Supplement: Uncertainty Visualization of Critical Points of 2D Scalar Fields for Parametric and Nonparametric Probabilistic Models

Tushar M. Athawale, Zhe Wang, David Pugmire, Kenneth Moreland, Qian Gong, Scott Klasky, Chris R. Johnson, and Paul Rosen

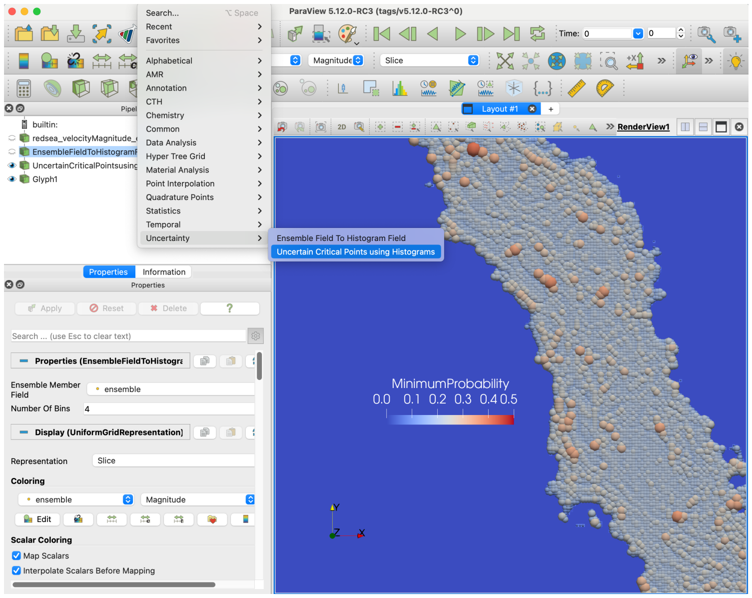


Fig. 1: Demonstration of our critical point uncertainty filter inside ParaView [2] to provide near-real-time results for the Red Sea ensemble dataset [6] with computations performed using VTK-m [5] parallel code on a GPU backend.

Abstract—This paper presents a novel end-to-end framework for closed-form computation and visualization of critical point uncertainty in 2D uncertain scalar fields. Critical points are fundamental topological descriptors used in the visualization and analysis of scalar fields. The uncertainty inherent in data (e.g., observational and experimental data, approximations in simulations, and compression), however, creates uncertainty regarding critical point positions. Uncertainty in critical point positions, therefore, cannot be ignored, given their impact on downstream data analysis tasks. In this work, we study uncertainty in critical points as a function of uncertainty in data modeled with probability distributions. Although Monte Carlo (MC) sampling techniques have been used in prior studies to quantify critical point uncertainty, they are often expensive and are infrequently used in production-quality visualization software. We, therefore, propose a new end-to-end framework to address these challenges that comprises a threefold contribution. First, we derive the critical point uncertainty in closed form, which is more accurate and efficient than the conventional MC sampling methods. Specifically, we provide the closed-form and semianalytical (a mix of closed-form and MC methods) solutions for parametric (e.g., uniform, Epanechnikov) and nonparametric models (e.g., histograms) with finite support. Second, we accelerate critical point probability computations using a parallel implementation with the VTK-m library, which is platform portable. Finally, we demonstrate the integration of our implementation with the ParaView software system to demonstrate near-real-time results for real datasets.

Index Terms—Topology, uncertainty, critical points, probabilistic analysis

1 Introduction

We divide this supplement into four parts. First, we present the detailed algorithms and illustrations for critical point probability computation for the four-pixel neighborhood case (Sec. 4.2 of the main paper). Second, we show quantitative and qualitative correctness of our algorithms for the uniform, Epanechnikov, and histogram (closed-form and semi-analytical) models through comparisons with the Monte Carlo (MC) solutions. Third, we present quantitative evaluation for the climate dataset [4] used in the main paper. Finally, we show seamless integration of our parallel VTK-m [5] implementation inside ParaView [2].

- *Tushar M. Athawale, Zhe Wang, David Pugmire, Kenneth Moreland, Qian Gong, and Scott Klasky are with the Oak Ridge National Laboratory. E-mail:{athawaletm, wangz, pugmire, morelandkd, gongq, klasky}@ornl.gov*
- *Chris R. Johnson and Paul Rosen are with the University of Utah E-mail: {crj, paul.rosen}@sci.utah.edu*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

2 Critical Point Probability Computation for Four-Pixel Neighborhood

2.0.1 Local Minimum Probability

The probability of point p being a local minimum, $\Pr(p = l_{min})$, can be computed by integrating the joint probability $\Pr_{joint}$ over its support where random variable X_1 is simultaneously smaller than all neighboring random variables (i.e., $X_2, \ldots, X_5$). Mathematically, the local minimum probability computation can be represented as follows:

$$\begin{aligned}
&\Pr(p = l_{min}) \\
&= \Pr[(X_1 < X_2) \text{ and } (X_1 < X_3) \text{ and } (X_1 < X_4) \text{ and } (X_1 < X_5)] \\
&= \int_{x_1=a_1}^{x_1=b_{min}} \int_{x_2=max(x_1,a_2)}^{x_2=b_2} \cdots \int_{x_5=max(x_1,a_5)}^{x_5=b_5} (\mathrm{Pdf}_{joint})\, \mathrm{d}x, \\
&\text{where } b_{min} = min(b_1, b_2, b_3, b_4, b_5)
\end{aligned} \tag{1}$$

Equation (1) represents the core integration formula for the computation of local minimum probability at a domain position p. We now explain the integral limits in Eq. (1) and our proposed piecewise integration algorithm to compute the formula in Eq. (1).

Limits a_1 and b_{min} of the outer integral in Eq. (1) : The outer integral of Eq. (1) indicates the portion of data range of a random variable X_1 (i.e., $[a_1, b_1]$) that can result in point p being a local minimum. In particular, the portion $[a_1, b_{min}]$ (with $a_1 < b_{min}$) of random variable X_1 can result in point p being a local minimum, where b_{min} denotes the minimum among $b_1, \ldots, b_5$. In contrast, the data range $[b_{min}, b_1]$ for $b_{min} \neq b_1$ cannot result in a point p as a local minimum (l_{min}) because it will be always greater than one of the random variables X_2, X_3, X_4, and X_5 depending on if $b_{min} = b_2$, $b_{min} = b_3$, $b_{min} = b_4$, or $b_{min} = b_5$ respectively. Mathematically, for any value $x_1 \geq b_{min}$, $\Pr(p = l_{min}) = 0$. In the case $b_{min} < a_1$, then $Pr(p = l_{min}) = 0$ because at least one random variable among X_i with $i \in \{2, 3, 4, 5\}$ will be always smaller than X_1.

Limits $max(x_1, a_i)$ and b_i of the inner integrals in Eq. (1): The four inner integrals in Eq. (1) integrate the joint distribution Pdf_{joint} over its support where random variables $X_2, \ldots, X_5$ are simultaneously greater than $x_1 \in [a_1, b_{min}]$ in the outer integral. The inner integral lower limits are $max(x_1, a_i)$ for $i \in \{2, 3, 4, 5\}$. The maximum is taken because the support of a random variable X_i is restricted to $[a_i, b_i]$. Thus, for $x_1 < a_i$ in any inner integral, the entire support $[a_i, b_i]$ with $i \in \{2, 3, 4, 5\}$ will always be greater than x_1, and the inner integral does not depend on the value of x_1. In contrast, for $x_1 > a_i$, the inner integration depends on the value of x_1 because x_1 assumes values in the support of distributions. It is guaranteed that the upper limit of inner integrals in Eq. (1) is greater than their respective lower limit, i.e., $b_i \geq max(x_1, a_i)$, for two reasons. First, for any random variable X_i, we assume $a_i < b_i$. Second, the maximum value of x_1 is equal to $b_{min} = min(b_1, b_2, b_3, b_4, b_5)$ based on the outer integral (see the previous paragraph), and it cannot exceed the upper limits b_i with $i \in \{2, 3, 4, 5\}$ in inner integrals. Depending upon if the $max(x_1, a_i)$ is equal to x_1 or a_i, the integral in Eq. (1) can be simplified and computed differently, which necessitates the evaluation of the integral in Eq. (1) in a piecewise manner, as described next.

Piecewise simplification of Eq. (1): The core integration formula in Eq. (1) can be simplified differently for different subsets of the range of the outer integral (i.e., $[a_1, b_{min}]$). We refer to each subset as a piece P. For a piece P, where $x_1 < a_i$, $\forall i \in \{2, 3, 4, 5\}$, the inner integrals in Eq. (1) attain the range $x_i \in [a_i, b_i]$, $\forall i \in \{2, 3, 4, 5\}$. In other words, the inner integrals do not depend on x_1. Thus, the integration over the entire support of random variables X_i, $\forall i \in \{2, 3, 4, 5\}$, simplifies Eq. (1) to the integration over a marginal distribution of X_1 for the piece P, i.e., $\int_{x_1 \in P} \mathrm{Pdf}_{X_1}(x_1)\, \mathrm{d}x_1$.

For a piece P, where $x_1 \in [a_2, b_2]$ and $x_1 < a_i$, $\forall i \in \{3, 4, 5\}$, the inner integrals in Eq. (1) attain the range $x_2 \in [x_1, b_2]$ and $x_i \in [a_i, b_i]$, $\forall i \in \{3, 4, 5\}$. In this case, only the first inner integral related to the range of random variable X_2 depends upon x_1. Thus, Eq. (1) simplifies as the integration over the joint distribution of X_1 and X_2 for the piece P, i.e., $\int_{x_1 \in P} \mathrm{Pdf}_{X_1}(x_1) \mathrm{Pdf}_{X_2}(x_2)\, \mathrm{d}x_2\, \mathrm{d}x_1$. In summary, various pieces of the outer integral in Eq. (1) can be simplified differently based on if the inner integrals depend on x_1 or not.

```
Function computeLocalMinimumProbability()
    Input: Intervals I = [[a1, b1], ..., [a5, b5]]
    Output: Pr(p = lmin)
    bmin = min(b1, ..., b5)
    if bmin < a1 then
        return 0
    else
        /* Task1: Break the outer integral in
           Eq. (1) into pieces Pi                    */
        Isorted = sort(I) based on start points ai
        ida1 = Index of the interval in Isorted that starts with a1
        idbMin =
          Index of the interval in Isorted that contains bmin

        /* Task2 and Task 3: Compute piece
           integrals                                 */
        localMinimumProbability =
          computePieceIntegrals(ida1, idbMin, Isorted, bmin)

        return localMinimumProbability
end
```

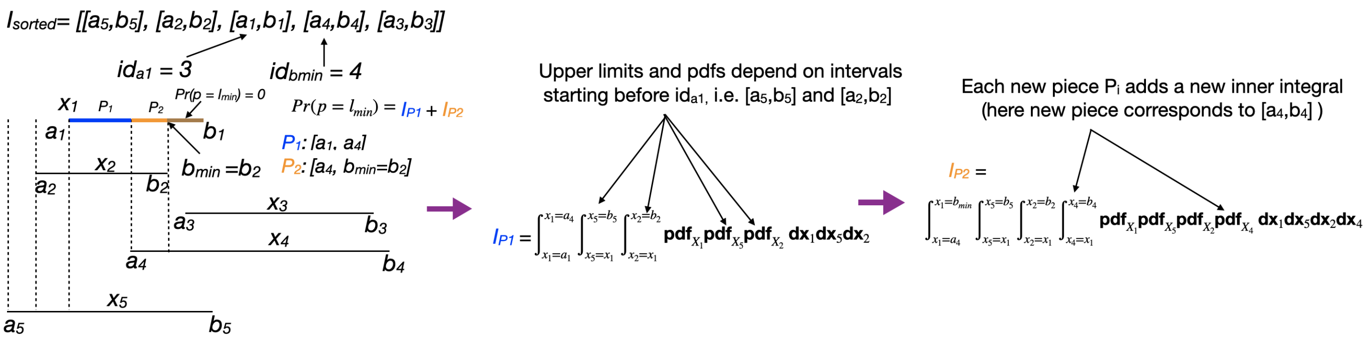
Algorithm 1: Computation of local minimum probability

Algorithm for computing local minimum probability: The computation of the core integration formula in Eq. (1) depends on the ordering of start points a_i with $i \in \{1, 2, 3, 4, 5\}$ and b_{min}. Thus, there are $6! = 720$ permutations of a_i and b_{min}. We, therefore, devise an efficient algorithm that computes the piecewise integrals on the fly depending on the observed permutation of a_i and b_{min} without needing to go through all permutations.

We now describe our Algorithm 1 for computation of local minimum probability at a domain position p (i.e., $\Pr[p = l_{min}]$). The algorithm comprises the three main tasks. **Task 1: Determination of pieces P_i of the outer integral in Eq. (1) needed for performing piecewise integration.** Initially, we compute b_{min}. If $b_{min} < a_1$, then there are no pieces or $\Pr(p = l_{min}) = 0$. If $b_{min} > a_1$, then we sort intervals representing uncertain data ranges (i.e., $[a_i, b_i]$) based on their start points a_i and keep them in the array named I_{sorted}. We note the index of the interval corresponding to $[a_1, b_1]$ in I_{sorted} (referred to as id_{a_1}) and the index of interval in I_{sorted} from the end that contains b_{min} (referred to as $id_{b_{min}}$), as a_1 and b_{min} constitute limits of the outer integral in Eq. (1). Any start points a_i contained in the range of indices id_{a_1} and $id_{b_{min}}$ determine the pieces P_i for integration. This process generalizes to any ordering of a_i to determine the pieces p_i of the outer integral in Eq. (1).

Next, we compute the integration for piece P_1 denoted as I_{P_1}. **Task 2: Integration over piece P_1 of the outer integral range $[a_1, b_{min}]$.** The integration for piece P_1 depends on the intervals that started before a_1 because the inner integral in Eq. (1) depends on x_1 for a random variable X_i (with $i \in 2, 3, 4, 5$) started before a_1, as max(x_1,a_i) is equal to x_1. The Task 2 in Algorithm 2, therefore, corresponds to finding the intervals that started before id_{a_1} (by looking up the I_{sorted} array), which also determines the upper limits of inner integrals for piece P_1 (denoted by the array $upLimits$ in Algorithm 2) depending on observed order of intervals. In particular, the $lowLimit$ and $upLimits[1]$ in Task 2 of Algorithm 2 store the limits of the outer integral for piece P_1. The remaining entries in the $upLimits$ array are used to store the upper limits of inner integrals based on the order they are observed.

The computation of the integral in Eq. (1) for piece P_1 (as well as any arbitrary piece P_i) simplifies to one of the five types of integration formulae, which we call integration templates. The integration templates are presented in Algorithm 3. Depending on the number of upper limits set (denoted by a variable h_i in Algorithm 3), the integral template varies. If there is no overlap with a piece, then the integral in Eq. (1) simplifies to the integration of the probability distribution of random variable X_1 over a piece, which corresponds to setting h_1 and the usage of Template 1 in Algorithm 3. If only one random variable is over-



(a) Task 1: Determine pieces P_i of the outer integral in Eq. (1)

(b) Task 2: Determine integral over a piece P_1

(c) Task 3: Determine integral over successive pieces P_i

Fig. 2: Tasks summarizing our algorithm for computing local minimum probability for uncertain data.

Function *computePieceIntegrals()*
 Input: $id_{a1}, id_{bMin}, I_{sorted}, b_{min}$
 Output: Sum of integrals over pieces P_i
 `/* Initialize the upper limits of outer and inner integrals for a piece */`
 $upLimits = [None, None, None, None, None]$
 $totalIntegral = 0$
 `/* Task2: Compute integral for piece` P_1 `*/`
 `/* Determine the upper limit of piece` P_1 `*/`
 $lowLimit = a_1$
 if $id_{a1} < id_{bMin}$ **then**
  $upLimits[1] = I_{sorted}[id_{a_1}+1][1]$
 if $id_{a1} == id_{bMin}$ **then**
  $upLimits[1] = b_{min}$
 `/* Find intervals started before` a_1 `and set respective upper limits for inner integrals */`
 for $k = 1$ *to* $(id_{a_1}-1)$ **do**
  $upLimits[k+1] = I_{sorted}[k][2]$
 end
 $I_{P_1} = integralTemplate(lowLimit, upLimits[1], upLimits[2], upLimits[3], upLimits[4], upLimits[5], I_{sorted})$
 $totalIntegral += I_{P_1}$
 `/* Task3: Compute integral for successive pieces */`
 for $k = (id_{a_1}+1)$ *to* id_{bMin} **do**
  `/* Determine piece limits */`
  $lowLimit = I_{sorted}[k][1]$
  if $k < id_{bMin}$ **then**
   $upLimits[1] = I_{sorted}[k+1][1]$
  else
   $upLimits[1] = b_{min}$
  `/* Determine the upper limit of new interval */`
  $upLimits[k] = I_{sorted}[k][2]$
  $I_{P_{next}} = integralTemplate(lowLimit, upLimits[1], upLimits[2], upLimits[3], upLimits[4], upLimits[5], I_{sorted})$
  $totalIntegral += I_{P_{next}}$
 end
 return $totalIntegral$
end

Algorithm 2: Computation of the integral for piece P_1 and successive pieces P_{next}

lapping with a piece, then the integral in Eq. (1) simplifies to integral over the joint distribution of X_1 and a random variable corresponding to overlapping interval. One overlapping interval corresponds to setting only h_1 and h_2 and the usage of Template 2 in Algorithm 3. Note that the parameters θ_i in Algorithm 3 capture the probability distribution functions to consider in templates by looking up the I_{sorted} array. The probability distributions to consider essentially depend on the ordering of intervals $[a_i, b_i]$ and are captured on the fly by parameters θ_i.

Having determined the integration for piece P_1, we compute integration for successive pieces (denoted as $I_{P_{next}}$ in Algorithm 2). **Task 3: Integration over piece P_i with $i > 1$.** Essentially, each new start point a_i with $i \in \{2,3,4,5\}$ observed between the outer integral limits a_1 and b_{min} of Eq. (1) creates a new piece. Generally speaking, each new start point a_i results in different simplification of Eq. (1) because $max(x_1, a_i)$ in Eq. (1) becomes equal to x_1 at each new start point. Thus, encountering a new start point adds one inner integral in a simplified form compared to the piece before encountering a new start point. Finally, integration of all pieces (i.e. I_{P_1} and $I_{P_{next}}$ in Algorithm 2) is summed to compute the local minimum probability at a point p, i.e., $\Pr(p = l_{min})$.

Illustration of local minimum probability computation: Fig. 2 illustrates our method for computing the local minimum probability for a domain point p. As shown for the example in Fig. 2, $a_5 < a_2 < a_1 < a_4 < b_2 < a_3 < b_1 < b_5 < b_3 < b_4$. Initially, we determine the range of a random variable X_1 that can result in point p being a local minimum. As shown in Fig. 2a, each value in the range $[x_1 = a_1, x_1 = (b_{min} = b_2)]$ has a nonzero probability of being simultaneously smaller than the neighboring random variables (i.e., $X_2, \dots, X_5$). In contrast, the range $[x_1 = (b_{min} = b_2), x_1 = b_1]$ is always greater than the random variable X_2, and therefore, cannot result in point p as a local minimum.

In Task 1, we determine the pieces P_i of the range $[x_1 = a_1, x_1 = (b_{min} = b_2)]$. As depicted in Fig. 2, the array I_{sorted} has intervals ordered by a_i, where $a_5 < a_2 < a_1 < a_4 < a_3$. For this I_{sorted}, $id_{a_1} = 3$ and $id_{b_{min}} = 4$. Since a_4 is a start point in interval indexed by $id_{b_{min}}$, it divides the outer integral range $[a_1, b_{min}]$ in Eq. (1) into two pieces (depicted in blue and orange in Fig. 2).

In Task 2, we determine the simplification of the formula in Eq. (1) for piece P_1. The simplification for piece $P_1 = [a_1, a_4]$ in Fig. 2 (denoted by blue) depends on the number of intervals that started before a_1. As observed in Fig. 2, the intervals $[a_5, b_5]$ and $[a_2, b_2]$ start before a_1. Since in piece P_1, $x_1 < a_3$ and $x_1 < a_4$, the formula in Eq. (1) integrates random variables X_3 and X_4 over their entire support and simplifies to the integration over the joint distribution of random variables X_1, X_5, and X_2 shown in Fig. 2b.

In Task 3, we determine the simplification of the formula in Eq. (1) for successive pieces formed by each new start point $a_i \in [a_1, b_{min}]$. In Fig. 2, the start point $a_4 \in [a_1, b_{min}]$ results in a new piece P_2 shown in orange. All inner integrals for piece P_2 stay the same as piece P_1 except for one newly added inner integral with limits $[x_1, b_4]$, as shown in Fig. 2c, because $x_1 = max(x_1, a_4)$ for piece P_2, unlike the piece P_1 in which $a_4 = max(x_1, a_4)$. Thus, we make such updates to inner integrals for each new piece corresponding to a new start point $a_i \in [a_1, b_{min}]$.

Time complexity: Algorithm 1 initially sorts intervals based on start points a_i with $i \in a_1 \dots a_5$ and b_{min} to determine pieces for integration (Task 1), which is a constant time operation. Task 2 and Task 3 in Algorithm 2 comprise a single loop, which runs a maximum of five times corresponding to five entries of a sorted interval array I_{sorted}. Each loop computes the integral template (Algorithm 3) on the fly in constant time. The algorithm is, therefore, linear time complexity with the number of input intervals and extremely efficient.

Function *integralTemplates()*

Input: Integral limits $l_1, h_1 = None, h_2 = None, h_3 = None, h_4 = None, h_5 = None$

Input: I_{sorted} denoting parametric distribution ranges sorted by start points a_i.

Output: Integral over piece $[l_1, h_1]$

/* Determine pdfs parameters θ based on intervals sorted based on start points a_i except for interval $[a_1, b_1]$ */

$k = 1$

for $i = 1$ *to* 5 **do**

  if $(i == id_{a_1})$ **then**

    continue

  else

    $\theta_k.a = I_{sorted}[i][1], \theta_k.b = I_{sorted}[i][2]$

    $k = k + 1$

end

/* Template 1 */

if $h_1 \neq None$ *and others* $= None$ **then**

  $I = \int_{l_1}^{h_1} pdf_{x,a_1,b_1}$

/* Template 2 */

else if $h_1 \neq None$ *and* $h_2 \neq None$ *and others* $= None$ **then**

  $I = \int_{l_1}^{h_1} \int_{x_1}^{h_2} pdf_{x,a_1,b_1} pdf_{x,\theta_1}$

/* Template 3 */

else if $h_1 \neq None$ *and* $h_2 \neq None$ *and* $h_3 \neq None$ *and others* $= None$ **then**

  $I = \int_{l_1}^{h_1} \int_{x_1}^{h_2} \int_{x_1}^{h_3} pdf_{x,a_1,b_1} pdf_{x,\theta_1} pdf_{x,\theta_2}$

/* Template 4 */

else if $h_1 \neq None$ *and* $h_2 \neq None$ *and* $h_3 \neq None$ *and* $h_4 \neq None$ *and others* $= None$ **then**

  $I = \int_{l_1}^{h_1} \int_{x_1}^{h_2} \int_{x_1}^{h_3} \int_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,\theta_1} pdf_{x,\theta_2} pdf_{x,\theta_3}$

/* Template 5 */

else if $h_1, \ldots, h_5 \neq None$ **then**

  $I = \int_{l_1}^{h_1} \int_{x_1}^{h_2} \int_{x_1}^{h_3} \int_{x_1}^{h_4} \int_{x_1}^{h_5} pdf_{x,a_1,b_1} pdf_{x,\theta_1} pdf_{x,\theta_2} pdf_{x,\theta_3} pdf_{x,\theta_4}$

return I

end

Algorithm 3: Integral templates for arbitrary piece P_i

2.0.2 Local Maximum Probability

Having derived a probabilistic framework for computation of local minimum probability (Sec. 2.0.1), the computation of the local maximum probability $Pr(p = l_{max})$ at each grid vertex p is fairly straightforward. Computation of the local maximum probability corresponds to computing $\Pr([(X_1 > X_2)$ and $(X_1 > X_3)$ and $(X_1 > X_4)$ and $(X_1 > X_5)])$, which is equivalent to computing $\Pr([(-X_1 < -X_2)$ and $(-X_1 < -X_3)$ and $(-X_1 < -X_4)$ and $(-X_1 < -X_5)])$. This negation format is equivalent to Eq. (1). Thus, we negate the intervals for random variables $X_1, \ldots, X_5$ to create new random variables $X_1' = -X_1, \ldots, X_5' = -X_5$. We then apply our proposed local minimum probability computation algorithm (Algorithm 1) to these new random variables X_i' for computing the local maximum probability in closed form.

2.0.3 Saddle Probability

Here, we provide a detailed description of the saddle point probability computation algorithm. As explained in the main paper, we derive only our closed-form computations and algorithm for the term $t_1 = \Pr[(X_1 < X_2)$ and $(X_1 > X_3)$ and $(X_1 < X_4)$ and $(X_1 > X_5)]$ in Eq. 4 of the main paper.

Function *computeSaddleProbability()*

Input: Intervals $I = [[a_1, b_1], \ldots, [a_5, b_5]]$. Without loss of generality, $a_2 < a_4$ and $b_3 < b_5$. So if $a_2 > a_4$ or $b_3 > b_5$ in the original data, then we swap the two intervals. Swapping these intervals do not change the probability computation in Eq.2 of the main paper.

Output: $Pr(p = l_s)$

$a_{max}^{alt} = max(a_1, a_3, a_5)$

$b_{min}^{alt} = min(b_1, b_2, b_4)$

if $b_{min}^{alt} \leq a_{max}^{alt}$ **then**

  return 0

else

  /* Task1: Break the outer integral in Eq. 2 of the main paper into pieces P_i */

  $sortedInterestPoints = sort([a_{max}^{alt}, a_2, a_4, b_3, b_5, b_{min}^{alt}])$

  $id_{a_{max}^{alt}} =$ Index of a_{max}^{alt} in $sortedInterestPoints$

  $id_{b_{min}^{alt}} =$ Index of b_{min}^{alt} in $sortedInterestPoints$

  /* Task2 and Task 3: Compute piece integrals */

  $saddleProbability = computePieceIntegrals(id_{a_{max}^{alt}}, id_{b_{min}^{alt}}, sortedInterestPoints, I)$

  return $saddleProbability$

end

Algorithm 4: Computation of saddle probability

$$
\begin{aligned}
t_1 &= \Pr[(X_1 < X_2) \text{ and } (X_1 > X_3) \text{ and } (X_1 < X_4) \text{ and } (X_1 > X_5)] \\
&= \int_{x_1=a_{max}^{alt}}^{x_1=b_{min}^{alt}} \int_{x_2=max(x_1,a_2)}^{x_2=b_2} \int_{x_3=a_3}^{x_3=min(x_1,b_3)} \int_{x_4=max(x_1,a_4)}^{x_4=b_4} \int_{x_5=a_5}^{x_5=min(x_1,b_5)} \ldots \\
&(\mathrm{Pdf}_{joint})\, \mathrm{d}x, \\
&\text{where } a_{max}^{alt} = max(a_1, a_3, a_5),\ b_{min}^{alt} = min(b_1, b_2, b_4)
\end{aligned} \tag{2}
$$

We now explain the integral limits in Eq. (2) and the piecewise integration algorithm to compute the formula in Eq. (2), similar to our explanations for the local minimum probability computation in Sec. 2.0.1.

Limits a_{max}^{alt} and b_{min}^{alt} of the outer integral in Eq. (2): First, we identify the portion of data range of a random variable X_1 (i.e., $[a_1, b_1]$) that can result in point p being a saddle. The portion $[a_{max}^{alt}, b_{min}^{alt}]$ of outer integral in Eq. (2) indicates the data values that can result in point p being saddle, where a_{max}^{alt} denotes the maximum value among a_1, a_3, and a_5, b_{min}^{alt} denotes the minimum value among b_1, b_2, and b_4, and $a_{max}^{alt} < b_{min}^{alt}$. The superscript *alt* represents alternate neighboring random variables of a random variable X_1 under consideration. The data portion $[a_1, a_{max}^{alt}]$ of random variable X_1 with $a_1 < a_{max}^{alt}$ will always be smaller than either X_3 or X_5 depending on if a_{max}^{alt} is equal to a_3 or a_5, respectively. Thus, the range $[a_1, a_{max}^{alt}]$ cannot result in point p as a saddle. Similarly, the data portion $[b_{min}^{alt}, b_1]$ of random variable X_1 with $b_{min}^{alt} < b_1$ will always be greater than either X_2 or X_4 depending on if b_{min}^{alt} is equal to b_2 or b_4, respectively. Thus, the range $[b_{min}^{alt}, b_1]$ cannot result in point p as a saddle. To the contrary, each point in the range $[x_1 = a_{max}^{alt}, x_1 = b_{min}^{alt}]$ has a nonzero probability of simultaneously being smaller than the random variables X_2 and X_4 and greater than the random variables X_3 and X_5, and therefore, represents a valid range for point p being a saddle.

Limits $max(x_1, a_i)$ and $min(x_1, b_i)$ of the inner integrals in Eq. (2): Inner integrals in Eq. (2) integrate the joint distribution Pdf_{joint} over its support where random variables X_2 and X_4 are simultaneously greater and X_3 and X_5 are simultaneously smaller than $x_1 \in [a_{max}^{alt}, b_{min}^{alt}]$ in the outer integral. The inner integral lower limit

```
Function computePieceIntegrals()
    Input: id_{a^{alt}_{max}}, id_{b^{alt}_{min}}, sortedInterestPoints, I
    Output: Sum of integrals over pieces P_i

    /* Task2: Compute integral for piece P_1 */
    /* Lower (l) and upper (h) limits of piece P_1 */
    l = sortedInterestPoints[id_{a^{alt}_{max}}]
    h = sortedInterestPoints[id_{a^{alt}_{max}} + 1]
    totalIntegral = 0
    /* Initialize the limits of inner integrals
       for piece P_1 assuming a_2, a_4 starting after
       a^{alt}_{max} and a_3, a_5 are starting before a^{alt}_{max} */
    tempd3 = a_3
    tempd5 = a_5
    tempu2 = None
    tempu4 = None
    /* Update the initialized limits depending on
       which of interest points a_2, a_4, b_3, b_5 lie
       before a^{alt}_{max}. */
    for k = 1 to (id_{a^{alt}_{max}} - 1) do
        if sortedInterestPoints[k] == a_2 then
            tempu2 = I[2][2] ;                    // i.e., b_2
        if sortedInterestPoints[k] == a_4 then
            tempu4 = I[4][2] ;                    // i.e., b_4
        if sortedInterestPoints[k] == b_3) then
            tempd3 = None
        if sortedInterestPoints[k] == b_5) then
            tempd5 = None
    end
    I_{P_1} = integralTemplate(l, h, l3 = tempd3, l5 =
      tempd5, h2 = tempu2, h4 = tempu4, I)
    totalIntegral += I_{P_1}

    /* Task3: Compute integral for successive
       pieces */
    for k = (id_{a^{alt}_{max}} + 1) to (id_{b^{alt}_{min}})) do
        /* Determine piece limits */
        l = sortedInterestPoints[k]
        h = sortedInterestPoints[k + 1]
        /* Update the lower or upper limit for
           each new interest point */
        if sortedInterestPoints[k] == a_2 then
            tempu2 = I[2][2] ;                    // i.e., b_2
        if sortedInterestPoints[k] == a_4 then
            tempu4 = I[4][2] ;                    // i.e., b_4
        if sortedInterestPoints[k] == b_3) then
            tempd3 = None
        if sortedInterestPoints[k] == b_5) then
            tempd5 = None
        I_{P_next} = integralTemplate(l, h, l3 = tempd3, l5 =
          tempd5, h2 = tempu2, h4 = tempu4, I)
          totalIntegral += I_{P_next}
    end
    return totalIntegral
end
```

Algorithm 5: Computation of integral for piece P_1 and successive pieces P_{next}

```
Function integralTemplates()
    Input: Integral limits
           l, h, l_3 = None, l_5 = None, h_2 = None, h_4 = None
    Output: Integral over piece [l, h]
    /* Template 1 */
    if l_3 ≠ None and others = None then
        I = ∫_l^h ∫_{l_3}^{x_1} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3}
    /* Template 2 */
    else if l_5 ≠ None and others = None then
        I = ∫_l^h ∫_{l_5}^{x_1} pdf_{x,a_1,b_1} pdf_{x,a_5,b_5}
    /* Template 3 */
    else if h_2 ≠ None and others = None then
        I = ∫_l^h ∫_{x_1}^{h_2} pdf_{x,a_1,b_1} pdf_{x,a_2,b_2}
    /* Template 4 */
    else if h_4 ≠ None and others = None then
        I = ∫_l^h ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_4,b_4}
    /* Template 5 */
    else if l_3 ≠ None and l_5 ≠ None and others = None then
        I = ∫_l^h ∫_{l_3}^{x_1} ∫_{l_5}^{x_1} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3} pdf_{x,a_5,b_5}
    /* Template 6 */
    else if l_3 ≠ None and h_2 ≠ None and others = None then
        I = ∫_l^h ∫_{l_3}^{x_1} ∫_{x_1}^{h_2} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3} pdf_{x,a_2,b_2}
    /* Template 7 */
    else if l_3 ≠ None and h_4 ≠ None and others = None then
        I = ∫_l^h ∫_{l_3}^{x_1} ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3} pdf_{x,a_4,b_4}
    /* Template 8 */
    else if l_5 ≠ None and h_2 ≠ None and others = None then
        I = ∫_l^h ∫_{l_5}^{x_1} ∫_{x_1}^{h_2} pdf_{x,a_1,b_1} pdf_{x,a_5,b_5} pdf_{x,a_2,b_2}
    /* Template 9 */
    else if l_5 ≠ None and h_4 ≠ None and others = None then
        I = ∫_l^h ∫_{l_5}^{x_1} ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_5,b_5} pdf_{x,a_4,b_4}
    /* Template 10 */
    else if h_2 ≠ None and h_4 ≠ None and others = None then
        I = ∫_l^h ∫_{x_1}^{h_2} ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_2,b_2} pdf_{x,a_4,b_4}
    /* Template 11 */
    else if l_3 ≠ None and l_5 ≠ None and h_2 ≠ None and
     h_4 = None then
        I =
          ∫_l^h ∫_{l_3}^{x_1} ∫_{l_5}^{x_1} ∫_{x_1}^{h_2} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3} pdf_{x,a_5,b_5} pdf_{x,a_2,b_2}
    /* Template 12 */
    else if l_3 ≠ None and l_5 ≠ None and h_4 ≠ None and
     h_2 = None then
        I =
          ∫_l^h ∫_{l_3}^{x_1} ∫_{l_5}^{x_1} ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3} pdf_{x,a_5,b_5} pdf_{x,a_4,b_4}
    /* Template 13 */
    else if l_3 ≠ None and h_2 ≠ None and h_4 ≠ None and
     l_5 = None then
        I =
          ∫_l^h ∫_{l_3}^{x_1} ∫_{x_1}^{h_2} ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3} pdf_{x,a_2,b_2} pdf_{x,a_4,b_4}
    /* Template 14 */
    else if l_5 ≠ None and h_2 ≠ None and h_4 ≠ None and
     l_3 = None then
        I =
          ∫_l^h ∫_{l_5}^{x_1} ∫_{x_1}^{h_2} ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_5,b_5} pdf_{x,a_2,b_2} pdf_{x,a_4,b_4}
    /* Template 15 */
    else if l_3 ≠ None and l_5 ≠ None and h_2 ≠ None and
     h_4 ≠ None then
        I = ∫_l^h ∫_{l_3}^{x_1} ∫_{l_5}^{x_1} ∫_{x_1}^{h_2} ∫_{x_1}^{h_4} pdf_{x,a_1,b_1} pdf_{x,a_3,b_3}
            pdf_{x,a_5,b_5} pdf_{x,a_2,b_2} pdf_{x,a_4,b_4}
    return I
end
```

Algorithm 6: Integral templates for arbitrary piece P_i

corresponds to $max(x_1, a_i)$ for $i \in \{2,4\}$ and upper limit corresponds to $min(x_1, b_i)$ for $i \in \{3,5\}$. These integral limits are derived using a reasoning similar as in the case of local minimum probability described Sec. 2.0.1. The maximum or minimum is taken because the support of random variable X_i is restricted to $[a_i, b_i]$. In particular, for $x_1 < a_i$ in inner integral with $i \in \{2,4\}$, the support $[a_i, b_i]$ will always be greater than x_1 and the inner integral does not depend on the value of x_1. Similarly, for $x_1 > b_i$ in any inner integral with $i \in \{3,5\}$, the support $[a_i, b_i]$ will always be smaller than x_1, and the inner integral does not depend on the value of x_1. In contrast, when x_1 assumes values in the support of a distribution, the inner integral then depends on x_1. It can also be verified that the lower limit of any inner integral does not exceed the upper limit of their respective integral. For example, $a_3 \le min(x_1, b_3)$ in all cases because our initial assumption is $a_3 < b_3$ for the random variable X_3 (see the Background and Problem Setting section of the main paper). Also, x_1 in the outer integral of Eq. (2) attains values in the range $[a^{alt}_{max} = max(a_1, a_3, a_5), b^{alt}_{min} = min(b_1, b_2, b_4)]$ (see the previous paragraph). Since the minimum value x_1 is equal to $max(a_1, a_3, a_5)$, $x_1 \ge a_3$ is guaranteed. Thus, $a_3 \le min(x_1, b_3)$ is true in all cases. A similar reasoning is applicable to the other inner integrals too.

Piecewise simplification of Eq. (2): The saddle probability again needs to be computed in a piecewise manner for reasons similar to the case of local minimum probability computation described in Sec. 2.0.1. Essentially, at each start point a_i with $i \in \{2, 4\}$ in Eq. (2), the value of $max(x_1, a_i)$ changes, which simplifies the integral computation differently. Similarly, for each end point b_i with $i \in \{3, 5\}$ in Eq. (2), the value of $min(x_1, b_i)$ changes, which simplifies the integral computation differently. The simplified formula of a piecewise integral, therefore, depends on the ordering of the points a_2, a_4, b_3, and b_5 with respect to the limits of the outer integral in Eq. (2), i.e., a^{alt}_{max} and b^{alt}_{min}.

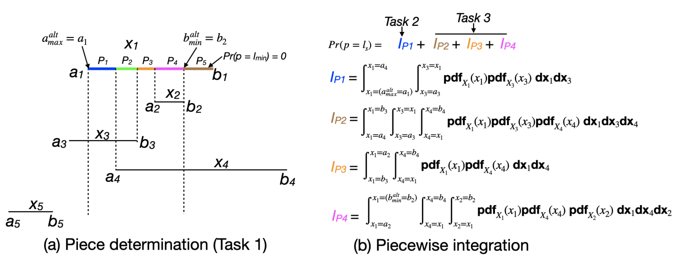


Fig. 3: Illustration of saddle probability computation. (a) Pieces are determined as a part of Task 1. Here, pieces are $P_1 = [a^{alt}_{max} = a_1, a_4]$, $P_2 = [a_4, b_3]$, $P_3 = [b_3, a_2]$, and $P_4 = [a_2, b^{alt}_{min} = b_2]$. (b) The saddle probability is computed by computing integral for the first piece P_1 (Task 2) followed by integrals for successive pieces P_{next} (Task 3) and summing all piecewise integrals.

Algorithm for computing saddle probability: The computation of the formula in Eq. (2) depends on the ordering of points a_2, a_4, b_3, and b_5 with respect to the limits of the outer integral in Eq. (2), i.e., a^{alt}_{max} and b^{alt}_{min}. Because of these six quantities, there are again $6! = 720$ permutations similar to the local minimum probability case. We, therefore, devise an efficient algorithm that can compute the integral in Eq. (2) without needing to go through all permutations. Our algorithm matches closely with the one for the local minimum probability computation. Our algorithm computes the simplified integrals per piece P_i on the fly depending on the observed permutation of points a_2, a_4, b_3, b_5, $a^{alt}_{max} = max(a_1, a_3, a_5)$, and $b^{alt}_{min} = min(b_1, b_2, b_4)$. If $b^{alt}_{min} \le a^{alt}_{max}$, the saddle probability is 0. If $a^{alt}_{max} < b^{alt}_{min}$, we divide our algorithm into three tasks. In Task 1, we determine the pieces P_i of the outer integral in Eq. (2) for performing piecewise integration (see Task 1 in Algorithm 4). For that, we sort the intervals based on the ordering of a_2, a_4, b_3, b_5, $a^{alt}_{max} = max(a_1, a_3, a_5)$, and $b^{alt}_{min} = min(b_1, b_2, b_4)$. This operation is similar to the I_{sorted} array computation in the case of local minimum probability computation in Algorithm 1. We then find the indices of a^{alt}_{max} and b^{alt}_{min} in the sorted intervals and points in between, which determine the pieces P_i.

In Task 2, we compute the integration for the first piece P_1 depending on which points in a_2, a_4, b_3, and b_4 are starting before or after a^{alt}_{max}. The algorithm for integration for the first piece P_1 is presented in Algorithm 5 as Task 2 (similar to the Task 2 in the case of local minimum probability computation in Algorithm 2). Finally, we compute the integration for successive pieces (denoted as $I_{P_{next}}$ in Algorithm 5) depending on the order of points a_2, a_4, b_3, and b_4 observed between a^{alt}_{max} and b^{alt}_{min}, which is similar to the Task 3 in Algorithm 2 for the local minimum probability computation. Again, we derive the integral templates (i.e., simplifications) similar to those in the case of local minimum probability computation, which are detailed in Algorithm 6.

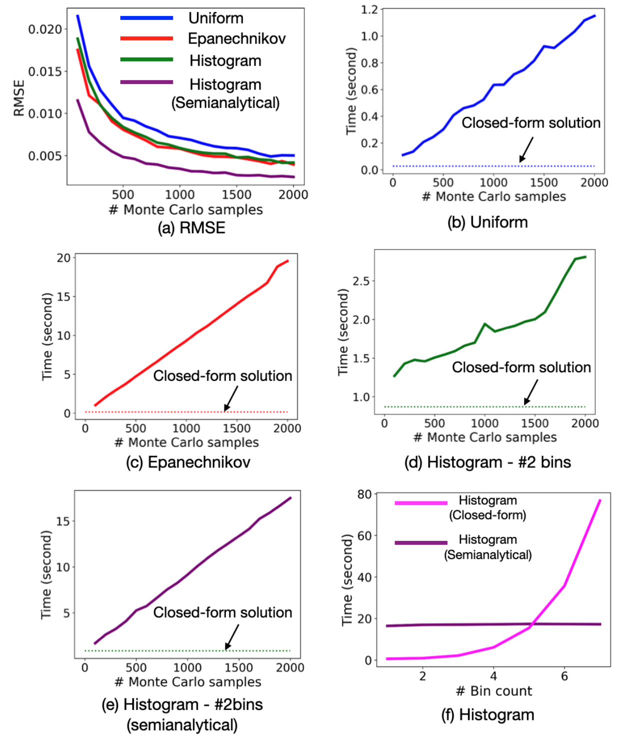


Fig. 4: Quantitative proof of correctness of our proposed closed-form computations and performance results. The RMSE between our closed-form/semianalytical solutions and the MC solution (image a) drops with the increase in the number of MC samples, which confirms the correctness of our algorithms. As depicted in images b-e, the closed-form computation (dotted line) provides significantly high performance compared to the MC sampling (solid curves). The closed-form histogram computation time exponentially grows with an increase in the number of bins, but the semianalytical solution time stays about constant (image f) for 1000 MC samples.

These integration templates compute the integral for each piece on the fly based on the observed piece limits.

Illustration of saddle probability computation: We illustrate the three tasks comprising the saddle probability computation in Fig. 3. For the example shown in Fig. 3, $a_5 < b_5 < a_3 < a_1 < a_4 < b_3 < a_2 < b_2 < b_1 < b_4$. The Task 1 includes determining the portion of random variable X_1 that can result in point p being a saddle and determining pieces for integration. As observed in Fig. 3a, each point outside the range $[x_1 = a^{alt}_{max}, x_1 = b^{alt}_{min}]$ has a zero probability of being a saddle. For example, in Fig. 3a, the range $[x_1 = b^{alt}_{min}, b_1]$ (i.e., $[b_2, b_1]$ shown in brown) has a zero probability of being smaller than the random variable X_2, which violets the condition of a saddle in Eq. (2). Thus, only the data range $[x_1 = a^{alt}_{max}, x_1 = b^{alt}_{min}]$ can result in point p being a saddle.

We then determine pieces of integration. For the example illustrated in Fig. 3a, the ordering of points of interest (points involved in the integration limits of Eq. (2)) that determines the piece limits is $b_5 < a^{alt}_{max} = a_1 < a_4 < b_3 < a_2 < b^{alt}_{min} = b_2$. Thus, there are four pieces $P_1 = [a^{alt}_{max} = a_1, a_4], P_2 = [a_4, b_3], P_3 = [b_3, a_2], P_4 = [a_2, b^{alt}_{min} = b_2]$, depicted by four colors in Fig. 3a. The piecewise integration process is again similar to the local minimum probability computation described in Sec. 2.0.1.

In Task 2, we compute the integration formula for piece $P_1 = [a_1, a_4]$ denoted by a blue range in Fig. 3a. We determine how many intervals are overlapping with the piece P_1 to compute integration over piece P_1, i.e., I_{P_1}. The process of finding which intervals are overlapping with the piece P_1 is similar to the one in the case of local minimum probability computation (the only difference is that the points of interest in the case of local minimum probability computation are $a_1, \ldots, a_5$ and b_{min}). Since $b_5 < (a^{alt}_{max} = a_1)$ and $a_4, a_2 > a_1$ in Fig. 3a, the intervals of random variables X_5, X_4, and X_2 do not overlap with the piece P_1.

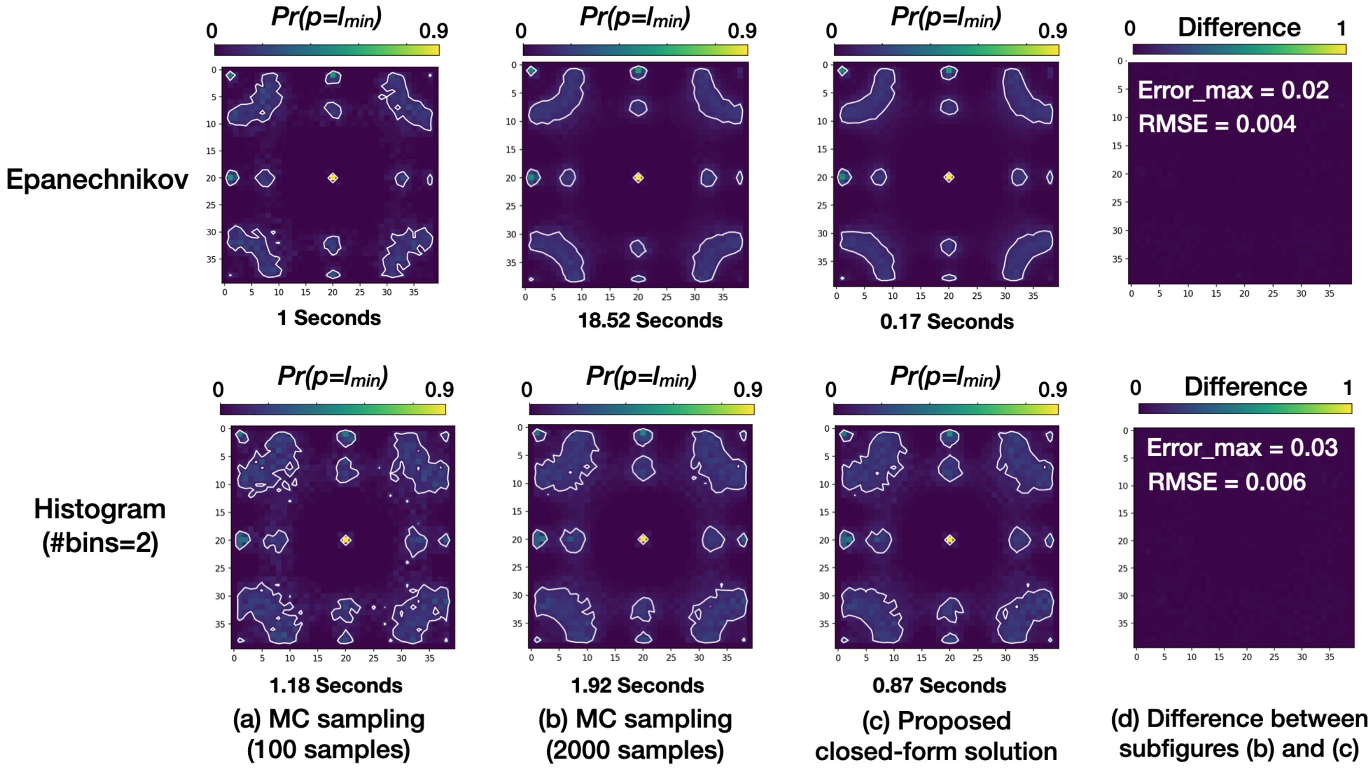


Fig. 5: Qualitative proof of correctness of our proposed closed-form computations (column c) for the Epanechnikov (top row) and Histogram models (bottom row) using the MC sampling approach (columns a and b) as the baseline. The solution obtained with 2000 MC samples converges to our closed-form computations (see the difference image in column d), thereby confirming the correctness of our methods and implementation. Our method increases the speed for both Epanechnikov and histogram with 2 bins. Note that the closed-form histogram computation, although accurate, exponentially grows with an increase in the number of bins.

Only the interval for random variable X_3, i.e., $[a_3, b_3]$ overlaps with the piece P_1. Thus, the core integration formula in Eq. (2) simplifies to a double integral over the marginal joint distribution of random variables X_1 and X_3, as shown with the formula for I_{P_1} in Fig. 3b.

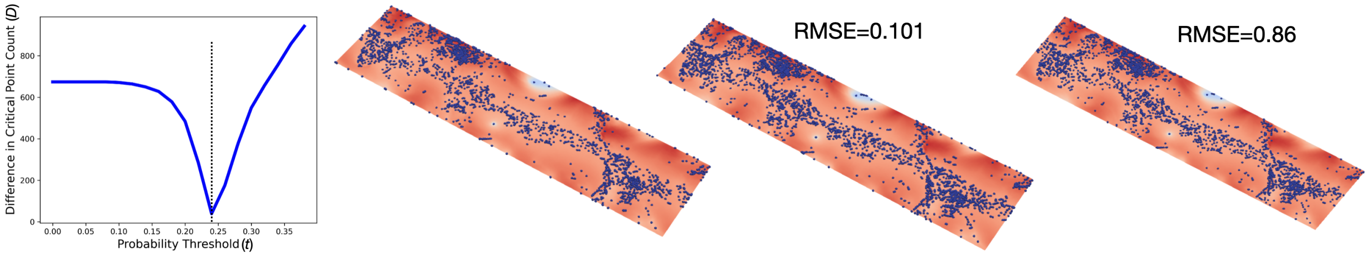
In Task 3, the integration is computed for successive pieces. In particular, for the pieces P_2 (green), P_3 (orange), and P_4 (magenta) in Fig. 3a, the integral updates based on if a new piece results from a new start point (i.e, a_i with $i \in \{2,4\}$) or end of an interval (i.e, b_i with $i \in \{3,5\}$). For example, for the piece P_2, the start point a_4 causes the interval for random variable X_4, i.e., $[a_4, b_4]$ to overlap with the piece P_2 in addition to the interval for random X_3, i.e., $[a_3, b_3]$. Thus, the core integration formula in Eq. (2) simplifies to a triple integral over the marginal joint distribution of random variables X_1, X_3, and X_4, as shown with the formula for I_{P_2} in Fig. 3b. On the contrary, the end point b_3 terminates the overlap of the interval $[a_3, b_3]$ with the piece P_3. Thus, the core integration formula in Eq. (2) simplifies to a double integral over the marginal joint distribution of random variables X_1 and X_4, as shown with the formula for I_{P_3} in Fig. 3b. The integration formula for the piece P_4 updates in a similar manner because of a new start point a_2.

3 Validation and Performance of the Proposed Algorithms

We show the quantitative results for the uniform, Epanechnikov, and histogram noise models and qualitative results for Epanechnikov and histogram noise models (similar to Fig. 6 in the main paper for the uniform noise) for the local minimum probability computations on the Ackley dataset [1] used in the main paper. Figure 4 shows the quantitative comparison of MC sampling solution and our closed-form computations using the proposed algorithms. For all uniform, Epanechnikov, and histogram (closed-form and semianalytical) noise models, the root mean squared error (RMSE) between the MC and closed-form solution drops as the number of MC samples is increased, as observed in Fig. 4a. This convergence confirms the correctness of our closed-form computations and algorithms. In Fig. 4a, the semianalytical (purple curve) solution has a lower RMSE than the histogram with MC sampling (green curve) for all neighbors per grid point, thereby showing the higher accuracy of our semianalytical approach. Further, as seen from Fig. 4b-e, the closed-form solutions provide the highest performance (dotted line) compared to the MC method (solid curves). As observed in Fig. 4f, the closed-form histogram computation time exponentially grows with an increase in the number of bins, as described in the main paper. The semianalytical histogram solution (the purple curve in Fig. 4f), however, takes an about constant amount of time for a fixed number of MC samples because it does not depend on the bin count (see the main paper for details). Figure 5 visualizes the qualitative results similar to the Fig. 6 of the main paper, but for the Epanechnikov and histogram noise models. It shows that the MC solution converges to our closed-form solution as we increase the number of samples from 100 (Fig. 5a) to 2000 (Fig. 5b). This qualitative convergence can be seen through the difference image in Fig. 5d that exhibits negligible maximum error ($Error_{max}$) and RMSE between Fig. 5c and Fig. 5b.

4 Quantitative Evaluation for the Climate Data

In Fig. 6, we present a quantitative evaluation for the climate data [4] used in the main paper. First, we computed the difference D between the number of critical points of the decompressed and original field as a function of probability threshold. In particular, at probability threshold t, we consider all critical points of decompressed data that have probability greater than t. Thus, for $t = 0$, all critical points of a decompressed field are considered. However, as we increase t, critical points with probability smaller than t get filtered out. For a probability threshold $t = 0$, the difference D is high, as all critical points of decompressed field are considered (which contain several

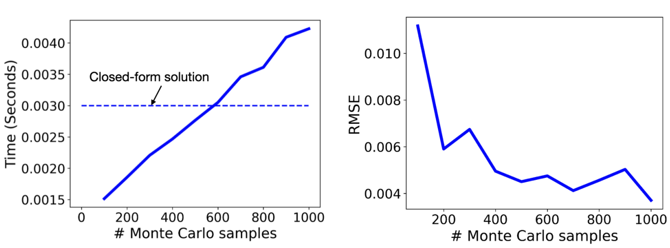


(a) Critical point count difference (b) Original field (c) Decompressed field [Probability threshold (t) = 0] (d) Decompressed field [Probability threshold (t) = 0.24]

Fig. 6: Quantitative evaluation of the climate data. (a) Difference (D) between the critical point count of original and decompressed fields plotted as a function of a probability threshold (t). In particular, for a threshold t, all critical points with probability smaller than t are removed in the decompressed field. At threshold $t = 0.24$, the number of critical points of the original and decompressed fields is nearly same. (b) Critical points of the original field. (c) Critical points of the decompressed data without applying filtering ($t = 0$). (d) Critical points of the decompressed data filtered by probability threshold of $t = 0.24$. The filtered critical points in (d) exhibit lower error (RMSE $= 0.86$) with respect to the nonfiltered critical points in (c) (RMSE $= 0.101$) with the original field in (a) as the reference, which is illustrative of the utility of probabilistic methods in identifying robust critical points.

new critical points from compression errors). As we start increasing the threshold, the difference D drops and reaches the minimum at a probability threshold $t = 0.24$, as observed in Fig. 6a. For probability threshold $t > 0.24$, the difference D again increases because the number of thresholded critical points in the decompressed field become smaller than the number of critical points in the original field. Thus, at $D = 0.24$, both the original and decompressed fields have about the same number of critical points.

We compute the root mean squared error (RMSE) to quantitatively evaluate probabilistic results. We consider the decompressed data without critical point filtering ($t = 0$) in Fig. 6c and with critical point filtering ($t = 0.24$) in Fig. 6d for evaluation. Initially, we assign 1 to each position in the original field where the critical point appears and 0 everywhere else to create a binary field. We generate similar binary fields denoting critical point positions for thresholds $t = 0$ (decompressed data with all critical points) and $t = 0.24$ (decompressed data with critical points that have probability greater than $t = 0.24$). We then compute the RMSE between these binary image representations. The RMSE for nonfiltered critical points in Fig. 6c with respect to original data critical points is 0.101. This RMSE drops to 0.86 as we filter critical points based on probability in Fig. 6d, which indicates that removing low-probability critical points from decompressed data provides answer closer to the true critical points in the original field, thereby presenting the utility of probability computations for critical points.



(a) Timing (b) Error

Fig. 7: Performance and accuracy results of our VTK-m implementation on the AMD GPU. (a) The performance reduces with an increase in the number of MC samples. (b) The RMSE between the MC solution and solution of our algorithms reduces with an increase in the number of MC samples, thereby confirming the correctness of our algorithms and parallel implementation.

The performance and accuracy results for the climate dataset using our VTK-m parallel implementation on the AMD GPU are shown in Fig. 7. The AMD GPU resource is courtesy of Oak Ridge National Laboratory's Frontier supercomputer [3]. The performance of the MC solutions decreases with an increase in the number of samples, as observed from Fig. 7a. The decrease in RMSE between the MC solution and our closed-form solution with an increase in the number of MC samples, as observed in Fig. 7b, confirms the correctness of our algorithms.

5 Integration of VTK-m Code with ParaView

One of the main benefits offered by VTK-m [5] includes its easy integration with ParaView using a plugin approach, thereby making VTK-m filters accessible to a broader community. Figure 1 visualizes the integration of our VTK-m critical point uncertainty code with ParaView. The ParaView plugin is generated by wrapping a VTK-m class inside VTK class, which can then be integrated in ParaView. Using GPU as a backend, reliable uncertainty visualizations can be generated using the proposed nonparametric models (i.e., histograms) in near-real-time within ParaView. We provide a small video demonstration as another supplement to show the usage of our VTK-m critical point uncertainty filter inside ParaView.

Acknowledgments

This work was supported in part by the U.S. Department of Energy (DOE) RAPIDS-2 SciDAC project under contract number DE-AC0500OR22725, NSF III-2316496, the Intel OneAPI CoE, and the DOE Ab-initio Visualization for Innovative Science (AIVIS) grant 2428225. This research used resources of the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725, and National Energy Research Scientific Computing Center (NERSC), which is a DOE National User Facility at the Berkeley Lab. We would also like to thank the reviewers of this article for their valuable feedback.

References

[1] D. H. Ackley. *A connectionist machine for genetic hillclimbing*. Kluwer Academic Publishers Norwell, MA, USA, 1987. doi: 10.1007/978-1-4613-1997-9 7

[2] J. Ahrens, B. Geveci, and C. Law. *ParaView: An End-User Tool for Large Data Visualization*, chap. 36, pp. 717–731. Elsevier, 2005. doi: 10.1016/B978-012387582-2/50038-1 1

[3] S. Atchley et al. Frontier: Exploring exascale. In *SC: High Performance Computing, Networking, Storage and Analysis*, November 2023. doi: 10.1145/3581784.3607089 8

[4] J.-C. Golaz, L. P. Van Roekel, X. Zheng, A. F. Roberts, J. D. Wolfe, W. Lin, A. M. Bradley, Q. Tang, M. E. Maltrud, R. M. Forsyth, et al. The DOE E3SM model version 2: Overview of the physical model and initial model evaluation. *Journal of Advances in Modeling Earth Systems*, 14(12):e2022MS003156, 2022. doi: 10.1029/2022MS003156 1, 7

[5] K. Moreland, C. Sewell, W. Usher, L.-T. Lo, J. Meredith, D. Pugmire, J. Kress, H. Schroots, K.-L. Ma, H. Childs, M. Larsen, C.-M. Chen, R. Maynard, and B. Geveci. VTK-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE Computer Graphics and Applications*, 36(3):48–58, 2016. doi: 10.1109/MCG.2016.48 1, 8

[6] S. Sanikommu, H. Toye, P. Zhan, S. Langodan, G. Krokos, O. Knio, and I. Hoteit. Impact of atmospheric and model physics perturbations on a high-resolution ensemble data assimilation system of the Red Sea. *Journal of Geophysical Research: Oceans*, 125(8):e2019JC015611, July 2020. doi: 10.1029/2019JC015611 1