Taylor & Francis
Taylor & Francis Group

Quality Technology & Quantitative Management

ISSN: (Print) 1684-3703 (Online) Journal homepage: http://www.tandfonline.com/loi/ttqm20

An effective online data monitoring and saving strategy for large-scale climate simulations

Xiaochen Xian, Rick Archibald, Benjamin Mayer, Kaibo Liu & Jian Li

To cite this article: Xiaochen Xian, Rick Archibald, Benjamin Mayer, Kaibo Liu & Jian Li (2018): An effective online data monitoring and saving strategy for large-scale climate simulations, Quality Technology & Quantitative Management, DOI: 10.1080/16843703.2017.1414112

To link to this article: https://doi.org/10.1080/16843703.2017.1414112

Published online: 22 Jan 2018.

QUALITY TECHNOLOGY & QUANTITATIVE MANAGEMENT, 2018
https://doi.org/10.1080/16843703.2017.1414112

Taylor & Francis
Taylor & Francis Group

An effective online data monitoring and saving strategy for large-scale climate simulations

Xiaochen Xian[a], Rick Archibald[b], Benjamin Mayer[b], Kaibo Liu[a] and Jian Li[c]

[a]Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI, USA; [b]ORNL Climate Change Science Institute, Oak Ridge, TN, USA; [c]School of Management and State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an, China

ABSTRACT

Large-scale climate simulation models have been developed and widely used to generate historical data and study future climate scenarios. These simulation models often have to run for a couple of months to understand the changes in the global climate over the course of decades. This long-duration simulation process creates a huge amount of data with both high temporal and spatial resolution information; however, how to effectively monitor and record the climate changes based on these large-scale simulation results that are continuously produced in real time still remains to be resolved. Due to the slow process of writing data to disk, the current practice is to save a snapshot of the simulation results at a constant, slow rate although the data generation process runs at a very high speed. This paper proposes an effective online data monitoring and saving strategy over the temporal and spatial domains with the consideration of practical storage and memory capacity constraints. Our proposed method is able to intelligently select and record the most informative extreme values in the raw data generated from real-time simulations in the context of better monitoring climate changes.

ARTICLE HISTORY
Accepted 3 December 2017

KEYWORDS
Big data; local extrema; raw simulation data; spatial and temporal domains

1. Problem statement

Climate research has been an important topic to study, which aims to understand the significant influence of climate changes on human society and environment. For example, by simulating climate changes, we are able to explore the details of local changes. These simulations enable understanding to assist in mitigation of detrimental effects. As natural climate change often takes place over thousands or even millions of years, large-scale simulation models have become an essential tool in the climate research to predict the future climate changes. In general, the simulation models consist of systems of differential equations that are derived to characterize a wide range of important drivers of climate changes, such as atmosphere, ocean, ices, and land. At each timestep, the simulation models run the differential equations on a three-dimensional grid over the simulations domain and evaluate physical interactions with its neighboring points. In this way, a snapshot of the simulation at each timestep can be sequentially generated. This process happens at 1850 times real-time.

CONTACT Kaibo Liu kliu8@wisc.edu

With the rapid advancement in computing and simulation technology, many large-scale climate models, such as Earth System Model (ESM), have been developed to achieve high model accuracy over a long period of simulation time (Collins, Johansen, Evans, Woodward, & Caldwell, 2015; Gerhard, Iacono, May, Müller, & Schäfer, 2014). These models provide an unprecedented opportunity for us to better understand the climate changes with more granular information; however, how to deal with the Big Data produced from these simulation models in real time has become a central challenge nowadays (Archibald, Evans, Drake, & White, 2010). Big Data is often characterized by '3Vs', referring to high Variety, high Velocity, and high Volume of information. Here, the simulation data generated from the climate research face the challenges of high velocity and high volume as a typical climate simulation program can generate one terabyte of data per day, or petabytes of data per year (Lautenschlager & Reinke, 2012). The primary reason that climate research produces such Big Data is because the investigations of climate changes over both spatial and temporal domains are critically important. On one hand, climate processes are driven by factors across a range of spatial scales, which requires the integrated study of both global and regional climate changes (Palmer, 2014). On the other hand, while some abrupt climate processes perform over a few minutes, they may interact with longer term natural climate processes which slowly evolve over periods that are longer than centuries.

Recently, some methods have been developed for online monitoring of Big Data from the quality control perspective (Huang, Kong, & Huang, 2014; Liu, Mei, & Shi, 2015; Ning & Tsung, 2012); however, they cannot be effectively used here due to the two unique challenges involved in the simulation process. (i) First, only limited simulation results can be written into the hard disk. While data generation process runs at a very high speed (Archibald et al., 2010), it is extremely slow to write the data to the hard disk comparing to the speed of data generation (Palmer, 2014). In other words, while the simulation data are continuously and sequentially generated in real time at a high speed, only a small proportion of the data can be permanently saved in practice. (ii) Second, the memory capacity is limited compared to the large volume of data generated from the simulation process. Since a large quantity of data is computed at each timestep and the simulation data are generated at a very high speed, the memory space will be used up quickly even in a short time window if it is not well managed. Consequently, a critical challenge here is how to leverage the limited memory capacity to effectively monitor the climate changes and timely select the most informative data to save into the hard disk in real time.

These two challenging issues are currently critical and well recognized in the routine practice at the Climate Change Science Institute of Oak Ridge National Laboratory (ORNL). ORNL owns the largest open science computing system in the United States, known as Titan, which has been widely used to generate large-scale, long-duration climate simulations with both high spatial and temporal resolution information (Mayer, Worley, da Silva, & Gaddis, 2015). In particular, Titan is able to generate numerical simulations on the globe that can be gridded with 3000×1500 points in a layer, and with 60 layers in total. Although the generation of one snapshot of simulation data only takes about 90 s, it needs up to 3 h in writing such one snapshot of information to the hard disk due to the slow process of writing data to disk as mentioned in challenge (i) As a result, the saving ratio, defined by the ratio of the amount of data saved in the hard disk over the total amount of data generated from simulations, is merely about $\frac{90}{3\times60\times60} = 1{:}120$. That is to say, saving is 120 times slower than the data generation process and about $\frac{119}{120} \approx 99.17\%$ of the data that produced from simulations has to be thrown away. In addition to the challenge of data storage, here the memory capacity is also limited compared to the speed and the volume of the generated data from simulations. In other words, if the important results are not timely written into the hard disk, the data temporarily stored in the memory space have to be discarded to make room for the new data generated at the next timestep. This memory constraint thus poses an additional challenge for effective monitoring and saving the simulation data in real time.

To address these limitations, several data reduction and compression algorithms have been developed in the literature. A review of the lossless and lossy compression algorithms for the climate data can be found in (Baker et al., 2014). While these methods have successfully demonstrated in some

applications (Lakshminarasimhan et al., 2011; Liu, Huang, Fu, Yang, & Song, 2013), their saving ratios vary in a range of 30%–80%, thus falling short in satisfying the application requirements at the ORNL (typically less than 1:100). In addition, most of these existing methods are offline post-processing techniques that cannot be effectively used in real time due to high computational cost.

The approach that currently implemented at the ORNL is to store the simulation data at a constant frequency over the temporal domain, i.e. only to save one snapshot of the generated data at every k timesteps, in which k is determined by the resource constraints mentioned above. In the followings, we call this approach as Constant Frequency-based Saving (CFS). Although the CFS approach is straightforward to implement, it may not effectively characterize the significant changes in the climate data. Figure 1 illustrates an example of one data stream generated from climate simulations. In particular, by using the CFS method, data points at every 3 timesteps are recorded, denoted by the red dots in Figure 1(a). It clearly shows that while the original data fluctuate over time, the CFS method cannot effectively characterize the significant changes in this data stream. In climate data, one of the key questions is what the extreme values are in the simulation results. This is because these extreme values contain important information for studying severe climate events, such as flooding and storm damage, which may significantly impact human activities and ecosystems. In addition, the extreme values are critically important for investigating the evidence and trends of climate changes (Meehl et al., 2000; Rahmstorf & Coumou, 2011). Although the simulated values in climate simulations involve random errors, the random errors can be negligible compared to the magnitude of simulated values. Therefore, how to effectively monitor and record the extreme values that are continuously generated from simulations has become an essential task in climate research. In this sense, a desired saving strategy is illustrated in Figure 1(b) which can automatically detect and record the extreme values while the total number of data points saved is still the same as the original CFS approach.

The objective of this paper is, thus, to develop an effective online monitoring and saving strategy that can intelligently detect and select the most informative data (in term of the extreme values) to save to the hard disk in real time such that large changes in the simulations can be recorded. Here, we would like to further clarify the meaning of the word 'effective' in our context. In particular, the 'effectiveness' should be evaluated by comparing with a baseline model, i.e. the currently used CFS approach at the ORNL, subject to the resource constraints that only a certain amount of simulation data can be saved in the hard disk and carried in the memory space in real time. As a result, the core challenge here is how to manage the limited resource constraints for data monitoring and saving simultaneously when the simulation results are continuously generated in real time, rather than the offline analysis that identifies the extreme values when all the simulation results are available. In fact, according to the real studies at ORNL, it is nearly impossible to save all the results generated from the simulation process as even saving the whole one-month simulation results will require 8 years! While

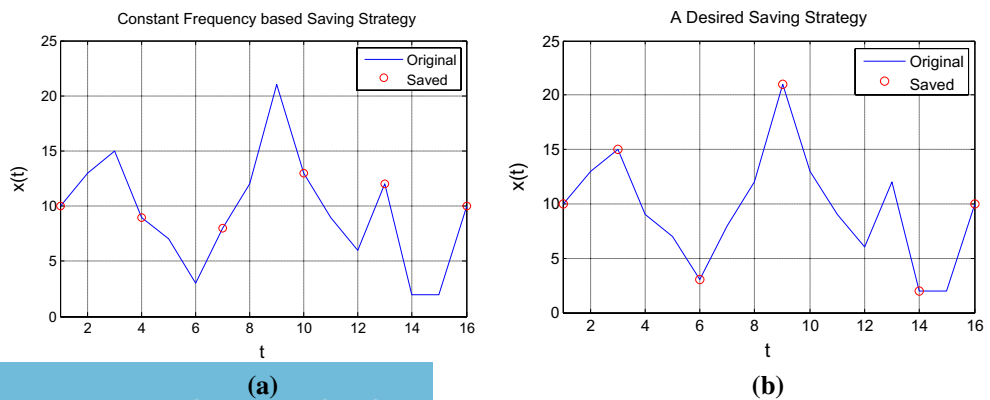

Figure 1. Illustration of (a) the CFS approach and (b) a desired saving strategy.

there are some approaches in the literature that only record some summary statistics of interest (e.g. the average of data streams) (Castruccio & Genton, 2016) or consider some in-transit techniques which process the simulation results while they are still resident in memory (Bennett, Comandur, Pinar, & Thompson, 2013; Bennett et al., 2012), these methods are not suitable to be used here as well. This is because the climate research is often done at the post-processing stage (i.e. specific scientific questions were not known at the time when the simulation was run), thus requiring to save the raw simulation data directly when it is generated in real time.

2. Data and approach

As climate research is based on simulation analysis as investigations of climate changes are on both the global and regional scales and often over hundreds or thousands of years. It is a common practice in this real application to consider the simulation data generated from a powerful simulation model as a substitution for the real data. In particular, the data-sets are generated based on the large-scale simulation models in Archibald et al. (2010), which are used to simulate zonal flow impinging on a mountain (Williamson, Drake, Hack, Jakob, & Swarztrauber, 1992). There are three different spatial resolutions involved in the data-sets: low resolution, medium resolution, and high resolution, and the detailed resolution information is listed in Table 1. At each time t, there are in total m data points generated from the simulation model. Figure 2 illustrates two snapshots of the high-resolution data-set at time $t = 6000$ and $t = 12{,}000$, which shows that each data stream dynamically changes over time and the magnitude of such changes are quite different for different data streams.

Suppose that in a climate simulation, there are in total m variables of interest. At each time t, $\boldsymbol{x}(t) = (x(1,t), \ldots, x(m,t))' \in \mathbb{R}^{m\times 1}$ denotes the generated simulation data of these m variables. Assume that the initial condition of the simulation model $\boldsymbol{x}(1)$ is known and the simulation process runs for T total number of timesteps, which creates a full data-set $X = \{x(i,t): i = 1, 2, \ldots, m, t = 1, 2, \ldots, T\}$. Denote $\boldsymbol{x}(i,\cdot) = (x(i,1), \ldots, x(i,T)) \in \mathbb{R}^{1\times T}$ to be the simulation data of the ith data stream. It should be noted that the distributions or the values of each data steam are generally unknown or unpredictable before conducting simulations. This is because the data generation needs to be done sequentially, i.e. the data generated at time $t + 1$ depend on the results before time t.

As mentioned in the first section, our interested problem faces two main resource constraints: (i) only limited data can be temporarily stored in the memory space; and (ii) only limited simulation results can be written into the hard disk. Here, we provide the detailed information:

(i) Denote $\mathcal{M}(t) \subseteq [\boldsymbol{x}(1), \ldots, \boldsymbol{x}(t)] \in \mathbb{R}^{m\times t}$ to be the set of data points that are temporally stored in the memory space at time t. In particular, we assume that the total number of data points in the set $\mathcal{M}(t)$ must be less than or equal to a predefined threshold M, i.e. $|\mathcal{M}(t)| \leq M$, where $|\cdot|$ denotes the cardinality of a set. This constraint requires us to dynamically manage the data in $\mathcal{M}(t)$ (i.e. either discarded or permanently saved to the hard disk) during the continuous data generation process.

(ii) Denote $\mathcal{L}(t) \subseteq [\boldsymbol{x}(1), \ldots, \boldsymbol{x}(t)] \in \mathbb{R}^{m\times t}$ to be the set of data points that have been permanently saved in the hard disk by time t. Once the data are in the set $\mathcal{L}(t)$, they cannot be removed or changed. Please note that the saving device normally reads or writes a batch of data points, which is called a 'block' at a time to minimize overhead. To be consistent with this practical setting as in the CFS approach, here we assume that it takes k timesteps to save a block of s data points.

Table 1. Dimensions of data-sets with three spatial resolutions.

	Number of data streams $m = m_1 \times m_2$	Number of total timesteps, T
Low resolution	$30 \times 64 = 1920$	3000
Medium resolution	$62 \times 128 = 7936$	6000
High resolution	$126 \times 256 = 32{,}256$	12,600

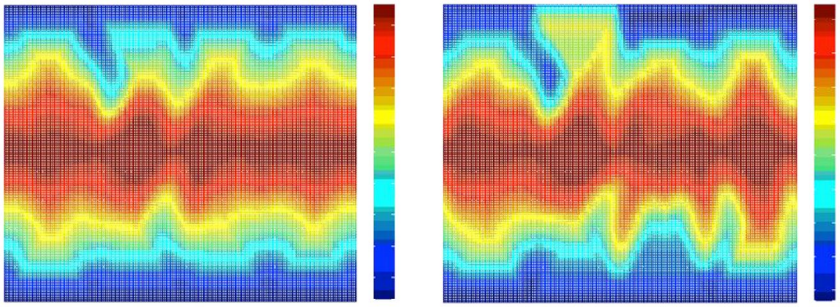

Figure 2. Snapshots of the high-resolution climate data-set at time (a) $t = 6000$ and (b) $t = 12{,}000$.

Specifically, this means that at time $t = n \cdot k$ ($n = 1, 2, \ldots \lfloor \frac{T}{k} \rfloor$, in which $\lfloor N \rfloor$ denotes the largest integer that is less than or equal to N), s data points $\{x(G): G = \{(i_1, t_1), \ldots, (i_s, t_s)\}\}$ need to be selected from the memory space $\mathcal{M}(t)$ and added into the set $\mathcal{L}(t)$, where $i_j = 1, 2, \ldots, m$, $t_j \leq t$:

$$\mathcal{L}(t) = \mathcal{L}(t-1) \cup \{x(G)\}, \mathcal{M}(t) = \mathcal{M}(t) \backslash \{x(G)\}. \tag{1}$$

Thus, the key challenge here is how to sequentially determine which s data points should be written into the hard disk at every k timesteps. Please note that for a simulation process with T total number of timesteps, there are only in total $S = s\lfloor \frac{T}{k} \rfloor$ data points that can be saved. In this way, $r = \frac{S}{mT}$ is the saving ratio, which denotes the percentage of data saved in the hard disk over the total amount of data generated from the simulation.

To highlight our main ideas, below we first give definitions of the local extrema and the shift indicators, respectively.

Definition 1: A data point $x(i, t)$ is called a local maximum (minimum) for the ith data stream $\boldsymbol{x}(i, \cdot)$ if

$$x(i,t) > x(i,t-1) \text{ and } x(i,t) > x(i,t+1)$$

$$(x(i,t) < x(i,t-1) \text{ and } x(i,t) < x(i,t+1)), \tag{2}$$

where $i = 1, \ldots, m$, $t = 2, \ldots, T-1$. Here, local maxima and local minima are collectively called local extrema. Moreover, we denote $\tau_{i,j}$ to be the time at which the ith data stream $\boldsymbol{x}(i, \cdot)$ achieves its jth local extremum, i.e. $x(i, \tau_{i,j})$ is the jth local extremum of the data stream $\boldsymbol{x}(i, \cdot)$.

Definition 2: The shift indicator $d(i, t)$ for the ith data stream $\boldsymbol{x}(i, \cdot)$ is defined to be the numerical difference between the data point $x(i, t)$ and the most recent local extremum before time t, i.e.

$$d(i,t) = \; x(i,t) - x\left(i, \tau_{i,j-1}\right), \text{ for } \tau_{i,j-1} < t \leq \tau_{i,j}, j \geq 1, \tag{3}$$

where $\tau_{i,0} = 1$.

Particularly, as a special case of Equation (3), $d\left(i, \tau_{i,j}\right)$ denotes the difference between two consecutive local extrema (i.e. the jth and the $(j-1)$th local extrema) for the ith data stream $\boldsymbol{x}(i, \cdot)$

$$d\left(i, \tau_{i,j}\right) = x\left(i, \tau_{i,j}\right) - x\left(i, \tau_{i,j-1}\right). \tag{4}$$

Here, if $d(i, \tau_{i,j}) > 0$, $x\left(i, \tau_{i,j}\right)$ is a local maximum; otherwise, $x\left(i, \tau_{i,j}\right)$ is a local minimum.

Recall that the objective of this paper is to develop an effective online data monitoring and saving strategy that can intelligently identify and select the local extrema to save to the hard disk such that

large changes in the simulations can be recorded. For different data streams, there may be differences in the magnitude of changes, and thus we need to not only identify the changes of a data stream over the temporal domain, but also screen out the most significant changes of all data streams over the spatial domain. To be specific, here we give a clear definition about the significant data points that are desired to be saved during simulations.

Definition 3: Let $\{|d(I_{(1)})|, |d(I_{(2)})|, \dots\}$ denote the decreasing order statistics of the shift indicators $\{|d(I_1)|, |d(I_2)|, \dots\}$ (i.e. $|d(I_{(1)})| \geq |d(I_{(2)})| \geq \dots$), where I_l is the pair of index (i_l, τ_{i_l,j_l}) of the local extremum $x(i_l, \tau_{i_l,j_l})$. The set of 'significant' local extrema E is defined as:

$$E = \left\{x\left(I_{(l)}\right) : l = 1, \dots, |\mathcal{L}(T)|\right\}. \tag{5}$$

According to Definition 3, the data points in E are the local extrema with the top $|\mathcal{L}(T)|$ largest shift indicators over both temporal and spatial domains during the entire simulation, where $|\mathcal{L}(T)|$ refers to the maximum number of points that can be saved after the simulation is done. Thus, our goal is to in real time, identify these local extrema such that the cardinality of the intersection of the set E and the saved data points $\mathcal{L}(T)$ can be maximized. Mathematically, this means

$$\max |E \cap \mathcal{L}(T)|. \tag{6}$$

2.1. Overview of the proposed online data monitoring and saving strategy

We would like to emphasize that while our interested problem (i.e. detecting significant local extrema with large shift indicators) seems straightforward when all the simulation data are available offline, such task is quite challenging to be conducted in real time as (i) the ranked statistics of the shift indicators requires knowing all the simulation data ahead and (ii) there are only limited storage and memory capacities available. To address these issues, we propose an online monitoring and saving strategy named as Local Extremum-based Saving (LES) in this subsection.

From the technical point of view, our proposed method has two innovative ideas. One is to dynamically and adaptively select the most informative data points (i.e. significant local extrema) across the spatial domain to characterize large changes over the temporal domain during real-time stimulations. If we compare our method with the CFS approach from the spatiotemporal perspectives, both methods still save the same number of data points into the hard disk; however, the difference is that the CFS algorithm first looks at the spatial domain (the snapshot information $\boldsymbol{x}(t)$ of all data points) and then consider the temporal domain (save information at every k timesteps), while our LES algorithm switches the spatial and temporal domains such that it can better manage the limited resources to identify and store the most informative data points generated from simulations in real time. In this sense, the CFS approach can be regarded as a special case of our proposed method. Another innovation of our proposed method is that it allows us to simultaneously manage the challenges of data monitoring and saving in an integrated manner given the resource constraints.

There are three essential components involved in the proposed LES strategy: (i) detecting local extrema from real-time simulations, (ii) updating data points based on the memory space constraint, and (iii) saving data points into the hard disk. Below we discuss each component in details.

2.1.1. Detecting local extrema from real-time simulations

In this subsection, we will discuss a method to detect the local extrema based on real-time simulations. Note that the ith data stream $\boldsymbol{x}(i, \cdot) = (x(i, 1), \dots, x(i, T))$ can be separated into a series of segments offline such that the data stream is monotonic within each segment (which we refer to as 'monotonic segment' hereafter) and any two consecutive segments have opposite monotonicity (one increasing and the other decreasing). We denote the $(j+1)$th monotonic segment by $\left\{x\left(i, \tau_{i,j}\right), \dots, x\left(i, \tau_{i,j+1}\right)\right\}$, where recall $x\left(i, \tau_{i,j}\right)$ is the jth local extremum of the ith data stream.

During online monitoring, with the newly generated data point $x(i,t)$ at time t, we need to determine whether it is a new local extremum. However, this is impossible to be conducted without knowing the generated data at the next timestep according to definition 1. As a result, instead we propose to determine whether the data point $x(i,t-1)$ is a local extremum at time t. To do this, we first define a reference point $z(i,t-1)$ that records the most recent local extremum for the ith data stream before time $t-1$ ($\tau_{i,j} < t-1 \leq \tau_{i,j+1}$), i.e. $z(i,t-1) = x(i,\tau_{i,j})$, where $x(i,\tau_{i,j})$ is the first point in the $(j+1)$th monotonic segment $\{x(i,\tau_{i,j}), x(i,\tau_{i,j}+1), \ldots, x(i,\tau_{i,j+1}-1), x(i,\tau_{i,j+1})\}$. To get the value of $z(i,t)$ online, at time t we first let $z(i,t) = z(i,t-1)$. Then, we compare the sign of $x(i,t) - x(i,t-1)$ and $x(i,t-1) - z(i,t)$ to conclude the status of the point $x(i,t-1)$. Specifically, we conclude $x(i,t-1)$ to be a local extremum if

$$(x(i,t-1) - z(i,t)) \cdot (x(i,t) - x(i,t-1)) < 0. \tag{7}$$

This means that $x(i,t-1)$ is the first point in the $(j+2)$th monotonic segment, i.e. $\tau_{i,j+1} = t-1$, and thus we calculate the shift indicator $d(i,t-1) = x(i,t-1) - z(i,t)$ and then update the reference point $z(i,t) = x(i,t-1)$. Otherwise, if the following condition holds:

$$(x(i,t-1) - z(i,t)) \cdot (x(i,t) - x(i,t-1)) \geq 0, \tag{8}$$

it means that the data point $x(i,t-1)$ is still in the $(j+1)$th monotonic segment $\{x(i,\tau_{i,j}), x(i,\tau_{i,j}+1), \ldots, x(i,\tau_{i,j+1}-1), x(i,\tau_{i,j+1})\}$, and thus it is not a local extremum. Consequently, we keep $z(i,t)$ unchanged.

These two different scenarios are illustrated in Figure 3. Figure 3(a) shows the case when $x(i,t-1)$ is not a local extremum, the reference point is thus kept unchanged and the point $x(i,t-1)$ is removed from memory space (which will be further discussed in Section 2.1.2). On the contrary, Figure 3(b) shows the case when $x(i,t-1)$ is indeed a local extremum, the reference point is updated as $z(i,t) = x(i,t-1)$.

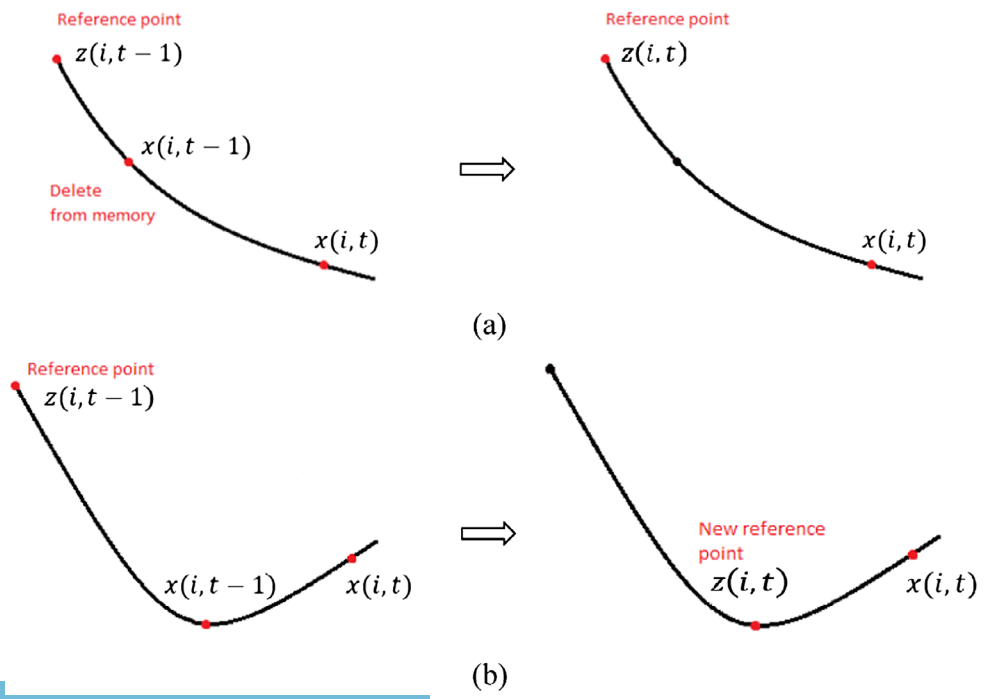

Figure 3. Illustration of updating the value of reference point under two scenarios: (a) when $x(i,t-1)$ is not a local extremum and (b) when $x(i,t-1)$ is a local extremum.

2.1.2. Updating data points based on the memory space constraint

As simulation data are generated much faster than written into the hard disk, many data points have to be temporally stored in the memory space before they can be written into the hard disk. However, due to the limited memory capacity compared with the large amount of continuously generated data, it is impossible to keep all the detected local extrema from Section 2.1.1 in the memory space. In other words, the system must dynamically 'throw away' some local extrema that are temporally stored in the memory space, in order to leave enough rooms to record more significant local extrema in real time. In this subsection, we propose an updating strategy such that the local extrema with large shift indicators can be preserved in the memory space.

Please note that the monitoring strategy proposed in Section 2.1.1 only requires knowing three data points $z(i,t), x(i,t-1), x(i,t)$ in order to detect local extrema in the ith data stream at time t. Based on this observation, we partition the data points that are temporally stored in the memory space $\mathcal{M}(t)$ at time t into two subsets:

$$\mathcal{M}(t) = \mathcal{M}_1(t) \cup \mathcal{M}_2(t). \tag{9}$$

Here, $\mathcal{M}_1(t)$ includes all data points that are necessary for detecting the local extrema. Thus, at the beginning of time t,

$$\mathcal{M}_1(t) = \{z(i,t), x(i,t-1), x(i,t) : i = 1, 2, \ldots, m\}. \tag{10}$$

$\mathcal{M}_2(t)$ contains the already detected local extrema at the beginning of time t:

$$\mathcal{M}_2(t) \subseteq \left\{ x\left(i, \tau_{i,j}\right), d\left(i, \tau_{i,j}\right) : i = 1, 2, \ldots, m, 1 \le \tau_{i,j} \le t-2 \right\}, \tag{11}$$

where $d(i, \tau_{i,j})$ is the shift indicator as defined in Equation (4). Please note that here $\tau_{i,j} \le t-2$ is because at the beginning of time t, we have not yet checked whether $x(i,t-1)$ is a local extremum or not. Below, we discuss how to adaptively manage and update the two subsets $\mathcal{M}_1(t)$ and $\mathcal{M}_2(t)$ separately.

First, recall that at time t, our proposed monitoring scheme in Section 2.1.1 can determine whether the data point $x(i,t-1)$ is a local extremum based on the set $\mathcal{M}_1(t)$. In the case that $x(i,t-1)$ is not a local extremum, it will then be removed from the set $\mathcal{M}_1(t)$ (as shown in Figure 3(a)); otherwise, the data point $x(i,t-1)$ and its shift indicator will be copied from $\mathcal{M}_1(t)$ into the set $\mathcal{M}_2(t)$:

$$\mathcal{M}_2(t) = \mathcal{M}_2(t) \cup \{x(i,t-1), d(i,t-1)\} \tag{12}$$

Second, it should be noticed that the cardinality of $\mathcal{M}_1(t)$ is fixed to be 3 m during the processing of the LES algorithm at any time $t \ge 2$. Thus, the main challenge here is how to adaptively update the set $\mathcal{M}_2(t)$ at each time given the predefined threshold M. Specifically, there are two critical questions to ask: (i) When to discard data points from the set $\mathcal{M}_2(t)$? and (ii) which data points should be discarded from the set $\mathcal{M}_2(t)$?

(i) The capacity of the set $\mathcal{M}_2(t)$ is restricted by both memory and storage constraints. On one hand, with the memory constraint $|\mathcal{M}(t)| \le M$,

$$\left|\mathcal{M}_2(t)\right| = |\mathcal{M}(t)| - 3m \le M - 3m. \tag{13}$$

On the other hand, recall that after time t, there are $\lfloor \frac{t}{k} \rfloor \cdot s$ data points that have already been permanently written into the hard disk. In other words, only $S - \lfloor \frac{t}{k} \rfloor \cdot s$ additional data points can be further recorded before the simulation finishes. This means that the subset $\mathcal{M}_2(t)$ only needs to keep equal to or less than $2\left(S - \lfloor \frac{t}{k} \rfloor \cdot s\right)$ number of data points (local extrema with their shift indicators) as the excessive data points cannot be written into the hard disk. In this way, we derive a dynamic upper bound $U(t)$ for the cardinality of the set $\mathcal{M}_2(t)$:

$$U(t) = \min\left(2\left(S - \lfloor \frac{t}{k} \rfloor \cdot s\right), M - 3m\right). \tag{14}$$

With this bound available, we can then answer the first question that once $|\mathcal{M}_2(t)| > U(t)$, we should start to discard data points from the set $\mathcal{M}_2(t)$.

(ii) To answer the second question, recall that our goal is to detect and save the local extrema with large shift indicators. This can be done by considering the decreasing order statistics of the shift indicators of all the data points from whole data streams that are stored in the set $\mathcal{M}_2(t)$. In particular, suppose at time t, $\mathcal{M}_2(t) = \left\{x\left(I_1\right), \ldots, x\left(I_q\right)\right\}$, where q is the total number of local extrema in the set $\mathcal{M}_2(t)$ and I_l is the pair of index (i,t) for the lth detected local extremum in the set $\mathcal{M}_2(t)$, we sort the absolute value of the shift indicators for all the local extrema in the set $\mathcal{M}_2(t)$ according to the decreasing order: $\left\{\left|d\left(I_{(1)}\right)\right|, \ldots, \left|d\left(I_{(q)}\right)\right|\right\}$. Then, we consider keeping the local extrema witW$U(t)$ shift indicators in the set $\mathcal{M}_2(t)$, i.e. $\mathcal{M}_2(t) = \left\{x\left(I_{(1)}\right), \ldots, x\left(I_{U(t)}\right)\right\}$.

2.1.3. Saving data points into the hard disk

Recall that at time , $n = 1, 2, \ldots \lfloor \frac{T}{k} \rfloor$, s data points that are temporally stored in the memory space can be written into the hard disk. Similar to the approach discussed in Section 2.1.2, we also consider the decreasing order statistics of the shift indicators for all the data points that are stored in the set $\mathcal{M}_2(t)$, and choose the data points with the largest s shift indicators to write into the hard disk. In particular, suppose $\{|d(I_{(1)})|, \ldots, |d(I_{(l)})|\}$ are the absolute value of the shift indicators for all the l data points in $\mathcal{M}_2(t)$ that are sorted in the decreasing order, i.e. $|d(I_{(1)})| \geq \ldots \geq |d(I_{(l)})|$. Then, the set F below includes the data points that are written into the hard disk at time $t = n \cdot k, n = 1, 2, \ldots \lfloor \frac{T}{k} \rfloor$:

$$F = \left\{x\left(I_{(j)}\right):, j = 1, 2, \ldots, s\right\}, \tag{15}$$

where $I_{(j)}$ is the pair of index $\left(i_{(j)}, t_{(j)}\right)$ for the local extremum with the jth largest shift indicator. Next, we update the set $\mathcal{L}(t)$ and $\mathcal{M}(t)$:

$$\mathcal{L}(t) = \mathcal{L}(t-1) \cup F, \ \mathcal{M}(t) = \mathcal{M}(t) \backslash F. \tag{16}$$

Please note that in some special cases, the number of significant local extrema in $\mathcal{M}_2(t)$ may be smaller than s, i.e. $|\mathcal{M}_2(t)| < s$ (especially in the beginning stage of the simulations). To fully utilize the storage resource, in such a case, we choose additional data points from the set $\mathcal{M}_1(t)$ to save into the hard disk. Specifically, we sort the current data points in $\mathcal{M}_1(t)$based on $\{|x(1,t) - z(1,t)|, \ldots, |x(m,t) - z(m,t)|\}$ and select the top $s - |\mathcal{M}_2(t)|$ data points to F.

2.2. An illustrative example for the LES algorithm

To better illustrate the idea of the proposed LES algorithm, we consider an example that involves two data streams $\boldsymbol{x}(1,\cdot)$ and $\boldsymbol{x}(2,\cdot)$, which are continuously and sequentially generated from a simulation study. Figure 4 shows the generated data points of these two data streams with the first $T = 10$ time-steps. Due to the resource constraints, we assume that at every $k = 5$ timesteps, only $s = 3$ data points can be written into the hard disk and only $M = 12$ data points can be temporally stored in memory at any time. Based on this problem setting, we can see that the total number of data points that can be written into the hard disk is $S = s\lfloor \frac{T}{k} \rfloor = 6$ and the saving ratio is $r = \frac{S}{mT} = \frac{3}{10}$.

Table 2 shows the detailed steps of the proposed LES algorithm when it is implemented based on the example described above. In particular, the columns '$x(i,t)$', '$z(i,t)$' and '$d(i,t)$' record the simulated data value, the reference point and the shift indicator of $\boldsymbol{x}(i,\cdot)$ at each time t, respectively. In addition, at each time t, Table 2 shows the details of the data points stored in $\mathcal{M}_1(t)$, $\mathcal{M}_2(t)$ and the total number of data points in $\mathcal{M}(t)$. It should be noticed that the memory space used by $\boldsymbol{x}(1,\cdot)$ keeps increasing with time, while the memory space used by $\boldsymbol{x}(2,\cdot)$ does not. This is because $\boldsymbol{x}(2,\cdot)$ is monotonically

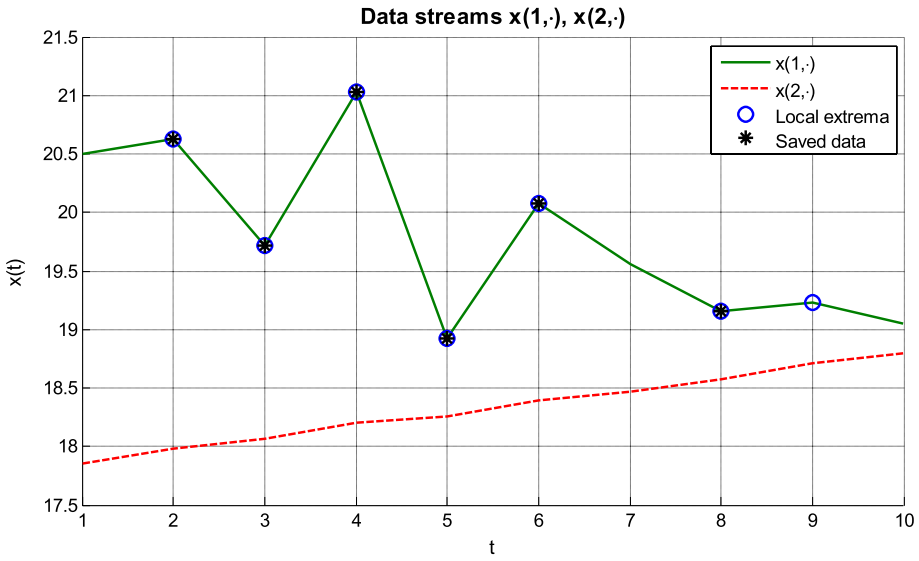

Figure 4. Plot of the generated two data streams in the illustrative example.

increasing from time $t = 1$ to $t = 10$, so there are no local extrema detected by the LES algorithm. On the contrary, since most data points in $\boldsymbol{x}(1, \cdot)$ are local extrema, the LES algorithm aims to 'remember' those local extrema as much as possible. In particular, once the new data point $x(i, t)$ is generated at time t, the LES algorithm stores the data point in $\mathcal{M}_1(t)$ and then determines if it is a local extremum at the following timestep.

Recall that at every $k = 5$ timesteps, three data points can be written in the hard disk. For example, at time $t = 5$, $\{x(1, 2), x(1, 3), x(1, 4)\}$ are selected to be written into the hard disk as they have the largest shift indicators and then removed from $\mathcal{M}_2(t)$ (bold in Table 2). Similarly, at time $t = 10$, the data points $\{x(1, 5), x(1, 6), x(1, 8)\}$ have the largest magnitude of shift indicators (bold in Table 2) and thus they are selected to be written into the hard disk and then removed from $\mathcal{M}_2(t)$. In addition, from Table 2 it can be seen that at $t = 10$, the number of data points stored in $\mathcal{M}_2(t)$ exceeds our derived bound $U(t) = \min(2(S - \lfloor \frac{t}{k} \rfloor \cdot s), M - 3m) = 0$ in Equation (14). Consequently, all data points have to be removed.

In Figure 4, the data points marked in circles and stars are the local extrema and the selected data points written in the hard disk by using our proposed LES algorithm, respectively. All the data points that the LES algorithm saved are from $\boldsymbol{x}(1, \cdot)$ since (i) there is no local extremum detected in the second stream yet; and (ii) the shift magnitude of the second data stream is not large enough compared to the first data stream. This is a reasonable result as if the second data stream still monotonically increases, we should wait for saving until the data stream reaches its local maximum. In such a case, it is more preferred to record the fluctuation of the first data stream instead. In this example, we can see that the most significant six local extrema are $x(1, 5)$, $x(1, 4)$, $x(1, 6)$, $x(1, 8)$, $x(1, 3)$ and $x(1, 2)$ since the sorted shift indicators of all local extrema are $|d(1, 5)| > |d(1, 4)| > |d(1, 6)| > |d(1, 8)| > |d(1, 3)| > |d(1, 2)| > |d(1, 9)|$. Therefore, this results show that LES algorithm can effectively detect and save the most informative local extrema in real time.

2.3. Flow chart of the LES algorithm

Figure 5 provides an overview of the LES algorithm.

Table 2. Details of implementing the LES algorithm for the illustrative example.

	$x(1,\cdot)$			$x(2,\cdot)$			
t	$x(1,t)$	$z(1,t)$	$d(1,t)$	$x(2,t)$	$z(2,t)$	$d(2,t)$	$\lvert\mathcal{M}(t)\rvert$
1	20.50	20.50	0	17.86	17.86	0	4
	$\mathcal{M}_1(t) = \{z(1,1) = 20.50, x(1,1) = 20.50, z(2,1) = 17.86, x(2,1) = 17.86\}$						
	$\mathcal{M}_2(t) = \{\emptyset\}$						
2	20.63	20.50	0.13	17.99	17.86	0.13	6
	$\mathcal{M}_1(t) = \{z(1,2) = 20.50, x(1,1) = 20.50, x(1,2) = 20.63, z(2,2) = 17.86, x(2,1) = 17.86, x(2,2) = 17.99\}$						
	$\mathcal{M}_2(t) = \{\emptyset\}$						
3	19.72	20.63	−0.91	18.07	17.86	0.21	8
	$\mathcal{M}_1(t) = \{z(1,3) = 20.63, x(1,2) = 20.63, x(1,3) = 19.72, z(2,3) = 17.86, x(2,2) = 17.99, x(2,3) = 18.07\}$						
	$\mathcal{M}_2(t) = \{x(1,2) = 20.63, d(1,2) = 0.13\}$						
4	21.03	19.72	1.31	18.21	17.86	0.35	10
	$\mathcal{M}_1(t) = \{z(1,4) = 19.72, x(1,3) = 19.72, x(1,4) = 21.03, z(2,4) = 17.86, x(2,3) = 18.07, x(2,4) = 18.21\}$,						
	$\mathcal{M}_2(t) = \{x(1,2) = 20.63, d(1,2) = 0.13, x(1,3) = 19.72, d(1,3) = -0.91\}$,						
5	18.92	21.03	−2.11	18.26	17.86	0.40	12
	$\mathcal{M}_1(t) = \{z(1,5) = 21.03, x(1,4) = 21.03, x(1,5) = 18.92, z(2,5) = 17.86, x(2,4) = 18.21, x(2,5) = 18.26\}$						
	$\mathcal{M}_2(t) = \{\mathbf{x(1,2) = 20.63, d(1,2) = 0.13, x(1,3) = 19.72, d(1,3) = -0.91, x(1,4) = 21.03, d(1,4) = 1.31}\}$						
Write the data points $x(1,2), x(1,3), x(1,4)$ into the hard disk. $\mathcal{M}_2(t) = \emptyset$, $\lvert\mathcal{M}(t)\rvert = 6$							
6	20.08	18.92	1.16	18.39	17.86	0.53	8
	$\mathcal{M}_1(t) = \{z(1,6) = 18.92, x(1,5) = 18.92, x(1,6) = 20.08, z(2,6) = 17.86, x(2,5) = 18.26, x(2,6) = 18.39\}$,						
	$\mathcal{M}_2(t) = \{x(1,5) = 18.92, d(1,5) = -2.11\}$						
7	19.56	20.08	−0.52	18.46	17.86	0.60	10
	$\mathcal{M}_1(t) = \{z(1,7) = 20.08, x(1,6) = 20.08, x(1,7) = 19.56, z(2,7) = 17.86, x(2,6) = 18.39, x(2,7) = 18.46\}$						
	$\mathcal{M}_2(t) = \{x(1,5) = 18.92, d(1,5) = -2.11, x(1,6) = 20.08, d(1,6) = 1.16\}$						
8	19.15	20.08	−0.93	18.58	17.86	0.72	10
	$\mathcal{M}_1(t) = \{z(1,8) = 20.08, x(1,7) = 19.56, x(1,8) = 19.15, z(2,8) = 17.86, x(2,7) = 18.46, x(2,8) = 18.58\}$						
	$\mathcal{M}_2(t) = \{x(1,5) = 18.92, d(1,5) = -2.11, x(1,6) = 20.08, d(1,6) = 1.16\}$						
9	19.23	19.15	0.08	18.71	17.86	0.85	12
	$\mathcal{M}_1(t) = \{z(1,9) = 19.15, x(1,8) = 19.15, x(1,9) = 19.23, z(2,9) = 17.86, x(2,8) = 18.58, x(2,8) = 18.71\}$						
	$\mathcal{M}_2(t) = \{x(1,5) = 18.92, d(1,5) = -2.11, x(1,6) = 20.08, d(1,6) = 1.16, x(1,8) = 19.15, d(1,8) = -0.93\}$						
10	19.06	19.15	−0.09	18.79	17.86	0.93	14
	$\mathcal{M}_1(t) = \{z(1,10) = 19.15, x(1,9) = 19.23, x(1,10) = 19.06, z(2,10) = 17.86, x(2,9) = 18.71, x(2,10) = 18.79\}$						
	$\mathcal{M}_2(t) = \{\mathbf{x(1,5) = 18.92, d(1,5) = -2.11, x(1,6) = 20.08, d(1,6) = 1.16, x(1,8) = 19.15, d(1,8) = -0.93, x(1,9) = 19.23, d(1,9) = 0.08}\}$						
Write the data points $x(1,5), x(1,6), x(1,8)$ into the hard disk.$\mathcal{M}_2(t) = \emptyset$, $\lvert\mathcal{M}(t)\rvert = 6$							
Exceed memory limits, remove $x(1,9)$ and $d(1,9)$ from memory space.							

3. Results

In this section, we will thoroughly evaluate the performance of the proposed LES algorithm and further compare with some baseline methods based on the climate simulation data-sets.

Please note that the problem considered in this paper is an open question which has not been well explored in the existing literature. To evaluate the effectiveness of our proposed strategy, besides the CFS algorithm that is currently used in ORNL, we also considered two additional possible approaches as the baseline methods in this case study. Recall that our proposed LES algorithm has two important components: (1) identifying the local extrema, and (2) ranking the shift indicators based on the decreasing order to determine the saving priority of the local extrema. To demonstrate that both components

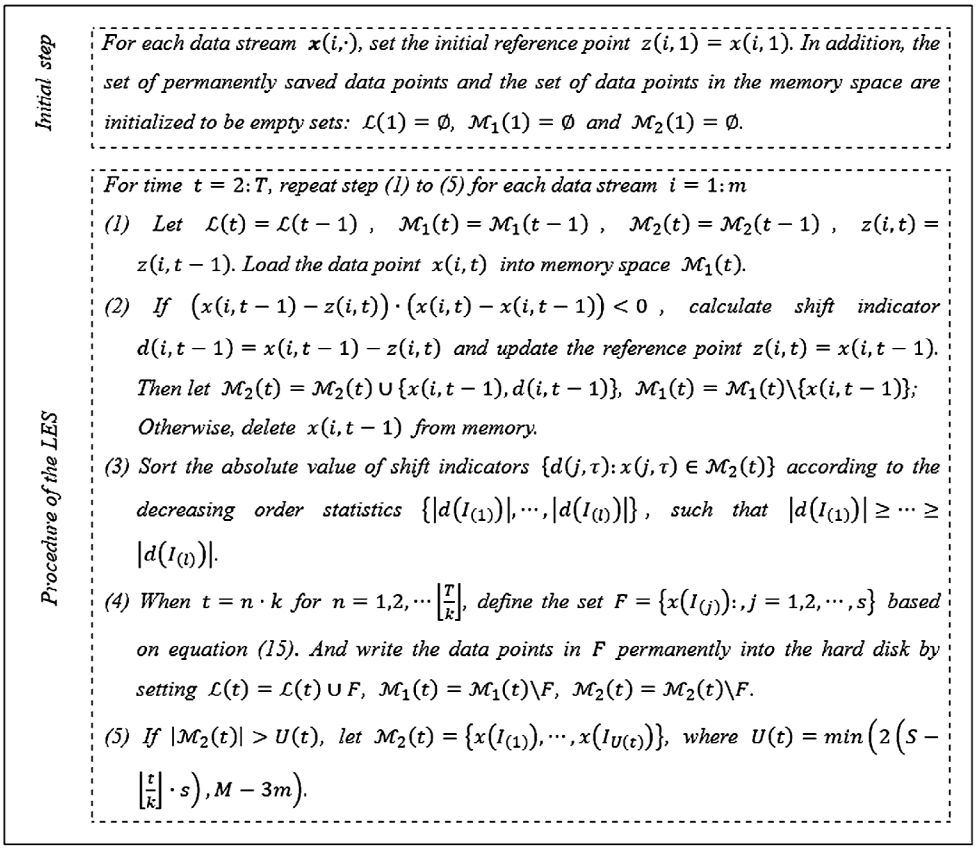

Figure 5. A flow chart of the LES algorithm.

are necessary, we thus consider two additional baseline methods that focus on only one of these two components. Specifically, in these two baseline methods, we still calculate shift indicators and update data points based on the memory constraints as introduced in Sections 2.1.1 and 2.1.2. However, in the first baseline method, we consider a parameter d_0 which refers to the smallest interested magnitude of changes for detection specified by practitioners and decide to save a local extremum $x(i, \tau_{ij})$ immediately if the absolute value of its shift indicator $|d(i, \tau_{ij})|$ is larger than the predefined threshold d_0. We call this method the Threshold-based Saving (TS) algorithm. As for the second baseline method, we simply sort the magnitude of shift indicators for all generated data points and decide to save the data based on the decreasing order of the shift indicators. Thus, unlike our proposed LES algorithm, this baseline method may not only save local extrema. We call this method the Largest Shift Indicator (LSI) algorithm. In this case study, we will evaluate the performance of the LES algorithm and compare with the aforementioned three baseline methods, i.e. the CFS, TS and LSI algorithms.

Recall that our objective is to detect and select the local extreme with large shift indicators to write into the hard disk during the real-time simulations. Specifically, according to Equation (6), this means to find the set $\mathcal{L}(T)$ such that the cardinality $|E \cap \mathcal{L}(T)|$ is maximized. In particular, we will thoroughly evaluate the performance of our proposed methods under different values of the saving ratio r in the three simulated data-sets mentioned above. We focus on a performance measure $\frac{|E \cap \mathcal{L}(T)|}{|E|}$, which characterizes the ratio of the number of significant local extrema that are finally saved in the hard disk over the total number of significant local extrema generated from simulations. In this way, a higher value in this performance measure $\frac{|E \cap \mathcal{L}(T)|}{|E|}$ indicates a more efficient algorithm.

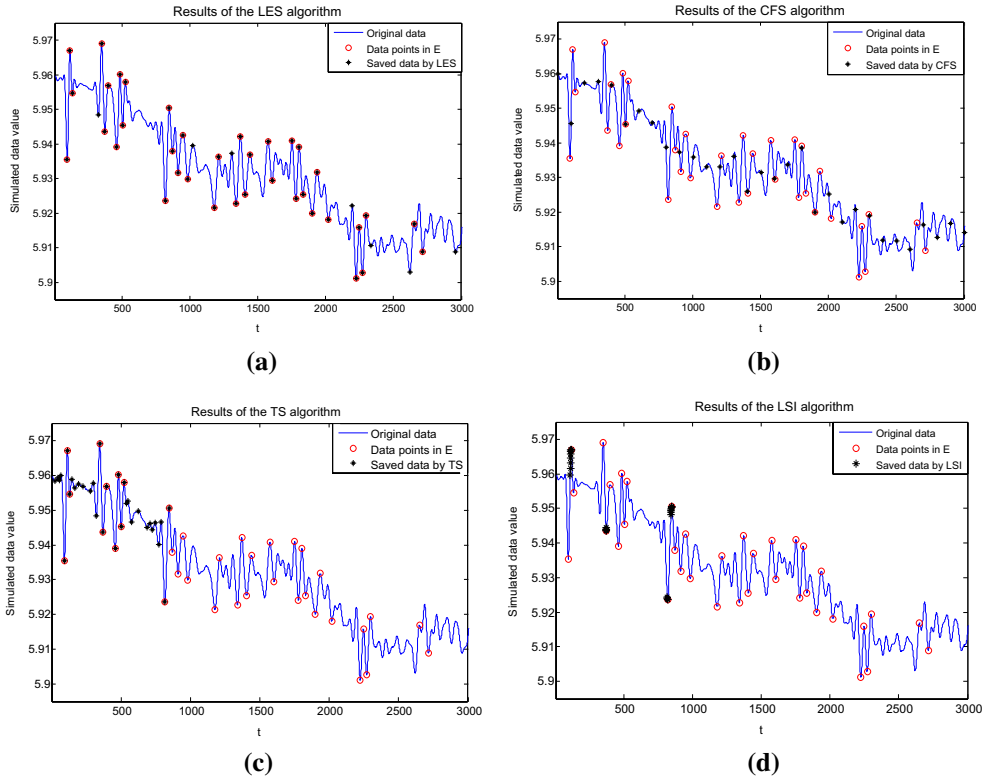

Figure 6. A comparison between (a) the LES algorithm, (b) the CFS algorithm, (c) the TS algorithm, and (d) the LSI algorithm for one data stream when the saving ratio is 1%.

Figure 6 illustrates a comparison between the LES, CFS, TS and LSI algorithms for one data stream generated from the low-resolution simulations, when the saving ratio is $r = 0.01$ (i.e. only 1% of the original data generated from simulations can be written into the hard disk). In this figure, the saved data are marked by 'star' and the data points in E (i.e. the significant local extrema) are highlighted in circles. From this example in Figure 6, we can see that the data points with the large shift indicators are automatically detected and saved into the hard disk by the proposed LES algorithm, while only the data points generated at the specific time with a fixed frequency can be saved by the CFS strategy. For the TS algorithm, Figure 6 shows the saving results when the pre-specified threshold d_0 is set to be 0.015. In that case, the saved data are all located at the beginning stage of the simulation, as the pre-defined threshold is too small and thus uses up all the saving resource long before the simulation ends. Likewise, we also find that d_0 cannot be set too large; otherwise, only few data points can be finally saved into the hard disk. In theory, pre-specifying an appropriate threshold d_0 is a challenging task as the distribution and the values of each data stream are often unknown before the simulation starts. Thus, without considering the rank of the shift indicators, the TS algorithm fails to effectively utilize the available saving resources during the simulation process. As for the LSI algorithm, although it focuses on saving the data points based on the decreasing order of the shift indicators, the result shows that the saved points are clustered around some local extrema and within a limited temporal window, which still fails to effectively leverage the available saving resources over the entire span of the simulation runs. Thus, this result shows that it is necessary to consider both detecting local extrema and ranking the shift indicators to determine the saving priority in real-time simulations.

Figure 7 further shows the saving results of the LES algorithm under saving ratios of $\frac{1}{150}$ and $\frac{1}{200}$ for the data stream shown in Figure 6(a). As the saving ratio gets smaller, both the number of data points

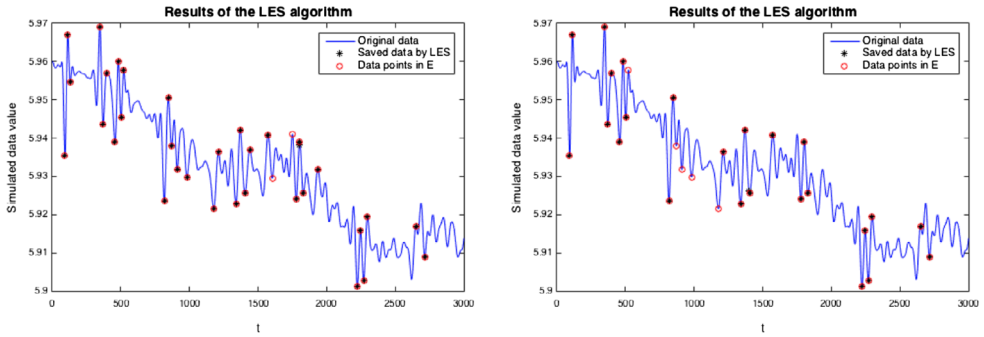

Figure 7. Saving results of the LES algorithm for saving ratios of (a) $r = 1/150$ and (b) $r = 1/200$ for one data stream in low resolution data.

Table 3. Performance comparison of the LES, CFS, TS, and LSI algorithms based on the medium-resolution data-set under different saving ratios $\frac{1}{100}$, $\frac{1}{150}$ and $\frac{1}{200}$.

		LES			CFS	
$r = \frac{1}{100}$	0.77	T	F	0.02	T	F
	S	366,959	109,201	S	8163	467,997
	D	109,201	47,030,639	D	467,997	46,671,843
$r = \frac{1}{150}$	0.84	T	F	0.01	T	F
	S	266,267	51,173	S	4297	3,13,143
	D	51,173	47,247,387	D	313,143	46,985,417
$r = \frac{1}{200}$	0.88	T	F	0.01	T	F
	S	209,744	28,336	S	2722	235,358
	D	28,336	47,349,584	D	235,358	47,142,562
		TS			LSI	
$r = \frac{1}{100}$	0.54	T	F	0.03	T	F
	S	255,262	220,898	S	12,570	463,590
	D	220,898	46,918,942	D	463,590	46,676,250
$r = \frac{1}{150}$	0.36	T	F	0.02	T	F
	S	113,406	204,034	S	6210	311,230
	D	204,034	47,094,526	D	311,230	46,987,330
$r = \frac{1}{200}$	0.27	T	F	0.02	T	F
	S	65,393	172,687	S	3835	234,245
	D	172,687	47,205,233	D	234,245	47,143,675

in E and the number of saved data points decrease. Even when the saving ratio is as small as $\frac{1}{200}$, the LES algorithm is still capable of capturing most of the extrema of the data stream.

Table 3 further summarizes and compares the results of the LES, CFS, TS and the LSI algorithms based on the medium-resolution data-set under different saving ratios $r = \frac{1}{100}, \frac{1}{150}$, and $\frac{1}{200}$. For each combination, there is a 2×2 block matrix that records the performance of each algorithm, in which the rows represent the number of data points that an algorithm chooses to save (S) and discard (D), while the columns list the number of data points that are actually in the set E (T) and not (F). For example in Table 3, when $r = \frac{1}{100}$, among the total 476,160 significant local extrema that are generated from the simulations, there are 366,959 significant local extrema that are finally saved in the hard disk, while 109,201 significant local extrema are missed by the LES algorithm. This is because as only limited memory space is available, certain local extrema with relatively smaller shift indicators have to be thrown away to make sure more significant local extrema can be stored in the memory space and eventually written into the hard disk. Please note that in this case, the LES algorithm indeed saved 109,201 data points that are not significant local extrema (i.e. not in the set E). This is because at the beginning stage of the simulation, there may not be enough significant local extrema generated

Table 4. The performance of the LES algorithm under different resolutions of the data-sets when the saving ratios are $\frac{1}{100}$, $\frac{1}{150}$ and $\frac{1}{200}$.

	Low resolution			Medium resolution			High resolution		
$r = \frac{1}{100}$	0.88	T	F	0.77	T	F	0.61	T	F
	S	50,494	7106	S	366,959	109,201	S	2,496,887	1,567,369
	D	7106	5,695,294	D	109,201	47,030,639	D	1,567,369	400,793,975
$r = \frac{1}{150}$	0.90	T	F	0.84	T	F	0.71	T	F
	S	34,620	3780	S	266,267	51,173	S	1,936,829	772,675
	D	3780	517,820	D	51,173	47,247,387	D	772,675	402,943,421
$r = \frac{1}{200}$	0.91	T	F	0.88	T	F	0.78	T	F
	S	26,139	2661	S	209,744	28,336	S	1,574,978	457,150
	D	2661	5,728,539	D	28,336	47,349,584	D	457,150	403,936,322

and stored in the memory space, and thus some other data points that are ranked high based on the descending order of the shift indicators (see Equation (15)) are also selected to write into the hard disk.

To consistently describe these cells, we define the entries of the 2×2 table by both their row and column names. The number of significant local extrema that is successfully saved by a saving strategy (i.e. the entry at the intersection of row 'S' and column 'T') is denoted as 'ST', and the number of insignificant data points that is discarded by a saving strategy is called 'DF'. These two diagonal categories thus show the effectiveness of a saving strategy, which are analogous to the 'True Positives' and 'True Negatives' in a confusion matrix. Likewise, 'SF' denotes the number of saved data points which are not in E and 'DT' denotes the number of significant local extrema that a saving strategy fails to save. These two off-diagonal entries are similar to the 'False Positives' and 'False Negatives' in a confusion matrix. Consequently, the performance measure $\frac{|E \cap \mathcal{L}(T)|}{|E|}$ is equivalent to the True Positive Rate (TPR) in a confusion matrix and the value of this measure is shown on the upper left corner of each 2×2 block.

From Table 3, the LES algorithm outperforms all the baseline methods in the sense that each 'confusion matrix' of the LES algorithm has much larger diagonal entries and smaller off-diagonal entries than that of the CFS, TS and LSI algorithms in all the considered scenarios. Also, the LES algorithm obtains the largest value in the performance measure $\frac{|E \cap \mathcal{L}(T)|}{|E|}$ at different values of the saving ratio r. This result shows that our proposed LES algorithm successfully manages the limited resources to maximize the detection and saving powers for these significant local extrema that are generated from real-time simulations. Recall that the saving ratio r characterizes how limited the storage resource is. According to the practical settings in the simulation model, here we consider the range of the saving ratio from $\frac{1}{100}$ to $\frac{1}{200}$. From Table 3, we can see that when r decreases, it means that the storage capacity becomes more restricted, and thus it is rational to see that ST and SF both decrease. However, a surprising result here is that the performance measure $\frac{|E \cap \mathcal{L}(T)|}{|E|}$ actually increases as the value of r decreases in our proposed LES algorithm. One possible reason is that as the saving ratio r decreases, the cardinality of both the target set E and the set of saved data points decreases accordingly. However, at a lower saving ratio, the LES algorithm waits longer before deciding to save a batch of data and thus it captures data points with larger shift indicators. Therefore, even though the number of saved data points decreases, the proportion of saved data in set E increases. This means even when the storage capacity becomes more restricted, the LES algorithm can still effectively leverage the available resources to dynamically capture the most significant local extrema.

To understand how the data resolution influences the performance of the LES algorithm, Table 4 shows the performance of the LES algorithm under different resolutions of the data-sets when the saving ratios are $\frac{1}{100}$, $\frac{1}{150}$ and $\frac{1}{200}$. As the resolution of the data increases, the memory capacity becomes more restricted, and thus we expect to see that the value of $\frac{|E \cap \mathcal{L}(T)|}{|E|}$ decreases.

Acknowledgment

The submitted manuscript is based upon work, authored in part by contractors [UT-Battelle LLC, manager of Oak Ridge National Laboratory (ORNL)], and supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program. Accordingly, the U.S. Government retains a

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Science Foundation [grant number NSF CMMI-1362529]; the Air Force Office of Scientific Research; National Natural Science Foundation of China [grant number 71402133], [grant number 71602155], [grant number 71572138], [grant number 11501209].

Notes on contributors

Xiaochen Xian received the BS degree in Mathematics from Zhejiang University, Hangzhou, China, in 2014. Currently she is a PhD candidate at the Department of Industrial and Systems Engineering, University of Wisconsin-Madison. Her research interests are focused on high-dimensional monitoring and optimal maintenance strategies.

Richard K. Archibald is a staff scientist in the Computational Mathematics group at ORNL. Archibald received both his BS in physics and his MS in mathematics from the University of Alberta in Edmonton, Canada in 1996 and 1998, respectively. He obtained his PhD in mathematics from the University of Alberta-Edmonton in 2002. Archibald's current research interests include developing uncertainty quantification algorithms for climate models, designing algorithms for the next generation of high-performance architecture, and establishing long-time integration steps for the dynamical core of climate models at high resolution.

Benjamin Mayer is a member of the Earth Systems Modeling Group in the Climate Change Science Institute at ORNL. Benjamin received his Master's degree in Computer Science from the University of Minnesota with work in creation of data mining algorithms for medical and network intrusion detection, parallel computing (language design, implementation and instruction), as well as databases. His work is interfacing with Scientists and Systems people to discover opportunities for easier and faster operations and to implement these discoveries.

Kaibo Liu received the BS degree in industrial engineering and engineering management from the Hong Kong University of Science and Technology in 2009, and the MS degree in statistics and the PhD degree in industrial engineering from the Georgia Institute of Technology in 2011 and 2013. Currently, he is an assistant professor at the Department of Industrial and Systems Engineering, University of Wisconsin-Madison. His research interests are focused on data fusion for process modeling, monitoring, diagnosis and prognostics.

Jian Li received the BS degree in automation from Tsinghua University in 2006, and the PhD degree in Industrial Engineering and Logistics Management from the Hong Kong University of Science and Technology in 2012. Currently, he is an associate professor at School of Management, Xi'an Jiaotong University. His research interests are focused on statistical data mining and statistical process control.

References

Archibald, R., Evans, K. J., Drake, J., & White, J. B. (2010). Multiwavelet discontinuous galerkin-accelerated exact linear part (ELP) method for the shallow-water equations on the cubed sphere. *Monthly Weather Review, 139*(2), 457–473.

Baker, A. H., Xu, H., Dennis, J. M., Levy, M. N., Nychka, D., Mickelson, S. A., … Wegener, A. (2014). A methodology for evaluating the impact of data compression on climate simulation data. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing, HPDC '14*, New York, NY, USA, ACM, 203–214.

Bennett, J. C., Abbasi, H., Bremer, P. T., Grout, R., Gyulassy, A., Jin, T., … Pebay, P. (2012). Combining *in situ* and in-transit processing to enable extreme-scale scientific analysis. *2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Salt Lake City, UT, 1–9.

Bennett, J. C., Comandur, S., Pinar, A., & Thompson, D. (2013). *Sublinear algorithms for in-situ and in-transit data analysis at the extreme-scale (No. SAND2013-3641C)*. Livermore, CA: Sandia National Laboratories (SNL-CA).

Castruccio, S., & Genton, M. G. (2016). Compressing an ensemble with statistical models: An algorithm for global 3D spatio-temporal temperature. *Technometrics, 58*(3), 319–328.

Collins, W. D., Johansen, H., Evans, K. J., Woodward, C. S., & Caldwell, P. M. (2015). Progress in fast, accurate multi-scale climate simulations. *Procedia Computer Science, 51*, 2006–2015.

Gerhard, N., Iacono, F., May, G., Müller, S., & Schäfer, R. (2014). A high-order discontinuous Galerkin discretization with multiwavelet-based grid adaptation for compressible flows. *Journal of Scientific Computing, 62*(1), 25–52.

Huang, S., Kong, Z., & Huang, W. (2014). High-dimensional process monitoring and change point detection using embedding distributions in reproducing kernel Hilbert space. *IIE Transactions, 46*(10), 999–1016.

Lakshminarasimhan, S., Shah, N., Ethier, S., Klasky, S., Latham, R., Ross, R., & Samatova, N. F. (2011). Compressing the incompressible with ISABELA: *In situ* reduction of spatio-temporal data. *Euro-Par 2011 Parallel Processing* (pp. 366–379). Berlin Heidelberg: Springer.

Lautenschlager, M., & Reinke, M. (2012). *Climate and environmental database systems, 386*. New York, NY: Springer Science & Business Media.

Liu, S., Huang, X., Fu, H., Yang, G., & Song, Z. (2013). Data reduction analysis for climate data sets. *International Journal of Parallel Programming, 43*(3), 508–527.

Liu, K., Mei, Y., & Shi, J. (2015). An adaptive sampling strategy for online high-dimensional process monitoring. *Technometrics, 57*(3), 305–319.

Mayer, B., Worley, P. H., da Silva, R. F., & Gaddis, A. L. (2015). Climate science performance, data and productivity on Titan. *Proceeding of Cray User Group Conference*, Chicago, Illinois.

Meehl, G. A., Karl, T., Easterling, D. R., Changnon, S., Pielke, R., Jr., Changnon, D., … Mearns, L. O. (2000). An introduction to trends in extreme weather and climate events: Observations, socioeconomic impacts, terrestrial ecological impacts, and model projections. *Bulletin of the American Meteorological Society, 81*(3), 413–416.

Ning, X., & Tsung, F. (2012). A density-based statistical process control scheme for high-dimensional and mixed-type observations. *IIE Transactions, 44*(4), 301–311.

Palmer, T. (2014). Climate forecasting: Build high-resolution global climate models. *Nature, 515*(7527), 338–339.

Rahmstorf, S., & Coumou, D. (2011). Increase of extreme events in a warming world. *Proceedings of the National Academy of Sciences, 108*(44), 17905–17909.

Williamson, D. L., Drake, J. B., Hack, J. J., Jakob, R., & Swarztrauber, P. N. (1992). A standard test set for numerical approximations to the shallow water equations in spherical geometry. *Journal of Computational Physics, 102*(1), 211–224.