

Important instructions:

- The exam duration is 2 hours.
 - Please start your answer just below each question (when possible) and continue on a new sheet of paper if necessary. Put your name on all sheets.
 - As mentioned in the course outline, only personal handwritten notes and annotated printouts of the material posted on the course Moodle page are allowed. All the rest, including electronic devices (laptop, smartphone, etc.), is forbidden.
 - Do not use red ink.
 - The size of the answer box is chosen with a quite conservative margin so do not worry if your answer is shorter than the box.
-

1. Consider this C++ code, in which lines 5, 7, 25, and 26 have been hidden:

```
1 #include <iostream>
2 using namespace std;
3
4 class Rectangle {
5
6 public:
7
8     int area(void) { return mWidth * mHeight; }
9 };
10
11 Rectangle::Rectangle(int w, int h)
12 {
13     mWidth = w;
14     mHeight = h;
15 }
16
17 int main() {
18     Rectangle * p_r;
19     Rectangle * a_r;
20     p_r = new Rectangle (1, 2);
21     a_r = new Rectangle[2] { {3,4}, {5,6} };
22     cout << "*p_r's area: " << p_r->area() << '\n';
23     cout << "a_r[0]'s area:" << a_r[0].area() << '\n';
24     cout << "a_r[1]'s area:" << a_r[1].area() << '\n';
25
26
27     return 0;
28 }
```

- Write the four missing lines directly in the code.
- Briefly justify your answers.
- What output does line 24 produce? Briefly explain why.

2. Consider the FTCS (forward time, centered space) scheme for the following diffusion problem:

$$\begin{aligned}u_t(t, x) &= \kappa u_{xx}(t, x), \quad x \in [0, 1], t \geq 0, \\u(t, 0) &= 0 = u(t, 1) \quad \text{for all } t \geq 0, \\u(0, x) &= u^0(x) \quad \text{for all } x \in [0, 1].\end{aligned}$$

- (a) Write the FTCS scheme, including the boundary and initial conditions.
- (b) Briefly give the meaning of each symbol (excluding the basic arithmetic, of course) that you have introduced. You can use a mix of words and equations.
- (c) Explain the notion of *refinement path*.
- (d) Give (without proof) a refinement path along which the FTCS scheme is stable. Explain what this means.
- (e) Give (without proof) a refinement path along which the FTCS scheme is *not* stable. Explain what this means.

3. Consider the time integration of the following diffusion equation:

$$\frac{\partial u(\theta, t)}{\partial t} = \alpha \frac{\partial^2 u(\theta, t)}{\partial \theta^2}.$$

The solution is also assumed to be cyclic, i.e., $u(\theta + 2\pi, t) = u(\theta, t) \forall \theta \in \mathbb{R}$. The solution is discretized over n equally spaced points in θ : $\theta_1 = 0, \theta_2 = 2\pi/n, \theta_3 = 4\pi/n, \dots$

You have written the following OpenCL kernel to integrate the equation over `num_iter` time steps. The constant F is $\alpha\Delta t/(\Delta x)^2$ where $\Delta x = 2\pi/n$.

Simplifying assumption *This kernel assumes that the number of processing elements of a compute unit is at least n so all threads are in the same compute unit. You do not have to consider any situation in which this is not the case.*

```
1 __kernel void _diffuse(__local float* a, float F, int num_iter) {
2     int i = get_local_id(0);
3     int prev, next, iter;
4     if (i == 0)
5         prev = get_local_size(0) - 1;
6     else
7         prev = i - 1;
8     if (i == get_local_size(0) - 1)
9         next = 0;
10    else
11        next = i + 1;
12    for (iter = 0; iter < num_iter; iter++)
13        a[i] = a[i] + F * (a[prev] + a[next] - 2 * a[i]);
14 }
15
16 __kernel void diffuse(__global float* glob, __local float* shared, float F, int
17     num_iter) {
18     int i = get_global_id(0);
19     shared[i] = glob[i];
20     _diffuse(shared, F, num_iter);
21     glob[i] = shared[i];
22 }
```

You observe that your kernel does not always provide the same answer, the answer even seems to be incorrect.

Answer the following questions:

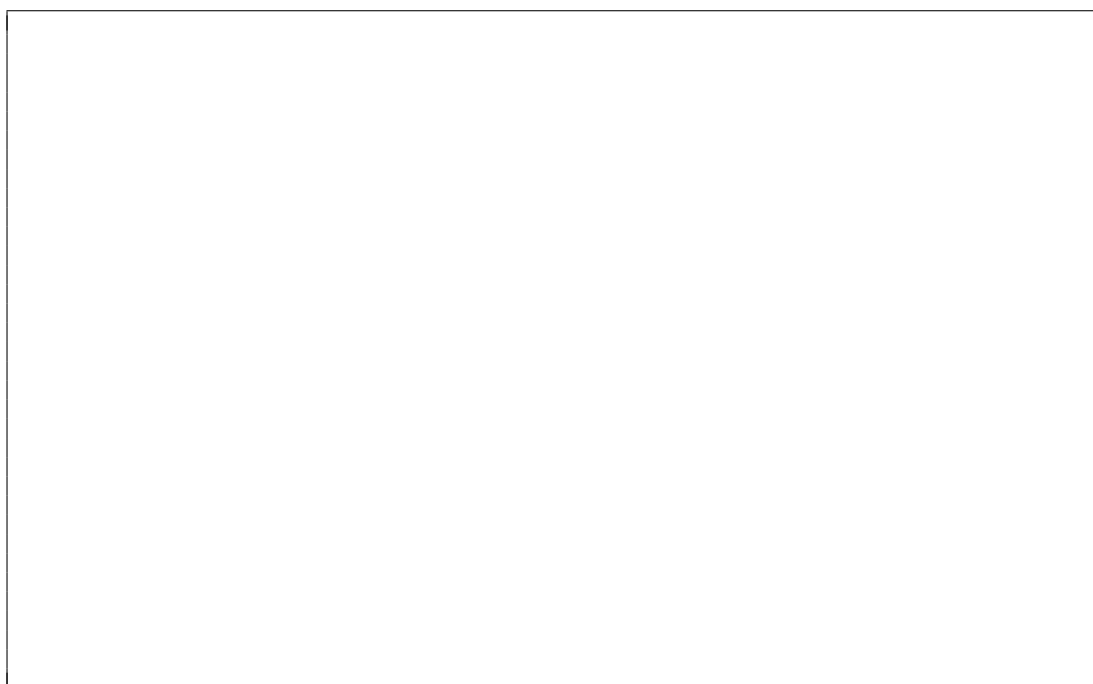
- (a) What is the issue causing the kernel to produce incorrect results ?



(b) You identify that depending on your device, if n is sufficiently small you always get the correct answer. More precisely, you get the correct answer in the following situations:

- $n \leq 8$ and you use your CPU as device for your kernel,
- $n \leq 32$ and you use your GPU as device for your kernel.

Explain why.



(c) How can you fix this issue you identified in a. ?

4. After the success of the kernel you developed in “Question 3”, you now want to target applications with much larger n . For this reason, you now want to solve it with several compute nodes. You have access to m compute nodes and enough material to connect m pairs of compute nodes together. Using these connections, the communication of k bytes takes a time equal to $\alpha + \beta k$.

Answer the following questions:

- (a) How would you interconnect the m compute nodes ?

- (b) How much time would the communication take at each time step of the integration in terms of α, β, m, n ?

- (c) You would like to compute $\sum_{i=1}^n |u(\theta_i, t)|$ to verify whether the solution is diverging. Which MPI collectives would you use ? How much time would this take in terms of α, β, m, n and also γ , the time taken by the compute node to execute one arithmetic operation.

UCLouvain
EPL-INMA
18 juin 2025

Exam : LINMA2710
Scientific Computing

First name :
Last name :
NOMA :

A large, empty rectangular box with a thin black border, occupying the central portion of the page. It is intended for the student to provide their personal details as indicated by the labels in the top right corner.