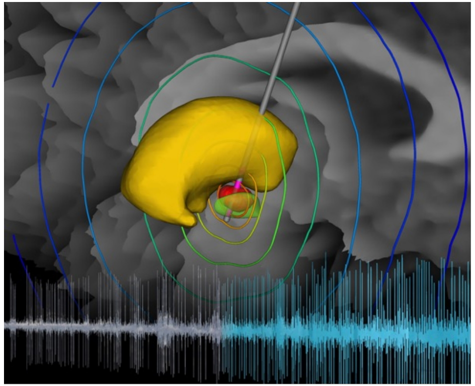
Data-Driven Computation of Probabilistic Marching Cubes for Efficient Visualization of Level-Set Uncertainty

Tushar M. Athawale[1], Zhe Wang[1], Chris R. Johnson[2], and David Pugmire[1]

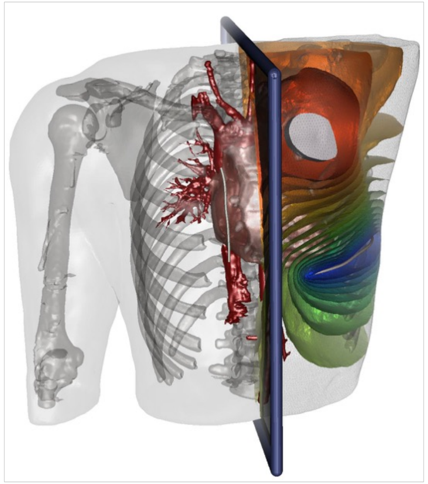
1 – Oak Ridge National Laboratory, Oak Ridge, USA
2 – Scientific Computing & Imaging (SCI) Institute, University of Utah, Salt Lake City, USA

EuroVis 2024 Short Papers

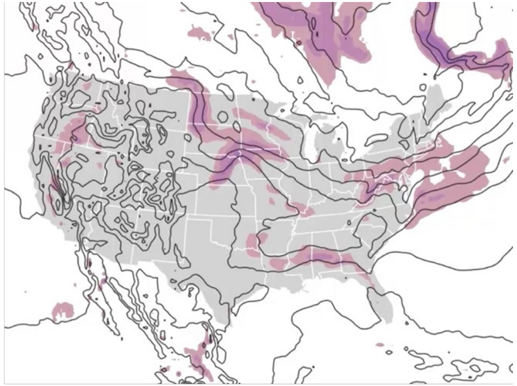
Level-Set Visualization



Deep Brain Stimulation (DBS)



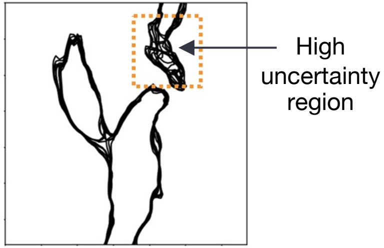
Bioelectric-field Simulation



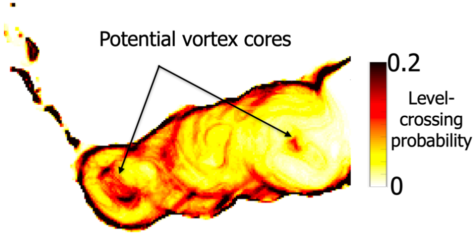
Temperature Field

Uncertainty Visualization of Level-Sets

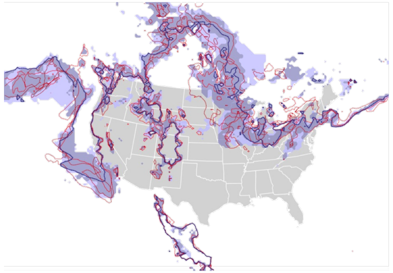
Uncertainty in data arises from various factors, including quantization errors, approximations used in simulations, model uncertainty etc.



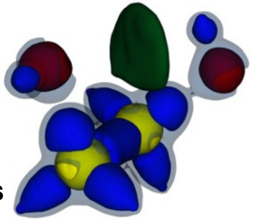
Spaghetti plot (climatology)
[Potter et al., 2009]



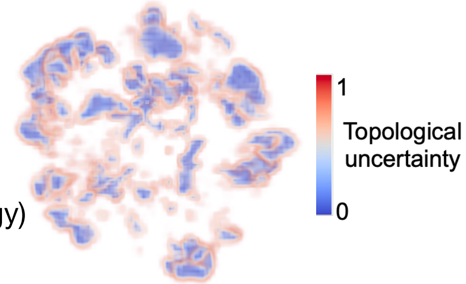
Probabilistic marching cubes (oceanology)
[Pöthkow et al., 2011,
Athawale et al., 2021]



Contour/surface boxplots (climatology)
[Whitaker et al., 2013, Genton et al., 2014]



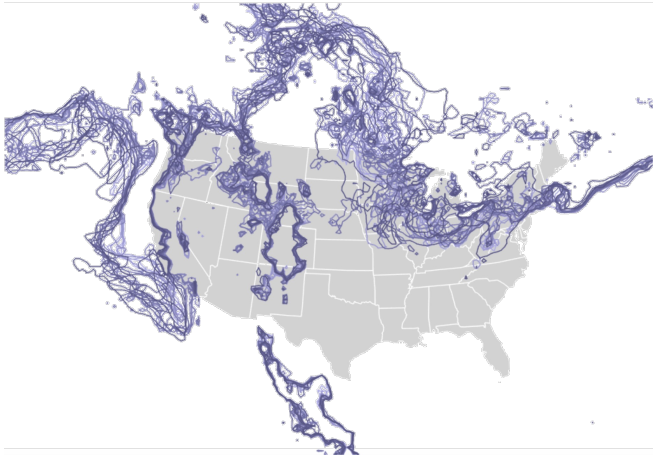
Feature confidence level-sets
(Molecular dynamics)
[Sane et al., 2020]



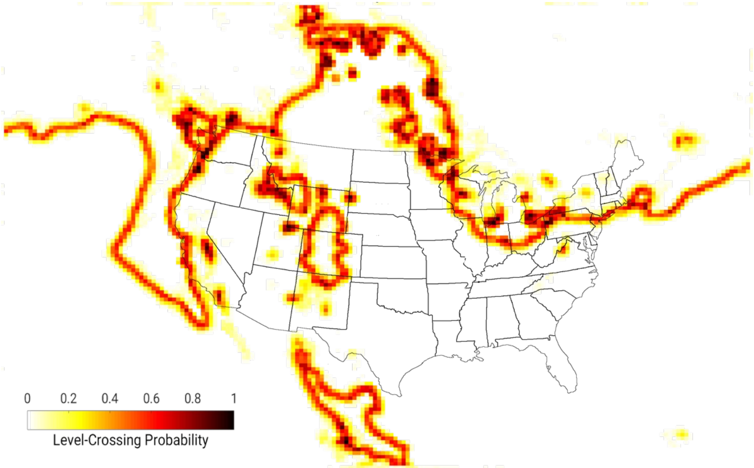
Fiber uncertainty (cosmology)
[Athawale et al., 2022]

Probabilistic Marching Cubes

Spaghetti plot
[Potter et al., 2009]



Probabilistic marching cubes
[Pöthkow et al., 2011, Athawale et al., 2021]



+ Computationally efficient
+ Direct display of high/low uncertainty among isocontours
- Clutter and occlusion (severe in 3D)

+ Mitigates occlusion/clutter
+ Highlights high probability (red) isocontour regions
- **Computationally expensive** (because of the required Monte Carlo sampling)

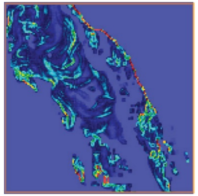
Probabilistic Marching Cubes

Problem: Computational cost of probabilistic marching cubes limits its use and prevents its integration with visualization tools/software (e.g., ParaView, VisIt)

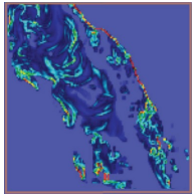
Existing Solutions:

Acceleration by Deep Learning (Han et al., 2022)

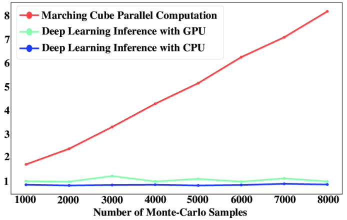
Up to 170× speedup, but requires training and limited to time-varying ensembles



Monte Carlo

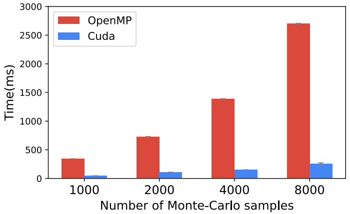


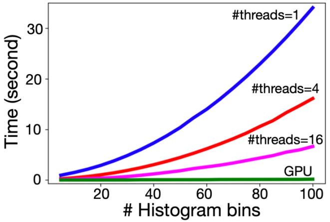
Machine-predicted



FunMC^2 filter: Acceleration using Many-Core GPUs (Wang et al., 2023)

Up to 396× speedup, but requires access to the GPU resources





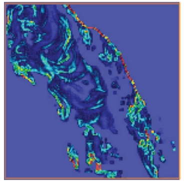
Probabilistic Marching Cubes

Problem: Computational cost of probabilistic marching cubes limits its use and prevents its integration with visualization tools/software (e.g., ParaView, VisIt)

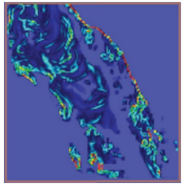
Existing Solutions:

Acceleration by Deep Learning (Han et al., 2022)

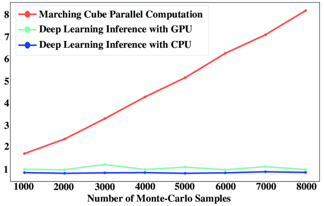
Up to 170× speedup, but requires training and limited to time-varying ensembles



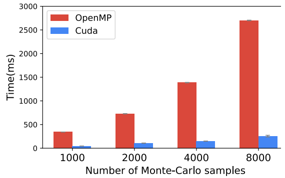
Monte Carlo

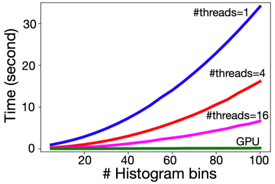


Machine-predicted



FunMC[2] filter: Acceleration using Many-Core GPUs (Wang et al., 2023) Up to 396× speedup, but requires access to the GPU resources



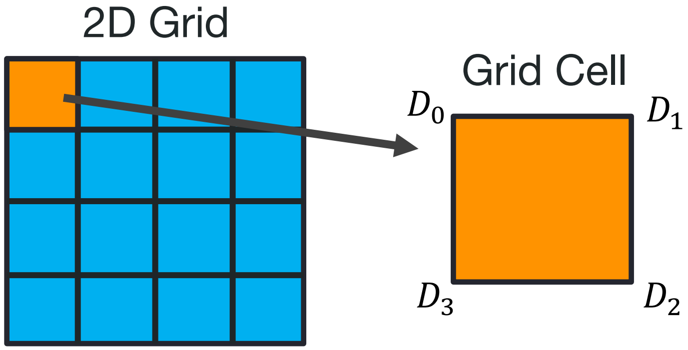


Our Contribution:

Data-driven methods (e.g., dimensionality reduction and correlation analysis) to reduce the amount of Monte Carlo sampling, and hence, achieve speedup

Probabilistic Marching Cubes

[Pöthkow et al., 2011, Athawale et al., 2021]



$$D_i = [d_i^0, d_i^1, \ldots, \mathrm{d}_i^{\mathrm{M}}]$$

M instances of a random variable, e.g., ensemble members

Uncertainty modeling with multivariate Gaussian distribution, i.e., $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{cov})$

Means: $\mu_i = \frac{1}{M} \sum_{m=1}^{M} d_i^m$

Covariance Matrix:

$$cov_{i,j} = \frac{1}{M-1} \sum_{m=1}^{M} (d_i^m - \mu_i)(d_j^m - \mu_j)$$

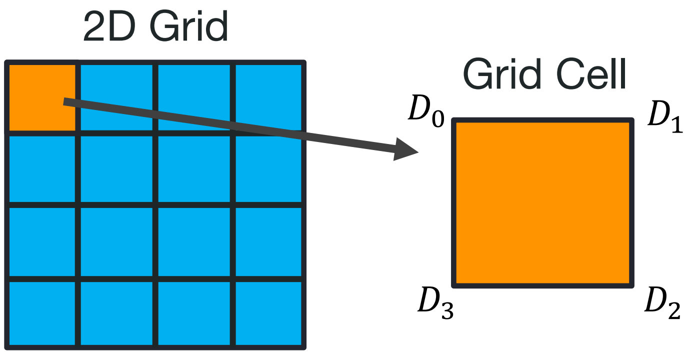
where i, j = 0, 1, 2, 3

Draw S Monte Carlo samples from $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{cov})$

Level-crossing probability $LCP = \frac{C}{S}$ if a level-set passes through C samples in a cell

Probabilistic Marching Cubes

[Pöthkow et al., 2011, Athawale et al., 2021]



$$D_i = [d_i^0, d_i^1, \dots, \mathrm{d}_i^{\mathrm{M}}]$$

!! Sampling a 4D space is computationally expensive

(Each multivariate Gaussian sample is 4D representing data at four vertices of a grid cell)

Uncertainty modeling with multivariate Gaussian distribution, i.e., $\mathcal{N}(\mu, cov)$

Means: $\mu_i = \frac{1}{M} \sum_{m=1}^{M} d_i^m$

Covariance Matrix:

$$cov_{i,j} = \frac{1}{M-1} \sum_{m=1}^{M} (d_i^m - \mu_i)(d_j^m - \mu_j)$$

where i, j = 0, 1, 2, 3

Draw S Monte Carlo samples from $\mathcal{N}(\mu, cov)$

Level-crossing probability $LCP = \frac{C}{S}$ if a level-set passes through C samples in a cell

Proposed Approach

We optimize this step to reduce the level of sampling and speedup computations

Draw S Monte Carlo samples from $\mathcal{N}(\mu, cov)$
Level-crossing probability $LCP = \frac{C}{S}$ if a level-set passes through C samples in a cell

(1) Eigenvalue decomposition technique

- Extract important *low-dimensional* structures (e.g., top eigenvalues)
- Perform sampling in a low-dimensional space (reduces the level of sampling)

(2) Adaptive probability model

- If *strong* correlation, use multivariate Gaussian sampling
- If *weak* correlation, use closed-form independent Gaussian models [Pöthkow et al., 2011, Athawale et al., 2021] (Closed-form solutions do not require sampling and are faster)

Eigenvalue Decomposition Technique

N-D sample: $\mathbf{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{cov})$

Mean: $\boldsymbol{\mu}$

Eigenvectors / Eigenvalues: **Eigenvalue decomposition of $\boldsymbol{cov}$**

Standard Normal Distribution (i.e., $\mathbf{Zi} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$)

$$\begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix} + \begin{bmatrix} \vec{V}_{11} & \cdots & \vec{V}_{n1} \\ \vdots & \ddots & \vdots \\ \vec{V}_{1n} & \cdots & \vec{V}_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1^{\frac{1}{2}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}$$

$$\boxed{D_j = \mu_j + \sum_{i=1}^{i=n} \vec{V}_{ij} \lambda_i^{0.5} Z_i}$$

Eigenvalue Decomposition Technique

★ ★ ★

N-D sample	Mean	Eigenvectors	Eigenvalues	Standard Normal Distribution (i.e., $\mathbf{Zi} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$)
$\mathbf{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{cov})$	$\boldsymbol{\mu}$	**Eigenvalue decomposition of $\boldsymbol{cov}$**		

$$\begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix} + \begin{bmatrix} \vec{V}_{11} & \cdots & \vec{V}_{n1} \\ \vdots & \ddots & \vdots \\ \vec{V}_{1n} & \cdots & \vec{V}_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1^{\frac{1}{2}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}$$

$$\boxed{D_j = \mu_j + \sum_{i=1}^{i=n} \vec{V}_{ij} \lambda_i^{0.5} Z_i}$$

Our idea: We do not need to loop over all eigenvalues if only a few eigenvalues are significant relative to others

Eigenvalue Decomposition Technique

★ ★ ★

N-D sample: $D \sim \mathcal{N}(\mu, cov)$

Mean: μ

Eigenvectors — **Eigenvalue decomposition of cov** — Eigenvalues

Standard Normal Distribution (i.e., $Zi \sim \mathcal{N}(0, 1)$)

$$\begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix} + \begin{bmatrix} \vec{V}_{11} & \cdots & \vec{V}_{n1} \\ \vdots & \ddots & \vdots \\ \vec{V}_{1n} & \cdots & \vec{V}_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1^{\frac{1}{2}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}$$

$$\boxed{D_j = \mu_j + \sum_{i=1}^{i=n} \vec{V}_{ij} \lambda_i^{0.5} Z_i}$$

Original formula

$\Rightarrow$

$$\boxed{D_j \approx \mu_j + \sum_{i=1}^{i=m} \vec{V}_{ij} \lambda_i^{0.5} Z_i}$$

Approximate formula (loop of m significant eigenvalues)

Eigenvalue Decomposition Technique

N-D sample $\mathbf{D} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{cov})$

Mean $\boldsymbol{\mu}$

Eigenvectors

Eigenvalues

Eigenvalue decomposition of $\boldsymbol{cov}$

Standard Normal Distribution (i.e., $\mathbf{Zi} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$)

$$\begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix} + \begin{bmatrix} \vec{V}_{11} & \cdots & \vec{V}_{n1} \\ \vdots & \ddots & \vdots \\ \vec{V}_{1n} & \cdots & \vec{V}_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1^{\frac{1}{2}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}$$

$$\boxed{D_j = \mu_j + \sum_{i=1}^{i=n} \vec{V}_{ij} \lambda_i^{0.5} Z_i}$$

Original formula

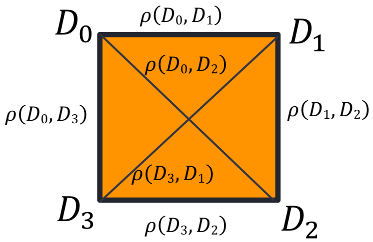
$$\Rightarrow$$

$$\boxed{D_j \approx \mu_j + \sum_{i=1}^{i=m} \vec{V}_{ij} \lambda_i^{0.5} Z_i}$$

Approximate formula (loop of m significant eigenvalues)

Corresponds to sampling of m-D space, and hence, achieves a speedup if m<n

Adaptive Probability Model



- Covariance matrix ***cov*** captures the correlation among data D_i. *What if the data are independent? Can we do better?*

- Our idea: Use the Pearson's correlation coefficient (ρ) to adaptively decide a probability model

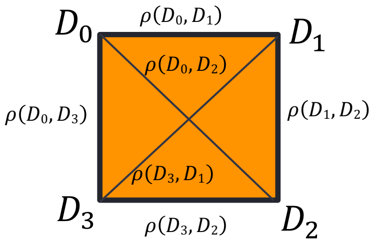
If $\max[\rho(D_i, D_j)] < 0.2\ \forall i, j$ and $i \neq j$

use independent model (no need of sampling, hence faster)

Else

use multivariate Gaussian model (slower) [we use the proposed eigenvalue decomposition approach]

Adaptive Probability Model



- Covariance matrix ***cov*** captures the correlation among data D_i. *What if the data are independent? Can we do better?*

- Our idea: Use the Pearson's correlation coefficient (ρ) to adaptively decide a probability model

If $\max[\rho(D_i, D_j)] < 0.2\ \forall i, j$ and $i \neq j$

use independent model (no need of sampling, hence faster)

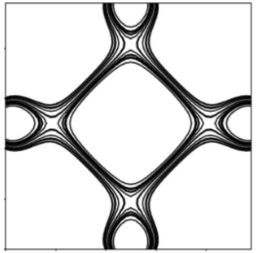
Else

use multivariate Gaussian model (slower) [we use the proposed eigenvalue decomposition approach]

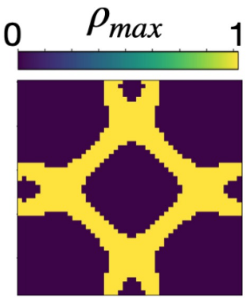
Drawback: Computation of pairwise correlation ρ is costly, but it can be precomputed since it does not depend on the **isovalue**

Results (Synthetic Data): Eigenvalue Decomposition Technique

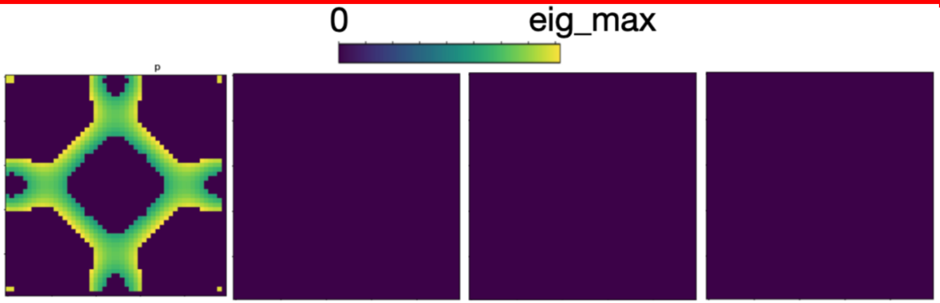
Ackley Function [Ackley, 1987]



Spaghetti plot (Correlated uncertain data)



Correlation (ρ)

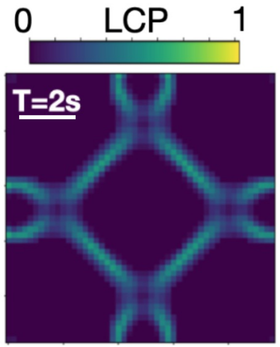


$\lambda_1 = 89.19$ $\lambda_2 = 0$ $\lambda_3 = 0$ $\lambda_4 = 0$

Eigenvalues (λ) [Most information is in the first eigenvector]

$$D_j = \mu_j + \sum_{i=1}^{i=n} \vec{V}_{ij} \lambda_i^{0.5} Z_i$$

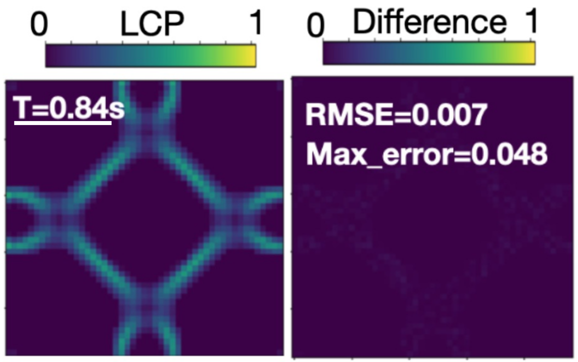
n=4



PMC (Original)

$$D_j \approx \mu_j + \sum_{i=1}^{i=m} \vec{V}_{ij} \lambda_i^{0.5} Z_i$$

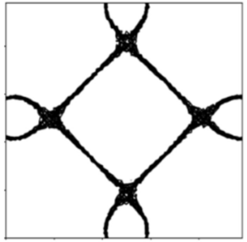
m=1



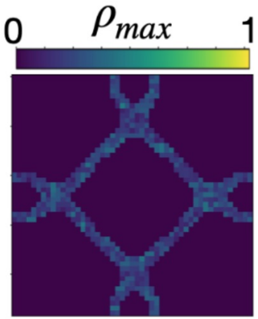
PMC (Eigenvalue-based)

Results (Synthetic Data): Adaptive Probability Model

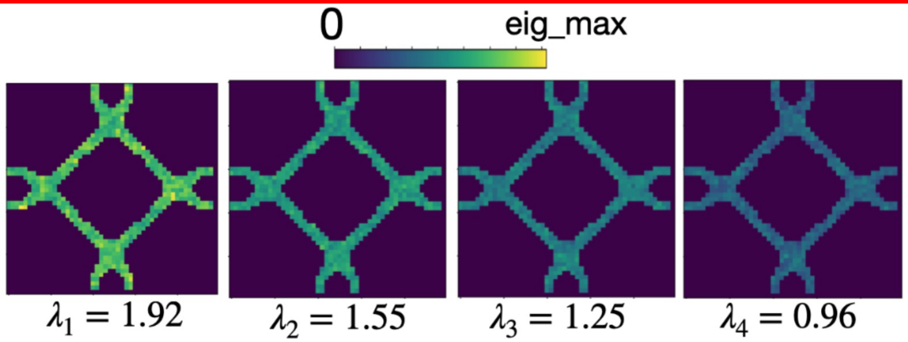
Ackley Function [Ackley, 1987]



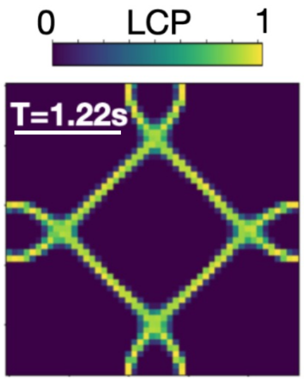
Spaghetti plot (Independent noise)



Correlation

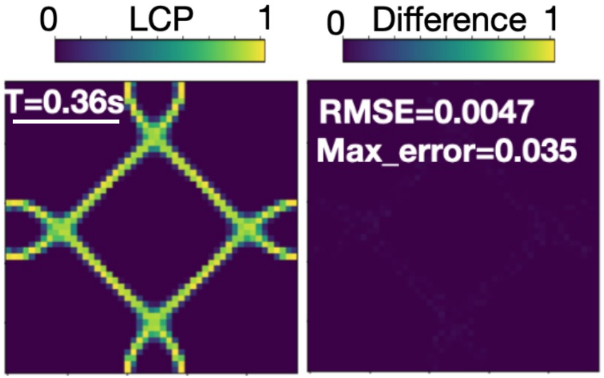


Eigenvalues (λ) [Information is in all eigenvectors]



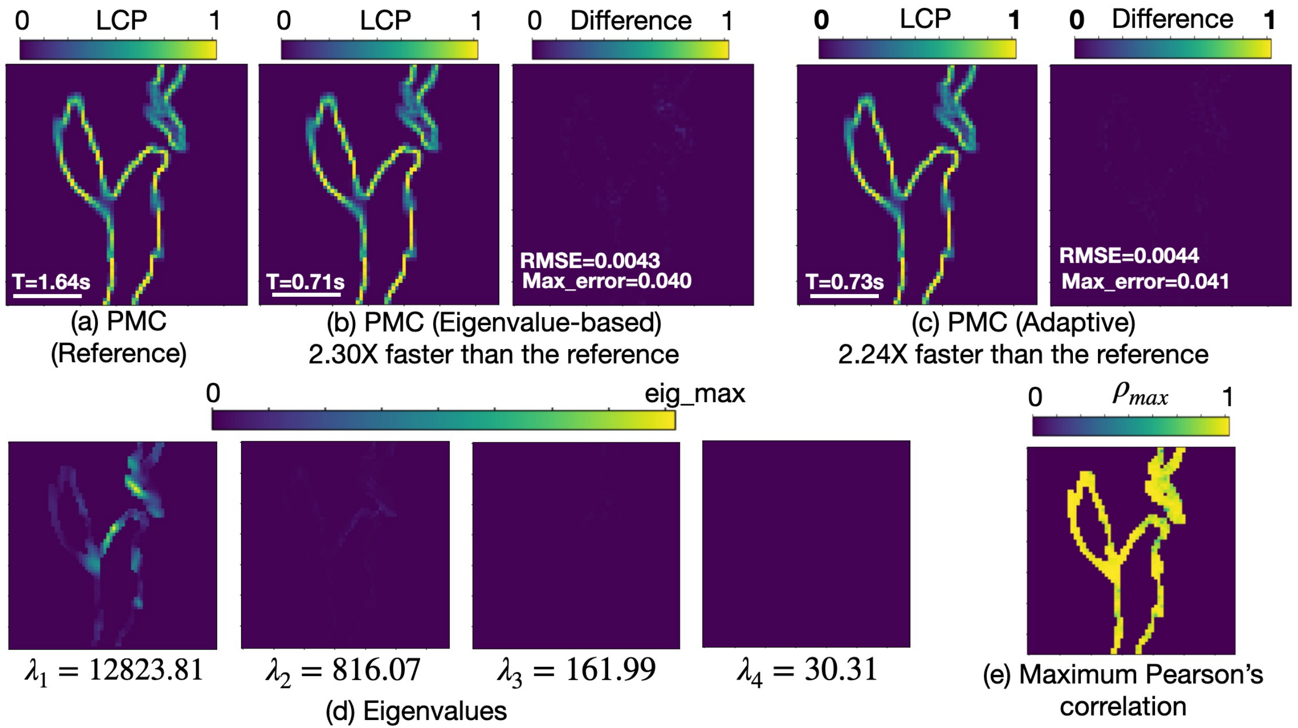
PMC (Original)

If $\rho_{max} < 0.2$:
Use fast independent model



PMC (Adaptive model)

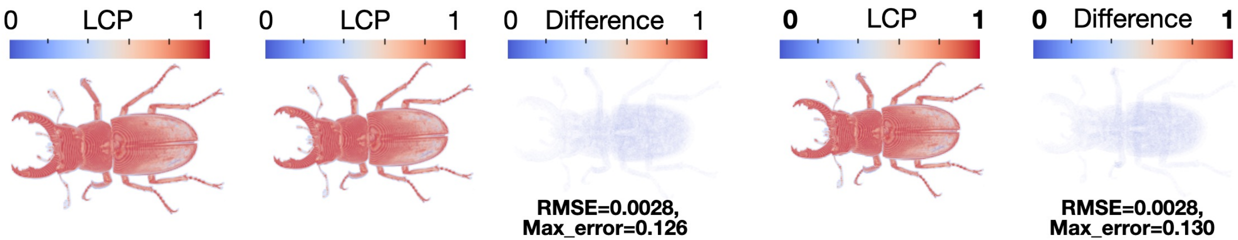
Results (Wind Dataset)



- Our proposed methods (b and c) are faster than the original PMC (a)
- Most information is in the first eigenvector
- Highly correlated uncertain data (So the both techniques provide a similar speedup)

Results (Beetle Dataset)

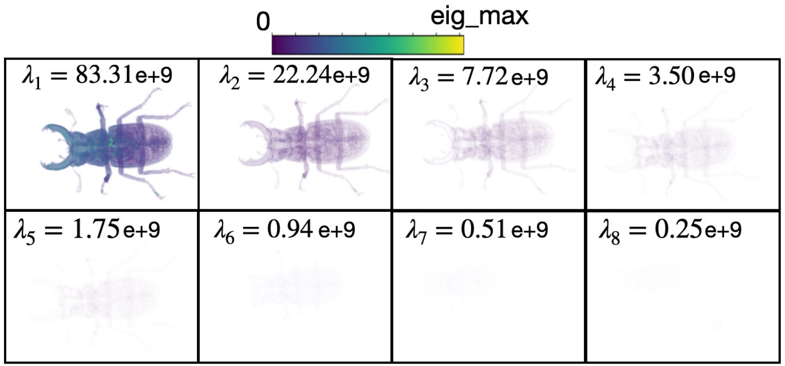
27.18 seconds **16.72 seconds** **17.09 seconds**



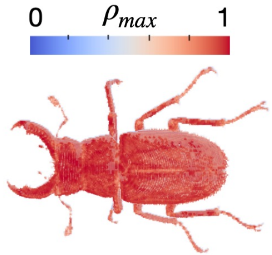
(f) PMC (Reference)

(g) PMC (Eigenvalue-based)
1.63X faster than the reference

(h) PMC (Adaptive)
1.59X faster than the reference



(i) Eigenvalues



(j) Maximum Pearson's correlation

Correlation precomputation time on a serial processor:
94.6 minutes

- Our proposed methods (b and c) are faster than the original PMC (a)
- Combined FunMC[2] [Wang et al., 2023] with our methods on the Frontier supercomputer
- Most information is in the four eigenvector
- Highly correlated uncertain data
(So the both techniques provide a similar speedup)

Conclusion and Future Work

- Novel data-driven solutions to reduce computational overhead of uncertainty visualization of level-sets
- *Eigenvalue decomposition* and *adaptive probability model* techniques to make adaptive compute decisions for a faster speed
- Integration with the FunMC^2 [Wang et al., 2023] filter to demonstrate the speedup on a Frontier supercomputer
- In the future, we will explore more data-driven methods for further acceleration and for various features, e.g., critical points, Morse complexes etc.

Acknowledgements

This work was supported in part by the U.S. Department of Energy (DOE) RAPIDS-2 SciDAC project under contract number DE-AC0500OR22725, the Intel OneAPI CoE, and the DOE Ab-initio Visualization for Innovative Science (AIVIS) grant 2428225.

Contact: tushar.athawale@gmail.com