Introduction to Scientific Computing

Martino Andrea Scarpolini

December 2025

Dates

April 8-21 2026.

Course Overview

This short series of lectures introduces the fundamental concepts and practices of scientific computing, with a particular focus on the numerical solution of ordinary and partial differential equations (ODEs and PDEs). Students will explore the mathematical and computational tools used to model and simulate physical, biological, and engineering systems. Throughout the course, we will employ numerical calculus to solve problems ranging from simple ODEs to classical PDEs arising in continuum mechanics and related fields.

A central objective of the course is to develop conceptual and practical understanding in contemporary scientific programming. We will examine standard compiled languages (C and Fortran), modern high-level languages (Python), and just-in-time compiled environments (Julia), highlighting their different roles in modern High Performance Computing (HPC). The emphasis will be placed on writing clear, efficient, and reproducible scientific code using modern programming paradigms.

As a last topic, since scientific computing is inherently collaborative, the course will explore essential software-engineering practices. Version control is one of the most important skills, and the main tool in this area is Git and its use for effective collaborative code development. Students will gain hands-on experience working on shared code repositories.

Because HPC systems are typically accessed through text-based interfaces, the course begins with a brief introduction to Unix-like operating systems and command-line shells (bash/zsh). Students are expected to have access to a Unix-like environment (e.g. Linux, macOS or Windows Subsystem Linux WSL) with standard compilation tools installed.

Schedule of Lessons

- **Lesson 1** [Wednesday 08/04/2026]: Unix-like operating systems and the Bash shell: navigation, scripting, and workflow automation.
    1. Overview of OS history
    2. PC architecture
    3. Linux kernel
    4. Processes and memory
    5. Using the shell: Bash
- **Lesson 2** [Friday 10/04/2026]: Introduction to scientific programming languages: Python, C/Fortran, and Julia. Interpreted vs. compiled vs. JIT-compiled paradigms.
    1. Overview of Programming Languages
    2. Compiled languages: C/Fortran
    3. Interpreted languages: Python
    4. JIT-compiled languages: Julia
- **Lesson 3** [Tuesday 14/04/2026]: Numerical methods for ODEs.
    1. Integrals: implementation of the Trapezoidal Rule
    2. Implementation of a simple ODE solver

- **Lesson 4** [Friday 17/04/2026]: Introduction to Git and setup of previous projects repositories.
    1. Three main git areas: modified, staged, committed
    2. git log
    3. Undoing Things: git show, git restore, git reset
    4. Branching and rebasing
- **Lesson 5** [Wed 21/04/2026]: Collaborative development with Git.
    1. Contribution to a class-shared repository