Course guide
250COD006 - 250COD006 - Scientific Programming and High Performance Computing

Last modified: 16/06/2026

Unit in charge: Barcelona School of Civil Engineering
Teaching unit: 751 - DECA - Department of Civil and Environmental Engineering.

Degree: MASTER'S DEGREE IN COMPUTATIONAL AND DATA-ASSISTED ENGINEERING (Syllabus 2026). (Compulsory subject).

Academic year: 2026 **ECTS Credits:** 5.0 **Languages:** English

LECTURER

Coordinating lecturer: Zorrilla Martínez, Rubén

Others: Zorrilla Martínez, Rubén
Rossi Bernecoli, Riccardo

TEACHING METHODOLOGY

The course follows an interactive, practice-oriented approach in which theoretical concepts are tightly coupled with implementation. Hence, each session combines:
- Conceptual explanations of scientific programming and high-performance computing principles.
- Live coding demonstrations, where key ideas are developed step by step, showing how theoretical concepts translate into efficient computational implementations.
- Hands-on exercises, carried out by students during the session, allowing immediate application of the concepts using Python.

A significant part of the course is delivered through interactive notebooks (e.g., Google Colab), enabling experimentation, reproducibility, and rapid iteration without complex setup. This approach facilitates active learning and allows students to explore variations of the methods in real time.

Throughout the course, students work on practical tasks and a progressive project. These activities emphasize performance analysis, scalability, and integration of numerical and data-driven components.

Independent work complements in-class activities, focusing on completing assignments, improving implementations, and analyzing computational performance, fostering autonomy and deeper understanding.

LEARNING OBJECTIVES OF THE SUBJECT

This course introduces the principles and practices required to develop efficient, scalable, and maintainable scientific software for computational and data-assisted engineering applications. It focuses on the implementation of numerical methods using modern programming paradigms and high-performance computing techniques within a Python-based ecosystem.

The course covers the full workflow of computational engineering, from software design and numerical robustness to performance optimization and parallel execution. Core topics include high-performance dense and sparse linear algebra and performance engineering concepts such as memory hierarchy, vectorization, and profiling. These are extended to parallel computing paradigms, including shared-memory parallelism, distributed-memory programming with MPI, and GPU acceleration.

In addition, the course addresses modern challenges in data-assisted engineering, such as handling large-scale datasets using out-of-core and distributed approaches, and integrating simulation workflows with data processing pipelines.

Through a combination of theoretical sessions and hands-on tasks, students develop the ability to design, implement, analyze, and optimize computational solutions for large-scale engineering problems.

STUDY LOAD

Type	Hours	Percentage
Hours large group	45,0	36.00
Self study	80,0	64.00

Total learning time: 125 h

CONTENTS

Programming paradigms and introduction to Python

Description:
- Procedural vs Object-Oriented Programming (OOP)
- Functional programming concepts
- OOP in large simulation codes (encapsulation, abstraction)
- Functional patterns in numerical computing (map, reduce, immutability)
- Compiled vs interpreted languages
- Static vs dynamic typing
- Introduction to Python syntax and virtual environments

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Algorithmic complexity & data structures

Description:
- Algorithmic complexity & data structures
- Modular design in Python (function, classes, types)
- Visualization with Python (matplotlib and PyVista)

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Professional code development

Description:
- Version control
- Professional development: code quality and code revision
- Testing strategies (continuous integration) and test-driven development
- AI-supported programming environments (IDEs)

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Numerical precision & stability in large-scale simulation

Description:
- Floating-point arithmetic
- Conditioning vs algorithmic stability
- Algorithmic complexity in simulation workflows
- Memory hierarchy and cache behavior
- Roofline model
- Vectorization and memory layout
- Identifying bottlenecks in FE-like loops

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

High-performance linear algebra

Description:
- BLAS/LAPACK concepts behind NumPy
- Efficient dense operations and performance benchmarking
- Sparse matrix formats (CSR, COO)
- Efficient sparse assembly
- Memory footprint analysis
- Brief overview of Krylov solvers and its implementation
- Preconditioning strategies
- Iterative solvers: convergence vs scalability
- Iterative solvers: numerical robustness in iterative solvers

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Introduction to parallelism

Description:
- AMDAHL's law
- Strong vs weak scaling
- CPU parallelism and memory layout: shared memory vs distributed memory parallelism
- GPU parallelism

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Shared memory parallelism

Description:
- Shared memory parallelism in Python (GIL and multiprocessing, asyncio fundamentals, task orchestration)
- SMP Task

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Distributed memory parallelism with MPI

Description:
- Distributed data structures
- Communication patterns
- Distributed matrix-vector products

Full-or-part-time: 15h
Theory classes: 6h
Self study : 9h

GPU parallelism

Description:
- GPU acceleration fundamentals with PyTorch
- GPU Task

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Parallelism wrap-up

Description:
- Hands-on session to work in the DMP, MPI and GPU projects

Full-or-part-time: 7h 30m
Guided activities: 3h
Self study : 4h 30m

Efficient debugging and performance analysis

Description:
- "Corrupted" life cycle hands-on experience
- Debugging tools
- Reduction bottlenecks
- Hands-on experience with profiling tools

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

Handling of extremely large datasets

Description:
- Out-of-core arrays
- Chunking strategies
- Task graphs and lazy evaluation
- Distributed Dask scheduler
- MPI vs Dask paradigms
- Efficient scientific I/O (HDF5)

Full-or-part-time: 7h 30m
Theory classes: 3h
Self study : 4h 30m

GRADING SYSTEM

The evaluation of the course is based on continuous assessment and a final project:
- Continuous assessment (40%): evaluation through practical assignments and hands-on tasks. Assessment criteria include correctness, code quality, and basic performance considerations.
- Final project (45%): development of a computational project integrating course concepts, such as numerical methods, parallel computing, and performance optimization. The deliverables of the project include the source code (version-controlled) as well as a technical report including performance analysis and demonstration of results. Evaluation criteria include technical correctness, scalability, performance analysis, and software design quality.
- Oral defense (15%): individual oral assessment based on the final project, evaluating the understanding of implemented methods, the design and parallelization decisions and the ability to analyze performance trade-offs.

BIBLIOGRAPHY

Basic:
- Dongarra, Jack ... [et al.]. Sourcebook of Parallel Computing. Amsterdam [etc.]: Morgan Kaufmann, 2003. ISBN 1558608710.
- Eijkhout, Victor. Introduction to High-Performance Scientific Computing [on line]. 3rd ed. 2022 [Consultation: 25/06/2026]. Available on: https://github.com/VictorEijkhout/TheArtofHPC_pdfs/blob/main/vol1/EijkhoutIntroToHPC.pdf.
- Scott, L.R.; Clark, T.; Bagheri, B. Scientific Parallel Computing. Princeton, New Jersey ; Oxford: Princeton University Press, 2005. ISBN 069111935X.
- Dongarra, Jack ... [et al.]. Numerical Linear Algebra on High-Performance Computers. Philadelphia: SIAM, 1998. ISBN 0898714281.