COURSE UNIT (MODULE) DESCRIPTION

Course unit (module) title	Code
Numerical Methods II	

Annotation

Lecturer(s)	Department(s) where the course unit (module) is delivered
Coordinator: **Markus Ambrosch** **Others:** **Barhka Bale** **Dr. Carlos Viscasillas Vázquez**	Faculty of Physics

Study cycle	Type of the course unit (module)
Bachelor	General university studies

Mode of delivery	Semester or period when the course unit (module) is delivered	Language(s) of instruction
Auditory teaching and practical teaching in computer rooms	Autumn Semester	English

Requisites	
Co-requisites (if relevant): Knowledge of the basic principles of python programming (conditional statements, loops, lists and arrays, functions, simple data visualization). For example, from the Numerical Methods I course.	**Additional requirements (if any):**

Number of ECTS credits allocated	Student's workload (total)	Contact hours	Individual work
5	128	64	64

Purpose of the course unit (module)
This course enables students to solve complex practical problems using Python coding. Students will be able to numerically simulate physical processes with Python code and to interpret the results. Students will gain experience with the theoretical concepts of the most common numerical optimization methods and novel machine learning techniques.

Learning outcomes of the course unit (module)	Teaching and learning methods	Assessment methods
Understanding of the theoretical concepts behind numerical problem-solving methods in mathematics, physics, and data analysis.	Lectures	Midterms and final exams
Ability to implement numerical problem-solving methods in Python code for mathematical and physical applications.	Seminars, homework	Attendance in seminars

Ability to assemble complex Python code from smaller blocks of simple code. Re-purposing existing code to solve a given numerical problem.	Seminars, homework	Attendance in seminars and presentation of code from homework

Content: breakdown of the topics	Contact hours						Individual work: time and assignments		
	Lectures	Tutorials	Seminars	Workshops	Laboratory work	Internship/work placement	**Contact hours, total**	**Individual work**	**Assignments**
1. **Review** of basic principles of programming in Python. Conditional statements, if/else, for-loops, functions. Numerical differentiation and integration in Python. Usage of Python packages. Reading/writing data from/to files. Visualization of data.	4		4				**8**	**8**	
2. **Monte-Carlo methods** Principle of Monte-Carlo methods, application of Monte-Carlo methods for numerical solving of mathematical problems (calculation of Pi, function integration), and for the simulation of physical experiments.	2		2				**4**	**4**	
3. **Optimization methods** Golden section search, Parabolic Interpolation, and Brent algorithm. Gradient Descent optimization.	2		2				**4**	**4**	
4. **Fourier transform** Principles of discreet and fast Fourier transform. Applying Fourier transforms to real-life data to understand physical processes. Signal filtering.	2		2				**4**	**4**	
5. **Numerical approximation of functions with Taylor series** Expressing functions with Taylor series expansion. Discussion of numerical truncation errors for Taylor series.	2		2				**4**	**4**	
6. **Complex numbers in Python** Creating complex numbers in Python, Mathematical operations with complex numbers in Python, Different	2		2				**4**	**4**	

representations of complex numbers.									
7. **Finite-Difference Time-Domain method**. in one and two dimensions. Application	4		4				**8**	**8**	
8. **Machine learning methods** Types of machine learning methods. Practical application of unsupervised and supervised machine learning methods.	4		4				**8**	**8**	
9. **Differential equations** Solving ordinary and coupled Differential equations in Python. Partial Differential equations in Python.	2		2				**4**	**4**	
10. **Nelder-Mead method** Basic principles of finding extrema in a multi-dimensional objective function with the Nelder-Mead method. Implementing the method for finding the extrema of two-dimensional functions in Python.	4		4				**8**	**8**	
11. **Graphical user interfaces** Creation of simple graphical user interfaces using Python libraries.	2		2				**4**	**4**	
12. **Parallel programming** Difference between serial and parallel processes. Practical applications of parallel programming. Disadvantages of parallel programming	2		2				**4**	**4**	
Total	**32**		**32**				**64**	**64**	

Assessment strategy	Weight, %	Deadline	Assessment criteria
Midterm exam	25		Theoretical knowledge about numerical methods and practical work on code (finding and correcting errors in a given piece of code, completing a given piece of code) are assessed.
Final exam	25		Theoretical knowledge about numerical methods and practical work on code (finding and correcting errors in a given piece of code, completing a given piece of code) are assessed.
Attendance in seminars	25		Physical attendance in the seminar sessions is obligatory. Missing seminar sessions will have a negative influence on the final grade.
Participation during seminars	25		Active work on code during the seminar sessions. Review of last week's code at the beginning of a seminar session.

Author	Publishing year	Title	Issue of a periodical or volume of a publication; pages	Publishing house or internet site
Required reading				
Qingkai Kong, Timmy Siauw, Alexandre Bayen	2020	Python Programming and Numerical Methods - A Guide for Engineers and Scientists		Elsevier, https://pythonnumericalmethods.berkeley.edu
Amin Zollanvari	2023	Machine Learning with Python - Theory and Implementation	Chapters 3, 13, and 14	Springer, https://link.springer.com/book/10.1007/978-3-031-33342-2
Recommended reading				
Alex Gezerlis	2020	Numerical Methods in Physics with Python		Cambridge University Press
James R. Parker	2021	Python: An Introduction to Programming		Mercury Learning & Information