CS677 Project

Parallel probabilistic data sampling algorithm implementation and performance scaling study

Group No. 4

Overall Plan

1. Implement the **Value-Based Sampling**, **Smoothness Based Sampling**, and **Joint Multi-Criteria** Sampling correctly and efficiently.
2. Evaluation and visualisation of the generated or sampled data, to validate correctness.
3. Parallelising all the sampling algorithms
4. Conduct the scaling study of the parallelised sampling algorithm

Parts and libraries/tools

Visualization

Libraries/Tools:

1. **VTK & pyevtk (in Python)**: For reading and writing vtk format datasets on disk
2. **VisIt**: Visualising original and sampled datasets
3. **NumPy and Matplotlib**: For processing (sampling) the datasets imported in python

Types of visualisation technique we would be using on our reconstructed data:

1. Volume-Based Visualization: Employing traditional ray casting-based techniques on both original and reconstructed data from the chosen percentage of sample points.

2. Isocontour-Based Visualization: We can specify any feature-specific iso values to render isosurfaces and visualise them interactively.

We would be using **Paraview** or **VisIt** to create the above visualisations and automate generation via respective libraries in python3.

Evaluation

1. **signal-to-noise ratio (SNR):**
    - To estimate the quality of the reconstruction.
    - SNR = $20*log(\sigma_{raw}/\sigma_{noise})$ where σ_{raw}= standard deviation of original data and σ_{noise} = standard deviation of the error of the reconstruction (the difference between the original data and the reconstruction).

- As the reconstruction improves, σ_{noise} gets smaller while σ_{raw} remains constant and the SNR increases. Larger values indicate better reconstructions.
- We will compare SNR values with sample ratio across random, value-based and join multi-criteria-based sampling methods.

2. **Correlation between the original and reconstructed datasets:**
    - Percentile scatter plot generating reconstructed data from random, value-based, and joint criteria-based sampling methods on sample data.
    - We would calculate the correlation coefficient between the reconstructed data value and the Original data value at the chosen sample ratio.
    - Compare these correlation coefficients with respective sample ratios across all three methods.

Parallelisation

The planned sampling algorithms would first be implemented using python3 in parallel using MPI.

Library/Tools:

1. **mpi4py (in Python):** A python3 wrapper over MPI with bindings of functions in C. We are going to perform the following algorithms in parallelisation:

    - **Simple Random Sampling (Just for Learning about the library/A sort of warmup):** Simple random sampling in mpi4py involves selecting random elements from a dataset distributed across multiple MPI processes. To achieve this, you can use a combination of numpy for generating random numbers and mpi4py for communication among processes.
    - **Value-Based Importance Sampling:** Value-based importance sampling is a statistical technique used in Monte Carlo simulations to estimate the expected value of a random variable by sampling from a different distribution, often referred to as the "importance distribution." This technique can be parallelised in a high-performance computing (HPC) environment, such as MPI (Message Passing Interface), to accelerate the computation by distributing the workload among multiple processes.
    - **Smoothness-based Importance Sampling:** Instead of just choosing an importance distribution with heavier tails, as in traditional importance sampling, smoothness-based importance sampling takes into account the smoothness or regularity of the function being integrated. This technique can also be parallelised in an MPI (Message Passing Interface) environment for high-performance computing. In this parallelised smoothness-based importance sampling approach, each MPI process generates its own target samples, computes the function values of interest, and contributes to the final estimate. The smoothness

factors are used to adjust the contribution of each sample based on how well it matches the smoothness characteristics of the function. We will adjust the number of samples and other parameters based on the accuracy and performance requirements of your specific application.

- **Joint Multi-Criteria Sampling:** Joint multi-criteria sampling, which combines both value-based and smoothness-based sampling strategies in parallel using MPI, can be an effective approach to estimate complex multidimensional integrals or expectations efficiently. This approach allows you to consider multiple criteria when selecting samples, enhancing the accuracy and convergence of Monte Carlo estimations. The parallelised joint multi-criteria sampling approach combines both value-based and smoothness-based criteria when selecting samples. We need to be sure to adapt the specific criteria, weights, and parameters based on our problem's requirements and characteristics.

Project Timeline

Duration	Expected Progress
25 September - 1 October	Verification of the correctness of Simple Random Sampling in its parallel paradigm by running it on different datasets. Perform the scaling study and visualisation of SRS with various datasets (To familiarise with the software interface and processing part).
2 October - 8 October	Implementation of the value-based sampling with the parallelisation and verification on some large datasets.
9 October - 15 October	Implementation of smoothness-based sampling with parallelisation and verification on some large datasets.
16 October - 22 October:	Implementation of joint multi-criteria sampling with parallelisation and verification on some large datasets.
23 October - 29 October:	Debugging and testing of all sampling techniques performed so far. Performing scaling study of all algorithms.
30 October - 5 November	Performing scaling study and creation of final report and documentation
5 November - 11 November	Evaluation Week.

Project Contribution

Student's Name	Corresponding Contributions
Siddharth Pathak	Understanding and Implementing Sampling Algorithms, i.e. Value-Based Sampling, Smoothness Based Sampling, and Joint Multi-Criteria.
Divya Gupta Desari Charithambika	Reconstructed Visualization and Evaluation as given in the paper. Perform a scaling study of all the parallelised algorithms.
Om Shivam Verma Palak Mishra	Parallelisation of the sampling algorithms using mpi4py and required skills and testing of the algorithm over a number of big datasets.

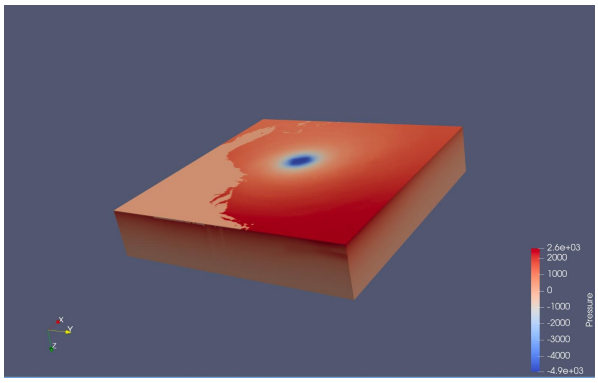
Workspace and Present Progress

All the work will be updated from time to time in the following GitHub repository:
https://github.com/omsv135/Parallel-Probabilistic-Data-Sampling

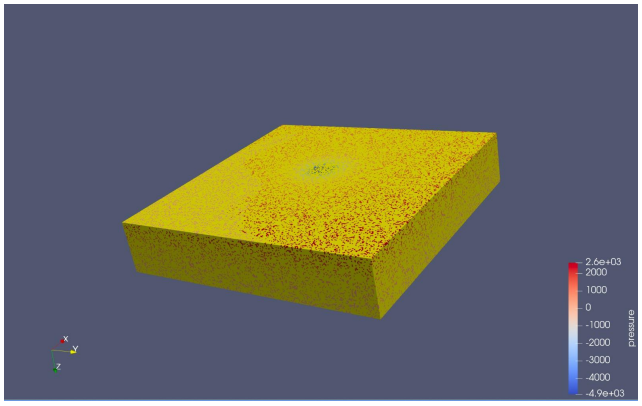
Till now, we have done the following tasks:

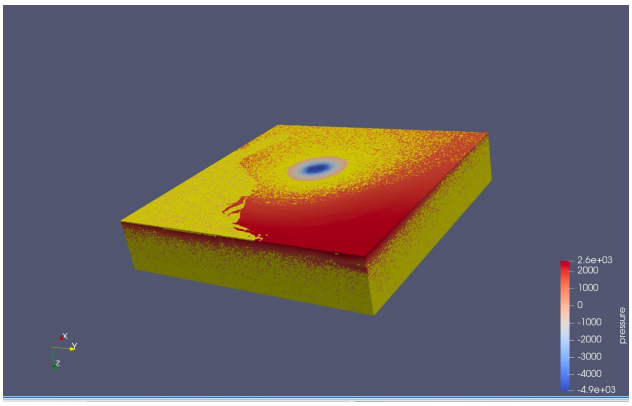
1. Implementation of Simple Random Sampling and its Parallelisation with MPI.
2. Development of an algorithm for Joint multi-criteria sampling technique as instructed in the paper.
3. Explored and studied various tools for evaluation and visualisation.

Original Dataset:

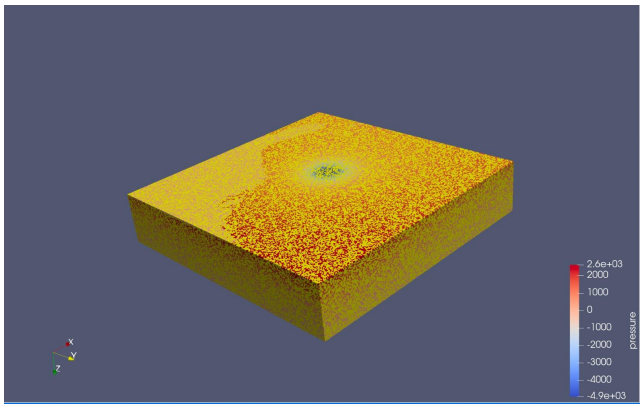


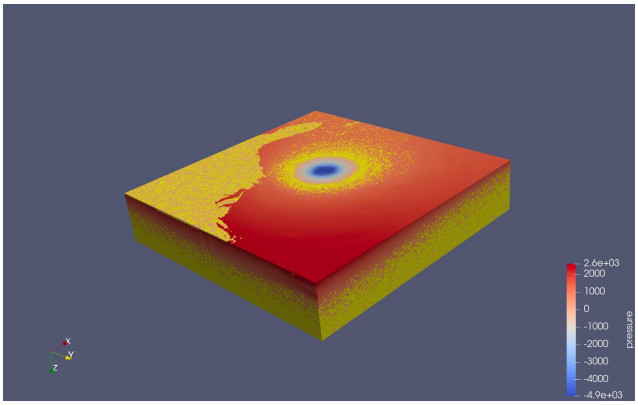
20% Sampling (Simple Random Sampling and Value-based sampling):



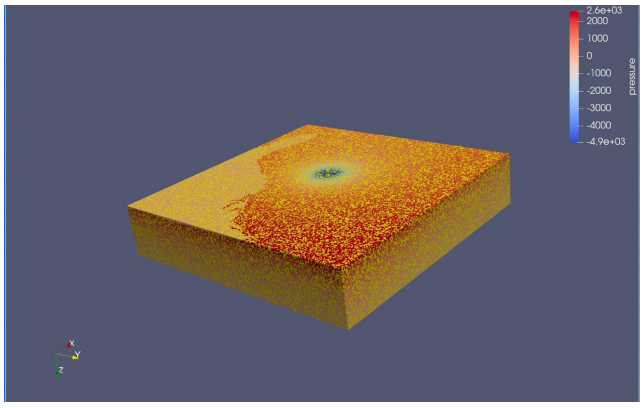


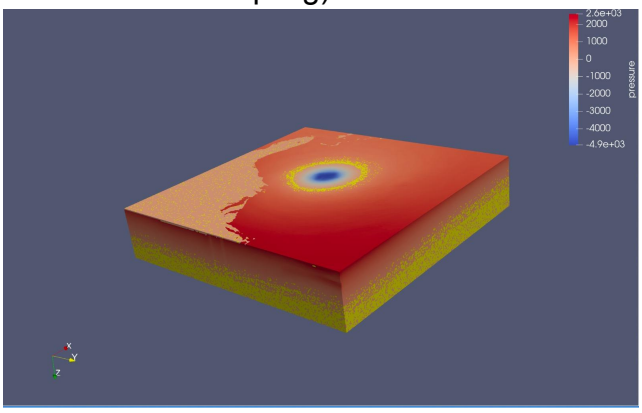
40% Sampling (Simple Random Sampling and Value-based sampling):





60% Sampling (Simple Random Sampling and Value-based sampling):





Algorithm 1 Fused Sampling Technique via Joint Multi-Criteria Sampling

Require: D (data), N (number of data points), M (number of samples), B (number of bins)
Ensure: I_F (importance function/histogram for selecting M samples from N data points)

$H \leftarrow histogram(D, N, B)$
$H \leftarrow sortascending(H)$
$I_F \leftarrow zeros(B^2)$
$C \leftarrow M/B$ ▷ Expected number of samples
$j = 0$
while $j < B$ **do**
  $c \leftarrow H[j]$ ▷ Count in j^{th} bin
  if $c_j < C$ **then** ▷ Case 1: Bin did not have sufficient data to sample
    $G_j \leftarrow computeGradient$(Points in j^th Bin)
    $G_j^{mag} \leftarrow computeGradientMagnitude(G_j)$
    $H_j \leftarrow histogram(G_i^{mag}, c_j, B)$
    $C_j \leftarrow c_j/B$
    $i = B - 1$
    while $i \geq 0$ and $c_j > 0$ **do** ▷ Smoothness Based sampling in j^{th} bin
      $c_{ji} \leftarrow H_j[i]$
      $I_F[B.j + i] = c_{ji}$
      $c_j = c_j - c_{ji}$
      $i = i - 1$
    end while
    $M \leftarrow M - c_j$
    $B \leftarrow B - 1$
    $C \leftarrow M/B$
    $j = j + 1$
  else ▷ Case 2: Bin has sufficient data to sample
    for $k \leftarrow j$ to B **do**
      $G_k \leftarrow computeGradient$(Points in k^th Bin)
      $G_k^{mag} \leftarrow computeGradientMagnitude(G_k)$
      $H_k \leftarrow histogram(G_k^{mag}, c_k, B)$
      $C_k \leftarrow c_k/B$
      $i = B - 1$
      while $i \geq 0$ and $c_k > 0$ **do** ▷ Smoothness Based sampling in all the remaining bins
        $c_{ki} \leftarrow H_k[i]$
        $I_F[B.k + i] = c_{ki}$
        $c_k = c_k - c_{ki}$
        $i = i - 1$
      end while
    end for
  end if
end while
for $j \leftarrow 1$ to B **do** ▷ Normalization of Importance function
  for $i \leftarrow 1$ B **do**
    $I_F[B.j + i] \leftarrow I_F[B.j + i]/H_j$
  end for
end for