ETH zürich

High Performance Computing for Science and Engineering I

P. Koumoutsakos
ETH Zentrum, CLT F 12
CH-8092 Zürich

Fall semester 2019

Exam

Issued: February 3, 2020, 09:00
Hand in: February 3, 2020, 12:00

Last Name:	
First Name:	
Student ID:	
Computer Hostname:	

With your signature you confirm that you:

- have read the exam directives,
- solved the exam on your own and without any external help,
- wrote your answers following the exam directives.

Signature:

Grading table

Question	Maximum score	Score	TA 1	TA 2
Question 1	20			
Question 2	25			
Question 3	25			
Question 4	20			
Question 5	25			
Question 6	20			
Question 7	20			

You can solve the questions in *any* order.

`NOTE: your final grade will be determined from the grade of` *this exam* `AND from your homework bonus (when applicable).`

Question 1: Parallel Scaling (20 points)

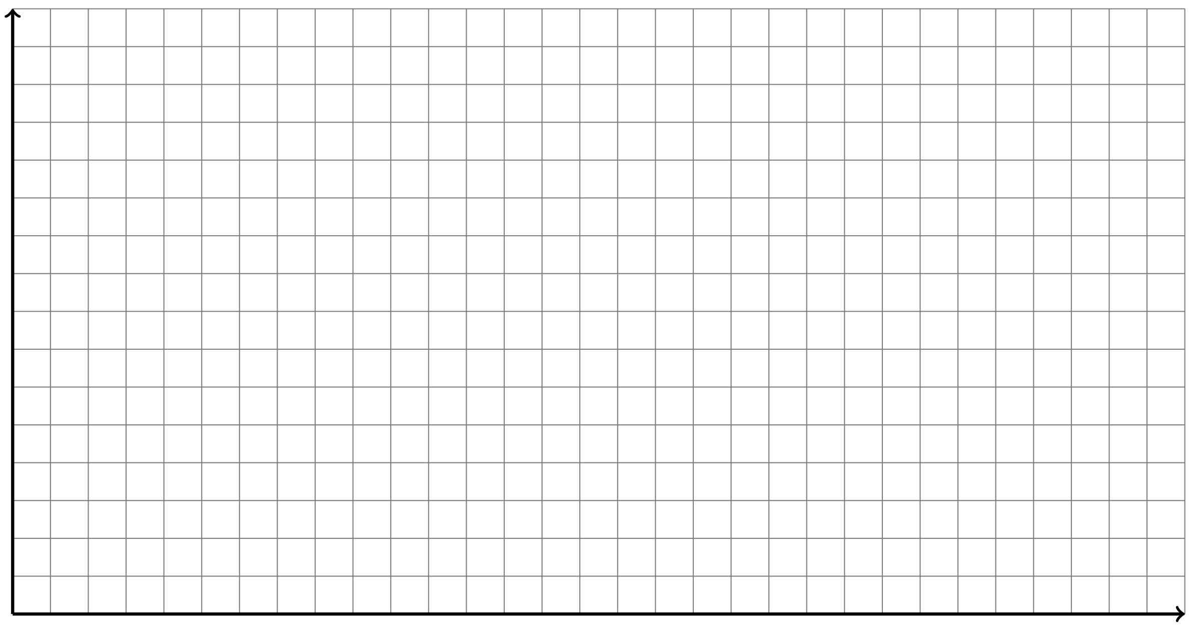
a) You are given a parallel code that computes the product of two square matrices of size N:

$$C = AB, \tag{1}$$

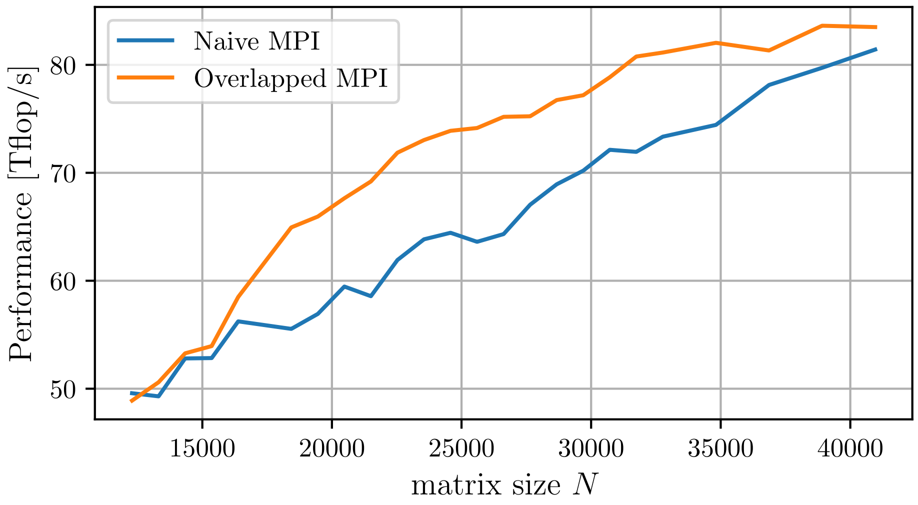
where $A, B, C \in \mathbb{R}^{N \times N}$.

Given the execution time depending on N and the number of processors P, draw four points of the weak scaling efficiency using the data for $N = 100$ and $P = 1$ as reference. Label the axes.

	runtime [ms]			
P	$N = 100$	$N = 200$	$N = 300$	$N = 400$
1	10.0	81.0	272.0	644.0
2	5.5	42.0	137.0	323.0
4	2.8	21.0	68.0	163.0
8	1.4	11.0	35.0	84.0
16	1.0	6.0	18.0	43.0
27	0.9	5.0	12.0	28.0
64	0.8	4.9	10.0	15.0



b) You are given two variants of the matrix-matrix multiplication described above, both parallelized using MPI. The first variant, *Naive MPI*, performs communication and computation operations in two different phases, achieving no overlap between them. The second one, *Overlapped MPI*, on the other hand, uses non-blocking communication and it has been optimized to overlap computation with communication. After benchmarking the performance of both variants, you produce the following *size scaling* plot:



The size scaling reports the floating point performance as a function of the problem size N running on 512 supercomputer nodes (16k cores). You are told by the author's code that the variants' running time reflects the following formulae:

$$T_{\text{naive}}(N) = T_{\text{computation}}(N) + T_{\text{communication}}(N) + T_{\text{startup}}, \tag{2}$$

$$T_{\text{overlap}}(N) = \max\{T_{\text{computation}}(N), T_{\text{communication}}(N)\} + T_{\text{startup}}, \tag{3}$$

where $T_{\text{computation}}(N) \propto N^3$ represents the time to compute a submatrix multiplication, $T_{\text{commmunication}}(N) \propto N^2$ represents the time spent in communication among neighboring ranks and T_{startup} is a positive constant representing the minimum time to create and push MPI messages.

Using this information, provide a succinct justification of the observed behavior:

i) Why does the *overlap* variant generally perform better than the *naive* variant?

ii) Why does the performance increase for both variants as problem size also increases?

iii) Why is the performance so low for both variants when using a small N (~ 12000)?

iv) Why does the performance gap between the two variants become smaller towards large N (~ 40000)?

Question 2: Particles and Grids (25 points)

We wish to compare Eulerian and semi-Lagrangian numerical methods and their HPC advantages and drawbacks using as an example the 1D advection equation for a field $q(x,t)$ with a constant velocity a

$$\frac{\partial q}{\partial t} + a\frac{\partial q}{\partial x} = 0 \tag{4}$$

and initial condition $q(x, t = 0) = q_0(x)$ on the domain $(-\infty, \infty)$.

In an Eulerian frame the numerical solution of this equation can be obtained by a finite difference method. Space is discretized with a uniform grid of step size h and time with a uniform time step Δt. The quantity $q_i^n \approx q(x_i, t^n)$ approximates q at grid node x_i on time t^n. The solution at $t^{n+1} = t^n + \Delta t$ can be obtained by various schemes, for example the first order upwind scheme for $a > 0$

$$q_i^{n+1} = q_i^n - \lambda(q_i^n - q_{i-1}^n), \tag{5}$$

where $\lambda = a\Delta t/h$ is the CFL number.

In a Lagrangian frame we solve Equation (4) using particles that are advected and remeshed on a uniform grid. The field $q(x,t)$ can be approximated by N particles, with strength Q_p, $p = 1, \ldots, N$, initially distributed on equidistant locations x_p with a spacing h along the x-axis. The particle strength can then be defined as $Q_p = q(x_p, t = 0)h$ and the field is approximated as

$$q^h(x,t) = \sum_{p=1}^{N} Q_p(t)\,\delta(x - x_p(t)) \tag{6}$$

where δ is the Dirac delta functional.

In a purely Lagrangian particle method the field q evolves by evolving the particle strengths and locations by a set of ordinary differential equations

$$\frac{\mathrm{d}Q_p}{\mathrm{d}t} = 0 \tag{7}$$

$$\frac{\mathrm{d}x_p}{\mathrm{d}t} = a. \tag{8}$$

In a semi-Lagrangian particle method the particles are advected by one time step and then their strengths are projected onto a uniform grid with spacing h. The steps to evolve the particles in a semi-Lagrangian method are as follows:

0. Initialize the point particles and strengths.
1. Evolve the discretized Equations (7) and (8) by one time step Δt.
2. Remesh/project the strength of the advected particles onto a grid with uniform spacing h so that moments $M_m = \int q(x,t)x^m dx$ of the pre-remeshed and remeshed fields are conserved. After remeshing the "old" particles are discarded and the "new" particle set is located on the grid nodes with the remeshed strengths.
3. Continue with step 1.

a) Advect the particles for one time step Δt. Use an explicit Euler time integrator and maintain the CFL condition (i.e. $|\lambda| < 1$). Justify the choice of an explicit Euler integrator for this equation.

In the following assume $a > 0$.

b) After the advection step, remesh/redistribute the particle strength Q_p carried by particle p at position x_p and at time t^{n+1} to its *two nearest* grid points. Derive the interpolation weights for the grid nodes.

   Hint: Ensure that the new particle weights conserve the moments $m \in \{0, 1\}$ of the field q.

c) Change your perspective from particles to grid nodes: Express the evolution of the particle strengths for a grid node x_i over time. What do you observe?

d) Discuss the parallel implementation of the particle-mesh algorithm you just derived. Compare the parallel implementation of the particle-mesh algorithm to a parallel implementation of the grid-based upwind scheme shown in Equation (5).

e) Discuss advantages and drawbacks of the purely Lagrangian method, the semi-Lagrangian (particle-mesh) method you derived and the finite difference method you were given. For this advection equation which one is the most accurate method?

Question 3: Principal Component Analysis (25 points)

a) Assume that x is uniformly distributed in $[-1, 1]$. Calculate the second moment $\mathbb{E}[x^2]$.

b) Assume that x is uniformly distributed in $[-2, 2]$. How does the second moment $\mathbb{E}[x^2]$ change? (calculate the value).

c) You are given a two dimensional dataset plotted in Figure 1, where the components x and y are uncorrelated and uniformly distributed, i.e. $x \sim \mathcal{U}([-2, 2])$ and $y \sim \mathcal{U}([-1, 1])$. **Calculate** the data covariance matrix and write down the trivial eigenvectors and eigenvalues. **Plot** the eigenvectors in the figure.

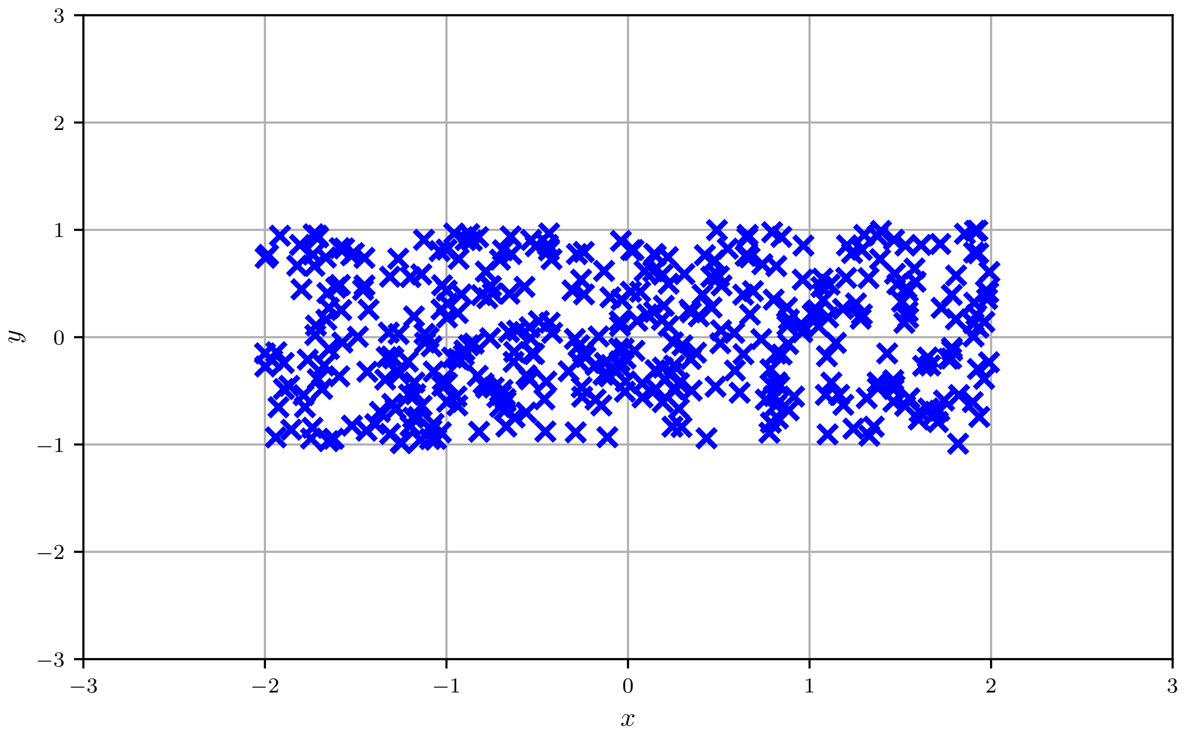


Figure 1

In the following, you are given a collection of $N = 8$ data points in a two dimensional space:

$$X = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 3 \\ 3 & 2 \\ 3 & 4 \\ 4 & 3 \\ 4 & 4 \\ 5 & 6 \end{pmatrix}, \tag{9}$$

with $X \in \mathbb{R}^{N \times D}$, plotted in Figure 2.

d) Center the data and calculate the data covariance matrix.

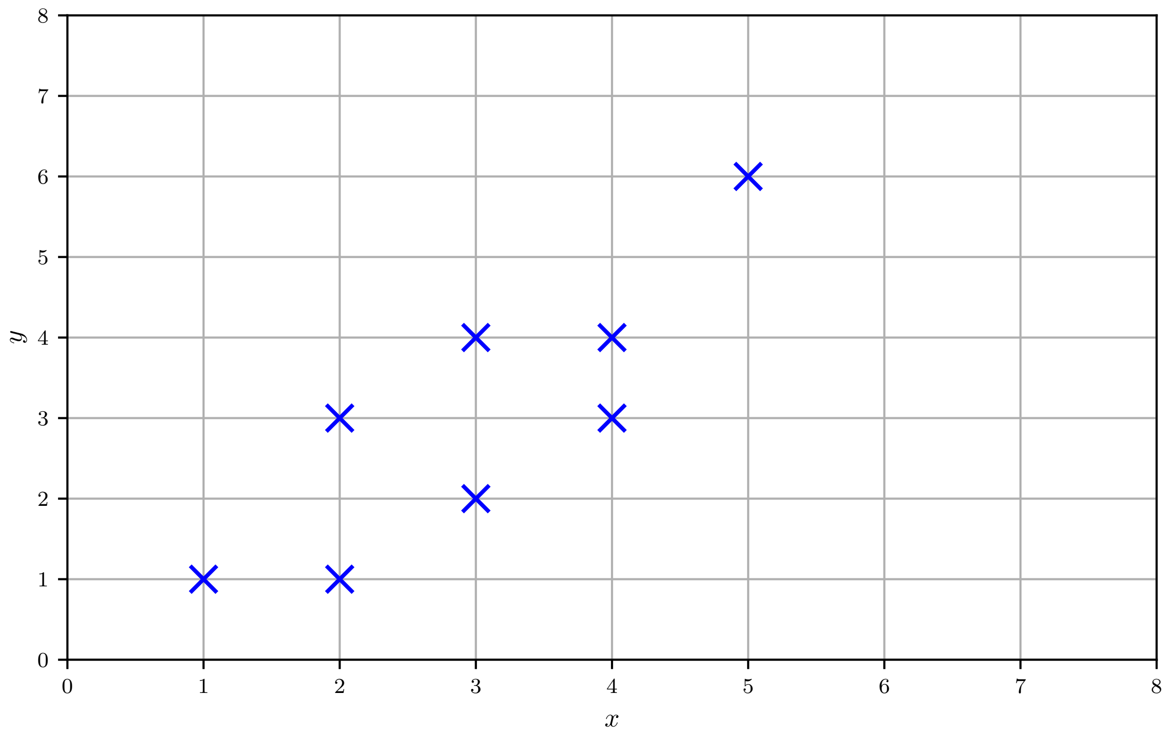


Figure 2

e) You are given that the eigenvalues of the covariance matrix are $\lambda_1 = 4.23$ and $\lambda_2 = 0.34$. **Compute** the principal eigenvector $\boldsymbol{u}_1$ of the data covariance matrix and **sketch** it in the figure. *Hint*: Use the fact that $C\boldsymbol{u} = \lambda\boldsymbol{u}$ and $|\boldsymbol{u}|_2 = 1$ for an eigenvector, and draw the eigenvector starting from the mean.

f) If we project the data to a reduced order space using the principal component computed by the first eigenvector of the covariance matrix, how much variance of the data is retained in the reduced order space?

g) Use the principal component $\boldsymbol{u}_1$ computed in the previous questions to project the **centered** data in a reduced order subspace. Compute the variance of the data in the original 2-D space, and the variance of the projected data. How much variance (in percentage) is retained in the reduced order space? Does this number agree with the value computed in the previous question, where we utilized the eigenvalues?

   Hint: Use the unbiased estimator $\sigma_X^2 = \frac{1}{N-1}\sum_{i=1}^{N} X_i^2$ for the variance for a random variable X. In multiple dimensions, the total variance is summed across dimensions.

h) The computational cost of many data-driven algorithms (e.g. nearest neighbor search or classification) scales quadratically, i.e. $\mathcal{O}(N^2)$, or even cubically $\mathcal{O}(N^3)$ with the data dimension N. In the era of big data, many datasets have a large dimensionality rendering the direct application of such algorithms computationally intractable. How can the PCA be used to alleviate the problem? Use at most five sentences.

Question 4: Roofline Model (20 points)

a) Describe in at most five sentences how the roofline model works.

b) In the lecture we have discussed various forms of parallelism. You are the code development team leader in a company and you have to explain to your executive the steps your team will take to improve the performance of a compute bound code without optimization. List the forms of parallelism your team is going to exploit (in the order you implement them) and explain in one sentence (for each parallel form) why you have chosen it at this position.

c) We are given two processors A and B:

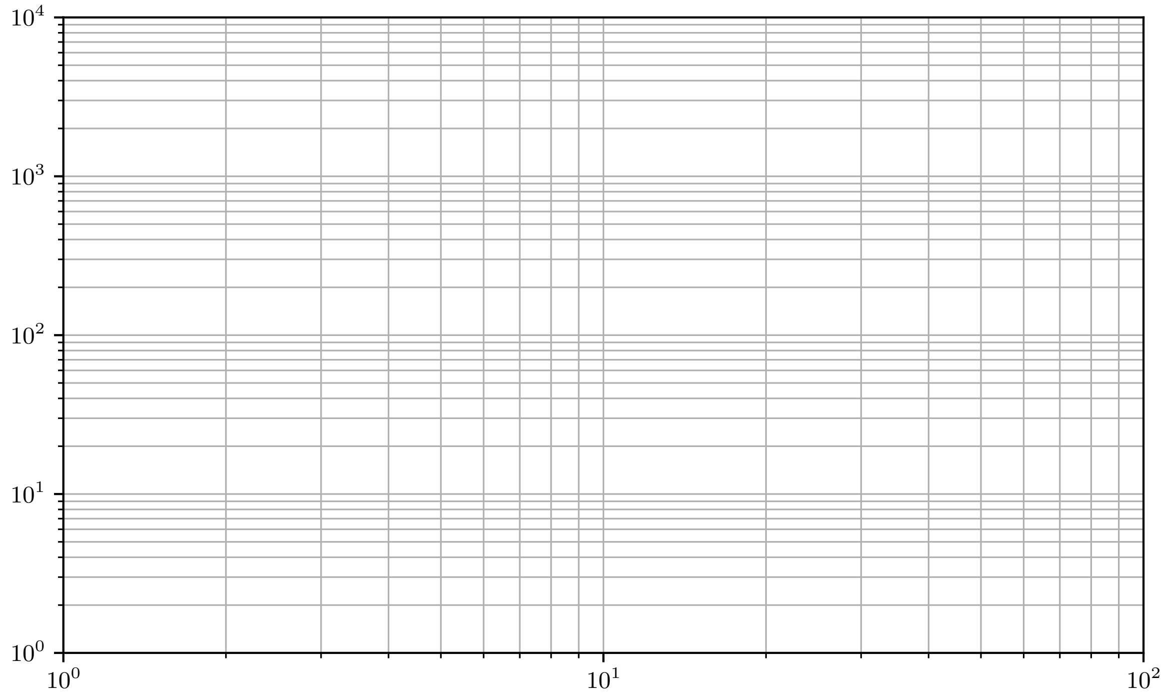
Processor	A	B
Frequency	2.5 GHz	2.0 GHz
Cores	4	16
Max. Memory Bandwidth	20 GB/s	100 GB/s
SIMD Register Width	128 bit	512 bit
FMA Instructions	No	Yes

Processor B is superscalar with a pipeline that supports simultaneous execution of *two* FMA instructions (fused-multiply-add) in the same cycle. Processor A is scalar *without* support for FMA. For processor A and B, determine:

i) The peak floating point performance π for single precision data

ii) The ridge point I_β given the processor peak floating point performance π and memory bandwidth β.

Draw the roofline ceilings for both processors A and B in the plot below. Do not forget to label the axes.



d) The discrete Fourier transform (DFT) of the real signal $\boldsymbol{x} \in \mathbb{R}^N$ is given by

$$X_k = \frac{1}{N} \sum_{j=0}^{N-1} x_j \omega_N^{-jk}, \tag{10}$$

where $\omega_N = e^{\frac{i2\pi}{N}}$ is the N-th root of unity. Assume that $k = 0, 1, \dots, N-1$ and that you have pre-computed the constant factors ω_N^{-jk}. Note that the factors $\omega_N^{-jk} \in \mathbb{C}$ and the transform $X_k \in \mathbb{C}$ are **complex**. Complex numbers are stored in memory with a real part and imaginary part, both are individual floating point numbers.

State your further assumptions for the questions below, if you make any.

i) What is the operational intensity of the DFT in double precision?

ii) What is the computational complexity of the DFT algorithm.

iii) The famous fast Fourier transform (FFT; J. Cooley and J. Tukey, 1965) has a complexity of $\mathcal{O}(N \log_2 N)$. Compared to your analysis of the DFT, how much faster/slower do you expect the FFT to be for a signal size of $N = 1024$ elements?

Question 5: Particle Strength Exchange for advection (25 points)

Consider the one dimensional advection equation

$$\frac{\partial q(x,t)}{\partial t} + a\frac{\partial q(x,t)}{\partial x} = 0 \qquad (11)$$

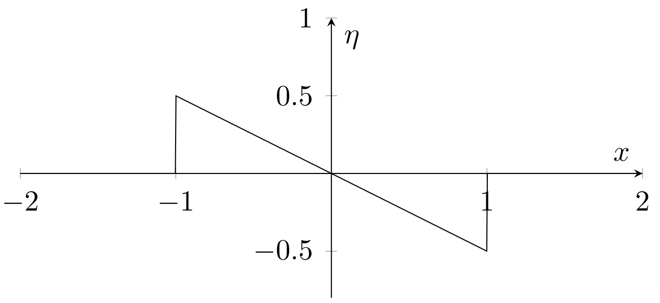
where a is a constant propagation speed. We are interested in solving Equation (11) with the Particle Strength Exchange method.

a) Starting with a Taylor series expansion of $q(y,t)$ around point x, derive an expression for the derivative $\frac{\partial q(x,t)}{\partial x}$ in terms of an integral operator.

b) We want to approximate $\frac{\partial q(x,t)}{\partial x}$ up to second order accuracy. What properties should the kernel $\eta_\varepsilon(x)$ satisfy?

c) Use the mid-point rule to retain a second order accurate approximation, and approximate the integral operator of $\frac{\partial q(x,t)}{\partial x}$ on particle locations x_p.

d) Assume that particles are constrained to a uniform grid with grid spacing h, and the volume of a particle is $\mathrm{Vol}_p = h$. Using the kernel $\eta_\varepsilon(x) = \varepsilon^{-1}\eta(x/\varepsilon)$ given by



$$\eta(x) = \begin{cases} -\frac{1}{2}x, & \text{if } |x| \le 1 \\ 0, & \text{otherwise} \end{cases}$$

derive the discretization scheme for the advection Equation (11) and relate it to finite differences. Discretize the time derivative using a backward Euler scheme and assume $\varepsilon = h$.

Question 6: Histogram with hybrid MPI + OpenMP (20 points)

You are studying a mysterious process which generates random samples and you would like to visualize their distribution. Your task is to compute the histogram of these samples, using a **hybrid MPI + OpenMP** approach. The provided skeleton code:

- Creates samples from the mysterious process;
- Dumps the computed histogram to a file "hist.txt".

The histogram file can be visualized by typing `make plot` (given "hist.txt", it will produce an image "hist.png").

The following steps must be taken in order to produce the histogram:

- Find the minimum value l and the maximum value u of all the samples;
- Compute the number of samples falling into each of the n bins, where the i^{th} bin contains sample values in the interval $[l + ih, l + (i + 1)h)$ and $h = (u - l)/n$.

Write your solution in `~/questions/coding_1_hybrid_histograms/main.cpp`.

a) Fill in the provided code to compute the histograms. Parallelize your code using MPI. Note: Each rank must call the `mysteriousProcess` function and store its own set of samples. The `seed` argument to `mysteriousProcess` must be different between the ranks to guarantee independent samples.

b) Parallelize the histogram computation further with OpenMP in order to obtain a **hybrid MPI + OpenMP** code. Avoid as much critical sections and atomic operations as possible. Consider that the number of samples is much larger than the number of bins.

Question 7: Monte Carlo, OpenMP and Diffusion (20 points)

We consider the steady state solution of the two-dimensional diffusion equation in the domain $\Omega := [0,\ 1]^2$:

$$\begin{cases} \frac{\partial^2 \rho(x,y)}{\partial x^2} + \frac{\partial^2 \rho(x,y)}{\partial y^2} = 0 & \text{in } (x,y) \in (0,1) \times (0,1) \\ \rho(x,y) = f(x,y) & \text{for } x \in \{0,1\} \text{ or } y \in \{0,1\}. \end{cases} \tag{12}$$

Here, $\rho(x,y)$ is the measure of the diffusing scalar quantity at position (x,y). We wish to approximate the solution at a single point $(x_0, y_0) \in \Omega$ using the Feynman-Kac method with Monte-Carlo sampling. The algorithm is as follows:

- Perform two-dimensional random walks starting from position $(x_0, y_0) \in \Omega$. Each step i of the walk increments the position along steps of constant size $d > 0$ and random direction $\theta_i \sim \mathcal{U}[0, 2\pi]$:

$$x_i = x_{i-1} + d\cos(\theta_i), \qquad y_i = y_{i-1} + d\sin(\theta_i), \tag{13}$$

- Each random walk terminates when it leaves the domain Ω; the location where the walk crosses the boundary is denoted as $(x_{\text{cross}}, y_{\text{cross}})$.
- The steady state solution of the diffusion equation in (x_0, y_0) is approximated by the expected boundary value $f(x_{\text{cross}}, y_{\text{cross}})$, i.e.:

$$\rho(x_0, y_0) = \mathbb{E}\left[f(x_{\text{cross}}, y_{\text{cross}})\right]. \tag{14}$$

Equation (14) can be approximated by Monte-Carlo sampling. Consider N independent random walks $X^{(1)}(x_0, y_0),\ X^{(2)}(x_0, y_0),\ \ldots,\ X^{(N)}(x_0, y_0)$, all starting from location (x_0, y_0), and then crossing the boundaries of the domain Ω at different locations $(x^{(1)}_{\text{cross}}, y^{(1)}_{\text{cross}}),\ (x^{(2)}_{\text{cross}}, y^{(2)}_{\text{cross}}),\ \ldots,\ (x^{(N)}_{\text{cross}}, y^{(N)}_{\text{cross}})$. The Monte-Carlo estimate of $\rho(x_0, y_0)$ is given by:

$$\rho(x_0, y_0) = \mathbb{E}_N\left[f(x_{\text{cross}}, y_{\text{cross}})\right] = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}_{\text{cross}}, y^{(i)}_{\text{cross}}) \tag{15}$$

Write an OpenMP parallel program which approximates the steady state solution at $x_0 = 0.3$ and $y_0 = 0.4$ using the Monte Carlo integration scheme based on random walks with step size $d = 0.01$. The function on the boundary (in Equation (12)) is given by $f(x,y) = x$. Perform batches of 100 random walks in parallel and stop when the sample standard deviation $s_N\left[f(x_{\text{cross}}, y_{\text{cross}})\right]$ is less than $\epsilon\mathbb{E}_N\left[f(x_{\text{cross}}, y_{\text{cross}})\right]$ with tolerance $\epsilon = 0.01$. Note that the sample standard deviation is given by:

$$s_N\left[f(x_{\text{cross}}, y_{\text{cross}})\right] = \sqrt{\frac{\mathbb{E}_N\left[f(x_{\text{cross}}, y_{\text{cross}})^2\right] - \mathbb{E}_N\left[f(x_{\text{cross}}, y_{\text{cross}})\right]^2}{N-1}} \tag{16}$$

and for the estimation of the expectation value you can use the formula presented in Equation (15).

Write your solution in

`~/questions/coding_2_omp_MC/main.cpp`

and adapt the `Makefile` *in the same folder to compile your program using* `make`.