HPCA (Advanced high performance computing algorithms and programming many core architectures)

Two thirds of this course is dedicated to CUDA programming studied in **Part I**. One third of this course is dedicated to large problems solving with multi-node architecture studied in **Part II**.

This **Part I** introduces GPU programming using CUDA API and then provides a large number of examples that will make the reader more and more comfortable with the hardware and the API used. The purpose is to give the technical background that opens scaling up applications to a large public of students, researchers, and engineers.

Part I starts with an Introduction and CUDA basics, in chapters 1, 2, and 3 then linear algebra examples in Chapter 4 and it ends with deep learning examples presented in Chapter 5.

1. Introduction: high Performance Computing till GPUs and what comes after
   a. High Performance Computing before GPUs
   b. Disruption due to GPUs
   c. Actual and future possible evolutions
2. First steps in parallel programming with CUDA/C
   a. How to install/use on local machine and how to use on Google cloud
   b. Device Query and Hello World
   c. Streaming Processors and their interpretation in terms of blocks and threads
   d. Using timers and CPU/GPU data transfer
3. Advanced steps in parallel programming with CUDA/C
   a. Memory architecture on the device: real cache
   b. Memory architecture on the device: virtual cache
   c. Memory allocation on the host
   d. Concurrency and asynchronous data transfer
4. Batch computing and advanced algorithms
   a. Batch matrix multiplication
   b. Batch LDLt factorization
   c. Batch Parallel Cyclic Reduction (PCR)
   d. Batch Merge Sort and Merge Sort
5. PyTorch and Numba used in Deep Learning (DL)
   a. Brief introduction to Python/Numba
   b. Linear regression with Monte Carlo simulated data
   c. Neural Network regression with Monte Carlo simulated data

Modern computers have an increasing number of computing units. To take advantage of this parallelism, it is therefore necessary to use and implement algorithms with a high level of concurrency.
In this **Part II**, different techniques for solving hollow linear systems that benefit from such a feature are presented. In particular, domain decomposition methods and multigrid methods that have an optimal asymptotic cost are detailed.

1. Sparse matrices
   a. Efficient data structures
   b. Matrix-vector products and other algebraic operations
   c. Parallelization on shared- and distributed-memory architecture

2. Iterative methods
    a. Fixed-point methods
    b. Polynomial methods
    c. Krylov methods
3. Basic preconditioning
    a. Parallelization on shared- and distributed-memory architectures
    b. Renumbering and coloring
    c. Use in Krylov methods
4. Domain decomposition methods
    a. Overlapping Schwarz methods
    b. Substructuring methods
    c. Coarse grid operators
5. Multigrid methods
    a. Geometric multigrid
    b. Algebraic multigrid
    c. Link with domain decomposition methods

Grades will be computed as follows: project (40%) + exam (60%).