ETH zürich

High Performance Computing for Science and Engineering for CSE

Spring Semester 2021

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

Mock Exam

Issued: May 31, 2021, 11:15
Hand in: –

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: _____

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 three** A4 sheets (six 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.

Grading table

Question	Maximum score	Score	TA 1	TA 2
Question 1	20			
Question 2	20			
Question 3	20			
Question 4	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: Applied Optimization & UQ (20 points)

Rumors spread that you are among the most notorious programmers and statisticians in the region of Appenzell. The hedge fund *Appenance LTD* approaches you and wants you to have a look at their price prediction model $\mathcal{M}_1$ for cattle trading. They say that their strategy is presumably predicting wrong confidence intervals and hence is heavily under-performing. This particular cattle breed is traded over the counter only during weekends at the local markets, and for that reason trades data $d = \{d_i\}_{i=1}^N$ is very limited and difficult to collect.

Without blinking an eye, you accept their offer and you search for the parameters $\hat{\theta} = (\hat{x_0}, \hat{\sigma}^2)$ that maximize the likelihood of the model, i.e.,

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta; d).$$

For obvious reasons the model can not be disclosed here, but we can tell that x_0 is a parameter of the computational model, and σ^2 describes the variance of the error model.

You use Korali's optimizer (CMA-ES) to find the parameters that maximize the log-likelihood. After 4 consecutive runs you observe the following outputs on the screen:

```
[Korali] CMA-ES Finished
[Korali] Optimum LogLikelihood (Maximize) found: -8.49194e-01
[Korali] Optimum LogLikelihood (Maximize) found at:
         X0 = +1.503e+00
         Sigma2 = +3.501e+00
[Korali] Stopping Criterium: Function value differences < (1.00e-12)
```

Listing 1: Run 1

```
[Korali] CMA-ES Finished
[Korali] Optimum LogLikelihood (Maximize) found: -8.49193e-01
[Korali] Optimum LogLikelihood (Maximize) found at:
         X0 = +3.004e+00
         Sigma2 = +0.998e+00
[Korali] Stopping Criterium: Function value differences < (1.00e-12)
```

Listing 2: Run 2

```
[Korali] CMA-ES Finished
[Korali] Optimum LogLikelihood (Maximize) found: -8.49192e-01
[Korali] Optimum LogLikelihood (Maximize) found at:
         X0 = +3.002e+00
         Sigma2 = +0.999e+00
[Korali] Stopping Criterium: Function value differences < (1.00e-12)
```

Listing 3: Run 3

```
[Korali] CMA-ES Finished
[Korali] Optimum LogLikelihood (Maximize) found: -8.49193e-01
[Korali] Optimum LogLikelihood (Maximize) found at:
          X0 = +1.503e+00
          Sigma2 = +3.500e+00
[Korali] Stopping Criterium: Function value differences < (1.00e-12)
```

Listing 4: Run 4

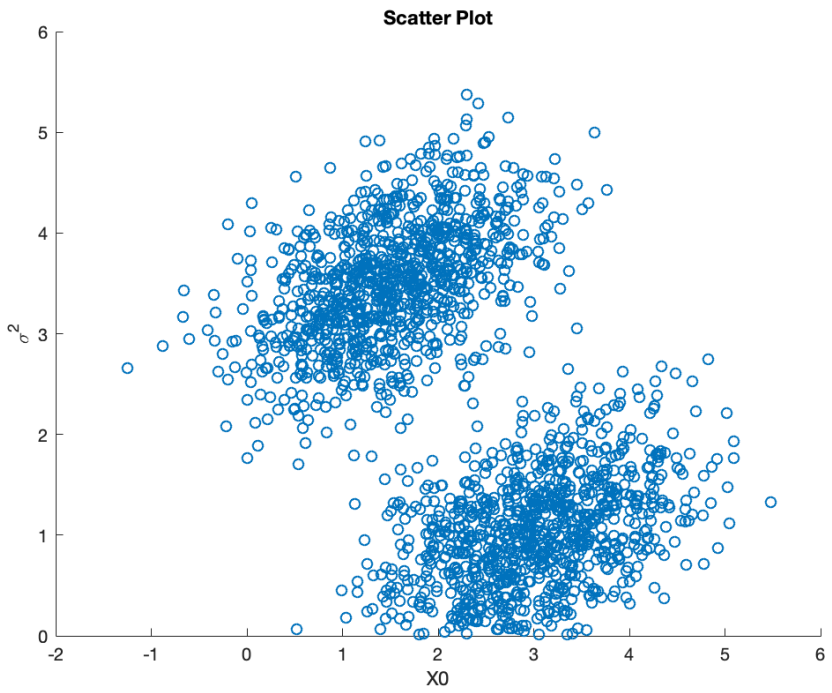
a) (8 points) You observe that CMA-ES converges to different optima although the parameters of CMA-ES are kept the same in each run (except for the seed). Answer the following questions:

- Provide an explanation of the fact CMA-ES returns different results given the same objective function and configuration.
- How many modes does the likelihood and the posterior distribution have, based on these 4 runs? State any necessary assumptions.
- Explain why the price prediction model $\mathcal{M}_1$ is predicting wrong confidence intervals, and propose a solution.

b) (6 points) In order to further analyze the model at hand, you want to sample the posterior distribution of the parameters $\theta = (x_0, \sigma^2)$ given the data d,

$$p(\theta|d) = \frac{p(d|\theta)p(\theta)}{p(d)}\,.$$

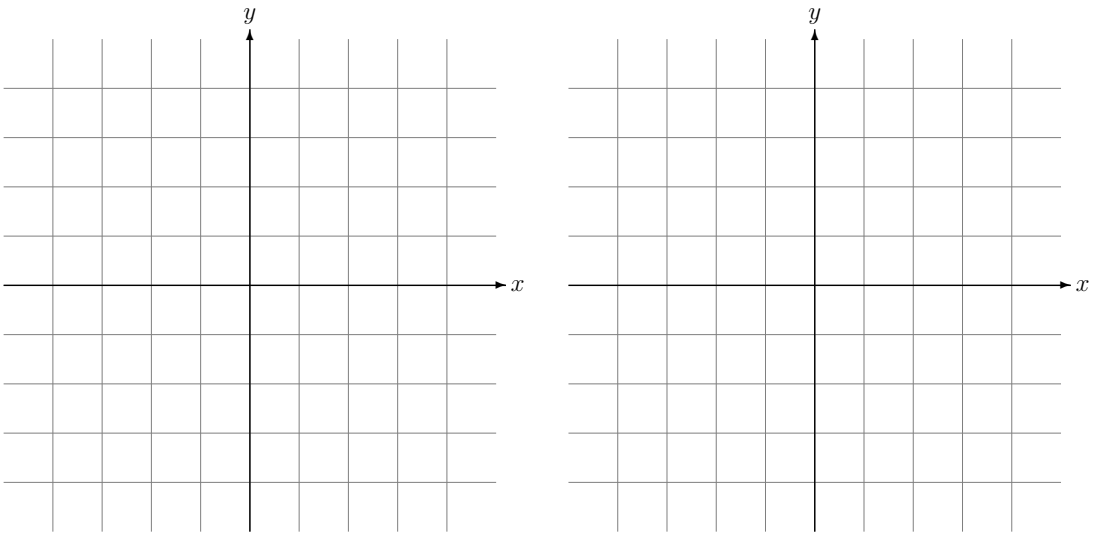
The sampling algorithm returns the following sample locations for the posterior distribution.



Provide 2 qualitative sketches:

- Sketch the marginal distribution $p(\sigma^2|d) = \int_{-\infty}^{+\infty} p(\theta|d)dx_0$.
- Sketch the marginal distribution $p(x_0|d) = \int_{-\infty}^{+\infty} p(\theta|d)d\sigma^2$.

Note: Don't forget to label the axes, and add ticks to the axes where possible (vertically and horizontally) in order to localize your graphs. If you make assumptions, please document it. Please make sure that your plots are consistent throughout.



c) (6 points) In the meantime, the research department of *Appenance LTD* came up with a more sophisticated pricing model $\mathcal{M}_2$, taking into account the PH-value of the grass the cows eat and the amount of days they graze in rain. The researchers find that the maximum likelihood for the new model $\mathcal{M}_2$ is larger than the maximum likelihood for $\mathcal{M}_1$ (based on the same data set d). Answer the following questions in 3 or less sentences each:

- How can you explain that model $\mathcal{M}_2$ achieves a higher likelihood.
- What other measures do you know from this course that allow you to select between $\mathcal{M}_1$ and $\mathcal{M}_2$?
- Explain why comparing the likelihoods of $\mathcal{M}_1$ and $\mathcal{M}_2$ is not the best practice, and why you would resort to other measures.

Question 2: Conjugate Prior and Laplace Approximation (20 points)

The Poisson probability distribution function is used to model the **number** of events $X \in \mathbb{N}_0$ that occur within a given time interval, assuming that the occurrence of an event is independent of the time since the last one. The probability density function of the Poisson distribution parametrized by the shape parameter $\lambda > 0$ is given by

$$P(X = x) = p_\lambda(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

a) (5 points) You are given N observations $\mathbb{X} = (X_1, \ldots, X_N)$. Assume that the observations are independent and identically distributed $X_i \sim p_\lambda(x)$. Show that the parameter λ indicates the expected average number of events, i.e. that the maximum likelihood estimate (MLE) $\hat{\lambda}$ of the parameter λ is given by the sample mean $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$. Start by writing down the likelihood function $\mathcal{L}(\lambda|\mathbb{X})$.

b) (10 points) A prior is called *conjugate prior with respect to a likelihood function* if the prior and the posterior belong to the same probability distribution family. Show that if the prior of λ follows a Gamma distribution, then the posterior distribution $p(\lambda|\mathbb{X})$ is a Gamma distribution too. In other words, the Gamma distribution is a conjugate prior for the Poisson probability distribution. Also specify the parameter of the resulting Gamma distribution.

c) (5 points) Approximate the posterior distribution $p(\lambda|\mathbb{X})$ of the previous task using the Laplace approximation method. You can omit time-consuming algebraic manipulations that involve second derivatives.

- The density function of the Gamma distribution with parameters α and β is:

$$f(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}\,, \quad \alpha, \beta > 0\,,$$

- Where Γ is the Gamma function defined as:

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} \mathrm{d}x\,,$$

Question 3: MPI Load-Balancing (20 points)

Assume a task that is executed in parallel by M^2 different MPI processes that are arranged in a two-dimensional $M \times M$ grid. The task is executed in several iterations. At a given iteration n, a workload of $w_{i,j}^n$ is assigned to each process (i, j). We assume that this changes with time, according to

$$w_{i,j}^{n+1} = w_{i,j}^n + S_{i,j}^n \tag{1}$$

where $S_{i,j}^n$ is a source term that may increase or decrease a process' workload.

Even if all processes start with the same workload, Equation (1) will result in load-imbalance. The severity of this can be quantified by the load-imbalance ratio, defined as

$$r^n = \frac{\max_{i,j} w_{i,j}^n}{\bar{w}^n} \quad , \quad \bar{w}^n = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} w_{i,j}^n \tag{2}$$

In this question, a unit workload $w_{i,j}^n = 1$ corresponds to a single execution of the function `work()` by one process that owns one copy of the following `struct`:

```
1  struct Block
2  {
3      double u[L];
4      int N[2];
5
6      Block() { /* Initialize to some values... */ }
7
8      // Do some flops, to simulate a unit workload w^n_{i,j}=1.
9      void work()
10     {
11         for (int j = 0; j < L; ++j)
12             u[j] = std::pow(u[j] + N[0], 0.1 * N[1]);
13     }
14 };
```

Moreover, the source term $S_{i,j}^n$ is computed as a random function of a process ID (i, j) and it can be positive or negative if work is added or removed from a process. It is computed in the `changeLoad()` function.

We define a load-balancing algorithm as a strategy of redistributing work among processes in such a way that the load-imbalance ratio decreases over time: $r^{n+1} \leq r^n$. In this question, we will study the following diffusion-like algorithm: after iteration n, each process will send/receive work to/from its neighboring processes $(i+1, j)$,$(i-1, j)$,$(i, j+1)$ and $(i, j-1)$ by comparing its workload with theirs.

If $w^n_{i,j} > w^n_{i+1,j}$, process (i,j) will send work equal to $\left[\frac{w^n_{i,j}-w^n_{i+1,j}}{\lambda}\right]$ to process $(i+1,j)$; otherwise, it will receive work from process $(i+1,j)$. Here, $[x]$ is the round towards zero function (e.g. $[\pm 3.9] = \pm 3$) and $\lambda = 8$ a user-defined constant. The same will happen for processes $(i-1,j)$,$(i,j+1)$ and $(i,j-1)$. This results in the following equation

$$w^{n+1}_{i,j} = w^n_{i,j} + \left[\frac{w^n_{i+1,j} - w^n_{i,j}}{\lambda}\right] + \left[\frac{w^n_{i-1,j} - w^n_{i,j}}{\lambda}\right] + \left[\frac{w^n_{i,j+1} - w^n_{i,j}}{\lambda}\right] + \left[\frac{w^n_{i,j-1} - w^n_{i,j}}{\lambda}\right]. \tag{3}$$

a) Your first task is to arrange the processes in a Cartesian $M \times M$ grid. Create a Cartesian communicator and a periodic grid of processes. Also, each process should know the rank IDs of its neighboring processes, as well as its own indices (i,j).

b) To facilitate exchange of workload between processes, create a custom MPI Datatype that allows processes to send/receive `Block` struct instances. Your approach must be portable.

c) Implement Equation (3), in order to make the computation faster by balancing workload among processes. Please refer to the skeleton code for more details.

Question 4: Matrix Multiplication with CUDA (20 points)

Let A and B be $N \times N$ matrices. The matrix multiplication operation $C = AB$ can be implemented in C++ as

```
void matMulCPU(int N, const real *a, const real *b, real *c) {
    for (int iy = 0; iy < N; ++iy)
    for (int ix = 0; ix < N; ++ix) {
        real sum = 0;
        for (int k = 0; k < N; ++k)
            sum += a[iy * N + k] * b[k * N + ix];
        c[iy * N + ix] = sum;
    }
}
```

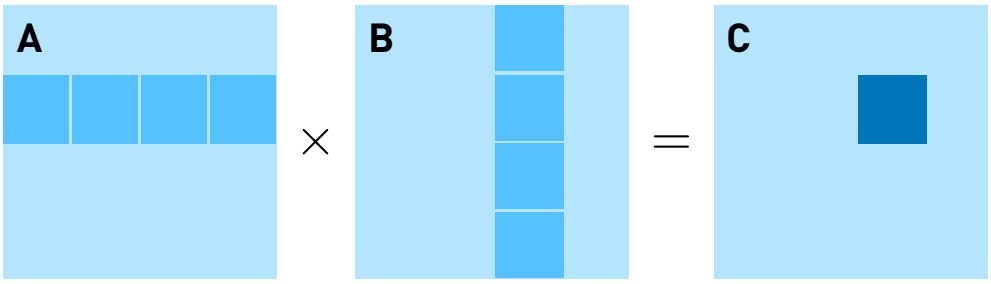
The task is to translate the above function into CUDA and apply specified optimizations. For simplicity, we will assume that N is a multiple of 64.

a) Translate directly the `matMatCPU` function into CUDA such that each thread computes one element of the output matrix `c`. Use blocks with 16×16 threads.

b) The kernel above can be optimized by introducting blocking and utilizing shared memory. Namely, the inner for loop `k` is split into `N / BLOCK_SIZE` stages, each handling `BLOCK_SIZE` elements:

```
for (int k0 = 0; k0 < N; k0 += BLOCK_SIZE)
    for (int k = 0; k < BLOCK_SIZE; ++k)
        sum += a[iy * N + (k + k0)] * b[(k0 + k) * N + ix];
```

Optimize the kernel by storing the necessary elements of `a` and `b` in shared memory and then using those shared arrays for the dot products.



c) Optimize the kernel from subquestion a) by making each thread compute 4 instead of only 1 element of the matrix `c`. Concretely, change the kernel such that the thread `(ix, iy)` computes values `(ix, 4*iy..4*iy+3)`. Why does this change improve the performance, for sufficiently large matrices?

d) Combine the optimization from previous two subquestions.