We illustrated how example algorithms made from these components can optimize I/O while running an HPC simulation, leading to the capture of many more interesting events than typical uniform check-pointing. We highlighted two additional use cases for the proposed framework: detecting interesting events in HPC simulations (the Marine Ice Sheet Instability in land ice data), and identifying optimal space-time subregions for the hyper-reduction step of a typical projection-based model reduction workflow. Finally, we demonstrated, in a study using HPC and MPI, that in situ event detection overhead can be on the order of magnitude of the simulation, and performance can be improved further with minor adjustments.

This work enables future research in several areas, such as the question of what should constitute an "interesting" event for a given simulation, or, ideally, how to define "interesting" for *any* given simulation. Apart from detecting events the proposed approaches can also identify numerical anomalies, which can help with debugging and interpretation of simulation results. In addition, it is possible that this framework can be used to classify events either in situ or as a post-processing technique by analyzing the signatures themselves; the signatures distill information from a large number of samples and are less expensive to analyze. Finally, we hope that experiments done using this framework will inspire HPC simulation code developers

to incorporate these capabilities into native code, allowing for even more efficient in situ event detection.

Acknowledgements This work was funded through U.S. Department of Energy Advanced Scientific Computing Research (ASCR) grant FWP #18019471. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

The authors gratefully acknowledge Dr. Stephen Price, Dr. Matt Hoffman and Dr. Mauro Perego for providing the MALI simulation data analyzed in Sect. 3.2, for engaging in many fruitful discussions regarding the physics of the Marine Ice Sheet Instability (MISI), and for assisting with the interpretation of our results within the context of MISI.

References

Ahmed T (2009) Online anomaly detection using KDE. In: IEEE global telecommunications conference, pp 1–8

Ansar S, Hussain M, Mazhar S, Manzoor T, Siddiqui K, Abid M, Jamal H (2019) Mesh partitioning and efficient equation solving techniques by distributed finite element methods: a survey. Arch Comput Meth Eng 26:1–16

Bamber J, Riva R, Vermeersen B, LeBrocq A (2009) Reassessment of the potential sea-level rise from a collapse of the West Antarctic Ice Sheet. Science 324(5929):901–903

Bandaragoda TR, Ting KM, Albrecht D, Liu FT, Wells JR (2014) Efficient anomaly detection by isolation using nearest neighbour ensemble. In: IEEE international conference on data mining, pp 698–705

Barrault M, Maday Y, Nguyen NC, Patera AT (2004) An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. Comptes Rendus Mathematique 339(9):667–672

Bennett JC, Bhagatwala A, Chen JH, Pinar A, Salloum M, Seshadhri C (2016) Trigger detection for adaptive scientific workflows using percentile sampling. SIAM J Sci Comp 38:240–260

Berger MJ, Oliger J (1984) Adaptive mesh refinement for hyperbolic partial differential equations. J Comput Phys 53(3):484–512

Blonigan PJ, Rizzi F, Howard M, Fike JA, Carlberg KT (2021) Model reduction for steady hypersonic aerodynamics via conservative manifold least-squares petrov-galerkin projection. AIAA J 59(4):1296–1312

Carlberg K, Farhat C, Cortial J, Amsallem D (2013) The gnat method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. J Comput Phys 242:623–647

Carlberg K, Barone M, Antil H (2017) Galerkin v. least-squares petrov–galerkin projection in nonlinear model reduction. J Comput Phys 330:693–734

Chaturantabut S, Sorensen DC (2010) Nonlinear model reduction via discrete empirical interpolation. SIAM J Sci Comp 32(5):2737–2764

Chen JH, Choudhary A, Supinski BD, DeVries M, Hawkes ER, Klasky S, Liao W-K, Ma K-L, Crummey JM, Podhorszki N et al (2009) Terascale direct numerical simulations of turbulent combustion using S3D. Comp Sci Discov 2(1):015001

Dalcín L, Paz R, Storti M (2005) Mpi for python. J Par Distri Comp 65(9):1108–1115

Drmac Zl, Gugercin S (2016) A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. SIAM J Sci Comp 38(2):A631–A648
Edwards TL, Nowicki S, Marzeion B et al (2021) Projected land ice contributions to twenty-first-century sea level rise. Nature 593:74–82
Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the second international conference on knowledge discovery and data mining. AAAI Press, pp 226–231
Everson R, Sirovich L (1995) Karhunen-loève procedure for gappy data. J Opt Soc Am A 12(8):1657–1664
Ffmpeg (2019) Online; accessed 2019-09-18. https://ffmpeg.org
Gardner AS, Moholdt G, Scambos T, Fahnstock M, Ligtenberg S, van den Broeke M, Nilsson J (2018) Increased West Antarctic and unchanged East Antarctic ice discharge over the last 7 years. The Cryosphere 12(2):521–547
Hartigan JA, Wong MA (1979) Algorithm AS 136: a K-Means clustering algorithm. App Stat 28(1):100–108
Hoffman M (2022) Personal correspondance
Hoffman MJ, Perego M, Price SF, Lipscomb WH, Zhang T, Jacobsen D, Tezaur I, Salinger AG, Tuminaro R, Bertagna L (2018) Mpas-albany land ice (mali): a variable-resolution ice sheet model for earth system modeling using voronoi grids. Geosci Model Develop 11(9):3747–3780
Holmes P, Lumley JL, Berkooz G (1996) Turbulence, coherent structures, dynamical systems and symmetry. Cambridge University Press
Imagecat (2022) Online; accessed 2022-01-11. https://imagecat.readthedocs.io
IPCC (2021) Representative Concentration Pathways (RCPs). https://sedac.ciesin.columbia.edu/ddc/ar5_scenario_process/RCPs.html
Jones E, Oliphant T, Peterson P et al (2001) SciPy: Open source scientific tools for Python –. Online; accessed 2019-09-18. http://www.scipy.org
Joughin I, Alley R (2019) Stability of the West Antarctic ice sheet in a warming world. Nature 4:506–513
Konduri A, Kolla H, Kegelmeyer WP, Shead TM, Ling J, Davis WL (2018) Anomaly detection in scientific data using joint statistical moments. J Comput Phys 387:522–538
Kunisch K, Volkwein S (2002) Galerkin proper orthogonal decomposition for a general equation in fluid dynamics. SIAM J Num Anal 40(2):492–515
LeGresley P (2006) Application of proper orthogonal decomposition (POD) to design decomposition methods. PhD thesis, Stanford University
Leung LR, Bader DC, Taylor MA, McCoy RB (2020) An introduction to the e3sm special collection: goals, science drivers, development, and analysis. J Adv Model Earth Sys 12(11):e2019MS001821
Ling J, Kegelmeyer WP, Aditya K, Kolla H, Reed KA, Shead TM, Davis WL (2017) Using feature importance metrics to detect events of interest in scientific computing applications. In: 2017 IEEE 7th symposium on large data analysis and visualization (LDAV). IEEE, pp 55–63
Liu FT, Ting KM, Zhou ZH (2012) Isolation-based anomaly detection. ACM Trans Knowl Disc Data 6:1–39
Message Passing Interface Forum (1994) Mpi: a message-passing interface standard. Technical report, University of Tennessee, USA
Nguyen NC, Peraire J (2008) An efficient reduced-order modeling approach for non-linear parametrized partial differential equations. Int J Num Meth Eng 76(1):27–55
Nguyen N, Patera A, Peraire J (2008) A 'best points' interpolation method for efficient approximation of parametrized functions. Int J Num Meth Eng 73:521–543
Parish EJ, Carlberg KT (2021) Windowed least-squares model reduction for dynamical systems. J Comput Phys 426:109939
Pattyn F, Morlighem M (2020) The uncertain future of the antarctic ice sheet. Science 367(6484):1331–1335

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Perego M, Price S, Stadler G (2014) Optimal initial conditions for coupling ice sheet models to earth system models. J Geophys Res, Earth Surface 119(9):1894–1917

Python. Online; accessed 2019-09-18. https://www.python.org

Rathinam M, Petzold L (2003) A new look at proper orthogonal decomposition. SIAM J Num Anal 41(5):1893–1925

Reese R, Gudmundsson G, Levermann A, Winkelmann R (2018) The far reach of ice-shelf thinning in Antarctica. Nat Clim Change 8:53–57

Robel AA, Seroussi H, Roe GH (2019) Marine ice sheet instability amplifies and skews uncertainty in projections of future sea-level rise. Proc Natl Acad Sci 116(30):14887–14892

Sasidharan A, Dennis JM, Snir M (2015) A general space-filling curve algorithm for partitioning 2d meshes. In: 2015 IEEE 17th international conference on high performance computing and communications, 2015 ieee 7th international symposium on cyberspace safety and security, and 2015 ieee 12th international conference on embedded software and systems, pp 875–879

Seroussi H, Nowicki S, Payne AJ, Goelzer H, Lipscomb WH, Abe-Ouchi A, Agosta C, Albrecht T, Asay-Davis X, Barthel A, Calov R, Cullather R, Dumas C, Galton-Fenzi BK, Gladstone R, Golledge NR, Gregory JM, Greve R, Hattermann T, Hoffman MJ, Humbert A, Huybrechts P, Jourdain NC, Kleiner T, Larour E, Leguy GR, Lowry DP, Little CM, Morlighem M, Pattyn F, Pelle T, Price SF, Quiquet A, Reese R, Schlegel N-J, Shepherd A, Simon E, Smith RS, Straneo F, Sun S, Trusel LD, Van Breedam J, van de Wal RSW, Winkelmann R, Zhao C, Zhang T, Zwinger T (2020) ISMIP6 Antarctica: a multi-model ensemble of the Antarctic ice sheet evolution over the 21st century. The Cryosphere 14:3033–3070

Sirovich L (1987) Turbulence and the dynamics of coherent structures, Part III: dynamics and scaling. Q Appl Math 45(3):583–590

Sun S, Pattyn F, Simon EG, Albrecht T, Cornford S, Calov R, Dumas C, Gillet-Chaulet F, Goelzer H, Golledge NR et al (2020) Antarctic ice sheet response to sudden and sustained ice-shelf collapse (ABUMIP). J Glaciol 66(260):891–904

Thomas RH, Bentley CR (1978) A model for holocene retreat of the west antarctic ice sheet. Quat Res 10(2):150–170

Thuerey N, Pfaff T (2018) MantaFlow. Online; accessed 2019-09-18. http://mantaflow.com

Ullrich PA, Zarzycki CM (2016) Tempestextremes v1.0: a framework for scale-insensitive pointwise feature tracking on unstructured grids. In: Geoscientific model development discussion

Walt SVD, Colbert SC, Varoquaux G (2011) The numpy array: a structure for efficient numerical computation. Comp Sci Eng 13(2):22

Weertman J (1974) Stability of the junction of an ice sheet and an ice shelf. J Glaciol 13(67):3–11

Welch P (1967) The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Trans Audio Electroacoust 15(2):70–73

Wu K, Zhang K, Fan W, Edwards A, Philip SY (2014) Rs-forest: a rapid density estimator for streaming anomaly detection. In: IEEE international conference on data mining, pp 600–609

Young D, Wright A, Roberts J, Warner R, Young N, Greenbaum J, Schroeder D, Holt J, Sugden D, Blankenship D, vanOmmen T, Siegert M (2011) A dynamic early East Antarctic Ice Sheet suggested by ice-covered fjord landscapes. Nature 474:72–75

Zhao M, Held IM, Lin SJ, Vecchi GA (2009) Simulations of global hurricane climatology, interannual variability, and response to global warming using a 50-km resolution GCM. J Clim 22:6653–6678