GHENT UNIVERSITY

# Course Specifications

Valid as from the academic year 2026-2027

## Scientific Computing for Physicists (C004649)

**Course size** *(nominal values; actual values may depend on programme)*

**Credits 5.0** **Study time 150 h**

**Course offerings in academic year 2026-2027**

| | | |
|---|---|---|
| A (semester 1) | English | Gent |

**Lecturers in academic year 2026-2027**

| | | |
|---|---|---|
| Leliaert, Jonathan | WE04 | lecturer-in-charge |
| Verstraelen, Toon | WE05 | co-lecturer |

| **Offered in the following programmes in 2026-2027** | **crdts** | **offering** |
|---|---|---|
| Bachelor of Science in Physics and Astronomy | 5 | A |
| Master of Science in Biomedical Sciences | 5 | A |
| Exchange programme Faculty of Sciences (bachelor's level) | 5 | A |
| Preparatory Course Master of Science in Physics and Astronomy | 5 | A |
| Preparatory Course Master of Science in Physics and Astronomy | 5 | A |

**Teaching languages**

English

**Keywords**

Computational Physics and Astronomy, Numerical Methods.

**Position of the course**

This course unit belongs to the learning pathway "Computer skills" in the Bachelor program Physics and Astronomy.
Computational methods are the "bicycles of the mind": computational algorithms offer new opportunities in experimental setups or in simulations of theoretical models, all of which would be unthinkable with simple manual calculation, a calculator or a spreadsheet. Therefore, creative application of programming skills and software libraries form an important set of skills for students in the Physics and Astronomy program. These skills will prove helpful in the remainder of the study program and they are also strongly appreciated assets on the job market.
This course will provide a basic theoretical understanding of numerical algorithms and their relevance to physics and astronomy. Hands-on sessions and projects during the term will complement the theory with hands-on skills to tackle various problems in the domain with computational methods. This course intends to bridge the gap between the basic course on programming (1st bachelor) and the course Computational Physics (master).

**Contents**

Theory:

1 Numerical limitations: precision (representation of floating point numbers), cpu speed (timing of algorithms) and memory (profiling).
2 Basic numerical building blocks in Python: lists versus arrays, mathematical array operations (element-wise and contractions), other operations (sorting, slicing, ...).
3 Basic visualization with 2D graphics: curves, bars, scatter plots, legends and axes, histograms, contour plots, heatmaps.
4 Professional programming practices: source code revisions, quality assurance
5 Numerical linear algebra: decompositions (QR, LU, Eigen, SVD) and solving linear problems.
6 Interpolation and data modeling: multidimensional linear regression, cubic

splines, chebyshev Interpolation and regularization, kriging and kernel ridge regression.
7 Numerical optimization: recap of terminology (stationary points, minima, saddle points, maxima), 1D and multidimensional, convexity, gradient algorithms (SD, CG, QN), gradient-free algorithms.
8 Finite differences. Application to solving ODEs.
9 Fast-fourier transform.
10 (Markov-Chain) Monte Carlo sampling, importance of priors.
11 Advanced numerical (efficiency) topics: vectorization, computational graphs, just-in-time compilation, ...

Project topics will be updated on a yearly basis. Each project focuses on one or two of items 5 to 10 in the list above, addressing a physics-oriented hands-on problem, also making use of the introductory points 1, 2 and 3.

**Initial competences**

1 Basic knowledge of Python programming: built-in datatypes, flow control, functions, classes, importing modules from the standard library.
2 Basic physics and mathematics: classical mechanics, electromagnetism, calculus, linear algebra

**Final competences**

1 Understand numerical limitations: precision and available resources (cpu time, memory, ...).
2 Understand working principles, strengths and limitations of computer algorithms used in computational physics and astronomy. (See course contents for more details.)
3 Tackle basic computational problems by programming in Python, mainly using NumPy and SciPy.
4 Know available Scientific programming libraries and find required functions or classes in their documentation.
5 Process input data coming from various sources (CSV, TXT, HDF5, NPZ) and output raw results in the same formats.
6 Create insightful visualizations of input data or computational results (2D plotting).
7 Installation of a Python development environment on any operating system of interest.

**Conditions for credit contract**

Access to this course unit via a credit contract is determined after successful competences assessment

**Conditions for exam contract**

This course unit cannot be taken via an exam contract

**Teaching methods**

Seminar, Lecture, Practical

**Extra information on the teaching methods**

Guided exercises in small groups

**Study material**

Type: Handbook

Name: Scientific Computing - An introductory survey (revised second edition)
Indicative price: € 120
Optional: yes
Language : English
Author : Michael T. Heath
ISBN : 978-1-61197-558-1
Number of Pages : 587
Available in the Library : Yes

Type: Software

Name: Jupyter Notebooks with lecture materials and worked out examples
Indicative price: Free or paid by faculty
Optional: no
Available on Athena : No
Online Available : Yes

**References**

- Strongly recommended: Scientific Computing, An Introductory Survey, Revised Second Edition – Michael T. Heath
- Other useful books: Numerical Recipes in Python

**Course content-related study coaching**

Website (Ufora) with additional material and references available. Ample opportunity for asking questions, both personally and by electronic mail.

**Assessment moments**

end-of-term and continuous assessment

**Examination methods in case of periodic assessment during the first examination period**

Oral assessment, Written assessment with open-ended questions, Written assessment open-book

**Examination methods in case of periodic assessment during the second examination period**

Oral assessment, Written assessment with open-ended questions, Written assessment open-book

**Examination methods in case of permanent assessment**

Assignment

**Possibilities of retake in case of permanent assessment**

examination during the second examination period is not possible

**Extra information on the examination methods**

- Closed-book theory exam (end-of-term evaluation): oral exam with written preparation. This will evaluate mainly points 1 and 2 of the final competences. Insight into these topics is required, whereas reproduction of theoretical is less critical.
- Open-book exercise exam (end-of-term evaluation): Computer-exercises in which the theory needs to be applied to concrete problems, which can be solved using the teaching materials and documentation of the software libraries used throughout the course. Additionally, insight in the exercises is evaluated through closed book written open questions.
- Permanent evaluation on active participation in the computational projects/homeworks during term, showing creative input and constructive peer assessment, applying good practices.

**Calculation of the examination mark**

- 50% theory (end-of-term evaluation)
- 35% Exercise exam (end-of-term evaluation)
- 15% Permanent evaluation on active participation in the computational projects during term, showing creative input and constructive peer assessment, applying good practices.
- Students who eschew period-aligned and/or non-period-aligned evaluation may be failed by the examiner.