COURSE OUTLINE

(1) GENERAL

SCHOOL	Engineering		
ACADEMIC UNIT	Department of Computer Engineering and Informatics		
LEVEL OF STUDIES	Undergraduate		
COURSE CODE	**CEID_NY343**	**SEMESTER**	**FALL**
COURSE TITLE	Scientific Computing		

INDEPENDENT TEACHING ACTIVITIES *if credits are awarded for separate components of the course, e.g. lectures, laboratory exercises, etc. If the credits are awarded for the whole of the course, give the weekly teaching hours and the total credits*	**WEEKLY TEACHING HOURS**	**CREDITS**
Lectures, Tutorials, Laboratory	3 (L), 1 (T), 1 (L)	5
Add rows if necessary. The organisation of teaching and the teaching methods used are described in detail at (d).	TOTAL	5

COURSE TYPE *general background, special background, specialised general knowledge, skills development*	Special background, skills development
Q	Recommended prerequisite knowledge: Good familiarity with the courses “Linear Algebra” (NY110), “Numerical Analysis and Implementation Environments” (NY240), “Introduction to Algorithms” (NY205), “Basic Aspects of Computer Architecture” (NY261), “Parallel Processing” (NY408) or equivalents.
LANGUAGE OF INSTRUCTION and EXAMINATIONS:	Greek (English if there are Erasmus students)
IS THE COURSE OFFERED TO ERASMUS STUDENTS	Yes
COURSE WEBSITE (URL)	https://eclass.upatras.gr/courses/CEID1151/

(2) LEARNING OUTCOMES

Learning outcomes

The course learning outcomes, specific knowledge, skills and competences of an appropriate level, which the students will acquire with the successful completion of the course are described.

Consult Appendix A

- *Description of the level of learning outcomes for each qualifications cycle, according to the Qualifications Framework of the European Higher Education Area*
- *Descriptors for Levels 6, 7 & 8 of the European Qualifications Framework for Lifelong Learning and Appendix B*
- *Guidelines for writing Learning Outcomes*

The course aims to provide students with principles of design, development and implementation of algorithms, methods and software tools for the numerical solution of problems based on high performance simulations and large scale data analysis in science and engineering. Using matrix computations as a vehicle, the course provides students with a thorough understanding of the computation-communication tradeoffs, the impact of hierarchical memory, the tradeoffs between the effects of the numerical errors (due to roundoff and its propagation and discretization) and performance, elements of the design of sparse and data sparse matrix computations, and presents approaches for the evaluation of performance and errors in computations.

Upon conclusion of the course the students ought to:

1. Be familiar with the different types of models involved in Scientific Computjng.
2. Understand the computation-communication tradeoffs in algorithm design and the impact of hierarchical memory.
3. Understand basic compiler techniques towards HPC.
4. Understand the principles of error analysis and the connection between backward error analysis, forward error and conditioning.
5. To understand the implications of the fused multiply and add instruction and methods for improving

the error in summation and multiplication.

6. To be familiar with superfast algorithms for matrix multiplication and the implications of these algorithms for the theory and the practice.
7. Be familiar with the design of dense matrix operations from matrix multiply to LU and QR factorizations based on the BLAS as well as with the principles behind the design of the LAPACK numerical library and its use in MATLAB.
8. Understand the principles of sparse matrix storage and simple sparse matrix computations.
9. Be familiar with the basic principles behind projection methods for solving linear systems.
10. Be familiar with a range of problem solving environments (MATLAB, Octave, Julia).
11. Be familiar with the basic categories of methods for discretizing differential equations and the characteristics of the resulting systems of equations.
12. To know selected algorithms of numerical linear algebra used in data and graph analytics.

Upon conclusion of the course the students are expected to have the following skills:

1. Understand how to measure the performance of numerical codes and important factors affecting their performance.
2. Understand how to increase the efficiency of numerical algorithms and pertinent software.
3. To know various techniques for implementing matrix operations based on the BLAS hierarchy.
4. To conduct forward and backward error analysis of simple numerical computations and to characterize the backward stability of the basic matrix factorization algorithms.
5. To be able to design algorithms for factorizing and solving banded systems.
6. To be able to use sparse storage schemes and perform matrix-vector multiplication based on related data structures. To know about SparseSuite and and how to recover matrices from it.
7. To know the Arnoldi algorithm and how it can be used to solve linear systems.

General Competences

Taking into consideration the general competences that the degree-holder must acquire (as these appear in the Diploma Supplement and appear below), at which of the following does the course aim?

Search for, analysis and synthesis of data and information, with the use of the necessary technology	*Project planning and management*
Adapting to new situations	*Respect for difference and multiculturalism*
Decision-making	*Respect for the natural environment*
Working independently	*Showing social, professional and ethical responsibility and sensitivity to gender issues*
Team work	*Criticism and self-criticism*
Working in an international environment	*Production of free, creative and inductive thinking*
Working in an interdisciplinary environment	*……*
Production of new research ideas	*Others…*
	…….

Search for, analysis and synthesis of data and information, with the use of the necessary technology
Adapting to new situations
Decision-making
Working independently
Production of free, creative and inductive thinking

(3) SYLLABUS

The importance of scientific computing in simulation and data driven science and engineering. The goal of scientific computing as an itinerary across models (mathematical, numerical, discrete, arithmetic, computational). Loss of information effects. Code profiling, performance evaluation criteria and good practices. Benchmarking and benchmarks. The central role of matrix computations in HPC. Computational models and the communication-computation tradeoffs in algorithm design for simple hierarchical memory systems. Computational kernels and the BLAS hierarchy. Basic principles of code design and compilation techniques for HPC. Matrix multiplication, from BLAS-1 to BLAS-3 and superfast methods. Arithmetic model: Review of floating-point arithmetic, backward error analysis, problem conditioning. Compensated summation methods. Block LU and BLAS-3 implementations of LU and QR. From LAPACK to MATLAB. Iterative refinement. Sparse and structured sparse matrix computations. Introduction to projection methods for numerical linear algebra and the Arnoldi algorithm. Discrete model: Simulations via the numerical solution of differential equations: The 2 point boundary value problem and initial value problems. Ranking in large networks and the example of PageRank.

(4) TEACHING and LEARNING METHODS - EVALUATION

DELIVERY *Face-to-face, Distance learning, etc.*	Ex cathedra.
USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY *Use of ICT in teaching, laboratory education, communication with students*	• **The course makes heavy use of the facilities offered by the e-Class environment. All course notes and transparencies are placed online, as well as problem sets, and pointers to the relevant literature.** • **Many of the lectures are also available in video form (in Greek) via OpenCourses.**
TEACHING METHODS *The manner and methods of teaching are described in detail.* *Lectures, seminars, laboratory practice, fieldwork, study and analysis of bibliography, tutorials, placements, clinical practice, art workshop, interactive teaching, educational visits, project, essay writing, artistic creativity, etc.* *The student's study hours for each learning activity are given as well as the hours of non-directed study according to the principles of the ECTS*	(see table below)

Activity	***Semester workload***
Lectures	3*13=39
Tutorials (exercises)	1*13=13
Laboratory exercises	1*13=13
Individual study , preparation and problem solving.	3*13=39
Weekend study	2*13=26
Study during the 3 “empty weeks” (2 weeks of vacation and 1 week of exam preparation).	5*3=15
Total (25-30 hours per ECTS unit).	***145***

STUDENT PERFORMANCE EVALUATION *Description of the evaluation procedure* *Language of evaluation, methods of evaluation, summative or conclusive, multiple choice questionnaires, short-answer questions, open-ended questions, problem solving, written work, essay/report, oral examination, public presentation, laboratory work, clinical examination of patient, art interpretation, other* *Specifically-defined evaluation criteria are given, and if and where they are accessible to students.*	• The language of instruction and examination is Greek. Special provisions (lecture notes and examinations in English) can be made for foreign students. • During the course students are assigned a number of laboratory projects. • The final grade is based on the assessment of the lab projects (typically 30% of the grade) and the final examination (typically 70% of the grade). • The grading policy is announced during the first lecture and is posted on the course website. • The final examination is written, of graded difficulty, and can consist of multiple choice questions, questions for short and long essay answers, as well as mini-lab-type questions. • Grade scale 0-10: To pass the class one needs to obtain a grade of 5 or higher.

(5) ATTACHED BIBLIOGRAPHY

- Suggested bibliography:

1. G. Golub and C. Van Loan, "Θεωρία και Υπολογισμοί με Μητρώα", εκδ. Πεδίο (2016). Μετάφραση του "Matrix Computations", 4th edition, The Johns Hopkins University Press.
2. C.W. Uberhuber, “Numerical Computation 1: Methods, Software and Analysis”, Springer, 1997.
3. J.N. Kutz, “Data-Driven Modeling & Scientific Computation”, Oxford University Press, 2013.
4. Stefan Goedecker and Adolfy Hoisie, Performance Optimization of Numerically Intensive Codes, SIAM, 2001.
5. Nicholas Higham. “Accuracy and stability of numerical algorithms”, SIAM, 2002.
6. LAPACK User’s Guide (on-line).

- Relevant scientific journals:

- SIAM Journal on Scientific Computing
- IEEE Computing in Science and Engineering Magazine
- SIAM Journal on Data Science
- SIAM Journal on Matrix Analysis and Applications
- Scientific Computing World Magazine