P. Koumoutsakos., S. Martin
ETH Zentrum, ML H44
CH-8092 Zürich

Mockup Exam

Issued: December 17, 2021, 12:15

Last Name:	
First Name:	
Student ID:	

With your signature you confirm that you:

- Have read the exam directives
- You solved the exam without any unauthorized help
- You wrote your answers following the outlined directives

Signature: _____

Exam directives. In order to pass the exam, the following requirements have to be met:

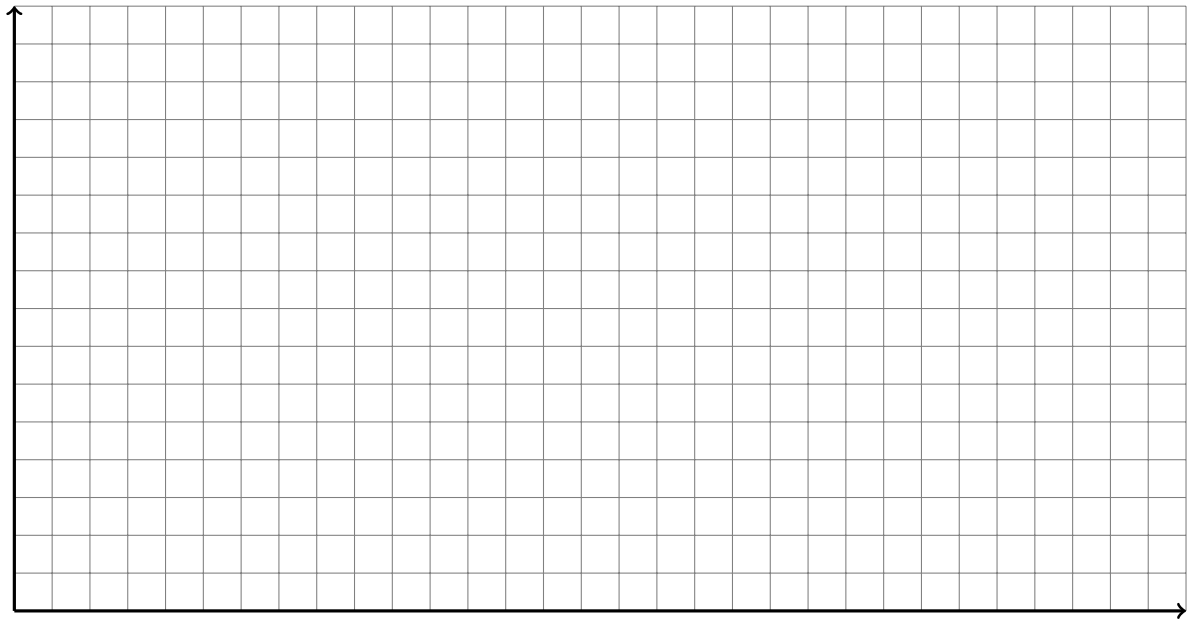
- Clear your desk (no cell phones, cameras, etc.): on your desk you should only have your Legi, your pen and your notes. We provide you with the necessary paper and exam sheets.
- Carefully read the first pages of the exam. Write your name and student ID where requested. Before handing in the exam, **SIGN ON THE FIRST PAGE**.
- Your notes (personal summary) should consist of **no more than four** A4 sheets (eight pages). The personal summary **must be handwritten**. You are not allowed to bring a copy of somebody else's summary.
- Your answers should be handwritten in blue or black pen (no pencils), clearly readable and in English. Only one answer per question is accepted. Invalid answers should be clearly crossed out.
- To answer new questions (e.g. Question 1, not sub-questions!), always use a new page. On the top-right corner of every page write your complete name and Legi-ID. Unless otherwise noted in the question, you should always hand-in your answers on paper!
- You must hand in: the exam cover, any extra notes provided by us, the sheets with the exam questions and your solutions. The exam will not be accepted if any of these items are missing.
- If something is disturbing you during the exam or preventing you from peacefully solving the exam, please report it immediately to an assistant. Later complaints will not be accepted.

Question 1: Parallel Scaling

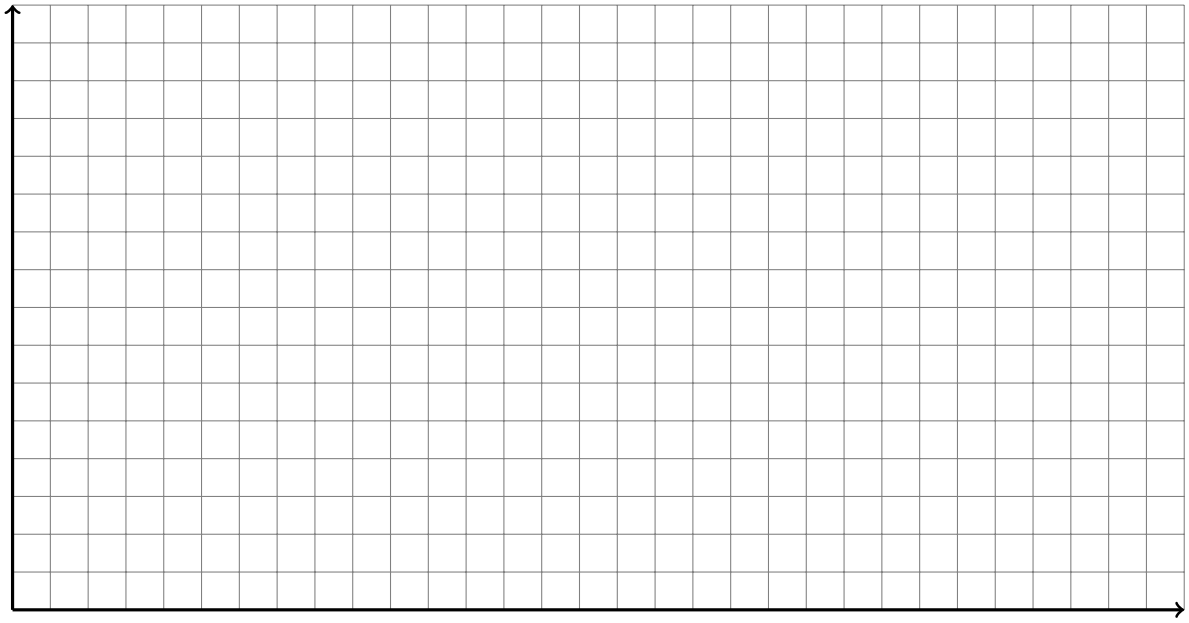
a) Assume we implemented an $\mathcal{O}(N^2)$ brute force solver for an N-body problem. The following table reports timing results for the solver for various number of particles N on P processor cores with a fixed number of time steps:

	runtime [s]			
P	$N = 1000$	$N = 2000$	$N = 4000$	$N = 8000$
1	12.0	60.0	216.0	816.0
2	6.0	30.0	108.0	408.0
4	4.0	15.0	54.0	204.0
8	3.0	9.0	31.0	110.0
16	2.4	6.0	21.6	54.4

1. Draw at least four points of the strong scaling plot for this program using the value for $N = 1000$ at $P = 1$ as a reference. On the solution sheet show all steps of your calculations. Don't forget to label the axes.



2. Draw at least three points of the weak scaling plot for this program, using the value for $N = 1000$ at $P = 1$ as a reference to estimate the parallelization overhead. On the solution sheet show all steps of your calculations. Don't forget to label the axes.



b) 1. Assume you have 10 cores that you can use to solve a problem in parallel and that 98% of your code is theoretically possible to parallelize. Can you get a speedup of 7? If so, how many cores are needed at least?

2. A programmer parallelized so far a part of the code that accounts for 60% of execution time. What is the current maximum speedup he can hope to achieve?

Question 2: Parallel Dependencies and Bug Hunting

a) Identify and explain any issues in the following OpenMP code snippet and propose a solution.

```
1     #pragma omp parallel for collapse(2)
2     for (int i=0; i<N; i++) {
3        y[i] = 0;
4        for (int j=0; j<N; j++)
5         y[i] += A[i][j] * x[j];
6     }
```

b) In the following fragments assume that all buffers have been allocated with sufficient size. Moreover, `rank` and `size` denote the rank of each process and the total number of MPI processes, respectively.

For each fragment note whether it deadlocks or not and explain why. In case of deadlock, propose a solution. Report any possible performance issues.

```
1     // block1.c
2     int ireq = 0;
3     for (int p=0; p<size; p++)
4     if (p!=rank)
5     MPI_Isend(sbuffers[p],buflen,MPI_INT,p,0,comm,&(reqs[ireq++]));
6     for (int p=0; p<size; p++)
7     if (p!=rank)
8     MPI_Recv(rbuffer,buflen,MPI_INT,p,0,comm,MPI_STATUS_IGNORE);
9     MPI_Waitall(size-1,reqs,MPI_STATUSES_IGNORE);
```

```
1     // block2.c
2     int ireq = 0;
3     for (int p=0; p<size; p++)
4     if (p!=rank)
5     MPI_Irecv(rbuffers[p],buflen,MPI_INT,p,0,comm,&(reqs[ireq++]));
6     MPI_Waitall(size-1,reqs,MPI_STATUSES_IGNORE);
7     for (int p=0; p<size; p++)
8     if (p!=rank)
9     MPI_Send(sbuffer,buflen,MPI_INT,p,0,comm);
```

c) For a known vector $A \in \mathbb{R}^N$, you are asked to compute

$$B_i = \sum_{j=0}^{K-1} A_{i+j} \ , \ i = 0, \ldots, N-K-1, \ , \ K \ll N \tag{1}$$

The following code is a serial implementation of the computation shown above:

```
1   // Assume B[...] are initialized to 0.
2   for (int i = 0; i < N - K; ++i)
3       for (int j = 0; j < K; ++j)
4           B[i] += A[i + j];
```

This loop can be parallelized by adding `#pragma omp parallel for` before line 2. Assuming T available threads, the complexity of that loop would by $\mathcal{O}(NK/T)$. Show that

$B_i = B_{i-1} + A_{K+i-1} - A_{i-1}$, $i = 1, \ldots, N-K-1$ and provide a serial code that implements this recursive relation (you still need to use the non-recursive formula to compute B_0).

d) The complexity of the serial implementation you provided in the previous subquestion is $\mathcal{O}(N+K)$. Propose a parallel implementation with OPENMP and T threads with has lower complexity. You can assume that N is divisible by T. What is the complexity of your solution? Hint: the optimal solution can be obtained by using only one `#pragma omp parallel` region without any `#pragma omp for` directives.

Question 3: Operational Intensity & Roofline Model

The operational intensity is a metric that relates the amount of work W (operations) to the number of bytes Q (data) that need to be transferred and is defined as

$$I = \frac{W}{Q} \quad \text{[ops/byte]}. \tag{2}$$

Your tasks

a) Determine the asymptotic bounds on the operational intensity $I(n)$ for the following matrix/vector operations, where n is the dimension of the vector. State your assumptions.

1. DAXPY: $y = \alpha x + y \quad \alpha \in \mathbb{R}; x, y \in \mathbb{R}^n$
2. SGEMV: $y = Ax + y \quad x, y \in \mathbb{R}^n; A \in \mathbb{R}^{n \times n}$
3. DGEMM: $C = AB + C \quad A, B, C \in \mathbb{R}^{n \times n}$
   Hint 1: Assume that A, B and C fit into the cache at the same time.
   Hint 2: Remember that double precision numbers are typically stored in 8 bytes.

b) Consider the 1D diffusion equation in a periodic domain with length L

$$u_t(x,t) = \alpha u_{xx}(x,t), \tag{3}$$

$$u_0(x) = u(x,0) = \sin\left(\frac{2\pi}{L}x\right), \tag{4}$$

where $0 \leq x < L$, $t > 0$ and $\alpha > 0$ is the diffusion coefficient. You are asked to solve this problem numerically by using a second order centered finite difference scheme and the explicit Euler method to advance in time. The domain is discretized with N uniformly spaced grid points. Discretization of Equation (3) leads to

$$u_i^{n+1} = u_i^n + \frac{\Delta t \alpha}{\Delta x^2}\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right), \tag{5}$$

where $u_i^n \approx u(x_i, t^n)$ is the approximate solution with $t^{n+1} = t^n + \Delta t$ and $x_i = i\Delta x$. The constant time step size and uniform grid spacing are denoted by Δt and Δx, respectively. Determine the operational intensity $I(N)$ for the scheme given in Equation (5).

c) In your position as an HPC expert, you are presented three possible systems (A, B, and C) for purchase, each with unique peak CPU performance and memory bandwidth configurations. Complete the table below indicating, for each system, whether the previously analyzed operations are memory or compute bound. Assume $n = 256$ for all cases. Write down all your assumptions and calculations.

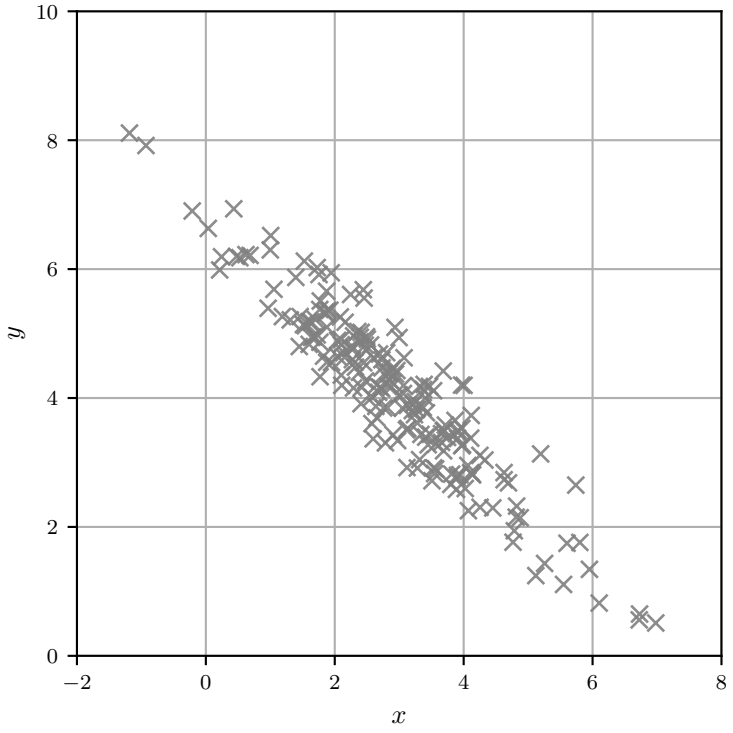
	System A 10 GFLOP/s 70 GB/s	System B 50 GFLOP/s 125 GB/s	System C 70 GFLOP/s 265 GB/s
DAXPY			
SGEMV			
DGEMM			
1D Diffusion			

Question 4: Principal Component Analysis

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

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

with $X \in \mathbb{R}^{N \times D}$, plotted below:



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

b) 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. Do not perform eigendecomposition.

c) 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 (in percentage) do we expect to retain in the reduced order space? Do not project the data.
Hint: Use the previously computed eigenvalues to answer this question.

d) 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 **centered** random variable X. In multiple dimensions, the total variance is summed across dimensions.

e) 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 5: Particles MPI (30 points)

In this task you will implement an N-body solver describing a system of positively charged inertialess particles with MPI parallelization. The provided skeleton code

- seeds N particles on a unit circle

$$\boldsymbol{x}_i = (x_i, y_i) = \big(\cos\frac{2\pi i}{N},\ \sin\frac{2\pi i}{N}\big), \qquad i = 0, \ldots, N-1;$$

  distributing them over P ranks such that each rank takes N/P particles (assume N is divisible by P);
- performs time steps calling functions `Step()` and `PrintStat()`;
- writes the particle positions to files `init.dat` and `final.dat`.

a) Implement `Step()` which computes the forces and advances the particles

$$\boldsymbol{f}_i = \sum_{\substack{j=0 \\ j\neq i}}^{N-1} \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{|\boldsymbol{x}_i - \boldsymbol{x}_j|^3},$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \Delta t\, \boldsymbol{f}_i^t,$$

where $|\boldsymbol{x}| = \sqrt{x^2 + y^2}$ and Δt is a given time step.
Each rank can make $\mathcal{O}(P)$ MPI calls[1] and use $\mathcal{O}(N/P)$ bytes of memory. Your solution will be given more points for overlapping communication and computation. You may use `make` to build the executable, `make run` to run it on 2 processors and `make plot` to create `image.png` with particle positions as shown in Figure 1.

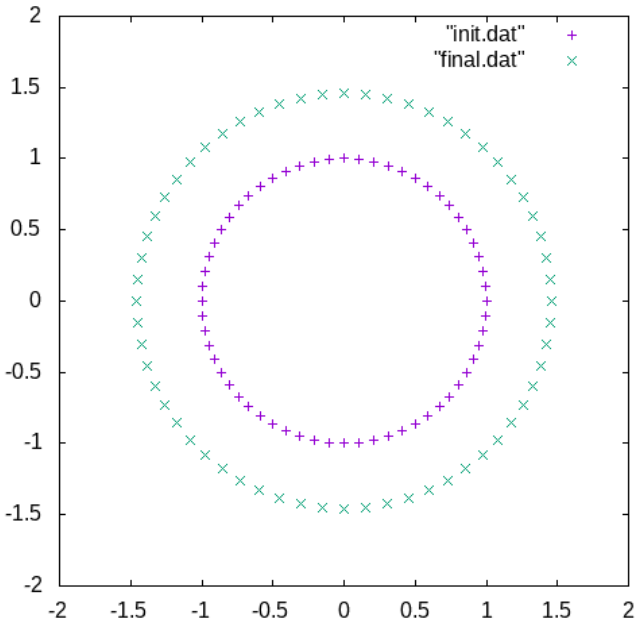


Figure 1: Expected output `image.png` produced by `make plot`.

[1] Notation $p = \mathcal{O}(q)$ means that $p < Mq$ for a large enough constant M.

b) Implement `PrintStat()` which computes and prints the mean and variance of the radial distance $r_i = |\boldsymbol{x}_i|$

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} r_i,$$
$$\sigma^2 = \frac{1}{N}\sum_{i=0}^{N-1} r_i^2 - \mu^2.$$

Expected output after 10 time steps:

```
mean=1          var=0
mean=1.0672     var=6.6613e-16
mean=1.1261     var=6.6613e-16
mean=1.1791     var=0
mean=1.2274     var=-8.8818e-16
mean=1.272      var=8.8818e-16
mean=1.3135     var=-2.2204e-16
mean=1.3524     var=4.4409e-16
mean=1.3891     var=4.4409e-16
mean=1.4239     var=1.3323e-15
mean=1.4571     var=0
```

Rounding errors may lead to differences up to 10^{-3} for the mean and 10^{-14} for the variance.

Question 6: Minimization with random walks (OpenMP) (30 points)

The following algorithm uses a set of random walks to minimize the function

$$f(x) = \int_0^x g(y)\,dy,$$

where $g(y) = y - 2$. The algorithm has parameters $N = 2000$, $T = 20$ and $M_{\max} = 20000$. Initialize N random walks x_i^t, $i = 1, \ldots, N$ on a regular grid in $[0, 1]$

$$x_i^0 = \frac{i - 0.5}{N}.$$

Perform T steps and at each step $t = 1, \ldots, T$,

- Advance the random walks

$$x_i^t = x_i^{t-1} + \frac{1}{N}\xi_i, \quad i = 1, \ldots, N,$$

  where $\xi_i \sim \mathcal{N}(0, 1)$ are samples from the standard normal distribution.

- Compute an approximation of $f(x_i^t)$ using the Monte Carlo estimate

$$f_i^t = \begin{cases} \frac{x_i^t}{M_i} \sum_{m=1}^{M_i} g(Y_m), & x_i^t \geq 0 \\ 0, & x_i^t < 0 \end{cases}, \quad i = 1, \ldots, N,$$

  where $M_i = \lfloor x_i^t M_{\max} \rfloor$ and $Y_m \sim \mathrm{Uniform}(0, x_i^t)$.

- Find the index

$$j = \arg\min_i f_i^t$$

  of a random walk that achieves the minimum value. Store the corresponding $x_{\min}^t = x_j^t$ and $f_{\min}^t = f_j^t$.

The skeleton code `~/omp_random/main.cpp` provides a serial implementation of the algorithm. Write your solution in the same file.

a) Parallelize the function `minimize()` using OpenMP by splitting the random walks among multiple threads (`TODO.a`). Make sure you do not introduce race conditions and that each thread has its own random generator initialized with a unique seed.

b) Compute the total number of evaluations of function `integrand()` for each thread and store in array `evals` (`TODO.b`).

c) Answer the following questions (in your code, `TODO.c`):
   - Run your program with two threads and report the values printed after `"Evaluations per thread (1e6):"`.
   - Is the number of evaluations the same for all threads? Explain in one sentence why.